Command Reference for BuildGates Synthesis and Cadence PKS

Product Version 5.0.13
December 2003

Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA

Contents

Preface

This preface contains the following sections:

About This Manual

This manual is a complete alphabetical collection of BuildGates synthesis commands. BuildGates synthesis can be run both in command line mode and in graphical user interface (GUI) mode.

Other Information Sources

For more information about related products, you can consult the sources listed here. The documents you have vary depending on your product licenses.

- *AmbitWare Component Reference*
- *BuildGates Synthesis User Guide*
- *CeltIC User Guide*
- *Common Timing Engine (CTE) User Guide*
- *Constraint Translation for BuildGates Synthesis and Cadence PKS*
- *Datapath for BuildGates Synthesis and Cadence PKS*
- *Delay Calculation Algorithm Guide*
- *Design for Test Using BuildGates Synthesis and Cadence PKS*
- *Distributed Processing for BuildGates Synthesis*
- *Glossary for BuildGates Synthesis and Cadence PKS*
- *GUI Guide for BuildGates Synthesis and Cadence PKS*

- *HDL Modeling for BuildGates Synthesis*
- *Low Power for BuildGates Synthesis and Cadence PKS*
- *Low Power Synthesis Tutorial*
- *Migration Guide for BuildGates Synthesis and Cadence PKS*
- *Modeling Generation for Verilog 2001 and the Verilog Datapath Extension*
- *PKS User Guide*
- *Synthesis Place-and-Route Flow Guide*
- *Verilog Datapath Extension Reference*
- *VHDL Datapath Package Reference*
- *Known Problems and Solutions in BuildGates Synthesis*
- *Know Problems and Solutions in Cadence PKS*
- *What's New in Cadence PKS*
- *What's New in BuildGates Synthesis*

BuildGates synthesis is often used with other Cadence® tools during various design flows. The following documents provide information about these tools and flows. Availability of these documents depends on the product licenses your site has purchased.

- *Cadence Timing Library Format Reference*
- *Cadence Pearl Timing Analyzer User Guide*
- *Cadence General Constraint Format Reference*

The following books are helpful references, but are not included with the CD-ROM documentation:

- IEEE 1364 Verilog HDL LRM
- TCL Reference, *Tcl and the Tk Toolkit*, John K. Ousterhout, Addison-Wesley Publishing Company

Documentation Conventions

Text Command Syntax

The list below describes the syntax conventions used for the BuildGates Synthesis text interface commands.

`literal`	Nonitalic words indicate keywords that you must enter literally. These keywords represent command or option names.
`argument`	Words in italics indicate user-defined arguments or information for which you must substitute a name or a value.
|	Vertical bars (OR-bars) separate possible choices for a single argument.
`[ ]`	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices from which you can choose one.
`{ }`	Braces are used to indicate that a choice is required from the list of arguments separated by OR-bars. You must choose one from the list. `{ argument1 | argument2 | argument3 }`
`...`	Three dots (...) indicate that you can repeat the previous argument. If the three dots are used with brackets (that is, `[argument]...`), you can specify zero or more arguments. If the three dots are used without brackets (`argument...`), you must specify at least one argument, but can specify more.
#	The pound sign precedes comments in command files.

Using Menus

GUI commands can take one of three forms.

CommandName	A command name with no dots or arrow executes immediately.
CommandName…	A command name with three dots displays a form for choosing options.

CommandName ->	A command name with a right arrow displays an additional menu with more commands. Multiple layers of menus and commands are presented in what are called command sequences, for example: *File – Import – LEF*. In this example, you go to the File menu, then the Import submenu, and, finally, the LEF command.

Using Forms

…	A menu button that contains only three dots provides browsing capability. When you select the browse button, a list of choices appears.
Ok	The *Ok* button executes the command and closes the form.
Cancel	The *Cancel* button cancels the command and closes the form.
Defaults	The *Defaults* button displays default values for options on the form.
Apply	The *Apply* button executes the command but does not close the form.

1

BuildGates Synthesis Commands

This chapter describes the commands and global variables used with BuildGates® Synthesis.

ac_shell

```
ac_shell [-64] [[-f] filename] [-cmdfile command_filename]
     [-logfile log_filename] [-cdsdocd {on | off}] [-queue] [-expire] [-version]
     [-help] [-continue] [-no_init] [-set var=value ...] [-which] [-gui]
     [-display machine:0] [-geometry intxint+int+int] [-fullscreen] [-large]
     [-limit] [-colormap colormap_file] [-unique]
```

Starts the BuildGates Synthesis tool. For more detail on starting `ac_shell`, refer to *Before You Begin* in the *BuildGates Synthesis User Guide*.

Options and Arguments

`-64`
: Starts a 64-bit `ac_shell` session instead of the default.
Default: 32 bit

`-cdsdocd {on | off}`
: Enables or disables the use of browser-based documentation for `ac_shell help`. When set to `on`, the full documentation set is available from the *Help* button in the GUI. When set to `off`, only the syntax is displayed in the command line.
Default: `on` when running in GUI mode and `off` when running in command line mode.

`-cmdfile` *`command_filename`*
: Specifies the name of the command file.
Default: `ac_shell.cmd`

`-colormap`
: Specifies the color map file.

`-continue`
: Does not exit after an error in Tcl script file.

`-display` *`machine:0`*
: Specifies the system display to use.

`-expire`
: Displays the expiration date of the license.

`-f` *`filename`*
: Displays name of Tcl script file to source at startup. You can use *`filename`* without specifying the `-f`.

`-fullscreen`

Allows using the entire screen for the `ac_shell`.

`-geometry` *`intxint+int+int`*

Sets the initial size and position of the GUI main screen window. Where *`width`* and *`height`* are in pixels. And *`xoff`* and *`yoff`* are the number of pixels from the corner; a negative value is measured from the bottom or right corners, and a positive value is measured from the top or left corners. No spaces are allowed between the values.

For example: `-geometry 800x400+10-30` means create a window 800 by 400 pixels with its left edge offset 10 pixels from the left edge of the screen and its bottom edge offset 30 pixels from the bottom of the screen

Valid only in GUI mode.

`-gui`

Invokes graphical user interface (GUI) mode

`-help`

Prints usage message for this command.

`-large`

Increases the memory limit above 2GB for the process running `ac_shell` if the memory is available.

`-limit`

Prints the current datasize limit and memory allocation limit for the machine on which the software will run.

`-logfile` *`log_filename`*

Displays the name of the log file.
Default: `ac_shell.log`

`-no_init`

Disables the sourcing of `~/.ambit/.acshrc`

`-queue`

Waits for a license if none is available.

`-set var=`*`value`*
: Initializes a Tcl variable.

`-unique`
: Instructs the tool to create unique names for all command and log files.

`-version`
: Displays the version of this `ac_shell`.

`-which`
: Displays the full path name of the `ac_shell` executable.

Example

The following command starts the BuildGates Synthesis graphical user interface and saves the session log in `cpu_design.log`:

```
> ac_shell -gui -logfile cpu_design.log
```

Related Information

`check option`

add_netconn

`add_netconn` *`net_id pin_id_list`*

Adds new net connections to the given `net_id` in the current module. Place connections in the same module as the specified net.

This command does not delete the connection between the pin and the net if one already exists. It will create a generic buffer between the existing net and the new net.

To delete a net connection, use `delete netconn`.

Use caution when using the add_netconn command. This command changes the connectivity of the netlist, which can change the functionality of the netlist. You may create timing loops, multiport nets, and short circuits, and connect multiple outputs to the same net. Use the `check timing` ***and*** `check netlist` ***commands to detect possible problems.***

Options and Arguments

`net_`*`id`*

Specifies the id of the net where you want to add connections.

`pin_id_list`

Specifies the id list of the pin you want to add to the net.

Example

The following command adds `CLK` pin connections of all the instances with the name `out1_reg_*` to `net1`:

```
> add_netconn [find -net net1] [find -pins out1_reg*/CLK]
```

Related Information

`create instance`

`create_net`

`create port`

alias

`alias` *`command_name name`*

Creates a simple alias for command names. It does not support aliasing a command with options to a new name. It only performs a simple renaming of a previously defined command.

A more general mechanism for renaming or repackaging commands is the built-in Tcl `proc` command.

Options and Arguments

`command_name`
Displays the alias name.

`name`
Displays the name of an existing command.

Database Impact

There is no effect on the database

Example

This command creates the `wv` alias for the `write_verilog` command:

```
> alias wv write_verilog
```

Related Information

`unalias`

all_children

```
all_children [-instances] [-hierarchical] [module_id | module_name]
```

Returns a Tcl list of object identifiers representing the child modules that are instantiated in the specified parent module. Use this routine recursively to traverse through the netlist in a top-down fashion, as shown in the example below.

Note: The list returned by the `all_children` command excludes black box children unless you use the `-instances` option.

Options and Arguments

`-instances`
: Returns a Tcl list of object identifiers representing the `instance_ids` of the instantiated child modules in the specified parent module, including the black box children.

`-hierarchical`
: Returns the `module_ids` (or the `instance_ids` if the `-instances` option is specified) of all the modules instantiated in the specified module and its submodules.

`module_id | module_name`
: Specifies the object identifier or name associated with a module. If neither is specified, the command returns the children of the current module.

Examples

- The following command gets the names of children of module a:

```
> get_names [all_children [find -module a]]
b c
```

- The following command writes a netlist for all children of module a:

```
foreach i [all_children [find -module a]] {
      set_current_module $i
      write_ver $i.ver
      }
```

- The following command traverses the hierarchy using attributes:

```
proc traverse_hier {mod} {
      set_current_module $mod
```

```
foreach i [all_children] {
     traverse_hier $i
     }
if {[get_attribute [find -module $mod] MY_ATTR] == ""} {
     set_attribute [find -module $mod] MY_ATTR "visited"
     # do action
     }
```

Related Information

all_parents

find

all_parents

```
all_parents [module_id | module_name]
```

Returns a Tcl list of object identifiers of modules that instantiate the specified module, similar to the `all_children` command. This routine is used to do a bottom-up traversal of the netlist hierarchy.

Options and Arguments

`module_id | module_name`
: Specifies the object identifier or name associated with a module. If neither is specified, the command returns the parents of the current module.

Related Information

`all_children`

`find`

bg_shell

```
bg_shell [-64] [[-f] filename] [-cmdfile command_filename]
     [-logfile log_filename] [-cdsdocd {on | off}] [-queue] [-expire] [-version]
     [-help] [-continue] [-no_init] [-set var=value ...] [-which] [-gui]
     [-display machine:0] [-geometry intxint+int+int] [-fullscreen] [-large]
     [-limit] [-colormap colormap_file]
```

Starts the BuildGates Synthesis tool. For more detail on starting `bg_shell`, refer to Before You Begin in the *BuildGates Synthesis User Guide*.

Options and Arguments

`-64`
: Starts a 64-bit `ac_shell` session instead of the default.
Default: 32 bit

`-cdsdocd {on | off}`
: Enables or disables the use of browser-based documentation for `bg_shell help`. When set to `on`, the full documentation set is available from the *Help* button in the GUI. When set to `off`, only the syntax is displayed in the command line.
Default: `on` when running in GUI mode and `off` when running in command line mode.

`-cmdfile` *`command_filename`*
: Specifies the name of the command file.
Default: `bg_shell.cmd`

`-colormap`
: Specifies the color map file.

`-continue`
: Does not exit after an error in Tcl script file.

`-display` *`machine:0`*
: Specifies the system display to use.

`-expire`
: Displays the expiration date of the license.

`-f` *`filename`*
: Displays the name of the Tcl script file to source at startup. You can use *`filename`* without specifying the `-f`.

`-fullscreen`

Allows using the entire screen for the `bg_shell`.

`-geometry` *`intxint+int+int`*

Sets the initial size and position of the GUI main screen window. Where *`width`* and *`height`* are in pixels. And *`xoff`* and *`yoff`* are the number of pixels from the corner; a negative value is measured from the bottom or right corners, and a positive value is measured from the top or left corners. No spaces are allowed between the values.

For example: `-geometry 800x400+10-30` means create a window 800 by 400 pixels with its left edge offset 10 pixels from the left edge of the screen and its bottom edge offset 30 pixels from the bottom of the screen

Valid only in GUI mode.

`-gui`

Invokes the graphical user interface (GUI) mode.

`-help`

Prints the usage message for this command.

`-large`

Increases the memory limit above 2GB for the process running `bg_shell` if the memory is available.

`-limit`

Prints the current datasize limit and memory allocation limit for the machine on which the software will run.

`-logfile` *`log_filename`*

Displays the name of the log file.
Default: `bg_shell.log`

`-no_init`

Disables the sourcing of `~/.ambit/.acshrc`

`-queue`

Waits for a license if none is available.

`-set var=`*`value`*
: Initializes a Tcl variable.

`-version`
: Displays the version of this `bg_shell`.

`-which`
: Displays the full path name of the `bg_shell` executable.

Example

The following command starts the BuildGates Synthesis graphical user interface and saves the session log in `cpu_design.log`:

```
> bg_shell -gui -logfile cpu_design.log
```

Related Information

`check option`

bgx_shell

```
bgx_shell [-64] [[-f] filename] [-cmdfile command_filename]
     [-logfile log_filename] [-cdsdocd {on | off}] [-queue] [-expire] [-version]
     [-help] [-continue] [-no_init] [-set var=value ...] [-which] [-gui]
     [-display machine:0] [-geometry intxint+int+int] [-fullscreen] [-large]
     [-limit] [-colormap colormap_file]
```

Starts the BuildGates Extreme Synthesis tool. For more detail on starting `bgx_shell`, refer to Before You Begin in the *BuildGates Synthesis User Guide*.

Options and Arguments

`-64`
: Starts a 64-bit `bgx_shell` session instead of the default.
Default: 32 bit

`-cdsdocd {on | off}`
: Enables or disables the use of browser-based documentation for `bgx_shell help`. When set to `on`, the full documentation set is available from the *Help* button in the GUI. When set to `off`, only the syntax is displayed in the command line.
Default: `on` when running in GUI mode and `off` when running in command line mode.

`-cmdfile` *`command_filename`*
: Specifies the name of the command file.
Default: `bgx_shell.cmd`

`-colormap`
: Specifies the color map file.

`-continue`
: Does not exit after an error in Tcl script file.

`-display` *`machine:0`*
: Specifies the system display to use.

`-expire`
: Displays the expiration date of the license.

`-f` *`filename`*
: Displays the name of the Tcl script file to source at startup. You can use *`filename`* without specifying the `-f`.

`-fullscreen`

Allows using the entire screen for the `bgx_shell`.

`-geometry` *`intxint+int+int`*

Sets the initial size and position of the GUI main screen window. Where *`width`* and *`height`* are in pixels. And *`xoff`* and *`yoff`* are the number of pixels from the corner; a negative value is measured from the bottom or right corners, and a positive value is measured from the top or left corners. No spaces are allowed between the values.

For example: `-geometry 800x400+10-30` means create a window 800 by 400 pixels with its left edge offset 10 pixels from the left edge of the screen and its bottom edge offset 30 pixels from the bottom of the screen

Valid only in GUI mode.

`-gui`

Invokes the graphical user interface (GUI) mode

`-help`

Prints the usage message for this command.

`-large`

Increases the memory limit above 2GB for the process running `bgx_shell` if the memory is available.

`-limit`

Prints the current datasize limit and memory allocation limit for the machine on which the software will run.

`-logfile` *`log_filename`*

Displays the name of log file.
Default: `bgx_shell.log`

`-no_init`

Disables the sourcing of `~/.ambit/.acshrc`

`-queue`

Waits for a license if none is available.

`-set var=`*`value`*
: Initializes a Tcl variable.

`-version`
: Displays the version of this `bgx_shell`.

`-which`
: Displays the full path name of the `bgx_shell` executable.

Example

The following command starts the BuildGates Extreme Synthesis graphical user interface and saves the session log in `cpu_design.log`:

```
> bgx_shell -gui -logfile cpu_design.log
```

Related Information

`check_option`

check_netlist

```
check_netlist [-module module] [-blocklevel] [-verbose] [-ignore_loops]
     [file_name]
```

Performs a number of checks on the structural connectivity of the netlist, including recursively defined modules, combinational feedback, undriven nets and pins, multiple driven nets and pins, and undriven ports.

This command can be applied to generic and mapped modules.

Note: The following applies to warning messages which may be printed out when using this command:

- A **multiple** driven net is a net that has multiple drivers. For example, two AND gates drive the same net which causes signal contention.
- A **potential multiple** driven net is a net that is driven by more than one bidirectional pin or port. It is allowable at any time for there to be only one driver while the rest of the bidirectional pins or ports are sinks. Otherwise, this causes signal contention, as in multiple nets above. Bidirectional pins can be created when an instance is a black box, therefore BuildGates Synthesis cannot determine the true direction of pins.

Options and Arguments

`-blocklevel`
: Checks the current module only, not the hierarchy.

`file_name`
: Displays optional file name for the output.

`-ignore_loops`
: Does not report combinational loops.

`-module` *`module`*
: Checks a specific module. By default, `check_netlist` checks the current module.

`-verbose`
: Prints detailed information for each violation check and also the summary. Also reports floating inputs or nets in RTL and gate level netlists.

Example

The following shows sample output for the `check_netlist` command:

```
> check_netlist
The number of recursively defined modules (at least):0
The number of combinational feedbacks(at least):0
The total number of undriven nets/pins:0
The total number of multiple driven nets:0
The total number of potential multiple driven nets:1
The total number of undriven ports:0
```

check_option

```
check_option [-hide | -show {-datapath | -power | -64 | -large | -PKS | -no_PKS |
     -gui | -visual}]
     [[-]datapath] [[-]power] [[-]64] [[-]large] [[-]PKS] [[-]no_PKS] [[-]gui]
     [[-]visual]
```

Determines which command line options the current `ac_shell` executable has been invoked. It returns all enabled options when the command is invoked without any arguments. Options which enable a feature are printed in capitalized letters. When invoking the `ac_shell`, all options invoking features can be entered in any upper and / or lower case combination.

The invokable options (`datapath, power, 64, large, PKS, no_PKS, gui,` and `visual`) may be specified with or without the preceding dash to return different types of values. The option without a dash will return a 1 or 0 to indicate `ac_shell` is running with or without the option enabled, respectively. The option with a dash will return the name of the enabled option or the empty set (“ “) if the option is not enabled. See examples below for clarification.

Options and Arguments

`-64`

If the current session was executed with `ac_shell -64`, the `check_option` command:
Returns 1 If the `64` option is specified.
Returns the word `-64` if the `-64` option is specified.

`-datapath`

If the current session was executed with `ac_shell -datapath` (the Datapath feature), the `check_option` command:
Returns 1 If the `datapath` option is specified.
Returns the word `-Datapath` if the `-datapath` option is specified.

`-gui`

If the current session was executed with `ac_shell -gui` (the graphical user interface), the `check_option` command:
Returns 1 If the `gui` option is specified.
Returns the word `-gui` if the `-gui` option is specified.

`-hide`

Temporarily disables an option that was invoked when `ac_shell` was started.

`-large`

If the current session was executed with `ac_shell -large`, the `check_option` command:
Returns 1 If the `large` option is specified.
Returns the word `-large` if the `-large` option is specified.

`-no_PKS`

If the current session was executed with `ac_shell -no_pks` (or without the -`pks` option), the `check_option` command:
Returns 1 If the no_`pks` option is specified.
Returns the word `-no_PKS` if the `-no_pks` option is specified.

`-PKS`

If the current session was executed with `ac_shell -pks` (the PKS license), the `check_option` command:
Returns 1 If the `pks` option is specified.
Returns the word `-PKS` if the `-pks` option is specified.

`-power`

If the current session was executed with `ac_shell -power` (the Low Power feature), the `check_option` command:
Returns 1 If the `power` option is specified.
Returns the word `-Power` if the `-power` option is specified.

`-show`

Re-enables an option that was previously disabled with the `-hide` option.

`visual`

This option provides the same functionality as the `-gui` option and is used only for backward compatibility with previous releases.
If the current session was executed with `ac_shell -visual` (the graphical user interface), the `check_option` command:
Returns 1 If the `visual` option is specified.
Returns the word `-visual` if the `-visual` option is specified.

Examples

- The following example shows the output of the `check_option` command for a session invoked with the PKS and GUI options:

```
> check_option [datapath | large | pks | power | gui]
0 0 1 0 1
```

- The following example shows the output for a session invoked with the PKS option:

```
> check_option -power -pks -large
-PKS
```

- The following example shows the commands and output for an `ac_shell` session started with the `-gui` and the `-large` options. This example demonstrates the disabling (`-hide`) and re-enabling (`-show`) of the `-large` option:

```
> check_option -gui -large
-gui -large

> check_option -hide -large
> check_option -gui -large
-gui

> check_option -show -large
> check_option -gui -large
-gui -large
```

create_instance

`create_instance [mod_name_or_id | cell_name_or_id]` *`instance_name`*

Creates a new instance in the current module that refers to the specified module or cell, and returns the ID number of the new object.

Use caution when using the* `create_instance` *command. Do not use this command without a very good reason: in most cases, there is no check for functionality.

Options and Arguments

`instance_name`
Specifies the name of the instance you want to create.

`mod_name_or_id | cell_name_or_id`
Specifies either the name or the id of the module or the technology cell for the instantiation created.

Example

The following command creates an instance named `i1` that refers to the module named `mod1` in the current module:

```
> create_instance mod1 i1
```

Related Information

`add_netconn`

`create_module`

`create_net`

`create_port`

`find`

create_module

`create_module` *`module_name`*

Creates an empty module with the given name, and returns the ID number of the new object. The created module is linked to all the black boxes in the netlist with the same name. This command also creates the corresponding ports.

Use caution when using the `create_module` ***command. Do not use this command without a very good reason: in most cases, there is no check for functionality.***

Options and Arguments

`module_name`

Specifies the name of the module you want to create.

Example

The following command creates a module named `mod1`:

```
> create_module mod1
```

Related Information

`do_copy_module`

`do_rebind`

create_net

`create_net -vector -begin` *`begin`* `-end` *`end net_name`*

Creates a scalar or vector net in the current view with the given name, and returns the ID number of the new object.

Use caution when using the* `create_net` *command. Do not use this command without a very good reason: in most cases, there is no check for functionality.

Options and Arguments

`-begin` *`begin`*

Specifies the begin bit number of the vector net, which is ignored if the `-vector` option is not specified.
Default: `0`

`-end` *`end`*

Specifies the end bit number of the vector net, which is ignored if the `-vector` option is not specified.

`net_name`

Specifies the name of the net you want to create.

`-vector`

Specifies that the net is a vector.
Default: `scalar`

Examples

- The following command creates a scalar net named `aNet`:

```
> create_net aNet
```

- The following command creates a vector net named `bNet[0:7]`:

```
> create_net -vector -begin 0 -end 7 bNet
```

Related Information

`add_netconn`

create_instance

create_port

create_port

```
create_port -direction { input | output | bidir } -vector -begin begin -end end
     port_name
```

Creates a port with the given name and direction for the current module, and returns the ID number of the new object. This automatically updates all the instances referring to the current module.

Use caution when using the create_port command because it can cause simulation failures. In many cases, there is not a check for functionality. Always verify the generated names.

Options and Arguments

`-begin` *`begin`*

Specifies the begin bit number of the vector net, which is ignored if the `-vector` option is not specified.
Default: `0`

`-direction {input | output | bidir}`

Specifies the direction of the port you want to create.
Default: `bidir`

`-end` *`end`*

Specifies the end bit number of the vector net, which is ignored if the `-vector` option is not specified.
Default: `0`

`port_name`

Specifies the name of the port you want to create.

`-vector`

Specifies that the port is a vector.
Default: `scalar`

Related Information

`add_netconn`

`create_instance`

delete_attribute

`delete_attribute` *`object_id attribute_name`*

Removes the specified `attribute_name` from the specified `object_id`.

Options and Arguments

`attribute_name`

Displays the name of the attribute to delete. Refer to the `set_attribute` command's set_attribute on page 257 for a list of tool-defined attributes.

`object_id`

Displays the object identifier of the module, instance, cellref, net, port, or pin from which to delete the specified *`attribute_name`*.

Example

The following command clears the attribute `net_node` in the object `$net`:

```
> delete_attribute $net net_node
```

Related Information

`get_attribute`

`set_attribute`

delete_aware_component

`delete_aware_component [-library` *`library_name`*`] { -all | `*`component_name`* `}`

Deletes the specified aware component or all aware components from the specified library.

Options and Arguments

`-all`	Specifies that all components in the specified library are deleted. If you use this option and no library is specified, all the components in all aware libraries are deleted.
`component_name`	Displays the name of the aware component.
`-library` *`libname`*	Selects the aware library specified by *`lib name`*. If no library or no *`component name`* is specified, the first component found in all aware libraries is deleted.

Example

The following command deletes the component `AWMYCOMP1` from the library `AWMYLIB`:

```
> delete_aware_component -library AWMYLIB AWMYCOMP1
```

Related Information

`report_aware_library`

`set_aware_component_property`

`set_aware_library`

delete_netconn

`delete_netconn` *`list_of_pin_ids`*

Deletes any net connections created with the `add_netconn` command.

Options and Arguments

`list_of_pin_ids`
Specifies the pin ids you want to delete from the net.

Related Information

`add_netconn`

`delete_object`

delete_object

`delete_object {`*`list_of_object_ids`*`}`

Deletes any design object. The object type is known to the object, therefore it is unnecessary to specify the type of object.

Be careful when deleting individual objects. To remove all the design objects from the database use the `do_remove_design` command.

Removes the specified design object from the database.

Options and Arguments

`{list_of_object_ids}`
Provides the list of object identifiers of the objects to be removed.

Related Information

`do_remove_design`

Example

The following command creates a new module, `new_mod`, by copying the contents of `mod_a` into `new_mod`. The object ID of the new module is returned and used to delete the newly created module:

```
> set id [do_copy_module mod_a new_mod]
28993

> delete_object $id
```

delete_unconnected_ports

```
delete_unconnected_ports [-preserve_busses] [-verbose]
```

Deletes the unconnected input and output ports in the netlist.

Options and Arguments

`-preserve_busses`

Recognizes the ports belonging to the busses not deleted so these port busses are preserved.

`-verbose`

Prints detailed information for each port detailed from the netlist.

do_blast_busses

```
do_blast_busses [-nets] [-ports] [-complex_ports] [-current_module]
```

Removes bus objects in the hierarchy or in the current module. When converted to scalars, the `hdl_array_generator` and `hdl_record_generator` globals are used to name the scalars for array and record busses. The `buscomp_generator` global is used to name the scalars of all other busses.

To remove all the bus nets from the top level and the lower levels of the design use the following command:

```
do_blast_busses -nets
```

Converts pin, port, and net bus objects in the current module or in the hierarchy to scalars. If instances of the module exist they are modified to connect each bit of the previously bussed port separately.

Note: The term blast (see argument descriptions) refers to the process of converting pins, ports, and net bus objects in the current module.

Options and Arguments

`-complex_ports`

Blasts complex bus ports, such as array ports and record ports, using the scalar names specified by the `hdl_array_generator` and `hdl_record_generator` global commands, respectively.

TIP: This command argument is useful for generating netlist ports that are easily recognized by formal verification tools.

`-current_module`

Blasts bus objects only in the current module.

`-nets`

Blasts bus nets as well as bus pins on the instance.

`-ports`

Blasts bus ports and complex ports as well as bus pins on the instance.

Examples

The following example shows two modules, a and b. "Before do_blast_busses:" contains bus objects before the `do_blast_busses` command is run. In "After do_blast_busses:" all the bus objects have been removed by the `do_blast_busses` command and converted to scalars:

- Before do_blast_busses:

```
module a(out);
      output [1:0] out;
      b b1(out);
endmodule
module b(out);
      output [1:0] out;
endmodule
```

- After do_blast_busses:

```
module a(\out[1], \out[0]);
      output \out[1] ;
      output \out[0] ;
      b b1(.\out[1] (\out[1] ), .\out[0] (\out[0] ));
endmodule
module b(\out[1] , \out[0] );
      output \out[1] ;
      output \out[0] ;
endmodule
```

- Using the hdl_array_generator and hdl_record generator with do_blast_busses:

```
package p is
  type arr is array (1 downto 0) of bit_vector(2 downto 0);
  type rec is
    record
      field1 : integer range 0 to 3;
      field2 : arr;
    end record;
end;
use work.p.all;
entity e is
    port(p_rec : in rec;
         q_rec : out rec);
end;
architecture a of e is
```

```
begin
  q_rec <= p_rec;
end;
set_global hdl_array_generator %s\[%d\]
set_global hdl_record_generator %s\[%s\]
do_build_generic
do_blast_busses
entity e is
   port (
_rec[field1][1]\: in std_logic;
_rec[field1][0]\: in std_logic;
_rec[field2][1][2]\: in std_logic;
_rec[field2][1][1]\: in std_logic;
_rec[field2][1][0]\: in std_logic;
_rec[field2][0][2]\: in std_logic;
_rec[field2][0][1]\: in std_logic;
_rec[field2][0][0]\: in std_logic;
      \q_rec[field1][1]\: out std_logic;
      \q_rec[field1][0]\: out std_logic;
      \q_rec[field2][1][2]\: out std_logic;
      \q_rec[field2][1][1]\: out std_logic;
      \q_rec[field2][1][0]\: out std_logic;
      \q_rec[field2][0][2]\: out std_logic;
      \q_rec[field2][0][1]\: out std_logic;
      \q_rec[field2][0][0]\: out std_logic
   );
end entity e;
```

Related Information

do_remove_design

do_rename

set_global buscomp_generator

set_global hdl_array_generator

set_global hdl_record_generator

do_build_generic

```
do_build_generic [-all] [-module name] [-extract_fsm] [-group_all_processes]
     [-group_named_processes] [-group_process list_of_processes]
     [-group_all_subprograms] [-group_subprograms list_of_subprograms]
     [-parameters | -generics tcl_list] [-sleep_mode]
```

Transforms the design read in by the commands `read_vhdl` and `read_verilog` into a hierarchical, gate-level netlist consisting of technology-independent logic gates, using components from the Ambit Synthesis Technology Library (ATL) and Extended ATL (XATL). The command performs constant propagation, loop unrolling, lifetime analysis, register inferencing, and logic mapping.

You must run `do_build_generic` after specifying the source Verilog or VHDL files for the initial design database and before calling `do_optimize`. You must run `do_build_generic` on a netlist even if it is already mapped to the target library. After running `do_build_generic`, any instance of a target library cell in the source description will remain mapped to that cell in the design database.

This command must be executed before any optimization commands can be applied. The generated netlist can then be written as a Verilog netlist (using the `write_verilog` command), a VHDL netlist (`write_vhdl`), and an AMBIT database (`write_adb`). These netlists can be loaded later for optimization and analysis using the `read_verilog`, `read_vhdl`, and `read_adb` commands, respectively.

The options associated with this command allow for customization and control of logical partitions grouped by various processes. When a `-group` switch is specified, the `do_build_generic` command creates a separate level of hierarchy for the logic generated for each of the specified RTL procedural constructs (process, named process, process listed in `list_of_processes`, subprogram, or subprogram listed in `list_of_subprograms`, respectively) in the design. For VHDL designs, a process is a VHDL process statement, and a subprogram is a VHDL function or procedure. For Verilog designs, a process is a Verilog initial or always block, and a subprogram is a Verilog task or function.

Note: Grouping processes or subprograms may result in sub-optimal designs because the logic optimizations performed during the `do_optimize` command are applied to each hierarchy separately and are not applied across hierarchy boundaries.

The `do_build_generic` options also allow the user to generate netlists for selected modules in the design hierarchy. The netlist created by this command uses technology-independent logic gates using components from the ATL.
Default: This command treats all procedural blocks (initial and always blocks in Verilog and processes in VHDL) as part of the module in which they appear without any hierarchy. When the grouping is done, a new level of hierarchy is created that only contains the logic represented by the selected blocks.

Tip

Use the `find` command to find all instances of a particular module. For example, the following command returns the names of all instance in module `level2_mod`.

```
find -of_cell_type level2_mod
```

Options and Arguments

`-all`

Builds a generic netlist for all modules in the design hierarchy. If there are multiple top-level modules in the design hierarchy, the `-all` or `-module` option must be specified. If `-all` is specified, the first module returned by `[find -top *]` is selected as the top of the design hierarchy and as the default top timing module.

`-extract_fsm`

Extracts `fsm` for registers marked with the state vector directive. Generates a finite state machine (FSM) to implement each variable identified by the state_vector synthesis directive. See "*Optimizing and Structuring Finite State Machines*" in the HDL Modeling for BuildGates Synthesis user guide for additional information.

`-generic {{ <generic1> <value1> } ... { <genericN> <valueN> }}`

Builds a netlist using the generic values specified in the Tcl list. The generic values must be integers. The term "generic" conforms to VHDL nomenclature. In Verilog the term "parameter" is used instead of the VHDL "generic". Generics or parameters can be selectively modified. Therefore, those that are not modified with the `-generic` option will retain their default value (the value specified in the module).

`-group_all_processes`

Creates a new level of hierarchy by grouping the logic from all the processes.

`-group_all_subprograms`

Creates a new level of hierarchy by grouping the logic from all of the subprograms.

`-group_named_processes`
: Creates a new level of hierarchy by grouping the logic from all the named processes.

`-group_process` *`list_of_processes`*
: Creates a new level of hierarchy by grouping the logic from all the named processes specified by the *`list_of_processes`*.

`-group_subprograms` *`list_of_subprograms`*
: Creates a new level of hierarchy by grouping the logic from the subprograms (`functions` and `tasks` in Verilog).

`-module` *`name`*
: Builds a generic netlist for the named module and all sub-modules in the hierarchy. Selects the named module as the top of the design hierarchy and as the default top timing module. The commands `get_hdl_hierarchy` and `get_hdl_top_level` can be used prior to `do_build_generic` to identify the modules in the database.

`-parameters` *`{{ <generic1> <value1> } ... { <genericN> <valueN> }}`*
: Provides a list of parameter names and values to use for the indicated generics. Each item of the Tcl list is a name-value pair. For example: {{WIDTH 4} {HEIGHT 7}}.

`-sleep_mode`
: Explores areas of the design where power could be saved by using the sleep mode technique and inserts sleep-mode logic for commitment during gate level power optimization.

Examples

- The following sequence of commands reads a Verilog file, `design.v`, performs high-level optimizations and resource allocation, and then reports the hierarchy.

```
> read_verilog design.v
> do_build_generic
> report_hierarchy
```

- The following commands build a generic netlist for all modules:

```
> do_build_generic
```

or:

```
> do_build_generic -all
```

or:

```
> do_build_generic -module des_top
```

- Two parameters are specified in the following module:

```
module m(q, d1, d2);
  parameter w1 = 4, w2 = 8;
  output [w1+w2-1:0] q;
  input [w1-1:0] d1;
  input [w2-1:0] d2;

  assign q = d1 * d2;
endmodule
```

  The following command specifies a new value for the `w2` parameter:

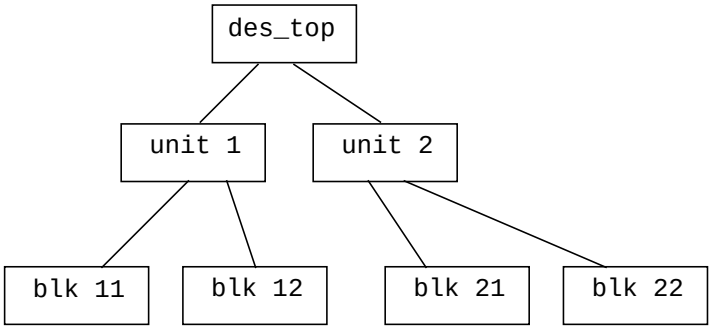
```
> do_build_generic -module m -generic { {w2 4} }
```

  The `w1` parameter in this case will retain its original value of `4`.

- The following command builds a generic netlist for `unit1`, `blk11`, and `blk12, as` shown in the Figure 1-1 on page 68:

```
> do_build_generic -module unit1
```

Figure 1-1 Building a Generic Netlist



- The following command elaborates the design for a module named `FOO` which has three parameters: `WIDTH`, `HEIGHT`, and `LENGTH` by providing specific values for the parameters:

```
> do_build_generic -design FOO parameters { {WIDTH 4} {HEIGHT 2} }
```

  If parameter `LENGTH` is not specified in the command line, the default value is used.

Related Information

do_dissolve_hierarchy

do_remove_design

get_hdl_hierarchy

get_hdl_top_level

read_symbol

read_verilog

read_vhdl

write_assertions

write_verilog

write_vhdl

do_change_module_architecture

```
do_change_module_architecture [-hierarchical]
     [-adder_architecture {rpl | ripple | fcla | cla | csel | csum}]
     [-multiplexer_architecture {encoded | decoded}]
     [-multiplier_architecture {booth | non_booth}] {list_of_modules}
```

Changes the architecture of the adder, multiplexer, and multiplier elements in the relevant Ambitware module. The Ambitware module is resynthesized with the specified architecture values and the old module is replaced by the new one. If the `-hierarchical` option is specified, the command applies to all the Ambitware sub-modules found in the specified module. If the list of modules is specified, the command is applied to all the modules successively.

Either the `-adder_architecture` or `-multiplier_architecture` option must be specified. Specifying both options is allowed.

This command applies only to modules built using Datapath synthesis.

License Requirement: BuildGates Extreme Synthesis, or BuildGates Physically Knowledgeable Synthesis (PKS) Synthesis

Options and Arguments

`-adder_architecture`
: Specifies the final adder architecture. The valid values are `rpl`, `ripple`, `fcla`, `cla`, `csel`, and `csum`.

`-hierarchical`
: Applies the command recursively to all the relevant Ambitware sub-modules in the specified module.

`-multiplexer_architecture`
: Specifies the multiplexer architecture. The valid values are `decoded` and `encoded`.

`-multiplier_architecture`
: Specifies the multiplier encoding scheme. The valid values are `booth` and `non_booth`.

Examples

- The following command changes the architecture of adder `AWDP_ADD_0` to `cla`:

```
> do_change_module -adder_architecture cla [find -module AWDP_ADD_0]
```

- The following command changes the architecture of all adder elements in the module *filter* to `cla`. The `-hierarchical` option is required because *filter* is not a datapath module:

  ```
  > do_change_module -hierarchical -adder_architecture cla [find -module filter]
  ```

- The following command changes the architecture of all adder elements in the module *filter* and its sub-modules to `cla`:

  ```
  > do_change_module -hierarchical -adder_architecture cla [find -module filter]
  ```

- The following command changes the architecture of all adder elements in the module *filter1*, *filter2*, and their sub-modules to `cla`:

  ```
  > do_change_module -hierarchical -adder_architecture cla [find -module filter1
  filter2]
  ```

do_change_name

`do_change_name {`*`object_id new_name`*`} | {-use_rules [-verbose] [-log `*`filename`*`]}`

Renames objects within the design database. The execution of this command will change the name of one object only. An error will result if you attempt to change to a name that already exists in the corresponding name space.

Upon execution of the command, an attribute is created on the renamed object whose value is the name being replaced. Use `get_info` or `get_attribute` to retrieve the name. The name of the attribute created reflects the type of object being renamed.

Note: The `do_change_name` command handles logical netlist changes only. It does not make any updates to the physical aspects of the design.

There are two modes of operation, single-object name change and rule-based name change:

- Single object:

  `do_change_name `*`object_id new_name`*

- Rules based:

  `do_change_name -use_rules [-`*`verbose`*`] [-`*`log file`*`]`

  - Uses conversion rules specified by the `set_global dcn_*` variables.
  - Process all objects hierarchically from current module down.

  Default: no conversion

The following is the order for rule processing:

1. Remove reserved word:

   Prefix the reserved words with `AMBIT_` so they don't appear in netlist.

2. Replace first restricted character:

   If the *first* character of the name is a restricted character, then it is replaced with the replacement character.

3. Replace last restricted character:

   If the *last* character of the name is a restricted character, then it is replaced with the replacement character.

4. Replace characters:
   - Pass 1: Any character in the name that is restricted is replaced with the replacement character.
   - Pass 2: Any character in the name that is *not* in the allowed list of characters is replaced by the replacement character.
5. Add prefix:

   The prefix is added if present.
6. Check length and chop back middle if required:

   The name length is checked. If it is too long, characters are removed from middle until the name is the correct length.
7. Check for name collisions and change name, if required:

   New names are checked for collisions with existing names. If there is a collision a new unique name is created.

The following are `set_global` variable conversion rules. See the Common Variables chapter of the *Global Variable Reference for BuildGates Synthesis and Cadence PKS* for additional details.

Global Variable	Comments
Net:	
`dcn_net_allow_conversion`	Set if conversion allowed *Default*: no conversion
`dcn_net_max_length`	Max name length
`dcn_net_allowed`	String of allowed characters
`dcn_net_first_restricted`	String of first restricted characters
`dcn_net_last_restricted`	String of last restricted characters
`dcn_net_map`	List of old net name and new net name
`dcn_net_remove_chars`	String of characters to remove
`dcn_net_replacement_char` -	Single replacement character
`dcn_net_reserved_words`	String of reserved words to change
`dcn_net_restricted`	String of all restricted characters
`dcn_net_prefix`	Prefix to add to name

Global Variable	Comments
Bus:	
dcn_bus_allow_conversion	Set if conversion allowed *Default*: no conversion
dcn_bus_map	List of old bus name and new bus name
dcn_bus_max_length	Max name length
dcn_bus_allowed	String of allowed characters
dcn_bus_restricted	String of all restricted characters
dcn_bus_first_restricted	String of first restricted characters
dcn_bus_last_restricted	String of last restricted characters
dcn_bus_replacement_char	Single replacement character
dcn_bus_remove_chars	String of characters to remove
dcn_bus_reserved_words	String of reserved words to change
dcn_bus_prefix	Prefix to add to name
Instance:	
dcn_inst_allow_conversion	Set if conversion allowed *Default*: no conversion
dcn_inst_max_length	Max name length
dcn_inst_allowed	String of allowed characters
dcn_inst_restricted	String of all restricted characters
dcn_inst_first_restricted	String of first restricted characters
dcn_inst_last_restricted	String of last restricted characters
dcn_inst_replacement_char	Single replacement character
dcn_inst_remove_chars	String of characters to remove
dcn_inst_reserved_words	String of reserved words to change
dcn_inst_prefix	Prefix to add to name
dcn_inst_map	List of old instance name and new instance name

Global Variable	Comments
Module:	
dcn_module_allow_conversion	Set if conversion allowed *Default*: no conversion
dcn_module_max_length	Max name length
dcn_module_allowed	String of allowed characters
dcn_module_map	List of old module name and new module name
dcn_module_restricted	String of all restricted characters
dcn_module_first_restricted	String of first restricted characters
dcn_module_last_restricted	String of last restricted characters
dcn_module_replacement_char	Single replacement character
dcn_module_remove_chars	String of characters to remove
dcn_module_reserved_words	String of reserved words to change
dcn_module_prefix	Prefix to add to name
Port:	
dcn_port_allow_conversion	Set if conversion allowed *Default*: no conversion
dcn_port_max_length	Max name length
dcn_port_allowed	String of allowed characters
dcn_port_restricted	String of all restricted characters
dcn_port_first_restricted	String of first restricted characters
dcn_port_last_restricted	String of last restricted characters
dcn_port_map	List of old port name and new port name
dcn_port_replacement_char	Single replacement character
dcn_port_remove_chars	String of characters to remove
dcn_port_reserved_words	String of reserved words to change
dcn_port_prefix	Prefix to add to name

Options and Arguments

`-log` *`filename`*

Writes the conversion progress messages to the specified file only valid for `-use_rules` option.

`new_name`

Determines the replacement name for the specified object.

`object_id`

Determines the object identifier of the specified object.

`-use_rules`

Enables rules-based renaming ability controlled by the `set_global` command's `dcn_*` variables.

`-verbose`

Prints conversion progress messages in the console (old name to new name) only valid for `-use_rules` option.

Example

The following commands change all occurrences of `oldname1` to `newname1` and so on:

```
> do_change_name [find -instance instance_name] new_name
> get_info [find -instance new_name]
> dcn_net_map [list [list oldname1 newname1] [list oldname2 newname2]]
```

Related Information

`set_global`

do_cleanup_netlist

```
do_cleanup_netlist [-hierarchical] [-mapped] [-remove_buffer_and_inverter_pair]
```

Removes all dangling or unconnected instances in the netlist without regard to timing. This command was designed for cleaning up generic netlists prior to timing optimization.

Note: The `do_cleanup_netlist` command does not remove hierarchical self-loops (loops that do not connect to any output).

Options and Arguments

`-hierarchical`
: Performs a hierarchical cleanup of the netlist.

`-mapped`
: Removes unreachable instances from a mapped netlist during the cleanup.

`-remove_buffer_and_inverter_pair`
: Removes all dangling or unconnected instances *and* all connected buffers and inverter pairs from a mapped netlist. *Default*: This is the default behavior for a generic netlist.

  Note: Because this command is not timing driven, this option will also strip out any buffer and inverter pairs added for timing or DRV fixing purposes.

Example

The following command removes floating instances (and any other instances that are not in hierarchical self-loops) in a netlist:

```
> do_cleanup_netlist -hierarchical
```

do_copy_module

`do_copy_module` *`module name`*

Creates a single new module by copying the contents of an existing module to the new module. You can only create one at a time. This command does not descend into the hierarchy of the design. That is, if the module contains instances of other hierarchical objects, those objects and their contents are not copied to the new module. It only works at the level from which it was called.

An example of this is to use the command to cache a module before making changes to the netlist. The effect of the change on the quality of the new netlist can be tested using the `get_area` or `get_module_worst_slack` commands.This command can also be used to perform a `uniquify_module` function.

Options and Arguments

`module`

Specifies the name of the source module.

`name`

Specifies the name of the new module. The name must be unique.

Examples

- The following example demonstrates the creation of a new module, `new_module1`, by copying the contents of the source module, a, to the new module:

```
> do_copy_module a new_module1
72705
```

The *`object_id`* of the new module is returned and the command can be used as follows:

```
> set_current_module [do_copy_module a new_module2]
72977
```

Optimization Loop

- The following example shows how to use `do_copy_module` in an optimization loop:

```
proc xform {} {
      set orig_slack [get_module_worst_slack]
      do_copy_module top temp_module
```

```
# do some transforms here with the goal of improving timing
      do_optimize -flatten on -effort high
      set new_slack [get_module_worst_slack]
      if { $new_slack > [expr $orig_slack + 0.001] } {

      # Commit this change if the new slack is better by > 1 ps
           delete_object [find -module temp_module]
      } else {

      # Abort this change since the slack is not significantly better.
           delete_object [find -module top]
           do_rename temp_module top
}
```

Related Information

do_rebind

do_create_hierarchy

```
do_create_hierarchy [-module name] [-instance name] [-no_feedthrough]
     [-dont_preserve_busses] instance_list
```

Creates a new level of hierarchy by placing a list of instances in a new module. The ports of the new module are derived automatically through the connections between the instances that are in the new module and the instances that are outside the new module.

The command `do_create_hierarchy` introduces a new module in the database and alters the existing hierarchy. By default, this command preserves the net and port busses in the newly created hierarchy.

Options and Arguments

`-dont_preserve_busses`
: Does not preserve the busses in the new hierarchy.

`-instance` *`name`*
: Specifies name of the instance created in the parent module. The default name is generated using `instance_generator`.

`instance_list`
: Specifies the list of instances that make up the new module.

`-module` *`name`*
: Specifies the name for the new module. The default name is automatically generated using the string set by the attribute.

`-no_feedthrough`
: Does not allow feedthrough wires to be created in the new hierarchy. This prevents the formation of feedthrough ports to model external uses of an input signal.

Attributes

`set_global instance_generator`

Example

The following command creates the new module `pcimod` by placing all instances whose name starts with `pci`.

```
> do_create_hierarchy -module pcimod [find -inst pci*]
```

Related Information

`do_dissolve_hierarchy`

`set_global`

do_delete_buffer

```
do_delete_buffer [-buf_inv_pairs] [-from_pin list_of_pins] [-inverters]
     list_of_buffer_or_inverter_names_or_ids
```

Deletes the specified buffers or inverters. The search for objects is carried out in the current module. If the object name has a hierarchy in it (for example `cnt99/inst1`), then the search is carried out at the appropriate hierarchy level.

Note: Use the `do_delete_buffer` command only on placed designs, either routed or unrouted. Do not use it on designs that are not yet placed.

Arguments and Options

`-buf_inv_pairs`
: When specified with `-from_pin, do_delete_buffer` removes all buffers and inverter pairs in the buffer tree, starting from the specified pins. This option preserves polarity.

`-from_pin` *`list_of_pins`*
: Deletes all buffers in a buffer tree, starting from the specified pins.

`-inverters`
: If specified, `do_delete_buffer` deletes only inverters. If not specified, `do_delete_buffer` deletes only buffers.

  Note: The `do_delete_buffer` command does not ensure that polarity is preserved. You must make sure that you are deleting inverters in pairs.

`list_of_buffer_or_inverter_names_or_ids`
: Specifies the list of buffers or inverters to be removed.

do_dissolve_hierarchy

```
do_dissolve_hierarchy [-hierarchical] [-view name]  [ {instance_list |
     module_list}]
```

Dissolves hierarchical instances of modules by flattening the hierarchy of the instance. Each instance specified in the `instance_list` is expanded into its parent module. The parent module now contains all the logic for the sub-modules. The hierarchical components inside the modules referenced by the instances are now components in the current module. The new names of the cells in the parent module are derived by appending a number to the original instance. Once dissolved, an instance no longer exists in the design and constraints can not be placed on it.

The netlist changes as the hierarchical components inside the modules referenced by the instances are now components in the current module.
Default: Dissolves the current module into its parent.

Options and Arguments

`-hierarchical`
: All hierarchical components of the instances are recursively expanded to flatten the current module. The current module now only has instances of the library cells and instances of the modules marked by the `set_dont_modify` command. If the instance list and the hierarchical option are not specified, then all instances of the current module are expanded in their respective parent modules.

`instance_list`
: Specifies list of all instances to be dissolved.

`module_list`
: Specifies list of all modules to be dissolved.

`-view` *`name`*
: Specifies list of all names to be dissolved.

Attributes

`set global hierarchy divider`

Example

The following figures show the hierarchical design before and after using this command:

```
> do_dissolve_hierarchy [find -instance unit2].
```

Figure 1-2 Before do_dissolve_hierarchy

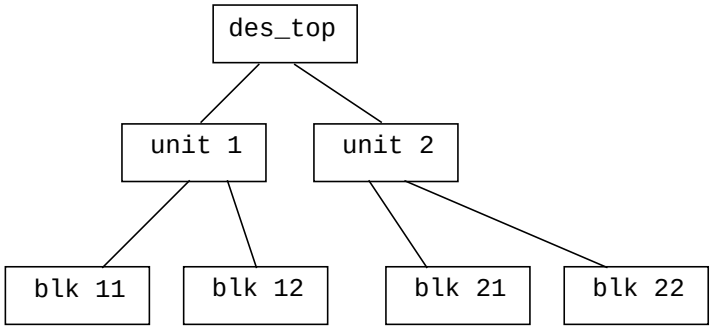
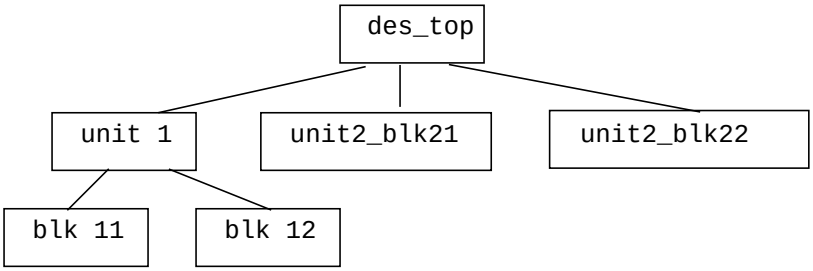


Figure 1-3 After do_dissolve_hierarchy



Related Information

do_create_hierarchy

do_uniquely_instantiate

set_current_module

set_dont_modify

do_extract_critical

```
do_extract_critical [-critical_ratio float] [-critical_offset float]
     [-fanin_depth integer] [-fanout_depth integer] [-hierarchical]
```

Extracts a section of the current module based on the worst slack. The `critical_ratio` option allows you to specify a wider extraction region based on a fraction of the worst slack. For example, a `critical_ratio` of `0` causes only the critical path to be extracted. A critical ratio value of `0.5` means that all components on paths having slack up to 0.5 times the worst (most negative critical) slack are also extracted. Likewise, the `critical_offset` option allows you to specify a wider extraction region based on a specific relative amount from the worst slack. For example, a `critical_offset` value of 1.5 means that all nets whose slack falls between the worst slack and the worst slack number minus1.5 time units are extracted including the two nets defining this region. The extracted module can be manipulated or written in the same way as any other design module.

When the `-hierarchical` option is used, the `do_extract_critical` command extracts the critical path of a design and writes it to a separate new module at each level of the hierarchy, starting with the current module. Thus, all of the hierarchical components on the critical path are extracted from the current module and the hierarchy of the components is maintained. In the absence of this option, extraction is done for the current level of hierarchy. It is not done for sub-levels.

The `do_extract_*` commands extract sections of the current module and write them to a separate visible module in the design hierarchy. The extracted section of the module is determined by the particular `do_extract_*` command used. These commands only work on the current module (identified by the `set_current_module` command). Paths between modules are not extracted. To extract a path that exceeds a module boundary, you must execute a `do_dissolve_hierarchy` command to group the desired logic within a single module boundary, then execute the particular `do_extract_*` command.

In all cases, when using the extraction commands, the extracted module contains more inputs and outputs than expected. This is necessary to capture the arrival and required constraints of the extracted piece of logic in the context of the original module. For example, a net that exists in the middle of a critical path that also fans out to non-critical logic will appear as an output of the extracted module.

A new module and an instance of that module is created by extracting the critical section of the netlist. The hierarchy is changed. You must run the `do_dissolve_hierarchy` command to restore the original hierarchy

Options and Arguments

`-critical_offset` *`float`*
: Indicates the range of slack as a relative amount of the worst slack in the module for which the gates would be extracted.
Default: `0`, indicating the critical path only.

`-critical_ratio` *`float`*
: Indicates the range of slack as a fraction of the worst slack in the module for which the gates would be extracted.
Default: `0`, indicating the critical path only.

`-fanin_depth` *`integer`*
: Specifies the number of levels of fanin logic to extract.
Default: `0`

`-fanout_depth` *`integer`*
: Specifies the number of levels of fanout logic to extract.
Default: `0`

`-hierarchical`
: Extracts the critical path through the hierarchy starting with the current module and placing the portion extracted from a particular module in its own module at each level of the hierarchy.

Example

The following example extracts the path with the worst slack (in the current module), gets the name of the module, and prints a hierarchy report showing where the critical module falls in the hierarchy.

```
> set extracted_mod [do_extract_critical]
73473

> get_name $extracted_mod
top_critical_0_critical_0

> report_hierarchy
|-top(g)
| |-top_critical_0(g)
| | |-top_critical_0_critical_0(g)
```

Related Information

do_dissolve_hierarchy

do_extract_fanin

do_extract_fanout

do_extract_non_critical

do_extract_fanin

```
do_extract_fanin [-level integer] [-threshold float] [-early] [-module module]
     [-tristate] [-sequential] list_of_object_id
```

Extracts a section of the netlist defined by the fanin-cone of the specified list of object identifiers. All the logic associated with the list of object ids is placed in one extracted module.

Note: For more information about extraction commands, see do_extract_critical on page 85 and do_extract_fanout on page 90.

A new module and an instance of the module is created by extracting a section of the netlist specified by the fanin-cone. An extra level of hierarchy is added to the design.

Options and Arguments

`-early`

Specifies early slack when used with the threshold option. *Default*: `late`. This option is also known as hold.

`-level` *`integer`*

Specifies a maximum number of logic levels to extract. *Default*: Extracts all the logic in the fanin cone.

`list_of_object_id`

Specifies a list of the nets, pins, or ports from which to start the extraction.

`-module` *`module`*

Works on the specified module, not the current module.

`-sequential`

Extracts sequential element into the new module. If not specified, the extraction will not include sequential elements. The value returned is the object id of the module created. If no module is created a null string is returned.

`-threshold` *`float`*

Extracts only the logic having a slack of the value specified by *`float`* or a worse slack value.The threshold option is an optional qualifier that limits the fanin cone based on slack, not on logic levels. For example, setting a threshold of -2 and a level of 4 extracts all gates within 4-levels of logic of the inputs to the

given objects that have slack of -2 or worse. The more negative the number for slack, the worse it is.

`-tristate`

Extracts a 3-level cone from any bit of the output bus. Cones of logic will include tristate cells and the respective fanin based on the other options specified. Without the `-tristate` option, command stops extraction on the logic cone when a tristate cell is found.

Example

The following command extracts a three-level cone from any bit of the output bus `out`:

```
> do_extract_fanin -level 3 [find -output -port out*]
```

Related Information

`do_dissolve_hierarchy`

`do_extract_fanin`

`do_extract_fanout`

`do_extract_non_critical`

do_extract_fanout

```
do_extract_fanout [-level integer] [-threshold float] [-early] [-module module]
     [-tristate] [-sequential] list_of_object_id
```

Extracts a section of the design determined by the fanout cone of the specified list of object identifiers. All the logic associated with the list of object ids is placed in one extracted module.

A new module and an instance of the module is created from the extracted section of the netlist, specified by the fanout cone. A level of hierarchy is added to the design.

Note: For more information on the extraction commands see the do_extract_critical on page 85 and do_extract_fanin on page 88.

Options and Arguments

`-early`

Specifies early slack when used with the threshold option. *Default*: `late`. This option is also known as hold.

`-level` *`integer`*

Specifies a maximum number of logic levels to extract. *Default*: Extracts all the logic in the fanout cone.

`list_of_object_id`

Specifies a list of the nets, pins, or ports from which to start the extraction.

`-module` *`module`*

Works on the specified module, not the current module.

`-sequential`

Extracts sequential element into the new module. If not specified, the extraction will not include sequential elements. The value returned is the object id of the module created. If no module is created a null string is returned.

`-threshold` *`float`*

Extracts only the logic having a slack of the value specified by *`float`* or a worse slack value. The threshold option is an optional qualifier that limits the fanout cone based on slack, not on logic levels. For example, setting a threshold of -2 and a level of 4 extracts all gates within 4-levels of logic of the inputs to the

given objects that have slack of -2 or worse. The more negative the number for slack, the worse it is.

`-tristate`

Extracts a 3-level cone from any bit of the input bus. Cones of logic will include tristate cells and the respective fanout based on the other options specified. Without the `-tristate` option, the command stops extraction on the logic cone when a tristate cell is found.

Example

The following command extracts a three-level cone from any bit of the input bus `in`:

```
> do_extract_fanout -level 3 [find -input -port in*]
```

Related Information

`do_dissolve_hierarchy`

`do_extract_fanin`

`do_extract_fanout`

`do_extract_non_critical`

do_extract_non_critical

`do_extract_non_critical [-critical_ratio` *`float`*`] [-critical_offset` *`float`*`]`

Extracts the section of the netlist that is not on the critical path. Specifying the `-critical_ratio` and `-critical_offset` options widens the region of the critical path to be excluded from extraction. This command is rarely used.

A new module and an instance of the module is created by extracting the non critical section of the netlist. A level of hierarchy is added to the design.

See the command do_extract_critical on page 85 for more general information.

Options and Arguments

`-critical_offset` *`float`*
: Indicates the range of slack as a relative amount of the worst slack in the module for which gates would be excluded from extraction.
Default: `0`, indicating the critical path only.

`-critical_ratio` *`float`*
: Indicates the range of slack as a fraction of the worst slack in the module for which gates would be excluded from extraction.
Default: `0`, indicating the critical path only.

Related Information

`do_dissolve_hierarchy`

`do_extract_fanin`

`do_extract_fanout`

`do_extract_non_critical`

do_fix_hold

```
do_fix_hold [-footprint] [-resize] [-buffer] [-dont_modify_children]
     [-no_design_rule] [-minimize] [-max_area float] [-dont_legalize]
     [-critical_ratio float] [-critical_offset float]
```

Attempts to fix hold times without trying to fix setup times. Allows the worsening of any setup slack to fix hold, but it tries to choose moves that minimize the impact on negative setups. If you do not specify the `-dont_legalize` option, `do_fix_hold` also legalizes the placement of the design.
Default: If the `-resize` or `-buffer` options are not specified, the command use both options to get the best possible results.

Note: The `do_fix_hold` command does not consider regions when moving cells during optimization, so it is possible for some cells to get moved outside their region. In general, cells within a region are moved only when there is a good reason, such as improving timing or routability. If your design requires that cells not be moved outside of their regions, consider using the `set_dont_move` command to mark the instances fixed after initial placement.

Options and Arguments

`-buffer`
: Allows buffers to be used to fix hold violations.

`-critical_offset` *`float`*
: Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*
: Indicates the range of slack for paths to be considered critical during hold fixing. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

  ```
  worst_slack * (1 - critical_ratio) +
  critical_offset
  ```
  Default: `0`

  Note: This option ignores path groups. It works on the worst slack of the overall design.

`-dont_legalize`

Prevents placement legalization of the design.

`-dont_modify_children`

Applies the optimization transform only to the portion of the design contained in the current module. In the absence of this option, the transforms are applied to the entire design contained in the current module including its hierarchical descendents (children).

`-footprint`

Substitutes cells with higher (lower) drive capability for the cells with lower (higher) drive, if both cells have the same footprints (in addition to the same functionality). This assures that there are no changes in the placement or routing of the cells. This is also referred to as in-place swapping of cells. This option is useful when operating on a netlist that has been already placed and no significant changes can be made to the netlist. This option implies `-resize` option.

`-max_area` *`float`*

Prevents timing optimization from increasing area beyond the specified value.

`-minimize`

Prevents negative setup from worsening and does not allow any positive setup to become negative, but allows positive setup to become less positive.

`-no_design_rule`

Tells optimization to completely ignore all design rules.

`-resize`

Replaces a cell with an equivalent cell with a different drive when the new cell contributes toward meeting the goal of the transformation. This only occurs if the library has a choice of cells with the same functionality, but on different drives

do_insert_buffer

```
do_insert_buffer list_of_nets
     [-fix_hold] [-fraction fraction_value] [-from_sink sink_pin_name]
     [-location {xloc yloc}][-macro macro_name]
     [-tolerance tolerance_value]
```

Attempts to insert one buffer in each net in the specified list and returns a list containing the object IDs of the inserted buffers. If no buffer was inserted, it returns an empty list.

Note: You must specify the net name as the first argument. You can specify all other arguments in arbitrary order.

If you do not specify any options, PKS inserts a buffer to improve design rule violations and late slack as much as possible. If you do not specify any options, and no improvement is possible, no buffer will be inserted.

Arguments and Options

`-fix_hold` Instructs PKS to try to fix hold violations on the net.

Note: This option only has an effect when used alone or with the `-macro` option.

`-fraction` *`fraction_value`*
Specifies target wire length of the resulting net driven by the original driver, as a fraction of the original wire length. If used with the `-from_sink` option, the fraction is measured from the target sink instead of the driver.

Note: This definition different than the `-value` option of the `reduceNetC` command in `do_xform_run_repair_file`.

`-from_sink` *`sink_pin_name`*
Specifies a particular sink pin which must be driven by the new buffer.

Note: This option has no effect unless you use it with the `-fraction` option.

`list_of_nets` Specifies the nets where buffers should be inserted. You may specify each net by name or object ID.

`-location {`*`xloc yloc`*`}`
Specifies that PKS needs to place a buffer at the point on the route or Steiner tree that is closest to the specified location.

Note: The location values must be in the same units used in the DEF file.

`-macro` *`macro_name`* Specifies the name of the buffer master cell to use; if not specified, the best master cell (in terms of timing, based on design rule violations and late slack) will be inserted.

`-tolerance` *`tolerance_value`*
Specifies a tolerance limit for buffer location as a fraction of the original net wire length. PKS inserts a buffer if a location is found within the tolerance limit, otherwise it leaves the net unchanged.

Calculation of the wire length variance is based on the amount by which the wire length of each new net exceeds the *`fraction_value`* specified with the `-fraction` option.

For example: if the *`fraction_value`* is 0.4, then ideally, the fraction of the wire length before the buffer is 0.4 and the fraction after the buffer is 0.6.

If the new nets have fractions 0.35 and 0.7, the relevant value is 0.1, because 0.7 exceeds 0.6 by 0.1.

If the new nets have fractions 0.45 and 0.7, the relevant value is 0.15, because in addition to the 0.1 mentioned above, 0.45 exceeds 0.4 by 0.05.

Note: The `-tolerance` option is only meaningful when used with the `-fraction` option.

do_optimize

```
do_optimize
     [-start_with {mapping | placement | clock_tree_insertion}]
     [{-stop_before | -stop_after} {mapping | placement | clock_tree_insertion}]
     [-power [no_gatelevel_opt | low | medium | high]] (requires BGX license)
     [-dont_uniquify] [-checkpoint]
```

Options that affect generic netlist optimization:

```
[-auto_dissolve_generic integer]
[-flatten {off | auto | on}]
[-priority {area | time}]
[-minimize {multiple_output | single_pass}]
[-phase_assignment]
[-dont_structure]
[-dont_remove_redundancy]
[-force]
```

Options affect generic and mapped netlist optimization:

```
[-distributed]
[-no_partition]
[-dont_propagate_constants]
```

Options that affect mapped netlist optimization:

```
[-auto_dissolve_mapped integer]
[-scan_file]
[-preserve_scan_configuration]
[-time_budget]
[-remap_for_timing]
[-reclaim_maximum_area]
[-stop_for_power_simulation]
```

Options that affect mapped and placed netlist optimization:

```
[-effort {none | low | medium | high}]
[-incremental]
[-dont_reclaim_area]
[-max_area float]
{[-critical_ratio ratio] [-critical_offset float]} | [-all_end_points]
[-target_slack float]
[-no_design_rule | -design_rule_only | -dont_fix_design_rules]
[-design_rules_have_been_fixed]
[-dont_downsize]
[-dont_buffer]
[-dont_clone]
[-dont_resize]
[-dont_swap_pins]
[-dont_restructure]
[-restructure_aware]
[-scan_reorder] (requires PKS license)
[-skip_scan_configuration]
```

Options that affect placed netlist optimization (*all require PKS license*):

```
[-insert_clock_tree]
[-skew_clock]
[-prevent_crosstalk]
[-crosstalk_threshold]
[-crosstalk_tolerance]
[-prevent_wire_self_heat]
[-congestion_effort {none | low | medium}
[-dont_legalize | -legalize_only]
```

Options for `do_optimize -ipo`

```
[-change_limit int]
[-change_file file]
[-checkpoint]
[-max_area float]
{[-critical_ratio ratio] [-critical_offset float]} | [-all_end_points]
[-target_slack float]
[-no_design_rule | -design_rule_only | -dont_fix_design_rules]
[-dont_downsize]
[-dont_resize]
[-dont_buffer}
[-prevent_crosstalk] (requires PKS license)
[-crosstalk_threshold] (requires PKS license)
[-crosstalk_tolerance] (requires PKS license)
[-prevent_wire_self_heat] (requires PKS license)
[-dont_legalize] (requires PKS license)
[-legalize_only] (requires PKS license)
```

Performs optimization on the current module as specified with the command `set current module`. Depending on the state of the design, optimization will include some or all of the following: uniquification, constant propagation, structuring, redundancy removal, technology mapping, timing-driven optimization, buffering of multiport nets, and design-rule fixing.

Child modules which are not marked with the `dont_modify` attribute are always included in the optimization process.

If the default `do_optimize` flow is used, all child modules are simultaneously optimized along with the parent module (current_module).

If the `-time_budget` option is used, some level of optimization is done on the non-uniquified child modules prior to uniquification and optimization of the parent module. With the `-time_budget` option, the initial timing optimization of the non-unique child modules uses time budgeting to derive the constraints for each child module. After each child module has undergone this initial optimization, uniquification is performed and the parent module and all child modules are simultaneously optimized as in the default flow. The use of the `-time_budget` option may improve the run time and quality of results obtained. The designs

most likely to benefit from this option are those with a fair amount of module reuse or large design databases.

Note: During optimization, all early timing constraints as well as all clock constraints are ignored. The following globals are temporarily set:

- `timing_disable_pulsewidth_checks on`
- `timing_disable_clockperiod_checks on`
- `latch_time_borrow_mode budget`

The clock network is implicitly considered `dont_modify`.

Note: The `do_optimize` command does not consider regions when moving cells during optimization, so it is possible for some cells to get moved outside their region. In general, cells within a region are moved only when there is a good reason, such as improving timing or routability. If your design requires that cells not be moved outside of their regions, consider using the `set_dont_move` command to mark the instances fixed after initial placement.

Options and Arguments

`-all_end_points`
: Works only on the worst endpoint, and gives up if it cannot improve that endpoint. This argument tells optimization to skip such an endpoint and continue working on the next endpoint until all endpoints that do not meet the target slack have been maximally optimized for timing.

`-auto_dissolve_generic [`*`integer`*`]`
: Works on *generic* netlists to decide whether to dissolve the instance of the module into the parent module instance. If the number of instances in the module is lower than the specified limit, the module is dissolved.
Default: If no value is specified, infinite is used and no action is taken.

`-auto_dissolve_mapped [`*`integer`*`]`
: Works on *mapped* netlists to decide whether to dissolve the instance of the module into the parent module instance. This option works from the bottom up; child instances are analyzed before their parent instances. If the number of module instances is lower than the specified limit, the module is dissolved.
Default: If no value is specified, infinite is used and no action is taken.

`-change_file` *`filename`*

Works only with IPO. Changes made to the netlist are written out to the file specified. The program must have write privileges to the directory where the file is to be created. The *`filename`* can be the full path name to a file. If not, the file is opened in the current directory.

Each line contains information about a specific change in the following format:

change *`old_cell new_cell hierchical_instance_name`*

where *`change`* is `ADD` or `DELETE` or `RESIZE`;

`old_cell` is the name of the cell bound to this instance initially. If it is a new instance, then it is printed as '-'.

`new_cell` is the name of the cell bounded to this instance after optimization is completed.

`-change_limit` *`integer`*

Works only with IPO. Indicates the maximum number of changes allowed in the netlist with regards to the number of instances added, deleted, and resized when doing IPO. Connectivity changes (that is, when connection to instances are added, deleted, etc) are not counted. Changes to a particular instance are counted once, for example, if an instance is first created, and then resized in the optimization process, it will be counted as one change in the netlist.

`-checkpoint`

Creates a checkpoint of the design in `checkpoint.adb` file after each major step. The timing optimization step may create intermediate checkpoints as there are several important changes taking place. The number of times checkpoints are done depends on the effort level and other options used. All other steps create a checkpoint at the end of that step.

There is only one `checkpoint.adb` file created. If a previous file exists, either from previous session or from previous steps in the current session, it is overwritten with the new adb format information.

`-congestion_effort {none | low | medium| high}`
Specifies the amount of effort PKS should put into congestion analysis
Default: `medium`):

`high`
Currently the same as medium. Reserved for future enhancements to congestion analysis.

`low`
Congestion analysis is done only at placement-setup time.

`medium`
Congestion analysis is done at placement setup as well as in every physical break operation.

`none`
No congestion analysis is done.

`-critical_offset` *`float`*
Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

Note: Do not use this option if you have specified path groups.

`-critical_ratio` *`float`*
Indicates the range of slack for paths to be considered critical toward the end of optimization for area reclamation and timing-driven buffer insertion, cloning, and resizing. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

```
worst_slack * (1 - critical_ratio) +
critical_offset
```
Default: `0`

Note: Do not use this option if you have specified path groups. Use the `-all_end_points` and `-target_slack` options instead.

`-crosstalk_threshold` *num*
Specifies a new threshold against which crosstalk prevention is performed. Use this with the `-prevent_crosstalk` option.
Default: `MAXFLOAT`

`-crosstalk_tolerance` *num*
Specifies a number, which together with the calculated or specified threshold, defines a range within which crosstalk violations will be tolerated and need not be fixed. Thus, the crosstalk violation at a net is calculated as the following:

violation = worst net transition time - (threshold + tolerance)

where a positive number indicates a violation. Use this with the `-prevent_crosstalk` option.
Default: `0`

`-design_rules_have_been_fixed`
Use this option if design rules violations on the netlist have already been fixed.

When starting with an unplaced design, the default behavior of optimization is to ignore design rules until later in the process. With this option, however, optimization honors design rules from the start.

When starting with a placed design, the default behavior is to fix design rule violations before optimizing timing. With this option, however, timing is optimized first, and then any design rule violations are fixed.

`-design_rule_only`
Performs only design rule fixing and placement legalization.

`-distributed`
Performs the optimization in distributed mode. This option cannot be applied in conjunction with `-time_budget`.

`-dont_buffer`
Prevents the use of buffers during optimization.

`-dont_clone`
Prevents the tool from cloning instances during optimization.

`-dont_downsize`
: Prevents downsizing of cells. Usually used to prevent area reclaim from downsizing drivers before routing.

`-dont_fix_design_rules`
: Flow skips all design rule fixing steps.

`-dont_legalize`
: Prevents placement legalization of the design.

`-dont_propagate_constants`
: Prevents propagation of constants during optimization.

`-dont_reclaim_area`
: Prevents the downsizing and removal of buffers and the decloning of instances to reduce area. By default, area reclamation is done when slack fixing ceases to find improvement. This option is useful when you want prevent area reclamation in parts of the design that are not timing critical. This option cannot be used with the `-reclaim_maximum_area` option.

`-dont_remove_redundancy`
: Prevents removal of redundancies during generic optimization.

`-dont_resize`
: Prevents resizing during optimization.

`-dont_restructure`
: Prevents restructuring routine from being called for timing optimization.

`-dont_structure`
: Prevents structuring during generic optimization.

`-dont_swap_pins`
: Prevents the swapping of pins during optimization.

`-dont_uniquify`
: Prevents the design from being uniquified. By default `do_optimize` will uniquify the design. This might be useful to save run time on structuring multiple instantiated modules.

`-effort {none | low | medium | high}`
Controls the CPU time spent during the timing optimization step.
Default: `medium`

`high`
Further improvements by applying various time- consuming algorithms.

`low`
Operation is done quickly to meet requirements through easy optimization steps.

`medium`
Extra time is spent searching for alternate mappings and structures to meet all the constraints.

`none`
No slack optimization is done for path groups that do not have an effort level previously set by the `set_path_group_options` command.

`-flatten [on | off | auto]`
Controls the operations employed for optimization of logic equations. If flatten option is set to `on`, the logic equations are flattened into a sum of products form before applying optimization. This generally helps in creating faster and optimal implementation of the logic, even though it may take up extra area (gates). For some circuits, this option may result in excessive run time or memory usage. If flatten option is set to `auto`, only certain flattening operations are performed; some of the time consuming steps are skipped. The flattening step may be disabled entirely by setting the option to `off`.
Default: `off`.

`-force`
Restarts optimization with all generic optimizations. In essence, the design is unplaced and unmapped and then reoptimized from scratch.

`-incremental`
Indicates that the design is already well-optimized and that design rule violations have been fixed. In addition, optimization occurs at the highest slew depth, unless the global `auto_slew_prop_selection` is set to `false`.

`-insert_clock_tree`

Enables automatic clock tree insertion. After automatic clock tree insertion, the tool automatically switches the `clock_mode` to propagated. All clock tree constraints must be properly set before you run optimization. See `do_build_clock_tree` for more details.

`-legalize_only`

Legalizes the placement of the design and performs additional timing optimizations.

Starts with `do_place -eco` and then runs buffer, resize, pinswap, and design rule fixing optimizations to try to recover from any slack and design rule violations that might have occurred from placement legalization, and then runs `do_place -eco` again. The command has an implied `-incremental` option.

`-max_area` *`float`*

Prevents timing optimization from increasing area beyond the specified value.

`-minimize {multiple_output | single_pass}`

Controls what two-level logic minimization strategy is used when sum-of-products (SoP) logic is extracted (see the `hdl extract sum of products logic` global variable) or a netlist is flattened (see `-flatten` option).

`multiple_output` indicates that equations for all outputs of module will be minimized all together, increasing the chance of product terms of equation being shared among different outputs.

`single_pass` forces BuildGates Synthesis to perform minimization in one pass, rather than iteratively, to reduce optimization effort for fast run time.

`-no_design_rule`

Tells optimization to completely ignore all design rules.

`-no_partition`

Prevents automatic partitioning of the design into smaller modules for optimization, irrespective of the size of the module. Each module boundary should be retained as the partition to

optimize. Specifying the `-no_partition` option generally provides better quality of results than omitting it (although runtime may increase).

`-phase_assignment`

Allows BuildGates Synthesis to invert the phase of outputs, perform minimization on both ON-set and OFF-set of the equation, and pick the simpler implementation of logic when PLA is extracted, or when a netlist is flattened to a two-level form. It can apply to either `single_output` or `multiple_output` strategy, depending on the setting of option `-minimize`. When OFF-set of equation is much bigger than ON-set, phase assignment optimization could be very expensive in terms of run time.

`-power`

Performs all types of power transformations to simultaneously optimize power, delay, and area. This command honors power constraints set with `set_power_optimization_options`.

If you have sleep-mode logic inserted from running the `do_build_generic -sleep_mode` command, `do_optimize -power` will first commit/decommit the logic based on potential power savings. The `-power` argument will also commit/decommit clock-gating logic as well as perform the physical knowledgeable decloning, cloning, and root gating if PKS is involved.

The effort level you include with the `-power` argument specifies the scope of the gate-level power optimization that you want. In general, the higher the level, the better the power results, but the longer the run time for the optimization.
Default: `medium`

Value: Specify one of the following four effort levels:

`high`
Performs gate-level power optimizations at the highest level. It does more iterations than the `low` and `medium` effort levels to minimize the power consumption.

`low`
Performs gate-level transformations, such as resizing, pin

swapping, and restructuring, to minimize power consumption while trying to meet timing.

medium
Performs the same gate-level transformations as the `low` effort level, but does more iterations to simultaneously optimize the power and timing. This level is recommend for a non-optimized circuit.

no_gatelevel_opt
No gate-level optimization is performed.

The following globals might affect the `-power` transformation:

`power_clock_gate_decommit_in_do_opt`
`power_clock_gate_insert_in_do_opt`
`power_clone_declone_in_do_opt`
`power_gatelevel_opt_in_do_opt`
`power_no_sleepmode_in_resource_sharing`
`power_opt_no_tcf`
`power_root_gate_in_do_opt`

`-preserve_scan_chain_configuration`
Preserves existing pre-placement configuration of scan chains.

`-prevent_crosstalk`
Prevents crosstalk problems heuristically by fixing the design such that no net in the design has a transition time longer than a given threshold.
Performed after the design is placed and design rules have been fixed. Options to `do_xform_prevent_crosstalk` are available to `do_optimize` through the following globals when `-prevent_crosstalk` is specified: `crosstalk_threshold` (*Default*: `MAXFLOAT`) and `crosstalk_tolerance` (*Default*: `0`).

`-prevent_wire_self_heat`
Performs wire-self-heat prevention after a design is placed and the design rules are fixed.

`-priority {area | time}`
Sets area or timing minimization to be the highest priority of the technology independent optimization step. If you have a loosely

constrained or unconstrained design, then the priority should be set to `area`.
Default: `time`

`-reclaim_maximum_area`
Specifies that the optimization engine performs area reclamation transforms, allowing any slack to worsen up to the worst negative slack in the design.
Default:, Area reclamation is only done if slack does not worsen or if it remains positive.

`-remap_for_timing`
Performs timing-driven remapping of the design. After the initial technology mapping is done, the `do_optimize` command may do remapping of the cells for timing optimization. If this option is not used then `do_optimize` command will not remap the design.

`-restructure_aware`
Enables the standard optimization engine to perform combinational logic restructuring on AmbitWare (ACL) modules. Without this option, restructuring can only be achieved through component swapping by the Datapath engine.

`-scan_file` *`file`*
Declares a file name to be given to the scan report file. This option is used when test synthesis is enabled.
Default: *`top_module`*`.scan`

`-scan_reorder` Reorders mapped scan chains during or after placement.

`-skew_clock`
Inserts buffers on the leaves of the clock tree (when working with a clock-tree inserted design). The clock must be in propagated mode.
Default: `do_optimize` does not modify the clock network.

The buffers will be inserted next to the receiving sequential element (flop, latch, RAM, and so forth) in order to improve the worst late slack of any path group in the design.

For more control over which buffers are used, you can set the `do_optimize_skew_clock_buffer_list` and

do_optimize_skew_clock_ignore_dont_utilize globals.

-skip_scan_configuration
Skips the scan chain connection pass which is implicitly run by do_optimize if TDRC data is present. This option is to be used on scan-mapped structural netlists only. Avoid using this option with mixed RTL/structural databases.This option allows for preconditioning of the netlist using do_optimize, while avoiding reconfiguration of the scan chains due to the presence of TDRC data. The goal is to preserve the scan chain architecture.

-start_with {mapping | placement | clock_tree_insertion}
Resumes optimizing where the do_optimize -stop_before *state_change* command stopped optimizing, or it skips any optimizations prior to the specified state change.

-stop_after {mapping | placement | clock_tree_insertion}
Stops optimization after the specified state change. A do_optimize -stop_after *state-change* can be resumed with just the do_optimize command.

-stop_before {mapping | placement | clock_tree_insertion}
Stops optimization before the specified state change.

-stop_for_power_simulation
Stops the optimization in a LPS flow at for an appropriate state suitable for performing the simulation for generating TCF. You should dump the netlist after this state and perform simulation and read back the TCF and continue timing and power optimization.

-target_slack *float*
Specifies the worst slack allowed for any path in the group. Optimization proceeds until the worst path has a slack equal to or greater than the target slack.
Default: 0.0

You can specify negative slack to compensate for over-constraining and to cut down on runtime.

You can specify a positive value to cause the tool to optimize to a positive slack number.

`-time_budget`

Uses time budgeting to derive constraints and optimize modules in the design hierarchy; performs a multi-pass optimization, first optimizing in a bottom-up fashion using the time budgeted constraints and then following with a hierarchical compile (default compile) from the root (top-level) module.

Global Variables

Set the following variables using the `set_global` command:

`do_optimize_skew_clock_buffer_list`

`do_optimize_skew_clock_ignore_dont_utilize`

`map_inversion_through_register`

`slew_propagation_mode`

`target_technology`

`time_budget_stop_before_uniquification`

`time_budget_min_size`

Examples

- The following example shows a series of `do_optimize` commands in BuildGates Synthesis:

```
> # optimize the generic netlist
> do_optimize -stop_before mapping

> # the next step is equivalent to "do_xform_map -hier"
> do_optimize -start_with mapping -stop_after mapping

> # optimize the mapped netlist
> do_optimize
```

- The following example is a series of `do_optimize` commands in PKS:

```
> # optimize the generic netlist
> do_optimize -stop_before mapping

> # the next step is equivalent to "do_xform_map -hier"
> do_optimize -start_with mapping -stop_after mapping

> # do preplacement optimizations on the mapped netlist
> do_optimize -stop_before placement

> # the next step is equivalent to "do_place"
> do_optimize -start_with placement -stop_after placement

> do_optimize -dont_legalize
> do_optimize -legalize_only
```

Related Information

`do_dissolve_hierarchy`

`do_xform_prevent_crosstalk`

`get_attribute`

`read_verilog`

`set_attribute`

`set_current_module`

`set_top_timing_module`

`write_verilog`

do_pipeline_check

```
do_pipeline_check
```

Checks if the current module is a properly pipelined module and returns the number of registers used for each pipeline stage. Use this command before `do_pipeline_retime` to make sure the design is completely combinational and has a proper clock port setup.

Options and Arguments

None

Related Information

`do_pipeline_retime`

do_pipeline_retime

```
do_pipeline_retime
```

Retimes the current_module to minimize the worst slack. Before retiming, a check is made to see if the module is retimable.

Options and Arguments

None

Related Information

`do_pipeline_check`

do_pop_module

`do_pop_module`

Restores the current module that was active before the last `do_push_module` command. Pop and push commands can be stacked arbitrarily. If there is no module on the stack, a `do_pop_module` command sets the `current_module` to an empty string.

Options and Arguments

None

Examples

See the examples for do_push_module on page 115.

Related Information

`do_push_module`

`get_current_module`

do_push_module

`do_push_module` *`module_id`*

Temporarily changes the current module by pushing the existing current module onto an internal stack, and changing the new current module to the module specified. Use the `do_pop_module` command to recover the previous `current_module`.

Note: This command does not affect the `set_top_timing_module`.

Options and Arguments

`module_id`

The object identifier of the module that is to be made the current module after pushing the current module on the stack.

Examples

- The following command returns the object ID of the current_module (`mod1`):

```
> get_names [get_current_module]
mod1
```

- In this example, module `mod2` is pushed onto the top of the stack, changing the current module to `mod2`:

```
> do_push_module [find -module mod2]
> get_names [get_current_module]
mod2
```

- Here, the previous current module is restored from the stack:

```
> do_pop_module
> get_names [get_current_module]
mod1
```

Related Information

`do_pop_module`

`get_current_module`

do_rebind

```
do_rebind [-create_inst_pins] [-delete_inst_pins]
     [-create_ports] instance_name_or_id cell_name_or_id
```

Changes the reference of any instance from a black box, technology cell, and hierarchical module to any other technology cell and hierarchical module.

A functional match is not required by `do_rebind,` but if the pins on the instance do not match the pins on the new technology, cell, and hierarchical module, do_rebind will try to do the functional match. In case the functional match also fails, the tool will try to make a match between the instance and new technology cell and hierarchical module based on the options specified.

Since a functional match is not a requisite, it is possible to corrupt the logical function of a netlist by rebinding cells inappropriately. Use the function get_equivalent_cells to determine the set of functionality equivalent cells for resizing purposes.

The timing analysis information is updated automatically for this change. There is no need to take any additional action to guarantee accurate timing reports or optimization after executing the `do_rebind` command. After using the `do_rebind` command, parasitics data is removed from the design and replaced by SDF predictors.

Options and Arguments

`-create_instance_pins`
: Creates extra pins on the instance for ports that have no instance pins with the same name.

`-create_ports`
: Creates extra ports on the hierarchical module for the instance pins that have no corresponding ports.This option is not valid with the `-delete_inst_pins` option. The `-create_ports` option is ignored when called for rebinding with a technology cell.

`-delete_inst_pins`
: Deletes the instance pins that have no corresponding port on the technology cell/hierarchical module.

`instance_name_or_id`
: Specifies the object ID or name of the instance to rebind.

`cell_name_or_id`
: Specifies the new cell name or ID to bind the instance to.

Examples

- The following example rebinds a technology cell instance:

```
> get_names [find -cellref IV*]
IV IVA IVAP IVDA IVDAP IVP
> get_equivalent_cells [find -cellref IV]
IVP IVA IV IVAP B5I B4I B5IP B4IP B2A
> do_rebind [find -instance iv] IVP
Info:    Bound instance 'iv' to cell 'IVP' <TCLCMD-705>
```

- This example rebinds a module instance:

Before:

```
module top (in, out);
      input in;
      output out;
      sub sub1(in, out);
      sub sub2(in, out);
endmodule
module sub(in, out);
      input in;
      output out;
      assign out = ~in;
endmodule
```

Script excerpt:

```
> do_copy_module sub sub_copy_1
> do_rebind sub1 sub_copy_1
```

After:

```
module sub(in, out);
      input in;
      output out;
      not i_0(out, in);
endmodule
module sub_copy_1(in, out);
      input in;
      output out;
      not i_0(out, in);
endmodule
module top(in, out);
      input in;
      output out;
```

```
        sub sub2(.in(in), .out(out));
        sub_copy_1 sub1(.in(in), .out(out));
    endmodule
```

Related Information

create_instance

do_copy_module

get_equivalent_cells

do_remove_design

```
do_remove_design [-hierarchical] [-all] [{list_of_module_name_or_id}]
```

The do_remove_design command deletes modules from the database.

Options and Arguments

`-all`

Deletes all modules. If you use the `-all` argument and provide a *`list_of_module_name_or_id`*, the list is ignored.

Note: Using the `-all` argument will not remove globals, `clock_propagation` mode, ideal clocks, and `clock_to` clock assertions (such as insertion delays, path exceptions, uncertainty, and so on). If you need to change these settings, you can either exit the tool and restart it, or use the appropriate `reset` commands.

`-hierarchical`

Deletes the specified module and its hierarchical submodules.

`list_of_module_name_or_id`

Deletes the specified modules.

Example

The following command deletes all modules of the current design from the database.

```
> do_remove_design -all
```

do_rename

```
do_rename [-hierarchical] [-net] [-instance] [-net_to_port]
```

Changes the names of instances or nets in a design. The generators are used to generate names for instances and the nets created in the design. The `ac_shell` has an instance generator and a net generator. The name generators defaults can be set by the `set_global` command. See the `set_global net_generator` and `set_global instance_generator` commands on setting net and instance names. You can change the naming convention for nets or instances previously generated in the current module by using the `do_rename` command.

To change the name of a specific net or instance use the `do_change_name` command.

Options and Arguments

`-hierarchical`

Specifies that the renaming is also to be done for the modules in the downward hierarchical path of the current module.

`-instance`

Indicates that only the instance names are changed, based on the current instance generator, set using the `set_global` command. The net names remain unchanged.

`-net`

Indicates that only the net names are modified based on the current net generator, set using the `set_global` command. The instance names remain unchanged.

`-net_to_port`

Sometimes the back-end layout tools require that the net connected to the port be of the same name as the port (to eliminate non structural statement, such as the `assign` statement in Verilog). This option changes the net names to match that of the ports, when possible.

Related Information

`do_blast_busses`

`do_change_name`

do_uniquely_instantiate

```
do_uniquely_instantiate [-hierarchical | instance_list]
```

Creates unique modules for the instances. Use this command when one instance of a module needs to be optimized differently from another. For this purpose, each instance is referenced by a unique module so that different constraints can be applied and different transformations can be performed on each module.

If no option is specified, then only the instances in the current module specified by the `set_current_module` are uniquified.

Note: The `do_optimize` command and the `do_xform_propagate_constants` command automatically uniquifies the whole design by default unless the `-dont_uniquify` option is used.

Options and Arguments

`-hierarchical`
: Associates all instances in the hierarchical tree of the current module with uniquely created modules.

`instance_list`
: Specifies that only the instances in the `instance_list` are uniquified. The `-hierarchical` option is ignored when `instance_list` is specified.

Example

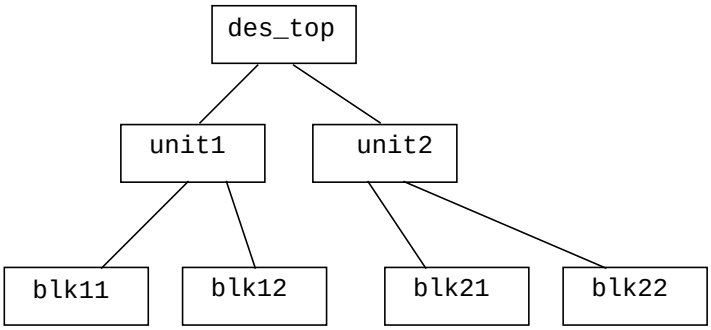
In the design shown in Figure 1-4 on page 123, `blk11` and `blk21` are instances of an adder (`mod_adder`) that need different constraints during optimization. The following command generates the unique modules:

```
> do_uniquely_instantiate blk11 blk21
```

The two instances would appear as follows (where `mod_adder_0` and `mod_adder_1` are unique instances of `mod_adder`):

```
mod_adder_0 blk11( …port connections…);
mod_adder_1 blk21( …port connections…);
```

Figure 1-4 Example of do_uniquely_instantiate



Related Information

do_dissolve_hierarchy

do_optimize

set_current_module

do_xform_buffer

```
do_xform_buffer [-dont_modify_children] [-no_design_rule] [-max_area float]
     [-critical_ratio float] [-critical_offset float]
```

Inserts buffers to improve timing on the critical path.

Use of this command is not recommended or encouraged. Use the following command instead:

```
do_optimize -effort low -dont_size -dont_clone -dont_swap_pins ...
```

Options and Arguments

`-critical_offset` *`float`*

Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*

Indicates the range of slack for paths to be considered critical for timing-driven buffer insertion. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

```
worst_slack * (1 - critical_ratio) +
critical_offset
```

Default: `0`

Note: This option ignores path groups. It works on the worst slack of the overall design.

`-dont_modify_children`

Applies the optimization transform only to the portion of the design contained in the current module. In the absence of this option, the transforms are applied to the entire design contained in the current module including its hierarchical descendents (children).

`-max_area` *`float`*

Prevents timing optimization from increasing area beyond the specified value.

`-no_design_rule`

Tells optimization to completely ignore all design rules.

do_xform_buffer_tree

```
do_xform_buffer_tree [-min_fanout value] [-non_hierarchical]
     [-input_ports | -output_ports | -ports | -pins] [-dont_remove_buffers] [net]
```

Inserts balanced buffer trees in the current module and all sub-modules. Each buffer tree insertion is made only if there is no worsening of the local slack at the buffer tree drive cell. If there is no change of slack or the net is unconstrained, the buffer tree is inserted.

One usage of buffer tree insertion is when assembling pieces of an unoptimized design, as in a time budgeting flow. For this reason, a form of the command is provided that creates buffer trees for nets that are connected to module ports or pins only, for greater speed.

Area is not considered. Design rule cost is also not considered, but in practice the design rule violations are usually reduced with a buffer tree in place because the sizing and fanout ratios are controlled in the buffer tree to be optimal for timing, and in most libraries this places the slews, capacitances, and fanouts well within the design rule limits.

By default, any existing buffer tree containing the target net is removed and replaced by a new buffer tree. Optionally, a buffer tree can be inserted with no existing buffer cells removed, but this is not recommended in general as the quality of results is higher when the entire buffer tree is replaced with a new tree.

The `do_xform_buffer_tree` step is performed automatically by the `do_optimize` flow, immediately after the technology mapping (`do_xform_map`) step.

Note: Inverted loads are not considered part of the buffer tree. The buffer tree is created from a network of non inverting single input/output buffer cells.

This command is not suitable for post-placement or back-annotated optimization, because it does not preserve annotated capacitances, resistances, or SDF delays as it inserts or replaces buffer trees. Any buffers created by this command will revert to the appropriate wire load model for delay calculation.

Be careful when using the *net* form of this command in a Tcl loop, as the command itself will be deleting and creating nets in the design. For example, do not attempt this code because the cached list of nets may become obsolete after the first buffer insertion:

```
foreach i [find -nets] {
   do_xform_buffer_tree $i; # may fail second time
   }
```

Options and Arguments

`-dont_remove_buffers`

Preserves all buffers in the original design. For a given net, only new buffer cells are added.
Default: The existing buffer tree is replaced with a new buffer tree.

`-input_ports | -output_ports | -ports | -pins`

Inserts buffers only on nets connected to input or output ports, or all ports, or all pins. Cannot be given with the *net* argument.
Default: All nets are considered for buffer tree insertion.

`-min_fanout` *value*

Do not insert a buffer tree when the number of loads is less than *n*.
Default: 1, which means consider all nets for buffer tree insertion regardless of fanout.

net

The object ID of a single net in the design that is to be replaced with a buffer tree. If not given, then all nets are considered for buffer tree insertion.

`-non_hierarchical`

Inserts buffers in the current module only.
Default: Inserts buffers in the current module and all unique children, recursively.

do_xform_clone

```
do_xform_clone [-dont_modify_children] [-no_design_rule] [-max_area float]
     [-critical_ratio float] [-critical_offset float]
```

Performs cloning transformations. This command splits up a net into two nets by duplicating the instance driving the original net to improve timing on critical path.

Important

Use of this command is not recommended or encouraged. Use the following command instead:

```
do_optimize -effort low -dont_size -dont_buffer -dont_swap_pins
```

Options and Arguments

`-critical_offset` *`float`*

Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*

Indicates the range of slack for paths to be considered critical for timing-driven cloning. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

```
worst_slack * (1 - critical_ratio) +
critical_offset
```

Default: `0`

Note: This option ignores path groups. It works on the worst slack of the overall design.

`-dont_modify_children`

Applies the optimization transform only to the portion of the design contained in the current module. In the absence of this option, the transforms are applied to the entire design contained in the current module including its hierarchical descendents (children).

`-max_area` *`float`*

Prevents timing optimization from increasing area beyond the specified value.

`-no_design_rule`

Tells optimization to completely ignore all design rules.

do_xform_fast_optimize

```
do_xform_fast_optimize [-effort { low | medium | high }] [-resize -buffer] [-one]
     [-critial_ratio float] [-critial_offset float]
```

Performs a simultaneous resizing and rebuffering. The algorithm is fast but the resulting slack may not be optimal.

Note: This command can only be used before placement and without backannotation.

Options and Arguments

`-buffer`
: Creates buffer trees to improve delays on nets, then skews these trees to improve delay on the critical path.

`-critical_offset` *`float`*
: Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*
: Indicates the range of slack for paths to be considered critical toward the end of optimization for area reclamation and timing-driven buffer insertion, cloning, and resizing. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

  ```
  worst_slack * (1 - critical_ratio) +
  critical_offset
  ```
  Default: `0`

  Note: This option ignores path groups. It works on the worst slack of the overall design.

`-effort {low | medium | high}`
: Controls the CPU time spent during the timing optimization step.
  Default: `medium`

`-one`
: Performs only one pass, does not iterate.

`-resize`

Replaces a cell with an equivalent cell with different drive if the new cell contributes toward meeting the goal of the transformation. This only occurs if the library has a choice of cells with the same functionality, but on different drives.

do_xform_fix_design_rule_violations

```
do_xform_fix_design_rule_violations [-footprint] [-resize] [-buffer]
     [-dont_modify_children] [-critical_ratio float] [-critical_offset float]
```

Provides a tighter control and selection on the transformations to be performed on the design. The `do_xform_fix_design_rule_violations` command is another version of the `do_optimize -design_rule_only` command. The `do_optimize` command is preferable for because of better runtime and QOR.

The transformations are applied only to the design rule violators. If none of the three options, `resize`, `buffer`, or `clone`, are used, then by default all three transformations are applied to get the best possible results. In effect, the default is the same as using all the three options.

When you do On and Off Chip Variation Analysis, the max DRV limit is obtained from the library used to compute the late path. Consequently, the min DRV limit is obtained from the library used to compute the early path. Set the pvt corner to be used for late and early path by setting the global `pvt_late_path` and `pvt_early_path`.

Note: The `do_xform_fix_design_rule_vioaltions` command does not legalize the design.

Important

Use of this command is not recommended or encouraged. Use the following command instead:

```
do_optimize -design_rule_only
```

Options and Arguments

`-buffer`
: Allows buffers to be used to fix design rule violations.

`-critical_offset` *`float`*
: Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*
: Indicates the range of slack for paths to be considered critical for design rule violation fixing. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

`worst_slack * (1 - critical_ratio) + critical_offset`
Default: `0`

Note: This option ignores path groups. It works on the worst slack of the overall design.

`-dont_modify_children`

Applies the optimization transform only to the portion of the design contained in the current module. In the absence of this option, the transforms are applied to the entire design contained in the current module including its hierarchical descendents (children).

`-footprint`

Substitutes cells with higher (lower) drive capability for the cells with lower (higher) drive, if both cells have the same footprints (in addition to the same functionality). This assures that there are no changes in the placement or routing of the cells. This is also referred to as in-place swapping of cells. This option is useful when operating on a netlist that has been already placed and no significant changes can be made to the netlist. This option implies `-resize` option.

`-resize`

Replaces a cell with an equivalent cell with different drive if the new cell contributes toward meeting the goal of the transformation. This only occurs if the library has a choice of cells with the same functionality, but on different drives.

do_xform_fix_hold

```
do_xform_fix_hold [-footprint] [-resize] [-buffer] [-dont_modify_children]
     [-no_design_rule] [-minimize] [-max_area float]
     [-critical_ratio float] [-critical_offset float]
     [-power]
```

Provides better control for transforming hold violators. It attempts to fix the hold times without trying to fix the setup times. If the `-resize` or `-buffer` options are not specified, the default behavior of the command is to use both options to get the best possible results.
Default: Allows the worsening of any setup slack to fix hold, but it tries to choose moves that minimize the impact on negative setups.

Important

This command will be obsoleted in an upcoming release. Please use `do_fix_hold` instead.

Options and Arguments

`-buffer`
Allows buffers to be used to fix hold violations.

`-critical_offset` *`float`*
Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*
Indicates the range of slack for paths to be considered critical during hold fixing. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

```
worst_slack * (1 - critical_ratio) +
critical_offset
```

Default: `0`

Note: This option ignores path groups. It works on the worst slack of the overall design.

`-dont_modify_children`
Applies the optimization transform only to the portion of the

design contained in the current module. In the absence of this option, the transforms are applied to the entire design contained in the current module including its hierarchical descendents (children).

`-footprint`

Substitutes cells with higher (lower) drive capability for the cells with lower (higher) drive, if both cells have the same footprints (in addition to the same functionality). This assures that there are no changes in the placement or routing of the cells. This is also referred to as in-place swapping of cells. This option is useful when operating on a netlist that has been already placed and no significant changes can be made to the netlist. This option implies `-resize` option.

`-max_area` *`float`*

Prevents timing optimization from increasing area beyond the specified value.

`-minimize`

Prevents negative setup from worsening and does not allow any positive setup to become negative, but allows positive setup to become less positive.

`-no_design_rule`

Tells optimization to completely ignore all design rules.

`-power`

Selects power as the secondary cost (instead of area) during hold time fixing optimization.

`-resize`

Replaces a cell with an equivalent cell with different drive if the new cell contributes toward meeting the goal of the transformation. This only occurs if the library has a choice of cells with the same functionality, but on different drives.

do_xform_fix_multiport_nets

```
do_xform_fix_multiport_nets [-dont_modify_children][-fix_constant_ports]
     [-no_design_rule]
```

Splits any net connected to more than one output port into different nets, each driven by a buffer, such that each net is connected to only one port. It transforms nets connected to multiple ports. This transformation may be necessary to make it easier for some of the placement and floorplanning tools to treat the netlist properly. This transformation also inserts a buffer on every pass-through net that directly connects input port to an output port.

Note: For best results, do not use `do_xform_fix_multiport_nets` on a netlist that may have large fanout multiport nets. This command does not handle large fanout nets efficiently. Multiport nets (or assigns) are best fixed during optimization.

Options and Arguments

`-dont_modify_children`
: Applies the optimization transform only to the portion of the design contained in the current module. In the absence of this option, the transforms are applied to the entire design contained in the current module including its hierarchical descendents (children).

`-fix_constant_ports`
: Performs buffer insertion on constant net (power or ground) driving output ports.
Default: The connections of constant nets to output ports are not modified.

`-no_design_rule`
: Tells optimization to completely ignore all design rules.

Related Information

`set_global fix_multiport_nets`

do_xform_footprint

```
do_xform_footprint [-quick] [-dont_modify_children] [-no_design_rule]
     [-fix_clock_net] [-dont_reclaim_area | -reclaim_maximum_area]
     [-max_area float] [-incremental] [-critical_ratio float]
     [-critical_offset float]
```

Important

This command is obsolete and will be removed in the next full release of the software. Use the `do_xform_resize -footprint` command for comparable results.

do_xform_ipo

```
do_xform_ipo [-change_limit integer] [-change_file filename] [-max_area float]
     [-checkpoint] [-dont_legalize] [-dont_swap_pins] [-dont_resize]
     [-dont_buffer] [-dont_climb_hill] [-dont_downsize]
```

This command is obsolete and will be removed in the next full release of the software. Use the `do_optimize -ipo` command for comparable results.

do_xform_insert_repeaters

```
do_xform_insert_repeaters [-force] [-levelize] [-load load_value]
     [-macro cell_master_name] [-minimize_delay] [-slew slew_value]
     list_of_net_names_or_ids
```

Adds repeaters to the list of specified nets. If neither the `-load` or `-slew` options are specified, buffers are added to meet existing design rule constraints on the nets. If both are specified, buffer locations are chosen to try to obtain slews and net loads less than or equal to the targets.

Use this command on a placed design, routed or unrouted. This command creates the same kind of routing as the original net. For example, if the net has detailed routing, the new buffer tree will also have detailed routing.

Note: You need to run incremental `froute` to complete the route. There is no incremental `groute`, so `groute` needs to be routed from scratch.

Options and Arguments

Note: If you use `do_xform_insert_repeaters` with a list of nets but no options, only design rule violations are fixed.

`-force`
: Instructs PKS to consider the `-slew` and `-load` targets as constraints. Also, forces PKS to try to insert at least one buffer, even if this is unnecessary, for fixing design rule violations or timing constraints.

`-levelize`
: Instructs PKS to try to minimize the number of levels of buffers used. This is useful for large-fanout nets, especially when setup slack is being optimized.

`list_of_net_names_or_ids`
: Specifies the list of nets to work on. If nets are not specified, then all nets in the current module and all modules contained in the current module, stopping at the soft block boundaries in case of physical hierarchy, are worked on.

`-load` *`load_value`*
: Specifies the target `max_load` for the newly created nets in pico-Farads.

`-macro` *`cell_master_name`*

Provides the name of the buffer cell master to insert. If the name is not specified, PKS chooses the best cell master based on fanout, load, and slew design rule constraints, and on fixing setup violations and minimizing area (in decreasing order of priority).

`-minimize_delay`

Instructs PKS to try to optimize setup slack on the net.

Note: If `do_xform_insert_repeaters -minimize_delay` results in negative slack, PKS automatically tries `-levelize` along with `-minimize_delay`. The buffer tree with the best slack is inserted.

`-slew` *`slew_value`*

Specifies the target slew on all input pins of the buffer tree, including the leaf pins.

Examples

- The following command inserts buffers on nets `net_name1` and `net_name2` in order to improve late slack through those nets:

```
> do_xform_insert_repeaters -minimize_delay {net_name1 net_name2}
```

- The following command inserts buffers on net with object id 123456 to fix design rule violations on that net:

```
> do_xform_insert_repeaters 123456
```

do_xform_map

```
do_xform_map [-no_partition] [-force] [-hierarchical] [-timing]
     [-critical_ratio float] [-critical_offset float] [-distributed]
     [-fanin_depth integer] [-fanout_depth integer]
```

Maps the generic netlist to the target technology library. It is used for timing-driven mapping. Timing is implied when using the `-timing`, `-fanin_depth`, or `-fanout_depth` options. If the `-timing` option is used in the absence of any of the other options then timing-driven mapping is only performed on the critical region as specified by the arguments of the `-critical_ratio`, `-critical_offset`, `-fanin_depth`, and `-fanout_depth` options whose defaults are all `0`.

This command maps a generic cell netlist (ATL) to a technology specific cell netlist.

Options and Arguments

`-critical_offset` *`float`*
: Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.
Default: `0`

`-critical_ratio` *`float`*
: Indicates the range of slack for paths to be considered critical for timing-driving mapping. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

```
worst_slack * (1 - critical_ratio) +
critical_offset
```

Default: `0`

Note: This option ignores path groups. It works on the worst slack of the overall design.

`-distributed`
: Performs the optimization in distributed mode.

`-fanin_depth` *`integer`*
: Extracts the non-critical fanins on the critical path. The *`integer`* value specified gives the level up to which the non-

critical fanins of the critical path are extracted.
Default: `0`

`-fanout_depth` *`integer`*
Extracts the non critical fanouts on the critical path. The integer value specified gives the level up to which these non-critical fanouts should be extracted in the critical path.
Default: `0`

`-force`
Restarts optimization with all generic optimizations. In essence, the design is unplaced and unmapped and then reoptimized from scratch.

`-hierarchical`
Performs mappings, unmappings, structuring, or removes redundancies on the modules in the downward path of the current module. It depends on the command being used. If this option is not given, then only the current module is affected by the command used.

`-no_partition`
Prevents automatic partitioning of the design into smaller modules for optimization, irrespective of the size of the module. Each module boundary should be retained as the partition to optimize.

`-timing`
Performs timing-driven mapping on the netlist.
Default, Area-driven mapping is performed.

Related Information

`do_optimize`

`do_xform_unmap`

do_xform_optimize_generic

```
do_xform_optimize_generic [-checkpoint][-clockgate][-flatten { on | auto | off}]
     [-minimize { single_output | multiple_output }][-phase_assignment]
     [-force][-no_partition][-priority { area | time }][-distributed]
     [-dont_uniquify][-dont_propagate_constants][-dont_structure]
     [-dont_remove_redundancies]
```

This command is obsolete and will be removed in the next full release of the software. Use the `do_optimize -stop_before mapping` command for comparable results.

do_xform_optimize_slack

```
do_xform_optimize_slack [-effort { low | medium | high }] [-no_design_rule]
     [-checkpoint] [-max_area float] [-critical_ratio float]
     [-critical_offset float] [-dont_reclaim_area | -reclaim_maximum_area]
     [-incremental] [-pks] [-distributed] [-power] -restructure_aware
     [-stop_before_placement] [-time_budget] [-remap_for_timing]
```

This command is obsolete and will be removed in the next full release of the software. Use the `do_optimize` command for comparable results.

do_xform_pre_placement_optimize_slack

```
do_xform_pre_placement_optimize_slack [-effort { low | medium | high }]
     [-design_rule_have_been_fixed] [-no_design_rule]  [-checkpoint]
     [-max_area float] [-critical_ratio float] [-critical_offset  float]
     [-distributed] [-dont_reclaim_area | -reclaim_maximum_area]  [-incremental]
     [-power] [-time_budget] [-remap_for_timing] [-dont_legalize]
```

Important

This command is obsolete and will be removed in the next full release of the software. Use the `do_optimize -stop_before placement` command for comparable results.

do_xform_prevent_crosstalk

```
do_xform_prevent_crosstalk [-threshold num] [-tolerance num]]
     [-change_file filename]
```

Fixes crosstalk problems in a routed design and prevents crosstalk problems during PKS optimization. The `do_xform_prevent_crosstalk` command is similar to the slew or transition DRV in that crosstalk prevention is a design rule to be fixed. Crosstalk problems are fixed heuristically such that no net in the design has a transition time longer than a given threshold. The threshold is the average transition time across all nets of the design at a given state.

Automatically calculates an average transition time the first time this command is used on a design unless the `-threshold` option is specified. The average transition time is calculated across all the nets in the design, taking the worst of all rise and fall transition times of each net. This threshold is stored with the design and all subsequent crosstalk operations or reports are carried out using the same threshold to avoid non-convergence, unless a new threshold is specified. See `reset_crosstalk_threshold` for information on how to reset the crosstalk threshold.

Note: Crosstalk prevention is valid only on a placed netlist. and can be performed while honoring other design rules at the same time. In addition, crosstalk prevention is only performed on data nets, NOT on clock nets.

Options and Arguments

`-change_file` *`filename`*
: Specifies a file into which the tool outputs the changes done to the netlist for crosstalk prevention. Output format is similar to `ipo_change_list`.

`-threshold` *`num`*
: Specifies a new threshold against which crosstalk prevention is performed.

`-tolerance` *`num`*
: Specifies a number, which together with the calculated or specified threshold, defines a range within which crosstalk violations will be tolerated and need not be fixed. Thus, the crosstalk violation at a net is calculated as the following: violation = worst net transition time - (threshold + tolerance) where a positive number indicates a violation.
Default: `0`

Related Information

`do_optimize`

`reset_crosstalk_threshold`

do_xform_prevent_wire_self_heat

```
do_xform_prevent_wire_self_heat [-change_file filename]
```

The do_xform_prevent_wire_self_heat command is similar to the do_xform_prevent_crosstalk command as wire-self-heat prevention is completed as just another capacitance limit design rule to be fixed. Valid only on placed designs.

Note: The do_xform_prevent_wire_self_heat command works only on data nets, *not* on clock nets. In addition, wire-self-heat prevention is completed while honoring and preserving other design rules.

Once wire-self-heat prevention has been performed, it will be stored with the design and used for all subsequent optimizations or design rule fixes, until do_xform_prevent_wire_self_heat is reset using the reset_wire_self_heat_prevention command.

Options and Arguments

-change_file *filename*
: Specifies a file into which the tool outputs the changes done to the netlist for wire-self-heat prevention.

Related Information

do_xform_prevent_crosstalk

report_crosstalk_violations

reset_wire_self_heat_prevention

do_xform_propagate_constants

```
do_xform_propagate_constants [-dont_uniquify]
```

Propagates constants throughout the design crossing hierarchical boundaries. The constants being propagated are the logic levels (0 or 1). This command also propagates unconnected property for the nets connected to those input ports that are not connected (driven) to any driver. The side effect is that it uniquifies all instances unless the `-dont_uniquify` option is given.
Default: Removes only constant latches. To remove constant flip-flops also, set the global option `preserve_constant_flops` to `false`.

Options and Arguments

`-dont_uniquify`
: Tells optimization not to uniquify the design. This can be useful to save run time on structuring multiple instantiated modules.

Related Information

`do_optimize`

do_xform_reclaim_area

```
do_xform_reclaim_area [-footprint] [-resize] [-buffer] [-clone]
     [-dont_modify_children] [-no_design_rule] [-critical_ratio float]
     [-critical_offset float] [-reclaim_maximum_area]
```

Applies only those transformations that reduce area, without worsening the worst negative slack. These include replacing cells with smaller area cells that have the same functionality, and removing unnecessary buffers or clones.

If none of the three options, `resize`, `buffer`, or `clone`, are used, then by default all three transformations are applied to get the best possible results. In effect, it is the same as using all the three options.

Options and Arguments

`-buffer`
: Removes unnecessary buffers from the netlist as long as timing is not worsened.

`-clone`
: Removes unnecessary clones from the netlist.

`-critical_offset` *`float`*
: Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*
: Indicates the range of slack for paths to be considered critical t for area reclamation. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

  ```
  worst_slack * (1 - critical_ratio) +
  critical_offset
  ```
  Default: `0`

  Note: This option ignores path groups. It works on the worst slack of the overall design.

`-dont_modify_children`
: Applies the optimization transform only to the portion of the

design contained in the current module. In the absence of this option, the transforms are applied to the entire design contained in the current module including its hierarchical descendents (children).

`-footprint`

Substitutes cells with higher drive capability for the cells with lower drive, if both cells have the same footprints (in addition to the same functionality). This assures that there are no changes in the placement or routing of the cells. This is also referred to as in-place swapping of cells. This option is useful when operating on a netlist that has been already placed and no significant changes can be made to the netlist. This option implies `-resize` option.

`-no_design_rule`

Tells optimization to completely ignore all design rules.

`-reclaim_maximum_area`

Performs area reclamation transforms on any slack that is better than the worst slack.
Default: Area reclamation is not done on any path that can worsen or create negative slack.

`-resize`

Replaces a cell with an equivalent cell with different drive if the new cell contributes toward meeting the goal of the transformation. This only occurs if the library has a choice of cells with the same functionality, but on different drives.

do_xform_remove_redundancy

```
do_xform_remove_redundancy [-hierarchical] [-no_partition]
```

Removes redundancies from the netlist. This command maps a generic cell netlist (ATL) to a generic cell netlist.

Options and Arguments

`-hierarchical`

Performs mappings, unmappings, structuring, or removes redundancies on the modules in the downward path of the current module. It depends on the command being used. If this option is not given, then only the current module is affected by the command used.

`-no_partition`

Prevents automatic partitioning of the design into smaller modules for optimization, irrespective of the size of the module. Each module boundary should be retained as the partition to optimize.

Related Information

`do_optimize`

do_xform_resize

```
do_xform_resize [-footprint] [-dont_modify_children] [-no_design_rule]
     [-critical_ratio float] [-critical_offset float][-max_area float]
     -upsize_only
```

Resizes the cells in the netlist. This command indicates that if the library has choice of cells with the same functionality but different drives, a cell can be replaced by an equivalent cell with different drive if the new cell contributes toward meeting the goal of the transformation.

Use of this command is not recommended or encouraged. Use the following command instead:

```
do_optimize -effort low -dont_clone -dont_buffer -dont_swap_pins
```

Options and Arguments

`-critical_offset` *`float`*

Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*

Indicates the range of slack for paths to be considered critical for timing-driven resizing. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

```
worst_slack * (1 - critical_ratio) +
critical_offset
```

Default: `0`

Note: This option ignores path groups. It works on the worst slack of the overall design.

`-dont_modify_children`

Applies the optimization transform only to the portion of the design contained in the current module. In the absence of this option, the transforms are applied to the entire design contained in the current module including its hierarchical descendents (children).

`-footprint`

Substitutes cells with higher (lower) drive capability for the cells with lower (higher) drive, if both cells have the same footprints (in addition to the same functionality). This assures that there are no changes in the placement or routing of the cells. This is also referred to as in-place swapping of cells. This option is useful when operating on a netlist that has been already placed and no significant changes can be made to the netlist. This option implies `-resize` option.

`-max_area` *`float`*

Prevents timing optimization from increasing area beyond the specified value.

`-no_design_rule`

Tells optimization to completely ignore all design rules.

`-upsize_only`

Allows control of the optimization process so that during resizing, only upsizing moves can be allowed. No downsizing moves are allowed.

Related Information

`do_optimize`

do_xform_restructure

```
do_xform_restructure [-effort {low | medium | high}] [-critical_ratio float]
     [critical_offset float] [-max_area float]
```

Performs restructuring and remapping of the critical path in order to meet the timing constraints.

Options and Arguments

`-critical_offset` *`float`*
: Adds the `-critical_offset` number to the value of the worst slack. Similar to the `-critical_ratio` option below.

`-critical_ratio` *`float`*
: Indicates the range of slack for paths to be considered critical toward the end of optimization for area reclamation and timing-driven buffer insertion, cloning, and resizing. Specify the argument *`float`* as a fraction (a number from 0 to 1, inclusive) of the worst slack. This option allows a range of worst slack to be considered critical. This should only be used on negative slack. The critical range considered is defined as follows:

  ```
  worst_slack * (1 - critical_ratio) +
  critical_offset
  ```

  Default: `0`

  Note: This option ignores path groups. It works on the worst slack of the overall design.

`-effort {low | medium | high}`
: Controls the CPU time spent during the timing optimization step.
  Default: `medium`

`-max_area` *`float`*
: Prevents timing optimization from increasing area beyond the specified value.

Related Information

`do_optimize`

do_xform_run_repair_file

```
do_xform_run_repair_file -set_dont_modify [true | false] repair_file_name
```

Runs the repair file that contains commands to repair nets found to have crosstalk violations. This command requires a PKS license.

Notes:

- Buffer insertions are not allowed on clock nets.
- After buffers are inserted during crosstalk repair, if they are not tagged as `set_dont_modify`, then when timing or DRV issues are fixed (post-repair), they could be removed.

Options and Arguments

`repair_file_name`

Specifies the name of the repair file.

`-set_dont_modify`

Marks all buffers inserted by and instances resized by `do_xform_run_repair_file` as `dont_modify`.

Default: `true`

Repair File Commands

The following are repair commands. Any line in the repair file that does not start with one of these five commands will be interpreted as a PKS Tcl command line.

Note: The repair file can contain comments; all characters after a # on a line are ignored.

bufferInsert

```
bufferInsert net_name -location x y [-macro macro_name]
[-optionalResizeTo macro_name][-kfactor float]
```

Inserts buffers at a specified net name defined by the *net_name* option. The arguments are as follows:

-kfactor float

Gives a maximum value constraint on the kfactor computed for the inserted buffer, given the net parasitics derived after the buffer is inserted. The k-factor is defined as the derivative of

delay through the buffer cell with respect to the capacitance of the net driven by the buffer. The units are kilo-Ohms, that is, nano-seconds for delay and pico-Farads for capacitance. The value specified should be greater than 0.

`-location` *x y*

Specifies the location where the buffers are inserted.

`-macro` *macro_name*

Defines the type of macro to use. PKS inserts the buffer in an available site near the specified location, cuts the existing wire, and alters the logical netlist with the newly inserted buffer and corresponding net.

net_name

Specifies the name of the net.

`-optionalResizeTo` *macro_name*

Resizes the driver of the net to the specified cell master if PKS cannot find an acceptable buffer location.

receiverReduceNetC

`receiverReduceNetC` *receiver_inst_name pin_name* `-value` *float* [`macro` *macro_name*] [`-resize` *macro_name*] [`-kfactor` *float*]

Performs the same functionality as the `ReduceNetC` command, except that the receiver instance starting point is specified. The arguments are as follows:

-kfactor float

Gives a maximum value constraint on the kfactor computed for the inserted buffer, given the net parasitics derived after the buffer is inserted. The k-factor is defined as the derivative of delay through the buffer cell with respect to the capacitance of the net driven by the buffer. The units are kilo-Ohms, that is, nano-seconds for delay and pico-Farads for capacitance. The value specified should be greater than 0.

`-macro` *macro_name*

Defines the type of macro to use.

`-optionalResizeTo` *macro_name*

Resizes the driver of the net to the specified cell master if PKS cannot find an acceptable buffer location.

pin_name

Specifies the pin name.

receiver_inst_name

Specifies the receiver instance starting point. More appropriate for multi-fanout nets, but can be applied to two pin nets.

-value *float*

Reduces the net wire length by X percent when used in conjunction with wire length reduction (in `reduceNetC` and `receiverReduceNetC`). For example, a value of 0.5 reduces wire length by 50%. The starting point of this percentage is at the receiver end of the net. A value of 0.3 says that the buffer is to be inserted at a point that cuts the wire length of the net by 30%, for example, after the transformation, the wire length of the net connected to the inserted buffer will be approximately 30% of the original net's wire length. The wire length of the net driven by the original driver will be approximately 70% of the original net's wire length.

reduceNetC

```
reduceNetC net_name -value float [-macro macro_name]
[-optionalResizeTo macro_name] [-kfactor float]
```

Inserts a buffer onto a routed net at the specified location. The arguments are as follows:

-kfactor float

Gives a maximum value constraint on the kfactor computed for the inserted buffer, given the net parasitics derived after the buffer is inserted. The k-factor is defined as the derivative of delay through the buffer cell with respect to the capacitance of the net driven by the buffer. The units are kilo-Ohms, that is, nano-seconds for delay and pico-Farads for capacitance. The value specified should be greater than 0.

-macro *macro_name*

Specifies the type of cell to insert. If none is specified, PKS determines the best macro with the least impact to timing. Once the appropriate location is determined, PKS inserts the buffer in an available nearby site, cuts the existing wire, and alters the logical netlist with the newly inserted buffer and corresponding net. Wroute is then used to make the local connections to the buffer.

net_name

Specifies the name of the net.

-optionalResizeTo *macro_name*

Resizes the driver of the net to the specified cell master if PKS cannot find an acceptable buffer location.

-value *float*

Specifies fraction of net wire length driven by the new buffer.: value needs to be between 0 and 1.

resize

```
resize inst_name -macro macro_name
```

Resizes the specified instance to the specified macro as long as they are LEQs. As the footprint of the LEQ may be larger, adjacent cells can be shifted slightly for overlap removal. The arguments are as follows:

instance_name

Specifies the name of the instance.

-macro *macro_name*

Defines the type of macro to use.

splitNet

```
splitNet net_name -value int [-macro macro_name] [-resize macro_name]
[-kfactor float]
```

Inserts multiple buffers onto a single net. The *net_name* option specifies the name of the net. The arguments are as follows:

-kfactor float

Gives a maximum value constraint on the kfactor computed for the inserted buffer, given the net parasitics derived after the buffer is inserted. The k-factor is defined as the derivative of delay through the buffer cell with respect to the capacitance of the net driven by the buffer. The units are kilo-Ohms, that is, nano-seconds for delay and pico-Farads for capacitance. The value specified should be greater than 0.

-macro *macro_name*

Defines the type of macro to use, and *macro_name* specifies the name of the macro. Once the appropriate location is

determined, PKS inserts the buffer in an available nearby site, cuts the existing wire, and alters the logical netlist with the newly inserted buffer and corresponding net. Wroute must then be used to make the local connections to the buffer.

net_name

Specifies the name of the net.

-optionalResizeTo *macro_name*

Resizes the driver of the net to the specified cell master if PKS cannot find an acceptable buffer location.

-value *int*

The net is split into n nets, where n is the value specified by the -value option.

do_xform_structure

```
do_xform_structure [-force] [-hierarchical] [-effort { low | medium | high }]
     [-flatten { on | auto | off }] [-priority { area | time }] [-no_partition]
```

Applies Boolean and algebraic algorithms and transformations to achieve logic optimization and logic structuring. These transformations are technology independent. This command is typically used prior to any technology dependent mapping or optimizations.

Options and Arguments

`-effort {low | medium | high}`
: Controls the CPU time spent during the timing optimization step.
Default: `medium`

`-flatten [on | off | auto]`
: Controls the operations employed for optimization of logic equations. If flatten option is set to `on`, the logic equations are flattened into a sum of products form before applying optimization. This generally helps in creating faster and optimal implementation of the logic, even though it may take up extra area (gates). For some circuits, this option may result in excessive run time or memory usage. If flatten option is set to `auto`, only certain flattening operations are performed; some of the time consuming steps are skipped. The flattening step may be disabled entirely by setting the option to `off`.
Default: `off`

`-force`
: Restarts optimization with all generic optimizations. In essence, the design is unplaced and unmapped and then reoptimized from scratch.

`-hierarchical`
: Performs mappings, unmappings, structuring, or removes redundancies on the modules in the downward path of the current module. It depends on the command being used. If this option is not given, then only the current module is affected by the command used.

`-no_partition`
: Prevents automatic partitioning of the design into smaller modules for optimization, irrespective of the size of the module.

Each module boundary should be retained as the partition to optimize.

`-priority {area | time}`
Sets area or timing minimization to be the highest priority of the technology independent optimization step. If you have a loosely constrained or unconstrained design, then the priority should be set to `area`.
Default: `time`

Related Information

`do_optimize`

do_xform_unmap

```
do_xform_unmap [-hierarchical] [-force]
     [list_of_instance_ids | list_of_instance_names]
```

Unmaps the netlist from the technology library. The result of the operation is that the netlist is back in the generic form and can be mapped again.

This command reverts a technology specific cell netlist to a generic cell netlist (ATL).

Options and Arguments

`-force`

Performs all optimization steps on a generic netlist only when used with `do_optimize`.
Default: Does not transform a previously mapped or optimized netlists. After applying the transformation, the netlist must be remapped.

When used with `do_xform_remove_redundancy`, unmaps a previously mapped netlist before applying the transformations.

When used with `do_xform_unmap`, unmaps cells with a `dont_modify` attribute, such as test cells and IO cells. These cells are not unmapped when using `do_xform_unmap`.

`-hierarchical`

Performs mappings, unmappings, structuring, or removes redundancies on the modules in the downward path of the current module. It depends on the command being used. If this option is not given, then only the current module is affected by the command used.

`list_of_instance_ids`

Specifies the IDs of the instances to be unmapped.

`list_of_instance_names`

Specifies the names of the instances to be unmapped.

Related Information

do_optimize

do_xform_map

dump_adb

```
dump_adb ac_shell_process_id
```

Dumps the .adb (Ambit Synthesis Database) file at the next most suitable time while allowing the run to continue. Using this command you can get a copy of the .adb file at any point during the optimization.

Once the command is invoked the following message appears in the console window and in the log file:

```
"Request received to dump ADB at date & time. Will save database at next suitable time in file current_module.adb.sig.num"
```

The *num* variable increments after every successful .adb file dump, starting from 1. This enables you to save different snapshots of the design and prevents the snapshots from being overwritten.

If the dump_adb command is invoked again before the database is written out, then the tool displays the following message in the console and in the log file:

```
"Received user request to dump ADB. Waiting for suitable time to dump ADB file"
```

When ac_shell dumps the .adb file, the following message is displayed to indicate that you can now use the saved adb file to perform other operations:

```
Database saved in file "top_module.adb.sig.num according to user request"
```

The command provides a snapshot of the database during a run.

Options and Arguments

ac_shell_process_id
: Specifies the process ID of an ac_shell.

eval_bottom_up

```
eval_bottom_up [-skip] [-incr] [-slack float] [-derive_context]
     [min_size leaf_cell_size] [-level level_number]
```

Traverses the design tree in a bottom-up fashion and pre-processes all modules to determine which ones meet the conditions specified by the options `-min_size`, `-slack`, and `-level`. Then it computes the time budgeting for these modules. The time budgeting will be done at once for all the pre-selected modules in the absence of the `-incr` option. Otherwise it is performed later. It performs another bottom-up tree traversing and visits each pre-selected module. If the `-incr` option is selected then it computes the time budget for each module as it visits them (incrementally). It sets the pointers `current_module` and `set top timing module`, to point to each module visited and then executes the series of `Tcl` commands contained in the command's argument. It places a `set_dont_modify` attribute on each instance's module after it is visited, and before it visits the next module. Once it reaches the top module, it clears all the `set_dont_modify` attributes for all modules.

Note: That the `eval_bottom_up` command works on both non-uniquified and uniquified netlists.

Options and Arguments

`-derive_context`
: This directs the `eval_bottom_up` command to perform a `do_derive_context` operation for all instances instead of a `do_time_budget` operation.

`-incr`
: Without this option, the `eval_bottom_up` command performs a time budgeting for all the instances in the design at the beginning of the operation. With this option, the time budgeting is done incrementally as each instance is being visited.

`-level` *`level_number`*
: This directs the `eval_bottom_up` command to start to execute its arguments from instances that are at a distance of `level_number` from the top cell (these are the bottom instances) to the top.

`-min_size` *`leaf_cell_size`*
: This directs the `eval_bottom_up` command to execute its arguments on instances that have a greater or equal amount of cell-instances than the `leaf_cell_size` number.

`-skip`

It prevents `eval_bottom_up` from performing a time budgeting operation for all the instances in the design.

`-slack` *`float`*

It performs a worst case setup slack timing analysis each module. It compares the result against the argument given with the option. If the value of the instance's worst case setup slack is less than the argument of the `-slack` option, then it executes the argument of the `eval_bottom_up` command.

Examples

- The following command performs a bottom-up evaluation without doing any time budgeting. It runs the `do_xform_buffer_tree` command at each visited instance.

```
> eval_bottom_up -skip {do_xform_buffer_tree}
```

- This following command performs time budgeting for all modules in the design and writes out constraints for each module. (output files follow the naming format *`module_name`*`_constraints.tcl` where the string *`module_name`* corresponds to the actual name of each module):

```
> eval_bottom_up {write_assertions [get_name [ get_current_module ] ]_constraints.tcl}
```

- This following command starts its bottom-up evaluation from those instances that have 3000 `cell_instances` or more. It performs incremental time budgeting. It only executes the list of arguments on those instances whose worst-case setup slack is less than 0 (instances that have negative slack):

```
> eval_bottom_up -incr -min_size 3000 -slack 0 {do_optimize -effort low
-no_design_rule}
```

- This following command starts its bottom-up evaluation from those instances located three levels below the top module. It performs incremental time budgeting. It only executes the commands contained in the Tcl file `fast_procs.tcl` on those instances whose slack is less than -1 ns:

```
> eval_bottom_up -incr -level 3 -slack -1 {source fast_procs.tcl}
```

- This following command performs a bottom-up evaluation from the bottom leaf instances and returns the name of the module associated with each instance:

```
> eval_bottom_up -skip {puts [get_name [get_current_module]]}
```

Related Information

`set_dont_modify`

find

```
find [-blackboxes] [-alias_only] [-bus] [-cellrefs] [-clocks] [-dont_modify]
     [-exact] [-exclude] [-full_path_name] [-glob | -regexp] [-hierarchical]
     [-inputs] [-instances] [-modules] [-nets] [-nocase] [-noclocks]
     [-of_cell_type cell_name] [-of_lib_type lib_name]
     [-of_pin_type { data | clock | output | reset | set }] [-outputs] [-pins]
     [-ports] [-registers] [-sop] [-techlib] [-top] [-scalar] name_list
```

Finds various design objects and prepares the list for other `ac_shell` commands. The design objects are found in the design database using glob style pattern matching, unless `-regexp` option is used. Multiple wildcard matching in a name is permitted.

Typically, the output of `find` command (list of objects found by this command) is an argument of another command that accepts a list of objects and performs some task. For example, all input ports of a module can be set to same arrival time as follows:

```
set_data_arrival_time 0.2 [find -ports -input] -clock master_ck
```

The search for finding objects is carried out in the current module. If the object name has hierarchy specified in it (for example `cnt99/*`), then the search is carried out at the appropriate hierarchy level.

Important

Every character after the "`/`" and up to the next space is considered escaped. Therefore, use a space with curly brackets if an escape character is used in the name with the `find` command. For example:

```
find -net -hier {mmio_o/\mmio_wr_data[2] }
```

Options and Arguments

`-blackboxes`
: Filters for `-cellref` and `-instances`. Return only objects that are blackboxes.

`-bus`
: Filters for nets, ports, and instance pins. Returns only bus components.

`-cellrefs`
: Searches for cell references.

`-clocks`
: Filters for `-pins` and `-ports`. Returns only clock pins or ports.

`-dont_modify`

Filters for all objects types. Returns objects with `dont_modify` flag set. Refer to `set_dont_modify`.

`-exact`

Searches only for exact name matches.

`-exclude` *`namelist`*

Excludes objects specified by *`namelist`*. This option works with any object type including cells. instances, net, modules, ports, pins, and techlibs.

`-full_path_name`

Returns the complete path name of the object, not the *`object_id`*. It returns the complete path of the nets, instances, and the instance pins from the current module. It returns the object name for all other object types.

`-hierarchical`

Searches the complete database hierarchically from current module.

`-inputs`

Filters for `-pin` and `-port`. Returns only input pins or ports.

`-instances`

Searches only for instances.

`-modules`

Searches for modules.

`-name_list`

Specifies the names of the objects to search. Pattern matching and regular expressions may be used to find many objects.

`-nets`

Searches only for net object types.

`-nocase`

Determines whether a case- insensitive search is performed. Use this switch in conjunction with the `-module` or the `-techlibs` switches to specify. It is a useful in searching for the VHDL modules.

`-noclocks`
: Filters for `-pins` and `-ports`. Returns only `nonclock` pins or ports.

`-of_cell_type` *`cell_name`*
: Filters for `-instances`. Returns only the instances of *`cell_name`*, which can be `*`.

`-of_pin_type {data | clock | output | reset | set}`
: Filters for `-pin`. Returns the respective pins for the sequential elements. This option allows you to apply path exceptions (false paths, multi-cycle paths) across reused modules.

`-outputs`
: Filters for `-pins` and `-ports`. Returns only output pins or ports.

`-pins`
: Searches only for instance pins.

`-ports`
: Searches only for ports.

`-regexp`
: Performs pattern matching using regular expression rules instead of glob style pattern matching by default.

`-registers`
: Filter for `-instances`. Returns only sequential instances.

`-sop`
: Filter for `-cellref` and `-instances`. Returns objects which are sum-of-products (SOP) logic.

`-scalar`
: Filter for nets, ports, and instance pins. Returns only the scalar components and filters out the bus components. Useful in the cases where there is a name collision between a scalar and bus component.

`-techlib`
: Searches for technology libraries.

`-top`

Filter for `-module`. Returns only the top level module.

Object Types and Filters

Valid object types and filters associated with those objects are shown in table 1-1. Only the applicable filters will be applied to the objects. If no filters are listed, each *`object_id`* that matches a name in the *`name_list`* is returned.

Table 1-1 Object Types and Filters

Object Type	Filter
instances	`blackbox`
	`dont_modify`
	`of_cell_type`
	`pla`
	`registers`
ports	`bus`
	`clocks`
	`input`
	`no_clocks`
	`output`
	`scalar`
nets	`bus`
	`dont_modify`
	`scalar`
cells	`dont_modify`
modules	`dont_modify`
	`nocase`
	`top`

Object Type	Filter
pins	`bus`
	`clocks`
	`input`
	`noclocks`
	`of_pin_type`
	`output`
	`scalar`
cellrefs	`blackboxes`
	`pla`

Glob Style Pattern Matching

The following symbols are used for wildcard matching:

`*`
: Matches `0` or more occurrence of any character.

?
: Matches any single character.

`[a,b,c]`
: Matches `a` or `b` or `c`.

Regular Expression Pattern Matching

Refer to your Tcl documentation for more information about regular expressions and pattern matching.

The following symbols are used to match regular expressions:

`.`
: Any character.

`*`
: Zero or more occurrence of preceding character or regular expression.

`+`
: One or more occurrence of preceding character or regular expression.

`^`
: Matches at the start of a string.

`$`
: Matches at the end of a string.

`\x`
: Matches character `x`.

`[chars]`
: Matches any character from the set.

`regexp1 | regexp2`
: Matches anything that matches `regexp1` or `regexp2`.

Related Information

`set_current_module`

Examples

- The following commands find all input ports:

```
> find -ports -input *
> find -port -input -regexp
```

- The following command finds all inout ports:

```
> find -ports -input -output *
```

- The following command finds all technology libraries that have LCA prefix:

```
> find -techlibs LCA*
```

- The following command finds the R3 register in all instances:

```
> find -instances -registers R3
```

- The following command finds all ports and all modules:

```
> find -port -mod *
```

- The following command finds all input ports and all modules:

  ```
  > find -port -mod -input *
  ```

  Note: The filter `input` applies only to ports and not modules

- The following command finds the top level module:

  ```
  > find -module -top *
  ```

- The following command finds instances that match `add1` or `add2`:

  ```
  > find -instance {add[1.2])
  ```

- The following command finds all input ports with name beginning with `scan`*N* and ending in _d:

  ```
  > find -input -port -regexp {^scan[0-9]+.*_d$}
  ```

get_area

`get_area [-cell {combinational | sequential | both} | -net] [-module` *`module`*`]`

Queries the area of the given mapped module or the current module, and downwards in the hierarchy. It gets the area for a design module.

Note: If a LEF is loaded in the current version, `get_area` will be based on that, not on the areas in the cell library. To `get_area` in an earlier version, set the global `use_lef_area` to `false`.

Unmapped (generic) logic does not count towards the area of a design. There are options for cell area and net area.

Options and Arguments

`-cell`

Returns the combinational or sequential cell area only, or both.
Default: Both

`-net`

Returns the net area as determined by the appropriate wireload model.

`-module` *`module`*

Returns the total area of a module if neither `-cell` nor `-net` is specified.
Default: The *`current_module`*.

Examples

- The following command returns the total area of the current module and all its children:

  ```
  > get_area
  ```

- The following command returns only the sequential cell area in the current module and its children:

  ```
  > get_area -cell sequential
  ```

- The following command returns only the net area in the current module and its children:

  ```
  > get_area -net
  ```

- The following command returns the total area of module `buffer2` and its children:

  ```
  > get_area -module buffer2
  ```

Related Information

`get_module_worst_slack`

`report_area`

`set_current_module`

get_attribute

`get_attribute` *`object_id attribute_name`*

Returns the value set for the specified attribute on the specified object. You can query tool- and user-defined attributes.

Options and Arguments

`attribute_name`

Specifies the name of the attribute to query. Refer to the `set_attribute` command's set_attribute on page 257 for a list of tool-defined attributes.

`object_id`

Specifies the object identifier of the module, instance, cellref, net, port, or pin you want to query.

Examples

- The following command returns the value of the `flatten` attribute for the module `A`:

```
> get_attribute [find -module A] flatten
```

- The following command returns the value of the `my_attribute` attribute on the net object `my_net`:

```
> get_attribute [find -net my_net] my_attribute
```

Related Information

`delete_attribute`

`get_info`

`set_attribute`

`set_global`

get_build_id

```
get_build_id
```

Returns the build id of the BG/PKS version. This command must be executed within the shell.

Related Information

get_version

get_buswidth

`get_buswidth` *`object_id`*

Returns the width of the specified bus object.

Options and Arguments

`object_id`

Specifies the object identifier associated with a bus.

Example

This command returns the width of the object `73540`:

```
> get_buswidth 73540
4
```

get_cell_area

`get_cell_area` *`cell_id`*

Returns the area of the specified cell.

Options and Arguments

`cell_id`

Specifies the object identifier associated with a cell.

Example

The following command returns the cell area of cell `60979`:

```
> get_cell_area 60979
16.000000
```

get_crosstalk_threshold

```
get_crosstalk_threshold [-recalculate]
```

Returns any existing crosstalk threshold stored with the design.

If no crosstalk threshold is stored, it returns the three times the average slew of all the nets present in the design.

Options and Arguments

`-recalculate`

Returns three times the average slew of all the nets present in the design, even if there is an existing crosstalk threshold.

Related Information

`do_optimize`

`do_xform_prevent_crosstalk`

`get_crosstalk_tolerance`

`report_crosstalk_violations`

get_crosstalk_tolerance

```
get_crosstalk_tolerance
```

Returns any existing crosstalk tolerance stored with the design.

If no crosstalk tolerance is stored, it returns 0.

Related Information

do_optimize

do_xform_prevent_crosstalk

get_crosstalk_threshold

report_crosstalk_violations

get_current_instance

```
get_current_instance
```

Returns the object ID of the instance which has been previously designated as the current instance by the command `set_current_instance`.

Related Information

`set_current_instance`

get_current_module

```
get_current_module
```

Returns the object ID of the module which has been designated as the current module previously by the command `set_current_module`.

Related Information

`set_current_module`

get_equivalent_cells

```
get_equivalent_cells [-exclude_self] cell_id
```

Returns a list of names of library cells that have the same functionality and pin names in non-decreasing order of cell area. Any one of the returned cell names can be used in a `do_rebind` call on an instance bound to *`cellid`*.

Options and Arguments

`-exclude_self`
: Excludes the name of *`cell_id`* from the list.

Related Information

`do_rebind`

get_global

`get_global [-default] global_variable_name`

Returns the current value for the specified global variable. The value of any global variable can be obtained using `get_global` command. The `set_global` command has a one-to-one correspondence with the `get_global` and `reset_global` commands. Every variable defined in `set_global` also applies to `get_global` and `reset_global`.

Options and Arguments

`-default global_variable_name`
: Returns the default value for the specified global variable.

Related Information

`set_global`

get_hdl_file

`get_hdl_file` *`module_name`*

Returns the file name corresponding to the module. This command can be invoked right after reading the HDL files into BuildGates Synthesis, without having to first generate a generic netlist using `do_build_generic`.

Options and Arguments

`module_name`

Specifies the name of the module.

Example

The following commands read in a Verilog module `TOPV` defined in a file `design.v` and a VHDL module `TOPVH` defined in a file `design.vhdl`. After the HDL files have been read into BuildGates Synthesis, the `get_hdl_file` command determines what file the modules belong to:

```
> read_verilog design.v
> get_hdl_file TOPV
design.v

> read_vhdl design.vhdl
> get_hdl_file TOPVH
design.vhdl
```

get_hdl_hierarchy

`get_hdl_hierarchy [`*`module_name`*`]`

Displays the design hierarchy where, for each design, it lists the names of the designs that are instantiated within, and whether the instantiations are parameterized (using parameters in Verilog or generics in VHDL) or not.

This command can be used to investigate the design hierarchy right after reading the HDL files into BuildGates Synthesis, without having to first generate a generic netlist using `do_build_generic`. This is very useful when design data is often organized into tens or even hundreds of HDL files.

If `get_hdl_hierarchy` is invoked without any arguments, it will return a TCL list containing information for each module that had been read using the `read_vhdl` or `read_verilog` command.

Options and Arguments

`module_name`

Specifies the name of the hierarchical module.

Examples

The subsequent examples use the following VHDL design that consists of three modules: `TOP`, `BOT`, and `BOTG`. This design is in a VHDL file called `design.vhd` and has been read in using the `read_vhdl` command.

```
entity BOTG is
   generic (WIDTH : natural := 1);
   port (O: out bit_vector(WIDTH-1 downto 0));
end;
architecture A of BOTG is
begin
   O <= (others => '1');
end;
entity BOT is
   port (O: out integer);
end;
architecture A of BOT is
begin
   O <= 25;
end;
```

```
entity TOP is
   port (O8: out bit_vector(7 downto 0);
         O1: out integer);
end;
architecture A of TOP is
begin
  I1 : entity work.BOT  port map (O1);
  I8 : entity work.BOTG generic map (8) port map (O8);
end;
```

- The following command returns the three modules in the HDL design pool: TOP, BOT and BOTG. It also indicates that while BOT and BOTG do not instantiate any other designs, TOP instantiates both BOT (n represents a non parameterized instantiation) and BOTG ('p' indicates a parameterized instantiation, since design BOTG contains generics):

  ```
  > get_hdl_hierarchy
  {TOP {{BOT n} {BOTG p}}}  {BOT {}}  {BOTG {}}
  ```

- The following command returns the hierarchy for a specific design (TOP):

  ```
  > get_hdl_hierarchy TOP
  {TOP {{BOT n} {BOTG p}}}
  ```

get_hdl_top_level

```
get_hdl_top_level
```

Displays a list containing the names of all top level designs, that is designs that are not instantiated by any other design.

This command can be invoked right after reading the HDL files into BuildGates Synthesis, without having to first generate a generic netlist using `do_build_generic`.

Example

The following example uses this VHDL design in file `design.vhdl`:

```
entity BOT1 is ....
architecture A of BOT1 is ....

entity BOT2 is ....
architecture A of BOT2 is ....

entity MYDES is ....
architecture A of MYDES is
begin
  I1 : entity work.BOT1  port map ...
  I8 : entity work.BOT2 port map ...
end;
```

This set of commands displays the-top level module in the design:

```
> read_vhdl design.vhdl
> get_hdl_top_level
MYDES
```

get_hdl_type

`get_hdl_type` *`module_name`*

Given a specific design name, returns the language (VHDL or Verilog) in which that specific design was represented.

This command can be invoked right after reading the HDL files into BuildGates Synthesis, without having to first generate a generic netlist using `do_build_generic`.

Options and Arguments

`module_name`
The name of the hierarchical module.

Example

The following commands read in a Verilog module `TOPV` defined in a file `design.v` and a VHDL module `TOPVH` defined in a file `design.vhdl`. After the HDL files have been read into BuildGates Synthesis, the `get_hdl_type` command determines what language the various modules were represented in:

```
> read_verilog design.v
> read_vhdl design.vhdl
> get_hdl_hierarchy
{TOPV {}} {TOPVH {}}

> get_hdl_type TOPVH
VHDL
> get_hdl_type TOPV
Verilog
```

get_info

`get_info` *`object_id`* [*`field`*]

Returns the Tcl data structure associated with the *`object_id`*, or the specific *`field`* of that *`object_id`*.
Default: Returns all available fields. It can be used to show attributes.

Options and Arguments

`field`

Specifies the field associated with the returned object ID.

`object_id`

Specifies only one *`object_id`* is accepted.

Examples

- The following commands return the data structure of `FD1`:

```
> find -cellref FD1
35331


> get_info 35331
{objecttype cellref} {name FD1} {dont_utilize false} {dont_modify false}
{module 17} {pins {35350 35366 35382 35398}}
```

- The following command passes the value returned by the `find` command to the `get_info` command, and returns the requested information:

```
> get_info [find -module state1]
{objecttype module} {{name state1} {dont_modify false} views {72690}}
```

- The following command returns the value of the `dont_modify` attribute of `state1:`

```
> get_info [find -module state1] dont_modify
false
```

- The following command returns the *`object_id`* of `xf_ncnt[1]`:

```
> find -net {xf_ncnt[1]}
99557
```

- The following command returns the data structure of `xf_ncnt[1]`:

```
> get_info [find -net {xf_ncnt[1]}]
{objecttype net} {name {xf_ncnt[1]}} {dont_modify false} {bus 99924}
{type 85799} {module 84385} {connections {102790 91638}}
```

- The following commands return the values of the `connections` and `bus` fields:

```
> get_info [find -net {xf_ncnt[1]}] connections
102790 91638
> get_info [find -net {xf_ncnt[1]}] bus
99924
```

- The following command returns the data structure of the bus field of `xf_ncnt[1]`:

```
get_info [get_info [find -net {xf_ncnt[1]}] bus]
=> {objecttype bus} {name xf_ncnt} {module 84385} {components {92789 92805
   92773 92757 99237 99269 99365 99397 99461 99493 99557 99893}}
```

- The following command creates a list of views of module `count5`, and returns a list of all port identifiers for the module:

```
> set view_list [get_info [find -module count5] views]

> foreach view $view_list {
>   set port_list [get_info $view ports]
> }
```

Related Information

`find`

`get attribute`

`set attribute`

get_message_count

```
get_message_count [message_ID] { error | warning | info | info_msg }
```

Returns the current message count of either the specified *message_ID* or the specified message type.

Options and Arguments

`error`
: Gets the count for message type `error`.

`info`
: Gets the count for message type `info`.

`info_msg`
: Gets the count for message type `info_msg`.

`message_ID`
: Specifies the ID of the message whose count is returned.

`warning`
: Gets the count for message type `warning`.

Related Information

`set_message_count`

`set_message_verbosity`

get_message_verbosity

`get_message_verbosity` *`message_ID`*

Returns the current message verbosity.

Options and Arguments

`message_ID`

Specifies the ID of the message whose verbosity level will be retrieved.

Related Information

`set_global message_verbosity_level`

get_names

`get_names` *`object_id`*

Returns the names of specified objects in the design.

Options and Arguments

`object_id`

Reports the *`object_id`* of the object whose name is specified by this command. The *`object_id`* is typically obtained using the find command. If the *`object_id`* represents a single object, a corresponding name is returned. If the *`object_id`* represents a list of objects, a list of names in the order of objects in the *`object_id`* list is returned.

Example

The following command returns the name of all input ports of the current module:

```
> get_names [find -input -ports *]
```

Related Information

`find`

get_net

```
get_net [-full_path_name] [-min_fanout integer] [-max_fanout integer]
     [list_of_net_path_or_id] [netname]
```

Returns net information at the current level of hierarchy. It is useful when finding nets with a pattern matching (wildcard, for example) and then filtering out certain nets based on other options of the command (that is -`min_fanout` or -`max_fanout`).
Default: Returns the net IDs at the current level of hierarchy and returns full path name of the net if `-full_path_name` is specified.

Options and Arguments

`-full_path_name`
: Returns full path name of net, instead of net ID

`list_of_net_path_or_id`
: Specifies the net path or ID. Can be a pattern or regular expression. If the argument is not given, it returns all nets under the hierarchy of current module.

`-max_fanout` *`integer`*
: Returns net whose number of fanouts is not greater than the maximum number specified in *`integer`*.

`-min_fanout` *`integer`*
: Returns net whose number of fanouts is not less than the minimum number specified in *`integer`*.

`netname`
: Returns the ID of the specified net name.

Examples

- The following command returns the net ID of `i_284`:

```
> get_net i_284
94341
```

- The following command returns all nets under the hierarchy of the current module with names beginning with `r` and having at least 300 fanouts. This command can be used to check if there are any high fanout nets:

```
> get_net -min_fanout 300 r*
r_245 r_250
```

- The following command returns the full path name of all nets with fanouts less than or equal to 10:

```
> get_net -full_path_name -max_fanout 10
I311/I210/n_230
```

- The following command returns the full path name of all nets with fanouts greater than 100:

```
> get_net -full_path_name -min_fanout 100
I311/I210/n_310 I311/I210/n_312
```

Related Information

`report net`

get_parent_instances

```
get_parent_instances
```

Returns the parent instances of the current module.

Nothing is returned if the current module is the top level module.

Example

The following command returns the IDs of the four parent instances of the current module:

```
> get_parent_instances
61030 61046 61062 61078
```

Related Information

`all_parents`

get_state_of_design

```
get_state_of_design
```

Returns the state of the design. Lets you know if the design is a generic netlist, or if it is mapped, placed, or if a clock tree has been inserted. The command will return one of the following strings: `generic`, `mapped`, `placed`, or `clock_tree_inserted`.

A design is considered mapped if all primary instances are mapped to a technology library. A design is considered placed if all instances are mapped and placed. A design is considered `clock_tree_inserted` if all instances are mapped and placed, and the clock mode is set to propagated.

Note: No check is made if there is actually a buffer tree present on any clock network.

The do_optimize command will use this information to determine what optimizations to run.

Note: ATL_LOGIC0/1 instances are excluded from the checks above. For BG(X) a design can only be `generic` or `mapped`.

Example

The following commands show the results of the `get_state_of_design` command after various commands:

```
> get_state_of_design
generic

> do_optimize -stop_after mapping...

> get_state_of_design
mapped

> do_optimize -stop_after placement...

> get_state_of_design
placed

> do_optimize -stop_after clock_tree_insertion...

> get_state_of_design
clock_tree_inserted
```

Related Information

`do_optimize`

get_tempfilename

```
get_tempfilename [directory_name] [prefix]
```

Returns the temporary file name from the system's default temporary directory. You can specify the directory name and prefix, if desired.

Options and Arguments

`directory_name`
: Specifies the name of the directory in which to create for the temporary file.
Default: Directory name is `/var/tmp/`.

`prefix`
: Specifies the prefix added to the file name.

Example

The following commands show the results of `get_tempfilename` when given different directory names and prefixes:

```
> get_tempfilename
/var/tmp/CAAa006T3

> get_tempfilename /tmp
/tmp/DAAa006T3

> get_tempfilename /tmp Run1_
/tmp/Run1_FAAa006T3
```

get_version

```
get_version
```

Returns the version of BG/PKS shell being executed.

Related Information

`get_build_id`

help

`help [`*`command`* `|` *`keyword`* `|` *`message_id`*`]`

Provides online help for all `ac_shell` commands. If no arguments are given to the `help` command, the usage syntax is displayed. This is also true for all `ac_shell` commands: When a command expects some arguments that are not provided, or incorrect arguments are provided, the usage syntax is displayed.

Options and Arguments

`command`

Displays the command's syntax.

`keyword`

Displays a list of all commands related to the specified keyword. The keyword can be any word, including partial match to one or more commands.

`message_id`

Displays an explanation of the message. Most of the messages—errors, warnings, or informative messages, are displayed in terse form, with a `message-id` associated with it. Requesting help with this `message-id` provides detailed information, including possible problems and suggestions for how to remove the problem.

highlight

`highlight` *`list_of_object_ids`* `[-center] [-verbose]`

Highlights schematic instances, nets, and ports, given a list of object IDs.

Note: This command is only useful when run from the GUI.

Options and Arguments

`-center`

Centers schematic on the last object in *`list_of_object_ids`*.

`-verbose`

Prints the name and object ID of each object that is being highlighted.

issue_message

`issue_message [-type {info | error | warning}] *message*`

Issues user messages. These messages are logged in a log file. There may also be no type as an option for this command and the message ID for issuing such an information message is `user-300`.

Options and Arguments

`-error`
: Issues the message ID for this message type: `user-100`.

`-info`
: Issues the message ID for this message type: `user-400`.

`-warning`
: Issues the message ID for this message type: `user-200`.

Example

The following commands show the some possible results:

```
> issue_message -type error "message"
ERROR: message<user-100>

> issue_message "message"
message <user-300>

> issue_message -type info "message"
INFO: message <user-400>

> issue_message -type warning "message"
WARNING: message <user-200>
```

limit

`limit {`*`cpu_seconds`* `|` `-cycles` *`number`*`}` *`tcl_command`*

Executes the specified Tcl command and issues a user interrupt after *`cpu_seconds`* have elapsed if the command did not finish by that time.

This command can be used to limit the time spent in a given optimization command. Actual termination of the *`tcl_command`* is at the discretion of the application and is analogous to issuing a user interrupt (`^C`).

A `0` is returned if the Tcl command completes without exceeding the limit. If the limit was exceeded, limit will return a positive number representing the number of cycles executed. This number can be used to reproduce the run using the `-cycles` *`number`* option.

Note: CPU times are an approximation made by the operating system. Any script based on limiting the CPU time can result in non-reproducible results. To reproduce a CPU-time-limited run, use the option `-cycles` *`number`*, with the number returned by the run you wish to reproduce.

Specifying `-cycles 0` is the same as specifying no limit.

Options and Arguments

`cpu_seconds`
: Specifies the approximate number of CPU seconds.

`-cycles` *`number`*
: Specifies the number of CPU cycles.

`tcl_command`
: Specifies a Tcl command.

Example

The following script automatically replays a previous CPU-time-limited run:

```
if { $replay } { source $replay_file }
else { set fp [open $replay_file w] }
...
set cmd1 "do_optimize -effort high"
if { $replay } {
     limit -cycles $nr1 $cmd1
} else {
```

```
     set cycles [limit 100 $cmd1]
     puts $fp "set nr1 $cycles"
}
...
if { $replay == 0 } { close $fp }
```

quit

```
quit
```

Exits `ac_shell` process.

read_adb

```
read_adb filename [-module module_name]  [-echo_assertions]
     [-no_assertions] [-overwrite] [-restore_globals]
```

Reads a previously written Ambit Synthesis database file (ADB) and rebuilds the database for further synthesis and analysis of the netlist. The ADB file is a binary file.

The synthesis data stored by `ac_shell` in a database can be written out in the ADB format at any stage during the synthesis process using the `write_adb` command.

If the module (design) in the ADB file contains references to technology cells, the technology library must be loaded prior to loading the design from the ADB file.

Options and Arguments

`filename`
: Specifies the `.adb` file to be read in to `ac_shell`.

`-echo_assertions`
: Echoes the timing assertions in `read_adb`, regardless of the `set_global echo_commands`.

`-module` *`module_name`*
: Name of the module to be loaded from the entire ADB file. An adb file can contain several modules. Modules from the file can be selectively loaded so that `ac_shell` database does not occupy unnecessary memory space.

`-no_assertions`
: Causes the assertions in the ADB being read in to be ignored. *Default*: Writes assertions into all ADBs and to apply all the assertions being read in.

  Note: Using this option will cause placement information to be ignored.

`-overwrite`
: Overwrites the existing module if the module is present in the database, otherwise the existing module is automatically renamed maintaining the module's link.

`-restore_globals`
: Reads *all* of the globals back into the current design. By default

none of the globals are read back through `read_adb`. When you restore globals, globals with the same names are changed from the global's current setting. A message displays showing the globals that have been changed. This message only displays when reading an .adb file.

If you use the `read_adb` command without the `-restore_globals` argument, a warning message is printed out for each global that has a different value (in the current session) than the global value stored in the .adb file.

WARNING: Use caution when using the `-restore_globals` option. Using this option will also restore the logfile and cmdfile, and will immediately start overwriting if such files already existed from a previous run. You can either move the original logfile (and cmdfile) away before loading the adb (or equivalently run the bg_shell in a different directory then before).

Related Information

`read_alf`

`read_verilog`

`read_vhdl`

`write_adb`

read_edif

`read_edif` *`file_name`*

Reads an EDIF v2.0.0 file.

Options and Arguments

`file_name`

Specifies the name of the EDIF input file.

Related Information

`set_global edifin_*`

`write_edif`

Refer to the *EDIF Interface* chapter in the *HDL Modeling for BuildGates Synthesis*.

read_symbol

`read_symbol` *`filename`*

Reads `.sym` symbol libraries. It sets the schematic symbol library search path to *`filename`*. If a current list of files is loaded (from a previous `read_symbol` or `read_symbol_update` command) the list is set to *`filename`*.

Note: This command enables you to read Synopsys `.slib` symbol libraries. However, the `.slib` symbol library must first be translated to EDIF format (`.edif`) in the Synopsys tool, then converted to the `.sym` format using the `edifconv` program in BuildGates. BuildGates only supports the conversion of `.edif` symbol libraries, not `.slib` symbol libraries.

The schematic symbol libraries are searched in the following order:

1. Specified directory in the *Schematic Preferences* dialog box from the BuildGates and PKS GUI *View* menu.
2. Current directory
3. Specified directory pointed to by AMBIT_AMVIEW_SYMLIB

The symbol library file name, if not specified with the `read_symbol` command the `read_symbol_update` command, or in the GUI *Schematic Preferences* dialog box, defaults to *`technology_name.sym`*.

Options and Arguments

`filename`

Specifies the name of the symbol library (`.sym`) to add to the schematic symbol library search list.

Example

See `read symbol update`.

Related Information

`read_symbol_update`

read_symbol_update

`read_symbol_update` *`filename`*

Appends *`filename`* to the schematic symbol library search list. It provides support for loading multiple symbol library files into the schematic viewer.

The schematic symbol libraries are searched in the following order:

1. Specified directory in the *Schematic Preferences* dialog box from the BuildGates and PKS GUI *View* menu.
2. Current directory
3. Specified directory pointed to by AMBIT_AMVIEW_SYMLIB

The symbol library file name, if not specified with the `read_symbol` command the `read_symbol_update` command, or in the GUI *Schematic Preferences* dialog box, defaults to *`technology_name.sym`*.

Options and Arguments

`filename`

The name of the file to add to the schematic symbol library search list.

Examples

The following command loads schematic symbol libraries; search order is the order in which the libraries are read in:

```
> set SYM_lib_dir ${root_dir}/LIBRARY/SYMBOL
> read_symbol $SYM_lib_dir/IBM_SA12E.sym
> read_symbol_update $SYM_lib_dir/IBM_SA12E_BC.sym
> read_symbol_update $SYM_lib_dir/IBM_SA12E_GA.sym
> read_symbol_update $SYM_lib_dir/IBM_SA12E_GROW.sym
> read_symbol_update $SYM_lib_dir/IBM_SA12E_IO.sym
> read_symbol_update $SYM_lib_dir/IBM_SA12E_SC.sym
```

Related Information

`read_symbol`

read_verilog

`read_verilog [-aware_library` *`aware_libname`*`] [-structural]` *`verilog_filenames`*

Reads in Verilog design files to `ac_shell`. When the `-aware_library` option is used, the verilog modules are stored as components in the specified AmbitWare library. For more details on AmbitWare libraries, refer to the *Datapath Option of Ambit BuildGates Synthesis and Cadence PKS*.

You can read in a compressed Verilog file in GNU zip format if the specified input file ends in the .gz suffix. No special option is needed to read a gzipped file. For example: `read_verilog` *`verilog_filename`*`.gz`. However, you cannot create a gzipped file with the `read_verilog` command.

The `-structural` argument does not support full RTL. It only handles the structural subset of Verilog, that is, module and gate instances, concurrent assignment statements, and simple expressions (references to nets, bit-selects and part-selects of nets, concatenations of nets, and the unary ~ operator).It is not necessary to run `do_build_generic` after using the `-structural` argument with the `read_verilog` command.

Options and Arguments

`-aware_library` *`aware_libname`*

Reads each component file, checks the syntax, and generates the corresponding binary dump file (`*.bd`) in the specified AmbitWare library. Adds an entry into the existing index file or creates a new index file in the aware library.

`-structural`

Creates a netlist directly for an input file containing only structural constructs, that is, module and gate instances, concurrent assignments, and simple expressions (nets, bit-selects, part-selects of nets, concatenation of nets, and unary operators). Allows a quick reading of a Verilog netlist. Reports an error for any non-structural construct, such as a `reg` or an `always` block. Structural modules in the input files are converted to their netlist representation while non-structural modules are rejected. There is no need to issue the `do_build_generic` command after reading in the netlist with the `-structural` option.

`verilog_filenames`

Specifies the Verilog design file or files to read into `ac_shell`. Multiple file names are separated by blank spaces.

Attributes

`buscomp_generator`

`hdl_verilog_vpp_arg`

`instance_generator`

`net_generator`

Database Impact

The database contains parse tree of Verilog source when all files are read in successfully.

Examples

- The following command loads the Verilog file `design.v` into the `ac_shell`:

  ```
  > read_verilog design.v
  ```

- The following command reads three Verilog designs into the AmbitWare library `ZM122200`:

  ```
  > read_verilog -aware_library ZM122200 zm_des1.v zm_des2.v zm_des3.
  ```

- The following command reads `netlist.v` and creates a netlist directly:

  ```
  > read_verilog -structural netlist.v
  ```

  Using the `-structural` argument fully replaces the following commands:

  ```
  > read_verilog netlist.v
  > do_build_generic
  ```

- The following command loads both design files and then automatically decompresses the gzipped file:

  ```
  > read_verilog sop.v addr.v.gz
  ```

Related Information

`do_build_generic`

`read_alf`

`read_library_update`

`read_symbol`

report_aware_library

set_aware_library

read_vhdl

`read_vhdl [-aware_library` *`aware_libname`*`] | [-library` *`libname`*`]` *`vhdl_filenames`*

Reads in VHDL design files to `ac_shell`. When the `-aware_library` option is used, the VHDL modules are stored as components in the specified AmbitWare library. For more details on AmbitWare libraries, refer to *Cadence Datapath Synthesis Option of BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

You can read in a compressed VHDL file in GNU zip format if the specified input file ends in the .gz suffix. No special option is needed to read a gzipped file. For example: `read_vhdl` *`vhdl_filename`*`.gz`. However, you cannot create a gzipped file with the `read_vhdl` command.

This command analyzes the VHDL files, performs basic synthesis subset checking, and dumps the intermediate representation to the directory associated with the specified aware or VHDL library or by default into the VHDL library WORK, if no library is specified.

If the specified library has been explicitly created using the `set_vhdl_library` command, then the intermediate representation of the VHDL design units is written to the corresponding directory. Otherwise, a temporary directory will be created for storing all design units analyzed into a specific library.

VHDL design units (that is, entities, architectures, packages) can be read in any order with the following restrictions:

- Packages must be read before any other units that refer to them.
- Entities must be read before any of their architectures.

There is no restriction on the order in which entities are read for synthesis.

The following packages are pre-compiled and do not need to be analyzed:

```
std.standard
std.textio
ieee.std_logic_1164
ieee.numeric_bit
ieee.numeric_std
```

To use this command, a design library must already exist.

Options and Arguments

`-aware_library` *`aware_libname`*

Reads each component file, checks the syntax, and generates the corresponding binary dump file (`*.bd`) in the specified aware library. Adds an entry into the existing index file or creates a new index file in the AmbitWare library.

`-library` *`libname`*

Specifies the name of the VHDL library into which the VHDL files will be analyzed. Multiple file names are separated by blank spaces.

`vhdl_filename`

Reads in VHDL design file or files. If no library file is specified, the VHDL files will be analyzed into the VHDL library WORK by default. Multiple file names are separated by blank spaces.

Attributes

`set_global buscomp_generator`

`set_global hdl_vhdl_case`

`set_global hdl_vhdl_preferred_architecture`

`set_global hdl_vhdl_read_version`

`set_global hdl_vhdl_reuse_units`

Examples

- The following command analyzes file `pack.vhd` into a library `MYLIB`:

```
> read_vhdl -library MYLIB pack.vhd
```

- The following command analyses the file `use.vhd` into library `WORK`:

```
> read_vhdl use.vhd
```

- The following command reads three VHDL designs into the AmbitWare library `ZM122200`:

```
> read_vhdl -aware_library ZM122200 zm_des1.vhd zm_des2.vhd zm_des3.vhd
```

- The following command loads both design files and then automatically decompresses the gzipped file:

```
> read_vhdl sop.vhd addr.vhd.gz
```

Related Information

report_aware_library

report_vhdl_library

set_vhdl_library

set_aware_library

record_macro

`record_macro` *`filename`*

Starts the logging of interactive commands into the specified file name. Recording stops when you exit `ac_shell`. The file can be “sourced” to recreate the session.

Options and Arguments

`filename`

Specifies the name of the directory and file in which to record the logged commands.

report_area

```
report_area [-summary] [-block] [-hierarchical] [-cells] [{ > | >> } filename]
     [-include_hard_macros]
```

Generates a report on the area of the netlist. The information contained in the report is dependent on the options used with the command.

Note: If a LEF is loaded in the current release, `get_area` will be based on that, not on the areas in the cell library. To `get_area` in an earlier version, set the global `use_lef_area` to `false`.

Options and Arguments

`{ > | >> }` *`filename`*

Specifies the name of the file in which report will be saved. If *`filename`* is not specified, the report is displayed on the standard output. Note that *`filename`* must be the last argument in the list.

`-block`

Provides a block summary. See the table below.

Block report for module `am`	**Current Module**	**Cumulative**
Number of combination instances	3	812
Number of non-combinational instances	0	438
Number of hierarchal instances	2	10
Number of blackbox instances	0	1
Total number of instances	5	1261
Area of combinational cells	19.00	4181.00
Area of non-combinational cells	0.00	9970.00
Total cell area	19.00	14151.00
Number of nets	471	1721
Area of nets	0.00	0.00
Total area	19.00	14151.00

`-cells`

Presents the area report on every cell (block) in the current module. The cell report includes summary report for current module, summary report of cumulative design hierarchy, name and count of cells used, type of cells used, area for each cell, total cell area, net area and total design area. If no options are specified, then a summary area report will be generated.

`-hierarchical`

Presents the area report for the entire hierarchy below the current module. If this option is not used, the data is presented only for the current module.

`-include_hard_macros`

Presents the area report for hard macros, as well as the area of standard cells. The default is to just include the area of standard cells.

`-summary`

Presents the area report in summary form. The summary information contains name of the current module, wire load model used, cell area, net area, and total area.

Examples

- The following command generates a detailed hierarchical report on each cell and stores the report in file `area.rpt`:

  ```
  > report_area -hierarchy -cells area.rpt
  ```

- The following command writes the report timing data to the file `my_file.rpt`:

  ```
  > report_timing > my_file.rpt
  ```

- The following command writes the area report data to the file `my_file.rpt`, overwriting the timing data:

  ```
  > report_area > my_file.rpt
  ```

- The following commands write timing and area information to `my_file.rpt`:

  ```
  > report_timing >  my_file.rpt
  > report_area   >> my_file.rpt
  ```

- The following examples show reports where the area is taken from a TLF (or ALF) file and a LEF file. The numbers in each *Cell Area* column represent the area in square microns (TLF) and in terms of the number of SITES used (LEF). You can convert the LEF area units to square microns by multiplying the SITE SIZE X* Y.

```
Source of Area : Timing Library
+-------------------------------------------------------------------------+
|  Module     |        Wireload       |  CellArea  | Net Area | Total Area |
|-------------+-----------------------+------------+----------+------------|
|CSR1KT_viter | TSMC256K_Conservative | 1114025.68 |   0.00   | 1114025.68
...
Source of Area : LEF
+-------------------------------------------------------------------------+
|  Module     |       Wireload        |  Cell Area | Net Area | Total Area |
|-------------+-----------------------+------------+----------+------------|
|CSR1KT_viter | TSMC256K_Conservative |  656313.00 |   0.00   | 656313.00  |
...
```

Related Information

report_design_rule_violations

report_fsm

report_hierarchy

report_timing

set_current_module

report_aware_library

```
report_aware_library [-library library_name] [-summary filename]
     [-component pattern]
```

Generates library, summary, and components that meet the *pattern* reports.

Options and Arguments

`-component` *pattern*
: Selects a component report of the components specified by *pattern*.

`-library` *library_name*
: Selects a report of the aware library *library_name*.

`-summary` *filename*
: Selects a summary report, storing the summary in *filename*.

Example

The following command generates a report on the library `AWLOGIC`.

```
> report_aware_library -library AWLOGIC
```

Related Information

delete_aware_component

set_aware_component_property

set_aware_library

report_crosstalk_violations

```
report_crosstalk_violations [-threshold num] [-reset_threshold] [-tolerance num]
     [-ignore_dont_modify_nets] [-ignore_clknets] [-verbose]
```

Uses the threshold from previous crosstalk prevention fixes and stored with the design. If a threshold does not exist, this command will calculate an average transition time for the threshold automatically. See the do_xform_prevent_crosstalk command for further information.
Default: Only reports nets with violations, including clock nets and dont_modify nets. These nets are flagged accordingly in the report similar to the report_design_rule_violations command.

Note: The report_crosstalk_violations command will NOT store this calculated threshold with the design. The -threshold and -reset_threshold options result in reports based on the specified or a recalculated threshold, respectively, but these options do NOT overwrite any existing threshold stored with the design or result in a new threshold being stored with the design.

Options and Arguments

-ignore_clknets
: Excludes clock nets with crosstalk violations from the report.

-ignore_dont_modify_nets
: Excludes dont_modify nets with crosstalk violations from the report.

-reset threshold
: Recalculates a threshold for the report.

-threshold num
: Specifies a threshold for the report.

-tolerance num
: Specifies a number, which together with the calculated or specified threshold, defines a range within that crosstalk violations will be tolerated and need not be fixed.
Default: 0

-verbose
: Includes nets with no crosstalk violations in the report. Each hierarchical net is reported only once, in the module containing its driver.

Example

The following command displays a crosstalk violations report similar to the one shown below:

```
> report_crosstalk_violations -verbose


Transition threshold for crosstalk prevention: nnn
Threshold tolerance for crosstalk prevention: nn
```

	net name	worst transition	violation
Module: A			
	ABC	nnn	nnn
	.	.	.
	.	.	.

```
Total Number of Nets = nn
Total Number of Violations = nn
```

Related Information

do_xform_prevent_crosstalk

reset_crosstalk_threshold

report_design_rule_violations

report_design_rule_violations

```
report_design_rule_violations [-ignore_clknet] [-min_drvs]
     [-current_module_only] [-verbose] [-ignore_dont_modify_nets]
     [{ > | >> } filename] [-summary]
```

Generates a report on all the design rule violations in the design. The information contained in the report depends on the options used.

Options and Arguments

`{> | >>} filename`
: Specifies the name of the report file. If *`filename`* is not specified, the report is displayed on standard output. *`filename`* must be the last argument in the list.

`-current_module_only`
: Only the violations in the current module are reported.

`-ignore_clknet`
: Does not report the clock net violations.

`-ignore_dont_modify_nets`
: Does not report design rule violations for nets that are set `dont_modify`.

`-summary`
: Provides a summary report of the violations on each net (instead of listing all of the violations). The worst violations are reported for every violating net.

`-verbose`
: Reports all design rules for every net and port in the design regardless of whether there were any violations or not.

Related Information

`report_area`

`report_fsm`

`report_hierarchy`

report_timing

set_table_style

report_fsm

```
report_fsm [-vector vector_name] [-state_table] [-encoding] [-hierarchical]
     [{> | >>} filename]
```

Generates a report on the finite state machines on the design. The contents of the report include the design name, number of states, inputs, outputs, and so on.

Options and Arguments

`{> | >>} filename`
: Saves the report with the *filename* entered. If the option is not used, the data is presented on the standard output. *filename* must be the last argument in the list.

`-encoding`
: Reports all state assignments for each selected FSM.

`-hierarchical`
: Displays the downward path of all FSMs in the current module.

`-state_table`
: Generates a state transition table in report form.

`-vector vector_name`
: Displays a report on the FSM represented by the state `vector` *vector_name*. If this option is not used, all FSMs in the current module are reported.

Example

The following command generates the report shown below:

```
> report_fsm -encoding -state_table
+--------------------------------------+
| Report         | report_fsm          |
|----------------+---------------------|
| Options        | -state -encoding    |
+----------------+---------------------+
| Date           | 20030514.025128     |
| Tool           | bgx_shell           |
| Release        | v5.xxxxxx0          |
| Version        | May 13 2003 23:52:05 |
```

```
+---------------+---------------------+
| Current Module | fsm1                |
+-----------------------------------+

+-------------------------------------------------+
|           Finite State Machine Report           |
|-------------------------------------------------|
| Design              | fsm1                      |
| State Vector        | state_reg                 |
| File                | test.v                    |
| Line                | 20                        |
+---------------------+---------------------------+
| States              | 5                         |
| Initial states      | (STATE0)                  |
| Equivalent states   | <None>                    |
| Unreachable states  | <None>                    |
| Terminal states     | <None>                    |
| Preserved states    | <None>                    |
| Transitions         | 13                        |
+---------------------+---------------------------+
| Inputs              | 1                         |
| Input names         | (reset)                   |
| Unused inputs       | <None>                    |
| Hold signal         | <None>                    |
+---------------------+---------------------------+
| Outputs             | 6                         |
| Output names        | (state_out[0:2] out[0:2]) |
+---------------------+---------------------------+
| Clock               | clk                       |
| Sense               | rising-edge               |
+---------------------+---------------------------+
| Encoding            | binary                    |
| Encoding bit length | 3                         |
+-------------------------------------------------+

+--------------+
|    State     |
|   Encoding   |
|--------------|
| STATE0 | 000 |
| STATE1 | 001 |
| STATE2 | 010 |
| STATE3 | 011 |
```

```
| STATE4 | 100 |
+--------------+
+-------------------------------------------------------------------+
|                     State Transition Table                        |
|-------------------------------------------------------------------|
| reset | CS     | NS     | state_out[2] state_out[1] state_out |
|       |        |        | [0] out[2] out[1] out[0]           |
+-------+--------+--------+------------------------------------+
| 1     | STATE0 | STATE0 | 000---                             |
| 0     | STATE0 | STATE1 | 000000                             |
| 1     | STATE1 | STATE0 | 001---                             |
| 0     | STATE1 | STATE2 | 001001                             |
| 1     | STATE2 | STATE0 | 010---                             |
| 0     | STATE2 | STATE3 | 010010                             |
| 1     | STATE3 | STATE0 | 011---                             |
| 0     | STATE3 | STATE4 | 011011                             |
| -     | STATE4 | STATE0 | 100---                             |
| 0     | STATE4 | STATE0 | 100100                             |
| 1     | *      | STATE0 | ---001                             |
| 0     | OTHERS | STATE0 | ---111                             |
| 1     | OTHERS | STATE0 | ---001                             |
+-------------------------------------------------------------------+
```

Related Information

report_area

report_design_rule_violations

report_hierarchy

report_timing

set_table_style

report_globals

`report_globals [-tcl_list] [-modified] [-group` *`group_name`*`] [{ > | >> }` *`filename`*`]`

Returns the current and default values for the specified global variables. If no arguments are specified with the command, the complete list of all globals will be reported.

Options and Arguments

`{> | >>}` *`filename`*

Saves the report with the *`filename`* entered. If the option is not used, the data is presented on the standard output. *`filename`* must be the last argument in the list.

`-group` *`group_name`*

Returns the current and default values of the specified global variables belonging to the specified group. Valid values for *`group_name`* include `hdl`, `aware`, `ta`, `opt`, `pks`, `dft`, `dist`, `dcn`, `edif`, `misc`, and `ui`.

`-modified`

Returns the current and default values of the specified global variables that have been changed from their default value.

`-tcl_list`

Returns the current and default values of the specified global variables as a Tcl list.

Examples

- The following command returns the current and default values of the user interface global variables that have been modified from their default values, as is subsequently shown:

```
> report_globals -group ui -modified
+--------------------------------+
|              ui                |
|--------------------------------|
|   Global    | Value | Default  |
|-------------+-------+----------|
| line_length |  126  |   80     |
+--------------------------------+
```

- The following command returns the current and default values of the user interface global variables.

```
> report_globals -group ui
```

```
+-------------------------------------------------------+
|                           ui                          |
|-------------------------------------------------------|
|         Global          |    Value     |   Default    |
|-------------------------+--------------+--------------|
| acsh_prompt             |   %t[%h]>    |   %t[%h]>    |
| logfile                 | ac_shell.log | ac_shell.log |
| message_verbosity_level |      7       |      7       |
| line_length             |     113      |      80      |
| echo_commands           |    false     |    false     |
+-------------------------------------------------------+
```

- The following command returns the current and default values of the test synthesis global variables.

```
> report_globals -group dft
```

```
+-------------------------------------------------------------+
|                            dft                              |
|-------------------------------------------------------------|
|               Global                |  Value  | Default |
|-------------------------------------+---------+---------|
| dft_enable_combinational_loop_check |  false  |  false  |
| dft_enable_race_condition_check     |  false  |  false  |
| dft_verbosity_level                 |    1    |    1    |
| dft_scan_avoid_control_buffering    |  false  |  false  |
| dft_allow_scan_path_inv             |  true   |  true   |
| dft_scan_path_connect               |  chain  |  chain  |
| dft_scan_output_pref                | non_inv | non_inv |
| dft_scan_enable_connect             |  true   |  true   |
| dft_scan_preserve_config            |  false  |  false  |
+-------------------------------------------------------------+
```

Related Information

reset_global

set_global

report_hierarchy

`report_hierarchy [-inst] [-tcl_list] [{> | >>} filename]`

Reports the structural hierarchy as it exists at various stages in the synthesis process. The hierarchy reported is always that of the netlist in memory, with the current module as the top of the hierarchy. The report uses the following symbols to distinguish hierarchical blocks:

b	Black box module
g	Generic (unmapped) module
o	Optimized module
x	`dont_modify` attribute is set for this module
m	Module contains a mapped view

Options and Arguments

`{> | >>} filename`

Saves the report with the *filename* entered. If the option is not used, the data is presented on the standard output. *filename* must be the last argument in the list.

`-inst`

Includes the instance name of the module instantiation in the output. Modules that are multiply instantiated will include a separate line of output for each instantiation.

`-tcl_list`

Reports the hierarchy in the form of a Tcl list. To use the output of the Tcl list to generate Tcl code, use a Tcl variable, as shown in the example below.

Examples

- The following command generates a report on the current hierarchy and save it in a file `design.hier.rpt`

  ```
  > report_hierarchy design.hier.rpt
  ```

- The following command displays the instance name of the module instantiation. Sample output follows the command.

  ```
  > report_hierarchy -inst
  ```

```
-my_top(g)
-mid1_0 -> my_mid1(m)
-mid1_1 -> my_mid1(m)
-mid2_0 -> my_mid2(g)
```

- The following command uses the Tcl variable `result` to save the `-tcl_list` code into the file `save_tcl_code` file.

```
> set result [report_hierarchy -tcl_list]
> echo $result > save_tcl_code
```

Related Information

`report_area`

`report_design_rule_violations`

`report_fsm`

`report_timing`

`set_table_style`

report_path_group_options

`report_path_group_options [-tcl_list] [{> | >>} filename]`

Produces a report of the path group options set by the `set_path_group_options` command. The report includes the name of the path group, the effort level set, a plus sign (+) if more than the worst path is to be optimized, the target slack, the number of endpoints, the worst slack of the group, and the Total Endpoint Failing Slack (TEFS), which is the sum, over all failing endpoints in the group, of the amount by which the slack misses the target slack.

Question mark entries in this report indicate that the options for that group have not been set and that the actual values are picked up from whatever command line options are passed to the `do_optimize` command.

This command is useful for checking which path group options were set (using `set_path_group_options` command). If you just want to report the timing of all path groups, use `report_path_group_timing`.

Options and Arguments

`> | >> filename`
: Redirects the output to the named file. If this option is omitted, the default is `stdout`.

`-tcl_list`
: Produces the report in Tcl list form instead of a tabular format. Useful for extracting information from the report in a Tcl program.

Example

The following command reports the group defined in the example for `set_path_group_options`. In this example the worst slack is -0.34, which meets the target slack of -1 for path group GRPA; therefore, the TEFS is 0.00.

```
report_path_group_options
+-------------------------------------------------------------+
|                  Path Group Options Report                  |
|-------------------------------------------------------------|
| PathGroup | Effort | All | Target |  Critical | Worst | TEFS |
|           |        | Pts |  Slack | Endpoints | Slack |      |
|-----------+--------+-----+--------+-----------+-------+------|
| <default> |      ? |  ?  |      ? |      ?/51 | -1.73 |    ? |
|      GRPA |   high |  +  |  -1.00 |       0/6 | -0.34 | 0.00 |
+-------------------------------------------------------------+
```

Related Information

report_path_group_timing

reset_path_group

set_path_group

set_path_group_options

report_resources

```
report_resources [-all] [-hierarchical] [-instance] [{ > | >> } filename]
     [-module module_name] [-mux] [-sequential] [-shared] [-summary]
```

Provides information about the datapath modules in the design. The summary report contains the information about different arithmetic operators in the datapath module. By default, resources are listed for the current module, but if -module option is used, the resources are separated for the specific module.
Default: Contains information about the architecture, size, format and corresponding RTL line number of datapath clusters in the datapath modules.

In addition, the report_resources command reports the resources generated by each of the ambitware generators.

Options and Arguments

-all

Reports all the aware resources (arithmetic, mux, and sequential components) in the current module.

filename

Writes the report in the summary file.

-hierarchical

Generates a report for the hierarchy under the current module or the specified module. The hierarchy of the current module or specified module is traversed and the report is generated for each datapath module found.

-instance

Generates a report for each instance of the datapath modules.

-module *module_name*

Specifies the module for which all the datapath resources are to be separated.

-mux

Reports the multiplexers (mux) generated in the netlist and their attributes. The attributes reported are the number of data inputs to the mux, the width of the mux, and the mux architecture.

The three possible mux architectures are *encoded*, *decoded*, and *mixed*. Encoded and decoded multiplexers will automatically

be identified with the `-mux` option alone. For architectures with both encoded and decoded output ports (mixed mux), the architecture of each individual output port is reported with the `-mux` option. When the `-mux` option is used in conjunction with the `-summary` option on a mixed mux, only its identification as a mixed mux will be reported.

In addition, the RTL source file name and the line number of the conditional statement to which the mux is associated are also reported.

When `-mux` is used in conjunction with the `-hierarchical` option, all the muxes generated in a hierarchical module are reported.

When `-mux` is used in conjunction with the `-module` option, the muxes in the specified module name are reported.

`-sequential`

Reports the sequential components in the netlist and the attributes associated with them. Reports the type (latch or flip-flop), width, and the asynchronous and synchronous set/reset values associated with each bit of the sequential element.

Used with `-hierarchical`, reports all sequential components in a hierarchical module.

Used with `-module` *`module_name`*, reports the sequential components in the module.

Used with `-summary`, reports a summary of the set/reset values of the sequential components in the current module.

`-shared`

Reports the datapath clusters that are being shared after resource sharing. By default, the `report_resources` command prints out the datapath resources in the source clusters being shared in addition to the resources in the shared module.

Used with `-hierarchical`, reports the datapath resources in all the shared modules along with the datapath resources in the unshared clusters in a hierarchical module.

Used with `-module` *`module_name`*, reports the datapath resources in the shared module and the datapath resources in the clusters before resource sharing.

Used with `-instance`, reports the instance name of the current module along with its other attributes.

`-summary`

Generates summary report for each datapath module. The summary contains module or instance name, type and number of each arithmetic operator in the module or instance.

Examples

- The following shows the report format for mux:

```
> report_resources -mux -hier -instance
+----------------+-----------------------+
| Report         | report_resources      |
+----------------+-----------------------+
| Options        | -seq -hier            |
+----------------+-----------------------+
| Date           | 20010926.155535       |
| Tool           | ac_shell              |
| Release        | v5.0-d147             |
| Version        | Sep 26 2001 10:43:34  |
+----------------+-----------------------+
| Current Module | mux                   |
+----------------+-----------------------+

+----------+------------+----------+------+-------------+-------+-----------+
|                            Multiplexers                                     |
+----------+------------+----------+------+-------------+-------+-----------+
| Instance |   Module   |   File   | Line | Data Inputs | Width | Swappable |
+-------------------------------------------------------------------------------+
|  i_55    | AWMUX_16_1 | design.v | 8    | 16          | 1     |  true     |
|  i_62    | AWMUX_4_1  | design.v | 20   | 4           | 1     |  true     |
+-------------------------------------------------------------------------------+
```

■ The following shows the report format for sequential elements
:

```
> report_resources -hier -seq
+--------------------------------------+
| Report         | report_resources  |
|----------------+----------------------|
| Options        | -hier -seq        |
+----------------+----------------------+
| Date           | 20030227.114726   |
| Tool           | bgx_shell         |
| Release        | v5.9-s009         |
| Version        | Feb 25 2003 11:49:53
+----------------+----------------------+
| Current Module | foo               |
+----------------------------------+

+-------------------------------------------------------------------------------------------
|                                Sequential Elements
|-------------------------------------------------------------------------------------------
|    Module    |        File         | Line |  Name |  Type | Width | AS | AR | SS | SR|
+--------------+---------------------+------+-------+-------+-------+----+----+----+------
| AWMEM_LATCH_ | /ambit/regress/bg/  | 8    | x_reg | LATCH | 1     | N  | N  | -  | - |
| x_reg        | hls/tests/reportRe  |      |       |       |       |    |    |    |   |
|              | so/sequential.v     |      |       |       |       |    |    |    |   |
| AWMEM_LATCH_ | /ambit/regress/bg/  | 8    | y_reg | LATCH | 1     | N  | N  | -  | - |
| y_reg        | hls/tests/reportRe  |      |       |       |       |    |    |    |   |
|              | so/sequential.v     |      |       |       |       |    |    |    |   |
| AWMEM_LATCH_ | /ambit/regress/bg/  | 8    | z_reg | LATCH | 1     | N  | N  | -  | - |
| z_reg        | hls/tests/reportRe  |      |       |       |       |    |    |    |   |
|              | so/sequential.v     |      |       |       |       |    |    |    |   |
+-------------------------------------------------------------------------------------------
```

- This shows the report format for the `-shared` option:

```
> report_resources -hier -shared
+----------------------------------------+
| Report         | report_resources      |
|----------------+-----------------------|
| Options        | -hier -shared         |
+----------------+-----------------------+
| Date           | 20030227.122903       |
| Tool           | bgx_shell             |
| Release        | v5.9-s009             |
| Version        | Feb 25 2003 11:49:53  |
+----------------+-----------------------+
| Current Module | case3                 |
+----------------+-----------------------+

+-------------------------------------------------------------------------------------+
|                              Arithmetic Resources                                   |
|-------------------------------------------------------------------------------------|
|    Module    |        File        | C |   Arch    | Op | Line | Out |   In    |
+--------------+--------------------+---+-----------+----+------+-----+-------------+
| AWDP_partiti |                    | 1 | ripple    | +  |      | 16u | 16ux16u |
| on_2         |                    |   |           |    |      |     |         |
|              |                    |   | booth     | *  |      | 16u | 16ux16u |
+--------------+--------------------+---+-----------+----+------+-----+-------------+
|              |                    | # | fcla/boot | *  | 17   | 16u | 16ux16u |
|              |                    |   | h         |    |      |     |         |
|              |                    | # | fcla      | +  | 12   | 16u | 16ux16u |
|              |                    |   | booth     | *  | 12   | 16u | 16ux16u |
|              |                    | # | fcla      | +  | 14   | 16u | 16ux16u |
+--------------+--------------------+---+-----------+----+------+-----+-------------+
| AWDP_MULT_2  | /ambit/regress/bg/ | 1 | ripple/bo | *  | 16   | 16u | 16ux16u |
|              | hls/tests/res_shar |   | oth       |    |      |     |         |
|              | e/incr/addmult.v   |   |           |    |      |     |         |
+-------------------------------------------------------------------------------------
```

#: Source cluster. A module can share multiple source clusters.

report_skewed_clock_pins

```
report_skewed_clock_pins [-all]
```

Reports each sequential element in the netlist that is driven by a buffer not driving any other gate. It also reports the name of the buffer, the delay added, the worst slack checked by this clock pin, and the worst slack triggered by this clock pin.

This command can be useful for getting an overview of which sequential elements in the netlist have been skewed at the leaf of the clock tree (see `do optimize -skew_clock`).

Options and Arguments

`-all`	Reports all clock pins of sequential elements in the netlist.

Example

The following command generates the subsequent report:

```
> report_skewed_clock_pins
+-------------------------------------------------+
|              Skewed Clock Pins Report           |
|-------------------------------------------------|
|      Pin |     Buffer(s) | Skew | D-Slack | Q-Slack |
|----------+---------------+------+---------+---------|
| Q_reg/CK | DLY1X1 BUFX20 | 0.98 |   -0.67 |   -0.69 |
+-------------------------------------------------+
```

Related Information

`do optimize -skew_clock`

report_vhdl_library

`report_vhdl_library [-verbose] [`*`library`*`] [{ > | >> } `*`filename`*`]`

Lists the mappings between all the defined VHDL libraries and the directories to which they are mapped. This command can be used to check where BuildGates Synthesis picks up VHDL units such as packages.

Options and Arguments

`file_name`
: Names the file to which the report is written.

`library`
: Lists the mapping for the specified VHDL library.

`-verbose`
: Lists the contents of the VHDL library.

Related Information

`set_global hdl_vhdl_read_version`

`set_vhdl_library`

report_wire_self_heat_violation

```
report_wire_self_heat_violation [-ignore_dont_modify_nets] [-ignore_clknets]
     [-verbose]
```

Reports all nets with wire-self-heat violations, such as a violation whose current measures exceed its corresponding current limit.

Options and Arguments

`-ignore_clknets`
: Excludes clock nets from the report.

`-ignore_dont_modify_nets`
: Excludes `dont_modify` nets from the report.

`-verbose`
: Reports all nets, including those without violations.

Example

The following command displays a report similar to the one shown below:

```
> report_wire_self_heat_violation -verbose
```

	Net	Slew	Freq	
		Cap	Limit	Violation
		I_ave	Limit	Violation
		I_peak	Limit	Violation
		I_rms	Limit	Violation
Module A:				
	ABC	nnn	nnn	
		nnn	nnn	nnn
		nnn	nnn	nnn
		nnn	nnn	nnn
		nnn	nnn	nnn

Net	Slew	Freq	
.	.	.	.
.	.	.	.
.	.	.	.

```
Total Number of Nets = nnn
Total Number of Violations = nnn
```

Related Information

do_optimize

do_xform_prevent_wire_self_heat

reset_wire_self_heat_prevention

reset_crosstalk_threshold

```
reset_crosstalk_threshold
```

Clears any existing crosstalk threshold stored with the design.

Use this command for any design that has previously gone through crosstalk prevention and has a threshold stored with it.

Related Information

do_optimize

do_xform_prevent_crosstalk

report_crosstalk_violations

reset_dont_modify

```
reset_dont_modify [-network] [-stop_at_complex_gates] [-hierarchical]
     [-no_hierarchical] list_of_object_ids
```

Removes the dont_modify attribute set by a previous set_dont_modify command, applied to a design module, instance, or port. Use set_cell_property for applying and removing properties on library cells.

Options and Arguments

-hierarchical

Valid only for module, net and port object types. It is the default for net and port object types.If -hierarchy is specified for modules, all lower levels of hierarchy below the specified module are also removed.

list_of_object_ids

Specifies the list of object IDs for the instances, modules, nets, pins, or ports to be reset. Ports must be either input or inout ports of modules or output or inout pins of instances. When ports are specified, it is equivalent to specifying the nets connected to those ports. See the set_dont_modify command.

-network

Removes the dont_modify attribute from the combinational path connected to the given object. This option is valid with net and port object types.

-no_hierarchical

Valid for only net and port object types. It is the default for module object ids. If -no_hierarchical is specified, the dont_modify attribute will be removed from nets and instances connected to the specified nets or ports that are in the same hierarchy or in the combinational path (if -network is specified) of the hierarchy.

-stop_at_complex_gates

Stops the tool's traversal of the hierarchy at the input to complex gates (non-buffer and non-inverter gates). The dont_modify flag will not be reset on objects beyond this point.

Note: To use this option, you must also specify the -network option.

Related Information

do_dissolve_hierarchy

do_uniquely_instantiate

set_cell_property

set_dont_modify

reset_failsafe

```
reset_failsafe
```

Restores the name of the file in which to store the database in the event of a fatal error to the default location of `/tmp/process_id`.

Related Information

`set_failsafe`

reset_global

`reset_global` *`global_variable_name`*

Sets the specified global variable to its default value.

Options and Arguments

`global_variable_name`
The name of the global variable to reset.

Example

The following command resets the VHDL version for writing out VHDL netlists to the default: 1993.

```
reset_globals hdl_vhdl_write_version
```

Related Information

`set_global`

reset_register_type

`reset_register_type` *`register_instance_id_list`*

Removes the register type constraint imposed by the previously issued `set_register_type` command. Subsequent mapping will not enforce mapping to the previously specified register type.

Options and Arguments

`register_instance_id_list`
Specifies the list of object IDs for the register instances.

Example

The following commands set, then remove the LD1 and FD1 register type constraint:

```
> set_register_type -latch LD1 -flip_flop FD1 [find -registers *]
...
> reset_register_type [find -registers *]
```

Subsequent mapping will not enforce mapping to register cells of the same type as LD1 and FD1.

Related Information

`set_register_type`

reset_vhdl_library

`reset_vhdl_library` *`library_name`*

Removes all VHDL units that have been analyzed into a specific library. This is useful if a package was mistakenly analyzed into the wrong library, or if you wish to clear the library of all VHDL units previously analyzed into it and start over.

Options and Arguments

`library_name`

Specifies the name of the library from which all analyzed units are to be deleted.

Examples

- The following command resets the library `MYLIB` after a VHDL package from `wrongpack.vhd` was analyzed into it:

```
> read_vhdl -library MYLIB wrongpack.vhd
> reset_vhdl_library MYLIB
```

- The following command reads the correct package into `MYLIB:`

```
> read_vhdl -library MYLIB rightpack.vhd
```

Related Information

`report_vhdl_library`

`set_vhdl_library`

reset_wire_self_heat_prevention

```
reset_wire_self_heat_prevention
```

Clears any existing wire-self-heat stored with the design.

Use this command for any design that has previously gone through wire-self-heat prevention and has a threshold stored with it.

Related Information

do_optimize

do_xform_prevent_crosstalk

report_wire_self_heat_violation

set_attribute

`set_attribute` *`object_id attribute_name attribute_value`*

Sets the value set for the specified attribute on the specified object. You can set tool- and user-defined attributes.

The `set_attribute` command can be used to disable some of the default `do_optimize` behavior. You can set the *`object_id`*, the *`attribute_name`*, and the *`attribute_value`*. The corresponding options for `do_optimize` are `-force`, `-no_partition`, `-priority`, `-flatten`, `-phase`, `-multiple_output`.

By default, `do_optimize` partitions the module. If you do not want to partition a particular module, then you can use the following command:

```
set_attribute object_id no_partition true
```

If you use the `do_optimize -no_partition` command and you want to partition a particular module, then use the following:

```
set_attribute object_id no_partition false
```

If the following command is used:

```
set_attribute [find -techlib lca300kv] default_operating_conditions WCCOM
```

the software treats `default_operating_conditions` as a user-specified attribute having a value of `WCCOM`. This attribute is set on the object `lca300kv`.

You can use the following command to query the attribute:

```
get_info [find -techlib lca300kv]
```

The `set_attribute` command can be used to set the operating conditions, but it is recommended that you use the `set_operating_conditions` command instead.

The `set_attribute` command is not limited to setting the attributes listed below. User-defined attributes are also valid.

Options and Arguments

`attribute_name`

Specifies the name of the attribute to set. Refer to the Attributes list below for a list of tool-defined attributes.

`attribute_value`

Sets the attribute to this value.

object_id

The ID of the module, instance, cellref, net, port, or pin to modify.

Attributes

The attributes below are defined by the BuildGates Synthesis tool. User-defined attributes are also valid.

`flatten {on | off | auto}`

Controls the operations used for the optimization of logic equations. If the flatten option is set to on, the logic equations are flattened into a sum of products form before applying the optimization. This helps to create a faster and more optimal implementation of the logic, even though it may use extra area (gates). If the flatten option is set to `auto`, only certain flattening operations are performed; some of the time consuming steps are ignored. The flattening step may be disabled entirely by setting the option to `off`.
Default: `off`.

`force {true | false}`

Forces this command to carry out all of its steps irrespective of the initial state of the design.

`no_partition {true | false}`

Indicates that the design should not be automatically partitioned into smaller modules for optimization, irrespective of the number of input pins on the cells in the module. Each module boundary should be retained as the partition to optimize.

`phase {true | false}`

Performs phase assignment on module *object_id* in logic minimization during structuring if the module is flattened into two levels of logic successfully with option `-flatten auto` or `-flatten on` (or attribute flatten) or on sum-of-products logic hierarchy, if any, that is inferred from constant case statement within the module *object_id*.
Default: `false`

Below are two examples.

Set phase attribute to `false` on all modules in the design hierarchy except module `module_A`, then optimize.

```
set_attribute module_A phase true
do_optimize
```

Apply phase assignment on all modules in the design hierarchy in structuring except module `module_B`, then optimize using the `-phase_assignment` option.

```
set_attribute module_B phase false
do_optimize -phase_assignment
```

`priority {area | time}`

Sets area or timing minimization to be the highest priority of the technology, independent of the optimization step. If you have a loosely constrained or unconstrained design, then the priority should be set to area.
Default: `time`

Examples

The following examples demonstrate how you can use the `set_attribute` command to change the default behavior of `do_optimize`:

- The following command prevents the partition of the module `module_x1`. By default, `do_optimize` partitions modules:

  ```
  > set_attribute module_x1 no_partition true
  ```

- The following command shows how you can partition a particular module even if you previously used the `do_optimize -no_partition` command. This command allows the partition of the module `module_x2`:

  ```
  > set_attribute module_x2 no_partition false
  ```

- The following command sets the user-defined attributes. The following command sets the attribute `my_attribute` on the net object `my_net` to a value of `my_value`:

  ```
  > set_attribute [find -net my_net] my_attribute my_value
  ```

- The following commands query the attribute, as shown below:

  ```
  > get_attribute [find -net my_net] my_attribute
  ```

  or

  ```
  > get_info [find -net my_net]
  ```

Related Information

`delete_attribute`

get_attribute

get_info

set_global

set_aware_component_property

```
set_aware_component_property [-library library_name] dont_utilize {true | false}
     component_name
```

Allows the property to be set on an aware component. The only allowed property is `dont_utilize`. When this property is set on a component, it is not available for use.

Options and Arguments

`component_name`
: Specifies the name of the component for which the property is to be set.

`dont_utilize {true | false}`
: Currently, the only allowed property is `dont_utilize`.

`-library library_name`
: Selects the name of the `aware` library to which the component belongs. If no library is specified, then the component is searched in all `aware` libraries (in the order determined by the global `aware_library_search_order`).

Example

The following command sets the `dont_utilize` property on the AWARITH_ABS component from the AWARITH library:

```
> set_aware_component_property -library AWARITH dont_utilize true AWARITH_ABS
```

Related Information

`delete_aware_component`

`set_aware_library`

`report_aware_library`

set_aware_library

`set_aware_library` *`library_name library_path`*

Maps the logical library name *`library_name`* to the physical path *`library_path`*.

Options and Arguments

`library_name`
: Specifies the name of the `aware` library.

`library_path`
: Specifies the path to map to *`library_name`*.

Example

The following example maps the logical name of the library `AWMYLIB` to the existing directory `/home/smith/libs/lib1` where the analyzed components physically reside:

```
> set_aware_library AWMYLIB /home/smith/libs/lib1
```

Related Information

`delete_aware_component`

set_cell_property

`set_cell_property` *`property value`* `-lib` *`library_name cell_names`*

Sets properties of one or more cells to specified values. These properties can be `dont_modify`, `dont_utilize`, and so on. The value of the properties can be set to `true` or `false`.

Note: The `set_tech_info`, `get_tech_info`, and `reset_tech_info` commands should be used instead of this command. These are powerful mechanisms which can go beyond the functionality of `set_cell_property.` They can be used to update library information. The `set_cell_property` command will be phased out in the future.

Important

The following commands and associated options do not cause optimization to automatically remap instances in an already mapped netlist:

```
set_tech_info -lib lib_name -cell cell_name -dont_utilize true
set_cell_property dont_utilize true [find -cellrefs cell_name]
```

If you set the `dont_utilize` attribute on a cell after mapping your design, the tool may still use that cell during subsequent optimization. To ensure that the cell is not used, you must either unmap your design or use the `do_optimize -force` option, and then re-map and re-optimize your design.

Options and Arguments

`cell_names`

Specifies the cells whose properties will be changed.

`-lib` *`library_name`*

Specifies the library containing a list of properties that will be changed.

`property`

Specifies the name of a property for the cell that needs to be set. Properties include:
`dont_modify {true | false}`
`dont_utilize {true | false}`
`objecttype` *`type`*
`module` *`id`*
`pins` *`id`*
All the properties of a cell can be listed using the command `get_info`.

value

Specifies a new value for the property.

Related Information

get_info

set_current_module

set_current_instance

```
set_current_instance [instance_name]
```

Sets the instance given by *instance_name* to be the current instance. Design objects referenced by subsequent commands can be found relative to this instance. All searches for design objects are started in this instance. If design objects are referenced hierarchically, *instance_name* is used as the root (top) of the hierarchy.

The set_current_instance command is used instead of the set current module command when a module has multiple instances and each instance requires a separate set of constraints.

Options and Arguments

instance_name

Specifies the name of the instance to become the current instance. If *instance_name* is not given, the current instance is set to the top instance of the hierarchy.

Example

The following example sets the current instance to MAC/MULTI, then later sets it back to top:

```
> set_current_instance MAC/MULT1
...
...
> set_current_instance
```

Related Information

get current instance

set current module

set_current_module

`set_current_module` *`module_name`*

Sets the module represented by *`module_name`* as the current module. Design objects referenced by subsequent commands can be found relative to this module. All searches for design objects are started in this module. If design objects are referenced hierarchically, *`module_name`* is used as the root (top) of the hierarchy.

If a module has multiple instances and each instance requires a separate set of constraints, then the `set_current_instance` command is used.

Note: When the `set_current_module` command is used, the *`current_instance`* is reset to the `set_top_timing_module`.

Options and Arguments

`module_name`

Names the module that is being used in the current context.

Example

This example sets the `mycounter` module as the current context. Now the constraints can be applied to the ports of `mycounter`:

```
> set_current_module mycounter
```

To apply constraints to ports of another module use the `set_current_module` command with another module name or use the hierarchical name of the ports.

Related Information

`do_uniquely_instantiate`

`get_current_module`

`set_top_timing_module`

set_dissolve_hierarchy

`set_dissolve_hierarchy {`*`obj_id_list`*`} {true | false}`

Sets the dissolve property for each module or hierarchical instance in the specified list. During optimization, modules or instances tagged `true` are dissolved into their parent modules. Modules or instances with the `dont_modify` attribute are not touched by this command.

Note: This command has no effect on AmbitWare (AW) or Datapath modules.

Options and Arguments

`{`*`obj_id_list`*`}`
: Lists the modules or hierarchical instances to be dissolved during optimization.

`{true | false}`
: Specifies the value to set the dissolve property.

Related Information

`do_create_hierarchy`

`do_dissolve_hierarchy`

set_dont_modify

```
set_dont_modify [-network] [-hierarchical | -no_hierarchical]
     [-stop_at_complex_gates] list_of_object_ids
```

Prevents instances, modules, nets, pins, or ports from being modified. This is useful when those objects have already been optimized. No further optimization will be done on the specified instances, modules, nets, pins, or ports. The command is also useful when macros are instantiated from a library.

Note: When an instance pin or port is specified, it is equivalent to specifying the nets connected to those pins or ports.

Once a module is marked for no modifications, the commands `do_uniquely_instantiate` or `do_dissolve_hierarchy` are ignored for that module. When a net is specified, all the connecting instances are affected by the `set_dont_modify` property. When an instance pin or port is specified, the net connected to the pin or port is marked as `dont_modify`.

The nets, modules or instances identified by this command are preserved “as is” during the synthesis process.

For nets, ports, and pins, all nets and instances connected hierarchically to the specified nets or ports will be marked `dont_modify`. If the `-network` option is specified, then hierarchies will be traversed to mark all nets and instances in the combinational path of specified nets and ports as `dont_modify`.

Note: PKS does not set the `dont_modify` attribute on generic instances.

Options and Arguments

`-hierarchical`

Valid only for module object types. It is the default for net and port object types.If -`hierarchy` is specified for modules, all lower levels of hierarchy below the specified module are also marked.

`-no_hierarchical`

Valid for only net and port object types. It is the default for module object ids. If `-no_hierarchical` is specified, then only nets and instances connected to the specified nets or ports that are in the same hierarchy or in the combinational path (if -network is specified) of the hierarchy will be marked `dont_modify`.

list_of_object_id

Specifies the list of object IDs for the instances, modules, nets, pins, or ports to be preserved. Ports must be either input or inout ports of modules or output or inout pins of instances. When ports are specified, it is equivalent to specifying the nets connected to those ports.

-network

Specifies that the dont_modify attribute applies to the combinational path connected to the given object. This option is valid with net and port object types.

-stop_at_complex_gates

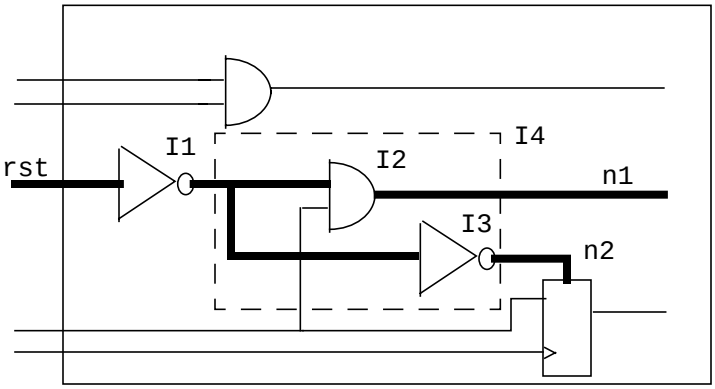
Stops the tool's traversal of the hierarchy at the input to complex gates (non-buffer and non-inverter gates). The dont_modify flag will not be set on objects beyond this point.

Note: To use this option, you must also specify the -network option.

Examples

- The following example shows how the highlighted nets in the figure below are preserved. (If the -no_hierarchical option was specified the highlighted path in the instance I4 (shown by the dotted boundary), and the nets n1, n2 would not be preserved):

```
> set_dont_modify  -network [find -net rst]
```



- The following command preserves the module myRAM and all of its hierarchy:

```
> set_dont_modify -hierarchical [find -mod myRAM]
```

- The following command marks the instance RAPID_FIFO for preservation:

```
> set_dont_modify [find -inst RAPID_FIFO]
```

- The following command marks the net and the two or more instances to which it connects as dont_modify:

```
> set_dont_modify [find -net n_190]
```

- The following command marks the net connected to the port as dont_modify, but not the other end instance:

```
> set_dont_modify [find -port port_A]
```

- The following command marks the net connected to the port and all the other end instances in the fanout cone as dont_modify, stopping at a hierarchical port or a sequential cell:

```
> set_dont_modify -network -no_hierarchical [find -port port_A]
```

- The following command marks the net connected to the port and all the other end instances in the fanout cone as dont_modify, without stopping at a hierarchical port:

```
> set_dont_modify -network [find -port port_A]
```

Related Information

do_dissolve_hierarchy

do_uniquely_instantiate

reset_dont_modify

set_current_module

set_failsafe

`set_failsafe` *`filename`*

Sets the name of the file in which to store the database in the event of a fatal error. The specified file name is used instead of the system default location of `/tmp/`*`process_id`*.

Options and Arguments

`filename`

Specifies the name of the directory and file in which to store the database in the event of a fatal error.

Related Information

`reset_failsafe`

set_global

`set_global` *`global_name value`*

Sets the global variables for the `ac_shell`. These variables are independent of the design, but affect the overall run procedure and `the ac_shell` policies on various global matters, for example, the Verilog module naming style, message verbosity level, slew limit, and line length in reports.

All global variables have default value when `ac_shell` starts up. Once a variable is set to a new value, it retains that value for the current `ac_shell` session unless a subsequent use of this command sets a new value for the variable. User-specified values for global variables are not saved in the Ambit Synthesis database (ADB) file.

The value of any global variable can be obtained using `get_global` command. The value of any global variable can be set to the default value using `reset_global` command. The `set_global` command has a one-to-one correspondence with the `get_global` and `reset_global` commands. Every variable defined in `set_global` also applies to `get_global` and `reset_global`.

For a complete list of globals and their descriptions, see *Global Variable Reference for BuildGates Synthesis and Cadence PKS*.

Options and Arguments

`global_name`
: The name of the global you wish to set.

`value`
: Any argument necessary for the global (for example: `true`, `false`, a string, or an integer or float value).

Related Information

`get_global`

`reset_global`

`set_attribute`

set_logic0

`set_logic0` *`port_list`*

Sets ports to `logic 0` so that the optimization tool can better optimize the design.

Options and Arguments

`port_list`
Lists the ports that are to be set to `logic 0`.

Example

The following command sets all ports with names beginning with "`rst`" to `logic 0`.

```
> set_logic0 [find -port rst*]
```

Related Information

`do_xform_propagate_constants`

`set_logic1`

set_logic1

`set_logic1` *`port_list`*

Sets ports to `logic 1` so that the optimization tool can do a better optimization of the design.

Options and Arguments

`port_list`

Lists the ports that are to be set to `logic 1`.

Example

This command sets all ports with names beginning with "`preset`" to `logic 1`.

```
> set_logic1 [find -port preset*]
```

Related Information

`do_xform_propagate_constants`

`set_logic0`

set_message_count

`set_message_count [message_ID | { error | warning | info | info_msg }] integer`

Sets the count for the specified `message_ID` or type of message (`error`, `warning`, `info`, or `info_msg`) to the value of *`integer`*. If *`integer`* is not specified, zero is assumed.

Options and Arguments

`error`
: Sets the count for message type `error`.

`info`
: Sets the count for message type `info`.

`info_msg`
: Sets the count for message type `info_msg`.

`integer`
: Sets the count value of the *`message_ID`* or message type.
Default: `0`

`message_ID`
: Specifies the ID of the message whose count will be changed.

`warning`
: Sets the count for message type `warning`.

Example

The following command resets the count on the `error` type of messages to zero.

```
> set_message_count error
```

Related Information

`get_message_count`

`set_message_verbosity`

set_message_verbosity

```
set_message_verbosity message_ID {on | off | 0-9}
```

Controls the level of verbosity of the messages generated by `ac_shell` in reporting information, warnings, and errors.

You can assign a new verbosity level to each message to filter the `ac_shell` messages given out by the `ac_shell`. The messages displayed by the `ac_shell` have a message ID attached to them. These IDs are used to assign new message verbosity levels. Each message has a default verbosity level assigned to it.

All fatal errors have a verbosity level of 0, 1 or 2. It is recommended that message verbosity level be set greater than 2.

Note: If message verbosity is set to 9 then it will override all other messages.

Options and Arguments

`message_ID`
: Specifies the ID of the message whose verbosity level will be changed.

`on | off | 0-9`
: Sets the verbosity level of the message. The message can be turned `on`, `off`, or set as a number between `0` and `9`. A lower number implies that more messages will be reported.

Examples

- The following command turns off "# delays not supported" warnings.

  ```
  > set_message_verbosity VLOGPT-035 off
  ```

- The following command sets "range for parameters ignored" warnings to verbosity 7.

  ```
  > set_global message_verbosity CDFG-345 7
  ```

Related Information

`set_global message_verbosity_level`

set_path_group_options

```
set_path_group_options group_name [-effort {none | low | medium | high}]
     [-target_slack float] [-all_end_points]
```

Specifies which do_xform_optimize_slack options to apply to the named group.

Options specified to the do_xform_optimize_slack command will apply only to groups that have no set_path_group_options specified and to the null group (the set of paths that do not belong to a named group).
Default: medium

Options and Arguments

-all_end_points

Without this option, optimization only works on the worst endpoint in the path group and gives up if it cannot improve that endpoint. If this option is used, when optimization has trouble optimizing endpoint 1, it does not skip this group. Instead it optimizes the next critical endpoint in that group.

This option works in conjunction with the -target_slack value.

-effort {none | low | medium | high}

Controls the CPU time spent during the timing optimization step.

high
Further improvements by applying various time- consuming algorithms.

low
Operation is done quickly to meet requirements through easy optimization steps.

medium
Extra time is spent searching for alternate mappings and structures to meet all the constraints. This is the default.

none
No slack optimization is done on the named group.

group_name

Specifies the name of the group to be optimized with the options specified in this command. Required argument.

-target_slack *float*

Specifies the worst slack allowed for any path in the group. Optimization proceeds until all paths in the group have slack equal to or greater than the target slack.
Default: 0.0.

You can specify negative slack to compensate for overconstraining.

Examples

- The following command directs optimization to proceed with various algorithms until the worst slack in the group is equal to or greater than -1ns and to continue until all endpoints in the group with slack less than -1ns are optimized:

```
set_path_group_options GRPA -effort high -target_slack -1 -all_end_points
```

Related Information

report_path_group_options

reset_path_group

set_path_group

Using Path Groups for Optimization in the *Common Timing Engine (CTE) User Guide*.

set_port_property

`set_port_property -boundary_optimization {true | false}` *`port_list`*

Sets the specified property on all ports in the port list.

Options and Arguments

`-boundary_optimization {true | false}`
: Specifies whether the specific ports of specific modules listed in *`port_list`* are to be kept isolated from boundary optimization (constant propagation and so forth).
Default: `true`

`port_list`
: Specifies the names or IDs of module ports that you want to keep isolated from boundary optimization.

Related Information

`do_xform_propagate_constants`

set_register_type

```
set_register_type [-exact] [-latch example_latch_cell_name]
     [-flip_flop example_flipflop_cell_name] register_instance_id_list
```

Specifies the latch or flip-flop library cell type for certain register instances in the design. A latch or flip-flop cell type is represented by an example latch or flip-flop; any latch or flip-flop that has the same sequential characteristics as the example latch or flip-flop is considered that type. A latch type, a flip-flop type, or both can be specified. The subsequent technology mapping enforces mapping to register cells of the same type for the specified register instances.

The `set_register_type` command is "sticky" in the sense that subsequent mapping and unmapping/remapping enforce mapping to the specified register type, unless command `reset_register_type` is issued or another `set_register_type` command is issued on the same register instances.

Example latch or flip-flop cells cannot be multi-bit register, even if the global `map_to_multibit_registers` is turned on.

The register type is ignored in a DFT scan flow where register instances will be mapped to appropriate scan registers based on the chosen scan style.

Options and Arguments

`-exact`
: Enforces an exact mapping to the example latch or flip-flop cell.

`-flip_flop`
: Enforces mapping to flip-flop cells of the same type as the example flip-flop.

`-latch`
: Enforces mapping to latch cells of the same type as the example latch.

`register_instance_id_list`
: Specifies the list of object IDs for the register instances.

Examples

- The following command enforces mapping to latch cells of the same type as LD1 and flip-flop cells of the same type as FD1 for all register instances in the design:

```
> set_register_type -latch LD1 -flip_flop FD1 [find -registers *]
```

- The following command enforces an exact mapping to FD1 for all register instances whose names match Q_reg_*.

```
> set_register_type -exact -flip_flop FD1 [find -registers Q_reg_*]
```

Related Information

reset register type

set global map to multibit registers

set_table_style

```
set_table_style -name table_name [-reverse_rows] [-major_sort integer]
     [-minor_sort integer] [-max_widths list_of_integers]
     [-min_widths list_of_integers] [-indent integer]
```

Disables printing of specific columns and allows for specifying the minimum and maximum size of each column. It allows for reversing the data, for controlling the left indent, and for controlling the sorting of the columns.

Note: Do *not* include a blank space between commas "," when specifying widths. If you are specifying a value for the width(s), make sure there is *no* leading or trailing blank space.

Options and Arguments

`-indent integer`
: Specifies the number of spaces to leave on the left for that particular table

`-major_sort integer`
: Specifies the column to be used in the major sort. If the column number is negative, it reverses the sort.

`-max_widths list_of_integers`
: Specifies the maximum widths of each column for example, the option `{0,5,7}` specifies that first column is to be zero width (deleted) and the second column is to remain and the previously specified width, or the default width if a previous width was not specified. Each column resizes itself to be between the range of the maximum and minimum widths specified for that column based on the data in that column. See examples.

`-min_widths`
: Specifies the minimum column width allowed. Each column resizes itself to be between the range of the maximum and minimum widths specified for that column based on the data in that column.

`-minor_sort integer`
: Specifies the column to be used in the minor sort. It applies to ALL rows that have the same values in the `major_sort` column. If the column number is negative, it reverses the sort.

`-name` *`table_name`*

Specifies the table that is being controlled. Each table has one of the following unique names:

```
area_slack1
area_slack2
inferred_register
pa_inst_int_pwr_rep_table
pa_inst_int_pwr_rep_table1
pa_inst_leak_pwr_rep_table
pa_inst_net_pwr_rep_table
pa_inst_fall_slew_table
pa_inst_rise_fall_slew_table
pa_inst_rise_slew_table
pa_module_fall_slew_table
pa_module_rise_fall_slew_table
pa_module_rise_slew_table
pa_tc_stats_table
popt_report_clock_gating_table
report_area_cell
report_area_cell_hier
report_area_cell_summary
report_area_summary
report_case_stats
report_comb_loop
report_library
report_library_cell
report_library_header
report_library_operating_condtions
report_power_inst
report_power_net
report_timing
report_timing_header
report_timing_summary
report_timing_prologue
report_fanin
report_fanout
report_fsm
report_fsm_transition
report_fsm_encoding
```

`-reverse_rows`

Reverses the data rows. This is useful, for example, if you want to trace back from an end point and the default report traces forward from the end point.

Examples

- The following command alters the table style in the main `report_library` table, allowing columns 1-3 (cell name, type, cell area) to be the default size, and adjusting column 10 (the function column) to have a maximum width of 40. The table will be sorted based on column 3, but when two cells have identical area, these will be sorted based on column 1 (the cell name):

  ```
  set_table_style -name report_library_cell -max_widths {,,,0,0,0,0,0,0,40}
  -major_sort 3 -minor_sort 1
  ```

  This disable columns 4-9 in the main `report_library` table:

  ```
  dont_modify flag
  dont_utilize flag
  footprint
  outputs
  inputs
  inouts
  ```

- The following command resizes the first column to a maximum range of 70 and specifies a default width of 50 for the second column. The default for the maximum width is 50 and the default for the minimum width is 5:

  ```
  > set_table_style -name report_timing -max_widths {70,}
  ```

  Note: Do *not* include a blank space between commas "**,** " when specifying widths. If you are specifying a value for the width(s), make sure there is *no* leading or trailing blank space.

- The following command tells the `report_timing` command to list the hierarchical pin and arrival times in the report:

  ```
  > set_global report_timing_format {hpin arrival}
  ```

- The following command specifies a default width of 50 for the first and second column:

  ```
  > set_table_style -name report_timing -max_widths {,}
  ```

- The following command specifies a default width of 50 for the first and second column:

  ```
  > set_table_style -name report_timing -max_widths {,}
  ```

Related Information

report_analysis_coverage

report_cell_instance -power

report_power

report_cell_instance_timing

report_clocks

report_ports

report_slew_for_power_analysis

report_timing

report_tc_stats

set_global line_length

set_unconnected

`set_unconnected` *`port_list`*

Sets the ports listed as unconnected. The optimization phase will remove the driving logic.

Note: This command is applicable for top-level output ports only.

This command will set an attribute, `_propagated_value`, on the ports. You can remove the attribute by using the `delete_attribute` command.

Options and Arguments

`port_list`

Specifies the list of port names to disconnect from.

Related Information

`delete_attribute`

`set_current_module`

`set_logic0`

`set_logic1`

set_vhdl_library

`set_vhdl_library` *`library_name directory_name`*

Defines a logical VHDL library name and the name of the directory used to store the analyzed VHDL units. The following libraries are currently preset on starting up `ac_shell`:

AMBIT	*`install_dir`*`/BuildGates/version/lib/tools/vhdl/1993/ambit`
IEEE	*`install_dir`*`/BuildGates/version/lib/tools/vhdl/1993/ieee`
STD	*`install_dir`*`/BuildGates/version/lib/tools/vhdl/1993/std`
TEMP	`/tmp/?/TEMP`
WORK	TEMP

The VHDL library `WORK` must be mapped to an existing logical library. To change the mapping, use `set_vhdl_library WORK` *`library_name`*.

Options and Arguments

`directory_name`
: The physical path name to the directory where VHDL units are stored.

`library_name`
: The logical name of the VHDL library

Examples

- This example shows how you can define libraries where the VHDL source file has a library(use) clause referring to libraries other than the five listed above, you can use this command to define those libraries, for example:

  ```
  > set_vhdl_library MYLIB /home/user_name/libs/mylib
  ```

- VHDL units can be analyzed into that library by reading source VHDL files.

  ```
  > read_vhdl -library MYLIB pack.vhd
  ```

  The analyzed files are sent to `/home/`*`user_name`*`/libs/`*`mylib`*.

  It is useful to explicitly specify libraries with the `set_vhdl_library` command because the VHDL units (such as packages) analyzed into such a library are not deleted when you exit from the synthesis tool. The VHDL units can be reused in a subsequent invocation of the synthesis tool.

If an attempt is made to analyze files into a library that has not been created with the `set_vhdl_library` command, a default directory for that library is created in the `/tmp` area (the scratch area pointed to by the environment variable `AMBIT_TMP_DIR`). However, such a library (and the corresponding directory) will be deleted when you exit from the synthesis tool.

Note: Two different libraries should not be mapped to the same physical directory.

Related Information

`read_vhdl`

`report_vhdl_library`

unalias

`unalias` *`name`*

Removes an alias created by the alias command

Options and Arguments

`name`

Specifies the name of an existing alias

Related Information

`alias`

write_adb

```
write_adb [-all] [-no_assertions] [-no_acl] [-hierarchical |-no_hierarchical]
     file_name
```

Writes design data stored by `ac_shell` to the database using the Ambit Synthesis database (ADB) file format. Writes out the entire hierarchy, automatically using the `-hierarchical` option with the command so you do not have to specify it. In ADB format, the data can be quickly loaded to perform further synthesis or analysis of data when running `ac_shell`. The `read_adb` command loads data from the `.adb` file into the database.

Tip

If the software terminates abnormally, an ADB recovery file is created in the /TMP file. To use a directory other than the default /TMP, use the environment variable `AMBIT_TMP_DIR`. For example: `setenv AMBIT_TMP_DIR ./adb_dir`.

When generating an ADB using the `write_adb` command, all globals are written to a special section in the `.adb` file. When reading an `.adb` file, you have the option of reading all of the globals back into the current design using the `read_adb -restore_globals` command.

Options and Arguments

`-all`
: Dumps all of the modules in the database.
Default: Dumps the current module and the hierarchy below.

`file_name`
: Specifies the name of the ADB output file.

`-hierarchical | -no_hierarchical`
: Writes out only the current module, if `-no_hierarchical` is specified.
Default: `-hierarchical`.

`-no_acl`
: Does not dump ACL modules.

`-no_assertions`
: The assertions are not included in the ADB that is currently being written out. By default the assertions are written into all ADBs and all the assertions read in are applied. When using the `read_adb` command, this flag causes the assertions in the ADB

being read in to be ignored.
Default: Writes assertions into all ADBs and to apply all the assertions being read in.

Note: Using this option will cause placement information to be ignored (not written out).

Example

This command writes out an `.adb` file of the current module, dumping it into a file called `mydesign.adb`:

```
> write_adb -no_hierarchical mydesign.adb
```

Related Information

`read_adb`

`write_verilog`

write_edif

`write_edif [-hierarchical] file_name`

Writes out designs in EDIF format.

Options and Arguments

`-hierarchical`

If the `-hierarchical` option is not specified, EDIF is only written out for the current module.

`file_name`

Specifies the name of the EDIF output file.

Example

The following example assumes that the current module is `TOP` which has the following hierarchical structure

```
report_hierarchy
     TOP(g)
      MIDDLE(g)
         BOTTOM(g)
```

The following command writes out an EDIF description of all the three modules: `TOP`, `MIDDLE`, and `BOTTOM`:

```
> write_edif -hierarchical out.edif
```

Related Information

`read_verilog`

`set_global edifout_*`

Refer to the *EDIF Interface* chapter in the *HDL Modeling for BuildGates Synthesis* manual.

write_globals

`write_globals [-modified] [-group` *`group_name`*`] [`*`file_name`*`]`

Writes out all globals in the system in a format that can be read back in through the `bg_shell` Tcl interface.

Options and Arguments

`file_name`

The name of the globals output file. Filename must be the last argument specified. If you do not specify a filename, the globals are written to the screen.

`-group` *`group_name`*

Includes only global variables belonging to the specified group. Valid values for *`group_name`* include `hdl`, `aware`, `ta`, `opt`, `pks`, `dft`, `dist`, `dcn`, `edif`, `misc`, and `ui`.

`-modified`

Includes only global variables that have been changed from their default value.

Examples

- The following command writes out all global variables that have been modified:

```
> write_globals -modified my_globals.tcl
```

- The following command reads the globals file back into the tool:

```
> source my_globals.tcl
```

Related Information

`report_globals`

write_verilog

`write_verilog [-hierarchical] [-equation] verilog_file_name`

Writes out a netlist stored in the database in Verilog format. The netlist is generated by the `do_build_generic` command or `do_optimize` command.

You can write out the Verilog netlist in a compressed Verilog file in GNU zip format if the specified output file ends in the .gz suffix. No special option is needed to write out a gzipped file. For example: `write_verilog verilog_file_name`.gz.The `read_verilog` command automatically reads compressed outputs.

If the netlist is written out after the `do_build_generic` command it contains instances of ATL and XATL cells. For gate level verification of this netlist, the Ambit Synthesis Verilog Library must be used for simulation.

If the netlist is written out after you use the `do_optimize` command then it contains instances of cells in the target technology library. To verify this netlist, a Verilog library of the target technology must be used for simulation.

Note: If you get assign statements in the output you must use the command `do_xform_fix_multiport_nets` or `set_global fix_multiport_nets`.

Options and Arguments

`-equation`

Writes the verilog output without reference to atl, xatl, or library cells. Instead, each combinational or tristate cell instance is represented with an equation, and a simulation model for each sequential cell is written out.

`-hierarchical`

Writes the netlist out as a hierarchical netlist. If this option is not used, the command writes out a netlist for the current module and any implied hierarchy created by BuildGates Synthesis, such as ACL modules.

`-rtl`

Writes RTL-style netlists for AmbitWare arithmetic components and multiplexer modules. This style can quickly simulate gate-level netlists and integrate with downstream FPGA synthesis tools.

verilog_file_name
Specifies the file in which the Verilog netlist will be saved.

Attributes

set_global hdl_verilog_out_columns

set_global hdl_verilog_out_compact

set_global hdl_verilog_out_source_track

set_global hdl_verilog_out_use_supply

Examples

- The following command saves the hierarchical netlist in the file `counter.v.net`:

  ```
  write_verilog -hierarchy counter.v.net
  ```

- The following command writes out a gzipped (compressed) file by appending the `.gz` extension to the file

  ```
  write_verilog addr.v.gz
  ```

Related Information

do_build_generic

do_optimize

read_verilog

write_assertions

write_vhdl

`write_vhdl [-hierarchical] [-equation] [-no_wrap] vhdl_file_name`

Writes out a VHDL netlist. The resulting netlist preserves the original VHDL types of the ports of the current module being written. As the internal representation of the module is in terms of bits, any required wrapper functions to convert between the port types and the internal bit type are also generated. This enables you to do a comparison of the netlist with the original VHDL description through simulation or verification.

You can write out the VHDL netlist in a compressed VHDL file in GNU zip format if the specified output file ends in the .gz suffix. No special option is needed to write out a gzipped file. For example: `write_vhdl vhdl_file_name.gz`. The `read_vhdl` command automatically reads compressed outputs.

All of the submodules will have equivalent `std_logic` or `std_logic_vector` ports. If the module being written did not originate from a VHDL source file, then all ports are written out in terms of equivalent `std_logic` ports.

Options and Arguments

`-hierarchical`
: Writes the netlist out as a hierarchical netlist. If this option is not used, the command writes out a netlist for the current module and any implied hierarchy created by BuildGates Synthesis, such as ACL modules.

`-equation`
: Writes the vhdl output without reference the atl, xatl, or library cells. Instead, each combinational or tristate cell instance is represented with an equation, and writes out a simulation model for each sequential cell.

`-no_wrap`
: Writes the current module with equivalent `std_logic` or `std_logic_vector` representation (that is, do not preserve the original VHDL port types).

`vhdl_file_name`
: Specifies the file to which the VHDL netlist will be written.

Examples

The following examples refer to the following hierarchy of VHDL entities:

```
report_hierarchy
-TOP(g)
-MIDDLE(g)
-BOTTOM(g)
```

- In the following example, the entity `TOP` will have its port types preserved while `MIDDLE` and `BOTTOM` will have `std_logic` ports.

```
> set_current_module MIDDLE
MIDDLE

> report_hierarchy -MIDDLE(g) -BOTTOM(g)
> write_vhdl -hierarchical netlist.vhd
```

- The following command specifies both entities `MIDDLE` and `BOTTOM` will have `std_logic` ports.

```
> write_vhdl -hierarchical -no_wrap netlist.vhd
```

- The following command writes out a gzipped (compressed) file by appending the `.gz` extension to the file:

```
write_vhdl addr.vhd.gz
```

Related Information

`read_vhdl`

`set_global hdl_vhdl_write_version`

2

CTPKS Commands

This chapter describes the CTPKS commands and attributes available with the Cadence® Physically Knowledgeable Synthesis (PKS) tool.

Note: Because CTPKS works simultaneously on the logic design and the physical design, most CTPKS commands are only relevant when used at the top level (corresponding to the entire physical design).

- ct_no_leaf
- ct_preserve
- ct_preserve_tree
- ct_specify_padding

do_build_clock_tree

```
do_build_clock_tree [-pin list_of_pins] [-save_structure filename]
     [-use_structure filename] [-noplace] [-no_gated {rising | falling}] [-power]
     [-fast] [-move_gated][-clock_model list_of_files] [-pad_after_gated]
     [-use_family list_of_families] [-use_buffers buffer_list]
     [-use_inverters inverter_list] [-effort {low | medium | high}]
     [-merge_delay_chains] [-no_isolation_buffer] [-fix_drv_on_excluded_pins]
     [-no_verbose]
```

Generates a clock tree using parameters set by the `set_clock_tree_constraints` command. If no constraints are set, the command has no effect. This command requires that all the modules that contain a clock net are uniquified. The command will fail otherwise.

Note: If no physical data is available (for instance, if you are using BuildGates Synthesis or BuildGates Extreme Synthesis), the `do_build_clock_tree` command will work in wireload model mode.

When you invoke this command, the following changes are made to the database:

- Buffers, inverters, or both are added.
- An isolation buffer is automatically inserted between the output of any drive boosters and any excluded clock tree pins to reduce the capacitance at the drive boosters. This isolation buffer prevents slew degradation at the drive booster output when there is a large number of excluded clock tree pins.
- The clock propagation mode is set to propagated.
- The `+ USE CLOCK` property is added in a DEF on all clock nets.
- The `+ SHIELDNET` property is propagated to all clock nets if it is present on the subtree root net.
- The `NONDEFAULTRULE`, and layer usage table information is propagated to all clock nets if it is present on the subtree root net.

Note: If you use a general clock model, the resulting shielding, `NONDEFAULTRULE`, and layer usage table information come from the general clock model file. See the General Clock Model section in the *BuildGates Synthesis User Guide* for more details about the general clock model.

Options and Arguments

`-clock_model` *`list_of_files`*

Loads general clock model definition files. Each resulting tree will match one family defined within these files. When the `-clock_model` switch is used, the global variable `ctpks_default_global_clock_model_file` is ignored. The use of a general clock model is not compatible with the use of a structure file. See the General Clock Model section in the *BuildGates Synthesis User Guide* for more details about the general clock model.

`-effort {low | medium | high}`

Specifies the level of CPU usage when building a clock tree. When you specify `-low`, CPU usage is low, but the quality of results (QoR) may be affected. When you specify `-high`, CPU usage is higher, but the QoR is better.
Default: `low`

`-fast`

Tells the tool to do a quick job. It picks only the fastest cell from all available buffers and inverters and builds a clock tree with it. The number of internal trials is also reduced. This option is useful to check if a clock tree can be built with the constraints you have chosen. In most cases, the resulting clock tree should not be used as is.

`-fix_drv_on_excluded_pins`

Instructs CTPKS to attempt to fix DRVs on such pins. Unless specified, CTPKS does not fix design rule violations on excluded pins.

`-merge_delay_chains`

Reduces the number of clock tree buffers (padding cells) used to balance different subtrees. Several subtrees are gathered and delayed at the same time using the same chain of clock tree buffers.

Note: This option cannot be used in conjunction with the `-save_structure` option. If you specify both options, CTPKS will issue an error message.

`-move_gated`

Specifies that the gated components can be moved to optimize performance.

`-no_gated {rising | falling}`

Specifies that the clock tree stops at the first cells downstream from the root that are neither buffer nor inverter. If the active edge at this point is not specified, the edge given on the command line is used.

`-no_isolation_buffer`

Prevents the insertion of isolation buffers.

`-no_verbose`

Disables message generation.

`-noplace`

Specifies that incremental Qplace will not be invoked.

Note: Some inserted buffers or inverters may be illegally placed.

`-pad_after_gated`

Forces the padding (buffers or inverters added in order to delay a particular point) to be inserted after the gating components. This switch will be ignored if the `ct_specify_padding` attribute is set on the gated component instance.

`-pin` *`list_of_pins`*

Specifies the pin of an instance or input port as the root of the clock tree. Without `-pin`, CTPKS will run on all clock trees previously specified with the `set_clock_tree_constraints` command.

`-power`

Adds a new criteria when CTPKS selects the best clock-tree topology. When CTPKS finds several topologies which meet constraints with a higher priority (DRVs, max delay, skew, and so on), it calculates the power usage of these topologies and chooses the one with the minimum power consumption. The consumption is measured on all clock objects, including buffers inverters, and registers.

Note: This option requires BuildGates Extreme, which includes the low power feature.

`-save_structure` *`filename`*

Writes the clock tree structure to a file whose syntax is identical to the syntax of `define_structure` on page 333. If you use a general clock model, the resulting net characteristics as defined in the general clock model will not be saved in the structure file.

Note: This option cannot be used in conjunction with the `-merge_delay_chains` option. If you specify both options, CTPKS will issue an error message.

`-use_buffers` *`buffer_list`*

Specifies the list of buffers to be used. If you do not want any buffer to be used, use empty braces, (`-use_buffers` {}).

`-use_family` *`list_of_families`*

Defines the list of families to be tried from the general clock model definition files given either by the `-clock_model` switch or by the `ctpks_default_global_clock_model_file` global variable. See the General Clock Model section in the *BuildGates Synthesis User Guide* for more details about the general clock model.

`-use_inverters` *`inverter_list`*

Specifies the list of inverters to be used. If you do not want any inverters to be used, use empty braces (`-use_inverters` {}).

`-use_structure` *`filename`*

Loads the clock tree structure you want to use from a file. CTPKS applies the semi-automatic mode to this structure. If the `from_pin` or `from_iopin` specified in the file matches the clock pin being treated by CTPKS, the corresponding tree structure is used. The syntax is similar to the syntax of `define_structure` on page 333. The use of a structure file is not compatible with the use of a general clock model.

Examples

- In the following example, the first command sets minimum and maximum insertion delays, maximum skew, and maximum transition on the clock tree starting from the port clk. The second command builds the clock tree using buffers `CLKBUFX1` and `CLKBUFX2` and no inverters. The third command builds the entire clock tree and saves the structure to a file named `tc1.struct`.

```
> set_clock_tree_constraints -min_delay 0.1 -max_delay 1.0 -max_skew 0.3
-max_tree_transition 0.3 -pin [find -port clk]


> do_build_clock_tree -use_buffers {CLKBUFX1 CLKBUFX2} -use_inverters {}


> do_build_clock_tree -save_structure tc1.struct
```

- The following is an example of `do_build_clock_tree` output:

```
Warning: Clock Propagation Mode is ideal <CTPKS-105>
Info: Setting clock propagation mode to propagated. <CTPKS-104>
+-------------------------------------------------------------------------+
|           Clock tree root clk           |  Clock Tree | Actual |  Area   |
|                                         | Constraints |        |         |
|-----------------------------------------+-------------+--------+---------|
|             Number of Inserted Instances |             |      3 | 134.400 |
|       Max. transition time at tree pins |       0.300 |  0.131 |         |
|       Min. insertion delay to leaf pins |       0.100 |  0.144 |         |
|       Max. insertion delay to leaf pins |       1.000 |  0.211 |         |
|              Max. skew between leaf pins |       0.300 |  0.067 |         |
|                     Number of Violations |             |      0 |         |
|                        Number of Buffers |             |      0 |   0.000 |
|                     Number of Inverters |             |      3 | 134.400 |
| Number of gated/pad/preserved Instances |             |      0 |   0.000 |
|                Total Number of Instances |             |      3 | 134.400 |
|                  Number of Excluded Pins |             |      0 |         |
|                      Number of Leaf Pins |             |      4 |         |
+-------------------------------------------------------------------------+
```

Related Information

do_uniquely_instantiate

report_clock_tree

report_clock_tree_violations

set_clock_tree_constraints

do_build_physical_tree

```
do_build_physical_tree list_of_pins_or_ports[-noplace] [-fast]
     [-clock_model list_of_files] [-use_family list_of_families]
     [-use_buffers buffer_list] [-use_inverters inverter_list]
     [-save_structure filename] [-use_structure filename]
     [-effort {low | medium | high}]
```

Creates a balanced tree starting from the given pins or ports specified by *list_of_pins_or_ports*. All buffers and inverters in the downstream tree are removed and replaced by the new tree. The tree stops at any cell that is not a buffer or an inverter. The attributes ct_preserve, ct_leaf, ct_excluded, ct_preserve_tree, and ct_dont_utilize are taken into account when building the tree. When there is no physical tree constraint at root, the smallest alternative tree (respecting design rules) is kept.

Unlike to the do_build_clock_tree command, do_build_physical_tree does not propagate 'use clock' attributes.

Note: If no physical data is available (for instance, if you are using BuildGates Synthesis or BuildGates Extreme Synthesis), the do_build_physical_tree command will work in wireload model mode.

Important

This command can only be used with pins and ports, not net names or net IDs. It cannot be used on constant pins or pins specified as dont_modify.

Options and Arguments

-clock_model *list_of_files*
: Loads general clock model definition files. Each resulting tree will match one family defined within these files. The global variable ctpks_default_global_clock_model_file is never taken into account for do_build_physical_tree. The use of the general clock model is not compatible with the use of a structure file. See the General Clock Model section in the *BuildGates Synthesis User Guide* for more details about the general clock model.

-effort {low | medium | high}
: Specifies the level of CPU usage when building a clock tree. When you specify -low, CPU usage is low, but the quality of results (QoR) may be affected. When you specify -high, CPU

usage is higher, but the QoR is better.
Default: `low`

`-fast`

Tells the tool to do a quick job. It picks only the fastest cell from all available buffers and inverters and builds a clock tree with it. The number of internal trials is also reduced.

`list_of_pins_or_ports`

Specifies the names or IDs of pins or ports from which to start physical tree.

`-noplace`

Specifies that incremental Qplace will not be invoked.

Note: Some inserted buffers or inverters may be illegally placed.

`-save_structure` *`filename`*

Writes the clock tree structure to a file whose syntax is identical to the syntax of `define_structure` on page 333.

`-use_buffers` *`buffer_list`*

Specifies the list of buffers to be used. If you do not want any buffer to be used, use empty braces, (`-use_buffers` {}).

`-use_family` *`list_of_families`*

Defines the list of families to be tried from the general clock model definition files given by the `-clock_model` option.

`-use_inverters` *`inverter_list`*

Specifies the list of inverters to be used. If you do not want any inverters to be used, use empty braces (`-use_inverters` {}).

`-use_structure` *`filename`*

Loads the clock tree structure you want to use from a file. CTPKS applies the semi-automatic mode to this structure. If the `from_pin` or `from_iopin` specified in the file matches the clock pin being treated by CTPKS, the corresponding tree structure is used. The syntax is similar to the syntax of `define_structure` on page 333.

Examples

- The following command builds a balanced clock tree starting from the pin `in1`:

```
> do_build_physical_tree D1
```

- The following is an example of output from the `do_build_physical_tree` command:

```
+-------------------------------------------------------------------------+
|          Physical tree root D1          |  Physical   | Actual |  Area  |
|                                         |    Tree     |        |        |
|                                         | Constraints |        |        |
|-----------------------------------------+-------------+--------+--------|
|            Number of Inserted Instances |             |      2 | 89.600 |
|       Min. insertion delay to leaf pins |       0.000 |  0.111 |        |
|       Max. insertion delay to leaf pins |       1.000 |  0.112 |        |
|             Max. skew between leaf pins |       0.500 |  0.001 |        |
|                    Number of Violations |             |      0 |        |
|                       Number of Buffers |             |      0 |  0.000 |
|                     Number of Inverters |             |      2 | 89.600 |
| Number of gated/pad/preserved Instances |             |      0 |  0.000 |
|               Total Number of Instances |             |      2 | 89.600 |
|                 Number of Excluded Pins |             |      0 |        |
|                     Number of Leaf Pins |             |      2 |        |
+-------------------------------------------------------------------------+
```

Related Information

`report_clock_tree`

`Attributes`

`set_clock_tree_constraints`

do_xform_optimize_clock_tree

```
do_xform_optimize_clock_tree {-asymmetric | -symmetric |
     -mix_symmetric_and_asymmetric | -rebalance_subtrees | -merge_delay_chains |
     -resize_gating_cells | -incremental [-effective_skew] |
     -useful_skew [-decreasing_slack_order | -increasing_slack_order]
     [-do_not_allow_slack_degradation] [-slack_limit] [-ignore_hold]
     [-balance_output_neg_slack] } [-no_gated {rising | falling}] [-no_drv]
     [-fix_drv_on_excluded_pins] [-use_buffers buffer_list]
     [-use_inverters inverter_list] [-smaller_cells_only]
     [-no_degradation_allowed] [-pin list_of_pins] [-use_repeaters]
```

Applies one or more transforms to clock trees. The transforms that are applied depend on the given argument. The applied transforms try to optimize the clock tree(s) according to the criteria to minimize the following in decreasing priority order:

- Number of design rule violations
- Worst maximum latency violation
- Worst skew violation
- Worst minimum latency violation
- Area

Depending on the transforms that are used, this command may produce the following modifications to the database: swap, insertion or deletion of buffers and inverters of the clock tree(s), placement modification of the buffers and inverters of the clock trees, swap of gated components of the clock trees.

Note: If no physical data is available (for instance, if you are using BuildGates Synthesis or BuildGates Extreme Synthesis), the `do_xform_optimize_clock_tree` command will work in wireload model mode.

Options and Arguments

`-asymmetric`
: Optimizes the subtrees of the clock tree(s) by resizing the buffers without keeping the symmetry of the subtrees: different buffers are allowed at the same level.

`-fix_drv_on_excluded_pins`
: Instructs CTPKS to attempt to fix DRVs on such pins. Unless specified, CTPKS does not fix design rule violations on excluded pins.

`-incremental [-effective_skew] [-no_degradation_allowed]`

Resizes one or more buffers that are producing the worst violation. If the worst violation is reduced enough, the transform applies recursively on the next worst violation.

When the worst violation cannot be improved any further by changing the buffer producing it, the incremental mode algorithm continues to try to improve timing by allowing some degradation in the worst violation, as long as successive optimizations result in further improvement.

`-effective_skew`
Works on effective skew minimization instead of global skew minimization.

`-no_degradation_allowed`
Stops optimizations when the worst violation cannot be improved any further by any change to the buffer producing it.

In contrast to the default behavior, optimization stops as soon as changes start degrading timing, regardless of whether successive optimizations would result in further improvement.

`-merge_delay_chains`

Reduces the number of clock tree buffers inserted to balance subtrees. Several subtrees are gathered and the same chain of buffers is used to delay them. Use this option in conjunction with the `-rebalance_subtrees` option.

`-mix_symmetric_and_asymmetric`

Applies the `-symmetric` option to symmetrical subtrees and `-asymmetric` option to asymmetrical subtrees.

`-no_drv`

Instructs CTPKS to not check for design rule violations.

`-no_gated {rising | falling}`

Specifies that the clock tree stops at the first cells downstream from the root that are neither buffer nor inverter. If the active edge at this point is not specified, the edge given on the command line is used.

`-pin` *`list_of_pins`*

Specifies the list of root pins of clock trees to work on. Without the `-pin` switch, the `do_xform_optimize_clock_tree` command will run on all clock trees previously specified with the `set_clock_tree_constraints` command.

`-rebalance_subtrees`

Balances the subtrees of the clock tree(s) by inserting padding buffers.

`-resize_gating_cells`

Optimizes the subtrees of the clock tree(s) by resizing the gated component of the clock tree(s).

`-smaller_cells_only`

Specifies that cell substitutions be performed only with cells the same size as the original or smaller.

`-symmetric`

Optimizes the subtrees of the clock tree(s) by resizing the buffers by level in order to keep the symmetry. If the subtrees are not symmetrical, this option does not optimize the clock tree.

Note: CTPKS does not build symmetrical clock trees when gating components are involved. CTPKS builds symmetrical subtrees from the output of each gating component, but then it adds buffers (padding cells) in front of these gating components to balance between all of these subtrees. This results in a final clock tree that is non-symmetrical.

`-use_repeaters`

Repeater insertion is done to reduce design rule violations (DRVs) and transition (slew) violations. The maximum number of repeaters CTPKS places in a repeater chain is 10. It selects the length of the repeater chain based on the cost function. The length which reduces the maximum number of DRVs, without degrading the max-delay/skew/min-delay, is selected as the best length and the repeater chain with that length is inserted.

The various modes in XFORM in which repeater insertion is done is as follows:

Mode 1—Incremental (with `-incremental` option)

Mode 2—Asymmetric (with `-asymmetric` option)

Mode 3—Symmetric (with `-symmetric` option)

Mode 1—Incremental

In the incremental mode, XFORM tries to fix the most critical violation and when this is fixed, it goes to the next critical violation. Whenever the most critical violation is a slew violation, then repeaters are inserted to fix it.

Mode 2—Asymmetric

This flow is same as the asymmetric flow. After resizing the current instance, if the `-use_repeaters` option is used, the following occurs:

1. Gets the fanout pins of the current instance, that is the inputs that the current instance is driving.

2. Iterates over the fanout-trace pins

a. If any of the fanout-trace pins has a transition violation, then it tries to insert repeaters ahead of this `fanout_tracepin`

Mode 3—Symmetric

The current level of the subtree is checked if it has any transition violations. If the level has transition violations, that is if any of the instance in that level drives a sink which has transition violation, then a repeater chain is inserted for each instance in the current level. The length of the repeater chain is also same for all the instances in the current level of the sub-tree.

`-useful_skew`

Applies the `useful_skew` transform on the clock trees defined by the root clock pins. The `useful_skew` transform attempts to fix a setup violation on a flip-flop of the clock tree by adding buffers or inverters (and by extension, by adding delay) before the clock pin. The fix will occur if no additional violation is created.

This transform finds the best solution in terms of area or number of inserted instances, and does not introduce new clock tree violations (except for skew).

In the useful skew methodology, the clock arrival times on each modified flip-flop are meant to be different (the skew is required to be non-zero).

The default behavior is to correct the worst negative slack. Correction stops when the worst slack cannot be improved any further.

The behavior options for `-useful_skew` are as follows:

`-balance_output_neg_slack`
Checks that the slack degradation on the data path for all leaf registers connected to the currently optimized leaf register is no larger than the slack obtained for the optimized register.

For example, consider a shift register containing just two flip-flops, where the slack at the first register is -0.8 ns, and the slack at the connected register is -0.4. During useful skew optimization of the first register, the slack at the connected register can worsen. If optimization results in a slack of -0.6 ns at first register, the connected register will never have a slack worse than -0.6 ns with this option.

This option prevents the useful skew algorithm from looping when data feedback loops between leaf registers are present (data output of a leaf register connected through logic to data input of another leaf register in its fanin cone).

`-decreasing_slack_order`
Reduces all individual setup slacks, starting from the worst slack and continuing as long as there is fixable negative slack.

`-do_not_allow_slack_degradation`
Does not allow any slack degradation if you use one of the arguments specifying order.

`-ignore_hold`
Ignores early slack (hold slack) degradations for all leaf registers.

`-increasing_slack_order`
Reduces all individual setup slacks, starting from the flip-flop with the smallest slack and continuing as long as there is fixable negative slack.

`-random_slack_order`
Reduces all individual setup slacks, in random order, as long as there is fixable negative slack.

If you do not specify an order, this option allows slack degradations smaller than the worst slack.

`-slack_limit`
Specifies a slack value. If the slack of a flip-flop is above the slack limit value, the command will skip it.

`-use_buffers` *`buffer_list`*
Specifies the list of buffers to be used. If you do not want any buffer to be used, use empty braces, (`-use_buffers` {}).

`-use_inverters` *`inverter_list`*
Specifies the list of inverters to be used. If you do not want any inverters to be used, use empty braces (`-use_inverters` {}).

Examples

- The following command reoptimizes the clock tree starting at the root pin `CLK` using buffers `BUFX1` and `BUFX2`:

```
> do_xform_optimize_clock_tree -pin CLK -incremental -use_buffers {BUFX1
BUFX2} -use_inverters {}
```

- The next command corrects timing violations by applying a useful skew transform on the clock tree starting at the root pin `CLK` using buffers `BUFX1` and `BUFX2`:

```
> do_xform_optimize_clock_tree -pin CLK -useful_skew -use_buffers {BUFX1
BUFX2} -use_inverters {}
```

- The following is an example of the report generated when using the `-useful_skew` option:

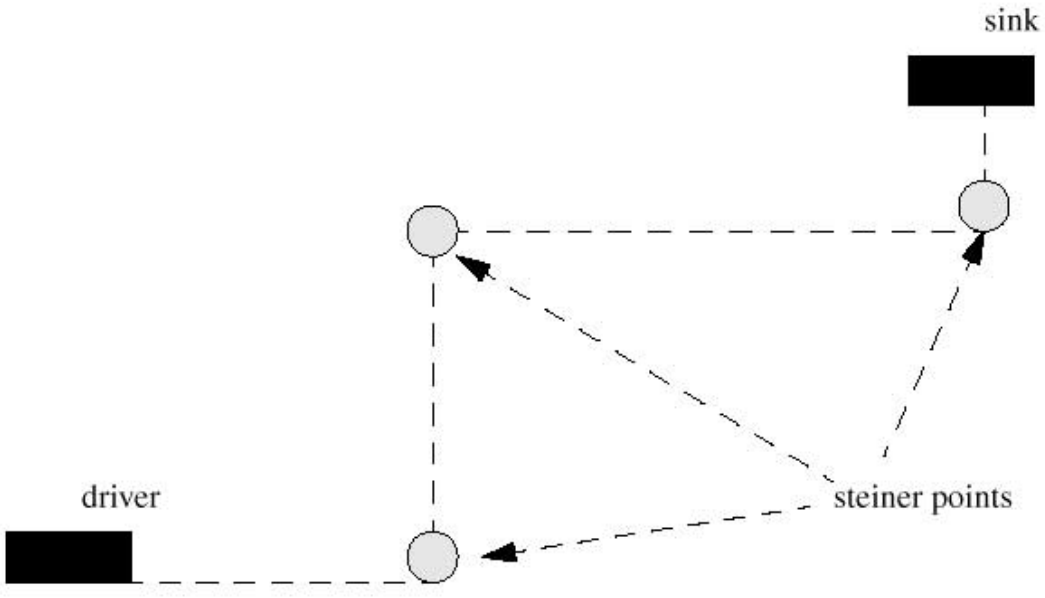
```
Info:    Useful skew : fixed slack on leaf pin dff1999/CP (start slack=-1.
00000000, end slack -0.72127247) using 2 instances <CTPKS-122>.
+-----------------------------------------------------------+
|  Report useful skew for root |   Before    |    After    |
|           pin clk            |  transform  |  transform  |
|------------------------------+-------------+-------------|
|     Number of negative slack |           2 |           2 |
```

```
|     Number of positive slack |        2000 |        2000 |
|     Number of adjusted slack |             |          31 |
|         Total negative slack | -1.50000000 | -1.46861553 |
|                Worst slack | -1.00000000 | -0.74734306 |
| Number of inserted instances |             |           2 |
+------------------------------------------------------------+
```

- The following example shows how repeaters are inserted.

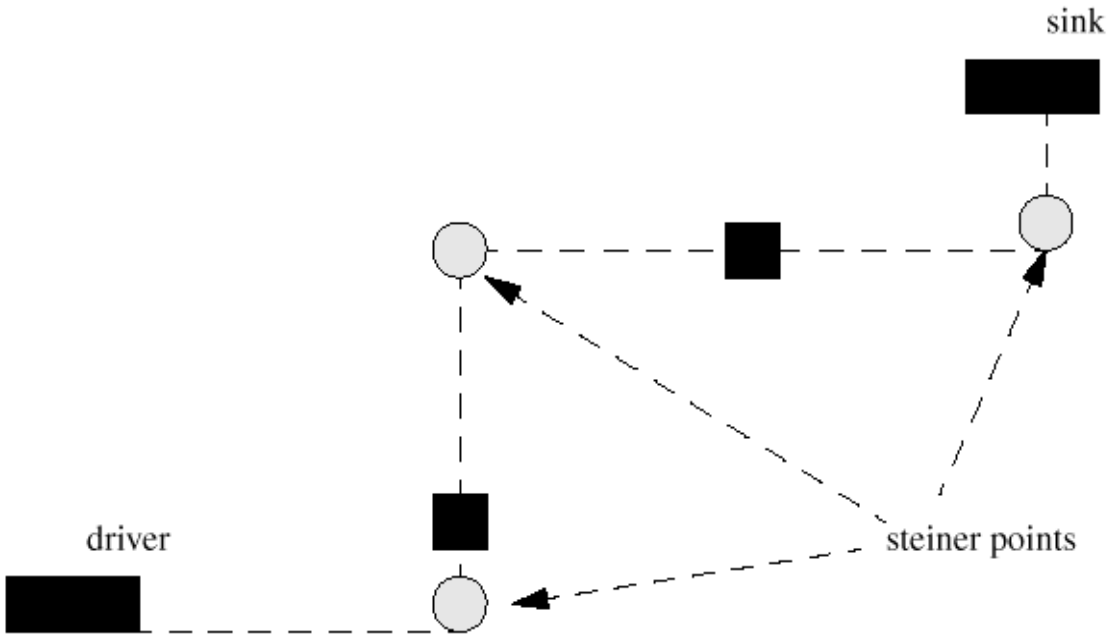
Assume that the driver and sink are placed as shown below and the transform got the Steiner points from the PKS as shown. Also, the celltype of the repeater is decided as follows:

```
If (celltype of sink is a buffer or inverter )
     then celltype of repeater is same as celltype of sink
else if ( celltype of driver is a buffer or inverter)
     then celltype of repeater is same as celltype of driver
else
     no repeater insertion is done.
```

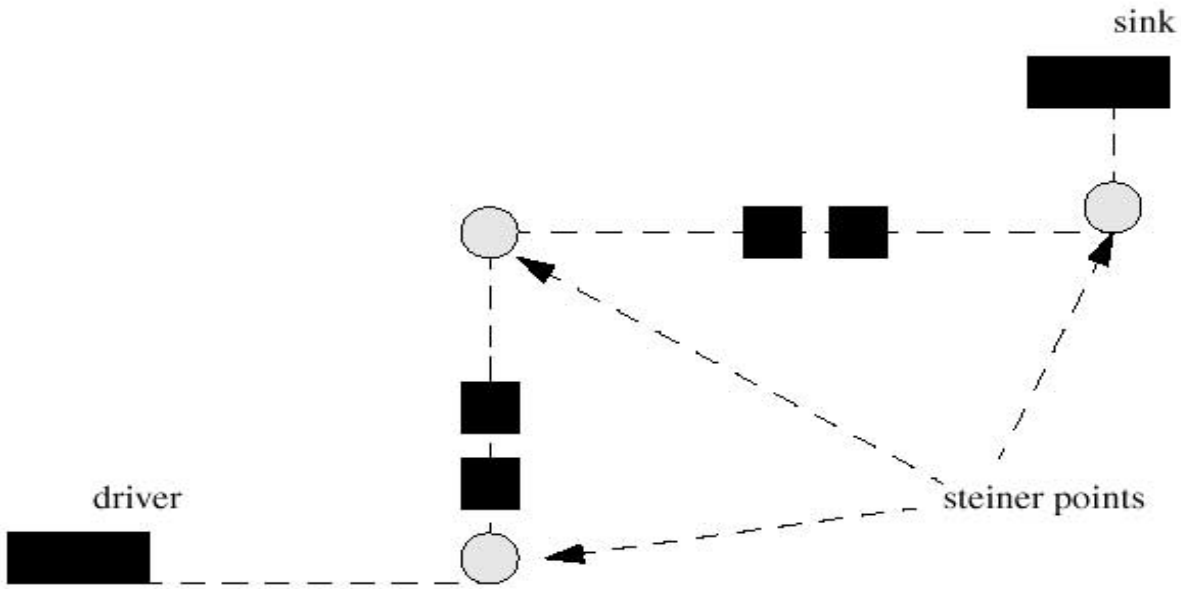


If the best repeater chain length is supposed to be 2, then the repeaters are inserted along the Steiner path, such that the repeaters are evenly placed along the path. The distance between the repeaters should be equal.

If the repeater is a buffer (nbRepeaters_ =2), then the insertion would be done as shown below.



If the repeater is an inverter (nbRepeaters_ =2), then the insertion would be done as shown below.



Related Information

`do_xform_optimize_clock_gate`

get_clock_tree_constraints

`get_clock_tree_constraints [-pin` *`list_of_pins`*`][-physical_tree]`

Displays the constraints on all clock trees or on a specific clock tree.

Options and Arguments

`-pin` *`list_of_pins`*

Specifies the list of pin of instance or input ports for which to return the constraints.

`-physical_tree`

Specifies that the constraints to be reported are for the `do_build_physical_tree` command for non-clock signals.

Example

The following command retrieves the constraints for the clock tree starting at the port `clk`:

```
get_clock_tree_constraints -pin [find -ports clk]
```

Output:

```
Info: Clock clk: min_delay 0.000000, max_delay 3.000000, max_skew 0.
100000, max_leaf_transition 0.030000 <CTPKS-111>
```

Related Information

`do_build_clock_tree`

`do_build_physical_tree`

`get_clock_tree_constraints`

`reset_clock_tree_constraints`

get_clock_tree_objects

```
get_clock_tree_objects [{-leaf_only | -tree_only}] [-pin list_of_pins]
     [-no_gated] [-buffers] [-inverters] [-physical_tree]
```

Returns a list of objects in the form specified by the selected argument.

Options and Arguments

`-buffers | -inverters`
: Returns a Tcl list of IDs from the library cells usable for the clock tree. Selecting `buffers` returns the buffer IDs. Selecting `inverters` returns the inverter IDs.

`{-leaf_only | -tree_only}`
: Returns a Tcl list of object IDs that are part of the clock tree. Selecting `leaf_only` returns only leaf instances; `tree_only` returns only tree elements (pins, instances, and nets).

`-no_gated`
: Returns a Tcl list of clock tree objects seen from the root pin up to the very first cells that are neither buffers nor inverters. Inputs of combinational cells are reported as leaves.

`-physical_tree`
: Use when physical constraints (`set_clock_tree_constraints -physical ...`) but no clock phase were applied at root pin.

`-pin` *`list_of_pins`*
: Returns clock tree objects from the root pins of clock trees specified by *`list_of_pins`*. You can invoke this command on any module in the hierarchy. If given by name, the root pin should be the complete path name as seen from the `top_timing_module`. If given by ID, the ID should be returned by a find command executed from the `top_timing_module` (see example below). If no pin is given, the pins on the clock tree constraints that have been set will be used.

Examples

- The following commands store the IDs of the clock tree objects (nets, pins, instances) as `my_list`:

```
set clock_root [find -hier -pin u1/u2/i_234/Y]
set my_list [get_clock_tree_objects -pin $clock_root]
```

- The following command displays the list of cells available for clock tree building:

```
issue_message "list of cells available for clock tree:
[get_names [get_clock_tree_objects -buffer -inverter]]"
```

report_clock_tree

```
report_clock_tree [-format {list_of_columns}] [{-depth_first_search
     | -breadth_first_search}] [{-increasing_latency | -decreasing_latency}]
     [{-leaf_only | -tree_only}] [-no_gated {rising | falling}] [-physical_tree]
     [-tcl_list] [-pin root_pin] [-min_max_only] [-pvt {min | typ | max}]
     [-effective_skew] [-summary] [-to clock_tree_instances_pins_list]
     [-through clock_tree_instances_list] [-print_effective_skew_leaves]
     [{> | >> file_name}]

     Valid columns: pin_name (hierarchical pin name), cellref (cell type), edge
     (^ (rising) or v (falling)), latency (delay from root pin on the considered
     edge), violation (violation codes), slew (slope of the signal on the
     considered edge), predecessor_pin (name of the preceding hierarchical pin),
     level (number of cells in the tree from the root), increment (delay from
     predecessor pin), x_y (X and Y location of the pin), fanout (number of
     successors and leaf pins (in parentheses) in tree), load (total capacitance
     on the net), info (informational codes), error (error code), power (power
     consumption)

     Violation codes: C (a maximum capacitance violation on the net connected to
     the pin), F(a fanout violation on the pin), L (the delay on the pin violates
     the constraint), S (a max sink violation on the net connected to the pin), T
     (a max transition violation on the pin), W (a wire self heat violation on the
     net connected to the pin)

     Info codes: A (associated pin (for clock gating networks) specified with the
     ct_lps_associated_gating_component attribute), e (pin or port defined as
     excluded with a ct_excluded attribute), G (pin belonging to a gating
     component), H (hierarchical physical pin), I (pin belonging to an input
     padcell), l (pin or port defined as a leaf with a ct_leaf attribute),
     L (clock pin of register recognized as a leaf), m (main pin (for clock gating
     networks) specified with the ct_lps_main_gating_component attribute),
     M (instance which is set dont_modify), O( pin belonging to an output padcell),
     p (instance with the ct_preserve attribute set), P (instance with the
     ct_preserve_tree attribute set to true)

     Error codes: C (combinational cell with disabled path), D (data input of
     register not defined as leaf or excluded), F (timing not available), N (point
     closest to the root not driving any leaf), O (output port not defined as leaf
     or excluded), R (reconvergent clock (clock phase reconvergence)) T (point in
     tree twice (connectivity reconvergence from clock root))
```

Generates a report that displays (in a text table) the clock tree structure from its root component to all its leaf cells. If both edges of the clock are used, the report appears in two parts; one for the rising edge and the other for the falling edge. Excluded pins always appear in both parts. Also, when a cell has insertion delay, the displayed latency value includes the insertion delay. The value appears with an asterisk at the end (for instance, 4.17*). When the insertion delay has a min and max value, the two latency values are displayed separated by a slash (for instance, 3.54/4.17*).

Important

The insertion delay calculated and used by all CTPKS commands is equal to the difference between the arrival time at the root and at the arrival time at the leaf. If the root is a port, and that port has a non-zero arrival time (that is, the port has a driver cell constraint), the insertion delay calculated by CTPKS will differ from that calculated by `report_timing` by the value of the arrival time at the root. See `report_timing` for more information on how that command handles insertion delay.

Options and Arguments

`{> | >> file_name}`
Specifies file redirection.

`-breadth_first_search`
Generates a report that displays the pins by level order (from the root). Pins (leaf or not) of the same level appear together.

`-depth_first_search`
Generates a report that displays the pins in the order in which they are found by going as far in the tree as possible. Leaf pins connected to the same branch appear together.

`-effective_skew`
Computes the effective skew value corresponding to the max latency difference at clock pins between any two related registers and displays it in the header summary.

`-format {list_of_columns}`
Specifies the report format (must type curly braces):

`pin_name`: Hierarchical pin name

`cellref`: Cell type

`edge`: ^ (rising) or v (falling)

`latency`: Delay from root pin on the considered edge

`violation`: violation code:
`C`: Maximum capacitance violation on net connected to the pin

F: Fanout violation on the pin
L: Latency (delay) on the pin violates the constraint
S: Max sink violation on the net connected to the pin
T: Max transition violation on the pin
W: Wire self heat violation on the net connected to the pin

slew: Slope of signal on the considered edge

predecessor_pin: Name of the preceding hierarchical pin

level: Number of cells in the tree from the root

increment: Delay from predecessor pin

x_y: X and Y location of the pin

fanout: Number of successors and leaf pins (in parentheses) in tree, for instance, 10(6L)

load: Total capacitance on the net

info: Informational code:
A: Associated pin (for clock gating networks) specified with the `ct_lps_associated_gating_component` attribute
e: Pin or port defined as excluded with a `ct_excluded` attribute
G: Pin belonging to a gating component
H: Hierarchical physical pin
I: Pin belonging to an input padcell
l: Pin or port defined as a leaf with a `ct_leaf` attribute
L: Clock pin of register recognized as a leaf.
m: Main pin (for clock gating networks) specified with the `ct_lps_main_gating_component` attribute
M: Instance which is set `dont_modify`
O: Pin belonging to an output padcell
p: Instance with the `ct_preserve` attribute set
P: Instance with the `ct_preserve_tree` attribute set to `true`

error: Displays a character that indicates the error that prevents building a clock tree:
C: Combinational cell with disabled path
D: Data input of register not defined as leaf or excluded
F: Timing not available
N: Point closest to the root not driving any leaf

0: Output port not defined as leaf or excluded
R: Reconvergent clock (clock phase reconvergence)
T: Point in tree twice (connectivity reconvergence from clock root)

power: Displays the power consumption

`{-increasing_latency | -decreasing_latency}`
Specifies the order in which the pins should appear when `depth_first_search` or `breadth_first_search` do not fully determine it. The input and output of a cell always appear on two consecutive lines, only output latency is considered for ordering.

`-leaf_only`
Generates a report that will not display intermediate cells (buffers, inverters, and gating cells).

`-min_max_only`
Generates a report that will only report the cells involved in the smallest and longest latency.

`-no_gated {rising | falling}`
Generates a report that will consider each input (of non buffer and non inverter cell) as a leaf pin. If no edge is specified, each pin of the tree is shown in both the rising edge and falling edge section unless the active edge can be derived for the timing model. If an edge is specified, it is used as an active edge for the pins whose active edge can not be derived from the timing model. If clock tree constraints exist on the root pin, the report will display constraints and flag violations in the violation column.

`-physical_tree`
Generates a report that will consider each input (of non buffer and non inverter cell) as a leaf pin. Each pin of the tree is shown in both the rising edge and falling edge section. This switch is used in conjunction with the `do_build_physical_tree` command or in order to detect a gated clock. If physical tree constraints exist on the root pin, the report will display constraints and flag violations in the violation column.

`-pin` *`root_pin`*
Specifies the root pin of the clock tree to display.

`-print_effective_skew_leaves`
Displays tables showing effective skew leaves. This option is only valid with the `-effective_skew` option.

`-pvt {min | typ | max}`
Specifies in which pvt the timing values displayed by the `report_clock_tree` command will be calculated.

`-summary`
Displays only the report header.

`-tcl_list`
Generates a report (Tcl list) that can be used by another Tcl script.

`-to` *`clock_tree_instances_pin_list`*
Reports clock tree paths between the clock tree root and the specified list of clock tree instance pins.

`-through` *`clock_tree_instances_list`*
Reports all clock tree paths going through each clock tree instance in the specified list.

`-tree_only`
Generates a report that will not display leaf cells (flip-flops). The fanout column should be used to see the number of leaf cells.

Examples

- The following example shows a report generated by the `report_clock_tree` command:

  Output:

```
+-----------------------------------------+
| Report              | report_clock_tree |
|---------------------+-------------------|
| Options             | -pin clk          |
+---------------------+-------------------+
| Date                | 20020103.144136   |
| Tool                | pks_shell         |
| Release             | v5.0-a080         |
| Version             | Dec 14 2001 14:29:26 |
+---------------------+-------------------+
| Module              | tc1               |
| Timing              | LATE              |
| Slew Propagation    | WORST             |
| Operating Condition | slow              |
```

PVT Mode	max
Tree Type	balanced
Process	1.00
Voltage	2.25
Temperature	125.00
time unit	1.00 ns
capacitance unit	1.00 pF
resistance unit	1.00 kOhm

Clock tree root clk	Clock Tree Constraints	Actual	Area
Max. transition time at tree pins	0.300	0.131	
Min. insertion delay to leaf pins	0.100	0.145	
Max. insertion delay to leaf pins	1.000	0.212	
Max. skew between leaf pins	0.300	0.067	
Number of Violations		0	
Number of Buffers		0	0.000
Number of Inverters		3	134.400
Number of gated/pad/preserved Instances		0	0.000
Total Number of Instances		3	134.400
Number of Excluded Pins		0	
Number of Leaf Pins		4	

pin clk	rising edge

pin_name	cellref	edge	latency	slew	violation	level	fanout
clk		^	0.00	0.00		0	1
i_43/A	INVX8	^	0.00	0.00		1	
i_43/Y	INVX8	v	0.06	0.07		1	1
i_44/A	INVX8	v	0.06	0.07		2	
i_44/Y	INVX8	^	0.14	0.13		2	3(2L)
i_448/A	INVX8	^	0.15	0.13		3	
i_448/Y	INVX8	v	0.21	0.07		3	2(2L)
dffn_bi/CKN	DFFNX1	v	0.21	0.07		4	
dff_bb/CK	DFFX1	^	0.15	0.13		3	

pin clk	falling edge

pin_name	cellref	edge	latency	slew	violation	level	fanout
clk		v	0.00	0.00		0	1
i_43/A	INVX8	v	0.00	0.00		1	
i_43/Y	INVX8	^	0.06	0.09		1	1
i_44/A	INVX8	^	0.06	0.09		2	
i_44/Y	INVX8	v	0.15	0.12		2	3(2L)
i_448/A	INVX8	v	0.15	0.12		3	
i_448/Y	INVX8	^	0.21	0.08		3	2(2L)
dff_bi/CK	DFFX1	^	0.21	0.08		4	
dffn_bb/CKN	DFFNX1	v	0.15	0.12		3	

- The following example shows a report that includes the effective skew leaves:

```
> report_clock_tree -summary -effective_skew -print_effective_skew_leaves -pin
FXLOGICI_KClk
```

Clock tree root FXLOGICI_KClk	Clock Tree Constraints	Actual	Area
Min. insertion delay to leaf pins	1.000	2.470	
Max. insertion delay to leaf pins	3.000	15.565	
Max. skew between leaf pins	0.100	13.094	
Effective skew between leaf pins		12.060	
Number of Violations		6319	
Number of Buffers		1	3.000
Number of Inverters		0	0.000
Number of gated/pad/preserved Instances		0	0.000
Total Number of Instances		1	3.000
Number of Excluded Pins		0	
Number of Leaf Pins		6323	

Effective skew leaves for pin FXLOGICI_KClk	
Min. effective skew leaf pin	TXCOMPI/Factor1_regx22x/CP
Max. effective skew leaf pin	FOGI/retime_fifoxInDataReg_regx86x/CP

report_clock_tree_violations

```
report_clock_tree_violations [-pin list_of_pins] [-no_gated {rising | falling}]
```

Generates a report that provides an exhaustive list of clock tree violations.

Options and Arguments

`-no_gated {rising | falling}`
: Generates a report that will consider each input (of non buffer and non inverter cell) as a leaf pin. If no edge is specified, each leaf pin is considered as both a rising edge and a falling edge pin unless the active edge can be derived for the timing model. If an edge is specified, it is used as an active edge for the pins whose active edge can not be derived from the timing model. This switch is used in conjunction with the `do_build_physical_tree` command.

`-pin` *`list_of_pins`*
: Specifies the pin of an instance or an input port as the root of the clock tree. Without `-pin`, it reports on all clock trees previously specified with a `set_clock_tree_constraints` command.

Example

The following command generates a report listing any violations for the clock tree starting at the pin `clk`:

```
report_clock_tree_violations -pin clk
```

```
+-----------------------------------------------------------+
|        Clock tree root clk        |  Clock Tree | Actual |
|                                   | Constraints |        |
|-----------------------------------+-------------+--------|
| Min. insertion delay to leaf pins |       0.100 |  0.144 |
| Max. insertion delay to leaf pins |       0.200 |  0.208 |
|       Max. skew between leaf pins |       0.050 |  0.064 |
+-----------------------------------------------------------+
+-----------------------------------------------------------+
|        Clock Tree Violations Summary         | Violations |
|----------------------------------------------+------------|
| No. of max. clock skew violations:           | 1          |
| No. of max. load capacitance violations:     | 0          |
| No. of max. rise transition time violations: | 3          |
| No. of max. fall transition time violations: | 0          |
| No. of min. rise required time violations:   | 0          |
| No. of min. fall required time violations:   | 0          |
| No. of max. rise required time violations:   | 1          |
| No. of max. fall required time violations:   | 1          |
+-----------------------------------------------------------+
```

```
+-------------------------------+
| Max. clock | Desired | Actual |
|    skew    |         |        |
| violations |         |        |
|------------+---------+--------|
| clk        | 0.050   | 0.064  |
+-------------------------------+
+-------------------------------+
|  Max. rise  | Desired | Actual |
|  transition |         |        |
|    time     |         |        |
|  violations |         |        |
|-------------+---------+--------|
| dff_bb/CK   | 0.100   | 0.116  |
| dffn_bb/CKN | 0.100   | 0.116  |
| i_332/A     | 0.100   | 0.116  |
+-------------------------------+
+-------------------------------+
|  Max. rise | Desired | Actual |
|  required  |         |        |
|    time    |         |        |
| violations |         |        |
|------------+---------+--------|
| dff_bi/CK  | 0.200   | 0.203  |
+-------------------------------+
+-------------------------------+
|  Max. fall  | Desired | Actual |
|  required   |         |        |
|    time     |         |        |
|  violations |         |        |
|-------------+---------+--------|
| dffn_bi/CKN | 0.200   | 0.208  |
+-------------------------------+
```

Related Information

do_build_clock_tree

set_clock_tree_constraints

reset_clock_tree_constraints

```
reset_clock_tree_constraints [-pin list_of_pins | -all] [-physical_tree]
```

Removes previously set clock tree constraints.

Options and Arguments

`-all`
: Specifies that all constraints are to be removed.

`-physical_tree`
: Specifies that the constraints to be removed are for the `do_build_physical_tree` command for non-clock signals.

`-pin` *`list_of_pins`*
: Specifies the pin of an instance or an input port as the root of the clock tree.

Example

In the following example, the first command removes constraints from the clock tree starting at the pin `clk`. The second command removes all constraints from all clock trees. The third command removes constraints for the `do_build_physical_tree` command for non-clock signals.

```
reset_clock_tree_constraints -pin [find -port clk]
reset_clock_tree_constraints -all
reset_clock_tree_constraints -pin [find -port reset] -physical_tree
```

Related Information

`get_clock_tree_constraints`

`set_clock_tree_constraints`

set_clock_tree_constraints

```
set_clock_tree_constraints [-pin list_of_pins] [-min_delay value]
     [-max_delay value] [-max_skew value] [-max_leaf_transition value]
     [-max_tree_transition value] [-self_heat] [-physical_tree]
     [-file CTS_CLK.cstr]
```

Specifies the clock tree constraints for a specific source or multiple sources if more than one pin is specified. Clock tree constraints are saved in `.adb` files.

The actual max slew for the leaf pins of the clock tree will be the smallest of the `max_leaf_transition`, the `max_tree_transition`, the global variable `slew_time_limit`, or the limit specified in the library.

The actual max slew for the non-leaf pins of the clock tree will be the smallest of the `max_tree_transition`, the global variable `slew_time_limit`, or the limit specified in the library.

Options and Arguments

`-file CTS_CLK.cstr`

Specifies that the following data be read into CTPKS from the CTS constraint file (All others tokens will be silently ignored):

```
MacroModel port | pin maxRise minRise maxFall
  minFall cap
MaxDelay delay
MinDelay delay
MaxSkew delay
SinkMaxTran delay
BufMaxTran delay
Buffer "clock tree cell list"
NoGating NO|falling|rising
LeafPort rising|falling
LeafPin rising|falling
ExcludedPort
ExcludedPin
ThroughPin
PreservePin
End
```

Notes:

- The buffer list will not saved into the adb, and will be

overwritten if you use the `-use_buffers` and `-use_inverters` flags with a build command.

- The `noGating` attribute will not be saved into the adb, and will be overwritten if you use the `-no_gated` option.
- Only `AutoCTSRootPin` is supported by this command. `ClockNetName` and `AutoClockNetName` are not supported.

`-max_delay` *`value`*

Specifies the maximum acceptable clock leaf pin insertion delay.

`-max_leaf_transition` *`value`*

Specifies the maximum acceptable transition time at any leaf pin of the clock tree.

`-max_skew` *`value`*

Specifies the maximum acceptable difference between any two clock leaf pin insertion delay (max_skew is smaller or equal to max_delay and min_delay).

`-max_tree_transition` *`value`*

Specifies the maximum acceptable transition time at any pin of the tree, even leaf pins.

`-min_delay` *`value`*

Specifies the minimum acceptable clock leaf pin insertion delay.

`-physical_tree`

Specifies that the constraints are for the `do_build_physical_tree` command for non-clock signals.

`-pin` *`list_of_pins`*

Specifies the pin of an instance or an input port as the root of the clock tree.

`-self_heat`

Calculates wire self heat effects on the nets in the clock tree. While building the clock tree, checks are made for wire self heat violations, and the clock tree that minimizes the number of violations or eliminates the violations is selected.

Example

In the following example, the first command sets various constraints for the clock tree starting at the pin `clk`. The second command sets various constraints for the clock tree starting at the pin `clock`, and specifies that the wire self heat effect is to be taken into account when building the clock tree.

```
set_clock_tree_constraints -min_delay 0.1 -max_delay 1.0 -max_skew 0.3
-max_tree_transition 0.3 -pin [find -port clk]


set_clock_tree_constraints -self_heat -pin clock -min_delay 0.5 -max_delay 1.0
-max_skew 0.4
```

Related Information

`get_clock_tree_constraints`

define_structure

Note: The `define_structure` command has been added to this chapter only for reference. This command is included with the Cadence Clock Tree Generator tool and provides significant information that supports several CTPKS commands mentioned in this chapter.

```
define_structure
     {from_pin ‘component_name’ ‘output_pin_name’ |
       from_iopin input_iopin_name'}
     [drive_booster 'cell_name' ['cell_name']]
     [noninverting_tree ['cell_name' parent_fanout ...]
          [to_rising_clock_pins 'cell_name' parent_fanout ...]
          [to_falling_clock_pins 'cell_name' parent_fanout ...]
          [to_pin 'component_name' 'pin_name' ['cell_name'] ...]
     [inverting_tree 'cell_name' ... parent_fanout ...]
          [to_rising_clock_pins 'cell_name' parent_fanout ...]
          [to_falling_clock_pins 'cell_name' parent_fanout ...]
          [to_pin 'component_name' 'pin_name' ['cell_name'] ...]
```

Defines the clock tree structure. The following are some usage notes to keep in mind:

- If you use this command, you must define the structure for every subtree of the overall clock tree topology.
- The `define_structure` command is optional for each clock tree command set (`specify_tree`/`set_constraints` command pair) you use in the constraints file.
- You cannot use both the `define_cells` command and the `define_structure` command for the same clock tree.
- You specify the subtrees with the `from_pin` and `from_iopin` options, and you can specify the subtrees in any order.
- Each subtree specification must start with either the `from_pin` or the `from_iopin` option. These options specify the given subtree root.
- The polarity of the `drive_booster` must always be noninverting. You can specify one buffer cell, two buffer cells, or two inverter cells. Specifying one inverter cell, or one buffer cell and one inverter, will cause an error.
- If you specify an inappropriate subtree topology option (for example, if you use only the `inverting_tree` option for a particular subtree that has only noninverted paths to clock pins), you will see an error message.
- If you omit a necessary subtree topology option (for example, if the subtree has both noninverted and inverted paths to clock pins and you specified only the `noninverting_tree` option), you will see an error message.

Options and Arguments

cell_name	Specifies the cell name of a buffer or inverter.
component_name	Specifies the component name.
drive_booster	Specifies that the given subtree root should have two inverters or one or two buffers connected directly to it to boost its drive.
input_iopin_name	Specifies the name of an input I/O pin.
inverting_tree	Specifies the inverted subtree topology from the given root to the clock pin subtree, gating pins, and block pins.
noninverting_tree	Specifies the non-inverted subtree topology from the given root to the clock pin subtree, gating pins, and block pins.
output_pin_name	Specifies the name of an output component pin.
parent_fanout	Specifies the number of fanouts associated with the parent of the previously specified buffer or inverter.
to_falling_clock_pins	Specifies the buffer subtree topology to falling-edge triggered clock pins (the subtree does not include gating pins or block pins).
to_pin	Specifies the padding in front of a gating pin or block pin. **Note**: The to_pin option can only be used to specify padding in front of input pins to gating components or input pins to blocks with the TLF insertion_delay construct for an internal clock.
to_rising_clock_pins	Specifies the buffer subtree topology to rising-edge triggered clock pins (the subtree does not include gating pins or block pins).

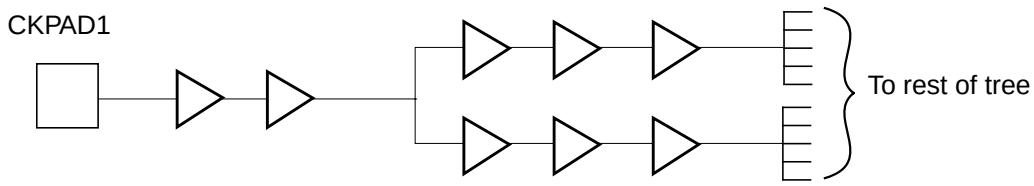
Example

In this example, the non-inverting buffer tree connected to pin CKPAD1 Z drives the CKBUF which is at the root of the to_rising_clock_pins tree as well as the three gating pins (CKNAND1 A, CKNAND2 A, and CKNAND3 A) and the CKBUF padding in front of the block pin

(IPBLOCK CK). The `to_pin 'IPBLOCK' 'CK' 'CKBUF' 'CKBUF'` statement specifies that two CKBUFs should be in front of the IPBLOCK CK pin to equal out delays. (The IPBLOCK has no `from_pin` statement because the tree does not go through this block.)

```
define_structure
     from_pin 'CKPAD1' 'Z'
        noninverting_tree 'CKBUF' 'CKBUF' 1 'CKBUF' 'CKBUF' 'CKBUF' 2
               to_rising_clock_pins 'CKBUF' 1 'CKBUF' 4 'CKBUF' 6
               to_pin 'CKNAND1' 'A'
               to_pin 'CKNAND2' 'A'
               to_pin 'CKNAND3' 'A'
               to_pin 'IPBLOCK' 'CK' 'CKBUF' 'CKBUF'
          from_pin 'CKNAND1' 'Z'
               inverting_tree 'CKINV' 1
                    to_rising_clock_pins 'CKBUF' 4
          from_pin 'CKNAND2' 'Z'
               inverting_tree 'CKINV' 1
                    to_rising_clock_pins 'CKBUF' 4
          from_pin 'CKNAND3' 'Z'
               inverting_tree 'CKINV' 1
                    to_rising_clock_pins 'CKBUF' 4
```

The schematic diagram for the noninverting tree connected to CKPAD Z is as shown below. Note that the tree contains repeaters.



The first, third, and fourth levels of buffers are considered repeaters and the clock tree generator distributes them evenly along the estimated routing path between the nonrepeating cells.

Attributes

```
set_attribute [cell_id ct_dont_utilize {true | false}]
     [pin_id ct_no_leaf {true | false} | "{true | false} pin_name"]
     [object_id ct_preserve {true | false}]
     [object_id ct_preserve_tree {true | false}]
     [pin_or_port_id ct_leaf {rising | falling} | "{rising | falling} {value |
      min_value max_value}"]
     [library_cell_pin_id ct_leaf {rising | falling}]
     [pin_id ct_excluded {true | false}]
     [instance_id ct_specify_padding {before | after}]
     [instance_id ct_lps_associated_gating_component name]
     [instance_id ct_lps_main_gating_component name]
```

Sets the value set for the specified attribute on the specified object. You can set tool- and user-defined attributes. See set_attribute on page 257 for a more detailed description of the this command.

Options and Arguments

cell_id ct_dont_utilize {true | false}
: Allows you to declare which cells you do not want to use in CTPKS. Selecting `true` tells CTPKS to ignore the cells specified by *cell_id*. Selecting `false` tells CTPKS to use the cells specified by *cell_id*.

instance_id ct_lps_associated_gating_component *name*
: See `ct_lps_main_gating_component` below.

instance_id ct_lps_main_gating_component *name*
: Defines a set of cells as belonging to same gating group. The name provided should be the same for all the instances of a group.

  The instances directly on the clock path should have the `ct_lps_main_gating_component` while the instances that are not on the clock path should have the `ct_lps_associated_gating_component` attribute. No buffering will be added between the “main” gating element and an “associated” element that is attached to the clock net. This prevents a sequential element used as a filter for the enabling of the gating component to be considered a leaf.

  All elements of a gating group will be moved together if the gating element is moved when the option -`move_gated` is used for `do_build_clock_tree`.

instance_id ct_specify_padding {before | after}
: Determines the behavior of the padding on this specific instance, when set on an instance, whether or not the switch -pad_after_gated is specified with the do_build_clock_tree command.

library_cell_pin_id ct_leaf {rising | falling}
: Treats all instances of the specified cell pin as a leaf pin when this pin would have otherwise been internal to the tree. CTPKS will not trace through this pin. Using a value other than rising or falling is treated as if no attribute was set.

object_id ct_preserve {true | false}
: Preserves an instance in the final tree. This option is only useful on buffers and inverters. Selecting true preserves an instance (specified by *object_id*) in the final tree. The instance is then treated as a gating component. A subtree is built from the output of the instance. Selecting false removes an instance (specified by *object_id*) from the final tree.

object_id ct_preserve_tree {true | false}
: Preserves a portion of the tree starting from an instance in the final tree. Selecting true preserves the portion of the tree starting from the instance (specified by *object_id*) in the final tree. Selecting false or any other value, treats the instance as normal when building the clock tree.

pin_id ct_excluded {true | false}
: Prevents CTPKS from considering some input pins as leaf pins in the generated buffer subtree. Excluded pins are connected directly to the output pin at the root of the generated buffer subtree. If the pin is an output pin, the pin has to belong to a gated component that has several outputs.

pin_id ct_no_leaf {{true | false} | "{true | false} *pin_name*"}
: When true, allows the tree to be considered through a pin identified as a clock pin that would otherwise be considered as a leaf. In the case of sequential gating component, any combinational arc will be followed. In the case of a purely sequential element such as in a clock divider, the sequential arc will be followed provided the output pin name is specified.

pin_or_port_id ct_leaf {{rising | falling} | "{rising | falling} {*value* | *min_value max_value*}"}
Treats the specified instance pin (input pin or output port) as leaf pins when these pins would have otherwise been internal to the tree. If a single value is associated to the edge, it is used to model the insertion delay of the downstream clock tree from this pin. If a pair of values are passed, the first value models the minimum insertion delay while the second value models the maximum insertion delay of the downstream logic.

Example

The following commands set attributes on various cellrefs, instances, and pins.

```
set_attribute [find -cellref ssnid6] ct_dont_utilize true
set_attribute [find -inst i_47] ct_preserve true
set_attribute [find -inst i_47] ct_preserve_tree true
set_attribute [find -pin u1/G] ct_no_leaf true
set_attribute [find -pin u2/CLK] ct_no_leaf "true Q"
set_attribute [find -pin gated_component/A0] ct_leaf rising
set_attribute [find -pin i_273/A] ct_leaf "rising 1.27 1.35"
set_attribute [find -pin decoder/Y3] ct_excluded true
```

Related Information

do_build_clock_tree

do_build_physical_tree

3

Distributed Synthesis Commands

This chapter describes the commands and global variables used with distributed synthesis.

check_batch

```
check_batch { queue | " "} { test_number | -all } [-debug] [-force] [-silent]
     [-timeout time] [-verbose] [-weight integer]
```

Checks a batch queue by running the `rac_shell` script in diagnostic mode.

For more information about the `rac_shell` script, see Appendix A, "Testing Distributed Synthesis," of the *Distributed Processing of BuildGates Synthesis* manual.

Options and Arguments

`-debug`	Displays detailed debug information.
`-force`	Forces `rac_shell` tests to run, even on hosts that are embargoed.
`queue` `|` `" "`	Checks the LSF batch *`queue`* or the current batch queue. (See also the `dist_batch_queue` global variable.)
`-silent`	Displays the minimum amount of information for each test.
`test_number` `|` `-all`	Runs the specified `rac_shell` test or all tests.
`-timeout` *`time`*	Specifies an optional timeout limit for each test, in *`days`*`-`*`hh`*`:`*`mm`*`:`*`ss`*`.`*`mmm`* format.
`-verbose`	Displays detailed log information printed for each test.
`-weight` *`integer`*	Uses the weight value to test the LSF Batch options that you have set with the `set_weight_batch_options` command. You must have already used `set_weight_batch_option` to set the weight to *`integer`* before using this option.

Example

```
rac_shell batch -batch_queue queue -test n
```

Related Information

`check_host`

`set_weight_batch_option`

check_dist

```
check_dist [-force] [-print] [-silent]
```

Verifies the settings of the distributed global variables and returns the number of errors detected. For example, if `dist_max_jobs` is set to 6 and `dist_min_jobs` is set to 8, an error is returned.

Options and Arguments

`-force`	Forces all of the tests to run, even on hosts that are embargoed.
`-print`	Displays more detailed log information for each test.
`-silent`	Displays the minimum amount of information for each test.

Related Information

`check_batch`

`check_host`

check_host

```
check_host { host | -all } { test_number | -all } [-debug] [-force] [-silent]
     [-timeout time] [-verbose]
```

Checks one or all of the hosts in a host list by running the `rac_shell` script in diagnostic mode.

For more information about the `rac_shell` script, see Appendix A, "Testing Distributed Synthesis," of the *Distributed Processing of BuildGates Synthesis* manual.

Options and Arguments

`-debug`	Displays detailed debug information.
`-force`	Forces `rac_shell` tests to run, even on hosts that are embargoed.
`host` `| -all`	Checks *`host`* or all of the hosts that are currently in the host list.
`-silent`	Displays the minimum amount of information for each test.
`test_number` `| -all`	Runs the specified `rac_shell` test, or all tests.
`-timeout` *`time`*	Sets a timeout period for each test, in *`days`*`-`*`hh`*`:`*`mm`*`:`*`ss`*`.`*`mmm`*.
`-verbose`	Displays detailed log information for each test.

Example

```
rac_shell catbert -test 2
```

Related Information

`set_host_list`

get_host_info

`get_host_info { host | -all [-force] } { attribute... }`

Returns information about the hosts in your host list or LSF batch queue. Allows specifying one or all hosts for which you want to return information. Also allows specifying one or all attributes for those hosts.

Options and Arguments

attribute	Requests the value of the specified *attribute* or, if no attribute is specified, all attributes of the specified hosts. You specify an attribute by its name. (See below for a list of attributes.)
`-force`	Requests the current value of attributes that can change during a job run. Otherwise, `get_host_info` returns the last known value of the attribute. The `-force` option also forces `get_host_info` to return information on embargoed hosts.
host | `-all`	Requests information for *host* or for all the hosts that are currently in the host list.

Attributes

`access`	Returns the access to the host. If `0` is returned, the host cannot be accessed. This attribute is dynamic. You must use the `-force` option to return the current value.
`cpus`	Returns the number of CPUs on the host.
`embargo`	Returns on if the host cannot run jobs; returns `off` if the host can run jobs. This attribute is dynamic. You must use the `-force` option to return the current value.
`embargo_secs`	Returns the temporary embargo time, in seconds.
`embargo_time`	Returns the temporary embargo time, in *hours*`:`*minutes*`:`*seconds*. If this value is nonzero, it represents the time period for which this host cannot run a remote job. (See also the `dist_embargo_delay` global variable.)

`jobs`	Returns the total number of jobs that ran on the host.
`load`	Returns the normalized load, between 0 and 100 percent. This attribute is dynamic. You must use the `-force` option to return the current value.
`max_load`	Returns the maximum load allowed for the host.
`memory`	Returns the memory size of the host, in MBytes.
`name`	Returns the given host name.
`named`	Returns the short name of the host.
`name_domain`	Returns the long host name, including the domain name.
`nice`	Returns the `nice` value for jobs on this host.
`release`	Returns the version number of distributed synthesis that you are running.
`rlogin_ok`	Returns the remote access permission to the host.
`ruser_ok`	Returns the remote access permission from the master to this host.
`speed`	Returns the clock speed of the CPU, in MHz.
`swap`	Returns the swap space, in MBytes.
`weight`	Returns the weight of the host (the relative size of the jobs that the host can handle). The size of the jobs that the host can handle increases as the weight increases, beginning with `1`. A weight of `0` indicates that the host can run any job. There is no upper limit on the weight of a host.

Examples

- The following command returns all attributes for a host:

```
get_host_info firefly
firefly
  {name firefly} {namedomain firefly} {cpus 1} {embargo off} {jobs 1} {kind
{SunOS sun4u}} {load 1} {load_time -28.737} {memory 512} {release v4.0} {rlogin
OK} {ruser OK} {speed 440} {swap 1025}
```

- The following command returns the value of a specific attribute:

  ```
  get_host_info firefly speed
  ```

- The following command returns the numbers of CPUs for all hosts (a line of information is printed for each host only if the `dist_verbose` global variable is nonzero):

  ```
  get_host_info -all cpus -force
  ```

  The `-force` option updates the attribute value:

- The following command finds the total number of accessible hosts:

  ```
  get_host_info -all access
  ```

- The following command returns the total number of CPUs that are currently running and accessible:

  ```
  get_host_info -all cpus
  ```

- The following command returns the recent load of all hosts:

  ```
  get_host_info -all load
  ```

- The following command determines the current, actual load of all hosts, including embargoed hosts:

  ```
  get_host_info -all load -force
  ```

- The following command determines the version of distributed synthesis that you are running:

  ```
  get_host_info firefly release
  v4.0
  ```

- The following command determines how many hosts are capable of running a job of a certain weight:

  ```
  get_host_info -all weight integer
  ```

Related Information

`set host config`

`set host list`

get_job_info

`get_job_info { job_id | -last_top | -all } attribute`

Returns the attribute values for a specific job, the top job, or all jobs.

Options and Arguments

attribute	Specifies the attribute whose value you want to return. (See below for a list of job attributes.)
job_id | -last_top | -all	Returns the attribute information for a specific job (*job_id*), the top job (-last_top), or all jobs (-all).

Attributes

batch	Returns 1 if the job is a batch job, otherwise 0.
batch_id	Returns the LSF job ID of a batch job.
batch_queue	Returns the name of the LSF batch queue to which the job was submitted.
cells	Returns the number of primitive and modifiable instances in the job module.
child_ids	Returns a list of job IDs for this job's child jobs.
children	Returns the number of child jobs.
cpu_secs	Returns the CPU time for the job, in seconds.
cpu_time	Returns the CPU time for the job, in *days-hours*:*minutes*:*seconds*.
depth	Returns the level of the job in the job hierarchy, calculated from the top job.
done_at	Returns the time at which the job completed, in *days-hours*:*minutes*:*seconds*.

`done_at_secs`	Returns the time at which the job completed, in seconds.
`exit_at`	Returns the time at which the job completed, in *`days-hours`*`:`*`minutes`*`:`*`seconds`*.
`exit_at_secs`	Returns the time at which the job completed, in seconds.
`exit_status`	Returns the job exit status. A value of `0` indicates that the job finished correctly. A non-zero value indicates that an error occurred.
`failures`	Returns the number of recoverable errors in the most recent attempt to run.
`family_ids`	Returns a list of all job IDs below a job or a top job.
`height`	Returns the number of remote job levels below and including this job.
`height_all`	Returns the number of all job levels below and including this job.
`host`	Returns the name of the host on which the job was launched. (See also the `rhost` job attribute.)
`instances`	Returns the number of modifiable instances in a job module.
`job_id`	Returns a unique job identifier for the job. This attribute value must be 1 or greater.
`job_secs`	Returns the entire job time, in seconds.
`job_time`	Returns the entire job time, in *`days-hours`*`:`*`minutes`*`:`*`seconds`*.
`kind`	Returns the job type: `structured`, `mapped`, `bottomup`, or `bottomupskip`.
`launched_at`	Returns the moment at which the job was launched, in *`days-hours`*`:`*`minutes`*`:`*`seconds`*.
`launched_at_secs`	Returns the moment at which the job was launched, in seconds.

`listed_at`	Returns the time at which the job was listed to run, in *`days-hours:minutes:seconds`*.
`listed_at_secs`	Returns the time at which the job was launched, in seconds.
`module`	Returns the ID of the module to be optimized by the job, or `0` if the ID is not available.
`most_jobs`	Returns the maximum number of jobs that actually ran in parallel at any time during the distributed run.
`name`	Returns the full name of the job module.
`name_path`	Returns the hierarchical path name of the job, specified by job name.
`name_truncated`	Returns the truncated name of the job, if it exceeds the name limit.
`new_at`	Returns the job creation time, in *`days-hours:minutes:seconds`*.
`new_at_secs`	Returns the job creation time, in seconds.
`nice`	Returns the `nice` value for the job. (See also the `dist_nice` global variable.)
`ordinal`	Returns the order (number) of the job in the run list.
`ordinal_ids`	Returns the list of dependent job IDs, in ascending ordinal order.
`parent_id`	Returns the job ID of the primary parent job.
`parent_ids`	Returns the list of all parent jobs, beginning with the primary parent.
`parents`	Returns the number of parent jobs.
`path`	Returns the location of the job in the hierarchy, specified by the *`job_id`*. For example, `1-12-13`.
`pid`	Returns the process ID of the job on the local host.

`queued_at`	Returns the moment the job was queued to batch queue, in *`days-hours:minutes:seconds`*.
`queued_at_secs`	Returns the moment when the job was queued to the batch queue, in seconds.
`remote_ids`	Returns a list of all remote job IDs below the top job.
`re_used`	Returns the module reuse count. A value of `0` indicates a unique module (no module reuse).
`rhost`	Returns the name of the remote host where the job ran.
`rpid`	Returns the process ID of the job on the remote host.
`run`	Returns the run number. The first retry is run number 2.
`run_secs`	Returns the run time, in seconds. (The run time is the elapsed time between the start and the exit time of the job.)
`run_time`	Returns the run time, in *`days-hours:minutes:seconds`*.
`shutoff`	Returns the shut-off timeout period, in seconds
`size`	Returns the size of the job.
`started_at`	Returns the launch time, in *`days-hours:minutes:seconds`*.
`started_at_secs`	Returns the launch time, in seconds.
`startup`	Returns the start-up timeout period, in seconds.
`startup_secs`	Returns the startup time, in *`seconds`*.
`startup_time`	Returns the startup time, in *`days-hours:minutes:seconds`*.
`status`	Returns one of the following job status values: `created`, `listed`, `waiting`, `ready`, `launched`, `queued`, `starting`, `running`, `exit`, `done`, `interrupted`, `killed`, `failure`, `error`, or `deleted`.

`timeout`	Returns the run-time timeout period, in seconds.
`top_id`	Returns the job ID of the top job in the distributed run.
`total_cpu_secs`	Returns the total CPU time of the job and its dependent jobs, in seconds.
`total_cpu_time`	Returns the total CPU time of the job and its dependent jobs, in *`days-hours:minutes:seconds`*.
`total_run_secs`	Returns the total run time of the job and its dependents, in seconds.
`total_run_time`	Returns the total run time of the job and its dependent jobs, in *`days-hours:minutes:seconds`*.
`total_size`	Returns the total job size of the job and its dependents.
`type`	Returns one of the following job types: `top`, `remote`, `master`, or `lumped`.
`wait_secs`	Returns the total wait time before a job is launched, in seconds.
`wait_time`	Returns the total wait time before a job is launched, in *`days-hours:minutes:seconds`*.
`weight`	Returns the weight of the job. When the weight is greater than `0`, the host must have a weight that is equal to or greater than the job weight. There is no upper limit on the weight that you can assign to a job. When the weight is 0, the job can run on any host.
`width`	Returns the maximum number of remote jobs that can run in parallel.
`width_all`	Returns the maximum number of all jobs that can run in parallel.

Examples

- The following command returns a list of all the jobs:

  ```
  get_job_info -all
  ```

- The following command returns a list of *`job_ids`* below the job specified, including the specified *`job_id`*:

```
get_job_info 2 family_ids
```

- The following command returns the value of a specific attribute:

```
get_job_info 2 cells
```

- The following command returns the *job_ids* of all top jobs:

```
get_job_info -all top_ids
```

- The following command returns the *job_ids* of all remote jobs:

```
get_job_info -all remote_ids
```

get_weight_batch_option

`get_weight_batch_option` *`integer`*

Gets the LSF batch options associated with the `weight` attribute.

Options and Arguments

`integer`	Specifies the weight of the host on the batch queue.

Related Information

`set_weight_batch_option`

kill_job

```
kill_job { job_id... | -last_top | -all } [-force] [-hierarchical] [-signal sig]
     [-silent] [-verbose]
```

Kills all processes when a failure is detected, and sends a signal to a specific job, to a job and all of its children, or to all jobs.
Default: SIGTERM, and the signal is sent only to jobs that are launched, started, or running.

Options and Arguments

`-force` — Sends a signal to all jobs for which the local or remote pid is available.

`-hierarchical` — Kills all child jobs of the specified job.

job_id | -last_top | -all
Kills the specified job (*job_id*), the most recent top job (-last_top), or all currently running jobs (-all).

`-signal` *sig* — Specifies the name or number of a UNIX/POSIX signal used to kill all local and remote job processes. Signals SIGHUP, SIGINT, SIGQUIT, or SIGTERM are recommended, but others can be used.

Note: SIGKILL and SIGSTOP are ignored.
Default: SIGTERM

`-silent` — Does not display the signals that the command sends to the remote processes.
Default: The value of the dist_kill_verbose global variable determines whether this is the default.

`-verbose` — Displays all signals that the command sends to kill the remote processes.
Default: The value of the dist_kill_verbose global variable determines whether this is the default.

Examples

```
kill_job -last_top -hierarchical -force
kill_job -all -force
```

Related Information

`remove job`

remove_host

```
remove_host { host | -all }
```

Permanently removes one, several, or all hosts from the host list. Removing a host from the host list removes all information about that host.

Options and Arguments

host | -all	Removes the specified host or all hosts from the host list.

Examples

```
remove_host host
remove_host -all
```

Related Information

set_host_list

remove_job

```
remove_job { job_id | -last_top | -all } [-hierarchical]
```

Permanently removes all information about the specified job. All job information exists only during the `ac_shell` run. It is lost upon exit.

Options and Arguments

`-hierarchical` Removes all child jobs of the specified job.

`job_id` | `-last_top` | `-all`
Removes the specified job (*`job_id`*), the most recent top job (`-last_top`), or all currently running jobs (`-all`).

Related Information

`kill_job`

report_job

```
report_job { job_id | -last_top | -all } [-hierarchical] [-parents]
```

Generates a report on one or several jobs.

Options and Arguments

`-hierarchical` Generates a report on the child jobs of the specified job.

`job_id` | `-last_top` | `-all`
Generates a report on *`job_id`*, the most recent top job, or all jobs.

`-parents` Generates a report on the parent jobs of the specified job.

Related Information

`get_job_info`

reset_dist_bits

`reset_dist_bits` *`list_of_module_name_or_ids`*

Sets the distributed synthesis `ac_shell` to 32 bits. Equivalent to `set_dist_bits 0`. *Default*: 32 bits.

Options and Arguments

`list_of_module_name_or_ids`
Resets the `-bits` value or module IDs to 32 bits.

Related Information

`set_dist_bits`

reset_dist_rlimit

`reset_dist_rlimit {`*`list_of_module_names_or_ids`*`}`

Resets the limit for distributed synthesis jobs.

Note: The value `reset_dist_rlimit...` is equivalent to `set_dist_rlimit {} ....`

Options and Arguments

{*list_of_module_names_or_ids*}
: Specifies the module names or IDs where you want to reset the limit for distributed synthesis jobs.

Related Information

`set_dist_rlimit`

reset_dist_point

```
reset_dist_point [-hier] {list_of_modules | -all}
```

Reverses the effect of a previous `set_dist_point` command (formerly the `reset_distribution_point` command). In other words, the modules specified are no longer explicitly set to be distribution points; the distributed synthesis tool determines whether they are distribution points.
Default: The size of a module determines whether it can become a distribution point.

Options and Arguments

`-hier` — Resets the entire hierarchy, starting from the modules specified in the *`list_of_modules`*.

`list_of_modules` `| -all` — Resets the specified module names or module IDs, or all modules in the hierarchy, starting from the current module.

Related Information

`set_dist_point`

reset_dist_weight

`reset_dist_weight` *`list_of_module_names_or_ids`*

Sets the `weight` attribute of the specified modules to `0` (formerly the `reset_weight` command). A module with a weight of 0 can run on any host.

Sets the weight of a module to indicate its size relative to other modules in your design. The higher the weight, the larger the module. When the distributed synthesis tool allocates jobs, it compares the weight of the job to the weight assigned to the host machines. A job is launched only on hosts whose weights are equal to or greater than the weight of the job.

Options and Arguments

`list_of_module_names_or_ids`
Resets the specified module names or module IDs.

Related Information

`set_dist_weight`

set_dist_bits

`set_dist_bits { non-negative_power_of_2 | list_of_module_names_or_ids }`

Specifies the `-bits` value for a distributed synthesis job on the module in the design to be either 32- or 64-bits. The `-bits` size of the job can never exceed the `-bits` size of the master `ac_shell`.

Options and Arguments

`non-negative_power_of_2 | list_of_module_names_or_ids`
Sets the `-bits` value or lists the module names or IDs.

Related Information

`reset_dist_bits`

set_dist_point

`set_dist_point { on | off | auto } [-hier] list_of_modules`

Controls whether a module, either in a list of modules or within the hierarchy of a listed modules, can become a distribution point. The list of modules can include module names or modules IDs (formerly the `set_distribution_point` command).
Default: The size of a module determines whether it can become a distribution point.

Important

Call `reset_distribution_point` for the modules that you want to set before you call `set_distribution_point`. This way you won't define conflicting distribution points for those modules.

Options and Arguments

`-hier`	Marks the entire hierarchy, starting from the modules specified in the *`list_of_modules`*.
`list_of_modules`	Specifies the module names or module IDs to set.
`on | off | auto`	Determines whether a module can be a distribution point, as follows: **`on`** Sets the module as a distribution point, no matter what its size. **`off`** Sets the module to not be a distribution point, no matter what its size. **`auto`** Sets the module as a distribution point, according to its size.

Examples

- The following command marks only the specified module as a distribution point:

```
reset_dist_point -hier [get_current_module]
set_dist_point on [get_current_module]
```

- The following command marks some modules and the entire hierarchies below those modules as distribution points. Allows the tool to mark the distribution points for the remaining parts of the hierarchy based on size.

  ```
  reset_dist_point -hier list_of_module_names_or_IDs
  set_dist_point on list_of_module_names_or_IDs
  ```

- The following command marks some modules as distribution points and sends out those modules as remote jobs. Allows the tool to mark distribution points based on size for the remaining parts of hierarchies. For example, if a child module is large enough to go out as separate job, it should do that, even if its parent (or some other ancestor) is marked as distribution point.

  ```
  reset_dist_point -hier list_of_module_names
  set_dist_point on list_of_module_names
  ```

Related Information

reset_dist_point

set_dist_rlimit

`set_dist_rlimit {-cycle` *`num_cycles`* `| time_`*`limit`*`} `*`list_of_module_names_or_ids`*

Sets a time limit for distributed synthesis jobs. When running distributed synthesis, the value of *`num_cycles`* or *`time_limit`* determines the maximum amount of time that can be spent processing the specified modules or IDs.

Options and Arguments

`-cycle` *`num_cycles`*
: Specifies the maximum number of cycles to be spent processing the specified modules or IDs.

`list_of_module_names_or_ids`
: Specifies the module names or IDs where you want to set the limit for distributed synthesis jobs.

`time_limit`
: Specifies the maximum amount of time (hh:mm:ss) to be spent processing the specified modules or IDs.

Examples

The `limit` must be specified as a single argument in the `set_dist_rlimit` command, for example:

```
set_dist_rlimit -cycle n ....
```

The following examples use the global `dist_rlimit` and the command `set_dist_rlimit` to limit jobs to a specific amount of CPU time:

```
 # limit all jobs to 2 hours CPU time
set_global  dist_rlimit  2:00:00
 # except  two large modules which require 10 hours
set_dist_rlimit 10:00:00 [module1 | module2]
 # go do it ....
do_optimize -distributed ...
....
```

Related Information

`reset_dist_rlimit`

set_host_config

`set_host_config { host... | -all } { attribute... | -auto [-force] }`

Configures the host machine by modifying the individual attributes of the host. The configuration of a host is defined by a list of attributes. One or all hosts can be configured automatically.

Use `get_host_info` to query the host status and check the values of the attributes for a host.

Options and Arguments

attribute	Configures the specified host *attribute*. (See below for a description of the host attributes.)
`-auto`	Configures the critical host attributes automatically: `cpus`, `embargo`, and `rlogin_ok`.
`-force`	Launches a remote `ac_shell` on the specified host, even if the host is embargoed or overloaded. Use the `-force` option with the `-auto` option. If the `-force` option is not used, only lightly loaded hosts and hosts which have not been embargoed are configured.
host | `-all`	Configures a specific host or all hosts currently in the host list. The host must be present in the current host list. (See also `set_host_list`.)

Attributes

`cpus` *integer*	Specifies the number of CPUs on the host. *Default*: `1`

Important

The `cpus` attribute must be set for correct distributed operation.

`embargo { on | off }`
Prevents remote jobs from running on the host when set to `on`.

`kind` *string*	Specifies the machine type. *Default*: The result from the csh command `uname -srpi`.

`max_load` *`percent`* — Specifies the maximum load allowed on the host. (See also the `dist_max_load` global variable.)

`memory` *`MBytes`* — Specifies the memory size of the host, in MBytes.

`name_domain` *`name`* — Specifies the full host name, including the domain name. For example, `ra.Cadence.com`.

`nice` *`integer`* — Specifies the `nice` value for remote jobs on the host. It must be a non-negative value. (See also the `dist_nice` global variable.)

`rlogin_ok { ok | no | unknown }` — Sets the permissions for remote access to the host. This attribute must be set for correct distributed operation.

`ruser_ok { ok | no | unknown }` — Sets the permission for access to the host from the master host. This attribute must be set for correct distributed operation.

`speed` *`MHz`* — Specifies the CPU speed, in MHz.

`swap` *`MBytes`* — Specifies the swap space, in MBytes

`user` *`name`* — Specifies your account name on this host.

`weight` *`integer`* — Specifies the weight of the host (the relative size of the jobs that the host can handle). The size of the jobs that the host can handle increases as the weight increases, beginning with `1`. A weight of `0` indicates that the host can run any job. There is no upper limit on the weight of a host.

Examples

- The following command sets the number of CPUs of a host:

```
set_host_config firefly cpus 2
```

- The following command removes an embargo on a host:

```
set_host_config firefly embargo off
```

- The following command removes the embargo on all hosts:

```
set_host_config -all embargo off
```

- The following command retrieves the critical configuration attributes (`cpus`, `rloginok` and `ruserok`) for the host:

```
set_host_config firefly -auto timeout
```

An ac_shell is launched on the specified host. The *timeout* limits the total run time of the automatic configuration; 0 means no timeout.
Default: 2:00 (2 minutes).

- The following command launches a remote ac_shell on all hosts to retrieve the critical configuration attributes:

  ```
  set_host_config -all -auto timeout
  ```

  timeout is an optional run-time timeout value for each remote ac_shell.
  Default: 2:00 (2 minutes).

- The following command sets the embargo switch for all hosts to on:

  ```
  set_host_config -all -embargo on
  ```

Related Information

get_host_info

set_host_list

`set_host_list` *`host`*`...`

Adds one or more hosts to the existing host list.

Options and Arguments

`host`	Specifies the name of the host to add to the host list.

Related Information

`check_host`

set_dist_weight

`set_dist_weight` *`integer list_of_module_names_or_ids`*

Sets the `weight` attribute of the specified modules. You can set the weight of a module to indicate its relative size. The higher the weight, the larger the module.

When the distributed synthesis tool allocates jobs, it compares the weight of the job to the weight of the host machines. A job is launched only on a host whose weights is equal to or greater than the weight of the job.

When you set the weight of a module to 0, it can run on any host.

Options and Arguments

`integer` Specifies the weight of the module.

`list_of_module_names_or_ids`
Specifies one or more module names or module IDs whose weight you want to set.

Related Information

`set_host_config`

`set_weight_batch_option`

`reset_dist_weight`

set_weight_batch_option

`set_weight_batch_option` *`integer any_LSF_bsub_option_string`*

Sets the LSF Batch options based on the `weight` attribute.

Options and Arguments

`any_LSF_bsub_option_string`
Specifies the LSF Batch options whose weight attribute you want to set.

`integer` Specifies the weight of the host on the batch queue.

Related Information

`get_weight_batch_option`

4

Low Power Synthesis (LPS) Commands

This chapter describes the Low Power Synthesis (LPS) commands for BuildGates® Synthesis and Cadence® PKS:

check_cg_logic

```
check_cg_logic [-instance list_of_dummyEnableHierIds]
     [-no-hier]
```

Checks the integrity of the clock-gating logic, that is, whether the gating instances are properly connected to the register banks of the same clock-gating domain. Returns an error if it detects some anomaly. You must read the timing and clock constraints, and inserted some clock-gating logic before you can use this command. Use this command when you start a new session with an updated netlist to check the integrity of your clock-gating logic.

Options and Arguments

`-instance` *`list_of_dummyEnableHierIds`*
: Specifies a list of dummy enable hierarchy instances that must be checked. Specify the instances using their identifiers (IDs). *Default*: Checks all dummy enable hierarchy instances starting from the current module.

`-no_hier`
: Limits checking of the dummy enable hierarchy instances to the current module.

Example

The following example shows where you use this command in the flow:

```
read_verilog design.v
do_build_generic
# set constraints (including clock constraints)

# Clock gating exploration
do_optimize -stop_for_power_simulation -power

write_verilog -hier my_design.v
write_clock_gating_attribute -file cgattr.tcl

# at this point you can run external (Cadence or third-party) tool
do_remove_design -all

# read the updated netlist
read_verilog my_design_updated.v

do_build_generic
# set constraints (including clock constraints)

# read attributes associated with clock-gating modules
source cgattr.tcl

# check current module
check_cg_logic -no_hier
```

Related Information

`get_gating_instance_list`

`write_clock_gating_attribute`

do_remove_cg_dummy_hierarchy

```
do_remove_cg_dummy_hierarchy
     [-remove_scan_enable_dummy]
```

Removes any clock-gating dummy module that was added during clock-gating insertion. All the dummy modules in the current module and in the hierarchy below the current module are removed. If a module is marked `dont_modify`, then all the dummy modules in that module or any modules in the hierarchy below it are kept. Optionally, also removes any scan enable dummy hierarchies.

You must insert clock-gating logic with LPS before you can use this command.

Note: You can only remove clock-gating dummy modules if they can be identified. Modules are identified by special attributes set on them.

You cannot decommit clock-gating logic or insert root-gating logic in modules where this command was issued. Because the dummy module is required to identify clock-gating logic, you must invoke this command after decommitting and reporting of clock-gating instances, and insertion of root-gating logic.

Options and Arguments

`-remove_scan_enable_dummy`
Removes any scan enable dummy hierarchies.

Example

The following example shows where you use this command in the flow:

```
read_tlf tech.tlf
read_verilog ckt.v
do_build_generic
do_optimize -power -stop_for_power_simulation
write_verilog -hier ckt.vc
<simulate>
read_tcf ckt.tcf
do_optimize -power # this command performs decommittment, root-gating insertion
get_gating_instance_list/ get_list_of_cg_instances
do_remove_cg_dummy_hierachy
write_verilog final.v
```

Related Information

`get_gating_instance_list`

do_xform_insert_sleep_mode

```
do_xform_insert_sleep_mode {-data_pin {list_of_pins}
     |-data_net {list_of_nets}} -enable {list_of_nets}
     [-function {and | or}]
```

Lets you manually insert LPS sleep-mode candidates. You must have read the netlist.

Options and Arguments

`-data_net {`*`list_of_net`*`}`
: Specifies the nets on which to insert the sleep mode logic. The specified nets must belong to the same module. Specify the nets using their identifiers (IDs) or names.

`-data_pin {`*`list_of_pins`*`}`
: Specifies the instance input pins, in front of which the sleep mode logic to be inserted. The specified pins must belong to the same instance. Specify the pins using their identifiers (IDs) or names.

  You must use this option if the nets on which the sleep mode logic must be inserted have more than one fanout.

`-enable {`*`list_of_nets`*`}`
: Specifies the nets which are the enable function for the sleep mode logic to be inserted. The nets can be in different hierarchical modules, in which cases extra ports may be created. Specify the nets using their identifiers (IDs) or names.

`-function {and | or}`
: Specifies the Boolean function for the enable function if the number of nets in option `-enable` is more than 1.
  Default: `and`

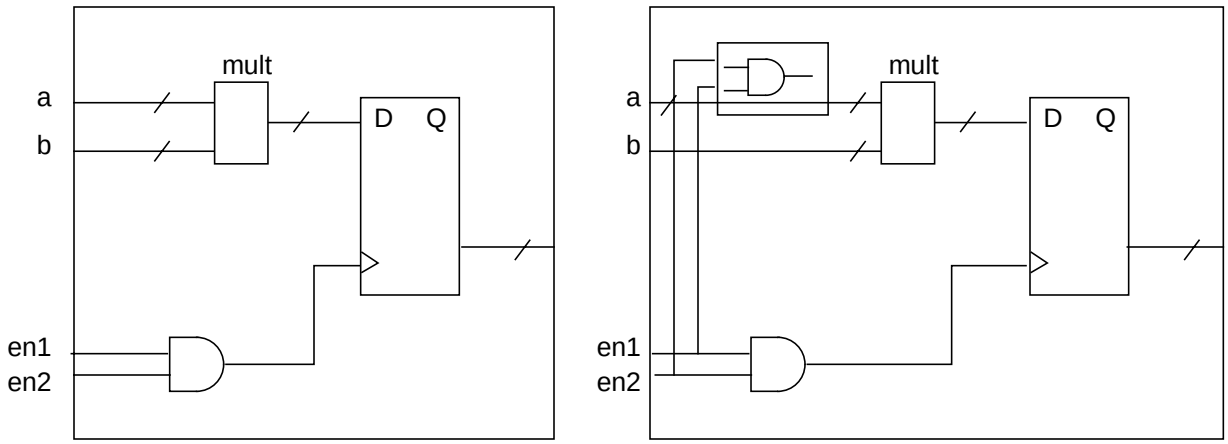
Example

The following command inserts a sleep mode module on pin a of instance `mult` and uses `en1` and `en2` as enable nets:

```
> do_xform_insert_sleep_mode -data_pin [find -pin mult/a*] \
     -enable [find -port -input en*]
```

Figure 4-1 shows the original schematic and the schematic with the sleep mode module inserted.

Figure 4-1 Schematic for do_xform_insert_sleep_mode Example



original schematic

schematic with sleep-mode module

Related Information

`do_build_generic -sleepmode`

`do_xform_optimize_power`

`set_sleep_mode_options`

do_xform_optimize_clock_gate

```
do_xform_optimize_clock_gate [-clone ] [-declone ]
     [-decommit] [-root_gate] [-rg_decommit]
     [-clock list_of_clocks]
```

Performs transformations (specifically de-cloning, cloning, adding root-gating logic) to reduce the clock tree insertion delay and clock skew, and decommits clock-gating on the clock network to minimize the power dissipation. You can run this command before or after `do_build_clock_tree`.

Note: You must place the design and source the clock constraints before you can use this command. To verify whether you set the clock constraints properly, you can run the `report_clock_tree` command with the `-sum` option. If the `report_clock_tree` command indicates a clock tree trace failure, you need to exclude these objects from clock tree consideration by using the `set_attribute ct_excluded true` command. For more details, refer to the *Cadence Physically Knowledgeable Synthesis (PKS) User Guide*. You must also read a toggle count format (TCF) file before you run this command, unless you set the global `power_opt_no_tcf` to `true`.

The circuit is modified such that:

- CTPKS has a higher probability of synthesizing a clock tree that will meet the clock constraints
- Final power dissipated by the post-clock tree inserted circuit is less than if there was no transformation

Note: The accessory logic in `clock-gating` (latch, control-gate) is not targeted for de-cloning or cloning (*or* is not affected by the `-clone` and `-declone` options).

Options and Arguments

`-clock list_of_clocks`
: Limits the transformations to the specified clocks.
Default: The transformation is performed on all clocks.

`-clone`
: Clones a gating cell such that the gated clock tree is balanced, localized, and consumes less power.

`-declone`
: Merges all logically equivalent gating cells to minimize the power dissipated in the clock network.

`-decommit`

Decommits clock-gating logic that does not significantly reduce power dissipation of the design or violate timing constraints. This optimization takes clock insertion delays into consideration while performing the timing checks.

`-rg_decommit`

Removes any root-gating logic specified after the clock tree is generated, if it does not save power or improve timing. The hierarchical module containing the logic inserted during root gating must be present for performing this optimization.

`-root_gate`

Reduces the overall clock tree power dissipation by inserting gating logic at the root or subroots of the clock tree.

Default Behavior

If neither of the `-clone, -declone, -decommit, -rg_decommit,` and `-root_gate` options are specified and if `clock_propagation_mode` is set to `ideal`, `-clone`, `-declone`, and `root_gate` are performed, but not necessarily in that order.

If neither of the `-clone, -declone, -decommit, -rg_decommit,` and `-root_gate` options are specified and if `clock_propagation_mode` is set to `propagated`, `-decommit` and `-rg_decommit` are performed.

If any of the `-clone, -declone, -decommit,-rg_decommit,` and `-root_gate` are options specified, then only that transformation will be applied.

De-cloning, cloning, and adding root-gating logic can only be performed if `clock_propagation_mode` is set to `ideal`.

Limitations

Currently, there is no automatic decommitment of the root-gating logic. To remove any inserted root-gating logic, use the `set_clock_gating_options -rg_ignore` command.

Cloning ignores *clock_gating_integrated_cells* with observability ports.

Example

The following command sets the maximum number of the enable signals per root gating logic to three:

```
> set_clock_gating_options -rg_max 3
> do_xform_optimize_clock_gate -root_gate
```

Related Information

do_optimize

do_xform_optimize_clock_tree

report_clock_tree

set_clock_gating_options

set_attribute

do_xform_optimize_power

```
do_xform_optimize_power [-effort {low | medium | high}]
     [-no_design_rule] [-ignore_timing] [-ipo]
     [-preserve_hold] [-critical_ratio {0.0-1.0}]
     [-critical_offset float] [-fix_clock_net]
     [-cell_replace] [-resize_by_footprint]
     [-sleep_mode] [-clock_gate] [-gate_level]
```

Analyzes and commits sleep-mode logic and clock-gating logic to the design when sleep-mode and clock gating is inserted during front-end optimization. This command also performs gate-level optimizations to improve average power consumption in the design, while honoring the power constraints set with `set_power_optimization_options`.

Note: You must insert switching activities into the design before you run this command, unless you set the global `power_opt_no_tcf` to `true`. To insert switching activities, use one of the following commands: `read_tcf`, `read_saif`, or `read_vcd -dynamic`.

Use this command instead of `do_optimize -power` if you have a timing-optimized netlist.

By default, this command applies any gate-level transformation for power optimization. The default behavior of the command for a non-placed and a placed design with different options is shown in Tables 4-1 and 4-2. On routed designs, however, only resizing is applied.

Table 4-1 Default Behavior for a Non-Placed Design (`bgx_shell` and `pks_shell`)

	`-clock_gate`	**`-sleep_mode`**	**`-gate_level`**
`do_xform_optimize_power`	yes	yes	yes
`do_xform_optimize_power -clock_gate`	yes	no	no
`do_xform_optimize_power -sleep_mode`	no	yes	no
`do_xform_optimize_power -gate_level`	no	no	yes

Note: If none of the three optimization options are specified and the design is not placed, then wire-load model-based clock gating, sleep mode and gate-level optimization is applied.

Table 4-2 Default Behavior for a Placed Design (`pks_shell`)

	`-clock_gate`	`-sleep_mode`	`-gate_level`
`do_xform_optimize_power`	no	no	yes
`do_xform_optimize_power -clock_gate`	yes	no	no
`do_xform_optimize_power -sleep_mode`	no	yes	no
`do_xform_optimize_power -gate_level`	no	no	yes

To reduce power using sleep-mode analysis, run these commands in the following order:

```
set_sleep_mode_options
do_build_generic -sleep_mode
```

To reduce power with clock-gating logic, run these commands in the following order:

```
set_clock_gating_options
do_optimize -power -stop_before_mapping
```

Options and Arguments

`-cell_replace`

Only replaces the current cells with functionally equivalent cells of the same size. This option is useful in the post-route multiple threshold voltage power optimization flow where low threshold voltage cells are replaced with high threshold voltage cells that have the same area and physical footprint.

`-clock_gate`

Controls the commitment of clock-gating logic.

If the design is not placed, clock-gating logic is decommitted based on the wire-load model estimation of the clock network.

If the design is placed, clock-gating logic is decommitted based on the CTPKS estimation of the clock network.

`-critical_offset` *`float`*

Specifies the offset for paths to be considered for the timing-driven power optimization. The floating number is added to the value of the worst slack.
Default: `0`

`-critical_ratio {0.0 - 1.0}`

Specifies the range of target slack for paths to be considered for timing-driven power optimization. The ratio is expressed as a percentage of the worst slack. This allows a range of worst slack which is considered critical by the command. Specify a positive number.
Default: `0`

The target slack used in timing-driven power optimization is computed as follows:

For negative worst slack

```
(worst_slack * (1 - critical_ratio) + critical_offset, worst_slack)
```

For positive worst slack

```
(worst_slack * (1 + critical_ratio) + critical_offset, worst_slack)
```

`-effort {low | medium | high}`

Specifies the scope of the gate-level power optimization that you want. Performs gate-level transformations, such as resizing, pin swapping, and restructuring, to minimize power consumption without worsening the current slack.

In general, a higher effort level implies more optimization iterations and longer run times than for a lower effort level. The `low` effort is recommended for post-timing optimized designs.
Default: `medium`

`-fix_clock_net`

Allows gate-level power optimization transformations on clock nets.
Default: The clock network is not modified by the power optimization transformations.

`-gate_level`

Performs gate-level power optimization.

If the design is not placed, wire-load model-based gate-level power optimization is applied.

If the design is placed, PKS integrated gate-level power optimization is applied.

`-ignore_timing`

Ignores all timing constraints while optimizing power

`-ipo`

Limits gate-level power optimization to resizing when optimizing for power.

`-no_design_rule`

Ignores design rule checking (such as slew rate) while optimizing power.

`-preserve_hold`

Prevents early slack from getting worse while optimizing power.

`-resize_by_footprint`

Resizes to cells with the same footprints.

Note: PKS optimization does not rely on footprints. but if foot print check is required, you must specify this option.

`-sleep_mode`

Controls the commitment of sleep modules.

If the design is not placed, sleep mode modules are committed based on power savings and delay penalty.

If the design is placed, committed sleep modules are decommitted if they violate timing.

Examples

- The following command allows to set a target slack of -0.47:

```
do_xform_optimize_power -critical ratio 1 -critical_offset -0.47
```

Related Information

`do build generic`

`do optimize`

`report_power`

Gate-level power optimization-only flow in the *Low Power for BuildGates Synthesis and Cadence PKS*

get_clock_gating_options

```
get_clock_gating_options [-tcl_list]
```

Displays the current clock-gating settings.

Options and Arguments

`-tcl_list`

Displays the settings of the clock gating options in Tcl list format. *Default*: The options and their values are shown in a tabular format.

Examples

- The following commands set all options to their default and show the settings in tabular format:

```
> set_clock_gating_options -default
> get_clock_gating_options
```

Returns the following output:

```
+---------------------------------+
|      Clock-Gating Options       |
|---------------------------------|
|     Options     |     Value     |
|-----------------+---------------|
| -auto_test_port |     false     |
|    -control     |     none      |
|  -control_mode  | use_test_mode |
|  -control_port  |    NOT_SET    |
|     -domain     |      all      |
|     -force      |    NOT SET    |
|  -gating_style  |     latch     |
|     -ignore     |    NOT SET    |
|   -max_fanout   |    NOT SET    |
|    -minsize     |       3       |
|   -no_timing    |     false     |
|    -observe     |     false     |
|   -obs_style    |     port      |
|   -remove_all   |     false     |
|  -rg_dissolve   |     false     |
|    -rg_force    |    NOT SET    |
|   -rg_ignore    |    NOT SET    |
|     -rg_max     |      10       |
|  -same_polarity |     false     |
|   -xor_depth    |       5       |
+---------------------------------+
```

- The following commands set some options and show the settings in tabular format:

```
> set_clock_gating_options -max_fanout 2 -control pre_seq_element \
> -observe -domain dft_domain -ignore {127987}
> get_clock_gating_options
```

Returns the following output:

```
+-----------------------------------+
|       Clock-Gating Options        |
|-----------------------------------|
|     Options     |      Value      |
|-----------------+-----------------|
| -auto_test_port |     false       |
|    -control     |     none        |
|    -control     | pre_seq_element |
|  -control_port  |    NOT_SET      |
|     -domain     |   dft_domain    |
|     -force      |    NOT SET      |
|  -gating_style  |     latch       |
|     -ignore     |    127987       |
|   -max_fanout   |       2         |
|    -minsize     |       3         |
|   -no_timing    |     false       |
|    -observe     |     true        |
|   -obs_style    |     port        |
|   -remove_all   |     false       |
|  -rg_dissolve   |     false       |
|    -rg_force    |    NOT SET      |
|   -rg_ignore    |    NOT SET      |
|     -rg_max     |      10         |
|  -same_polarity |     false       |
|   -xor_depth    |       5         |
+-----------------------------------+
```

- The following commands set some options and show the settings in Tcl list format:

```
> set_clock_gating_options -max_fanout 2 -control pre_seq_element \
> -observe -no_latch -force {121203} -ignore {127987}
> get_clock_gating_options -tcl_list
```

Returns the following output:

```
{-auto_test_port false}  {-control pre_seq_element}
{-control_mode use_test_mode}  {-control_port NOT_SET} {-domain dft_domain}
{ -force { } } {-gating_style latch} { -ignore {127987} } {-max_fanout 2}
{-minsize 3} {-no_timing false} {-observe true} {-obs_style port}
{-remove_all false} {-rg_dissolve false} { -rg_force { } } { -rg_ignore { } }
{-rg_max 10} {-same_polarity false}  {-xor_depth 5}
```

Related Information

set_clock_gating_options

get_clock_tree_power

```
get_clock_tree_power clock_pin [-tree_only]
```

Returns the total power consumed by all the instances on the clock tree that have the specified pin or port as clock root. Only one clock pin can be specified. Use the `get_power_display_unit` command to show the power units.

Note: You must insert a clock tree, before you can use this command.

Options and Arguments

`clock_pin`

Specifies the port or instance output pin that is the root of the clock tree. Any port in the clock tree can be specified.

`-tree_only`

Excludes the power dissipated by the clock sink objects (sequential elements).
Default: The power consumed by the sequential elements is included.

Example

The following command returns the total power consumed by all the instances on the clock tree that have pin `ctrl/clk` as clock root:

```
> get_clock_tree_power ctrl/clk
```

Returns an output similar to the following:

```
{81830 7.290e-07 }
```

The first number is the clock pin ID and the next number is the total power consumption.

Related Information

`get_power_display_unit`

`report_clock_tree`

get_dynamic_peak_power

```
get_dynamic_peak_power
```

Reports the peak dynamic power—the maximum of the dynamic power values calculated by `read_vcd` for the incremental timing windows.

Note: To report the dynamic peak power, you must have run `read_vcd -dynamic -time_period` *`time`*, and the specified time window must be larger than zero. If you change any slew or capacitance values for the design, you need to read in the VCD file again to recalculate the dynamic power.

Related Information

`get_power_display_unit`

`read_vcd`

get_dynamic_power

```
get_dynamic_power [-instance {list_of_instances}]
     [-net {list_of_nets}]
```

Traverses the hierarchy and returns the average cycle-by-cycle power consumed for the specified instances or nets of the current module in the period during which the events are monitored. This period is defined with the `read_vcd` command. Use the `get_power_display_unit` command to show the power units.

LPS computes the power for each transition in the VCD file that is relevant to the specified instances and nets.

Note: You must read a VCD file (using `read_vcd -dynamic`) before you can run the `get_dynamic_power` command. If you change any slew or capacitance values for the design, you need to read in the VCD file again to recalculate the dynamic power.

Options and Arguments

`-instance {list_of_instances}`
: Specifies the instances for which you want to obtain the power. Instances must be separated by a space.

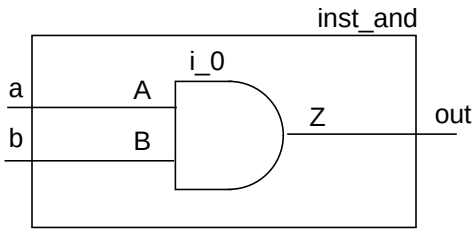
`-net {list_of_nets}`
: Specifies the nets for which you want to obtain the power. Nets must be separated by a space. In this case, the `get_dynamic_power` command reports for each net the power that is consumed by the capacitance of the net and by the capacitance of the pins driven by the specified net.

Note: If you do not use any of these options (default), the command returns the power for all the instances and all the nets in the current module.

Examples

For the following examples, consider the circuit shown in Figure 4-2.

Figure 4-2 Schematic for get_dynamic_power Example



- The following commands return the average cycle-by-cycle power for all the instances and all the nets in module `inst_and`:

```
> read_vcd -dynamic -module inst_and and.vcd
> get_dynamic_power
> 1.739e-3
```

- The following commands return the average cycle-by-cycle power for instance `i_0`:

```
> read_vcd -dynamic -module inst_and and.vcd
> get_dynamic_power -instance i_0
> 1.413e-3
```

- The following commands return the average cycle-by-cycle power for net a:

```
> read_vcd -dynamic -module inst_and and.vcd
> get_dynamic_power -net a
> 1.730e-4
```

- The following commands return the average cycle-by-cycle power for nets a and b:

```
> read_vcd -dynamic -module inst_and and.vcd
> get_dynamic_power -net {a b}
> {1.730e-4 1.533e-4}
```

Related Information

`get_power`

`get_power_display_unit`

`read_vcd`

get_gating_instance_list

```
get_gating_instance_list
     [-instance {list_of_instances}] [-hier]
```

Returns a list with the following elements:

- Top-level clock identifier
- Dummy enable hierarchy identifier associated with the clock-gating logic
- Enable net identifier
- Identifiers of all clock-gating instances in the clock-gating domain

In case of multiple clock-gating domains, the command returns a list for each domain.

Note: You must insert clock-gating logic with LPS before you can use this command.

Important

This command does not work after clock-gate commitment or when the dummy hierarchies are removed with the `do_remove_cg_dummy_modules` command.

Options and Arguments

`-hier`

Searches for clock-gating logic down the hierarchy, starting from the current module.

`-instance {`*`list_of_instances`*

Specifies the instances that must be searched for clock-gating logic. You must specify the instances using their identifiers (IDs). *Default*: The tool looks for clock-gating logic in the current module.

Examples

- The following command returns information for clock-gating logic in the hierarchy, starting from instance `78499`:

  ```
  > get_gating_instance_list -instance {78499} -hierarchy
  ```

  Returns an output similar to the following:

  ```
  {77877 73411 71621 76067}
  ```

where members of each quadruplet have the following meaning:

```
{clkId dummyEnableHierId enableNetId gatingInstanceId}
```

- The following command returns information for all clock-gating logic in the hierarchy, starting from the current module:

  ```
  > get_gating_instance_list -hierarchy
  ```

  Returns an output similar to the following:

  ```
  {77877 78979 77509 79523} {77877 73411 71621 76067}
  ```

Related Information

get_list_of_cg_instances

get_list_of_cg_instances

```
get_list_of_cg_instances
     -enable_id dummyEnableHierId [-include_ff]
```

Obtains all the logic instances for the specified enable identifier. The enable is associated with one of the clock-gating logic blocks.

Note: You must insert the clock-gating logic with LPS and run the `get_gating_instance_list` command to get the dummy enable hierarchy identifiers, before you can use this command.

Options and Arguments

`-enable_id`

Specifies the dummy hierarchy for which the associated clock gating logic must be listed. You must specify the enable identifier (ID).

`-include_ff`

Includes the flip-flops associated with the given clock-gating logic in the returning list. In other words, includes the flip-flops driven by the clock-gating logic.

Examples

- The following command obtains the identifiers of all logic instances associated with enable identifier `73411`:

  ```
  > get_list_of_cg_instances -enable_id 73411
  ```

  Returns an output similar to the following:

  ```
  73411 76067 73763
  ```

- The following command obtains the identifiers of all logic instances associated with enable identifier `73411`, including the flip-flops driven by the clock-gating logic:

  ```
  > get_list_of_cg_instances -enable_id 73411 -include_ff
  ```

  Returns an output similar to the following:

  ```
  73411 76067 73763 73235 80003 79715 79635
  ```

Related Information

`get_gating_instance_list`

get_power

```
get_power [-instance {list_of_instances}]
     [-net {list_of_nets}]
```

Traverses the hierarchy and returns the total power for the specified instances or nets of the current module. Use the `get_power_display_unit` command to show the power units.

Note: If you did not read a TCF file, LPS uses default toggle counts for each primary input and sequential cell output (see `power default prob` and `power default toggle rate` globals), performs probabilistic analysis on the rest of the design incrementally, and generates the power.

Options and Arguments

`-instance {list_of_instances}`
: Specifies the instances for which you want to obtain the power. Instances must be separated by a space.

`-net {list_of_nets}`
: Specifies the nets for which you want to obtain the power. Nets must be separated by a space. In this case, the `get_power` command reports for each net the power consumed by the capacitance of the net and the capacitance of the pins driven by the specified net.

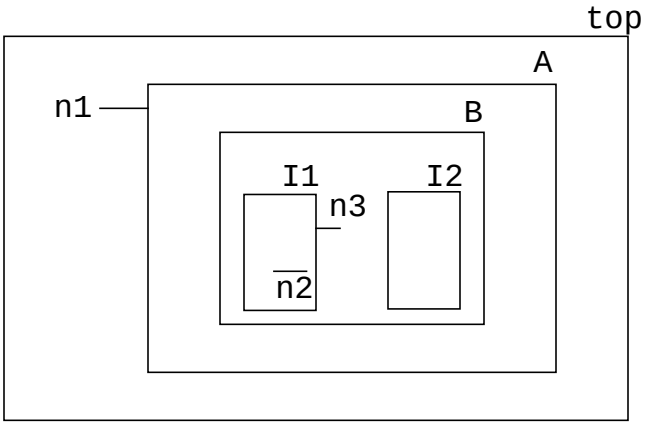
Note: If you do not use any of these options (default), the command returns the power for the all the instances and all the nets in the current instance and the hierarchy below.

Examples

For the following examples consider the schematic in Figure 4-3 on page 400.

Note: The precision of the power value is three digits. So, if you have a total power of `0.00345`, it is displayed as `0.345E-2`. You can change the units used to display the power units with the `set_power_display_unit` command.

Figure 4-3 Schematic for get_power Examples



- The following command returns the total power for all the instances and all the nets in the current module:

  ```
  > get_power
  ```

  Returns an output similar to the following:

  ```
  0.234
  ```

- The following command returns the total power for instance A/B:

  ```
  > get_power -instance A/B
  ```

  Returns an output similar to the following:

  ```
  0.119
  ```

- The following command returns the total power for instances A/B/I1, A/B/I2, and A/B:

  ```
  > get_power -instance {A/B/I1 A/B/I2 A/B}}
  ```

  Traverses the hierarchy as shown in Figure 4-3 and returns an output similar to the following:

  ```
  {3.45E-04 5.27E-04 0.119}
  ```

- The following command returns the total power for nets n1, A/B/I1/n2, and A/B/n3, where n1 is a net in the top module, n2 is a net in instance I1, and n3 is a net in instance B:

  ```
  > get_power -net {n1 A/B/I1/n2 A/B/n3}
  ```

  Returns an output similar to the following:

  ```
  {0.565E-5 0.12E-6 1.1E-6}
  ```

Related Information

`get_current_instance`

`power_internal_power_scaling` `global`

`read_tcf`

`report_power`

`report_slew_for_power_analysis`

get_power_display_unit

```
get_power_display_unit
```

Returns the power display unit.

Example

The following command displays the units used in the power reports.

```
> get_power_display_unit
```

Returns an output similar to the following:

```
mW
```

where `mW` is the value you specified with the `set_power_display_unit` command.

Related Information

`set_power_display_unit`

get_power_optimization_options

```
get_power_optimization_options [-tcl_list]
```

Displays the current settings you have for power optimization.

Options and Arguments

`-tcl_list`

Displays the settings of the power optimization options in Tcl list format.
Default: The options and their values are shown in a tabular format.

Examples

- In the following example, the first command instructs LPS to focus on optimizing the leakage power only, the second command shows the weight factors used for the power components during power optimization in tabular format.

```
> set_power_optimization_options -instance_leakage_power 1
> get_power_optimization_options
```

Returns the following:

```
 +--------------------------------------+
 |      Power Optimization Options      |
 |--------------------------------------|
 |         Options          |   Value   |
 |--------------------------+-----------|
 | -instance_internal_power |    0.00   |
 |  -instance_leakage_power |    1.00   |
 |        -net_power        |    0.00   |
 +--------------------------------------+
```

- The following command returns the previous information in Tcl list format.

```
> get_power_optimization_options -tcl_list
```

Returns the following:

```
{-instance_internal_power 0.00}   {-instance_leakage_power 1.00}
{-net_power 0.00}
```

Related Information

set_power_optimization_options

get_sleep_mode_instance_list

```
get_sleep_mode_instance_list
     [-instance {list_of_instances}][-hierarchy]
```

Returns a list of object identifiers (IDs) for each sleep mode module in the following form:

```
{sleepMode_1 dataBus_1 enable_1_1 enable_1_2 ... enable_1_k}
{sleepMode_2 dataBus_2 enable_2_1 enable_2_2 ... enable_1_k}
```

The actual enable function of a sleep mode module may be a complex Boolean function of a set of signals in the original design. In this case, the number of enable signals reported for that sleep mode module is more than one.

For example, for the `sleepMode_1` sleep module, the corresponding controlled data bus identifier is `dataBus_1`. The enable function for this module is a complex Boolean function of the signals `enable_1_1`, `enable_1_2`, ...

Note: You must run `do_build_generic -sleep_mode` before you can use this command.

Options and Arguments

`-hierarchy`
: Traverses the hierarchy (that is, all sub-modules below the current module) and reports all sleep mode modules found.
Default: The report is run on the current module.

`-instance {`*`list_of_instances`*
: Reports all sleep mode modules in the specified instances. You must specify the instances using their identifiers (IDs).
Default: Reports all sleep mode modules starting from the current module.

Example

The following command reports all sleep mode modules found in the hierarchy of the current module.

```
> get_sleep_mode_instance_list -hier
```

Returns an output similar to the following:

```
{245859 227748 227813 227781} {247475 227188 227813 227781}
```

In this case, two sleep-mode modules are found.The first number in each list identifies the sleep-mode module. The second number identifies the data bus. The next two numbers identify the enable signals.

Related Information

`do_build_generic -sleep_mode`

get_sleep_mode_options

```
get_sleep_mode_options [-tcl_list]
```

Lists all sleep mode options you have set using the `set_sleep_mode_options` command.

Options and Arguments

`-tcl_list`

Displays the settings of the sleep mode options in Tcl list format. *Default*: The options and their values are shown in a tabular format.

Examples

- The following command lists all sleep mode options in Tcl list format:

  ```
  > get_sleep_mode_options -tcl_list
  ```

  Returns the following output:

  ```
  {-no_dissolve true}    {-timing_driven full}    {-force NOT_SET}
  {-remove_all NOT_SET}    {-ignore NOT_SET}
  ```

- The following command lists all sleep mode options in tabular format:

  ```
  > get_sleep_mode_options
  ```

  Returns the following output:

  ```
  +-----------------------------+
  |     Sleep Mode Options      |
  |-----------------------------|
  |    Options     |   Value    |
  |----------------+------------|
  |  -no_dissolve  |    true    |
  | -timing_driven |    full    |
  +-----------------------------+
  ```

  The tabular format displays only the options that were set.

Related Information

`set_sleep_mode_options`

read_saif

`read_saif -instance` *`instance_name saif_file`*

Reads switching activity information in Synopsys activity interchange format (SAIF) for power estimation.

The SAIF file provides more detailed information about the switching behavior of nets and ports, which leads to more accurate power estimation. Besides net and port signal probability and transition activity information, the SAIF file also contains the transition activities of path and pin power arcs. If the cell has conditional leakage power, the SAIF file contains the signal probabilities of these when conditions.

Important

For the power calculation, LPS assumes that you specify the power arcs for each port and the when conditions for the leakage power in SAIF file, in the same order as they appear in the technology library.

After reading the SAIF file, you can use the `get_power` or `report_power` commands to get power estimates.

Options and Arguments

`-instance` *`instance_name`*

Specifies the top module in the design. Activities are asserted and events observed on all the nets in this module and in the instances that belong to the hierarchy of this module.

`saif_file`

Specifies the file containing the switching activity information. The file must be in SAIF format. The file can have any name, suffix, or length.

Example

In the following example, ports A, B, and C have pin-based power arcs, while port Y has path-based power arcs. The SAIF file contains the transition activities for each power arc. For the leakage power, the file also contains the signal probability of the when condition. This additional information allows for a more accurate power estimation, because the tool does not need to calculate the transition activity of the power arcs based on the ports signal probability and transition activities. Similarly, for leakage power, instead of calculating the probability of the when conditions, the tool can derive the probability directly from the SAIF file.

```
(INSTANCE U111
     (PORT
          (C
               (T0 199650) (T1 250) (TX 100)
               (COND (!B*!A) (RISE) (TC 0) (IG 0)
               COND (!B*!A) (FALL) (TC 0) (IG 0)
               )
          )
          (A
               (T0 199650) (T1 250) (TX 100)
               (COND ((!C*B) | (!C*!B) | (C*B)) (RISE) (TC 1) (IG 0)
               COND ((!C*B) | (!C*!B) | (C*B)) (FALL) (TC 2) (IG 0)
               )
          )
          (Y
               (T0 199650) (T1 250) (TX 100)
               (COND ((B*A) | (!B*A) | (B*!A)) (RISE)
               (IOPATH C (TC 1) (IG 0)
               )
               COND ((B*A) | (!B*A) | (B*!A)) (FALL)
               (IOPATH C (TC 0) (IG 0)
               )
               COND (C*!A) (RISE)
               (IOPATH B (TC 0) (IG 0)
               )
               COND (C*!A) (FALL)
               (IOPATH B (TC 0) (IG 0)
               )
               COND (C*!B) (RISE)
               (IOPATH A (TC 0) (IG 0)
               )
               COND (C*!B) (FALL)
               (IOPATH A (TC 0) (IG 0)
               )
               COND_DEFAULT (TC 2) (IG 0)
               )
          )
          (B
               (T0 199850) (T1 0) (TX 150)
               (COND ((!C*A) | (!C*!A) | (C*A)) (RISE) (TC 0) (IG 0)
               COND ((!C*A) | (!C*!A) | (C*A)) (FALL) (TC 0) (IG 0)
               )
          )
     )
     (LEAKAGE
          (COND (A *B *C) (T0 155000) (T1 45000)
          COND (!A *B *!C) (T0 50000) (T1 150000)
          )
     )
)
```

Related Information

get_power

report_power

read_tcf

```
read_tcf [-hier_down] [-scale scale] tcf_file
```

Reads toggle counts from the specified file and puts assertions into the database, so they can be used for power estimation and optimization.

To run power analysis or optimization, you should read in a toggle count format (TCF) file. Without a TCF file, the tool uses default toggle count values at the primary inputs and sequential cell outputs and performs probabilistic analysis after that.

Note: You must read the synthesis library and the netlist before you can use this command.

Options and Arguments

`-hier_down`

Traverses the hierarchy and sets the same probability and toggle count values on nets or pins whose names match the net or pin name with the wild card in the TCF file.
Default: The activity values are only added to the current module, even if the module is hierarchical.

For example, assume your TCF file contains

```
instance (a/b/*) {
  net {
    n* : "0.5 500";
    }
}
```

With `hier_down` specified, the tool traversers all the instances under `a/b` and sets the probability and toggle count values to 0.5 and 500 respectively, on all the nets starting with `n*`.
Default: Only the instances in `a/b` are annotated but not the instances in the hierarchy below these instances.

`-scale` *`scale`*

Scales the simulation duration. Use a positive floating number.
Default: 1.0

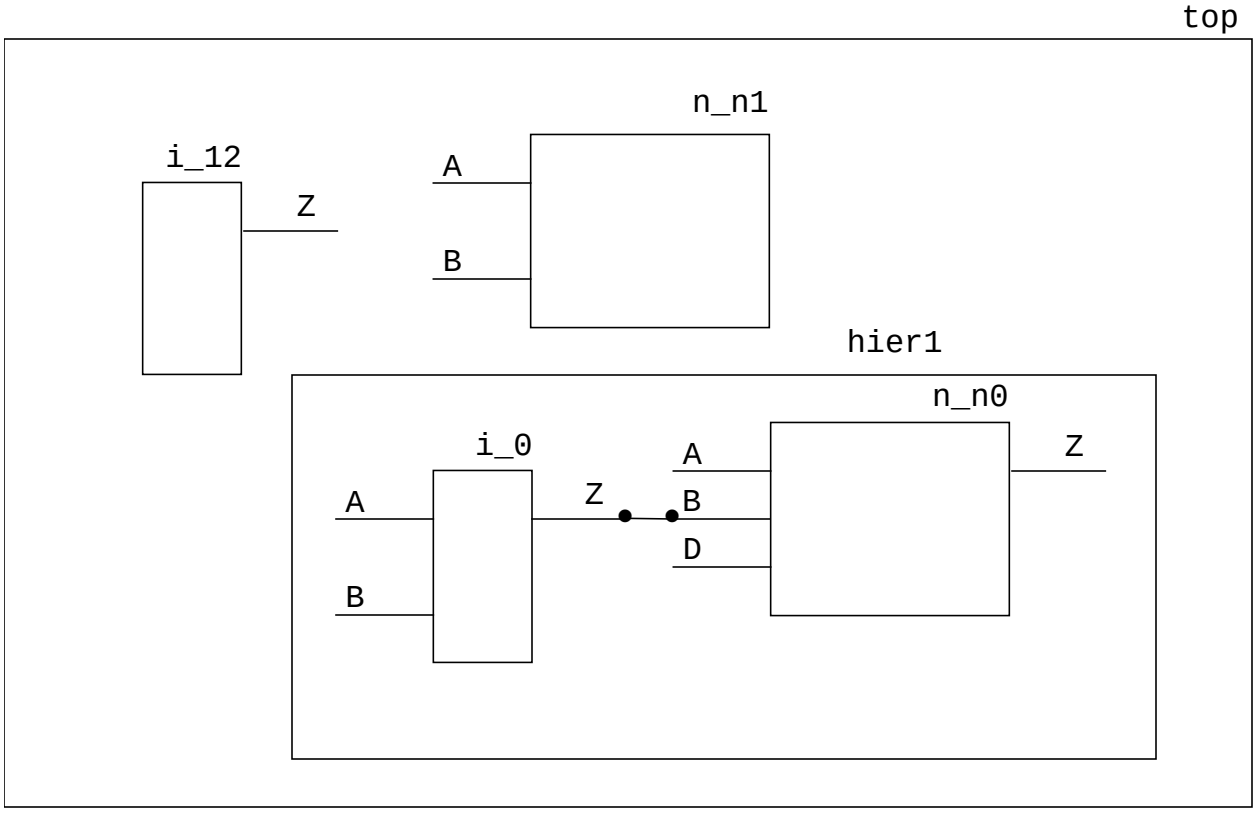
`tcf_file`

Specifies the file containing the toggle counts. The file must be in TCF format. The file can have any name, suffix, or length.

Examples

The following examples refer to the schematic shown in Figure 4-4 on page 410.

Figure 4-4 Schematic for read_tcf Example



- In the following TCF file, the `No` value for `HierDown` indicates not to read this file with the `-hier_down` option:

```
tcffile () {
   tcfversion   :       "1.0";
   generator    :       "BGPower Verilog PLI";
   date         :       "Wed Aug 9 16:45:44 2000";
   duration     :       "1.500000e+05";
   unit         :       "ns";
   HierDown    :       "No";
   instance () {
     pin () {
         "i_12/Z"       :       "0.566 747";
         "n_n1/B"       :       "0.516 475";
         "hier1/i_0/Z"  :       "0.5 500";
         "hier1/n_n0/Z" :       "0.5 500";
         "hier1/n_n0/A" :       "0.5 500";
         "hier1/n_n0/D" :       "0.5 500";
         "hier1/i_0/A"  :       "0.5 500";
         "hier1/i_0/B"  :       "0.5 500";
```

```
        "n_n1/A"       :       "0.61 516";
      }
    }
 }
```

To read this TCF file (`example1.tcf`), use the following command:

```
> read_tcf example1.tcf
```

- In the following TCF file (`example2.tcf`), the duration is half of the duration of the previous example. Furthermore, the value of `HierDown` is set to `Yes`. This implies that the probability and transition count will be set to 0.5 and 500 for all the pins in the hierarchy of `hier1`.

```
tcffile () {
   tcfversion   :       "1.0";
   generator    :       "BGPower Verilog PLI";
   date         :       "Wed Aug 9 16:45:44 2000";
   duration     :       "0.75000e+05";
   unit         :       "ns";
   HierDown    :       "Yes";
   instance () {
     pin () {
         "i_12/Z"       :       "0.566 747";
         "n_n1/B"       :       "0.516 475";
         "hier1/*"      :       "0.5 500";
         "n_n1/A"       :       "0.61 516";
      }
    }
 }
```

To make the probability and transition density on all pins the same as in the previous example, use the following command:

```
> read_tcf -hier_down -scale 2.0 example2.tcf
```

The `-scale` option makes the new duration for this TCF file equal to 1.5e+5.

Related Information

`set_sleep_mode_options`

read_tcf_update

```
read_tcf_update [-hier_down] [-weight w]
     [-scale scale] tcf_file
```

Updates the probability and transition density of pins and nets.

Note: You must read the synthesis library and the netlist before you can use this command.

- If the probability and transition density were not previously user asserted, the new probability and transition density are given as follows:

  ```
  prob_new = prob_spec;
  td_new = td_spec;
  ```

  where `prob_new` and `td_new` are the new values, and `prob_spec` and `td_spec` are the values specified in the TCF file.

- If the probability and transition density were previously user asserted, the new probability and transition density are calculated as follows:

  ```
  prob_new = (prob_prev + w * prob_spec)/(1+w)
  td_new = (td_prev + w * td_spec)/(1+w)
  ```

  where `prob_prev` and `td_prev` are the previous values, and `prob_spec` and `td_spec` are the values specified in the new TCF file.

  In the database, the transition activity (count) is stored as transition density. However, for writing the TCF file using the `write_tcf` command, the duration is needed to convert transition density into transition count. For this purpose, the duration is internally stored. This duration is updated as follows:

  - If duration was not previously stored:

    ```
    duration_new = duration_spec
    ```

  - If duration was previously stored:

    ```
    duration_new = ( 1 + w) * duration_old
    ```

    For example, if you specify

    ```
    w = (simulationTime_spec)/(simulationTime_prev)
    ```

    (where `simulationTime_spec` and `simulationTime_prev` are the new and old simulation period) then `prob_new` and `td_new` will be as follows:

    ```
    prob_new = (prob_prev * simulationTime_prev + prob_spec * simulationTime_spec)
    /(simulationTime_prev + simulationTime_spec);
    td_new = (td_prev * simulationTime_prev +
    td_spec*simulationTime_spec)/(simulationTime_prev + simulationTime_spec);
    duration_new = simulationTime_prev + simulationTime_spec
    ```

    In the example above, `prob` and `td` are updated based on the simulation times.

Options and Arguments

`-hier_down`

Traverses the hierarchy and sets the same probability and toggle count values on nets or pins whose names match the net or pin name with the wild card in the TCF file**.**
Default: The activity values are only added to the current module, even if the module is hierarchical.

For example, assume your TCF file contains

```
instance (a/b/*) {
  net {
    n* : "0.5 500";
  }
}
```

With `hier_down` specified, the tool traversers all the instances under `a/b` and sets the probability and toggle count values to 0.5 and 500 respectively, on all the nets starting with `n*`.
Default: Only the instances in `a/b` are annotated but not the instances in the hierarchy below these instances.

`-scale` *`scale_factor`*

Scales the simulation time in the TCF file. Use a positive floating number.
Default: `1.0`

`tcf_file`

Specifies the file containing the toggle counts to be used. There are no restrictions on filenames or length of the filenames. The toggle count file does not need to use a `.tcf` suffix, but must be in TCF format.

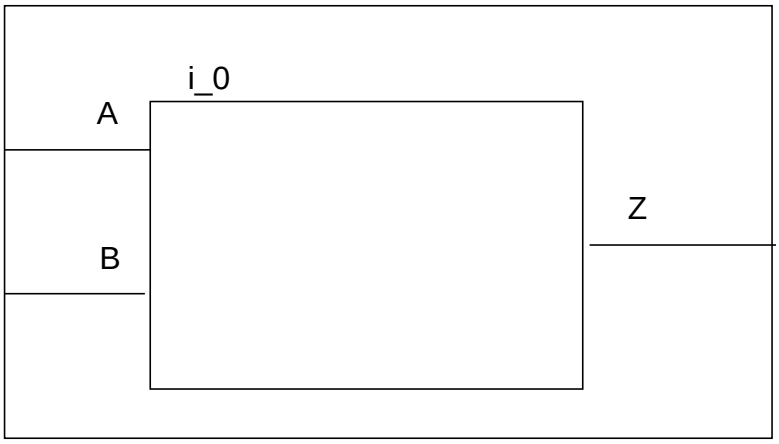
`-weight` *`w`*

Specifies the weight value. Use a positive floating number.
Default: `1.0`

Examples

Consider the schematic shown in Figure 4-5 on page 414.

Figure 4-5 Schematic for read_tcf_update Example



- Assume you have the following TCF file (`example.tcf`) for this schematic:

```
tcffile () {
   tcfversion   :       "1.0";
   generator    :       "BGPower Verilog PLI";
   date         :       "Wed Aug 9 16:45:44 2000";
   duration     :       "1.000000e+05";
   unit         :       "ns";
   HierDown    :       "No";
   instance () {
     pin () {
         "i_0/A"        :       "0.5 500";
         "i_0/B"        :       "0.6 600";
         "i_0/Z"        :       "0.7 700";
      }
    }
 }
```

Now execute the following command.

```
> read_tcf_update -weight 0.5 example.tcf
```

The duration and transition counts in the new TCF file are updated as follows:

```
tcffile () {
   tcfversion   :       "1.0";
   generator    :       "BGPower Verilog PLI";
   date         :       "Wed Aug 9 16:45:44 2000";
   duration     :       "1.500000e+05";
   unit         :       "ns";
   HierDown    :       "No";
   instance () {
     pin () {
        "i_0/A"        :       "0.5 750";
        "i_0/B"        :       "0.6 900";
        "i_0/Z"        :       "0.7 1050";
      }
    }
 }
```

- Consider the original TCF file again and execute the following command:

```
> read_tcf_update -weight 0.5 -scale 2 example.tcf
```

The duration and transition counts in the new TCF file are now updated as follows:

```
tcffile () {
   tcfversion   :       "1.0";
   generator    :       "BGPower Verilog PLI";
   date         :       "Wed Aug 9 16:45:44 2000";
   duration     :       "1.500000e+05";
   unit         :       "ns";
   HierDown    :       "No";
   instance () {
     pin () {
         "i_0/A"        :       "0.5 417";
         "i_0/B"        :       "0.6 500";
         "i_0/Z"        :       "0.7 584";
      }
    }
 }
```

Related Information

read_tcf

set_switching_activity

write_tcf

read_vcd

```
read_vcd [-static | -dynamic] -module module_name
     [-scale scale] [-start_time start_monitoring_time]
     [-end_time end_monitoring_time] [-time_window time]
      vcd_file
```

Reads a Value Change Dump (VCD) file to either assert switching activities on nets or pins, or estimate the average cycle-by-cycle power of the design.

Use this command for either the static or dynamic power analysis:

- For static power analysis (to compute the signal probabilities and switching activities from the VCD file), use `read_vcd -static`. Follow this command by commands to estimate static power (for example, `get_power`)
- For average dynamic (or average cycle-by-cycle) power estimation, use `read_vcd -dynamic`. Follow this command by commands to estimate dynamic power (for example, `get_dynamic_power` or `get_dynamic_peak_power`)

  This command also reports the percentage of nets in the VCD file that have switching information.

Options and Arguments

`-dynamic`
: Stores the necessary information to perform cycle-by-cycle power estimation on the specified instances and nets.

`-end_time` *`end_monitoring_time`*
: Specifies the time you want to end monitoring the switching activities or events. Specify a value larger than zero in nanoseconds. If you do not specify this option, the activities or events are monitored till the end (last timestamp of the VCD file).

`-module` *`module_name`*
: Specifies the top module in the design. Activities are asserted and events observed on all the nets in this module and in the instances that belong to the hierarchy of this module.

`-scale` *`scale_factor`*
: Scales the simulation duration. Use a positive floating number. *Default*: 1.0

`-start_time` *`start_monitoring_time`*

Specifies the time you want to start monitoring the switching activities or events. Specify a value equal to or larger than zero in nanoseconds. If you do not specify this option, the first timestamp in the VCD file is considered as the start time to monitor.

`-static`

Calculates the switching activity and transition density of each of the nets and pins from the time you want to start monitoring the switching activities to the time you want to stop monitoring, and then stores the information as assertions on the nets and pins. This is the default option.

`-time_window` *`window`*

Specifies the time increment, in nanoseconds, in which you want LPS to divide the period during which the events are monitored. The specified time window must be larger than zero for LPS to calculate the dynamic power for each time window.

`vcd_file`

Specifies the name of the value change dump file.

Related Information

`get_dynamic_peak_power`

`get_dynamic_power`

`get_power`

`lpsvcd2tcf`

`read_tcf`

report_clock_gating

```
report_clock_gating [-detailed] [-not_gated] [-gated]
     [-bank_width number]
```

Reports clock gating information for the design. The information depends on whether the clock gating has been inserted or not:

- For designs without clock-gating insertion, reports the percentage of register banks that can potentially be clock-gated or not.
- For designs with clock-gating logic inserted, the information depends on whether the report is requested for registers that are gated or not.
  - **Gated**—Reports for each clock-gating domain, the instance name of the gating cell, the net that enables the clock-gating logic, the size of the register bank that is clock gated, and whether the clock gating was forced.
  - **Not gated**—Reports whether clock gating was forced on the registers or not.

To adjust the format of the table generated with `report_clock_gating`, specify `popt_report_clock_gating_table` as the table name when using the `set_table_style` command.

Options and Arguments

`-bank_width` *number*

Creates two categories of register banks for the report: one category with a width larger than or equal to the specified width, and one category with a width smaller than the specified width.

If no value is specified, the bank width defaults to the value of the `-minsize` option of the `set_clock_gating_options` command. If that option was omitted, the number defaults to 3.

`-detailed`

Adds the names of registers to the report.

Before clock-gating insertion, reports the registers for each category. See `-bank_width` option and Examples on page 419 for more information.

After clock-gating insertion, reports the registers in each clock-gating domain, and the library cell used for the clock-gating cell.

`-gated` Reports for each clock-gating domain, the instance name of the gating cell, the net that enables the clock-gating logic, the size of the register bank that is clock gated, and whether the clock gating is forced by the user.

Only applies if clock-gating logic was inserted.

`-not_gated` Reports for each register that could not be clock gated, whether the user tried to force clock gating on the register or not.

Only applies if clock-gating logic was inserted.

Examples

- The following command reports the clock-gating candidates before clock-gating insertion:

```
report_clock_gating
+-------------------------------------------------------------------------+
|                          Clock Gating Potential Report                  |
|-------------------------------------------------------------------------|
|                    Category                      |  Number of | Percentage |
|                                                  |  Domains   |     of     |
|                                                  |            |  Registers |
|--------------------------------------------------+------------+------------|
| Clock gating candidates with bank width >= 3     | 4          | 80.769231% |
| Clock gating candidates with bank width < 3      | 0          | 0.000000%  |
| Clock gating skipped (Const enable/User ignore) | 1          | 19.230769% |
+-------------------------------------------------------------------------+
```

- The following command gives a detailed report of the clock-gating candidates before clock-gating insertion:

```
report_clock_gating -detailed
+-------------------------------------------------------------------------+
|                          Clock Gating Potential Report                  |
|-------------------------------------------------------------------------|
|                    Category                      |  Number of | Percentage |
|                                                  |  Domains   |     of     |
|                                                  |            |  Registers |
|--------------------------------------------------+------------+------------|
| Clock gating candidates with bank width >= 3     | 4          | 80.769231% |
| Clock gating candidates with bank width < 3      | 0          | 0.000000%  |
| Clock gating skipped (Const enable/User ignore) | 1          | 19.230769% |
+-------------------------------------------------------------------------+

Clock Gatable registers with bank width >= 3 are :
   { regs/small_active/data_out_reg_2 regs/small_active/data_out_reg_1
regs/small_active/data_out_reg_0}
   { regs/big_normal/data_out_reg_7 regs/big_normal/data_out_reg_6
regs/big_normal/data_out_reg_5 regs/big_normal/data_out_reg_4
regs/big_normal/data_out_reg_3 regs/big_normal/data_out_reg_2
regs/big_normal/data_out_reg_1 regs/big_normal/data_out_reg_0}

Clock Gating skipped for following registers :
   {  ctrl/STATE_reg_4 ctrl/STATE_reg_4 ctrl/STATE_reg_4 ctrl/STATE_reg_4 ctrl/STATE_reg_4}
   <POPT-027>.
```

- The following command reports the inserted clock-gating logic:

```
report_clock_gating
```

```
+-------------------------------------------------------------+
|                  Clock Gating Summary Report                |
|-------------------------------------------------------------|
| Total Number of Gating Cells : 4                            |
| Total Number of Registers : 26                              |
| Total Number of Gated Reg (bank width >= 3) : 21 (80.769231%) |
| Total Number of Gated Reg (bank width < 3) : 0 (0.000000%)  |
| Total Number of Non Gated Reg : 5 (19.230769%)              |
+-------------------------------------------------------------+
+-------------------------------------------------------------------------------------------+
|                          Clock Gating Report For Gated Registers                          |
|-------------------------------------------------------------------------------------------|
| CG Domain | Enable Net|           CG Instance                        | CG Width |   Force|
|           |           |                                              |          |   Gated|
|-------------------------------------------------------------------------------------------|
| Domain #0  | enable    | regs/medium_less_active/i_2_LPS_CG_GATING_1 | 5        | N      |
| Domain #1  | enable    | regs/medium_constrained/i_2_LPS_CG_GATING_1 | 5        | N      |
| Domain #2  | enable    | regs/small_active/i_2_LPS_CG_GATING_1       | 3        | N      |
| Domain #3  | enable    | regs/big_normal/i_2_LPS_CG_GATING_1         | 8        | N      |
+-------------------------------------------------------------------------------------------+
```

- The following command gives a detailed report of the inserted clock-gating logic:

```
report_clock_gating -detailed
```

```
+-------------------------------------------------------------+
|                Clock Gating Summary Report                  |
|-------------------------------------------------------------|
| Total Number of Gating Cells : 4                            |
| Total Number of Registers : 26                              |
| Total Number of Gated Reg (bank width >= 3) : 21 (80.769231%) |
| Total Number of Gated Reg (bank width < 3) : 0 (0.000000%)  |
| Total Number of Non Gated Reg : 5 (19.230769%)              |
+-------------------------------------------------------------+
+-----------------------------------------------------------------------------------------------------------+
|                                Clock Gating Report For Gated Registers                                    |
|-----------------------------------------------------------------------------------------------------------|
|  CG Domain | Enable Net |                CG Instance                |  CG Width  |   Force    |  CG Cell   |
|            |            |                                           |            |   Gated    |            |
|------------+------------+-------------------------------------------+------------+------------+------------|
| Domain #0  | enable     | regs/medium_less_active/i_2_LPS_CG_GATING_1 | 5          | N          | OAI21XL    |
| Domain #1  | enable     | regs/medium_constrained/i_2_LPS_CG_GATING_1 | 5          | N          | OAI21XL    |
| Domain #2  | enable     | regs/small_active/i_2_LPS_CG_GATING_1       | 3          | N          | OAI21XL    |
| Domain #3  | enable     | regs/big_normal/i_2_LPS_CG_GATING_1         | 8          | N          | OAI21XL    |
+-----------------------------------------------------------------------------------------------------------+
```

```
       The following register groups have been clock gated :

   { regs/medium_less_active/data_out_reg_0 regs/medium_less_active/data_out_reg_1 regs/medium_less_active/
        data_out_reg_2 regs/medium_less_active/data_out_reg_3 regs/medium_less_active/data_out_reg_4 }
           { regs/medium_constrained/data_out_reg_0 regs/medium_constrained/data_out_reg_1 regs/medium_constrained/
        data_out_reg_2 regs/medium_constrained/data_out_reg_3 regs/medium_constrained/data_out_reg_4 }
          { regs/small_active/data_out_reg_0 regs/small_active/data_out_reg_1 regs/small_active/data_out_reg_2 }
          { regs/big_normal/data_out_reg_0 regs/big_normal/data_out_reg_1 regs/big_normal/data_out_reg_2 regs/big_
        normal/data_out_reg_3 regs/big_normal/data_out_reg_4
        regs/big_normal/data_out_reg_5regs/big_normal/data_out_reg_6 regs/big_normal/data_out_reg_7 }  <POPT-027>.
```

- The following command reports the registers that could not be gated:

```
report_clock_gating -not_gated
```

```
+------------------------------+
|  Clock Gating Report For Non |
|        Gated Registers       |
|------------------------------|
|     FF name      |   Force   |
|                  |  Nongated |
|------------------+-----------|
```

```
| ctrl/STATE_reg_0 | N          |
| ctrl/STATE_reg_1 | N          |
| ctrl/STATE_reg_2 | N          |
| ctrl/STATE_reg_3 | N          |
| ctrl/STATE_reg_4 | N          |
+------------------------------+
```

Related Information

set_clock_gating_options

report_power

```
report_power [-instance {list_of_instances}]
     [-net {list_of_nets}] [-maxcount nworst] [-hier]
     [>|>> report_file] [-tcl_list] -dynamic
```

Reports the power consumption for the specified instances.
Default: The `report_power` command provides the total power of the current module.

Note: Before you use `report_power`, you must set the top timing module correctly because the power for each net is computed by using the assertions set with respect to the top timing module. It is not necessary to set it to a specific module, but you must set it to manipulate the power assertions only.

If you did not read a TCF file, LPS uses default toggle counts for each primary input and sequential cell output (specified through the `power_default_prob` and `power_default_toggle_rate` globals).

The following table indicates the table names to be specified with the `set_table_style` command if you want to adjust the format of the tables.

Table Name for set_table_style	Corresponds to Second Table for
`report_power_inst`	`report_power [-instance]`
`report_power_net`	`report_power -net`

Options and Arguments

`-dynamic`

Reports the dynamic power estimation. You must first run the `read_vcd -dynamic` command, otherwise the command reports zero numbers.

`-hier`

Works in conjunction with `-instance` and `-net` options.

Traverses the hierarchy starting from the current module to report the power consumption of all instances (`-instance`) or of all nets (`-net`) in the hierarchy below the current module.
Default: Power consumption is only reported in the instances or nets of the current module without traversing the hierarchy below the current module.

`-instance {`*`list_of_instances`*`}`
Specifies the instances for which to report the power dissipation.
Default: The power is only reported for the current module. With the `-hier` option, the hierarchy is traversed.
Default: The power for each instance in current module is reported.

`-maxcount` *`nworst`*
Limits the number of entries in the report to the specified number that have the worst (highest) power value.
Default: The five worst power values for all instances in your design are reported.

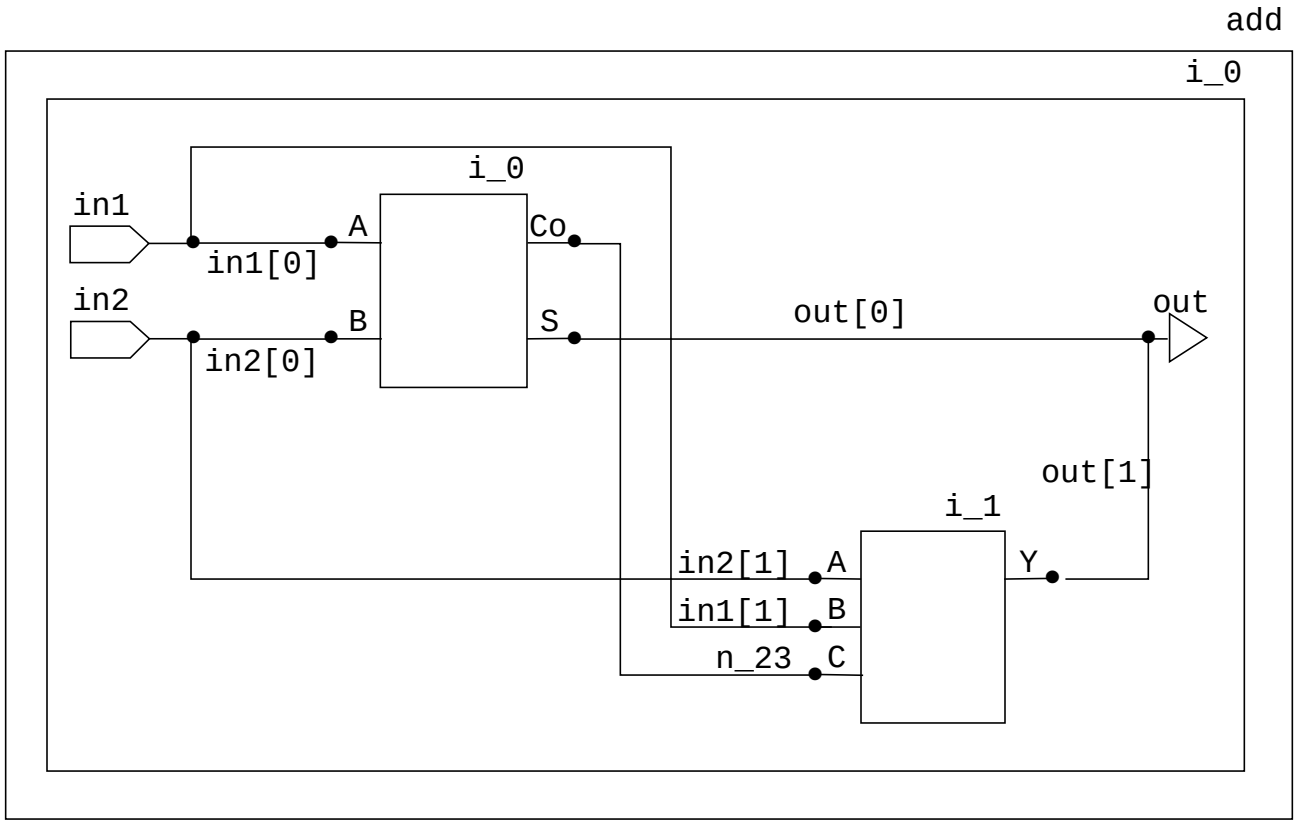
`-net {`*`list_of_nets`*`}`
Specifies the nets for which to report net statistics, such as the signal probability, toggle rate, net capacitance, and the total power of each of the nets. If you do not specify any net with this option, the power for each net in the current module is reported. In this case, the `report_power` command reports for each net the power consumed by the capacitance of the net and the capacitance of the pins driven by the specified net.

`report_file`
Specifies the name of the file to which to write the report. The filename must be the last argument included with this command.
Default: The report is written to `stdout`.

`-tcl_list`
Displays the report in Tcl list format.
Default: The report is shown in a tabular format.

Examples

Figure 4-6 Schematic for report_power Examples



- The following command reports the power of the current module to file `rep.txt`:

```
> report_power > rep.txt
```

```
+-------------------------------------------+
| Report               | report_power       |
|----------------------+--------------------|
| Options              | > rep.txt          |
+----------------------+--------------------+
| Date                 | 20020220.151310    |
| Tool                 | pks_shell          |
| Release              | v5.0-b012          |
| Version              | Feb 19 2002 02:05:47 |
+----------------------+--------------------+
| Module               | add                |
| Operating Condition  | slow               |
| Process              | 1.000000           |
| Voltage              | 1.620000           |
| Temperature          | 125.000000         |
| time unit            | 1.00 ns            |
| capacitance unit     | 1.00 pF            |
| power unit           | 1.00uW             |
+-------------------------------------------+
```

```
+-------------------------------------------------------------------------------------+
|                                        add                                          |
|-------------------------------------------------------------------------------------+
|                    |          | Library | Internal |   Leakage  |   Net    |   Total  |
|                    |          |         |   Cell   |            |          |          |
|--------------------+----------+---------+----------+------------+----------+----------|
| Module             | Instance |  Cell   | Power(uW)| Power(uW)  | Power(uW)| Power(uW)|
|--------------------+----------+---------+----------+------------+----------+----------|
| add                |          |         |   0.8737 |  2.916e-04 |   0.4229 |    1.296 |
+-------------------------------------------------------------------------------------+
```

- The following command reports the power of all instances in the current module:

```
> report_power -instance
```

```
+-----------------------------------------+
| Report              | report_power      |
|---------------------+-------------------|
| Options             | -inst > rep.txt   |
+---------------------+-------------------+
| Date                | 20020220.150657   |
| Tool                | pks_shell         |
| Release             | v5.0-b012         |
| Version             | Feb 19 2002 02:05:47 |
+---------------------+-------------------+
| Module              | add               |
| Operating Condition | slow              |
| Process             | 1.000000          |
| Voltage             | 1.620000          |
| Temperature         | 125.000000        |
| time unit           | 1.00 ns           |
| capacitance unit    | 1.00 pF           |
| power unit          | 1.00uW            |
+-----------------------------------------+
+-------------------------------------------------------------------------------------+
|                                        add                                          |
|-------------------------------------------------------------------------------------+
|                    |          | Library | Internal |   Leakage  |   Net    |   Total  |
|                    |          |         |   Cell   |            |          |          |
|--------------------+----------+---------+----------+------------+----------+----------|
| Module             | Instance |  Cell   | Power(uW)| Power(uW)  | Power(uW)| Power(uW)|
|--------------------+----------+---------+----------+------------+----------+----------|
|AWDP_ADD_partition_0|          |         |   0.8737 |  2.916e-04 |   0.1678 |   1.0418 |
|add                 |          |         |   0.8737 |  2.916e-04 |   0.4229 |   1.2969 |
+-------------------------------------------------------------------------------------+
```

- The following command traverses all the instances in the current module and reports their power to file `rep.txt`:

```
> report_power -instance -hier > rep.txt
```

```
+-----------------------------------------+
| Report              | report_power      |
|---------------------+-------------------|
| Options             | -inst -hier > rep.txt |
+---------------------+-------------------+
| Date                | 20020220.151119   |
| Tool                | pks_shell         |
| Release             | v5.0-b012         |
| Version             | Feb 19 2002 02:05:47 |
+---------------------+-------------------+
| Module              | add               |
| Operating Condition | slow              |
| Process             | 1.000000          |
| Voltage             | 1.620000          |
| Temperature         | 125.000000        |
| time unit           | 1.00 ns           |
```

```
| capacitance unit    | 1.00 pF               |
| power unit          | 1.00uW                |
+-----------------------------------------------+
+-------------------------------------------------------------------------------------------+
|                                          add                                              |
|-------------------------------------------------------------------------------------------+
|                     |          | Library | Internal |   Leakage  |  Net     |   Total  |
|                     |          |         |   Cell   |            |          |          |
|---------------------+----------+---------+----------+------------+----------+----------|
| Module              | Instance |  Cell   | Power(uW)| Power(uW)  | Power(uW)| Power(uW)|
|---------------------+----------+---------+----------+------------+----------+----------|
|                     | i_0/i_0  | ADDHXL  |   0.0852 |  1.458e-04 |   0.1079 |   0.1933 |
|                     | i_0/i_1  | XOR3X2  |   0.7885 |  1.458e-04 |   0.0599 |   0.8485 |
|AWDP_ADD_partition_0|          |         |   0.8737 |  2.916e-04 |   0.1678 |   1.0418 |
|add                  |          |         |   0.8737 |  2.916e-04 |   0.4229 |   1.2969 |
+-------------------------------------------------------------------------------------------+
```

- The following command reports the power for instance `i_0` to file `rep.txt`:

```
> report_power -instance i_0 > rep.txt
```

```
+-----------------------------------------+
| Report              | report_power      |
|---------------------+-------------------|
| Options             | -inst i_0 > rep.txt  |
+---------------------+-------------------+
| Date                | 20020220.151447      |
| Tool                | pks_shell            |
| Release             | v5.0-b012            |
| Version             | Feb 19 2002 02:05:47 |
+---------------------+-------------------+
| Module              | add                  |
| Operating Condition | slow                 |
| Process             | 1.000000             |
| Voltage             | 1.620000             |
| Temperature         | 125.000000           |
| time unit           | 1.00 ns              |
| capacitance unit    | 1.00 pF              |
| power unit          | 1.00uW               |
+-----------------------------------------+'
+-------------------------------------------------------------------------------------------+
|                                          add                                              |
|-------------------------------------------------------------------------------------------+
|                     |          | Library | Internal |   Leakage  |  Net     |   Total  |
|                     |          |         |   Cell   |            |          |          |
|---------------------+----------+---------+----------+------------+----------+----------|
| Module              | Instance |  Cell   | Power(uW)| Power(uW)  | Power(uW)| Power(uW)|
|---------------------+----------+---------+----------+------------+----------+----------|
|AWDP_ADD_partition_0|          |         |   0.8737 |  2.916e-04 |   0.1678 |   1.0418 |
+-------------------------------------------------------------------------------------------+
```

- The following command traverses the hierarchy of instance `i_0` and reports the power to file `rep.text`:

```
> report_power -instance i_0 -hier > rep.txt
```

```
+-------------------------------------------------+
| Report              | report_power              |
|---------------------+---------------------------|
| Options             | -inst i_0 -hier > rep.txt |
+---------------------+---------------------------+
| Date                | 20020220.151727           |
| Tool                | pks_shell                 |
| Release             | v5.0-b012                 |
| Version             | Feb 19 2002 02:05:47      |
+---------------------+---------------------------+
```

```
| Module              | add                    |
| Operating Condition | slow                   |
| Process             | 1.000000               |
| Voltage             | 1.620000               |
| Temperature         | 125.000000             |
| time unit           | 1.00 ns                |
| capacitance unit    | 1.00 pF                |
| power unit          | 1.00uW                 |
+-----------------------------------------------+

+-----------------------------------------------------------------------------------+
|                                        add                                        |
|--------------------+----------+---------+----------+------------+----------+----------|
|                    |          | Library | Internal |   Leakage  |  Net     |   Total  |
|                    |          |         |   Cell   |            |          |          |
|--------------------+----------+---------+----------+------------+----------+----------|
| Module             | Instance |  Cell   | Power(uW)| Power(uW)  | Power(uW)| Power(uW)|
|--------------------+----------+---------+----------+------------+----------+----------|
|                    | i_0/i_0  | ADDHXL  |   0.0852 |  1.458e-04 |   0.1079 |   0.1933 |
|                    | i_0/i_1  | XOR3X2  |   0.7885 |  1.458e-04 |   0.0599 |   0.8485 |
|AWDP_ADD_partition_0|          |         |   0.8737 |  2.916e-04 |   0.1678 |   1.0418 |
+-----------------------------------------------------------------------------------+
```

- The following command reports the power of all the nets in the current module to file `rep.text`:

```
> report_power -net > rep.txt
```

```
+-----------------------------------------+
| Report              | report_power      |
|---------------------+-------------------|
| Options             | -net > rep.txt    |
+---------------------+-------------------+
| Date                | 20020220.151852   |
| Tool                | pks_shell         |
| Release             | v5.0-b012         |
| Version             | Feb 19 2002 02:05:47 |
+---------------------+-------------------+
| Module              | add               |
| Operating Condition | slow              |
| Process             | 1.000000          |
| Voltage             | 1.620000          |
| Temperature         | 125.000000        |
| time unit           | 1.00 ns           |
| capacitance unit    | 1.00 pF           |
| power unit          | 1.00uW            |
+-----------------------------------------+

+-----------------------------------------------------------------------------------+
|                                        add                                        |
|-------------+----------------+--------------+--------------+--------------+----------|
|     Module  |       Net      |  Probability |  Trans. Den. |  Capacitance | Power(uW)|
|-------------+----------------+--------------+--------------+--------------+----------|
|             |      out[1]    |      *0.4999 |       *0.0126 |    3.629e-03 |   0.0599 |
|             |      out[0]    |      *0.5190 |    *8.493e-03 |    3.629e-03 |   0.0404 |
|             |      in1[1]    |      *0.5025 |    *4.255e-03 |       0.0183 |   0.1023 |
|             |      in1[0]    |      *0.5065 |    *4.388e-03 |    8.209e-03 |   0.0473 |
|             |      in2[1]    |      *0.4936 |    *4.063e-03 |       0.0102 |   0.0546 |
|             |      in2[0]    |      *0.4994 |    *4.105e-03 |    9.449e-03 |   0.0509 |
+-----------------------------------------------------------------------------------+
```

- The following command traverses the hierarchy and reports the power of all the nets in the current module to file `rep.txt`:

```
    > report_power -net -hier > rep.txt
```

Report	report_power
Options	-net -hier > rep.txt
Date	20020220.152111
Tool	pks_shell
Release	v5.0-b012
Version	Feb 19 2002 02:05:47
Module	add
Operating Condition	slow
Process	1.000000
Voltage	1.620000
Temperature	125.000000
time unit	1.00 ns
capacitance unit	1.00 pF
power unit	1.00uW

add

Module	Net	Probability	Trans. Den.	Capacitance	Power(uW)
i_0	out[0]	*0.5190	*8.493e-03	3.629e-03	0.0404
i_0	n_23	*0.2435	*4.255e-03	0.0121	0.0675
i_0	out[1]	*0.4999	*0.0126	3.629e-03	0.0599
	in1[1]	*0.5025	*4.255e-03	0.0183	0.1023
	in1[0]	*0.5065	*4.388e-03	8.209e-03	0.0473
	in2[1]	*0.4936	*4.063e-03	0.0102	0.0546
	in2[0]	*0.4994	*4.105e-03	9.449e-03	0.0509

- The following command reports the power of instances `i_0/n_23` and `in1[1]`:

```
    > report_power -net {i_0/n_23 in1[1] }
```

Report	report_power
Options	-net i_0/n_23 in1[0] > rep.txt
Date	20020220.152352
Tool	pks_shell
Release	v5.0-b012
Version	Feb 19 2002 02:05:47
Module	add
Operating Condition	slow
Process	1.000000
Voltage	1.620000
Temperature	125.000000
time unit	1.00 ns
capacitance unit	1.00 pF

```
| power unit          | 1.00uW                          |
+---------------------------------------------------------+
+-------------------------------------------------------------------------------------+
|                                        add                                          |
|-------------------------------------------------------------------------------------+
|     Module   |       Net      |  Probability |  Trans. Den. |  Capacitance | Power(uW)|
|--------------+----------------+--------------+--------------+--------------+----------|
|     i_0      |       n_23     |      *0.2435 |   *4.255e-03 |       0.0121 |   0.0675 |
|              |       in1[0]   |      *0.5065 |   *4.388e-03 |    8.209e-03 |   0.0473 |
+-------------------------------------------------------------------------------------- +
```

- The following command reports both the net power and instance power to file `rep.txt`:

```
> report_power -net -instance i_0 > rep.txt
```

```
+-------------------------------------------------+
| Report               | report_power             |
|----------------------+--------------------------|
| Options              | -net -inst i_0 > rep.txt |
+----------------------+--------------------------+
| Date                 | 20020220.153024          |
| Tool                 | pks_shell                |
| Release              | v5.0-b012                |
| Version              | Feb 19 2002 02:05:47     |
+----------------------+--------------------------+
| Module               | add                      |
| Operating Condition  | slow                     |
| Process              | 1.000000                 |
| Voltage              | 1.620000                 |
| Temperature          | 125.000000               |
| time unit            | 1.00 ns                  |
| capacitance unit     | 1.00 pF                  |
| power unit           | 1.00uW                   |
+-------------------------------------------------+
```

```
+-------------------------------------------------------------------------------------+
|                                        add                                          |
|-------------------------------------------------------------------------------------+
|     Module   |       Net      |  Probability |  Trans. Den. |  Capacitance | Power(uW)|
|--------------+----------------+--------------+--------------+--------------+----------|
|              |     out[1]     |      *0.4999 |      *0.0126 |    3.629e-03 |   0.0599 |
|              |     out[0]     |      *0.5190 |   *8.493e-03 |    3.629e-03 |   0.0404 |
|              |     in1[1]     |      *0.5025 |   *4.255e-03 |       0.0183 |   0.1023 |
|              |     in1[0]     |      *0.5065 |   *4.388e-03 |    8.209e-03 |   0.0473 |
|              |     in2[1]     |      *0.4936 |   *4.063e-03 |       0.0102 |   0.0546 |
|              |     in2[0]     |      *0.4994 |   *4.105e-03 |    9.449e-03 |   0.0509 |
+-------------------------------------------------------------------------------------+
```

```
+-------------------------------------------------------------------------------------+
|                                        add                                          |
|-------------------------------------------------------------------------------------+
|                      |         | Library | Internal |   Leakage  |  Net     |   Total  |
|                      |         |         |   Cell   |            |          |          |
|----------------------+---------+---------+----------+------------+----------+----------|
| Module               |Instance |  Cell   | Power(uW)| Power(uW)  | Power(uW)| Power(uW)|
|----------------------+---------+---------+----------+------------+----------+----------|
|AWDP_ADD_partition_0 |         |         |   0.8737 |  2.916e-04 |   0.1678 |   1.0418 |
+-------------------------------------------------------------------------------------+
```

- The following command reports the three nets with the worst power:

```
> report_power -net -maxcount 3
```

```
+-------------------------------------------------+
| Report               | report_power             |
```

Options	-net -maxcount 3 > rep.txt
Date	20020220.153921
Tool	pks_shell
Release	v5.0-b012
Version	Feb 19 2002 02:05:47
Module	add
Operating Condition	slow
Process	1.000000
Voltage	1.620000
Temperature	125.000000
time unit	1.00 ns
capacitance unit	1.00 pF
power unit	1.00uW

add					
Module	Net	Probability	Trans. Den.	Capacitance	Power(uW)
	in1[1]	*0.5025	*4.255e-03	0.0183	0.1023
	out[1]	*0.4999	*0.0126	3.629e-03	0.0599
	in2[1]	*0.4936	*4.063e-03	0.0102	0.0546

Related Information

get_power

power_internal_power_scaling global

report_slew_for_power_analysis

```
report_slew_for_power_analysis
     [-instance {list_of_instances}] [-rise] [-fall]
     { > | >> } report_file
```

Reports the user-asserted slews and the set of instances with the asserted value.

Note: You must read the synthesis library and the netlist before you can use this command.

The following table indicates the table names to be specified with the `set_table_style` command if you want to adjust the format of the tables generated with `report_slew_for_power_analysis`.

Table Name for set_table_style	Corresponds to Table for
`pa_module_rise_fall_slew_table`	`-fall -rise`
`pa_module_fall_slew_table`	`-fall`
`pa_module_rise_slew_table`	`-rise`
`pa_inst_rise_fall_slew_table`	`-instance {`*`instances`*`} -fall -rise`
`pa_inst_fall_slew_table`	`-instance {`*`instances`*`} -fall`
`pa_inst_rise_slew_table`	`-instance {`*`instances`*`} -rise`

Options and Arguments

`-fall`

Reports the slew time for falling edges.

If neither `-fall` or `-rise` are specified, both the user-defined rise and fall slews are reported.

`-instance {`*`list_of_instances`*

Specifies the instance for which the report is to be run.
Default: The report is given for all instances in the current module.

`-rise`

Reports the slew time for rising edges.

If neither `-fall` or `-rise` are specified, both the user-defined rise and fall slews are reported.

Examples

Consider the schematic shown in Figure 4-6 on page 424 again. In the following examples, you can assume that you set rise and fall slew values on the instances `i_0/i_1` and `i_0/i_0`.

- The following command reports the rise and fall slew values for instance `i_0/i_0`:

  ```
  > report_slew_for_power_analysis -rise -fall -instance i_0/i_0
  ```

  Returns the following output:

  ```
  +-------------------------------------------+
  |          User Defined Slew Values         |
  |-------------------------------------------|
  |   Instance   |     Rise     |     Fall     |
  |--------------+--------------+--------------|
  |   i_0/i_0    |   0.100000   |   0.200000   |
  +-------------------------------------------+
  ```

- The following command reports the rise slew value for instance `i_0/i_0`:

  ```
  > report_slew_for_power_analysis -rise -instance i_0/i_0
  ```

  Returns the following output:

  ```
  +-----------------------------+
  |  User Defined Slew Values   |
  |-----------------------------|
  |    Instance  |     Rise     |
  |--------------+--------------|
  |   i_0/i_0    |   0.100000   |
  +-----------------------------+
  ```

- The following command reports the fall slew value for instance `i_0/i_0`:

  ```
  > report_slew_for_power_analysis -fall -instance i_0/i_0
  ```

  Returns the following output:

  ```
  +-----------------------------+
  |  User Defined Slew Values   |
  |-----------------------------|
  |    Instance  |     Fall     |
  |--------------+--------------|
  |   i_0/i_0    |   0.200000   |
  +-----------------------------+
  ```

- The following command reports the rise and fall slew values for instance `i_0/i_0`:

  ```
  > report_slew_for_power_analysis -instance i_0/i_0
  ```

  Returns the following output:

```
+-----------------------------------------+
|          User Defined Slew Values       |
|-----------------------------------------|
|   Instance   |     Rise     |     Fall    |
|--------------+--------------+-------------|
|   i_0/i_0    |   0.100000   |   0.200000  |
+-----------------------------------------+
```

- The following command reports the rise and fall slew values for all instances in the current module:

  ```
  > report_slew_for_power_analysis
  ```

  Returns the following output:

  ```
  +-----------------------------------------+
  |          User Defined Slew Values       |
  |-----------------------------------------|
  |   Instance   |     Rise     |     Fall    |
  |--------------+--------------+-------------|
  |   i_0/i_0    |   0.100000   |   0.200000  |
  +-----------------------------------------+
  |              |              |             |
  |   i_0/i_1    |   0.300000   |   0.400000  |
  +-----------------------------------------+
  ```

- The following command reports the rise slew value for all instances in the current module:

  ```
  > report_slew_for_power_analysis -rise
  ```

  Returns the following output:

  ```
  +-----------------------------+
  |  User Defined Slew Values   |
  |-----------------------------|
  |    Instance  |     Rise     |
  |--------------+--------------|
  |   i_0/i_0    |   0.100000   |
  |--------------+--------------|
  |   i_0/i_1    |   0.300000   |
  +-----------------------------+
  ```

Related Information

reset_slew_for_power_analysis

set_slew_for_power_analysis

report_tc_stats

```
report_tc_stats [-prob | -td]
     [-type {assert | noassert | all}]
     [-instance instance_name]
     [-range {n1 n2 ...}] [-tcl_list]
```

Reports the number of nets in the various ranges of either signal probability or transition density.

Note: You must read the synthesis library and the netlist before you can use this command.If you did not read a TCF file, LPS uses default toggle counts for each primary input and sequential cell output (specified through the power_default_prob and power_default_toggle_rate globals).

If you want to adjust the format of the table generated with report_tc_stats, specify pa_tc_stats_table as the table name when using the set_table_style command.

Options and Arguments

-instance *instance_name*
: Reports the statistics of a particular instance. There is no need to use this argument if statistics are needed for the whole design.

  Note: You can only report statistics on hierarchical instances.

-prob | -td
: Reports statistics based on signal probability (-prob) or transition density (-td).
  Default: The transition density is reported and a warning is issued to state this unless you specify the -tcl_list argument with this command. See Examples on page 435.

-range {*n1 n2...*}
: Provides the range values for the signal probability or transition density.
  Default: The following values for signal probability and transition density are used to determine the ranges:

```
   -prob:     0  0.1  0.2  0.3  0. 4  0.5  0.6  0.7  0.8  0.9
   -td:       0.0001  0.00005  0.00001  0.00005  0.001 0.005  0.01  0.05  0.1  0.5
```

-tcl_list
: Reports the statistics in Tcl list format.
  Default: The report is shown in tabular format.

`-type {assert | noassert | all}`

Specifies which nets to report. Choose one of the following:

`assert`	Reports nets for which the signal probability or transition density was specified in a TCF file.
`noassert`	Reports nets that have a signal probability or transition density calculated by the software.
`all`	Reports both types of nets, as specified with the `assert` and `noassert` options.This is the default action for this command. If no type is specified, a warning is issued, unless you have specified the `-tcl_list` argument with the command. See the Examples.

Examples

- The following command reports the number of nets for the ranges of transition density in tabular format:

  ```
  > report_tc_stats
  ```

  Returns the following report:

  ```
  --> WARNING: No range of prob/td specified. Using default value. <PA-145>.
  --> WARNING: No type of prob/td specified. Reporting all nets. <PA-146>.
  +---------------------------------+
  |    Transition Density (td)      |
  |      Statistics For Nets        |
  |---------------------------------|
  |        td         |   #nets     |
  |-------------------+-------------|
  | 0.000000-0.000050 |     11      |
  +-------------------+-------------+
  | 0.000050-0.000100 |     2       |
  +-------------------+-------------+
  | 0.000100-0.000500 |     0       |
  +-------------------+-------------+
  | 0.000500-0.001000 |     6       |
  +-------------------+-------------+
  | 0.001000-0.005000 |    137      |
  +-------------------+-------------+
  | 0.005000-0.010000 |     16      |
  +-------------------+-------------+
  | 0.010000-0.050000 |     1       |
  +-------------------+-------------+
  | 0.050000-0.100000 |     0       |
  +-------------------+-------------+
  | 0.100000-0.500000 |     0       |
  +-------------------+-------------+
  |    0.500000->     |     0       |
  +---------------------------------+
  ```

- The following command reports the number of nets for the ranges of transition density in Tcl list format:

  ```
  > report_tc_stats -tcl_list
  ```

  Returns the following report:

  ```
  {0.000000 0.000050 11} {0.000050 0.000100 2} {0.000100 0.000500 0}
  {0.000500 0.001000 6} {0.001000 0.005000 137} {0.005000 0.010000 16}
  {0.010000 0.050000 1} {0.050000 0.100000 0} {0.100000 0.500000 0}
  {0.500000 - 0}
  ```

Related Information

`read_tcf`

reset_slew_for_power_analysis

```
reset_slew_for_power_analysis [-rise] [-fall]
     [-instance {list_of_instances}]
```

Resets the slew times for the power analysis.

Note: You must read the synthesis library and the netlist before you can use this command.

Options and Arguments

`-fall`
: Resets the user-specified slew time of the falling edges.

  If neither `-fall` or `-rise` are specified, both the user-defined slew times are reset.

`-instance {`*`list_of_instances`*
: Resets any user-specified slew times for the specified instances. *Default*: The slew times are reset for all instances in the current module.

`-rise`
: Resets the user-specified slew time of the rising edges.

  If neither `-fall` or `-rise` are specified, both the user-defined slew times are reset.

Examples

Consider the schematic shown in Figure 4-6 on page 424. In the following examples, you can assume that you set rise and fall slew values on the instances `i_0/i_1` and `i_0/i_0`.

- The following commands reset the rise user slew value on instance `i_0/i_0`:

```
> reset_slew_for_power_analysis -rise -instance i_0/i_0
> report_slew_for_power_analysis -instance i_0/i_0
```

  Returns the following report:

```
+-------------------------------------------+
|          User Defined Slew Values         |
|-------------------------------------------|
|   Instance   |     Rise     |     Fall     |
|--------------+--------------+--------------|
|   i_0/i_0    |      -       |   0.200000   |
+-------------------------------------------+
```

- The following commands reset the fall user slew value on instance `i_0/i_1`.

```
> reset_slew_for_power_analysis -fall -instance i_0/i_1
> report_slew_for_power_analysis -instance i_0/i_1
```

Returns the following report:

```
+-------------------------------------------+
|          User Defined Slew Values         |
|-------------------------------------------|
|   Instance   |     Rise     |     Fall     |
|--------------+--------------+--------------|
|   i_0/i_1    |   0.300000   |      -       |
+-------------------------------------------+
```

- The following commands reset the slew values on the current module and on all the instances in the current module.

```
> reset_slew_for_power_analysis
> report_slew_for_power_analysis
```

Returns the following report:

When no user slew values are set, issues the following warning:

```
--> WARNING: No user slew value is set in the current context. No need to
    reset/report it. <PA-241>.
```

Related Information

`report_slew_for_power_analysis`

`set_slew_for_power_analysis`

reset_switching_activity

```
reset_switching_activity
     [-instance {list_of_instances}]
     [-net {list_of_nets}] [-pin {list_of_pins}]
```

Resets the switching activity.

Note: You must read the synthesis library and the netlist before you can use this command.

Options and Arguments

`-instance`

Resets the probability and activity assertions of the pins and nets connected to the specified instance(s). You can specify a list of instance identifiers or a list of hierarchical instance names.

If a hierarchal instance is provided, all the net pins in the hierarchy will be reset.

`-net`

Resets the probability and activity assertions of the specified nets. You can specify a list of net identifiers or a list of hierarchical net names.

`-pin`

Resets the probability and activity assertions of the specified pins. You can specify a list of pin identifiers or a list of hierarchical pin names.

Note: If neither `-instance`, `-pin`, or `-net` is specified, the probability and activity will be reset for all the nets in the current module.

Examples

Consider the schematic shown in Figure 4-6 on page 424.

- The following command resets the probability and transition density of all nets and pins under the instance `i_0/i_1`.

  ```
  > reset_switching_activity -instance i_0/i_1
  ```

- The following command traverses all instances under `i_0` and resets the probability and transition density of all the nets and pins.

  ```
  > reset_switching_activity -instance i_0
  ```

- The following command resets the probability and transition density for the specified pins and nets:

```
> reset_switching_activity -pin {i_0/i_1/A i_0/i_0/B} \
> -net {i_0/i_0/in1[0]  i_0/i_1/n_23}
```

- The following command resets the probability and transition density of all nets and pins under instance `i_0/i_1`and for the specified nets and pins:

```
> reset_switching_activity -instance i_0/i_1 \
> -net i_0/out[0] -pin i_0/A
```

Related Information

`read_tcf`

`set_switching_activity`

set_clock_gating_options

```
set_clock_gating_options [-default]
     [-auto_test_port {true | false}]
     [-control {none | pre_seq_element |post_seq_element}]
     [-control_mode {use_scan_enable | use_test_mode}]
     [-control_port signal_name]
     [-domain {all | dft_domain | clock_net}]
     [-no_timing {true|false}] [-observe {true|false}]
     [-obs_style {port | register | reg_module}]
     [-xor_depth depth] [-same_polarity {true|false}]
     [-exclude {list_of_cells}
     [-force {list_of_instances}]
     [-ignore {list_of_instances}]
     [-remove_all {true|false}]
     [-gating_style {none | latch| flipflop}]
     -reset {true | false} [-max_fanout size]
     [-minsize reg_bank_size] [-rg_dissolve {true|false}]
     [-rg_force] [-rg_ignore] [-rg_max number]
```

Sets various clock-gating functions for clock-gating insertion and for clock-gating logic commitment during gate-level power optimization. This command does not overwrite any options previously set by this command.

Note: You must read the synthesis library and the netlist before you can use this command.

Important

Options are preserved even after you run the `do_remove_design -all` command, which removes the design.

LPS can automatically map to the dedicated gating cells in the library if they have either the attribute `is_clock_gating_cell` (set to `true`) or `clock_gating_integrated_cell`.

Support for Multiple Clock Domains

To support multiple clock domains controlled by different test signals, you must use the `set_test_mode_setup` command to associate the clock with the corresponding test-mode pin. See the set_test_mode_setup on page 783 command for more information.

For more information about multiple clock domains, see *Low Power for BuildGates Synthesis and Cadence PKS*.

Options and Arguments

`-auto_test_port {true|false}`

Inserts test-mode pins automatically at the top module for all the clock domains when inserting DFT logic during clock gating. This is only done if the clock domains with DFT settings do not have a corresponding test-mode pin defined. The name of the added test-mode pins are composed of the value of the global environment variable `dft_test_mode_port_name_prefix`, followed by a numeric. If this global is set to the default, `TEST_MODE`, then the first test_mode pin added is called `TEST_MODE_1`.
Default: `true`

You can only set this option if you set `-control` to a value other than `none`.

`-control {none | pre_seq_element | post_seq_element}`

Controls the addition of controllability logic to the gated clocks.

`none` (default)	Does not add any controllability logic.
`pre_seq_element`	Adds the controllability logic prior to the sequential element.
`post_seq_element`	Adds the controllability logic after the sequential element.

If the `gating_style` is `none`, you can set either of the `pre_seq_element` or `post_seq_element` options. They both have the same effect.

If you set `-domain` to either `dft_domain` or `clock_net`, the clock is made controllable (controllability logic is inserted) by associating the test-mode pin with the corresponding clock as defined by the `set_test_mode_setup` command.

Because of the gated logic, it might not be possible to control the clock signal feeding the set of registers affecting clock gating, which can affect design testability. Using this argument gives you that control.

`-control_mode {use_scan_enable | use_test_mode}`

Species the type of signal to use to make the gated clock controllable.

You can only set this option if `-control` is currently set to a value other than `none`.

`use_scan_enable`	Uses the scan enable of the registers driven by the corresponding gated clock. All the registers that are driven by the gated clock must be driven by the same scan enable signal.
`use_test_mode` (default)	Uses the test mode signal. The signal used depends on the settings of the `-control_port`, `-domain` and `-auto_test_mode` options.

`-control_port` *`signal_name`*

Specifies the name of the signal to use to control the gated clock for all clock domains. The signal can be either a port or an internal pin. You must specify the complete path from the current module. You should also specify the signal as a test_mode signal using the `set_test_mode_setup` command.

You can only set this option if `-control` is currently set to a value other than `none` and `-domain` is set to `all`.

`-default`

Sets all options to the default values and removes any attributes set by `-force` and `-ignore`. See Examples for the software default values.

`-domain {all|dft_domain|clock_net}`

Specifies how to group gating signals for controllability, observability, or both. Choose one of the following:

`all` (default)	Groups all gating signals in the same observability logic. This is the default when you also set the `-observe` option. Makes all clocks controllable by a single test-mode pin, if you also have `-control_mode` set to `use_scan_enable`. You can only set this value if you set `-control` to `pre_seq_element` or `post_seq_element`.
`dft_domain`	Groups all gating signals of clock domains that are associated with the same test-mode signal together in the same observability logic, if you also have `-observe` set for this command. Makes all clocks associated with a given test mode pin controllable by this test mode pin if the association between the clocks and test-mode pin is done through the `set_test_mode` command.
`clock_net`	Groups all gating signals driven by the same clock signal under DFT settings. This option is only relevant for observable domains. The controllability is governed by the association in the `set_test_mode_setup.`

This option is only valid if you `-control_mode` is set to `use_test_mode.`

For a detailed description of the three different ways the tool can trace the clock net back through the design, see *Low Power for BuildGates Synthesis and Cadence PKS*.

`-exclude {`*`list_of_cells`*`}`

Prevents the specified library cells from being used for clock gating logic. You must specify the cells using their identifiers (IDs).

`-force {`*`list_of_instances`*`}`

Applies to clock gating only.

Commits clock-gating logic for the specified instances. You must specify the instances using their identifiers (IDs) or their hierarchical instance name.

If you set this option on the flip-flops before inserting clock-gating logic, LPS will force insertion of clock-gating logic for these flip-flops.

If you set this option on the flip-flops after inserting clock-gating logic, LPS will blindly commit insertion of clock-gating logic for these flip-flops without checking timing or power constraints.

To force commitment of any gating logic, you should set this option after `do_optimize -stop_for_power_sim -power` or `do_optimize -stop_after mapping -power.`

You can set this option on any flip-flop and it commits the clock-gating domain the flip-flop belongs to.

`-gating_style {none | latch | flipflop}`

Controls whether to add any logic to prevent glitches on the enable signal. Choose one of the following options:

`none`	Does not add any logic. The clock-gating logic does not contain any latches nor flip-flops. If you specify `none`, you need to source the timing constraints before inserting clock gates.
`latch` (default)	Uses a latch to prevent glitches on the clock-gating enable signal.
`flipflop`	Uses a flip-flop to prevent glitches on the clock-gating enable signal.

`-ignore {`*`list_of_instances`*`}`

Does not insert or commit clock-gating logic for the specified instances. You must specify the instances using their identifiers (IDs).

If you specify `set_clock_gating_options -ignore` prior to inserting the clock-gating logic, the clock-gating logic for the instances specified is not inserted.

If you specify `set_clock_gating_options -ignore` after inserting the clock-gating logic, clock-gating logic for the specified instances is not committed.

You can set this option on any flip-flop and it decommits the clock-gating domain the flip-flop belongs to.

`-max_fanout` *`number`*

Clones the clock-gating cell whenever this cell is driving more registers than the specified number.

LPS clones only the clock-gating cell and not the entire clock-gating logic (latch, controlling logic, and so on).

`-minsize` *`reg_bank_size`*

Enables clock-gating insertion for any register bank larger than the specified size.
Default: The register bank is clock gated based on the estimated power savings it can produce.

For optimal results, let the tool decide the minimum register size based on the power estimation.

`-no_timing {true|false}`

Ignores timing when inserting clock-gating logic. Even if the clock-gating logic causes a timing violation, the logic will be committed.
Default: `true`

`-observe {true|false}`

Makes all gated signals observable. Each gating signal is added to an XOR tree and the output of the XOR tree is connected to either a port or observability register.
Default: `false`.

Note that the XOR tree can span through several modules. All observability signals in a module corresponding to a clock domain under DFT settings are put into an XOR tree in that

module and a single observability signal crosses the module boundary, when necessary.

Clock gating makes the gating signals unobservable. When you set this option, you enable the addition of extra logic to make the signals observable again.

You can only set this option if `-control` is currently set to a value other than `none`.

This option may cause module changes because a new port may be added for the observable signal.

The `-domain` and `-obs_style` options control the set of observability points to be inserted in a single XOR tree.

`-obs_style {port|register|reg_module}`

Controls how to create the observability domain. Choose one of the following:

`port` (default)	Creates a port in the top module for each observability domain and its logic is connected to this port through an XOR tree.
`register`	Connects the observability logic to an observability register that is inserted in the scan chain. This is limited by the `-xor_depth` value. *Default*: The register is clocked with the inverted signal of the physical clock net driving the registers in that observability domain. The register is inserted at the outermost hierarchy that has the corresponding clock signal. The `-same_polarity` option changes this default. The logic can span across modules except in modules where the clock signal is not present. If you select this option, you must also set `-obs_domain clock_net`.

`reg_module` Completes the observability logic for an observability domain in the register in that module. This avoids the observability port creation in any hierarchical module.

If you select this option, you must also set `-domain clock_net`.

Note: You can only set the `-obs_style` option if you set `-observe`.

`-remove_all {true|false}`
Removes all inserted clock-gating logic.
Default: `true`

`-reset {true|false}`
Automatically adds reset logic to the glitch removing logic.
Default: `false`

When you set this option to `true`, only those clock-gating integrated cells in the library that have a reset pin are used. When you set this option to `false`, only those clock-gating integrated cells in the library without a reset pin are used.

If none of the registers in the clock-gating domain has an asynchronous reset or preset signal, no reset is added to the glitch removal logic.

If c1,c2, ..., cn represent the asynchronous reset signals for each of the registers in the clock-gating domain, and p1,p2,...,pn represent the asynchronous preset signals for each of the registers in the clock gating domain, LPS determines the reset signal for the glitch removal latch/FF as:

```
(c1+p1) AND (c2+p2) AND ... AND  (cn+pn).
```

or

If each register in the clock-gating domain has an asynchronous reset signal c*x* and an asynchronous preset signal p*x*, LPS determines the reset signal for the glitch removal latch/FF as:

```
(c1+p1) AND (c2+p2) AND ... AND  (cn+pn).
```

`-rg_dissolve {true|false}`
Dissolves the hierarchical module created during root gating.
Default: `true`

`-rg_force {`*`list_of_instances`*`}`
Automatically considers the specified list of sequential elements and clock-gating instances for root gating, without considering cost.

Note: You cannot force the tool to consider the clock gating latch for root gating.

`-rg_ignore {`*`list_of_instances`*`}`
Ignores the specified list of sequential elements and clock-gating instances for root gating. If you specify any root-gating logic instance in the list, the instance will be decommitted after you invoke the `do_xform_optimize_clock_gate -root` command.

`-rg_max` *`number`*
Specifies the maximum number of enable signals for every root-gating logic to be inserted. Specify a positive integer.
Default: 10. If you specify the `-rg_force` option at the same time, the `-rg_max` option will be ignored.

`-same_polarity {true|false}`
Sets the clock polarity for the observability register to that of the driving registers in the clock-gating domain.
Default: The clock polarity of the observability register is opposite to that of the driving registers of the clock-gating domain. This default provides an additional half cycle to allow the signal to propagate through the XOR tree.

You can only set this option if you set `-obs_style` to `register` or `reg_module`.

`-xor_depth` *`depth`*
Adds a register whenever the depth of the XOR tree is equal to the specified depth. This ensures that the maximum depth of the XOR tree is limited to the value specified here.
Default: 5

You can only set this option if you set `-obs_style` to `register` or `reg_module`.

Examples

- The following command prevents some OAI cells from being used for clock-gating logic:

```
> set_clock_gating_options [-exclude find -cellrefs {OAI_2 OAI_3}]
```

- The following command sets all options to the default values and removes any attributes set by `-force` and `-ignore:`

```
> set_clock_gating_options -default
> get_clock_gating_options
```

Returns the default values:

```
+---------------------------------+
|      Clock-Gating Options       |
|---------------------------------|
|     Options     |     Value     |
|-----------------+---------------|
| -auto_test_port |     false     |
|    -control     |     none      |
|  -control_mode  | use_test_mode |
|  -control_port  |    NOT_SET    |
|     -domain     |      all      |
|     -force      |    NOT SET    |
|  -gating_style  |     latch     |
|     -ignore     |    NOT SET    |
|   -max_fanout   |    NOT SET    |
|    -minsize     |       3       |
|   -no_timing    |     false     |
|    -observe     |     false     |
|   -obs_style    |     port      |
|   -remove_all   |     false     |
|  -rg_dissolve   |     false     |
|    -rg_force    |    NOT SET    |
|   -rg_ignore    |    NOT SET    |
|     -rg_max     |      10       |
|  -same_polarity |     false     |
|   -xor_depth    |       5       |
+---------------------------------+
```

- The following commands limit the number of registers driven by a single clock gating cell to 2, indicates to add the controllability logic before the sequential element and to make all gating signals observable, commits clock gating logic for instance `121203,` and decommits the lock gating logic for instance `127987`:

```
> set_clock_gating_options -max_fanout 2 -control pre_seq_element \
> -observe -force [find -hier -register reg_1]
> -ignore [find -hier -register reg_3]
> get_clock_gating_options
```

Returns the settings in tabular format:

```
+-----------------------------------+
|       Clock-Gating Options        |
|-----------------------------------|
|     Options     |      Value      |
|-----------------+-----------------|
| -auto_test_port |      false      |
|    -control     | pre_seq_element |
|  -control_port  |     NOT_SET     |
|  -control_mode  |  use_test_mode  |
|     -domain     |       all       |
|     -force      |     121203      |
|  -gating_style  |      none       |
|     -ignore     |     127987      |
|   -max_fanout   |        2        |
|    -minsize     |        3        |
|   -no_timing    |      false      |
|    -observe     |      true       |
|   -obs_style    |      port       |
|   -remove_all   |      false      |
|  -rg_dissolve   |      false      |
|    -rg_force    |     NOT SET     |
|   -rg_ignore    |     NOT SET     |
|     -rg_max     |       10        |
|  -same_polarity |      false      |
|   -xor_depth    |        5        |
+-----------------------------------+
```

- The following commands indicate in addition to the previous example not to add any logic to prevent glitches on the enable signal:

```
> set_clock_gating_options -max_fanout 2 -control pre_seq_element \
> -observe -gating_style none -force {121203} -ignore {127987}
> get_clock_gating_options -tcl_list
```

  Returns the settings in Tcl list format:

```
{-auto_test_port false}  {-control pre_seq_element} {-control_mode
use_test_mode}  {-control_port NOT_SET} { -force {121203 } } {-gating_style
latch} { -ignore {127987} } {-max_fanout 2} {-minsize 3} {-no_timing false}
{-observe true} {-obs_style port} {-remove_all false} {-rg_dissolve false}
{ -rg_force { } } { -rg_ignore { } } {-rg_max 10} {-same_polarity false}
{-xor_depth 5}
```

- The following commands control different clocks with different test mode pins, group all gating signals associated with test pin `tp1` in the same controllability logic, and do the same for all gating signals associated with test pin `tp2`:

```
> set_test_mode_setup tp1 1 -clock clk1
> set_test_mode_setup tp2 1 -clock {clk2 clk3}
> set_clock_gating_options -control pre_seq_element -domain dft_domain
```

- The following command creates an observability tree with a depth no greater than 10, observed by register per module:

```
> set_clock_gating_options -control post_seq_element -domain clock_net
-auto_test_port -observe true -obs_style reg_module -xor_depth 10
```

- The following commands decommit all clock-gating domains except for `i_5_LPS_CG_GATING_6`:

```
> set_clock_gating_options -ignore [find -hier -inst *LPS_CG_GATING*]
> set_clock_gating_options -force [find -hier -inst i_5_LPS_CG_GATING_6]
```

- The following command uses the scan enable of the registers to make the gated clock controllable:

```
set_scan_mode SE 1
check_dft_rules
set_clock_gating_options -control_mode use_scan_enable
do_optimize -power
```

Related Information

`do_xform_optimize_clock_gate`

`get_clock_gating_options`

set_power_display_unit

`set_power_display_unit` *`power_unit`*

Causes any power reports to be displayed in the units specified.

Options and Arguments

`power_unit`

Specifies the power unit. You can specify any of the following units: kW, W, mW, uW, nW, pW, fW, and aW. The power units are case insensitive.
Default: mW

Example

The following command sets the unit to be used in the power reports to mW:

```
> set_power_display_unit nW
```

Related Information

`get_power_display_unit`

set_power_optimization_options

```
set_power_optimization_options
     [-instance_leakage_power leakage_factor]
     [-instance_internal_power internal_factor]
     [-net_power net_factor] [-default]
     [-power_constraint target_power] [-default]
```

Sets the weight factors for each component in the total power calculation during optimization only.

During power optimization, LPS calculates the power as:

$$total\_power = total\_instance\_power + net\_factor \times total\_net\_power$$

where

$$instance\_power = leakage\_factor \times instance\_leakage\_power + internal\_factor \times instance\_internal\_power$$

Default: All power components are given an equal weight factor of 1. The weight factors instruct LPS to optimize a different aspect of the total power. For more information about power concepts and how LPS calculates the power, refer to Power Analysis Concepts in *Low Power for BuildGates Synthesis and Cadence PKS*.

Note: This command does not change the results obtained with `get_power` and `report_power`. It only changes the power cost function during optimization.

Options and Arguments

`-default`
: Sets all weight factors back to the default values.

`-instance_internal_power` *`internal_factor`*
: Specifies the weight of the instance internal power in the power calculation. Specify a floating number between 0 and 1. When not specified, this weight factor is set to 0.0.

`-instance_leakage_power` *`leakage_factor`*
: Specifies the weight of the instance leakage power in the power calculation. Specify a floating number between 0 and 1. When not specified, this weight factor is set to 0.0.

`-net_power` *`net_factor`*

Specifies the weight of the net power in the power calculation. Specify a floating number between 0 and 1. When not specified, this weight factor is set to 0.0.

`-power_constraint` *`target_power`*

Specifies the power optimization target. Specify the target power in the units specified with `get_power_display_unit`. LPS interprets the target power differently depending on the setting of the other command options (see Examples below for more details). If not specified, the target power defaults to `-1`, which allows the power optimizer to return the best result possible.

Note: Setting this constraint is no guarantee of meeting that constraint. Setting this constraint only ensures that the optimization stops when the constraint is met. Also, if the optimizer meets the constraint, LPS resets the constraint so that the power optimizer continues to optimize the design when you issue more power optimization commands to lower the power even more. If the user target is too optimistic and LPS stops before meeting the target, you should try power optimization commands by increasing the effort level.

Examples

- The following command optimizes leakage power only:

```
> set_power_optimization_options -instance_leakage_power 1
```

  The weight factors for the instance internal power and net power are set to 0.

- The following command optimizes dynamic switching power only:

```
> set_power_optimization_options -instance_internal_power 1 -net_power 1
```

  The weight factor for the instance leakage power is set to 0.

- The following command optimizes all power components, but skews the optimization towards leakage power optimization:

```
> set_power_optimization_options -instance_leakage_power 0.8
  -instance_internal_power 0.2 -net_power 0.2
```

- The following command specifies that the target power is the target for the total power.

```
> set_power_optimization_options -instance_internal_power 0.5 \
> -instance_leakage_power 0.9 -net_power 1.0 -power_constraint 123.1960
```

- The following command specifies that the target power is the target for the leakage power.

```
> set_power_optimization_options -instance_internal_power 0 \
> -instance_leakage_power 1 -net_power 0 -power_constraint 23.19
```

Related Information

`get_power_optimization_options`

Power Analysis Concepts in *Low Power for BuildGates Synthesis and Cadence PKS*

set_sleep_mode_options

```
set_sleep_mode_options
     [-timing_driven {none | partial | full}]
     [-no_dissolve {true|false}]
     [-remove_all {list_of_instances}] [-default]
     [-ignore {list_of_instances}]
     [-force {list_of_instances}]
```

Lets you choose settings for the sleep-mode logic commitment or decommitment. You can set these options any time before you commit or decommit the sleep-mode logic.

Note: You must read the synthesis library and the netlist, and run `do_build_generic -sleep_mode` before you can use this command.

Options and Arguments

`-default`
: Sets all options for this command to the default settings.

`-force {list_of_instances}`
: Specifies a list of sleep-mode candidates that will be committed at the time of the sleep-mode commitment phase. Specify a list of instance identifiers.
Default: All sleep-mode logic is committed.

`-ignore {list_of_instances}`
: Removes the inserted sleep-mode logic for the specified instances only if it improves timing by doing so. If no instances are specified all inserted sleep-mode logic become candidates.

`-no_dissolve {true|false}`
: Does not dissolve the hierarchy of a gating module when it is committed during sleep mode.
Default: `true`

`-remove_all {list_of_instances}`
: Blindly removes the inserted sleep-mode logic for the specified instances. If no instances are specified, all inserted sleep-mode logic is removed.

`-timing_driven {none | partial | full}`
Determines to what degree timing is considered when committing sleep-mode logic. Choose one of the following:

`none`	Does not consider timing when committing the sleep-mode gating logic. Only power is considered.
`partial`	Commits all modules based on power, but then runs timing optimization to improve slack. If the timing constraints are not met, the command tries to improve the slack by decommitting gated modules.
`full` (default)	Commits each gating module if committing it does not violate a timing constraint.

Example

The following commands set all options to the default settings:

```
> set_sleep_mode_options -default
> get_sleep_mode_options
```

Reports the following settings:

```
+----------------------------+
|     Sleep Mode Options     |
|----------------------------|
|    Options     |   Value   |
|----------------+-----------|
|  -no_dissolve  |    true   |
| -timing_driven |    full   |
+----------------------------+
```

Related Information

`get_sleep_mode_options`

set_slew_for_power_analysis

```
set_slew_for_power_analysis
     {-rise slew | -fall slew | -rise slew -fall slew
     | user_slew} [-instance {list_of_instances}]
```

Sets the slew times for the specified instances to be used during power analysis.

While estimating the power, the tool first tries to get the user-defined slew for an instance. If no user-defined slew is available, the tool uses the slew calculated by the timing analyzer. The slew is calculated based on the value of the `power slewmode for power analysis` global. This global takes the values `worst`, `best`, `avg.`

Options and Arguments

`-fall` *`slew`*
: Specifies the value of the fall slew time.

`-instance {`*`list_of_instances`*`}`
: Specifies the instances to which the specified slew times apply during power analysis. If any instance is hierarchical, the user-defined slew is used during power analysis for all primitive instances inside that hierarchy.
Default: The slew values apply to all instances of the current module.

`-rise` *`slew`*
: Specifies the value of the rise slew time

`user_slew`
: Specifies both the rise and fall slew times.

Examples

For the following examples, see the schematic shown in Figure 4-6 on page 424.

- The following command sets the rise slew value to `0.1` and the fall slew value to `0.2` on instance `i_0/i_0`:

  ```
  > set_slew_for_power_analysis -rise 0.1 -fall 0.2 -instance i_0/i_0
  ```

- The following command sets the rise slew value to `0.1` and the fall slew value to `0.2` on instance `i_0` and all the instances below `i_0`:

  ```
  > set_slew_for_power_analysis -rise 0.1 -fall 0.2 -instance i_0
  ```

- The following command sets the rise and fall slew values to `0.2` on instance `i_0` and all the instances below `i_0`:

  ```
  > set_slew_for_power_analysis 0.2 -instance i_0
  ```

- The following command sets the rise slew value to `0.2` on instance `i_0/i_0`:

  ```
  > set_slew_for_power_analysis -rise 0.2 -instance i_0/i_0
  ```

Related Information

`report_slew_for_power_analysis`

`reset_slew_for_power_analysis`

set_switching_activity

```
set_switching_activity [-prob probability]
     [-td transition_density]
     -net {list_of_nets} -pin {list_of_pins}
```

Sets the switching activity (probability and toggle count).

You must read the synthesis library and the netlist before you can use this command.

Note: The `set_switching_activity` command is not meant to set the switching activity of each net one-by-one. Rather, it is meant to set the switching activity of some of the nets that are critical to your design.

Options and Arguments

`-net {`*`list_of_nets`*`}`
: Specifies the nets on which the probability and activity assertions are set. You can specify a list of the net identifiers or hierarchical net names.

`-pin {`*`list_of_pins`*`}`
: Specifies the pins on which the probability and activity assertions are set. You can specify a list of the pin identifiers or hierarchical pin names.

`-prob` *`probability`*
: Specifies the probability value. Specify a value between 0 and 1.
Default: Corresponds to the value of the `power_default_prob` global variable.

`-td` *`transition_density`*
: Specifies the transition density. Specify a positive floating number larger than zero.
Default: Corresponds to the value of the `power_default_toggle_rate` global variable.

Note: You can specify a list of nets, pins or a combination. If no option is given an error message is displayed.

Examples

Consider the schematic shown in Figure 4-6 on page 424.

- The following command sets the probability to `0.5` and the transition density to `0.0006` on the specified nets and pins:

```
> set_switching_activity -prob 0.5 -td 0.0006 -net {in1[0] in2[0]}|
> -pin {i_0/i_1/A i_0/i_1/B}
```

- The following command sets the probability to `0.5` and the transition density to `0.0006` on the specified nets:

```
> set_switching_activity -prob 0.5 -td 0.0006 -net {in1[0] in2[0]}
```

- The following command sets the probability to `0.5` and the transition density to `0.0006` on the specified pins:

```
> set_switching_activity -prob 0.5 -td 0.0006 -pin {i_0/i_1/A i_0/i_1/B}
```

Related Information

`read_tcf`

`read_tcf_update`

`reset_switching_activity`

Setting Switching Activity Takes a Long Time in *Low Power for BuildGates Synthesis and Cadence PKS*

write_clock_gating_attribute

```
write_clock_gating_attribute [-file filename]
```

Dumps all the attributes which are needed to perform clock-gating transformations in a Tcl script. You must explore the clock gating and read the clock constraints before you can use this command.

By sourcing the Tcl script generated by this command you can commit or decommit clock-gating logic at any stage in the flow starting from the updated Verilog netlist.

Important

Do not edit this file or LPS cannot guarantee its functionality.

Options and Arguments

`-file filename`

Specifies the name of the file in which to dump the attributes.
Default: `clockGatingAttributes.tcl`

Example

The following example shows where you use this command in the flow:

```
read_verilog design.v
do_build_generic

# set timing constraints (including clock constraints)

# Clock gating exploration
do_optimize -stop_for_power_simulation -power

write_verilog -hier my_design.v
write_clock_gating_attribute -file cgattr.tcl

# at this point you can run an external (Cadence or third-party) tool
do_remove_design -all

# read the updated netlist
read_verilog my_design_updated.v

do_build_generic

# set timing constraints (including clock constraints)

# read attributes associated with clock-gating modules
source cgattr.tcl
```

After reading the attributes, all clock-gating transformations can be performed.

Related Information

`check cg logic`

write_psf

`write_psf` *`file_name`*

Writes the level 1 power specification file (PSF). This power information model is used for defining currents and maximum voltage drop limits.

The PSF file contains the power and ground current demands (in amperes) for each of the flat instances. The PSF file can also be used for IR drop analysis of power grid.

From the current module the command will traverse all the flat instances and reports the average current drawn from the supply.

Note: You must read the synthesis library and the netlist before you can use this command. If you did not read a TCF file, LPS uses default toggle counts for each primary input and sequential cell output (specified through the `power_default_prob` and `power_default_toggle_rate` globals).

Options and Arguments

`-file` *`filename`*

Specifies the name of the file to which to write the power information.

Example

The following command writes out the power information in PSF form to the `psf.out` file:

```
> write_psf psf.out
```

The following is an example of a PSF file:

```
INST_PWR_L1 {
     NAME: "i_1";
     VDD_CURRENT: 1e-8;
     GND_CURRENT: 1e-8;
     }
```

write_sleep_mode_attribute

```
write_sleep_mode_attribute [-file filename]
```

Dumps all the attributes which are needed to perform sleep-mode transformations in a Tcl script. You must explore sleep mode before you can use this command.

By sourcing the Tcl script generated by this command you can commit or decommit sleep-mode logic at any stage in the flow starting from the updated Verilog netlist.

Do not edit this file or LPS cannot guarantee its functionality.

Options and Arguments

`-file filename`

Specifies the name of the file in which to dump the attributes.
Default: `sleepModeAttributes.tcl`

Example

The following example shows where you use this command in the flow:

```
read_verilog design.v

# sleep-mode exploration & insertion
do_build_generic -sleep_mode

write_verilog -hier my_design.v
write_sleep_mode_attribute -file smattr.tcl

# at this point you can run an external (Cadence or third-party) tool
do_remove_design -all

# read the updated netlist
read_verilog my_design_updated.v

# sleep-mode modules already inserted, so no -sleep_mode
do_build_generic

# read attributes associated with sleep mode modules
source smattr.tcl
```

Related Information

`do_build_generic -sleepmode`

write_tcf

```
write_tcf [-hier | -flat]
     [-instance {list_of instances}]
     -type [-assert | noassert | all]
     [-duration simulation_period] tcf_file
```

Writes the probability and toggle count of the pins and nets in the specified instances. It also includes the switching activity of all output ports of the modules.

Note: You must read the synthesis library and the netlist before you can use this command.

Options and Arguments

`-assert | noassert | all`
Prints either asserted, non-asserted, or both (based on the option) signal probability and transition count of the pins and nets.
Default: `all`

`-duration` *`simulation_period`*
Multiplies the transition density (td), so that the final toggle count is an integer. For example, if td is 3e-3/ns and the simulation period is 1e5 ns, the printed toggle count will be 300.
Default: 1e5 ns

`-hier | -flat`
Specifies the format of the TCF file.
Default: `-hier`

`-instance {`*`list_of_instances`*`}`
Specifies the instances for which the probability and toggle count data is written. You can specify a list of the instance identifiers or hierarchical instance names.
Default: Current module

`tcf_file`
Specifies the name of the file in which to store the TCF data.

Examples

Consider the schematic shown in Figure 4-6 on page 424. Assume that the following TCF file (`example.tcf`) has been read in.

```
tcffile () {
   tcfversion    :       "1.0";
   generator     :       "BGPower Verilog PLI";
   genversion    :       "1.0";
   date          :       "Thu Feb 21 15:19:52 2002";
   duration      :       "1.201000e+05";
   unit          :       "ns";
   instance () {
     pin () {
        "i_0/i_0/S"     :        "0.518989  1020";
        "i_0/i_1/Y"     :        "0.499933  1510";
        "i_0/i_0/A"     :        "0.506495  527";
        "i_0/i_1/B"     :        "0.502498  511";
        "i_0/i_0/B"     :        "0.499417  493";
        "i_0/i_1/A"     :        "0.493589  488";
        "i_0/i_1/C"     :        "0.243457  511";
     }
   }
}
```

- The following command writes only the asserted probability and transition count for the nets of instance `i_0/i_0`:

  ```
  > write_tcf -flat -type assert -instance i_0/i_0 tcf_file
  ```

  This results in the following TCF file:

  ```
  tcffile () {
     tcfversion    :       "1.0";
     generator     :       "BGPower LPS";
     genversion    :       "1.0";
     date          :       "Thu Feb 21 15:23:28 2002";
     duration      :       "1.201000e+05";
     unit          :       "ns";
     instance () {
       net () {
          "i_0/out[0]"    :        "0.518989  1020";
          "i_0/n_23"      :        "0.243457  511";
          "i_0/in1[0]"    :        "0.506495  527";
          "i_0/in2[0]"    :        "0.499417  493";
       }
     }
  }
  ```

- The following command writes only the nonasserted probability and transition count for the nets of instance `i_0/i_0`. Because all nets are user asserted, the net section of the new TCF file (below) is empty.

  ```
  > write_tcf -flat -type noassert -instance i_0/i_0 tcf_file
  ```

  This results in the following TCF file:

  ```
  tcffile () {
     tcfversion    :       "1.0";
     generator     :       "BGPower LPS";
     genversion    :       "1.0";
     date          :       "Thu Feb 21 15:26:21 2002";
     duration      :       "1.201000e+05";
     unit          :       "ns";
     instance () {
  ```

```
      net () {
      }
   }
}
```

- The following commands first resets the probability and transition count of net `i_0/n_23`. LPS then calculates the probability and transition count of the net i_0/n_23. Next, LPS writes the nonasserted probability and transition count for the nets of instance `i_0/i_0`. Because the probability and transition count of net `i_0/n_23` was reset, net `i_0/n_23` now appears in the net section.

  ```
  > reset_switching_activity -net i_0/n_23
  > write_tcf -flat -type noassert -instance i_0/i_0 tcf_file
  ```

  This results in the following TCF file:

  ```
  tcffile () {
     tcfversion    :       "1.0";
     generator     :       "BGPower LPS";
     genversion    :       "1.0";
     date          :       "Thu Feb 21 15:28:40 2002";
     duration      :       "1.201000e+05";
     unit          :       "ns";
     instance () {
       net () {
          "i_0/n_23":        "0.252952  488";
       }
     }
  }
  ```

- The following command writes only the probability and transition count for the nets of instance `i_0/i_0` that were asserted. Net `i_0/n_23` is not written out as it is no longer user asserted.

  ```
  > write_tcf -flat -type assert  -instance i_0/i_0 tcf_file
  ```

  This results in the following TCF file:

  ```
  tcffile () {
     tcfversion    :       "1.0";
     generator     :       "BGPower LPS";
     genversion    :       "1.0";
     date          :       "Thu Feb 21 15:30:20 2002";
     duration      :       "1.201000e+05";
     unit          :       "ns";
     instance () {
       net () {
          "i_0/out[0]"    :        "0.518989  1020";
          "i_0/in1[0]"    :        "0.506495  527";
          "i_0/in2[0]"    :        "0.499417  493";
       }
     }
  }
  ```

- The following command writes both the user asserted and tool calculated values for instance `i_0/i_0` and all nets below it:

  ```
  > write_tcf -hier -instance i_0/i_0 tcf_file
  ```

This results in the following TCF file:

```
tcffile () {
   tcfversion    :       "1.0";
   generator     :       "BGPower LPS";
   genversion    :       "1.0";
   date          :       "Thu Feb 21 15:36:18 2002";
   duration      :       "1.201000e+05";
   unit          :       "ns";
   instance() {
     instance("i_0") {
       net () {
         "out[0]"       :         "0.518989  1020";
         "n_23"         :         "0.252952  488";
         "in1[0]"       :         "0.506495  527";
         "in2[0]"       :         "0.499417  493";
       }
     }
   }
}
```

- The following command writes the user asserted and tool calculated values for instances `i_0/i_0` and `i_0/i_1` and all nets below them:

```
> write_tcf -hier -instance {i_0/i_0 i_0/i_1} tcf_file
```

This results in the following TCF file:

```
tcffile () {
   tcfversion    :       "1.0";
   generator     :       "BGPower LPS";
   genversion    :       "1.0";
   date          :       "Thu Feb 21 15:37:26 2002";
   duration      :       "1.201000e+05";
   unit          :       "ns";
   instance() {
     instance("i_0") {
       net () {
         "out[0]"       :         "0.518989  1020";
         "n_23"         :         "0.252952  488";
         "in1[0]"       :         "0.506495  527";
         "in2[0]"       :         "0.499417  493";
         "out[1]"       :         "0.499933  1510";
         "in2[1]"       :         "0.493589  488";
         "in1[1]"       :         "0.502498  511";
       }
     }
   }
}
```

- The following command writes out the default TCF format which is hierarchical. Because no instance is specified, all the nets under the current module are reported:

```
> write_tcf tcf_file
```

This results in the following TCF file:

```
tcffile () {
   tcfversion    :       "1.0";
   generator     :       "BGPower LPS";
   genversion    :       "1.0";
```

```
    date          :       "Thu Feb 21 15:38:36 2002";
    duration      :       "1.201000e+05";
    unit          :       "ns";
    instance() {
      instance("i_0") {
        net () {
          "out[0]"       :         "0.518989  1020";
          "n_23"         :         "0.252952  488";
          "in1[0]"       :         "0.506495  527";
          "in2[0]"       :         "0.499417  493";
          "out[1]"       :         "0.499933  1510";
          "in2[1]"       :         "0.493589  488";
          "in1[1]"       :         "0.502498  511";
        }
      }
    }
}
```

Related Information

read_tcf

read_tcf_update

Low Power for Existing BuildGates Synthesis Commands

Some low power functions are built into existing BuildGates Synthesis commands. Please see the appropriate command documentation for more information about the low power functionality.

- `check_library -power`
- `do_build_clock_tree -power`
- `do_build_generic -sleep_mode`
- `do_optimize -power`
- `do_xform_optimize_slack -power`
- `report_cell_instance -power`

5

PKS Commands

This chapter provides a listing of the Cadence® physically knowledgeable synthesis (PKS) commands available with the PKS tool.

This chapter describes the following PKS commands:

add_physical_connection

```
add_physical_connection -net net_name_or_id [-cluster_pin cluster_pin_list]
     [-connectivity_rule list_of_connectivity_rules]
     [-physical_pin physical_pin_list] [-type type]
```

Connects true physical pins. If any pin is not found, no connection is made. In other words, this command does not create pins. It is better to define the pins on subclusters first before defining at the parent clusters. No connection is made to a logical pin (one that already exists in the logical netlist). This is done using logical level commands such as `add_netconn`.

Options and Arguments

`-cluster_pin` *`list_of_cpinnames`*
: Defines the list of cluster pin names in a current cluster or a direct soft-block sub-cluster.

`-connectivity_rule` *`list_of_connectivity_rules`*
: Specifies a list of instance/pin combinations that need to be connected to this physical net. The syntax is a list of strings separated by white space. Each string is of the form "*`INSTNAME PINNAME`*" with a space between the names, or of the form "`*` *`PINNAME`*". The only wildcard supported is "*". No partial wildcards (for instance, `{dir1/hier/* pin1}`) are allowed. Connectivity rules are not inherited by *push* and *pull* operations.

`-net` *`physical_netname`*
: Specifies the name of a physical net in the current cluster context.

`-physical_pin` *`list_of_pinnames`*
: Defines a set of specific physical cell instance pins to which the physical net is connected to. The tool matches the pin names (such as `A/B/C`) with that of a physical instance pin name in the current cluster. A match is made if there is a pin `C` for a physical instance of name `A/B` in the current cluster. If no match is found, it looks for a logical instance with the hierarchy name `A/B` in the top context containing a physical pin C.

`-type {special | no_special}`
: Specifies which router will be used to connect the pins. Selecting `special` uses the special router to connect all pins, which will

appear in the SPECIAL NETS section. Selecting `no_special` uses wroute to connect all pins.
Default: `special`

Examples

The following commands state that if root cluster R contains a soft cluster C, and a power net VDD needs to be connected to top-level port VDD (assuming that the power port on C that receives the same net is called VDD!).

```
set_current_cluster /C
create_physical_pin VDD!
create_physical_net VDD -use power
add_physical_connection -net VDD -cluster_pin VDD! -connectivity_rule {"*VDD"}
set_current_cluster /
create_physical_pin VDD
create_physical_net -use power VDD
add_physical_connection -net VDD -cluster_pin {VDD C/Vdd!} -connectivity_rule
{"*VDD"}
```

Related Information

create_physical_net

create_physical_pin

set_current_cluster

check_design

`check_design` *`filename`*

Reports the cells of the design that don’t exist in the physical library or have different pin names.

Options and Arguments

`filename`

Specifies the name of the file that stores the report.

Related Information

check_libraries_and_design_compatibility

check_library

check_libraries_and_design_compatibility

```
check_libraries_and_design_compatibility
```

Checks the compatibility between physical and logical libraries to anticipate possible stoppages during optimization. It also checks the compatibility between the physical libraries and the design and pin name inconsistencies. The return values are `TCL_ERROR` and `TCL_OK`. The `TCL_ERROR` value states that there is an incompatibility between libraries and the design and optimization will stop. The `TCL_OK` value states that the libraries and design are compatible and optimization will proceed.

Related Information

check_design

check_library

check_library

`check_library [-logical] [-physical] [-power] filename`

Verifies the different properties associated with the design libraries, such as the power properties and the compatibility between the physical and logical libraries. This command also checks to see if pin names are the same.

Options and Arguments

filename
: Specifies the name of the file that stores the report.

-logical
: Reports the cells of the logical libraries that don't exist in the physical libraries.

-physical
: Reports the cells of the physical libraries that don't exist in the logical libraries and reports pin name problems.

-power
: Identifies libraries that contain power information.

  If you specify `check_library -power` *filename*, the command checks whether the cell has a power model and reports the names of cells that do not have a power model. Names are printed to the screen if a filename is not specified. You need an LPS license to run the command with this option.

  If you specify `check_library -power -logical -physical` *filename*, the command checks for a power model and then continue with a check for logical and physical models. Results are printed into the specified file. You need both an LPS and PKS license to run the command this way.

Examples

```
check_library -logical library_log.rpt
check_library -physical library_phy.rpt
check_library library.rpt
```

Related Information

check_design

check_libraries_and_design_compatibility

create_blockage

```
create_blockage [-bbox {lx ly ux uy}] [-layer list_of_layernames]
     [-instance name_or_id]
     [-type {placement_only | routing_only | soft_block | all}]
```

Creates a user-defined blockage in a specified layer or in all layers.

Note: PKS may ignore a blockage on a layer not specified by the `set_preroute_parameters` command. PKS may also ignore a blockage if it is smaller than the minimum size specified by the `set_preroute_parameters` command. However, these blockages will still be considered during global and final routing steps.

Options and Arguments

`-bbox {lx ly ux uy}`

Describes the shape and location of the blockage using the four coordinates {*lx ly ux uy*}.

`-instance name_or_id`

Attaches the blockage to any instance of CLASS BLOCK in the design (this needs to be an instance in the logical netlist). However, it has effect only when the block instance is placed. It has no effect otherwise. When the `-instance` option is specified in conjunction with a `-bbox` option, then the bbox is interpreted as applied to that portion of the instance in the normal 'N' configuration. Multiple such blockages can be applied to a particular instance to simulate a non-rectangular blockage pattern on an instance. If the `-bbox` option is not specified then by default, the blockage will be assumed to cover the whole instance.

NOTE: The bbox is relative to the specification of the block as in LEF. The translation to the die (absolute) coordinates will happen at the time of blockage usage.

In this context, it can be viewed as similar to the *Overlap Layer* definition in LEF; but can be done dynamically during the flow and also applied to only one instance of a macro as opposed to statically as with the Overlap Layer and LEF approach. These blockages will be used by the tool in addition to the LEF Overlap layer obstacles for the macro.

`-layer` *`list_of_layernames`*

Creates blockage only on the layers specified. This is an optional argument. If not specified, it will create a blockage at that location on all routing layers.

`-type {placement_only | routing_only | soft_block | all}`

Identifies the type of blockage you are creating. The blockage you create normally applies to both placement as well as routing.

`placement_only`
Specifies the blockage will only affect placement and will not cause any routing blockages. This could be useful in known congestion hotspots, like around macro block corners, to prevent the placer from adding cells in those corners that would add to congestion.

`all`
Specifies each blockage is applied for both placement and routing considerations based on the preroute parameters. If the `-type` option is not specified, `all` is used, and the blockage applies to both placement and routing.
Default: `all`

Advanced Options

`routing_only`
Specifies, the blockage will only affect routing and will not cause any placement blockages. A `routing_only` blockage can be used to add blockages for the Steiner router in PKS. As mentioned above, this blockage will not be seen by the global or final router and will not be output in the `.def` or `.wdb` file. This option can be used to make the Steiner router avoid certain regions without creating a real blockage that will be seen by the router. This kind of control is needed only for very specific situations and as such should be used with care. If the `-layer` option is given with a `routing_only` blockage, it needs to be consistent with the `set_preroute_parmeters` command settings to be considered.

`soft_block`
Creates a blockage that represents an external soft block to be handled appropriately during grow. When replacing pins and hard macros they will not overlap the soft-block boundaries in the

cluster being grown. This requires that you specify the above information for each block.

Any top-level blockage that is not of `soft_block` type is not included in the overlap check. It is to be expected that hard blocks may overlap with preroutes. This option is only useful in dynamic floorplanning situations.

Examples

- The following command creates a blockage on all layers for a specified bbox:

  ```
  create_blockage {10 20 30 30}
  ```

  Note: This by definition causes both a placement and a routing blockage for that region, resulting in modeling keepouts.

- The following command creates a blockage only on metal1:

  ```
  create_blockage {10 20 30 30} metal1
  ```

  Note: Depending on the place or route blockage layers specified using the `set_preroute_parameters` command, this may or may not define a place or route blockage.

Related Information

remove_blockage

report_blockage

create_layer_usages_table

`create_layer_usages_table` *`filename`* `[-number_intervals` *`int`*`]`

Creates a layer usages table for a routed design.

Options and Arguments

`filename` Name of output file.

`-number_intervals` *`int`*
Number of intervals you want in the layer usages table.

Additional Information

read_layer_usages

set_layer_usages_table

set_net_physical_attribute

write_layer_usages

create_physical_cluster

```
create_physical_cluster -name cluster_name
     [-allow_place_overlap {true | false}]
     [-allow_route_overlap {true | false}]
     {[-bbox {lx ly ux uy}] | [-utilization float] [-aspect_ratio float] |
     [-width float] [-height float]} [list_of_inst_or_ids]
     [-relative_cluster cluster_name] [-type {soft_block | region}]
```

Creates a sub-cluster of the specified name at the current cluster level. If the cluster already exists, the command will error out. After creating the physical cluster, you can use the `-modify_physical_cluster` command to place the cluster and stretch it appropriately.

Options and Arguments

`-allow_place_overlap {true | false}`
Allows the parent cluster to place over any placeable areas in the newly created cluster. Note that it does not place over any overlap layer obstacles created on the cluster. If `false` is selected, the newly created cluster is completely blocked for all placement at the parent level, even if overlap blockages are defined.
Default: `false`

`-allow_route_overlap {true | false}`
Allows the parent cluster to route over any routable areas in the newly created cluster. Note that it does not route over any overlap layer obstacles created on the cluster. If `false` is selected, the newly created cluster is completely blocked for all routing at the parent level, even if overlap blockages are defined.
Default: `false`

`-bbox {lx ly ux uy} | -utilization float -aspect_ratio float | -width float -height float`

You can use one of three methods (below) to define the initial starting shape for the physical cluster.

`-bbox` `{lx ly ux uy}`
Describes the shape and location of the cluster using the four coordinates `{lx ly ux uy}`. All coordinates are assumed to be absolute with respect to `UNPLACEDTOP` (except if `-relative_cluster` is specified).

-utilization *float* -aspect_ratio *float*
Estimates the appropriate bounding box by specifying the desired initial -utilization (0-100) and -aspect_ratio (>0). For this to be useful, the cluster should also be associated with instances. The placement information (if applicable) of any contained instances will be ignored. The available row area is estimated based on row spacing rules of the containing soft cluster. No estimation blockages are considered.

-width *float* -height *float*
Specifies the starting shape using the -width and -height options.

Note: If no options are specified, the value of 80% utilization and an aspect ratio of 1 will be used to estimate the initial starting shape. If no cells are assigned, then the default bbox size of 1x1 will be used.
Default: 80%

list_of_inst_or_ids
Specifies the name of the instances or IDs.

-name *cluster_name*
Specifies the name of the subcluster at the current cluster level.

-relative_cluster *cluster_name*
Treats the -bbox coordinates as relative to that cluster. Note that an error will result if the cluster specified by -relative_cluster is not a parent cluster of the current cluster and also a child cluster of UNPLACEDTOP.

-type {soft_block | region}
Defines the type of cluster you want to create. Note that if you request a -type soft_block and more than one or no hierarchical instance is specified, the command will error out.

Example

The following command creates a physical cluster C2 and allows for placement overlaps:

```
create_physical_cluster -name C2 -allow_place_overlap true
```

Related Information

modify_physical_cluster

create_physical_instance

```
create_physical_instance instance_name [-location location]
     [-macro macroname] [-orient {N | S | W | E | FE | FW | FS | FN}]
     [-state {placed | fixed | unplaced | cover}]
     [-type {spare | filler | core | cover}]
```

Creates a physical instance in the `set_current_cluster` context. The macro must exist in the LEF file. If not, an error is given. Note that this will always create an instance in the current cluster even if the name is intended to be physically hierarchical. The *name* must not clash with the logical instance name (or physical instance name) at the containing soft-block cluster level.

Note: *name* is non-hierarchical so you need to be in the appropriate `set_current_cluster` context prior to issuing this command.

Options and Arguments

`-location` *loc*
: Specifies the location where the physical instance will be placed. When specified, automatically changes the state from unplaced (if applicable) to placed. If `-location` is not specified, the physical instance will be in an unplaced state and will not be accounted for in LUT calculations. Location coordinates are absolute in the context of UNPLACED_TOP.
Note that you must type the curly braces in the command syntax.

`-macro` *macroname*
: Specifies the name of the macro.

`-orient {N | S | W | E | FE | FW | FS | FN}`
: Specifies the orientation of the physical instance.

`-state {placed | fixed | unplaced | cover}`
: Identifies the state of the instance as `placed`, `fixed`, `unplaced`, or `cover`. Specifying the state as `fixed` or `cover` results in an error if the physical instance is not currently in a placed state, or the `-location` option is not specified at the same time.

`-type {spare | filler | core | cover}`
: Specifies the macro type.

Example

The following command creates a physical instance in the current cluster at the absolute location (200, 300) assuming there are no unplaced clusters other than the top:

```
create_physical_instance -macro ANDX2 -location {200 300}
```

create_physical_net

```
create_physical_net net_name [-use {power | ground | clock | signal | analog |
     tieoff}]
```

Creates specially-routed nets, such as VDD and GND. It can also be used for physical only Wrouted nets.

Options and Arguments

net_name
: Specifies the non-hierarchical name to be associated with the physical net. The name must not clash with a logical net or another physical net in the same cluster.

`-use {power | ground | clock | signal | analog | tieoff}`
: Specifies the type of physical net you are creating.

Example

The following command creates physical net VDD and designates it as a power net:

```
create_physical_net VDD -use power
```

create_physical_pin

```
create_physical_pin pin_name [-layer layername] [-bbox bounding_box]
     [-location location] [-size size][-index pin_index] [-side side_char]
     [-state placement_state] [-type {special | no_special}]
     [-use {power | ground | clock | signal | analog | tieoff}]
```

Creates a pin of specified *name* in the current cluster. This command can only be applied on a soft cluster. The name of the pin cannot clash with any logical pin for that soft cluster. It is also not interpreted for hierarchy.

Options and Arguments

`-bbox` *bounding_box*

Defines the geometry for the pin.

`-index` *pin_index*

Defines the relative index for the pin on the side it has been assigned. Accepts any *number* value greater than 1. This is used in conjunction with the `-side` option. Cannot be used in conjunction with the `-location` or `-bbox` options (except during grow).

`-layer` *layername*

Defines the layer of where the pin is placed.

name

Specifies the name of the pin your creating.

`-side` *side_char*

Defines the side that the pin is assigned. Can be one of the following: `{left | right | bottom | top}`. This is used in conjunction with the `-index` option. Cannot be used in conjunction with the `-location` or `-bbox` options (except during grow).

`-state` *placement_state*

Defines the state that the pin is assigned. Can be one of the following: `{fixed | placed | cover | unplaced}.`

Default: `unplaced`

`-type {special | no_special}`
Specifies if the physical pin is of type `special` or `no_special`.

`-use {power | ground | clock | signal | analog | tieoff}`
Specifies the type of physical pin you are creating.

Example

The following command creates pin A on layer M2 as an unplaced signal pin positioned as pin 2 on the top side:

```
create_physical_pin A -layer M2 -index 2 -side top -use signal -state unplaced
```

Related Information

get_pin_location

create_placement_area

`create_placement_area [-name` *`name`*`] lx ly ux uy`

Creates a named placement area. The placement areas work in conjunction with DEF rows. Use the placement areas to limit the region where you want the cells to be placed. Useful for:

- Low utilization design

  Such as forcing QP to a cluster in the middle.

- Part of a semi-hierarchical flow
- Avoiding thin strips of row area (around macros)

The command generates an error if:

- The specified box intersects the chip die box.
- The specified name already exists.

The command generates a warning if:

- The specified box intersects another placement area.
- The specified box is adjusted to account for the rows along the edges. Supports both horizontal and vertical rows.

Options and Arguments

`lx`
: The lower left *x* coordinate.

`ly`
: The lower left *y* coordinate.

`-name` *`name`*
: Name of the placement area.

`ux`
: The upper right *x* coordinate.

`uy`
: The upper right *y* coordinate.

Example

```
create_placement_area -name a1 100. 200. 1000. 1100.
```

Related Information

remove_placement_area

report_placement_area

do_extract_lef_model

```
do_extract_lef_model [-all | -cluster list_of_clusters] -file filename
```

Passes a simplified LEF model of all soft-block sub-clusters to Qplace and Wroute. The LEF file contains basic abstracts. Information contains size, soft-block cluster pins, and route blockages (at the level of the internal blockage maps) if the `allow_overlap` argument is `true`. If not, a block on all routing layers is introduced. For hierarchical designs, use this command to create a LEF file for each soft-block cluster if it is being written to a DEF file and read into an external tool, such as for final route. This is because the DEF file will instantiate any soft-block sub-clusters as LEF instances.

Note: You can only model soft-block clusters. Cluster names can be hierarchical.

Options and Arguments

`-all | -cluster {list_of_clusters}`
: Creates a macro definition for each soft block in the current top-timing module (only current module soft blocks, no child soft blocks). Note that `-all` creates a macro definition of all soft blocks and `-cluster` creates a macro definition of all soft blocks specified in `list_of_clusters`. The cluster names that you specify must be with respect to the current top-timing module. Use the `do_push` command if you want to get a macro listing of the child soft blocks.

`-file filename`
: Specifies the name of the LEF file you are extracting.

Example

The following command creates a LEF macro abstract for `SOFT_BLK1` and `SOFT_BLK2` clusters as specified in `-cluster {list_of_clusters}`. The macro name will be the module name of the instance associated with the soft-block cluster:

```
do_extract_lef_model -cluster {"SOFT_BLK1", "SOFT_BLK2"} -file my.lef
```

do_extract_route_parasitics

`do_extract_route_parasitics` *`wdb_name`*

Note: This command is being replaced by the `read_wdb -rspf_only` command. This command is still available but will become obsolete in the next full release of the software.

Reads in a Wroute database that contains either global and or detailed routing of the design, extracts the parasitics from the routing segments, and annotates them to the nets in the design for timing analysis use. After extraction, the routing information is deleted.

The Wroute database contains physical library information (LEF), design netlist information (DEF), and global and or detailed routing information. The Wroute database has to be consistent with ADB or in-core design data at the time of extraction. Otherwise, parasitics cannot be extracted and annotated correctly. As an example, the ADB saved right before or after `do_route` is consistent with the WDB that is saved at the end of `do_route`. If one of them has been changed after saving, they can potentially become inconsistent.

Options and Arguments

`wdb_name`

Specifies the name of the Wroute database.

Related Information

read_wdb

do_groute

```
do_groute -input_db_name filename [-assign_routing_direction {true | false}]
     [-def_name list_of_file_names] [-def_out_name filename]
     [-discourage_planar_routing {true | false}] [-follow_pins]
     [-force {true | false}] [-ggrid_max_size 16] [-ggrid_min_size 5]
     [-ggrid_mode {user | align | uniform}] [-ggrid_opt_size 10]
     [-groute_num_opt_pass num] [-ignore_unplaced_component {true | false}]
     [-no_follow_pins] [-output_db_name filename]
     [-output_full_def {true | false}] [-pin_access_compat_mode {true | false}]
     [-processors] [-region_relax_percent] [-route_area {lx|ly|ux|uy}]
     [-route_bottom_layer_limit value] [-route_on_grid_only {true | false}]
     [-route_pin_access_compat_mode {auto | normal | offGrid | upViaEnclosure |
     upVia | upViaReserve | narrowPin}] [-route_select_net [~] list_of_net_names]
     [-route_select_net_first {true | false}]
     [-route_select_net_layer_range {layer:layer | layer}]
     [-route_soft_top_layer_limit {true | false}]
     [-route_stripe_center_connect {true | false}]
     [-route_stripe_follow_pin_only {true | false}]
     [-route_top_layer_limit value]
     [-route_use_range_rule obstruction | pin | obstruction+pin]
     [-straighten_net net_name_list] [-timing_driven {true | false}]
     [-timing_net_percentage_delta [0 to 100]] [-use_auto_ggrid {true | false}]
     [-use_max_xy_spacing {true | false}] [-version]
```

Starts Wroute in standalone mode to do a global routing on the design and saves the global routing information in the Wroute database. See do_wroute on page 535 for more information.

Note: You must specify the path to the `wroute` executable, either by including it in your UNIX search path, or setting the `pks wroute exe` global.

Options and Arguments

`-assign_routing_direction {true | false}`
: Tells the router, when true, to automatically assign routing directions in the channels between blocks to take advantage of available routing tracks. This variable is useful, for example, if channel congestion between blocks occurs because the preferred routing tracks are perpendicular to the orientation of the channel. In this case, this variable lets the router reassign the preferred routing direction in the channel.
Default: `false`

`-def_name list_of_file_names`
: Specifies the name of the DEF file(s) that Wroute reads.

`-def_out_name` *`filename`*

Specifies the name of the output DEF file that Wroute writes to.

Note: This option is primarily used to change the output DEF *`filename`*. If this option is not used, the DEF is automatically written to the following default DEF filename: `wdb_name.def`.

`-discourage_planar_routing {true | false}`

Eliminates the use of planar routing over blocks
Default: `false`

`-follow_pins`

Removes existing power rails (if applicable) and generates new ones.

`-force {true | false}`

Stops global routing, if `false`, when more than 90% of the nets are final routed. If true, will continue global routing while preserving all final routes. Note that global routing will continue and also issue a warning when < 90% of the design is final routed.

`-ggrid_max_size 16`

Sets the maximum gcell grid size in tracks. *`ggrid`* size has a significant effect on QOR (quality of results). It is recommended you use a smaller *`ggrid`* size unless your design has a large number of obstacles or is memory constrained. See `-ggrid_min_size` and `-ggrid_opt_size` for more options.

`-ggrid_min_size 5`

Sets the minimum gcell grid size in tracks. *`ggrid`* size has a significant effect on QOR (quality of results). It is recommended you use this *`ggrid`* size unless your design has a large number of obstacles or is memory constrained. See `-ggrid_max_size` and `-ggrid_opt_size` for more options.

`-ggrid_mode {align | uniform | user}`

Sets the global routing grid generation mode. The global routing grid partitions the routing portion of a standard cell design into rectangles called gcells. The arguments for this variable set the grid as follows:

align
Generates a global routing grid that aligns with the channels and macro block boundaries in your design. This argument improves routing quality for block-based designs because, by aligning the grids, the global router can route more effectively in empty gcells in the channel area.

uniform
Generates a uniform global routing grid.

user
Uses the global routing grid defined in the DEF file.
Default: `user`

`-ggrid_opt_size 10`
Sets the optimal gcell grid size in tracks. *`ggrid`* size has a significant effect on QOR (quality of results). It is recommended you use a smaller *`ggrid`* size unless your design has a large number of obstacles or is memory constrained. See `-ggrid_max_size` and `-ggrid_min_size` for more options.

`-groute_num_opt_pass` *`num`*
Sets the number of optimization passes in groute, specified by the value *`num`*. If no value is given for *`num`*, the default value is used.
Default: Auto tuned.

`-ignore_unplaced_component {true | false}`
Specifies not to route to the pins of unplaced components.
Default: `false`

`-input_db_name` *`filename`*
Specifies the database directory.

`-no_follow_pins`
Removes existing power rails (if applicable).

`-output_db_name` *`filename`*
Specifies the router's output database. The database is in binary form. To save your design into a router database without running routing, specify the database *`filename`* with this option and set all the routing variables to `false`. The router will save your design into database format without making any routing passes.

Note: This option is primarily used to change the output database *`filename`*. If this option is not used, the database is automatically written to the following default output database filename: `design_name.wdb`. This option is useful, for example, if you are routing several different floorplans or placements. You can use this option to save each database that the router creates to a different routing binary. Then you can load the best results from the saved database.

`-output_full_def {true | false}`
Outputs a full DEF file, when `true`, instead of a DEF file containing only placement information.
Default: `true`

`-pin_access_compat_mode {true | false}`
Enables pin access compatibility mode.
Default: `false`

`-processors`
Specifies the number of processors to use while running global routing. The maximum is 4.
Default: 1

`-region_relax_percent`
Controls group and region constraints. Constraints are ignored if the percent is negative. Takes values 0.0 - 1.0.
Default: `0`

`-route_area {lx ly ux uy}`
Routes the specified area. The coordinates for the area are in database units. If no arguments are given, the full chip is routed.

`-route_bottom_layer_limit [0]`
Limits the bottom routing layer to the specified value. A value of 0 disables this option. Routing below the specified layer is allowed, but at a very high cost. Thus, the wire length below the specified layer is kept short. For example, in a design with six layers, a value of 4 requests that the router start routing at the fourth layer.

`-route_on_grid_only {true | false}`
Routes to the on-grid geometries of port only.
Default: `false`

`-route_pin_access_compat_mode {auto | normal | offGrid | upViaEnclosure | upVia | upViaReserve | narrowPin}`
Sets the option for processing pin geometries. The arguments for this variable are:

`auto`
Processes the pin geometries as it did in previous versions of the software. This is the default value for this environment variable.

`normal`
Processes the pin geometries as it did in previous versions of the software. This is the default value for this environment variable.

`offGrid`
Shrinks the pin geometries by one-half the wire width. The reminder of the pin geometry is valid for the router to make a connection using a regular wire. The router checks the connection using a one-half wire width plus spacing rule.

`upViaEnclosure`
Shrinks the pin geometries by one-half a via width. The remainder of the pin geometries is valid for the router to make a connection using a via that is fully enclosed by the pin. The router checks the connection using a one-half via width plus spacing rule.

`upVia`
Does not shrink the pin, so the full size of the pin geometry is valid for the router to make a connection using a via. The router checks the connection using a one-half via width plus spacing rule.

`upViaReserve`
Shrinks the pin geometries by one-half the wire width. The remainder of the pin geometries is valid for making a connection using a via. The router checks the connection using a one-half via width plus spacing rule.

narrowPin
Does not shrink the pin geometries. The full size of the pin geometries is valid for making a connection using a regular wire. The router checks the connection using a one-half via width plus spacing rule.

`-route_select_net [~] list_of_net_names`
where ~ represents negation.
Using ~ routes all but the specified nets.
No ~ routes only the specified nets.

`-route_select_net_first {true | false}`
Routes the selected nets first, followed by all other nets.
Default: `false`

`-route_select_net_layer_range {layer:layer | layer}`
Routes the selected nets on the specified layer range.

`-route_soft_top_layer_limit {true | false}`
Specifies that the top routing limit is soft, if set to `true`.
Default: `false`

`-route_stripe_center_connect {true | false}`
Connects to the center of the stripe only. Does not connect to POWER and GROUND pins.
Default: `false`

`-route_stripe_follow_pin_only {true | false}`
Connects to the follow pin stripe only.
Default: `false`

`-route_top_layer_limit [value]`
Limits the top routing layer to the specified value. A value of 0 disables this argument. A value of n means the placer uses routing resources from layer `1` to layer `n`.
Default: `0`

`-route_use_range_rule (obstruction | pin | obstruction+pin)`
Specifies the clearance distance used for spacing rule checks using one or more of: pin, obstruction, or obstruction+pin.
Default: `obstruction+pin`

`-straighten_net` *`net_name_list`*
Straightens the selected nets specified by `net_name_list`.
Default: `[ ]`

`-timing_driven {true | false}`
Turns on timing-driven routing when set to true. To perform timing-driven routing, the router decides the order in which to route the constrained nets based on their timing criticality, and routes them accordingly. The most timing-critical nets are routed first to achieve the shortest routes.

The router modifies its routing priorities as appropriate during the routing process. For example, if the router places a net low on the priority list to start, but later realizes that the net must be routed more directly to meet timing constraints, it will reroute the net. The router also considers the layer RC values when determining how to route the timing-critical nets.
Default: `true`

`-timing_net_percentage_delta [0 to 100]`
Specifies the percentage change in the timing-critical nets.
Default: `0`

`-use_auto_ggrid {true | false}`
Uses Wroute's default auto tune settings, if set to `true`. If `false`, uses the PKS smaller ggrid size.
Default: `true`

`-use_max_xy_spacing {true | false}`
Uses the maximum values of `x` and `y` for spacing checks.
Default: `false`

`-version`
Checks the version number and build information for routing. This option only works after a gate-level netlist is loaded.

do_initialize_floorplan

```
do_initialize_floorplan
```

Initializes the die box and other floorplan parameters based on the floorplan parameters set using the `set_floorplan_parameters` command or using the `read_def` or `read_pdef` commands. This command also initializes the pin positions based on the pin locations from `read_def` or `read_pdef` commands, or from the pin indices set by the user.

Related Information

report_floorplan_parameters

do_insert_filler_cells

```
do_insert_filler_cells -macro {list_of_filler_cell_macro_names}
     [-prefix string] [-area {lx ly ux uy}]
```

Inserts filler cells in empty regions of rows in a legally placed design. If the design is not legally placed, an error is generated.

Options and Arguments

`-area {lx ly ux uy)`
: Specifies the area where filler cells are to be placed. The area is a rectangle where *lx* and *ly* represent the x and y coordinates of the lower left corner of the rectangle, and *ux* and *uy* represent the x and the y coordinates of the upper right corner. Coordinate values are in microns.

`-macro {list_of_filler_cell_macro_names}`
: Creates a list of LEF macro cells that can be used as filler cells. You must type the curly braces when providing the list of macros.

`-prefix string`
: Specifies the prefix to the filler cell name that will be added.
Default: PKSFILLER_

Example

The following command uses the ANDX2 macro as a filler cell:

```
do_insert_filler_cells -macro {ANDX2}
```

Related Information

do_remove_filler_cells

do_place

```
do_place [-area_util {l b r t util}] [-block {true | false}]
     [-clock_buffer_site site] [-cong_max_util util]
     [-cong_mode2 {true | false}] [-cut_ratio float]
     [-def_name list_of_file_names] [-def_out_name file_name]
     [-eco {true | false}] [-fix_placed_cell {true | false}] [-batch_mode]
     [-free_track_percent_1 0-1] [-free_track_percent_2 0-1]
     [-free_track_percent_3 0-1] [-free_track_percent_4 0-1]
     [-free_track_percent_5 0-1] [-free_track_percent_6 0-1]
     [-free_track_percent_7 0-1] [-free_track_percent_8 0-1]
     [-free_track_percent_9 0-1] [-groute_analysis report_filename]
     [-groute_num_opt_pass num] [-incremental {true | false}]
     [-incr_size size] [-llc_ignore_layer_1 {true | false}]
     [-llc_ignore_layer_2 {true | false}] [-llc_ignore_layer_3 {true | false}]
     [-llc_ignore_layer_4 {true | false}] [-llc_ignore_layer_5 {true | false}]
     [-llc_ignore_layer_6 {true | false}] [-llc_ignore_layer_7 {true | false}]
     [-llc_ignore_layer_8 {true | false}] [-llc_ignore_layer_9 {true | false}]
     [-llc_prewire_keep_out {true | false}] [-mega_cell_keep_out_x distance]
     [-mega_cell_keep_out_y distance]
     [-net_weight "tcl_or_net_name_list integer_weight"]
     [-output_full_def {true | false}] [-pin {concurrent|refine}]
     [-pin_layer {left|bottom|right|top} layer_name]
     [-pin_same_side {true | false}] [-place_area {l b r t}]
     [-place_io_net_weight [1 to 100]] [-region_exclusive {true | false}]
     [-region_exclusive_name name1 name2 ...] [-region_relax_percent [0 to 1]]
     [-route_bottom_layer_limit [0 to 9]] [-route_top_layer_limit [0 to 9]]
     [-scan_reorder] [-spare_cell cell_name_list] [-timing_driven {true | false}]
     [-timing_io_net_weight [1 to 10]] [-timing_weight weight]
     [-use_region {true | false}] [-version]
```

Creates the initial placement of the design.

Options and Arguments

`-area_util l b r t util`

Defines the row utilization for an area (*l* = left, *b* = bottom, *r* = right, *t* = top, *util* = utilization}. The *l*, *b*, *r*, and *t* arguments are region coordinates, in database units (integer). Utilization is defined as the ratio of the instance area and the available area for placement in the given region (0 < *util* < 1). *Default*: " "

`-batch_mode`

Invokes `qplace` in a standalone batch mode and reads the placement data back into PKS. Works with all `do_place -eco` options.

Note: Can only be used in conjunction with the `-eco true` option.

`-block {true | false}`

Places the blocks.
Default: `false`

`-clock_buffer_site` *`site`*

Reserves *n*-sites (integer or floating number) around every core component connected to nets marked +USE CLOCK for the clock buffers to be inserted later on during clock tree synthesis.
Default: 0.

`-cong_max_util util`

Defines the maximum local utilization. *`util`* is a float value, and the range is from 0-1.
Default: Auto tuned.

`-cong_mode2 {true | false}`

Allows the placer, when set to `true`, to reserve extra empty space after each pass, producing placement results with high cell density in routable areas and low cell density in congested areas. As a result, congestion is spread out more evenly to provide better routability, and die size is optimized for the smallest possible area. This variable is especially appropriate for high utilization designs with no region constraints and placement run in non-timing mode.
Default: In general, use the default setting, which is auto tuned as follows:

True for core cell placement without timing and region constraints.

False for core cell placement with timing constraints because of the die size optimization. Some areas might become too congested to allow room for buffers or resized gates added as the result of timing-related engineering change orders (ECOs).

False for core cell placement with region constraints because cells in some regions might not spread out, which reduces routability in that area.

True for incremental, block, and ECO placement.

`-cut_ratio` *`float`*

Defines the ratio between the cuts in the horizontal direction versus the cuts in the vertical direction. This is used to adjust connection densities in either direction.
Default: `1` (0.01 < r < 100)

`-def_name` *`list_of_file_names`*

Defines additional DEF filenames.
Default: There is no default setting.

`-def_out_name` *`filename`*

Specifies the name of the output DEF file.

`-eco {true | false}`

Makes minor placement modifications for the engineering change order (ECO). If applicable, turn region constraints on to place the component in the requested region without moving the already-placed components. When used with the `-net_weight` option, it allows you to place the new component as close as possible to the already-placed components. Use with the `-timing_driven` option to control the locations of the new components.
Default: `true`

`-fix_placed_cell {true | false}`

Fixes all the placed components if true.
Default: `false`

`-free_track_percent_1 0-1`

Defines the percentage of free routing tracks for layer 1specified by the user-defined value from `0-1` (.5 = 50%). The `1` in this command can be replaced with the values [1-9] depending upon which layer is specified.
Default: Auto tuned.

`-groute_analysis report_file_name`
Performs groute to generate the congestion map. Analysis is turned off using " ".

`-groute_num_opt_pass num`
Defines the number of optimization passes in groute. `num` is an integer value.
Default: Auto tuned.

`-incremental {true | false}`
Makes incremental placement and optimizations based on existing placement. Incremental placement moves cells locally only. For this reason, it might not achieve the values you set with `-cong_max_util` or `-area_util`. If so, turn off incremental placement and run placement optimization again.
Default: `false`

`-incr_size` *`size`*
Uses the number of instances as a starting point for incremental placement. Incremental size is in number of instances.

`-llc_ignore_layer_1 {true | false}`
Turns the legal location check off (true) for specified layer `1`. The `1` in this command can be replaced with the values [`1-9`] depending upon which layer you specify.
Default: Auto tuned.

`-llc_prewire_keep_out {true | false}`
Places core cells away from prewires with appropriate design rule.
Default: `false`

`-mega_cell_keep_out_x` *`distance`*
Provides space (in microns) in the horizontal direction around the macro cells.

`-mega_cell_keep_out_y` *`distance`*
Provides space (in microns) in the vertical direction around the macro cells.

`-net_weight "`*`tcl_or_net_name_list integer_weight`*`"`
Assigns weight on the nets specified by

tcl_or_net_name_list to give priority for placement in minimizing their wire length. You must type the double quotation marks in the syntax, as in the following examples:

```
do_place -net_weight "tmp1 100"
or
set listA {tmp1 tmp2 tmp3}
do_place -net_weight "$listA 10"
```

`-output_full_def {true | false}`
Outputs a complete DEF instead of a DEF with only placement information.

`-pin {concurrent | refine}`
Performs pin placement only. Used only with the Cadence ultra placer. The `concurrent` argument places all cells and pins at the same time and ignores initial placement. The `refine` argument re-places only pins. All cells must be pre-placed to use the `refine` option, otherwise the placer quits.

Pins marked as `fixed` or `cover` are not touched during placement. Pins marked as `placed` are not moved if you specify the `concurrent` option, but can be moved if you specify the `refine` option.
Default: `false`

`-pin_layer {left | bottom | right | top}` *layername*
Designates a layer name for pins on the left, bottom, right, and top sides. Used only with the Cadence ultra placer. Designs with fewer than three routing layers use metal1 and metal2 for the default layers. Designs with three or more routing layers use metal2 and metal3 for the default layers.

`-pin_same_side {true | false}`
Places pins to their original side. Used only with the Cadence ultra placer.
Default: `false`

`-place_area {`*l b r t*`}`
Places instances within the left, bottom, right, or top bounding box areas. The `l`, `b`, `r`, and `t` arguments are region coordinates, in database units (integer). Used only with the Cadence ultra placer.

`-place_io_net_weight [1 to 100]`
Specifies the weight factor for nets connected to blocks, pads, and IOs.
Default: `1`

`-region_exclusive {true | false}`
Turns the exclusive region on if `true`.
Default: `false`

`-region_exclusive_name` *`name1 name2 ...`*
Specifies that only the components in the specified groups will be placed exclusively in the corresponding regions of the specified groups. Note that this command must be used with the `-region_exclusive` option.
Default: `""`.

`-region_relax_percent [0 to 1]`
Specifies a float value (such as 0.2), which controls the group and region constraints (constraints are ignored if the percentage is negative).
Default: `0`

`-route_bottom_layer_limit 0-9`
Limits the bottom routing layer to the specified value. Used only with the Cadence ultra placer.
Default: `0`

`-route_top_layer_limit 0-9`
Limits the top routing layer to the specified value. Used only with the Cadence ultra placer.
Default: `0`

`-scan_reorder`
Disconnects the scan chain (if any) before placement and reconnects them based on the scan element's physical location after placement. See *Test Synthesis for Ambit BuildGates and Cadence PKS* for more information.

`-spare_cell` *`cell_name_list`*
Specifies the spare cells to be placed randomly. Qplace will automatically place the spare cells in the design randomly within the die area.

Requirements for incorporating spare cells includes:
A spare cell is automatically detected if a cell has at least one signal or clock pin (other than power, ground, feedthrough, and so) and that no pin (any pin) connects to a signal or clock net which is at least a two-pin net.

If you do not want to use auto detection mode, need to include more cells as spare cells, or have a subset of spare cells, use the `-spare_cell` option. For example:

```
set spareCells {spare1 spare2 spare3 spare4}
do_place -spare_cell $spareCells -timing_driven
false
```

where spareN represents the instances in the design.
Default: Auto detect

`-timing_driven {true | false}`
Performs timing-driven placement. In general, the placer auto tunes the timing arguments to provide the best possible placement results. If running the placer with the default settings does not give you satisfactory results, you can reset some arguments and rerun the placer.
Default: `false`

`-timing_io_net_weight`
Specifies an extra weight factor for the IO timing net.

`-timing_weight` *`weight`*
Places weight on the timing-driven placement effort. Sacrifices wire length to meet timing.
Default: Automatically generated.

`-use_region {true | false}`
Uses the non-area cluster as region constraints. Use this option to ignore any regions that have been read in using the read_def command.

Default: `true` unless the `-eco false` option is specified.

`-version`
Checks the version number and build information for placement. This option only works after a gate-level netlist is loaded.

Example

The following command places all blocks in the design:

```
do_place -block true
```

do_pull

```
do_pull [-cluster cluster_name | -instance inst_name] [-no_blockages]
     [-no_floorplan] [-no_physical] [-no_rows] [-physical_only]
     [-propagate_constraints]
```

Copies (pulls) *design constraints* and *physical design characteristics* (associated with a specified soft-block cluster or instance) from the lower-timing module context and applies them to the current top-timing module context.

Note: The following information will be ignored:

- Information that was tagged as *shadow* by the `do_push` command.
- Information that was inherited by the `do_push` command, unless the information was redefined in the lower-timing module context. If the information has changed, it will replace the existing information in the current top-timing module context.

This command primarily copies all place and route information and replaces the existing information in the current top-timing module context.

The *design constraints* include timing, electrical, power, and test. Refer to the `do_derive_context` command for more information.

The *physical design characteristics* include:

- Placement information for all instances hierarchically contained in the soft-block cluster
- Placement information for all physical instances contained in the current module
- Physical cluster-specific information, such as bounds
- Floorplan information associated with the current module (rows, tracks, gcells, floorplan spec, preroute spec, and power stripe spec)
- Special net connectivity and wiring
- Regular net routing
- Blockages
- Cluster pin information
- DEF properties and definitions
- Shadow physical instances to model any instances that overlap the current module, but are contained in other soft clusters (top-level buffers)

Options and Arguments

`-cluster` *`cluster_name`* | `-instance` *`inst_name`*
: Specifies the name of the soft-block cluster or instance containing the design characteristics you want to pull.

`-no_blockages`
: Specifies that no blockage information should be pulled.

`-no_floorplan`
: Specifies that no floorplan information should be pulled. Floorplan information includes cluster specifications, preroute specifications, and floorplan parameters.

`-no_physical`
: Specifies that only the timing characteristics of the specified soft-block cluster (or instance) is pulled from the lower-timing module context and placed in the current-timing module context.

`-no_rows`
: Specifies that no row information should be pulled.

`-physical_only`
: Specifies that only the *physical design characteristics* (of the specified soft-block cluster or instance) will be pulled from the lower-timing module context and placed in the current top-timing module context. Note that the *design constraints* will not pass.

`-propagate_constraints`
: Pulls the timing constraints on a boundary pin in the lower-timing context and applies them to a boundary pin in the top-timing context.
Note: The boundary pin in the top-timing context must be directly connected to the boundary pin in the lower-timing context. It also pulls the module port user assertions and applies them to the hierarchical module instance pin in the top-timing context. This command propagates the following constraints:
– slew time assertion
– input delay assertion
– drive assertion
– clock root assertion
– external delay assertion
– constant assertion

– data arrival time assertions
– clock arrival time assertions
– data required time assertions
– clock required time assertions

If the instance pin of the block is directly connected to a top level input port, the above constraints are transferred to it. Otherwise, these constraints are ignored, except the `set_constant_for_timing` command, which is retained on the instance pin of the block.

Examples

- The following command pulls `inst1` (soft block) and places it below `inst0` in the current cluster:

  ```
  do_pull -instance {inst0/inst1}
  ```

- The following command pulls `inst1` (soft block) and places it below `inst0` in the current cluster, but does not pull any floorplan information:

  ```
  do_pull -instance {inst0/inst1} -no_floorplan
  ```

- The following command pulls `inst1` (soft block) and places it below `inst0` in the current cluster, but does not pull any blockage information:

  ```
  do_pull -instance {inst0/inst1} -no_blockages
  ```

- The following command pulls the information from cluster `C2` (subcluster of `C1`) and places it in cluster `C1`. Cluster `C1` must be present in the current top-timing module:

  ```
  do_pull -cluster {C1/C2}
  ```

- The following command pulls the information from cluster `C2` (subcluster of `C1`) and places it in cluster `C1`, but only pulls the timing constraints of the boundary pins from `C2` into `C1`:

  ```
  do_pull -propagate_constraints -cluster {C1/C2}
  ```

- The following command pulls the information from cluster `C2` (subcluster of `C1`) and places it in cluster `C1`, but only pulls the timing characteristics from C2 into C1:

  ```
  do_pull -no_physical -cluster {C1/C2}
  ```

Related Information

`do_derive_context`

`do_push`

`set_top_timing_module`

do_push

```
do_push [-cluster cluster_name | -instance inst_name]
     [-budget_style {time_budget | derive_context] [no_physical] [-no_shadow]
     [-physical_only]
```

Copies (pushes) design constraints and physical design characteristics (associated with a specified soft-block cluster or instance) from the current top-timing module context to a lower-timing module context.

Note: Objects overlapping the soft-block cluster in the top-timing module context are also pushed to the lower-timing module context unless you specify the `-no_shadow` option. This is one of the fundamental capabilities for performing top-down hierarchical synthesis.

Design constraints include:

- Timing
- Electrical
- Power
- Test.

Physical design characteristics include:

- Placement information for all instances hierarchically contained in the soft-block cluster
- Placement information for all physical instances contained in the current module
- Physical cluster-specific information, such as cluster bounds

  Cluster bounds are the coordinate limits of its bounding box, for example, if the cluster has a bounding box of {0 5 100 200}, then 0 is the lower bound in the x- direction, 100 is the upper bound in x, 5 is the y lower bound while 200 is the y upper bound. Generally everything belonging to a cluster must physically lie within a bounding box defined by its bounds.
- Floorplan information associated with the current module (rows, tracks, gcells, floorplan spec, preroute spec, and power stripe spec)
- Special net connectivity and wiring
- Regular net routing
- Blockages
- Cluster pin information
- DEF properties and definitions

- Shadow physical instances to model any instances that overlap the current module, but are contained in other soft clusters (top-level buffers)

Options and Arguments

`-budget_style {time_budget | derive_context}`
Specifies that timing characterization is to be done using the time budgeting or derive context method.

`-cluster` *`cluster_name`* `| -instance` *`inst_name`*
Specifies the name of the soft-block cluster or instance containing the design characteristics you want to push.

`-no_physical`
Specifies that only the timing characteristics of the specified soft-block cluster or instance will be pushed from the top-timing context and placed in the lower-timing context.

`-no_shadow`
Specifies that no objects overlapping the soft-block cluster in the top-timing module context are to be pushed to the lower-timing module context. Useful in a situation where you just want to update the contents of a lower-timing module with a `read_def`.

`-physical_only`
Specifies that only the *physical design characteristics* (of the specified soft-block cluster or instance) will be pushed from the top-timing module context and placed in the lower-timing module context. Note that the *design constraints* will not be passed. Useful in a situation where you just want to update the contents of a lower-timing module with a `read_def`.

Examples

- The following command pushes `inst1` (soft block) below `inst0` in the current cluster:

```
do_push -instance {inst0/inst1}
```

- The following command pushes `inst1` (soft block) below `inst0` in the current cluster, but does not push any objects overlapping `inst1`:

```
do_push -instance {inst0/inst1} -no_shadow
```

- The following command pushes `inst1` (soft block) below `inst0` in the current cluster, but only the timing characteristics of `inst1` are pushed:

  ```
  do_push -instance {inst0/inst1} -no_physical
  ```

- The following command pushes cluster `C2` (soft block subcluster of `C1`) to the lower-timing module:

  ```
  do_push -cluster {C1/C2}
  ```

  Cluster `C1` remains present in the current top-timing module.

Related Information

`do_derive_context`

do_pull

`set_top_timing_module`

do_refine_place

```
do_refine_place
```

Starts Encounter™ in batch mode and runs the `refinePlace` command to correct flawed cell locations. Once the legalization is complete, the final placement is read back into PKS through a DEF file generated by Encounter.

Note: You must specify an Encounter executable for PKS to use, either by having the executable in your UNIX search path, or by setting the `pks_encounter_exe` global.

Additional Information

`pks_encounter_exe`

do_remove_filler_cells

```
do_remove_filler_cells [-area {lx ly ux uy}]
```

Removes all filler cells in the entire design or in the specified area that do not have a placement state of FIXED or COVER.

Options and Arguments

`-area {lx ly ux uy}`

Specifies the area from which filler cells are to be removed. The area is a rectangle where *lx* and *ly* represent the x and y coordinates of the lower left corner of the rectangle, and *ux* and *uy* represent the x and the y coordinates of the upper right corner. Coordinate values are in microns.

Related Information

do_insert_filler_cells

do_remove_route

```
do_remove_route [-all] [-net net_name]
```

Deletes the specified routes in the design. If no options are specified, this command removes all regular routes generated by Wroute (no power routing is removed).

Options and Arguments

`-all`
: Deletes all routes in the design, including power routes.

`-net net_name`
: Deletes all routes under the specified *net_name*.

Examples

- The following command deletes all routes routed by Wroute:

  ```
  do_remove_route
  ```

- The following command deletes all routes in the design:

  ```
  do_remove_route -all
  ```

- The following command deletes all routes under the specified *net_name*:

  ```
  do_remove_route -net net_name
  ```

Related Information

do_route

do_reset_floorplan

```
do_reset_floorplan -rows -instances -pins
```

Can be issued after loading a design.

Options and Arguments

`-instances`

Specifies that all standard cells are to be unplaced. This command does not affect the placement of hard blocks or pad cells.

`-pins`

Resets the pin location information of all pins in the design.

`-rows`

Removes all user rows in the database and uses the floorplan parameters to generate cell rows.

Example

```
do_reset_floorplan -rows -instances
```

Related Information

do_initialize_floorplan

report_floorplan_parameters

set_floorplan_parameters

do_route

```
do_route [-assign_routing_direction {true | false}]
     [-def_name list_of_file_names] [-def_out_name filename]
     [-discourage_planar_routing {true | false}] [-follow_pins]
     [-force {true | false}] [-ggrid_max_size 16] [-ggrid_min_size 5]
     [-ggrid_mode {user | align | uniform}] [-ggrid_opt_size 10]
     [-groute_num_opt_pass num] [-ignore_unplaced_component {true | false}]
     [-no_follow_pins] [-output_db_name filename]
     [-output_full_def {true | false}] [-pin_access_compat_mode {true | false}]
     [-region_relax_percent] [-route_area {lx|ly|ux|uy}]
     [-route_bottom_layer_limit value] [-route_on_grid_only {true | false}]
     [-route_pin_access_compat_mode {auto | normal | offGrid | upViaEnclosure |
     upVia | upViaReserve | narrowPin}] [-route_select_net [~] list_of_net_names]
     [-route_select_net_first {true | false}]
     [-route_select_net_layer_range {layer:layer | layer}]
     [-route_soft_top_layer_limit {true | false}]
     [-route_stripe_center_connect {true | false}]
     [-route_stripe_follow_pin_only {true | false}]
     [-route_top_layer_limit value]
     [-route_use_range_rule obstruction | pin | obstruction+pin]
     [-straighten_net net_name_list] [-timing_driven {true | false}]
     [-timing_net_percentage_delta [0 to 100]] [-use_auto_ggrid {true | false}]
     [-use_max_xy_spacing {true | false}] [-version]
```

Starts Wroute to do a global routing on the design and saves the global routing information in the Wroute database.

Options and Arguments

`-assign_routing_direction {true | false}`

Tells the router, when `true`, to automatically assign routing directions in the channels between blocks to take advantage of available routing tracks. This variable is useful, for example, if channel congestion between blocks occurs because the preferred routing tracks are perpendicular to the orientation of the channel. In this case, this variable lets the router reassign the preferred routing direction in the channel.
Default: `false`

`-def_name` *`list_of_file_names`*

Specifies the name of the DEF file(s) that Wroute reads.

`-def_out_name` *`filename`*

Specifies the name of the output DEF file that Wroute writes to.

Note: This option is primarily used to change the output DEF *`filename`*. If this option is not used, the DEF is automatically written to the following default DEF filename: `wdb_name.def`.

`-discourage_planar_routing {true | false}`

Eliminates the use of planar routing over blocks
Default: `false`

`-follow_pins`

Removes existing power rails (if applicable) and generates new ones.

`-force {true | false}`

Stops global routing, if `false`, when more than 90% of the nets are final routed. If true, will continue global routing while preserving all final routes. Note that global routing will continue and also issue a warning when < 90% of the design is final routed.

`-ggrid_max_size 16`

Sets the maximum gcell grid size in tracks. *`ggrid`* size has a significant effect on QOR (quality of results). It is recommended you use a smaller *`ggrid`* size unless your design has a large number of obstacles or is memory constrained. See `-ggrid_min_size` and `-ggrid_opt_size` for more options.

`-ggrid_min_size 5`

Sets the minimum gcell grid size in tracks. *`ggrid`* size has a significant effect on QOR (quality of results). It is recommended you use this *`ggrid`* size unless your design has a large number of obstacles or is memory constrained. See `-ggrid_max_size` and `-ggrid_opt_size` for more options.

`-ggrid_mode {align | uniform | user}`

Sets the global routing grid generation mode. The global routing grid partitions the routing portion of a standard cell design into rectangles called gcells. The arguments for this variable set the grid as follows:

align
Generates a global routing grid that aligns with the channels and macro block boundaries in your design. This argument improves routing quality for block-based designs because, by aligning the grids, the global router can route more effectively in empty gcells in the channel area.

uniform
Generates a uniform global routing grid.

user
Uses the global routing grid defined in the DEF file.

Default: `uniform`

`-ggrid_opt_size 10`
Sets the optimal gcell grid size in tracks. *`ggrid`* size has a significant effect on QOR (quality of results). It is recommended you use a smaller *`ggrid`* size unless your design has a large number of obstacles or is memory constrained. See `-ggrid_max_size` and `-ggrid_min_size` for more options.

`-groute_num_opt_pass` *`num`*
Sets the number of optimization passes in groute, specified by the value *`num`*. If no value is given for *`num`*, the default value is used.
Default: Auto tuned.

`-ignore_unplaced_component {true | false}`
Specifies not to route to the pins of unplaced components.
Default: `false`

`-no_follow_pins`
Removes existing power rails (if applicable).

`-output_db_name` *`filename`*
Specifies the router's output database. The database is in binary form. To save your design into a router database without running routing, specify the database *`filename`* with this option and set all the routing variables to `false`. The router will save your design into database format without making any routing passes.

Note: This option is primarily used to change the output

database *filename*. If this option is not used, the database is automatically written to the following default output database filename: design_name.wdb. This option is useful, for example, if you are routing several different floorplans or placements. You can use this option to save each database that the router creates to a different routing binary. Then you can load the best results from the saved database.

`-output_full_def {true | false}`
Outputs a full DEF file, when `true`, instead of a DEF file containing only placement information.
Default: `true`

`-pin_access_compat_mode {true | false}`
Enables pin access compatibility mode.
Default: `false`

`-region_relax_percent`
Controls group and region constraints. Constraints are ignored if the percent is negative. Takes values 0.0 - 1.0.
Default: `0`

`-route_area {lx ly ux uy}`
Routes the specified area. The coordinates for the area are in database units. If no arguments are given, the full chip is routed.

`-route_bottom_layer_limit [0]`
Limits the bottom routing layer to the specified value. A value of 0 disables this option. Routing below the specified layer is allowed, but at a very high cost. Thus, the wire length below the specified layer is kept short. For example, in a design with six layers, a value of 4 requests that the router start routing at the fourth layer.

`-route_on_grid_only {true | false}`
Routes to the on-grid geometries of port only.
Default: `false`

`-route_pin_access_compat_mode {auto | normal | offGrid | upViaEnclosure | upVia | upViaReserve | narrowPin}`
Sets the option for processing pin geometries. The arguments for this variable are:

auto
Processes the pin geometries as it did in previous versions of the software. This is the default value for this environment variable.

normal
Processes the pin geometries as it did in previous versions of the software. This is the default value for this environment variable.

offGrid
Shrinks the pin geometries by one-half the wire width. The reminder of the pin geometry is valid for the router to make a connection using a regular wire. The router checks the connection using a one-half wire width plus spacing rule.

upViaEnclosure
Shrinks the pin geometries by one-half a via width. The remainder of the pin geometries is valid for the router to make a connection using a via that is fully enclosed by the pin. The router checks the connection using a one-half via width plus spacing rule.

upVia
Does not shrink the pin, so the full size of the pin geometry is valid for the router to make a connection using a via. The router checks the connection using a one-half via width plus spacing rule.

upViaReserve
Shrinks the pin geometries by one-half the wire width. The remainder of the pin geometries is valid for making a connection using a via. The router checks the connection using a one-half via width plus spacing rule.

narrowPin
Does not shrink the pin geometries. The full size of the pin geometries is valid for making a connection using a regular wire. The router checks the connection using a one-half via width plus spacing rule.

`-route_select_net [~] list_of_net_names`
where ~ represents negation.
Using ~ routes all but the specified nets.
No ~ routes only the specified nets.

`-route_select_net_first {true | false}`
Routes the selected nets first, followed by all other nets.
Default: `false`

`-route_select_net_layer_range {layer:layer | layer}`
Routes the selected nets on the specified layer range.

`-route_soft_top_layer_limit {true | false}`
Specifies that the top routing limit is soft, if set to `true`.
Default: `false`

`-route_stripe_center_connect {true | false}`
Connects to the center of the stripe only. Does not connect to POWER and GROUND pins.
Default: `false`

`-route_stripe_follow_pin_only {true | false}`
Connects to the follow pin stripe only.
Default: `false`

`-route_top_layer_limit [value]`
Limits the top routing layer to the specified value. A value of 0 disables this argument. A value of n means the placer uses routing resources from layer `1` to layer `n`.
Default: `0`

`-route_use_range_rule (obstruction | pin | obstruction+pin)`
Specifies the clearance distance used for spacing rule checks using one or more of: pin, obstruction, or obstruction+pin.
Default: `obstruction+pin`

`-straighten_net net_name_list`
Straightens the selected nets specified by `net_name_list`.
Default: `[ ]`

`-timing_driven {true | false}`
Turns on timing-driven routing when set to true. To perform timing-driven routing, the router decides the order in which to route the constrained nets based on their timing criticality, and routes them accordingly. The most timing-critical nets are routed first to achieve the shortest routes.

The router modifies its routing priorities as appropriate during the routing process. For example, if the router places a net low on the priority list to start, but later realizes that the net must be routed more directly to meet timing constraints, it will reroute the net. The router also considers the layer RC values when determining how to route the timing-critical nets.
Default: `true`

`-timing_net_percentage_delta [0 to 100]`
Specifies the percentage change in the timing-critical nets.
Default: `0`

`-use_auto_ggrid {true | false}`
Uses Wroute's default auto tune settings, if set to `true`. If `false`, uses the PKS smaller ggrid size.
Default: `true`

`-use_max_xy_spacing {true | false}`
Uses the maximum values of `x` and `y` for spacing checks.
Default: `false`

`-version`
Checks the version number and build information for routing. This option only works after a gate-level netlist is loaded.

Example

```
do_route -timing_driven true -ggrid_mode align
```

do_wroute

```
do_wroute [-antenna_method] [-antenna_sum_gate_area] [-froute_allow_port_short]
     [-froute_antenna_cell_pass] [-froute_antenna_top_layer_limit integer_value]
     [-froute_auto_save_interval] [-froute_auto_stop]
     [-froute_fix_antenna_pass integer_value] [-froute_manufacturing_grid]
     [-froute_on_grid_only] [-froute_on_grid_via_threshold]
     [-froute_optimization] [-froute_pin_enclose_wire {true | false}]
     [-froute_prefer_on_grid] [-froute_runtime_limit value]
     [-froute_search_repair] [-froute_taper_distance]
     [-froute_taper_pin_selection {ruleBased | inputOnly | all}]
     [-froute_use_max_x_y_spacing]
     [-froute_use_range_rule {pin | obstruction | pin+obstruction}]
     [-froute_xtalk_threshold integer_value]
     [-groute_discourage_planar_routing {true | false}] [-groute_ggrid_mode]
     [-groute_incremental] [-include filename ...] [-input_db_name filename]
     [-input_def_name {filename | "list_of_files"}] [-input_eco {true | false}]
     [-input_gcf_constraints {filename | "list_of_files"}]
     [-input_gcf_timing_libraries {filename | "list_of_files"}]
     [-input_ldef_comment_char] [-input_lef_name {filename | "list_of_files"}]
     [-log_file filename] [-no_license_sharing] [-output_db_name filename]
     [-output_def_name filename] [-processor] [-route_area]
     [-route_bottom_layer_limit] [-route_chip_assembly_mode {true | false}]
     [-route_final] [-route_global] [-route_modify_preroute_pass]
     [-route_select_net {netname | "list_of_nets"}]
     [-route_select_net_first {true | false}]
     [-route_select_net_layer_range {bottom_layer | "bottom_layer:top_layer"}]
     [-route_soft_top_layer_limit {true | false}] [-route_straighten_net [~]
     {netName | "list_of_files" | < filename | "fileName"}]
     [-route_stripe_center_connect {true | false}]
     [-route_top_layer_limit integer_value] [-timing_driven_routing]
     [-timing_fall_transition] [-timing_net_percentage_delta]
     [-timing_rise_transition] [-timing_use_est_cap]
     [-xtalk_fix_net {netName | "list_of_files" | < filename}]
     [-xtalk_rule "class_a [|class_b] ... + class_a [|class_b] ... [,class_a
     [|class_b] ... + class_a [|class_b] ...] ..."]
     [-xtalk_rule_threshold threshold]
```

Starts Wroute to do a final routing on the design. Wroute will not generate its own log file. All log messages will be channeled into `pks_shell` output and also saved in `pks_shell.log` or the file you specify with the `-log_file` option.

Important

You must specify one of the following options, otherwise the `do_wroute_eco` command will error out:

- `-input_db_name`
- `-input_def_name`

The do_wroute command automatically calls read_wdb after finishing routing. The read_wdb messages are written to the log file as a Groute call from ULTRA PLACER.

The wroute executable is typically found on the UNIX search PATH or can be set using the following global command:

```
set_global pks_wroute_exe path/wroute
```

If the pks_wroute_exe global is not set or set to a null string, the following message appears:

```
"wroute: on the search PATH will be used."
```

If the wroute executable is not on the UNIX search PATH, the command will error out.

The do_wroute command will signal a warning if the do_route command version and wroute command version are not a known match. A similar warning will be issued if the WDB version written by the do_route command does not match with the WDB version expected by the wroute command. These are for error checking and do not indicate actual mismatches. You are responsible for making sure that the WDB file you pass to the do_wroute command is compatible (do_route version 72 and wroute version 35 is a known match).

The do_wroute command supports all Wroute options with a changed name style. The rule of thumb is: all letters in the do_wroute option name appear as they do in the Wroute option name but in lower case. Precede all capital letters in the original name with a '_' in the new name. For example:

Wroute Option Name	do_wroute Option Name
routeSelectNet	-route_select_net
outputDbName	-output_db_name
AntennaMethod	-antenna_method

All option values are the same as they appear in Wroute. For example:

Wroute Option Value	do_wroute Option Value
TRUE or true	TRUE or true
FALSE or false	FALSE or false
0.25	0.25

Wroute Option Value	do_wroute Option Value
"clk*,net1,net2"	"clk*,net1,net2"
"file1 file2 ..."	"file1 file2 ..."
obstruction+pin	obstruction+pin

Currently, this is the only mechanism to run Wroute through PKS. Until REF files are supported, PKS can not transfer global routing automatically through pks_shell to Wroute.

Options and Arguments

For more details, refer to the *Ultra Router Reference* available with the Silicon Ensemble documentation set.

-antenna_method
: Maps to the Silicon Ensemble Verify.Antenna.Method environment variable.

-antenna_sum_gate_area
: Maps to the Silicon Ensemble Verify.Antenna.SumGateArea environment variable.

-froute_allow_port_short
: Maps to the Silicon Ensemble WRoute.Allow.Port.Shorts environment variable.

-froute_antenna_cell_pass
: Maps to the Silicon Ensemble WRoute.Antenna.Cell.Pass environment variable.

-froute_antenna_top_layer_limit *integer_value*
: Starts adding antenna diode cells if process antenna violations remain after layer hopping reaches the specified layer.

-froute_auto_save_interval
: Maps to the Silicon Ensemble WRoute.AutoSave.Interval environment variable.

-froute_auto_stop
: Maps to the Silicon Ensemble WRoute.Auto.Stop environment variable.

`-froute_fix_antenna_pass` *`integer`*
Maps to the Silicon Ensemble `WRoute.Fix.AntennaPass` environment variable.

`-froute_manufacturing_grid`
Maps to the Silicon Ensemble `FRoute.Manufacturing.XGrid` environment variable.

`-froute_on_grid_only`
Maps to the Silicon Ensemble `Wroute.OnGrid.Via.Only` environment variable.

`-froute_on_grid_via_threshold`
Maps to the Silicon Ensemble `WRoute.OnGrid.Via.Threshold` environment variable.

`-froute_optimization`
Maps to the Silicon Ensemble `WRoute.Optimization` environment variable.

`-froute_pin_enclose_wire {true | false}`
Specifies whether to force the router to connect to a pin if it encloses a wire or connect to the center of a wire if it is wider than the pin.

`-froute_prefer_on_grid`
Maps to the Silicon Ensemble `WRoute.Prefer.OnGrid` environment variable.

`-froute_runtime_limit` *`value`*
Maps to the Silicon Ensemble `WRoute.RunTimeLimit` environment variable. Units in seconds.

`-froute_search_repair`
Maps to the Silicon Ensemble `WRoute.SearchRepair` environment variable.

`-froute_taper_distance`
Maps to the Silicon Ensemble `WRoute.Taper.Distance` environment variable.

`-froute_taper_pin_selection {ruleBased | inputOnly | all}`
Specifies which wires the router tapers.

`-froute_use_max_x_y_spacing`
Maps to the Silicon Ensemble `Rule.ClearanceMeasure` environment variable.

`-froute_use_range_rule {pin | obstruction | pin+obstruction}`
Lets you use the spacing rule that is defined in the LEF file.

`-froute_xtalk_threshold` *`integer_value`*
Specifies the maximum length for parallel wire segments. If the router finds parallel wire segments that exceed the specified length, it attempts to space out the wires by one additional track within a global routing cell (gcell) to avoid crosstalk. If the gcell does not have enough room, the router completes the routing without the additional spacing.

`-groute_discourage_planar_routing {true | false}`
Specifies whether to penalize routing over blocks when only one routing layer is available. Planar routing (routing on one layer) increases the possibility that wires will cross.

`-groute_ggrid_mode`
Maps to the Silicon Ensemble `WRoute.GGridMode` environment variable.

`-groute_incremental`
Maps to the Silicon Ensemble `WRoute.Incremental.Global` environment variable.

`-include` *`filename ...`*
Lets you use secondary configuration files for the router, so you can include options in additional files. You can have up to ten nesting levels of options.

`-input_db_name` *`filename`*
Specifies the database directory. Maps to the Silicon Ensemble `WRoute.Input.DbName` environment variable.

`-input_def_name {`*`fileName`*`|"`*`list_of_files`*`"}`
Specifies the name of the DEF file to be input to the router.

`-input_eco {true | false}`
Makes the router more error tolerant by allowing it to delete problematic nets and rerouting the affected nets. Examples of problematic nets are nets with loops and nets for which the router cannot extract connectivity.

`-input_gcf_constraints {`*`fileName`* `|` `"`*`list_of_files`*`"}`
Specifies the general constraint format (GCF) constraint file or files. You use GCF files to define system-level constraints (SLCs) and timing libraries. You can use the same GCF file for the constraints and the timing libraries.

`-input_gcf_timing_libraries {`*`fileName`* `|` `"`*`list_of_files`*`"}`
Specifies the GCF file or files containing the timing libraries. This option is required to run timing-driven routing.

`-input_ldef_comment_char`
Maps to the Silicon Ensemble `Input.Comment.Delimiter` environment variable.

`-input_lef_name {`*`fileName`* `|` `"`*`list_of_files`*`"}`
Specifies the name or names of the LEF files to be used. Maps to the Silicon Ensemble `WRoute.Addition.LefName` environment variable.

`-log_file` *`filename`*
Specifies the name to use for the Wroute log file.

`-no_license_sharing`
Runs Wroute version 34 or older, which does not take license token from PKS and will checkout its own license.

`-output_db_name` *`filename`*
Specifies the directory to use for the output database. Maps to the Silicon Ensemble `WRoute.Output.DbName` environment variable.

Note: If you specify this option with no filename, the tool will error out. If you omit this option, no database file will be written out.

`-output_def_name` *`filename`*

Specifies the name to use for the output DEF file. Maps to the Silicon Ensemble `WRoute.Output.DefName` environment variable.

Note: If you specify this option with no filename, the tool will error out. If you omit this option, no DEF file will be written out.

Note: By default, `do_wroute` writes the output file in the DEF version of the file you originally loaded, or DEF version 5.4, whichever is the newer version.

`-processor`

Maps to the Silicon Ensemble `WRoute.Processor` environment variable.

`-route_area`

Maps to the Silicon Ensemble `WRoute.Route.Area` environment variable.

`-route_bottom_layer_limit`

Maps to the Silicon Ensemble `WRoute.BottomLayer.Limit` environment variable.

`-route_chip_assembly_mode {true | false}`

Specifies whether to scan the design to determine the necessity for chip-assembly routing and changes heuristics during detailed routing, depending on the percentage of the area of the design that is covered by blocks, rings, or pads. In chip-assembly mode, the router runs an order of magnitude faster than in no-chip-assembly mode at the expense of slightly degraded routing quality.

`-route_final`

Maps to the Silicon Ensemble `Wroute.Final` environment variable.

`-route_final (TRUE)`

Maps to the Silicon Ensemble `WRoute.Incremental.Final` environment variable.

`-route_global`
Maps to the Silicon Ensemble `WRoute.Global` environment variable.

`-route_modify_preroute_pass`
Maps to the Silicon Ensemble `WRoute.Modify.PreroutePass` environment variable.

`-route_select_net {`*`netname`* `|` `"`*`list_of_nets`*`"}`
Maps to the Silicon Ensemble `WRoute.SelectNets` environment variable.

`-route_select_net_first {true | false}`
Maps to the Silicon Ensemble `WRoute.SelectNet.First` environment variable.

`-route_select_net_layer_range`
`{`*`bottom_layer`* `|` `"`*`bottom_layer`*`:`*`top_layer`*`"}`
Specifies a layer range for selected nets if all nets are being routed. You can specify a bottom layer for routing the selected nets or both a bottom and top layer.

`-route_soft_top_layer_limit {true | false}`
Specifies whether to allow routing on layers higher than the one set by `routeTopLayerLimit`.

`-route_straighten_net [~] {`*`netName`* `|` `"`*`list_of_nets`*`" | "<` *`fileName`*`"|`
`" `*`fileName`*`"}`
Deletes specified nets in a fully routed design and reroutes them as straight as possible.

`-route_stripe_center_connect {true | false}`
Controls whether the router connects to the center of wide stripes only, not to power or ground pins.

`-route_top_layer_limit` *`integer_value`*
Maps to the Silicon Ensemble `WRoute.TopLayer.Limit` environment variable.

`-timing_driven_routing`
Maps to the Silicon Ensemble `WRoute.Timing.Driven` environment variable.

`-timing_fall_transition`
Maps to the Silicon Ensemble `WRoute.Timing.Fall.Transition` environment variable.

`-timing_net_percentage_delta`
Maps to the Silicon Ensemble `WRoute.Timing.Net.Percentage.Delta` environment variable.

`-timing_rise_transition`
Maps to the Silicon Ensemble `WRoute.Timing.Rise.Transition` environment variable.

`-timing_use_est_cap`
Maps to the Silicon Ensemble `WRoute.Timing.Use.Est.Cap` environment variable.

`-xtalk_fix_net {`*`netName`* `|` `"`*`list_of_nets`*`"` `|` `<` *`fileName`*`}`
Performs postroute crosstalk fixing on selected nets by pushing adjacent routing aside. Use this option on a fully routed database during final routing only.

`-xtalk_rule "`*`class_a`* `[|`*`class_b`*`] ... +` *`class_a`* `[|`*`class_b`*`] ... [,`*`class_a`* `[|`*`class_b`*`] ... +` *`class_a`* `[|`*`class_b`*`] ...] ..."`
Specifies the interaction rules for crosstalk classes. The default crosstalk class for all nets is `0`. Cadence recommends you use `xtalkRule "0+0"` if you need to minimize crosstalk.
Default: `false` (Crosstalk class for all nets.)

`-xtalk_rule_threshold` *`threshold`*
Specifies the wire-length threshold for nets to be affected by the class interaction rule for crosstalk.

Examples

Refer to the *Ultra Router Reference* available with the Silicon Ensemble documentation set.

Additional Information

read_wdb

write_wdb

do_wroute_eco

```
do_wroute_eco -input_db_name filename [-antenna_method] [-antenna_sum_gate_area]
     [-froute_allow_port_short] [-froute_antenna_cell_pass]
     [-froute_antenna_top_layer_limit integer_value]
     [-froute_auto_save_interval] [-froute_auto_stop] [-froute_fix_antenna_pass]
     [-froute_manufacturing_grid] [-froute_on_grid_only]
     [-froute_on_grid_via_threshold] [-froute_optimization]
     [-froute_pin_enclose_wire {true | false}] [-froute_prefer_on_grid]
     [-froute_runtime_limit] [-froute_search_repair] [-froute_taper_distance]
     [-froute_taper_pin_selection {ruleBased | inputOnly | all}]
     [-froute_use_max_x_y_spacing]
     [-froute_use_range_rule {pin | obstruction | pin+obstruction}]
     [-froute_xtalk_threshold integer_value]
     [-groute_discourage_planar_routing {TRUE | FALSE}] [-groute_ggrid_mode]
     [-groute_incremental] [-include filename ...]
     [-input_def_name {filename | "list_of_files"}] [-input_eco {TRUE | FALSE}]
     [-input_gcf_constraints {filename | "list_of_files"}]
     [-input_gcf_timing_libraries {filename | "list_of_files"}]
     [-input_ldef_comment_char] [-input_lef_name] [-log_file filename]
     [-no_license_sharing] [-output_db_name filename]
     [-output_def_name filename ] [-processor] [-route_area]
     [-route_bottom_layer_limit] [-route_chip_assembly_mode {TRUE | FALSE}]
     [-route_final] [-route_global] [-route_modify_preroute_pass]
     [-route_select_net] [-route_select_net_first]
     [-route_select_net_layer_range] [-route_soft_top_layer_limit {TRUE | FALSE}]
     [-route_straighten_net [~] {netName | "list_of_files" | "< filename"|
    " fileName"}] [-route_stripe_center_connect {TRUE | FALSE}]
     [-route_top_layer_limit] [-timing_driven_routing] [-timing_fall_transition]
     [-timing_net_percentage_delta] [-timing_rise_transition]
     [-timing_use_est_cap]
     [-xtalk_fix_net {netName | "list_of_files" | < filename}]
     [-xtalk_rule "class_a [|class_b] ... + class_a [|class_b] ... [,class_a
     [|class_b] ... + class_a [|class_b] ...] ..."]
     [-xtalk_rule_threshold threshold]
```

Makes the router more error tolerant by allowing it to delete problematic nets and reroute the affected nets. Examples of problematic nets are nets with loops and nets for which the router cannot extract connectivity. Also runs global routing in incremental mode on an existing database. During incremental global routing, the router deletes and globally reroutes nets with violations. This command is similar to the following `do_wroute` command:

```
do_wroute -input_eco true -groute_incremental true
```

See do_wroute on page 535 for more information.

You must use the `-input_db_name` option to specify an input file, otherwise the `do_wroute` command will error out.

The do_wroute_eco command automatically calls read_wdb after finishing routing. The read_wdb messages are written to the log file as a Groute call from ULTRA PLACER.

The wroute executable is typically found on the UNIX search PATH, or can be set using the following global command:

```
set_global pks_wroute_exe path/wroute
```

If the pks wroute exe global is not set or set to a null string, the following message appears:

```
"wroute: on the search PATH will be used."
```

If the wroute executable is not on the UNIX search PATH, the command will error out.

Options and Arguments

For more details, refer to the *Ultra Router Reference* available with the Silicon Ensemble documentation set.

`-antenna_method`
: Maps to the Silicon Ensemble `Verify.Antenna.Method` environment variable.

`-antenna_sum_gate_area`
: Maps to the Silicon Ensemble `Verify.Antenna.SumGateArea` environment variable.

`-froute_allow_port_short`
: Maps to the Silicon Ensemble `WRoute.Allow.Port.Shorts` environment variable.

`-froute_antenna_cell_pass`
: Maps to the Silicon Ensemble `WRoute.Antenna.Cell.Pass` environment variable.

`-froute_antenna_top_layer_limit` *`integer`*
: Starts adding antenna diode cells if process antenna violations remain after layer hopping reaches the specified layer.

`-froute_auto_save_interval`
: Maps to the Silicon Ensemble `WRoute.AutoSave.Interval` environment variable.

`-froute_auto_stop`
Maps to the Silicon Ensemble `WRoute.Auto.Stop` environment variable.

`-froute_fix_antenna_pass` *`integer`*
`Maps to the Silicon Ensemble WRoute.Fix.AntennaPass environment variable.`

`-froute_manufacturing_grid`
Maps to the Silicon Ensemble `FRoute.Manufacturing.XGrid` environment variable.

`-froute_on_grid_only`
Maps to the Silicon Ensemble `Wroute.OnGrid.Via.Only` environment variable.

`-froute_on_grid_via_threshold`
Maps to the Silicon Ensemble `WRoute.OnGrid.Via.Threshold` environment variable.

`-froute_optimization`
Maps to the Silicon Ensemble `WRoute.Optimization` environment variable.

`-froute_pin_enclose_wire {true | false}`
Specifies whether to force the router to connect to a pin if it encloses a wire or connect to the center of a wire if it is wider than the pin.

`-froute_prefer_on_grid`
Maps to the Silicon Ensemble `WRoute.Prefer.OnGrid` environment variable.

`-froute_runtime_limit` *`value`*
Maps to the Silicon Ensemble `WRoute.RunTimeLimit` environment variable. Units are in seconds.

`-froute_search_repair`
Maps to the Silicon Ensemble `WRoute.SearchRepair` environment variable.

`-froute_taper_distance`
Maps to the Silicon Ensemble `WRoute.Taper.Distance` environment variable.

`-froute_taper_pin_selection {ruleBased | inputOnly | all}`
Specifies which wires the router tapers.

`-froute_use_max_x_y_spacing`
Maps to the Silicon Ensemble `Rule.ClearanceMeasure` environment variable.

`-froute_use_range_rule {pin | obstruction | pin+obstruction}`
Lets you use the spacing rule that is defined in the LEF file.

`-froute_xtalk_threshold` *`integer`*
Specifies the maximum length for parallel wire segments. If the router finds parallel wire segments that exceed the specified length, it attempts to space out the wires by one additional track within a global routing cell (gcell) to avoid crosstalk. If the gcell does not have enough room, the router completes the routing without the additional spacing.

`-groute_discourage_planar_routing {true | false}`
Specifies whether to penalize routing over blocks when only one routing layer is available. Planar routing (routing on one layer) increases the possibility that wires will cross.

`-groute_ggrid_mode`
Maps to the Silicon Ensemble `WRoute.GGridMode` environment variable.

`-groute_incremental`
Maps to the Silicon Ensemble `WRoute.Incremental.Global` environment variable.

`-include` *`filename`* `...`
Lets you use secondary configuration files for the router so you can include options in additional files. You can have up to ten nesting levels of options.

`-input_db_name` *`directory`*
: Specifies the database directory. Maps to the Silicon Ensemble `WRoute.Input.DbName` environment variable. This option is required.

`-input_def_name {`*`fileName`*`|"`*`list_of_files`*`"}`
: Specifies the name of the DEF file to be input to the router.

`-input_eco {true | false}`
: Makes the router more error tolerant by allowing it to delete problematic nets and rerouting the affected nets. Examples of problematic nets are nets with loops and nets for which the router cannot extract connectivity.

`-input_gcf_constraints {`*`fileName`* `| "`*`list_of_files`*`"}`
: Specifies the general constraint format (GCF) constraint file or files. Use GCF files to define system-level constraints (SLCs) and timing libraries. You can use the same GCF file for the constraints and the timing libraries.

`-input_gcf_timing_libraries {`*`fileName`* `| "`*`list_of_files`*`"}`
: Specifies the GCF file or files containing the timing libraries. This option is required to run timing-driven routing.

`-input_ldef_comment_char`
: Maps to the Silicon Ensemble `Input.Comment.Delimiter` environment variable.

`-input_lef_name {`*`filename`* `| "`*`list_of_files`*`"}`
: Maps to the Silicon Ensemble `WRoute.Addition.LefName` environment variable.

`-log_file` *`filename`*
: Dumps the Wroute log file.

`-no_license_sharing`
: Runs Wroute version 34 or older, which does not take license token from PKS and will checkout its own license.

`-output_db_name` *`filename`*

Specifies the directory to use for the output database. Maps to the Silicon Ensemble `WRoute.Output.DbName` environment variable.

Note: If you specify this option with no filename, the tool will error out. If you omit this option, no database file will be written out.

`-output_def_name` *`filename`*

Specifies the name to use for the output DEF file. Maps to the Silicon Ensemble `WRoute.Output.DefName` environment variable.

Note: If you specify this option with no filename, the tool will error out. If you omit this option, no DEF file will be written out.

Note: By default, `do_wroute` writes the output file in the DEF version of the file you originally loaded, or DEF version 5.4, whichever is the newer version.

`-processor`

Maps to the Silicon Ensemble `WRoute.Processor` environment variable.

`-route_area`

Maps to the Silicon Ensemble `WRoute.Route.Area` environment variable.

`-route_bottom_layer_limit`

Maps to the Silicon Ensemble `WRoute.BottomLayer.Limit` environment variable.

`-route_chip_assembly_mode {true | false}`

Specifies whether to scan the design to determine the necessity for chip-assembly routing and changes heuristics during detailed routing, depending on the percentage of the area of the design that is covered by blocks, rings, or pads. In chip-assembly mode, the router runs an order of magnitude faster than in no-chip-assembly mode at the expense of slightly degraded routing quality.

`-route_final`
Maps to the Silicon Ensemble `Wroute.Final` environment variable.

`-route_final`
Maps to the Silicon Ensemble `WRoute.Incremental.Final` environment variable.

`-route_global`
Maps to the Silicon Ensemble `WRoute.Incremental.Final` environment variable.

`-route_modify_preroute_pass`
Maps to the Silicon Ensemble `WRoute.Modify.PreroutePass` environment variable.

`-route_select_net {`*`netname | "list_of_nets"`*`}`
Maps to the Silicon Ensemble `WRoute.SelectNets` environment variable.

`-route_select_net_first {true | false}`
Maps to the Silicon Ensemble `WRoute.SelectNet.First` environment variable.

`-route_select_net_layer_range`
`{`*`bottom_layer`*` | "`*`bottom_layer`*`:`*`top_layer`*`"}`
Specifies a layer range for selected nets if all nets are being routed. You can specify a bottom layer for routing the selected nets or both a bottom and top layer.

`-route_soft_top_layer_limit {true | false}`
Specifies whether to allow routing on layers higher than the one set by `routeTopLayerLimit`.

`-route_straighten_net [~] {`*`netName`*` | "`*`list_of_nets`*`" | "< `*`fileName`*`"|`
`" `*`fileName`*`"}`
Deletes specified nets in a fully routed design and reroutes them as straight as possible.

`-route_stripe_center_connect {true | false}`
Controls whether the router connects to the center of wide stripes only, not to power or ground pins.

`-route_top_layer_limit` *`integer`*
Maps to the Silicon Ensemble `WRoute.TopLayer.Limit` environment variable.

`-timing_driven_routing`
Maps to the Silicon Ensemble `WRoute.Timing.Driven` environment variable.

`-timing_fall_transition`
Maps to the Silicon Ensemble `WRoute.Timing.Fall.Transition` environment variable.

`-timing_net_percentage_delta`
Maps to the Silicon Ensemble `WRoute.Timing.Net.Percentage.Delta` environment variable.

`-timing_rise_transition`
Maps to the Silicon Ensemble `WRoute.Timing.Rise.Transition` environment variable.

`-timing_use_est_cap`
Maps to the Silicon Ensemble `WRoute.Timing.Use.Est.Cap` environment variable.

`-xtalk_fix_net {`*`netName`* `|` `"`*`list_of_nets`*`"` `|` `<` *`fileName`*`}`
Performs postroute crosstalk fixing on selected nets by pushing adjacent routing aside. Use this option on a fully routed database during final routing only.

`-xtalk_rule "`*`class_a`* `[|`*`class_b`*`] ... +` *`class_a`* `[|`*`class_b`*`] ... [,`*`class_a`* `[|`*`class_b`*`] ... +` *`class_a`* `[|`*`class_b`*`] ...] ..."`
Specifies the interaction rules for crosstalk classes. Cadence recommends you use `xtalkRule "0+0"` if you need to minimize crosstalk.
Default: `false` (Crosstalk class for all nets.)

`-xtalk_rule_threshold` *`threshold`*
Specifies the wire-length threshold for nets to be affected by the class interaction rule for crosstalk.

Examples

Refer to the *Ultra Router Reference* available with the Silicon Ensemble documentation set.

Additional Information

read_wdb

write_wdb

do_xform_tcorr_eco

```
do_xform_tcorr_eco [-opt_passes num] [-fix_design_rule] [-reclaim_area]
```

Important

This command will be removed in the next full release of the software. Use the `do_optimize -legalize_only` option to legalize the placement of the design.

Performs one pass of placement spreading and QP-eco followed by a loop of `opt_passes` iterations of TC/QP-eco

where the TC command is `do_xform_timing_correction -quick -incremental -resize -dont_fix_design_rules -dont_reclaim_area`

This command is typically used as a cleanup step after obtaining the results from `do_optimize`. It results in a legal placement of the design.

Options and Arguments

`-fix_design_rule`
Turns on fixing design rules during timing correction.

`-opt_passes <num>`
Defines the number of optimization passes.
Default: `1`

`-reclaim_area`
Turns on reclaim area during timing correction.

Examples

- The following command performs only legalization on the design:

  ```
  do_xform_tcorr_eco -opt_passes 0
  ```

- The following command performs legalization and reoptimization including area reclaiming on the design:

  ```
  do_xform_tcorr_eco -opt_passes 1 -reclaim_area
  ```

generate_supply_rails_on_rows

```
generate_supply_rails_on_rows
```

Removes any existing supply rails on rows, then generates new rails over the rows.

Related Information

remove_supply_rails_on_rows

report_supply_rails_on_rows

set_supply_rails_on_rows

get_cluster

`get_cluster` *`instance_name_or_id`*

Returns the cluster immediately associated with the specified instance or ID.

Options and Arguments

`inst_name_or_id`
Specifies the name of the instance or ID.

Related Information

create_physical_cluster

remove_physical_cluster

get_cluster_contents

```
get_cluster_contents cluster_name [-hier] [-tcl_list]
     [-instances | -clusters | -physical_instances]
```

Returns the instances directly associated to a cluster or immediate subclusters.

Options and Arguments

cluster_name
: Specifies a hierarchical physical cluster name.

-clusters
: Lists the cluster names of all subclusters (recursively if the -hier option is specified). Note that it does not descend into soft_block subclusters.

-hier
: Lists the instances that are contained with children subclusters that are of type region or group (similarly for a list of subclusters).

-instances
: Lists all instances that are directly associated with that cluster and with clusters below if -hier is specified. Note that it does not descend into soft_block subclusters.

-tcl_list
: Returns the report as a list.

Examples

- The following command provides a list of the immediate subclusters of the root cluster:

```
get_cluster_contents / -clusters
```

- The following command provides a list of all clusters defined in a design:

```
Proc getAllClusters {cluster_name} {//pseudo-code only. Not strictly legal Tcl
Set cc [get_cluster_contents cluster_name -hier -clusters]
If [get_cluster_physical_info $cc .eq. soft_block]
GetAllClusters $cc
Puts "$cc"
}
getAllClusters /
```

Related Information

get_cluster_physical_info

get_cluster_names

```
get_cluster_names [-children] [-root]
```

Returns the name of either the root cluster, the children, or both. This command works with the current top timing module.

Options and Arguments

`-children`
: Specifies the name of the cluster's children.

`-root`
: Specifies the name of the root cluster.

Related Information

`set_top_timing_module`

get_cluster_physical_info

```
get_cluster_physical_info cluster_name [-tcl_list] [-soft_block]
     [-associated_instances] [-placed] [-unplaced_top] [-origin] [-bbox]
     [-utilization] [-max_utilization] [-row_utilization] [-max_half_perim]
     [-max_x_perim] [-max_y_perim] [-allow_place_overlap] [-allow_route_overlap]
```

Returns a Tcl list of known attributes on the cluster, such as type, location, and so on.

Options and Arguments

cluster_name

Specifies the name of the cluster you want information about.

Example

```
get_cluster location
```

Related Information

get_cluster

get_current_cluster

```
get_current_cluster
```

Lets you locate where you are in the cluster hierarchy. When traversing through a hierarchy you may forget where you are, this command returns the name of the current cluster.

get_current_congestion

```
get_current_congestion
```

Returns a measure of the congestion in the design (expressed as a percentage) for horizontal and vertical routes. The first number reports congestion in horizontal routes. The second number reports congestion in vertical routes.

The reported numbers represent the average number of excess routing tracks required per bin, expressed as a percentage of bin routing supply. Mathematically, the values reported are as follows:

```
congestion/supply * 100
```

where `congestion` equals the total number of all tracks in excess of supply in congested bins, and `supply` equals the total number of tracks available in all bins.

The calculations for horizontal and vertical routing are done separately.

get_current_utilization

```
get_current_utilization
```

Returns the current standard cell row utilization of a design by dividing the current total standard cell area by the effective sites in the design computed after accounting for all.

get_ground_net

`get_ground_net`

Returns the name of the ground net that was set using the `set_ground_net` command.

Related Information

get_power_net

set_ground_net

set_power_net

get_library_layer_offset

`get_library_layer_offset` *`layername`*

Returns the offset value for a given routing layer (specified in the physical library). Only routing layers may have offsets, so the layername is the name of a routing layer. The offset value is in user units (similar to LEF data).

Note: The offset value is retrieved from the database that stores physical library data.

Options and Arguments

`layername`

Specifies the name of the routing layer.

Related Information

set_library_layer_offset

get_logic_0_net

`get_logic_0_net`

Gets the name of the tie low net used in `write_def` to dump the connectivity associated with the tie lows.

Related Information

get_logic_1_net

get_logic_1_net

`get_logic_1_net`

Gets the name of the tie high net used in `write_def` to dump the connectivity associated with the tie high.

Related Information

get_logic_0_net

get_min_porosity_for_over_block_routing

`get_min_porosity_for_over_block_routing`

Returns the current value set by `set_min_porosity_for_over_block_routing`

Related Information

set_min_porosity_for_over_block_routing

get_min_wire_length

`get_min_wire_length`

Gets the current value of the minimum wire length variable.

Related Information

set_min_wire_length

get_physical_info

```
get_physical_info [{-instance | -pin | -net} {object_name | FNPid}] [-bbox]
     [-center] [-dbunits] [-def_name] [-def_properties] [-direction] [-halo]
     [-hperimeter] [-ipin_name name] [-layer_usage_table] [-lef_multiplier]
     [-length] [-location] [-ndr_rule] [-orientation] [-direction] [-type] [-size]
     [-state] [-table] [-use name] [-vias]
```

Queries the physical information for a specified object.

Note: If no arguments are specified, all information is returned for the specified object.

All outputs of get_physical_info are in HêP units, not database units.

Options and Arguments

`-bbox`
: Returns the specified object's bounding box.

`-center`
: Returns the specified object's center (*x*, *y* coordinates). Note that the `-net` argument is not available with this argument.

`-dbunits`
: Specifies that the output be in database units.

`-def_name`
: Returns the specified object's DEF name as it would appear in the DEF file. One application would be to associate a net appearing in a timing critical path with a net name in the DEF file. The returned name conforms to DEF syntax with respect to escaping rules.

`-def_properties`
: Returns all the DEF properties of the pin, instance, or net.

`-direction`
: Returns the direction (input, output, feedthrough, or none) of the signal flow through a physical pin. Note that this argument is used only with the `-pin` argument.

`-halo`
: Returns the left, right, bottom, and top halos of the physical.

instance. Note that this argument is used only with the `-instance` argument.

`-hperimeter`

Returns the half perimeter of the specified physical net.

`{-instance | -pin | -net} {object_name | FNPid}`

Identifies the physical instance, physical pin, or physical net specified by object_name. FNPid specifies logical objects. `object_name` is interpreted non-hierarchically in the current cluster context.

`-ipin_name` *`name`*

Identifies an instance pin specified by name. Use this argument in conjunction with the `[-instance object name | FNPid]` argument.

`-layer_usage_table`

Returns the layer usage table information for the specified physical net.

`-lef_multiplier`

Returns the LEF multiplier information for the specified physical net.

`-length`

Returns the wire length of the specified physical net. If routed, it will return the length based on the route.

`-location` *`loc`*

Returns the location of a pin, instance, or instance pin.

`-ndr_rule`

Returns the NDR rules information for the specified physical net.

`-orientation`

Returns the specified object's orientation in DEF format (N, S, W, E, FW, FE, FS, FN). Note that the `-net` argument is not available with this argument.

`-size`

Returns the specified object's width and height (*x*, *y* coordinates). Note that `-net` is not available with this argument.

`-state`

Returns the specified object's placement state. Note that `-net` is not available with this argument.

`-table`

Outputs in ASCII table format.

`-type`

Returns the physical pin type (special route or not). Note that this argument is used only with the `-pin` argument.

`-use` *name*

Returns the DEF_USE for the pin or net (such as SIGNAL, CLOCK, and so on).

`-vias`

Returns the number of vias for the specified physical net. If the net is not routed, this number will come from the LUT and Steiner estimation. If it is routed, it will come from the actual vial count in the net.

Examples

- The following command returns the location of instance `I1`, if find is successful:

```
get_physical_info -instance [find -instance I1] -location {20 30}
```

- The following command returns the location of instance pin `I1/A`:

```
get_physical_info -instance [find -instance I1] -ipin_name A -location {25, 34}
```

- The following command returns the location of the FILLER1 instance if found in the current cluster context:

```
get_physical_info -instance FILLER1 -location {35, 40}
```

Related Information

set_physical_info

get_pin_location

`get_pin_location` *`pin_name`* `[-side] [-index]`

Gets the side, the index, or both assigned to the *`pin_name`*.

Options and Arguments

`-index`
: Displays the index of the pin.

`pin_name`
: Specifies the name of the pin whose side and index are displayed.

`-side`
: Displays the side assigned to the pin.

Examples

- The following command displays the side and the index assigned to the clk pin:

  ```
  get_pin_location clk
  ```

- The following command displays the side assigned to the clk pin:

  ```
  get_pin_location clk -side
  ```

Related Information

set_pin_location

get_power_net

```
get_power_net
```

Returns the name of the power net that was set using the `set_power_net` command.

Related Information

set_power_net

get_ground_net

set_ground_net

get_route_availability

`get_route_availability -layer` *`layername`*

Provides the raw routing layer percentage that is used by the congestion analysis engine after considering preroutes. The default value is 100% for each routing layer.

Options and Arguments

`layername`
Specifies the name of the routing layer.

Example

The following command returns the raw routing layer (`metal4`) percentage used by the congestion analysis engine after considering preroutes:

```
get_route_availability -layer metal4
```

Related Information

set_route_availability

get_special_netpins

```
get_special_netpins -type { power | ground }
```

Gets a list of the special pins in the current module that are of the type specified by the `-type` argument.

Options and Arguments

`-type {power | ground)`
: Determines the special pins type. The special pins type is defined as either power or ground.

Related Information

set_special_netpin

get_steiner_capacitance

`get_steiner_capacitance` *`list_of_net_name_or_id`* `[-min]`

Gets the total steiner wire capacitance of the specified nets. If more than one net is passed, the capacitances are returned as the sum of the capacitances of the individual nets. If a net is listed more than once, the capacitance is added for each occurrence of the net in the list.

Note: There are no `-horizontal` or `-vertical` arguments.

The resultant capacitance does not include pin capacitance, only wire capacitance. You can get the pin capacitance using the `get_timing` or `report_net` commands.

Options and Arguments

`list_of_net_name_or_id`
Specifies the net or list of nets or ids.

`-min`
Reports the minimum capacitance value.

Related Information

get_steiner_length

get_steiner_resistance

`get_timing`

`report_net`

get_steiner_channel_width

```
get_steiner_channel_width
```

Returns the current value (the default value is 0, which means `set_steiner_channel_width` is off).

Related Information

set_steiner_channel_width

get_steiner_length

`get_steiner_length [-horizontal|-vertical] list_of_nets`

Reports the sum of the steiner wire lengths (in microns) for all nets specified in *list_of_nets*.

Options and Arguments

`-horizontal`
: Reports the sum of the horizontal lengths (in microns) for the given net(s).

list_of_nets
: Specifies the name of the network using either the full hierarchical name of a net(s) or an ID (wildcards are supported).

`-vertical`
: Reports the sum of the vertical lengths (in microns) for the given net(s).

get_steiner_resistance

`get_steiner_resistance` *`list_of_net_name_or_id`* `[-min]`

Gets the total steiner wire resistance of the specified nets. If more than one net is passed, the resistances are returned as the sum of the resistances of the individual nets. If a net is listed more than once, the resistance is added for each occurrence of the net in the list.

Note: There are no `-horizontal` or `-vertical` arguments.

Options and Arguments

`list_of_net_name_or_id`
Specifies the net or list of nets or IDs.

`-min`
Reports the minimum resistance value.

Related Information

get_steiner_capacitance

get_steiner_length

`get_timing`

`report net`

modify_physical_cluster

```
modify_physical_cluster -name cluster_name
     [-add_instance list_of_inst_names_or_ids]
     [-allow_place_overlap {true | false}] [-allow_route_overlap {true | false}]
     {[-bbox {lx ly ux uy}] | [-utilization float] [-aspect_ratio float] |
     [-width float] [-height float]} [-location {lx ly}]
     [-remove_instance list_of_names_or_ids] [-type {soft_block | region}]
     [-unplaced {true | false}]
```

Modifies an existing physical cluster created by the `create_physical_cluster` command. Note that this command can only be used on an existing cluster and invalidates appropriate blockage maps and timing.

Options and Arguments

`-add_instance` *`list_of_inst_names_or_ids`*
: Adds instances to the physical cluster. The command will error if the addition of instances invalidates the soft cluster containment rules (`containingSOFTCLUSTER`). If the addition of instances is successful, and the cluster does not have a `-bbox` specification and is unplaced, the estimated `-bbox` is modified to reflect the new contained cell area.

`-allow_place_overlap {true | false}`
: Allows the parent cluster to place over any placeable areas in the physical cluster. Note that it does not place over any overlap layer obstacles created on the cluster. If `false` is selected (default), the cluster is completely blocked for all placement at the parent level, even if overlap blockages are defined.

`-allow_route_overlap {true | false}`
: Allows the parent cluster to route over any routable areas in the physical cluster. Note that it does not route over any overlap layer obstacles created on the cluster. If `false` is selected (default), the cluster is completely blocked for all routing at the parent level, even if overlap blockages are defined.

`-bbox {`*`lx ly ux uy`*`} | -utilization` *`float`* `-aspect_ratio` *`float`* `| -width` *`float`* `-height` *`float`*
: Use one of three methods (below) to modify the shape of the physical cluster.

`-bbox {`*`lx ly ux uy`*`}`
Modifies the shape and location of the cluster using the four coordinates {*lx ly ux uy*}. All coordinates are assumed to be absolute. Using this argument will make the cluster PLACED. Note that you must type the curly braces in the syntax.

`-utilization` *`float`* `-aspect_ratio` *`float`*
Modifies the bounding box by specifying the `-utilization` (0-100) and `-aspect_ratio` (>0). For this to be useful, the cluster should also be associated with instances.

Note that the `-utilization` percentage is the *target* utilization for the cluster. Whenever the utilization percentage is reset this way, a new bounding box will be derived based on the current logical cell area.

The placement information (if applicable) of any contained instances will be maintained. The available row area is estimated based on row spacing rules of the containing soft cluster. No estimation blockages are considered.

`-width` *`float`* `-height` *`float`*
Modifies the width and height of the bounding box.

`-location {`*`lx ly`*`}`
Specifies the location of the lower-left corner of the cluster using the *lx* and *ly* coordinates. Note that you must type the curly braces in the syntax. Using this argument will make the cluster PLACED.

`-name` *`cluster_name`*
Specifies the name of the subcluster at the current cluster level.

`-remove_instance` *`list_of_names_or_ids`*
Removes instances from a physical cluster. The command will error if the removal of instances invalidates the soft cluster containment rules (containingSOFTCLUSTER). If the removal of instances is successful, and the cluster does not have a `-bbox` specification and is unplaced, the estimated `-bbox` is modified to reflect the new contained cell area.

`-type {soft_block | region}`
Defines the type of cluster you want to create. Note that if you

request a `-type soft_block` and more than one hierarchical instance is specified, the command will error out. If a cluster currently tagged as a `soft_block` cluster is changed to a `region` cluster (soft block), routing information may be lost. This occurs because routes are held on a per soft-cluster level basis; when a non-soft cluster becomes a soft-cluster, the routes are split into two components, resulting in the routes being destroyed.

`-unplaced {true | false}`

Changes the placement status of the cluster. When the cluster is changed from `PLACED` to `UNPLACED`, the location of the cluster and all of its contents will remain unchanged. If `true`, sets the `UNPLACEDTOP` variable of a currently placed cluster to itself (also sets any placed sub clusters to their appropriate values). In other words, the parent cluster rows can now be placed over the cluster even if it has been set to `-allow_place_overlap false`. If set to `false`, will make an unplaced cluster placed.

Examples

```
modify_physical_cluster -name a1 -type region
modify_physical_cluster -name A1 -allow_overlap true
```

Related Information

create_physical_cluster

pks_shell

```
pks_shell [-64] [[-f] filename] [-cmdfile command_filename]
     [-logfile log_filename] [-cdsdocd {on | off}] [-queue] [-expire] [-version]
     [-help] [-continue] [-no_init] [-set var=value ...] [-which] [-gui]
     [-display machine:0] [-geometry intxint+int+int] [-fullscreen] [-large]
     [-limit] [-colormap colormap_file]
```

Starts the Cadence Physically Knowledgeable Synthesis (PKS) tool. For more detail on starting `pks_shell`, refer to *Before You Begin* in the *PKS User Guide*.

Options and Arguments

`-64`
: Starts a 64-bit `pks_shell` session. Default is 32 bit.

`-cdsdocd {on | off}`
: Enables or disables the use of browser-based documentation for `pks_shell help`. When set to `on`, the full documentation set is available from the *Help* button in the GUI. When set to `off`, only the syntax is displayed in the command line. The default is `on` when running in GUI mode and `off` when running in command line mode.

`-cmdfile command_filename`
: Specifies the name of the command file.
Default: `pks_shell.cmd`

`-colormap`
: Specifies the color map file.

`-continue`
: Specifies to not exit after an error in Tcl script file.

`-display machine:0`
: Specifies the system display to use.

`-expire`
: Displays the expiration date of the license.

`-f filename`
: Specifies the name of the Tcl script file to source at startup. You can use *filename* without specifying the `-f`.

`-fullscreen`

Uses entire screen for the `pks_shell`.

`-geometry` *`int`*`x`*`int`*`+`*`int`*`+`*`int`*

Sets the initial size and position of the GUI main screen window. Where *`width`* and *`height`* are in pixels. And *`xoff`* and *`yoff`* are the number of pixels from the corner; a negative value is measured from the bottom or right corners, and a positive value is measured from the top or left corners. No spaces are allowed between the values.

For example: `-geometry 800x400+10-30` means create a window 800 by 400 pixels with its left edge offset 10 pixels from the left edge of the screen and its bottom edge offset 30 pixels from the bottom of the screen

Valid only in GUI mode.

`-gui`

Starts the graphical user interface (GUI) mode

`-help`

Prints usage message for this command.

`-large`

Increases the memory limit above 2GB for the process running `pks_shell` if the memory is available.

`-limit`

Prints the current data size limit and memory allocation limit for the machine on which the software will run.

`-logfile` *`log_filename`*

Specifies the name of the log file.
Default: `pks_shell.log`

`-no_init`

Disables sourcing of `~/.ambit/.acshrc`

`-queue`

Waits for a license if none is available.

`-set var=`*`value`*
: Initializes a Tcl variable.

`-version`
: Displays version of this `pks_shell`.

`-which`
: Displays the full path name of the `pks_shell` executable.

Example

The following command starts the PKS graphical user interface and saves the session log in `cpu_design.log`:

```
pks_shell -gui -logfile cpu_design.log
```

Related Information

`check option`

prune_routes

`prune_routes [-keep_top` *`percent_value`*`]`

Removes the routes for a given percentage of nets.

Options and Arguments

`-keep_top` *`float`*

Sorts the nets in your design according to the difference between the maximum source-to-sink distance from Steiner routes and real routes. The routes are retained for the top *`float`* percent of nets (those with the largest difference). The routes for all of the remaining nets will be deleted. Pruning some of the nets' routes allows PKS to perform more optimization tasks on the retained routes.

The specified percentage value (*`float`*), can be between 0 and 100. The default is 0.

read_cap_table

`read_cap_table` *`filename`*

Replaces LEF capacitance values with values computed from the specified cap table file. Use this command after all LEF files have been read in (that is, after the `read_lef` and/or `read_lef_update` commands) but prior to any optimization.

LEF supports AreaCap and EdgeCap per unit length on a per layer basis. The cap table provides coupling and area cap for various spacings between same-layer routes. Based on this data, PKS recalibrates the LEF EdgeCap and AreaCap values assuming there is a specific probability (70%, 25%, or 5%) of coupling happening to a route one, two, or three tracks away. For layer widths not the same as those in the cap table, a linear interpolation is performed (especially for NonDefaultRule specifications).

A layer name can be either the same as that used in the LEF or in the format "Mn" or "mn", where n is an integer corresponding to the layer order number in the LEF.

A basic cap table consists of several sections of cap information. Each section is for one layer and one particular width and must have the following format:

```
<layer name>
width(um)  space(um) Ctot(Ff/um)  Cc(Ff/um)    Carea(Ff/um) Cfrg(Ff/um)
...
...
data lines (one data line for one space)
...
...
```

The table can contain multiple sections (different widths) for the same layer, but for the table to be usable, there must be at least one section for each routing layer.

Important

The basic section of the cap table must start with a line containing just the keyword `BEGIN_BASIC` and end with a line containing just the keyword `END_BASIC`. Everything before `BEGIN_BASIC` and after `END_BASIC` will be ignored.

Options and Arguments

`filename`

Specifies the name of the cap table file.

Related Information

read_lef

read_lef_update

read_cap_table_update

read_cap_table_update

```
read_cap_table_update
```

Revaluates edge capacitance and reset layer capacitance, and derives a new unit capacitance to be used in any subsequent RC extraction.

The new unit capacitance is computed after the routes have been read in. The distribution of routes on the layers is analyzed, and the distribution profile (the percentages of two adjacent routes at 1-pitch, 2-pitch, and 3-pitch spacings) is derived using a bin map method. An average profile per layer is then derived from the bin values. This average profile is used together with a capacitance table to revaluate edge capacitance and reset layer capacitance per unit length.

Related Information

read_lef

read_lef_update

read_cap_table

read_change_file

`read_change_file` *`change_file`*

Reads antenna change information from the *`change_file`* into the `pks_shell`. These are netlist changes that result from inserting antenna diodes into the netlist during detail routing.

Options and Arguments

`change_file`

Specifies the change file that contains the antenna change information.

read_def

```
read_def [-do_not_strip_top_level_delimiter] [-hier_delimiter char]
     [-ignore_dist_cells] [-ignore_pwrgnd_pads] [-no_region]
     [-ignore_cover] {-scan_only | -ignore_scan} filename
```

Reads a Design Exchange Format (DEF) file and reads in cell placement and IO placement information, as well as final routes from the DEF. All other information in the DEF file is ignored.

Note: The database is modified when a DEF file is loaded.

The following applies when `read_def` is used with a Scan command:

For a chain containing flip-flops driven by an internal clock domain, the tool will first try to match a `scan_data_port` command with `-clk` argument that matches the internal clock domain pin. If no match is found, then a second match will be made with the “root” clock pin (top level clock pin driving the internal clock signal).

The DEF reader must be enhanced to parse and optionally store the scanDEF section contained within a full DEF file. The default behavior is to both parse and store the scanDEF section, unless the `read_def` command was issued with the `-ignore_scan` argument. In this situation, the scanDEF section will be parsed by the reader but not stored to a data structure.

Additionally, the DEF reader must also be enhanced to read a standalone scanDEF file. For example:

```
read_verilog structuralNetlist
do_build_generic
read_def full.def
read_def -ignore_scan full.def
read_def -scan_only scan.def
```

The scan chain configuration parsed and stored from the scanDEF file, will take precedence over any BuildGates test synthesis specific configuration commands including:

```
set_number_of_scan_chains
set_max_scan_chain_length
set_dft_compatible_clock_domain
set_dont_touch_scan
```

In the event that any of the above commands have been entered into the shell prior to or after creating the scanDEF data structure, a WARNING message will be returned to the user. For example:

Example 1:

```
set_number_of_scan_chains 2
read_def -scan_only scanDEF.def
```

```
WARNING  Scan chain configuration established through scanDEF format. ... Ignoring
command:  set_number_of_scan_chains
```

Example 2:

```
read_def -scan_only scanDEF.def
set_number_of_scan_chains 2


WARNING  Scan chain configuration established through scanDEF format.... Ignoring
command:  set_number_of_scan_chains
```

Options and Arguments

`-do_not_strip_top_level_delimiter`
: Forces the DEF reader to ignore the top-level delimiter character (if it exists in the input DEF) in situations where the Verilog is a flat Verilog and has the delimiter as a part of the name.

  Example:
  The flat Verilog name is: `/a/b/c`
  The DEF name is: `/a/b/c`

  In the example above, the Verilog name is a flat name; therefore `read_def` should not strip the first `/` character. By default, `read_def` strips the top-level delimiter to find the instance internal to the PKS database.

`filename`
: Specifies the name of the file to be read.

`-hier_delimiter` *`char`*
: Specifies the delimiter character to be used with instance and pin names. The default value is `/`.

`-ignore_cover`
: Forces the `read_def` to ignore cover macros.

`-ignore_dist_cells`
: Forces the `read_def` to ignore filler cells when reading the input def file.

`-ignore_pwrgnd_pads`
: Forces the `read_def` to ignore physical only pads, such as power and ground pads.

`-ignore_scan`	Causes the scan chains section to be excluded from the DEF output.
`-no_region`	Forces `read_def` to Ignore all region-related information (region definition and region reference) in the DEF file.
`-scan_only`	Excludes all `write_def` outputs except the scan chains section.

Examples

- The following command reads design cell placement information and also reads the IOs from the DEF file (`DesignFile.def`):

  ```
  read_def DesignFile.def
  ```

- The following command designates the `/` character as the hierarchy delimiter for instance and pin names in the DEF file (`temp.def`):

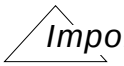
  ```
  read_def -hier_delimiter / temp.def
  ```

Related Information

write_def

read_gns

`read_gns [-hier] -gns_view` *`view_name`* `-gns_lib` *`lib_name`* `-gns_tech` *`tech_lib_name`*



The Genesis database is no longer supported. This command will be removed in a future release.

Updates the PKS physical data structure from a Genesis database (placement, floorplan, detailed routes, and global routes).

The read_gns command does not make any change to the Genesis database.

Changes to the PKS database depend on the presence or absence of the `-hier` argument:

- With the `-hier` argument, the entire hierarchy, with the exception of any hard blocks, is updated from the Genesis database.
- Without the `-hier` argument, only the current module is updated in the PKS database

Note: The `read_gns` command only updates a PKS database, it does not create one.

Options and Arguments

`-hier`
: Reads a design that contains a physical hierarchy.

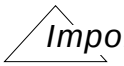
`-gns_lib` *`lib_name`*
: Specifies which Genesis library the command should use.

`-gns_tech` *`tech_lib_name`*
: Specifies which technology library the command should use.

`-gns_view` *`view_name`*
: Specifies which Genesis view the command should use.

read_layer_usages

```
read_layer_usages filename
```



This command performs operations similar to those found in other commands. If this command is used, the following commands are disabled and will have no effect:

- set_lef_multiplier
- set_layer_usages_table
- set_net_physical_attribute

Reads the layer usages table that contains wire length range definitions. See the *PKS User Guide* for more information about layer usages tables.

Using this command changes the default layer usages table (both clock and non-clock data) and changes all capacitance values.

Options and Arguments

filename
Specifies the name of the wire length usage table file.

Example

```
read_layer_usages usage.txt
```

Related Information

create_layer_usages_table

set_layer_usages_table

set_net_physical_attribute

write_layer_usages

read_lef

`read_lef [-comment_char` *`comment_char`*`] [-min` *`min_lef_filename`*`] `*`filename`*

Reads in the physical library data contained in the LEF file. The LEF file can be read only once during each `pks_shell` session, otherwise a warning is issued.

Options and Arguments

`-comment_char` *`comment_char`*

Comments out the text following the *`comment_char`* on the same line. You may comment out whole lines if needed. The default *`comment_char`* is #.

`filename`

Specifies the name of the LEF file.

`-min` *`min_lef_filename`*

Defines all sets of min RC values as well as any subsets of them. You can use this command argument as often as needed. Each subsequent command will append a new set of min RC values or will reset the min RC values.

Specifying this argument reads only the min LEF RC data. The RC data will be read only for those layers, vias, nondefault layers, or nondefault vias that have already been read by the `read_lef` or `read_lef_update` commands using the `-min` option. All other data from *`min_lef_filename`* will be ignored.

Note: The `read_lef_update -min` command can not be used in conjunction with the `set_net_physical_attribute` command.

Examples

- The following command reads the physical data from the `abc.lef` file:

  ```
  read_lef abc.lef
  ```

- The following command reads the physical data from the `amc.lef` file, designating the @ symbol as a comment character:

  ```
  read_lef amc.lef -comment_char @
  ```

Related Information

read_lef_update

read_lef_update

`read_lef_update [-comment_char` *`comment_char`*`] [-min` *`min_lef_filename`*`] `*`filename`*

Reads in the physical library data contained in the LEF file and adds this data to the data read in by the previous `read_lef` operation. This command can only be used after the `read_lef` command is issued.

Note: This command does not overwrite or update any library data read in by previous `read_lef` and `read_lef_update` operations. It only adds new data. If you wish to update capacitance values, see read_cap_table on page 587.

Options and Arguments

`-comment_char` *`comment_char`*

Comments out the text following the *`comment_char`* on the same line. You may comment out whole lines if needed. The default *`comment_char`* is #.

Note: The comment character used in this command may differ from the one used with the `read_lef` or other `read_lef_update` commands. The comment character should be implicitly defined in each `read_lef_update` command, otherwise the comment character defined in a previous `read_lef` or `read_lef_update` command is used.

`filename`

Specifies the name of the LEF file.

`-min` *`min_lef_filename`*

Defines or redefines all sets of min RC values as well as any subsets of them. You can use this command argument as often as needed. Each subsequent command will append a new set of min RC values or will reset the min RC values.

Specifying this argument reads only the min LEF RC data. The RC data will be read only for those layers, vias, nondefault layers, or nondefault vias that have already been read by the `read_lef` or `read_lef_update` commands. All other data from *`min_lef_filename`* will be ignored.

Note: Do not use the `read_lef_update -min` command after you have issued the `set_net_physical_attribute` command.

Example

The following command reads the physical data from the `amc_macro.lef` file and appends this data to the data read in by the previous `read_lef` operation. During the read operation all $ symbols are designated as comment characters:

```
read_lef_update amc_macro.lef -command_char $
```

Related Information

read_lef

read_pdef

```
read_pdef pdef_file_name [-flow] [-Release Status] [-pdef_version version_tag]
     [-use_name_keyword] [-ieee_location] [-print_all_messages] [-incremental]
      [-xoffset x_offset [-logical_prefix logical_prefix]
     [-instance instance_name_or_id]
```

Reads a PDEF file conforming to the V2.0 syntax. Additional constructs from the IEEE-P1481 syntax include:

- Support for RECT, ROW_ORIENT, ORIGIN, and CONTENTS_LOCATION attributes on clusters.
- Support for PIN construct and LOC attributes for pins.
- Support for WireLoad attributes.

Note: The database is modified when a PDEF file is loaded.

Options and Arguments

`-flow`

Maintains file integrity between the ADB and PDEF files. Use this argument after loading an ADB file. If the PDEF file contains CELL information, this information must correspond to instances in the ADB file at the time the command is issued. For example, if you run the optimize command first, it may change the netlist and therefore potentially invalidate the CELL information.

`-ieee_location`

Uses the IEEE 1481-1999 (also referred to as IEEE or 3.0) location definition instead of the PDEFVERSION 2.0 location definition found in the PDEF file. Note that the PDEFVERSION 2.0 location definition positions cells in the lower left-hand corner. Refer to IEEE 1481-1999 specifications for a definition of IEEE locations. This argument overrides the PDEFVERSION specification in the PDEF file and the -pdef_version argument.

pdef_file_name

Specifies the name of the source PDEF file.

`-pdef_version` *version_tag*

Uses the PDEFVERSION specified by *version_tag* instead of the PDEFVERSION tag specified in the PDEF file. For example, if

the PDEF file contains a PDEFVERSION tag of 2.0, this argument enables you to use the 3.0 version (and vice-versa). Note that the original PDEFVERSION tag in the PDEF file will remain untouched. The specified *version_tag* value can be 2.0, or 3.0 (also referred to as IEEE or IEEE 1481-1999).

`-Release` *Status*

Sets the design state to a *placed* status if there is any CELL LOC information in the PDEF file. Any subsequent use of the do_optimize command is launched in the PKS mode. If you don't want to use this mode, reload the design (minus the read_pdef command).

`-use_name_keyword`

Uses the Name keyword structure characteristically associated with PDEFVERSION 2.0. This argument overrides the PDEFVERSION specification in the PDEF file and the -pdef_version argument.

Example

```
read_pdef pdef_file_name -flow -Release Status
```

Related Information

write_pdef

read_wdb

`read_wdb [-rspf_only {true | false}] wdb_name`

Reads routes and netlist changes (antenna diodes) from the `.wdb` file.

You can do two things with this information:

1. Keep the routes inside PKS for further use, for example, to `write_adb` to save routes in ADB, to do a timing analysis based on routed parasitics, or to do a post-route IPO based on routing segments.
2. Do not keep the routes, for example to save memory, but extract the RC information and annotate it into the `pks_shell` for timing analysis. The extracted RC information can also be saved in the form of a reduced standard parasitic format (RSPF) file.

Options and Arguments

`-rspf_only {true | false}`
: Extracts and annotates RSPF information.

`wdb_name`
: Specifies the name of the wroute database (WDB).

Related Information

write_wdb

remove_blockage

```
remove_blockage {[{x y}] | [-bbox {lx ly ux uy}]} [-layer list_of_layernames]
     [-instance name_or_id] [-type {placement_only | routing_only | soft_block |
     regular}] [-all]
```

Removes a blockage type created with the `create_blockage` command. The exact arguments must be given to remove a specified block.

Options and Arguments

`-all`
: Removes all the blockages.

`-instance` *`name_or_id`*
: Removes the blockage associated with any instance of CLASS BLOCK in the design specified by `name_or_id`.

`-layer` *`list_of_layernames`*
: Removes blockages only on the specified layers.

`-type {placement_only | routing_only | soft_block | regular}`
: Removes the blockage type specified by `placement_only`, `routing_only`, `soft_block`, or `regular`.

`{x y} | -bbox {lx ly ux uy}`
: `{x y}`
Removes all blockages containing the specified x y coordinates.
`-bbox {lx ly ux uy}`.
Removes the blockage bounding box specified by the *`lx ly ux uy`* coordinates.

Examples

```
remove_blockage {10 10}
remove_blockage -layer METAL1
remove_blockage -all
```

Related Information

create_blockage

report_blockage

remove_physical_cluster

`remove_physical_cluster -name` *`cluster_name`*

Removes a physical cluster created by the `create_physical_cluster` command. *`cluster_name`* is interpreted as a hierarchical cluster name. Any instances that were associated to the removed cluster are now associated to the parent cluster. If the removed cluster is a `soft_block`, all routing information is removed from that cluster. Note that you cannot remove the root cluster.

Options and Arguments

`-name` *`cluster_name`*

Specifies the name of the physical cluster being removed.

Related Information

create_physical_cluster

remove_physical_connection

```
remove_physical_connection -net net_name_or_id [-cluster_pin cluster_pin_list]
     [-physical_pin physical_pin_list]
```

Removes true physical pins created by the `add_physical_connection` command.

Note: Wildcard connectivity cannot be removed.

Options and Arguments

`-cluster_pin` *`cluster_pin_list`*
: Defines the cluster pin names in a current cluster or a direct soft_block sub-cluster.

`-net` *`physical_netname`*
: Specifies the name of a physical net in the current cluster context.

`-physical_pin` *`physical_pin_list`*
: Defines a set of specific physical cell instance pins to which the physical net is connected to.

Example

The following command removes net `n_310` from the design:

```
remove_physical_connection -net n_310
```

Related Information

add_physical_connection

remove_physical_instance

`remove_physical_instance` *`instance_name`*

Removes the physical instance created by the `create_physical_instance` command.

Options and Arguments

`name`

Specifies the name of the physical object (non-hierarchical) that is searched for in the `current_cluster` context.

Related Information

create_physical_instance

remove_physical_net

`remove_physical_net` *`net_name`*

Removes the physical net created by the `create_physical_net` command.

Options and Arguments

`net_name`

Specifies the name of the physical object (non-hierarchical) that is searched for in the `current_cluster` context.

Related Information

create_physical_net

remove_physical_pin

`remove_physical_pin` *`name`*

Removes the physical pin created by the `create_physical_pin` command.

Options and Arguments

`name`

Specifies the name of the physical object (non-hierarchical) that is searched for in the `current_cluster` context.

Related Information

create_physical_pin

remove_placement_area

`remove_placement_area [-by_name name] [-by_position {x y}] [-all]`

Removes a specified placement area.

Note: Using this command can cause significant timing changes in ECO due to unplaceable areas.

Options and Arguments

`-all`
: Removes all placement areas.

`-by_name`
: Removes a placement area based on its name.

`-by_position {x y}`
: Removes a placement area based on its location specified by the *x*, *y* coordinate pairs used. Note that you must type the curly braces in the command syntax.

name
: Specifies the name of the placement area.

Examples

```
remove_placement_area -by_name a1
remove_placement_area -by_position [3524. 2480.}
remove_placement_area -all
```

Related Information

create_placement_area

report_placement_area

remove_supply_rails_on_rows

```
remove_supply_rails_on_rows
```

Removes any supply rails that are lying on rows.

Related Information

generate_supply_rails_on_rows

report_supply_rails_on_rows

set_supply_rails_on_rows

report_block_halo

```
report_block_halo list_of_instance_names
```

Reports the block halo that has been set around all of the macro blocks in the design. If no block halo is set, the default is reported.

Options and Arguments

list_of_instance_names
Specifies the name of the instance being checked.

Example

```
report_block_halo blockA/i_10
```

Related Information

set_block_halo

report_blockage

```
report_blockage [-instance name_or_id] [-type {placement_only | routing_only |
     soft_block | regular}]
```

Reports the blockage specifications created with the `create_blockage` command. The exact arguments must be given to report on a specified block. If no arguments are specified, all blockages are reported in the ASCII table.

Options and Arguments

`-instance` *`name_or_id`*
: Reports the blockage specifications of any instance of CLASS BLOCK in the design specified by `name_or_id`.

`-type {placement_only | routing_only | soft_block | regular}`
: Reports on the blockage type specified by `placement_only`, `routing_only`, `soft_block`, or `regular`.

Example

The following example assumes you created the blockages using the following commands:

```
create_blockage -bbox {0 0 266.50 284.54} -type routing_only -instance I2/u1_pgcore/fifo/
gfx_fram_i/ram_64_32
create_blockage -bbox {0 0 129.56 446.08} -type routing_only -instance I2/u1_pgcore/clut/rd_clut/
ram_64_32
```

The following command displays the report below:

```
the report_blockage
```

```
+-------------------------------------------------------------------------------------+
|  Type   |         bBox            |Layer_blocked |           Instance_name           |
|---------+-------------------------+--------------+-----------------------------------|
| routing | 0.00 0.00 266.50 284.54 |     All      |I2u1_pgcore/fifo/gfx_fram_i/ram_64_32|
| routing | 0.00 0.00 129.56 446.08 |     All      |I2/u1_pgcore/clut/rd_clut/ram_64_32  |
+-------------------------------------------------------------------------------------+
```

Note: The `Instance_name` entries in the report contain full hierarchical paths to eliminate any confusion.

Related Information

create_blockage

remove_blockage

report_cluster

`report_cluster` *`cluster_name`* `[-tcl_list]`

Reports the following design information:

- Number of components, nets, pins, physical instances, physical nets, physical pins
- Number of core instances (fixed/placed/unplaced)
- Number of block instances (fixed/placed/unplaced)
- Number of cover cells
- Number of clusters (region/soft)
- Number of sites / core instance size
- Average row utilization
- Number of power specifications

Options and Arguments

`-tcl_list`

Reports a Tcl list of the design information.

Example

The following command reports on cluster C1 and outputs the information in a Tcl list:

```
report_cluster C1 -tcl_list
```

report_floorplan_parameters

```
report_floorplan_parameters [-tcl_list]
```

Reports information about the floorplan. To change and control these parameters, use the `set_floorplan_parameters` command.

Options and Arguments

`-tcl_list`

Reports a Tcl list of floorplan parameters.

Example

The following is a typical output file for the `report_floorplan_parameters` command:

```
+-----------------------------------------------------+
|     FloorplanParameter      |    Value    |  Source |
|-----------------------------+-------------+---------|
|     -bbox_initial <lx>      |    0.000    | Derived |
|     -bbox_initial <ly>      |    0.000    | Derived |
|     -bbox_initial <ux>      |   750.400   | Derived |
|     -bbox_initial <uy>      |   748.000   | Derived |
|      -bbox_start <lx>       |    0.000    | Default |
|      -bbox_start <ly>       |    0.000    | Default |
|      -bbox_start <ux>       |   750.400   | Default |
|      -bbox_start <uy>       |   748.000   | Default |
| -lr_io_to_core_distance <l> |    4.800    | Default |
| -lr_io_to_core_distance <r> |    4.800    | Default |
| -tb_io_to_core_distance <b> |    6.000    | Default |
| -tb_io_to_core_distance <t> |    6.000    | Default |
|    -aspect_ratio_initial    |    1.000    |  User   |
|      -fixed_floorplan       |    true     | Default |
|       -height_initial       |   600.000   | Default |
|       -width_initial        |   400.000   | Default |
|         -max_height         |      -      | Default |
|         -max_width          |      -      | Default |
|  -row_utilization_initial   |   70.018%   | Derived |
|  -row_utilization_current   |   70.011%   | Derived |
|    -max_row_utilization     |   90.000%   | Default |
|  -max_local_row_utilization |  100.000%   | Default |
|        -row_spacing         |    0.000    | Default |
|       -abut_row_pairs       |    false    | Default |
|    -flip_alternate_rows     |    false    | Default |
|       -lr_block_halo        |    0.000    | Default |
|       -tb_block_halo        |    0.000    | Default |
|    -top_layer_limit         |     5       | Default |
|   -bottom_layer_limit       |     1       | Default |
+-----------------------------------------------------+
```

Related Information

do_initialize_floorplan

set_floorplan_parameters

report_grow_parameters

`report_grow_parameters`

Reports information about how the design is to grow. To change and control these parameters, use the `set_grow_parameters` command.

Example

The following shows a typical output file for the `report_grow_parameters` command:

```
+-------------------------------------------------------+
|        Grow Parameter       |    Value     |  Source |
|-----------------------------+--------------+---------|
|        -grow_origin         |      lr      | Derived |
|      -max_left_offset       |   100.000    | Default |
|     -max_bottom_offset      |    0.000     | Default |
|      -max_right_offset      |    0.000     | Default |
|      -max_top_offset        | 2147483.750  | Default |
+-------------------------------------------------------+
```

Related Information

set_grow_parameters

report_net_distribution

```
report_net_distribution [-tcl_list]
```

Reports information on signal nets:

- Number of term nets (0/1/2/3/4/5/6/7/8/9/10-19/20-29/30-39/40-49/50-99/>=100)
- Average terms per net
- Total number of nets
- Total number of nets with route information
- Total number of unplaced nets
- Total wirelength of all fully placed nets (horizontal/vertical)

Options and Arguments

`-tcl_list`

Reports a Tcl list of the signal net information.

report_net_rc

`report_net_rc [-net net_name_or_id_list] [-tcl_list]`

Reports the following resistance and capacitance values for nets:

- *NetName*—the name of the specified net(s).
- *Steiner R/C*—the pure steiner calculated values independent of any route data.
- *SteinerLeng H/V*—the horizontal and vertical wirelength values.

All information is generated from the current top timing module. Use the do_push command to collect data at the sub levels.

The reported net capacitance is in two parts: `Ignore Route` and `With Route`. For a placed netlist the `Ignore Route` and `With Route` capacitances are same. For a routed netlist, the `Ignore Route` part has steiner based net length capacitance, and the `With Route` part has routed net length capacitance.

Note: This command reports the capacitance of the net only, not input pins. The reported net capacitance is based on PKS extraction only, and is independent of the standard parasitic values read in through `read_spf` and `read_spef`.

Options and Arguments

`-net list_of_nets`

Specifies the name of the nets you are reporting on.

Note: The list of nets you provide must include the top level of any net for which you want a report. If you are not sure whether one of the nets in your list belongs to a logically lower level net, use the following command:

`report_net_rc -net [get_physical_info -def_name netname]`

`-tcl_list`

Reports a Tcl list of the resistance and capacitance values.

Example

The following command generates the report below:

```
report_net_rc -net [find -net fmyx_w_fbys_1*]
```

	Ignore Route				With Route			
	RC		Length		RC		Length	
NetName	R	C	(H)	(V)	R	C	(H)	(V)
fmyx_w_fbys_1	646.01526	0.4069874	953.77301	789.005	817.95959	0.58929509	900.34698	1704.109
fmyx_w_fbys_10	460.57712	0.2679894	525.58398	616.52899	649.78772	0.30598477	498.849	804.01398
fmyx_w_fbys_10	460.57712	0.2679894	525.58398	616.52899	649.78772	0.30598477	498.849	804.01398
fmyx_w_fbys_11	502.02869	0.30004334	715.10498	570.35999	747.94464	0.40084186	726.73297	985.86798
fmyx_w_fbys_12	571.04242	0.34186554	934.06799	536.927	734.40808	0.43712389	922.75299	962.64801
fmyx_w_fbys_13	381.87839	0.23812148	654.495	370.31201	509.46411	0.23627302	621.84003	381.866
fmyx_w_fbys_14	441.82913	0.27537292	743.78497	440.64401	738.41486	0.52558601	623.37598	1690.954
fmyx_w_fbys_15	420.7468	0.26710063	570.51801	570.29303	533.51868	0.28635997	623.45398	618.70898
fmyx_w_fbys_16	315.18405	0.18819347	573.802	239.14	450.31918	0.2295174	590.54199	412.42999
fmyx_w_fbys_17	431.91675	0.28434074	599.13599	614.87201	581.57349	0.29739299	565.70203	718.11603
fmyx_w_fbys_18	597.97314	0.36327013	779.63898	772.11798	831.82684	0.65578598	941.67499	2006.908
fmyx_w_fbys_19	341.48364	0.23199838	758.68903	246.211	434.00464	0.24630332	763.573	331.57599

report_overlap

```
report_overlap [-detail] [-snap]
```

Reports number of overlapping instances in the design.

Options and Arguments

`-detail`

Prints a detailed report, including the x and y co-ordinates for the overlapping instances.

`-snap`

Snaps to the grid before checking for overlaps.

Example

- The following example shows a sample report:

```
> report_overlap -detail
Instnaces Overlapping: i12 (at 2180,1080) and i11 ( at 1080,1080) <PHY-613>.
Instnaces Overlapping: i45 (at 3600,1080) and i12 ( at 2180,1080) <PHY-613>.
Info:    3 overlapping instance(s) in the design. <PHY-614>.
```

report_physical_library

```
report_physical_library [-tcl_list]
```

Reports on LEF data:

- Unit Rh, Rv, Ch, Cv
- Number of layers (name/direction/routing)
- Number of vias (regular/nonregular)
- Number of NDRs, LUTs, LEFmultipliers
- Number of LUTs

Options and Arguments

`-tcl_list`

Reports a Tcl list of the LEF data.

report_placement_area

`report_placement_area`

Lists all of the placement areas in the database. Placement areas with no names will contain a question mark (*?*) in the name string.

Example

The following is an example of the output generated by the `report_placement_area` command:

```
+-----------------------------------------------+
| Name | lx       | ly      | ux      | uy     |
|------+----------+---------+---------+--------|
|  a1  | -1066.41 | -481.00 | 1066.00 | 770.80 |
+-----------------------------------------------+
```

Related Information

create_placement_area

remove_placement_area

report_preroute_parameters

```
report_preroute_parameters [-tcl_list]
```

Provides you with a report to view the preroute parameters that have been set.

Options and Arguments

`-tcl_list`

Reports the Tcl list preroute parameters.

Related Information

set_preroute_parameters

report_supply_rails_on_rows

```
report_supply_rails_on_rows
```

Outputs the supply rail specifications currently being used. Among other things, the output will indicate whether the specifications were derived automatically from the LEF library, or were input by the user through the `set_supply_rails_on_rows` command.

Related Information

generate_supply_rails_on_rows

remove_supply_rails_on_rows

set_supply_rails_on_rows

report_unplaced

```
report_unplaced [-instances] [-clusters | -physical_instances | -all]
```

Reports instance and cluster information.

Options and Arguments

`-clusters`
: Reports unplaced clusters.

`-instances`
: Reports any unplaced block instances and fixed standard cell instances.

Example

The following command provides a report of unplaced clusters and instances in your design:

```
report_unplaced -instances -clusters
```

reset_dont_move

`reset_dont_move` *`instance_list`*

Removes the attribute placed on an instance(s) by the `set_dont_move` command.

Options and Arguments

`instance_list`

Lists the hierarchical names (wildcards accepted) or object IDs of the instance(s).

Related Information

set_dont_move

reset_net_physical_attribute

```
reset_net_physical_attribute [-layer_usages_table] [-lef_multiplier]
     [-non_default_rule] {[-non_clock_tree_nets] | [-clock_tree_nets] | [-all] |
     [-net list_of_nets]} [-shieldNet] [-use]
```

Resets the method of capacitance and resistance (RC) calculation set by a `set_net_physical_attribute` command.

The options for this command are similar to those for the set_net_physical_attribute command. Please see that command for detailed descriptions of each option.

There are three ways to change the RC calculation for a net:

- `-layer_usages_table`
- `-lef_multiplier`
- `-non_default_rule`

There are four ways to apply changes to a design:

- `-all`
- `-clock_tree_nets`
- `-net`
- `-non_clock_tree_nets`

If you use the `-net` option, each net from the specified list of nets corresponding properties defining the way of RC calculation set for the net earlier will be ignored and current default properties will be used.

If you use the `-non_clock_tree_nets`, `-clock_tree_nets`, or `-all` option, then the corresponding current default properties defining the method of RC calculation (`-layer_usages_table`, `-lef_multiplier`, or `-non_default_rule`) will be ignored and initial default properties generated by the tool will be used.

Options and Arguments

See set_net_physical_attribute on page 655 for details on the options and arguments for this command.

Examples

- The following command specifies that the current default LUT will be used for RC calculation for net `m_clk`.

  ```
  reset_net_physical_attribute -layer_usages_table -net m_clk
  ```

- The following command specifies that the initial default LUT generated by the tool will be used for RC calculation for all nets.

  ```
  set_net_physical_attribute -layer_usages_table LUT1 -al
  ```

Related Information

set_net_physical_attribute

set_block_halo

`set_block_halo` *`instName`* `[-left` *`lval`*`] [-right` *`rval`*`] [-top` *`tval`*`] [-bottom` *`bval`*`]`

Defines the region around a macro reserved for routing. This command internally creates placement areas which cause the rows to adhere to the block halo. This only applies values for macro cells as specified in the LEF file. This is an overriding value; this value is in affect even if the global defaults are changed. This only affects optimization space and should not result in a timing change. This also affects placement resources.

The options `-right`, `-left`, `-top`, and `-bottom` refer to the original rotation of the block as specified in the LEF and not as it appears on the screen. For example, if a block is rotated 180 degrees and you specify `set_block_halo` *`block_inst_name`* `-top 10` then the halo actually appears on the bottom side of the block in the placement.

Note: Specifying block halos will snip rows in the database for all macros which have at least one non-zero halo value.

Options and Arguments

`-bottom` *`bval`*

Defines the halo distance on the bottom side. The default value is *`0`*.

`instName`

Specifies the name of the instance.

`-left` *`lval`*

Defines the halo distance on the left side. The default value is *`0`*.

`-right` *`rval`*

Defines the halo distance on the right side. The default value is *`0`*.

`-top` *`tval`*

Defines the halo distance on the top side. The default value is *`0`*.

Example

```
set_block_halo top/block/macroInst -left 10 -right 10 -top 25 -bottom 30
```

Related Information

report_block_halo

set_block_rc_rule

```
set_block_rc_rule {-instance blockInst_or_id | -all_blocks}
     {-auto | -lut LUTname}
```

Applies a separate layer usages table (LUT) for over-block-routing to all routing segments that route through a block instance.

Options and Arguments

`-all_blocks`

Specifies that the LUT will be used for all over block routing whenever it occurs. Note that the use of this argument does not override any specific instance assignment done previously.

`-auto`

Performs automatic LUT characterization. Specifies that the system will automatically customize block specific LUTs by analyzing the obstacle description and normalizing the block LUT table to reflect the actual layer availability.

Note that auto LUT characterization for a specific net is only done at the LEF description level. It does not consider any instance specific blockages, such as the blockages created with the `create_blockage` command. In other words, the block LUT for all occurrences of a certain macro will be the same. You can override this by loading custom LUTs on a per instance basis.

`-instance` *`blockInst_or_id`*

Identifies the name of the block instance or id that the LUT is being applied to.

`-lut` *`LUTname`*

Identifies the name of the LUT you are referencing. The *`LUTname`* must be the table name of a previously read in LUT definition.

Example

```
set_block_rc_rule -all_blocks -auto
```

Related Information

set_global_auto_block_rc_rule

set_current_cluster

```
set_current_cluster [-name cluster_name | cluster_name]
```

Changes the root cluster (provided by default) to the cluster specified by `-name` *`hier_cluster_name`*. Use this command prior to using any existing floorplan related command, such as commands for setting floorplan spec, creating a blockage, creating placement areas, and so on. You can also use this command to define the cluster to directly contain physical instances or for querying physical instance information, as well as for new commands for creating and modifying cluster definitions from the command line. Note that any cluster name that starts with a / will be considered an absolute path; where / is used to refer to the root physical cluster. Any cluster name that does not start with a / will be considered to be specified in the context of the current active cluster. The tool uses / as the default cluster hierarchy separator.

Note: Changing the current cluster will have no effect on batch commands, including placement, routing, and optimization. The command is intended for interactive editing or for querying commands only. Also note that changing the current cluster does not change the current module.

Options and Arguments

`-name` *`cluster_name`*

Sets the new cluster name (specified by *`cluster_name`*) as the current cluster. Use `/` as a hierarchy divider, use `\` to escape the hierarchy divider, and use `\\` as a backquote.

Example

The following command specifies cluster C3 as the current cluster:

```
set_current_cluster -name C3
```

set_default_core_site

`set_default_core_site` *`core_site_name`*

By default, when your library has multiple core sites, PKS chooses the one referred to by the greatest number of standard cells in the library. Use this command to specify a particular core site to use as the default.

Options and Arguments

`core_site_name`	The name of the core site you want to set as default.

set_dont_move

`set_dont_move` *`instance_list`*

Identifies the instances that will not be moved during optimization. The instances will not be moved, but can be deleted or resized in-place unless you also assert a `set_dont_modify` on them. The instances are written out with the +FIXED attribute in the DEF file. This property is not an attribute and can not be viewed by the `get_info` or `get_physical_info` commands. This property is preserved in an ADB file.

Note: The recommended usage is to use this command after the related instances have already been placed using `do_place`, or after reading the PDEF or DEF file.

To remove the attribute on an instance, use the `reset_dont_move` *`instance_list`* command.

Options and Arguments

`instance_list`

Lists the hierarchical names (wildcards accepted) or object IDs of the instance(s).

Example

```
set_dont_move blockA/i_10 blockB/i_40
```

Related Information

reset_dont_move

get_physical_info

set_floorplan_parameters

```
set_floorplan_parameters [-abut_row_pairs|-no_abut_row_pairs]
     [-aspect_ratio_initial float] [-bbox_initial float float float float]
     [-by_tracks] [-fixed_floorplan | -no_fixed_floorplan]
     [-flip_alternate_rows | -no_flip_alternate_rows] [-height_initial float]
     [-lr_io_to_core_distance {float | float float}]
     [-tb_io_to_core_distance {float | float float}] [-lr_block_halo float]
     [-tb_block_halo float] [-max_height float] [-max_row_utilization float]
     [-max_local_row_utilization float][-max_width float] [-report_only]
     [-row_spacing float] [-row_utilization_initial float] [-width_initial float]
     [-route_top_layer_limit value] [-route_bottom_layer_limit value]
```

Sets floorplan specification parameters for the root cluster.

Options and Arguments

`-abut_row_pairs`
: Keeps flipped row pairs together. Used with the `flip_alternate_rows` command.

`-aspect_ratio_initial`
: Defines the aspect ratio for the initial floorplan. This is used in conjunction with `row_utilization`, `io_to_core_distance`, and `row_spacing` to determine the initial floorplan. Note that the aspect ratio = width/height.

`-bbox_initial`
: Defines the initial bounding box of the design. It defines which bounding box to use when the design is initially converted into a *placed* state. If specified, the other parameters in this block are not used: `aspect_ratio_initial`, `height_initial`, and `width_initial`.

`-by_tracks`
: Indicates that the `io_to_core_distance`, `row_spacing`, `lr_block_halo`, and `tb_block_halo` values are to be interpreted in terms of tracks of the lowest routing layer in each direction.

`-flip_alternate_rows`
: Causes every other row to be oriented opposite the one preceding it (such as: North, flipped South, North, flipped South, and so on).

`-height_initial`

Defines the height of the initial floorplan. Given `row_utilization_initial`, `io_to_core_distance`, and `row_spacing`, the width of the design is calculated.

`-lr_block_halo`

Defines the amount of block halo on the left/right sides. The default value is *0*.

`-lr_io_to_core_distance`

Defines the left and right io-to-core distance. The io-to-core distance is the distance between the die box of a given edge and the rows that are used for the core cell placement. The default value is 6 metal pitches using the lowest vertical routing layer.

`-max_height`

Defines the maximum height that the design can grow. Sets the limit of growth for a particular floorplan.

`-max_local_row_utilization`

Controls the amount of row area that is made available in each local bin (region) for PKS optimization. PKS optimization will not accept any moves that would cause the local utilization to increase beyond this target. Normally this is set to 100% which means that all the bin's row area is made available to optimization

Note: Some part of the row area could be subtracted if the bin is deemed to be congested. That is, a certain number (X) of the available sites could be unusable due to routing congestion. If the bin is not congested, X will be zero. If the bin is very congested, X can be as high as the total number of sites.

For example, if there are 100 sites in a bin, optimization can place up to 100-X sites instances in that bin. If, however, you set `-max_local_row_utilization` to 95, then optimization can place only 95-X sites.

For designs with very low utilization, setting this option to a number between 90 and 100 may help ensure that the design has no local routing congestion and that it can be easily legalized.

`-max_row_utilization`

Defines the maximum row utilization allowed. This is a constraint that the tool will attempt to maintain during optimization. If the maximum row utilization is exceeded, a “grow” on the design is attempted.

`-max_width`

Defines the maximum width that the design can grow. Sets the limit of growth for a particular floorplan.

`-report_only`

Reports the set and derived values if those arguments are issued. Will not change any of the specification parameters in the database. Recommended if you are unsure of the relationships between values.

`-route_top_layer_limit` *`value`* and `-route_bottom_layer_limit` *`value`*

Limits the top or bottom routing layer to the specified value. Takes an integer value from 1 to the number of all routing layers. This value is stored per cluster and effects the cluster and all child clusters which can inherit this value.

Using these arguments causes a timing reset and creates a new layer usages table (LUT). For example, in a 6-layer technology, if bottom = 2 and top = 5, a new LUT is created that does not use M1 and M6. The new LUT will be used on the cluster and for all nets associated with the cluster (on macro blocks, a LUT specific to each macro is derived based on this new cluster LUT and used on all macros of the same cell). These arguments have no affect on user-defined LUTs.

Causes the route availability on unused layers to be set to 0. Routing blockages on unused layers will have no affect.

Added to the `do_route` option list automatically so when global route is called it uses the proper layers (such as `-route_top_layer_limit` and `-route_bottom_layer_limit`). A layer limit explicitly added to the `do_route` option list will override what is set as a floorplan parameter.

Note: A sub cluster, if without a layer limit set by `set_floorplan_parameter`, will inherit the layer limit from the closest parent cluster which has the layer limit set this way.

`-row_spacing`

Defines the space between the rows created in the core. The rows start in the lower left of the core area and are stepped by the core site height plus the `row_spacing`. The last row is at least the io-to-core distance from the upper right, but may be more than this minimum distance.

`-row_utilization_initial`

Defines the row utilization used when creating the initial floorplan. The final utilization may be different depending on the optimizations done, but it will never be more than the `max_row_utilization`.

`-tb_block_halo`

Defines the amount of block halo on the top/bottom sides. The default value is *0*.

`-tb_io_to_core_distance`

Defines the top and bottom io-to-core distance. The io-to-core distance is the distance between the die box of a given edge and the rows that are used for the core cell placement.

`-width_initial`

Defines the width of the initial floorplan. Given `row_utilization_initial`, `io_to_core_distance`, and `row_spacing`, the height of the design is calculated.

Related Information

do_initialize_floorplan

report_floorplan_parameters

set_ground_net

```
set_ground_net
```

Sets the ground net to be used to connect tie-low connections for a specific cluster. This command operates on the current area cluster.

Related Information

get_ground_net

get_power_net

set_power_net

set_grow_anchors

```
set_grow_anchors [-horizontal { left | right } ] [-instance anchor_name]
     [-vertical { top | bottom }] [ list_of_macro_names ]
```

Supplements the `set_floorplan_parameters` command to enforce certain patterns of the post-grow motion of hard macros. The `set_grow_parameters` and `set_pin_status` commands also support grow functionality.

Note: This command only affects hard macros. It does not affect clusters.

Options and Arguments

`-horizontal {left | right}`
: Freezes the macro in the horizontal direction so that its left or right side distance to the bounding box remains unchanged (or changes minimally to avoid overlap with any external blockage).

`-instance` *`anchor_name`*
: Sets a macro name (specified by *`anchor_name`*) as an anchor so every satellite macro specified by the *`list_of_macro_names`* will move in sync with the anchor. In other words, the relative position(s) between the anchor and the satellite macro(s) remains unchanged. Note that if *`list_of_macro_names`* is omitted, all macros (except the anchor) are considered satellite macros.

`list_of_macro_names`
: Specifies the name of the macro(s) being affected. Note that if *`list_of_macro_names`* is omitted, all macros are affected by the specified argument.

`-vertical {top | bottom}`
: Freezes the macro in the vertical direction so that its top or bottom side distance to the bounding box remains unchanged (or changes minimally to avoid overlap with any external blockage).

Example

The following command freezes the macro in the horizontal direction so that its left side distance to the bounding box remains unchanged (or changes minimally to avoid overlap with any external blockage):

```
set_grow_anchors -horizontal left
```

Related Information

set_floorplan_parameters

set_grow_parameters

set_pin_status

set_grow_parameters

```
set_grow_parameters [-max_box {lx ly ux uy}] [-max_bottom_offset value]
     [-max_left_offset value] [-max_right_offset value]
     [-max_top_offset value] [-max_offsets {bottom_offset left_offset
     right_offset top_offset}]  [-origin {ll | lr | ul | ur | cc}]
     [-start_box {lx ly ux uy}]
```

Supplements the `set_floorplan_parameters` command to set the maximum offsets for growing the specified root cluster. The `set_grow_anchors` and `set_pin_status` commands also support grow functionality.

Options and Arguments

`-max_bottom_offset` *value*
: Sets the bottom offset to its maximum allowed growth. This enables you to set an individual offset *value* in microns.

`-max_box {`*lx ly ux uy*`}`
: Specifies the maximum size a bounding box can grow.
lx = lower *x* coordinate
ly = lower *y* coordinate
ux= upper *x* coordinate
uy= upper *y* coordinate
Note that you must type the curly braces in the command syntax.

`-max_left_offset` *value*
: Sets the left offset to its maximum allowed growth. This enables you to set an individual offset *value* in microns.

`-max_offsets {`*bottom_offset left_offset right_offset top_offset*`}`
: Specifies the maximum allowed growth in each direction from the initial bounding box of the cluster. All offsets must be specified. Units in microns. Note that you must type the curly braces in the command syntax.

`-max_right_offset` *value*
: Sets the right offset to its maximum allowed growth. This enables you to set an individual offset *value* in microns.

`-max_top_offset` *value*
: Sets the top offset to its maximum allowed growth. This enables you to set an individual offset *value* in microns.

`-origin {ll | lr | ul | ur | cc}`
Specifies a fixed corner for the cluster. Note that setting an origin sets two max_offsets, which cannot be overridden. For example: origin `ll` sets `max_left_offset=0` and `max_bottom_offset=0`.

`-start_box {`*`lx ly ux uy`*`}`
Specifies the initial bounding box of the sub cluster you want to grow. If not specified, it assumes that the cluster bounding box is the same as that of the whole chip (die box). To check the initial die box dimensions of the top-level cluster, use the `set_floorplan_parameters -initial_bbox` command. Note that you must type the curly braces in the command syntax.

Examples

- The following command specifies the maximum size the bounding box can grow in the specified x and y directions:

  ```
  set_grow_parameters -max_box {lx ly ux uy}
  ```

- The following command specifies the initial size of the subcluster's bounding box:

  ```
  set_grow_parameters -start_box {700 0 1200 500}
  ```

- The following command specifies the lower right corner as the fixed corner for the cluster:

  ```
  set_grow_parameters -origin lr
  ```

- The following command sets the left offset to a maximum growth of 100 (microns):

  ```
  set_grow_parameters -max_left_offset 100
  ```

Related Information

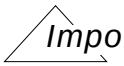
set_floorplan_parameters

set_grow_anchors

set_pin_status

set_layer_usages_table

`set_layer_usages_table -name table_name filename`



This command performs operations similar to those found in other commands. If this command is used, the read_layer_usages command is disabled and will have no effect.

Reads named layer usages tables. This command is used to load tables for use by `set_net_physical_attribute`.

The layer usages table specifies how nets will be routed by specifying what percentage of each wire will be routed on each layer. See *Appendix A* of the *PKS User Guide* for more information about layer usages tables.

Options and Arguments

`-name table_name filename`
: Loads tables for use by the `set_net_physical_attribute` command. The data is read from the file specified and a *table_name* is generated and associated with the layer usages table that was read. The *table_name* is used as an argument for the `set_net_physical_attribute` command, for example: `set_net_physical_attribute -layer_usages_table table_name -all`. The *table_name* can not be overwritten. If *table_name* already exists, an error message is issued and the data is ignored.

Example

The following command loads the specified table from the layer usages table (*LUT1*) that was read in. The resulting table is used by the `set_net_physical_attribute` command.

```
set_layer_usages_table -name Table1 LUT1
```

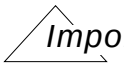
Related Information

create_layer_usages_table

set_net_physical_attribute

set_lef_multiplier

```
set_lef_multiplier -name multiplier_name [-layer layer_name]
     [-capacitance c]  [-edge_capacitance e] [-resistance r] [-via_resistance v]
```



Important

If this command is used, the read_layer_usages command is disabled and will have no effect:

Specifies a multiplier to be used in conjunction with the capacitance and resistance values in the LEF file.

More than one layer can be associated with each multiplier (specified by the `-name` option). You must issue separate commands for each layer to be associated with a particular multiplier.

You can specify more than one set of resistance and capacitance multipliers for a layer by using different multiplier names.

Note: If you issue a `set_lef_multiplier` command with a layer and data that have already been specified, the data are overwritten.

Options and Arguments

`-capacitance` *c*

Specifies a float value, *c*, that the capacitance for the specified layer will be multiplied by.

`-edge_capacitance` *e*

Defines a float value, *e*, that the edge capacitance for the specified layer will be multiplied by.

`-layer` *layer_name*

Defines the layer that the multipliers will be applied to. This is an optional field. If this argument is omitted, multiplier values specified by `-capacitance`, `-edge_capacitance`, and `-resistance` will be applied to all routing layers. The multiplier value specified by `-via_resistance` will also be applied to all routing layers, with the exception of the highest routing layer.

`-name` *multiplier_name*

Defines how the multiplier will be referenced by `set_net_physical_attribute`. This is a required option.

`-resistance` *r*

Defines a float value, *r*, that the resistance for the specified layer will be multiplied by.

`-via_resistance` *v*

Defines a float value *v*, that the `via_resistance` for the specified lower layer will be multiplied by.

This option applies to the via with the layer specified by the `-layer` option as the lower layer. If applied to the top layer, an error message is generated and the layer will not be associated with the specified multiplier.

Example

The following command defines `mult1_table` as the multiplier name that will be used with the `set_net_physical_attribute` command.

```
set_lef_multiplier -name mult1_table
```

Related Information

set_net_physical_attribute

set_library_layer_offset

`set_library_layer_offset` *`layername offset_value`*

Sets the offset value for a given routing layer. Only routing layers may have offsets, so *`layername`* is the name of the routing layer.

Note: The *`offset_value`* is stored in the database that stores physical library data.

Options and Arguments

`layername`

Specifies the name of the routing layer.

`offset_value`

Defines the *`offset value`* for a given routing layer is in user units (similar to LEF data).

Related Information

get_library_layer_offset

set_logic_0_net

`set_logic_0_net` *`net_name`*

Sets the net name associated with the pins that are tied low in the design.

Note: If a design contains multiple ground and power nets, you must use this command to specify one of the nets as the default `logic0` net.

Options and Arguments

`net_name`

Specifies the name of the net.

Related Information

set_logic_1_net

set_logic_1_net

`set_logic_1_net` *`net_name`*

Sets the net name associated with the pins that are tied high in the design.

Note: If a design contains multiple ground and power nets, you must use this command to specify one of the nets as the default `logic1` net.

Options and Arguments

`net_name`

Specifies the name of the net.

Related Information

set_logic_0_net

set_min_porosity_for_over_block_routing

```
set_min_porosity_for_over_block_routing val
```

Defines the minimum porosity (min-porosity) a block must have to enable the Steiner router to route over blocks when performing routing estimation. When a LEF description of a block is read into PKS, the porosity is automatically calculated from the metal geometries defined in the LEF for that block. The porosity is calculated as the total unblocked track length divided by the total track length, and is calculated in both the horizontal and vertical directions. When PKS performs Steiner routing, it will route over a block if the min-porosity value is less than both the horizontal and the vertical porosity for that block (see examples below).

Options and Arguments

val

Defines the percentage of routable tracks (0-100). The default value is set to the minimum of `1/Nh` or `1/Nv`, where `Nh` and `Nv` are the number of horizontal and vertical routing layers in the design.

Examples

- In the following example, a design uses five layers of metal and contains a block with layers 1-3 completely obstructed. Odd layers are horizontal and even layers are vertical.

  PKS calculates the horizontal porosity to be 33.3%:

  ```
  Metal1 porosity: 0%
  Metal3 porosity: 0%
  Metal5 porosity: 100%
  Total horizontal porosity = (0 + 0 + 100) / 300 = 33.3%
  ```

  PKS calculates the vertical porosity to be 50%:

  ```
  Metal2 porosity: 0%
  Metal4 porosity: 100%
  Total vertical porosity = (0 + 100)/ 200 = 50%
  ```

  For this block to be considered for over-the-block routing by the Steiner router, you must set the minimum porosity value to be less than or equal to 33.3%:

  ```
  set_min_porosity_for_over_block_routing 33
  ```

- In this next example, consider the same design, but with the block completely obstructed in layers 1-4.

  PKS calculates the horizontal porosity to be 33.3% (same as above).

PKS calculates the vertical porosity to be 0%:

```
Metal2 porosity: 0%
Metal4 porosity: 0%
Total vertical porosity = 0%
```

For this block to be considered for over-the-block routing by the Steiner router, you must set the min-porosity value to 0:

```
set_min_porosity_for_over_block_routing 0
```

Note: If your design contains both of the blocks mentioned above and you set the min-porosity value to 30, the Steiner router will route over the first block (example 1), and go around the second block (example 2). When the steiner router routes over a block, it does not look at how many nets have been routed over. If it has determined that this block can be routed over, then all nets will route over it. It isn't until the global routing phase that the tool can determine which nets will go over a block and which will go around it.

Related Information

get_min_porosity_for_over_block_routing

set_min_RC_multipliers

`set_min_RC_multipliers [-res` *`float`*`] [-cap` *`float`*`] [-edge_cap` *`float`*`]`

Enables you to reset RC values uniformly for all layers, vias, nondefault layers, and nondefault vias. You can use this command as often as needed. Each subsequent command will reset the min RC values.

Options and Arguments

`-cap` *`float`*

Specifies the multiplier to be used with capacitance values.

`-edge_cap` *`float`*

Specifies the multiplier to be used with edge-capacitance values.

`-res` *`float`*

Specifies the multiplier to be used with resistance values.

set_min_wire_length

`set_min_wire_length` *`value`*

Sets the minimum wire length value used during optimization. In other words, if a net length is calculated to be less than this value, for timing (RC calculation) purposes its wire length is treated as <value>. The intention is to build a minimum tolerance for cell movements (such as overlap removal) later on without altering achieved timing. Units are in microns.

Options and Arguments

`value`

Specifies the minimum wire length value.

Example

The following command sets the minimum wire length to 20 microns for use during optimization:

```
set_min_wire_length 20
```

Related Information

get_min_wire_length

set_net_physical_attribute

```
set_net_physical_attribute [-layer_usages_table table_name]
     [-lef_multiplier multiplier_name] [-non_default_rule rule_name]
     {[-non_clock_tree_nets] | [-clock_tree_nets] | [-all] | [-net list_of_nets]}
     [-shieldNet list_of_special_net_names] [-use type]
```

Important

This command performs operations similar to those found in other commands. If this command is used, the read_layer_usages command is disabled and will have no effect.

Changes the way capacitance and resistance (RC) are calculated for a specific net or class of nets.

There are three ways to change the RC calculation for a net:

- `-layer_usages_table`
- `-lef_multiplier`
- `-non_default_rule`

There are four ways to apply the changes to a design:

- `-all`
- `-clock_tree_nets`
- `-net`
- `-non_clock_tree_nets`

Note: If one of the four arguments above is not selected, this command is ignored. If more than one argument is selected, only one argument will be used and in the priority listed above.

Options and Arguments

`-all`

Loads the default settings for `-clock_tree_nets` and `-non_clock_tree_nets`. Does not overwrite information previously set with the `-net` argument.

`-clock_tree_nets`

Loads the default settings for clock tree nets. Overwrites information previously set with the `-clock_tree_nets`

argument or with the `-all` argument for clock tree nets. Does not overwrite information previously set with the `-net` argument, `-non_clock_tree_nets` argument, or `-all` argument for non clock tree nets.

`-layer_usages_table` *`table_name`*
Defines the percentages of each metal layer used to calculate the composite resistance and capacitance for each routing direction (can also differ based on wire length). Applies the *`table_name`* to the nets specified. The *`table_name`* must already be defined using the `set_layer_usages_table` command. A `layer_usages_table` may not be changed once it is created; a new named table must be created if needed.

`-lef_multiplier` *`multiplier_name`*
Scales the values in the LEF file which impacts all resistance and capacitance values for the PKS session. Applies the *`multiplier_name`* to the nets specified. The *`multiplier_name`* must already be defined using the `set_lef_multiplier` command. LEF multipliers may be changed after they are applied. The change will take effect as soon as the *`multiplier_name`* is changed. Note that you must use this argument with the `-all` argument, otherwise an error message is issued and the command will not be executed. It is not supported with a net or a group of nets.

`-net` *`list_of_nets`*
Applies only to the supplied list of nets. Overwrites information previously set with the `-all`, `-net`, `-clock_tree_nets`, or `-non_clock_tree_nets` arguments.

`-non_clock_tree_nets`
Loads the default settings for non clock tree nets. Overwrites information previously set with the `-non_clock_tree_nets` argument or with the `-all` argument for non clock tree nets. Does not overwrite information previously set with the `-net` argument, `-clock_tree_nets` argument, or `-all` argument for clock tree nets.

`-non_default_rule` *`rule_name`*
Applies the named rule to the nets specified. The named rule must be specified in the LEF file.

`-shieldNet` *`list_of_special_net_names`*
: Defines the shielding net for each net specified by *`list_of_special_net_names`* or the *`list_of_nets`* defined by the `-net` argument. Each net name in the list must correspond to a special net that already exists in the design. If not, a warning message is issued and the command will fail. This argument must be used in conjunction with the `-net` argument, otherwise an error message is issued and the command will not be executed.

`-use {clock | signal | power | ground | tieoff | analog | scan | reset | data}`
: Defines the net(s) usage. The valid net types are `clock`, `signal`, `power`, `ground`, `tieoff`, `analog`, `scan`, `reset`, and `data`. Note that you must use this argument with the `-net` argument, otherwise an error message is issued and the command will not be executed.

Examples

```
set_net_physical_attribute -layer_usages_table BASE -lef_multiplier ONE -all
set_net_physical_attribute -layer_usages_table CLOCK -clock_tree_nets
set_net_physical_attribute -non_default_rule WIDE_WIRE_1 -net CLK_PLL -use clock
set_net_physical_attribute -lef_multiplier ONEp5 -layer_usages_table T1
-non_default_rule DOUBLE_WIDTH -all
```

```
 Group          layer_usages_table   lef_multiplier   non_default_rule
--------        ------------------   --------------   ------------
 CLK_PLL          CLOCK                ONE             WIDE_WIRE_1
 clock_tree_nets  CLOCK                ONE             WIDE_WIRE_2
 non_clock_tree   BASE                 ONE
```

This result is exactly the same as using:

```
set_net_physical_attribute -layer_usages_table BASE -lef_multiplier ONE -all
set_net_physical_attribute -layer_usages_table CLOCK -non_default_rule WIDE_WIRE_2
-clock_tree_nets
set_net_physical_attribute -non_default_rule WIDE_WIRE_1 -net CLK_PLL
```

Related Information

reset_net_physical_attribute

set_physical_info

```
set_physical_info  [ [-instance | -pin] object_name | FNPid] [-ipin_name name]
     [-location loc] [-ll | -ur | -lr | -ul] [-type {special | no_special}] [-state
     {unplaced | placed | fixed | cover}]
     [-orientation {N | S | W | E | FE | FW | FS | FN}] [-use name]
```

Indicates whether a certain pin (timing or physical) needs to be connected by a special router or by Wroute. By default all timing pins are “regular” in nature. All pins of a physical net are `special` in nature.

If you specify an argument that does not pertain to an object, the command will skip it. If no arguments are given, an error or warning will result.

Note: To add physical information to nets, use the `add_physical_connection` command, which also allows a setting for `special | no_special`.

Options and Arguments:

`[-instance | -pin] object_name | FNPid]`
: Identifies a physical instance or physical pin specified by object_name. `FNPid` specifies logical objects.

`[-ipin_name name]`
: Identifies an instance pin specified by name. Use this argument in conjunction with the `[-instance object_name | FNPid]` argument.

`[-ll | -ur | -lr | -ul]`
: Specifies the location coordinates of the instance footprint on the die (lower left, upper right, lower right, or upper left).
Default: `-ll`.

`[-location loc]`
: Specifies the location of the pin or instance.

`[-orientation {N | S | W | E | FE | FW | FS | FN}]`
: Specifies the orientation of a pin or instance.

`[-state {unplaced | placed | fixed | cover}]`
: Specifies the state of a pin or instance.

`[-type {special | no_special}]`
: Specifies the type of pin or instance.

`[-use name]`

Identifies the pin being used specified by name.

Related Commands

get_physical_info

add_physical_connection

set_physical_instance

`set_physical_instance [-xpos x] [-ypos y] [-xtrans xt] [-ytrans yt] instance`

Sets the absolute or relative position of an instance, such as the location and orientation. Note that the object must be placed or this command has no effect on the design.

Options and Arguments

`-xpos x`
Sets the absolute x position of the instance.

`-xtrans xt`
Translates the instance on the x-axis by the *xt* amount.

`-ypos y`
Sets the absolute y position of the instance.

`-ytrans yt`
Translates the instance on the y-axis by the *yt* amount.

Example

```
set_physical_instance -xpos 10 i_1
```

Related Information

get_physical_info

set_pin_location

```
set_pin_location pin_name {[-side side] [-index number] | [-location location]
     [-layer layername] [-box {lx | ly | ux | uy}]}
```

Sets one or more of the following: pin side, pin index, pin layer, pin location, or pin geometry.

Options and Arguments

`-box {lx | ly | ux | uy}`
Defines the geometry for the pin.

`-index`
Defines the relative index for the pin on the side it has been assigned. Any number greater than 1. This is specified along with the `-side` argument.

`-layer` *`layername`*
Defines the layer of where the pin is placed.

`-location {x y}`
Defines the location for the pin. This is mandatory but cannot be used in conjunction with the `-side` argument.

`pin_name`
Specifies the name of the pin whose side has to be set.

`-side`
Defines the side that the pin is assigned. Can be one of the following: {left | right | bottom | top}. This is mandatory but cannot be used in conjunction with the `-location` argument.

Examples

- The following command sets the location of the `clk` pin to the left side of the die and sets the relative order to be the last pin on the respective side:

  ```
  set_pin_location clk -side left
  ```

- The following command sets the location of the `clk` pin to the left side of the die and also sets the relative order to be the 19th pin on the left side:

  ```
  set_pin_location clk -side left -index 19
  ```

- The following command sets the pin geometry to be a rectangular box with origin `{0,0}` and height and width 100 relative to the pin location:

  ```
  set_pin_location clk -box {0 0 100 100}
  ```

- The following command sets the pin location to `{10000, 20999}`:

  ```
  set_pin_location clk -location {10000 20999}
  ```

- The following command sets the pin location in the `M1` layer:

  ```
  set_pin_location clk -layer M1
  ```

Related Information

get_pin_location

set_pin_status

```
set_pin_status [-index] [-side {left | right | bottom | top}]
     [-state {fixed | cover}] [pin_names_or_ids]
```

Supplements the `set_floorplan_parameters` command to force specified pins to obey certain rules during grow. If no *`pin_names_or_ids`* are specified, all pins are chosen. The `set_grow_anchors` and `set_grow_parameters` commands also support grow functionality.

Options and Arguments

`-index`
: Indicates that the pins are specified by numerical IDs, which are their track IDs (before grow). Note that this argument is not needed if the pins are specified by numerical IDs in relative rank order along the side, from the bottom to top on the left and right sides and from the left to right on the bottom and top sides.

`pin_names_or_ids`
: Specifies the name or numerical ID of the cluster pin.

`-side {left | right | bottom | top}`
: Applies only when the pins are specified by numerical IDs. It selects only the pins (specified by *`pin_names_or_ids`*) on the specified side of the box. If no *`pin_names_or_ids`* are specified, all of the pins on the specified side of the box are chosen. Note that this argument is not needed if the pins are specified by numerical IDs in relative rank order along the side, from the bottom to top on the left and right sides, and from the left to right on the bottom and top sides.

`-state {fixed | cover}`
: Specifies that the pin is to move only along the same track that it occupied before grow (`fixed`) or that the pin is to stay at the same location (`cover`).

Example

The following command specifies that the sixth pin on the left side is set to fixed:

```
set_pin_status -state fixed -side left 6
```

Related Information

set_floorplan_parameters

set_grow_anchors

set_grow_parameters

set_power_net

`set_power_net`

Sets the power net to be used to connect tie-high connections for a specific cluster. This command operates on the current area cluster.

Related Information

get_power_net

get_ground_net

set_ground_net

set_power_stripe_spec

```
set_power_stripe_spec -direction {vertical|horizontal} -layer name -width W
     [-start_from SF] [-stop_from EF] [-number_stripes NS] [-stripe_spacing SP]
     [-net_spacing DS] [-net_name name1 name2...] [-delete_def_routes]
```

Lets you generate power stripes for one or several power nets. This command describes the rules of how to generate power stripes; it does not describe the power stripes themselves. This command is useful when modeling PG related blockages for congestion with grow on.

More than one `set_power_stripe_spec` command may be issued in one session. It will not check the intersect of power stripes defined by different commands. Creating power stripes with this command will not prevent the design from growing.

Options and Arguments

`-delete_def_routes`
: Deletes the power stripes that were created based on the DEF information that if there are power stripes created based on the information from DEF, they should be deleted before creating power stripes.

`-direction {vertical | horizontal}`
: Defines the direction of the power stripe(s). This argument is mandatory.

`-layer name`
: Defines the power stripe(s) layer name. This argument is mandatory.

`-net_name name1 name2...`
: Defines the list of net names of where to create power stripes. At least one net name should be specified.

`-net_spacing DS`
: Defines the distance between power stripes of the two neighboring nets in the list of net names. This argument is mandatory if you create power stripes for more than one net. If this argument is not defined, only power stripes for the first net in the list is created.

`-number_stripes NS`
: Defines how many power stripes to create. If there is not enough space to create *NS* stripes, than the maximum number of stripes

fitting into the space defined by *SF* and *EF* is created. The default value is 1. If *NS* is not defined, the tool creates as many stripes as possible to fill space defined by `start_from` and `stop_from` parameters.

`-start_from SF`

Defines the distance to the middle of the first power stripe from the left side of the row bounding box (identifies where to start creating the power stripes). If `start_from` and/or `stop_from` are not defined, the default value of 100 microns is used. Zero is a valid value for these two parameters.

`-stop_from EF`

Defines the distance to the middle of the last power stripe from the right side of the row bounding box (identifies where to stop creating power stripes). The default value is 0.

`-stripe_spacing SP`

Defines the distance between the power stripes. This argument is mandatory if you are creating more than one power stripe. If *SP* is not defined, the default value of 200 microns is used. If *SP* is zero, only one stripe is created (the value of the `number_stripes` parameter is ignored if it is more than one).

`-width W`

Defines the width of the power stripe. This argument is mandatory.

Example

The following command sets the direction of the power stripes in the vertical direction on layer `M3` with a width of 3:

```
set_power_stripe_spec -direction vertical -layer M3 -width 10
```

set_preroute_parameters

```
set_preroute_parameters [-min_place_obstacle_size float]
     [-min_route_obstacle_size float]
     [-place_obstacle_layers {layName1 layName2 ..}]
     [-route_obstacle_layers {layName1 layName2 ..}]
     [-use_pads_for_place_obstacles {0|1}] [-use_pads_for_route_obstacles {0|1}]
     [-halo_hor halo_width] [-halo_ver halo_width]
```

Sets a placement blockage (rows will be cut) if an obstacle exists in any of the specified `place_obstacle_layers`, and is at least of `min_place_obstacle_size`.

It can also set a routing blockage (steiner router detours around it) if an obstacle exists in all of the specified `route_obstacle_layers`, and is at least of `min_route_size`.

If your design has area pads and you wish to affect placement/routing, set the `use_pads_for*` variables.

If you wish to use halos, use the `-halo_hor` and `-halo_ver` options.

Options and Arguments

`-halo_hor` *`halo_width`*
: Allows you to specify horizontal halos around pre-route obstructions.
Default: `0`

`-halo_ver` *`halo_width`*
: Allows you to specify vertical halos around pre-route obstructions.
Default: `0`

`-min_place_obstacle_size` *`float`*
: Sets the minimum size of a pre-route or blockage that will be seen as a placement obstruction. Default is 6 metal1 pitches.

`-min_route_obstacle_size` *`float`*
: Identifies the minimum size of a pre-route or blockage that will be seen as a routing obstruction. Default is 6 metal1 pitches.

`-place_obstacle_layers {`*`layName1 layName2 ..`*`}`
: Specifies the list of layers to be seen as placement obstructions. Default is the lowest two routing layers.

`-route_obstacle_layers {`*`layName1 layName2 ..`*`}`
Identifies the list of layers that will be seen as routing obstructions. The default requires that all layers must be in an area for the area to be considered a routing obstruction.

`-use_pads_for_place_obstacles <0|1>`
Tells PKS to use pads (IO or Area) as placement obstacles. The default is 1 (true).

`-use_pads_for_route_obstacles <0|1>`
Tells PKS to use pads (IO or Area) as routing blockages. The default is 1 (true).

set_route_availability

`set_route_availability -layer` *`layername val`*

Sets the raw routing layer percentage that is used by the congestion analysis engine after considering preroutes. Note that setting this value only has an effect during the congestion analysis within PKS. This value is not used as a guide for `do_route`.

Options and Arguments

`layername`

Specifies the name of the routing layer.

`val`

Sets the routing layer percentage that is used by the congestion analysis engine after considering preroutes.

Example

The following command makes 20% of the routing layer (`metal4`) available to the congestion analysis engine after considering preroutes (which means 80% of the routing layer is occupied by preroutes):

```
set_route_availability -layer metal4 20
```

Related Information

get_route_availability

set_special_netpin

`set_special_netpin -type {power | ground}` *`list_of_pin_names`*

Assigns a list of pin names to a special pin connected to a special net of the same name.

Options and Arguments

`list_of_pin_names`
Specifies the name of a list of pins separated by spaces.

`-type power | ground`
Defines the special pin type. The type options are `power` or `ground`.

Example

`set_special_netpin -type ground` *`list_of_pin_names`*

Related Information

get_special_netpins

set_steiner_channel_width

```
set_steiner_channel_width float
```

Enhances the steiner router block avoidance by detouring preferably along sides of the macro that are free of routing obstructions within the specified distance. Timing will be reset if the number is set to a different value.

Options and Arguments

float

Example

The following command sets the Steiner channel width to 20:

```
set_steiner_channel_width 20
```

Related Information

get_steiner_channel_width

set_steiner_mode

```
set_steiner_mode {0 | 1 | 2}
```

Provides three Steiner-routing algorithms to control the Steiner routing functionality in your design. The mode should be set at the beginning of your script.

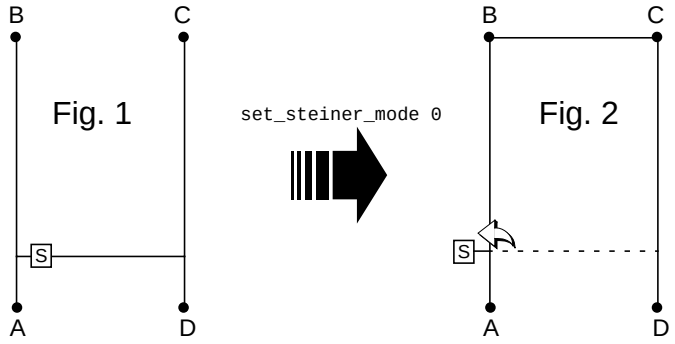
Note: Care should be taken when using this command. It is recommended that you evaluate each mode on your complex floorplan to obtain the optimal results.

Options and Arguments

`set_steiner_mode 0 (wirelength)`

In this mode, the Steiner router attempts to connect all pins, minimizing the total wirelength.

Note: This mode is fast and more than adequate for most designs.



The figure above shows that minimizing the Steiner tree is not always the optimal choice. Assume S (Fig. 1) moves slightly to the left (Fig. 2), the resulting tree in Fig. 2 is in fact optimal from a wirelength perspective, but the delay to pin D has become significantly worse. Care must be taken when using `set_steiner_mode 0`.

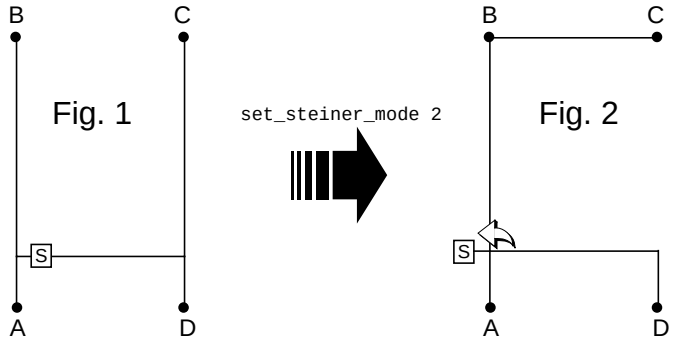
`set_steiner_mode 1`

This mode is a hybrid of Steiner modes `0` and `2` and is not recommended. Note that this mode will be obsolete in the next full release of the software.

`set_steiner_mode 2 (wiredelay)`

Invokes a new timing-conscious Steiner router. When connecting sinks, it tries to balance minimizing the total wirelength versus minimizing the maximum distance between the source and the farthest sink.

Note: Use this command if your design is very timing critical or has many large complex routing obstacles or is highly utilized. This mode is less susceptible to small cell perturbations that cause larger delays owing to a daisy-chain effect that can occur when using Steiner mode `0`.



In the figure above, assume S (Fig. 1) moves slightly to the left (Fig. 2), the resulting tree in Fig. 2 is optimal from a wirelength perspective, and also minimizes the distance between S (source) and the farthest sink.

set_supply_rails_on_rows

```
set_supply_rails_on_rows [-location float -width float -supply {power | ground}
     -layer string] | [-derived]
```

Sets the specifications for the power and ground rails that run through the cell rows. If this command is not used, the specifications are automatically extracted from the LEF library. The supply rail specifications are used during routing congestion analysis to account for routing resources occupied by the supply rails.

Options and Arguments

`-derived`
: Indicates that the supply rail specifications are to be derived automatically from the LEF library. If this argument is given, all existing supply rail specifications are deleted.

`-layer`
: Provides the name of the layer that the rail lies on.

`-location`
: Provides the location of the center line of the rail from the bottom (horizontal rows) or left side (vertical rows) of each cell.

`-supply`
: Provides the use of the rail.

`-width`
: Provides the width of the rail.

Examples

```
set_supply_rails_on_rows -location 9.7 -width 0.2 -layer metal1 -supply power
set_supply_rails_on_rows -derived
```

Related Information

generate_supply_rails_on_rows

remove_supply_rails_on_rows

report_supply_rails_on_rows

vbg_pks_display_ilist

`vbg_pks_display_ilist -add` *`instance_name_list`*

Displays in the Graphical User Interface (GUI) the instances specified in *`instance_name_list`*. Specifically, the desired instances are displayed in the Physical Browser of the GUI. This command is useful for debugging.

Options and Arguments

`-add`

Adds instances to the Physical Browser in the Graphical User Interface.

`instance_name_list`

Specifies the instances that are to be displayed.

Examples

- The following command displays the instances returned by the `find` command:

  ```
  vbg_pks_display_ilist [get_names [find -inst *53]]
  ```

- The following command shows how the empty display list below returns the display mode to its default view:

  ```
  vbg_pks_display_ilist {}
  ```

vbg_pks_group_delete

```
vbg_pks_group_delete {group_name | all}
```

Removes highlights from the objects specified in the DEF *group_name* within the Graphical User Interface (GUI). Specifically, the DEF objects will be de-highlighted from the PKS viewer of the GUI.

Options and Arguments

`all`

Specified that all previously specified DEF groups be removed from the highlight list.

group_name

The DEF group name to be de-highlighted.

vbg_pks_group_display

`vbg_pks_group_display` *`group_name color`*

Highlights all objects specified in the DEF *`group_name`* with the specified color within the Graphical User Interface (GUI). Specifically, the highlighted DEF objects appear in the PKS viewer of the GUI.

Options and Arguments

`color`

Designates color for highlighting objects. If *`color`* is not specified, then the default color is used.

`group_name`

Specifies the DEF group to be highlighted.

write_def

```
write_def [-placement {all | cell | io}] [-netlist] [-hier_delimiter char]
     [top_level_delimiter] [-no_pin_placement] [-number_of_groups num]
     [-cells_per_group num] [-scan_only | -include_scan]
     [-computed_rows] [-no_regions] [-version {5.3 | 5.4}] filename
```

Writes a Design Exchange Format (DEF) view of the current module when physical cell placement information is known for the design. The default operation provides cell placement coordinates, IO pin placement information, and netlist information. It will also write routes read into PKS out to the DEF file. The output is determined by setting the desired command arguments.

When handling designs with mixed pins and pads, the `write_def` command will output placement information for all pins not associated with pads. For pins with associated pads, the placement information will not be output in the DEF file.

Options and Arguments

`-computed_rows`
: Writes the computed rows in PKS to a DEF.

`filename`
: Specifies the name of the DEF file.

`-hier_delimiter` *`char`*
: Defines the character that is used as the delimiter for the hierarchical names.

  Default: /

`-netlist`
: Outputs cell placement and netlist information for all nets connecting the cells.

`-no_regions`
: Does not output any region-related information (region definition and region reference) in the DEF file.

`-placement {all | cell | io}`
: Outputs the *x*, *y* coordinates for all cell and IO pins, cells only, or IO pins only in the design.

`-scan_only | -include_scan`

`-scan_only`
`Excludes all write_def outputs except the scan chains section.`

`-include_scan`
Dumps the scanDEF data to DEF file when `write_def` is used with `-include_scan`.

`-top_level_delimiter`

Controls whether the delimiter for the top-level hierarchy is used in the DEF file. The default value is `false`. If `true`, the delimiter character used is based on the `-hier_delimiter` argument.

`-version {5.3 | 5.4}`

Writes out a 5.3 or 5.4 DEF file. By default, the DEF writer writes out information in the 5.5 syntax. Using this argument enables you to get a 5.3 or 5.4 DEF file for use with flows that cannot handle the newer 5.5 syntax. However, some information may be lost in the process for constructs not supported in 5.3 and 5.4.

Examples

- The following command writes all DEF information to the `full.def` file:

  ```
  write_def full.def
  ```

- The following command writes the cell and IO placement information to the `DesignName.def` file:

  ```
  write_def -placement all DesignName.def
  ```

- The following command writes the cell and IO placement information and all netlist information in the design to the `DesignName.def` file:

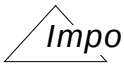
  ```
  write_def -netlist DesignName.def
  ```

Related Information

read_def

write_gns

```
write_gns -hier -placement {io | cells | all} -gns_view view_name
     -gns_lib lib_name -gns_tech tech_lib_name [-create_gns_lib]
```



Important

The Genesis database is no longer supported. This command will be removed in a future release.

Writes a PKS design to a Genesis database. The `write_gns` command does not make any changes to the PKS database. It does make changes to the Genesis database depending on the presence or absence of the `-hier` argument:

- With the `-hier` argument, a view (*`view_name`*) is created for the top level of the design and any lower-level clusters, with the exceptions of hard blocks. The Genesis view for the top level instantiates the second level. The second level view instantiates the third level, and so forth. Hard blocks are always represented by abstract views and cannot be modified.
- Without the -hier argument, a view (view_name) is created for the top level of the design. Abstract views (using a LEF model description) are created for any second-level soft blocks. The Genesis view for the top level instantiates any second-level soft blocks. Second-level soft blocks cannot be modified, but their placement can be changed within the top level.

Note: If the Genesis view already exists, it is overwritten.

Options and Arguments

`-create_gns_lib`

Calls the `write_gns_lib` command as follows:

```
write_gns_lib -gns_library lib -gns_techlib
techLib -gns_lib_path ./lib
```

`-gns_lib` *`lib_name`*

Specifies the Genesis library to which the command should write the design.

`-gns_tech` *`tech_lib_name`*

Specifies which technology library the command should use.

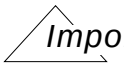
`-gns_view` *`view_name`*
: Specifies the name to use for the Genesis view.

`-hier`
: Creates Genesis views for all levels of a hierarchical design.

`-placement {io|cells|all}`
: Outputs the *x*, *y* coordinates for all cell and IO pins, cells only, or IO pins only in the design.

write_gns_lib

```
write_gns_lib -gns_library lib_name -gns_techlib tech_lib_name
     -gns_lib_path lib_path
```



The Genesis database is no longer supported. This command will be removed in a future release.

Creates the Genesis standard cells libraries and the associated technology file and writes them to the Genesis database.

Options and Arguments

`-gns_lib` *`lib_name`*

Specifies the Genesis library to which the command should write the design.

`-gns_lib_path` *`lib_path`*

Specifies the path the command should use.

`-gns_tech` *`tech_lib_name`*

Specifies which technology library the command should use.

write_layer_usages

`write_layer_usages [-autoLUT] [-name` *`table_name`*`] `*`filename`*

Lets you write out the layer usages tables stored in the system. If the `-name` argument is used, only the named table is written into the specified *`filename`*. If the `-name` argument is not used, all loaded layer usages tables are written out. Note that this does not include the initial default LUTs. Use this command anytime after issuing the `read_layer_usages` command.

Options and Arguments

`-autoLUT`

Dumps automatically generated LUTs (such as those used for `over_block_routing`).
Note: Cadence recommends that you use these tables for informational purposes only.

`filename`

Specifies the name of the file that will contain the layer and contact usages information.

`-name` *`table_name`*

States that only the specified table data is written to the specified *`filename`*. If not used, all loaded tables and default values are written to the specified *`filename`*.

Related Information

create_layer_usages_table

read_layer_usages

set_layer_usages_table

set_net_physical_attribute

write_pdef

```
write_pdef pdf_file_name [-bbox_rect]
     [-location_ieee_1481] [-proprietary_avant_location]
     [-cluster hier_cluster_name] [-version version_tag]
     [-divider hierarchy_divider_char]
     [-pin_delimiter pin_delimiter_char]
     [-bus_delimiter bus_prefix_delimiter_char buf_suffix_delimiter_char] [-
     distance_unit distance_unit]
     [-resistance_unit resistance_unit]
     [-capacitance_unit capacitance_unit] [-computed_rows]
     [-instance instance_name_or_id] [-max_depth depth]
     [-max_soft_block_depth depth] [-xoffset x_offset] [-yoffset y_offset]
```

Note: The global `pdef_style` can be `STANDARD` or `IBM`. The default is `STANDARD`.

Writes a PDEF file conforming to v2.0 syntax. Additional constructs from the IEEE-P1481 syntax include:

- Support for `RECT`, `ROW_ORIENT`, `ORIGIN`, and `CONTENTS_LOCATION` attributes on clusters.
- Support for `PIN` constructs and `LOC` attributes for pins.
- Support for `WireLoad` attributes.

Options and Arguments

`-bbox_rect`
: Represents the cluster bounding box in `RECT` construct. Default is to represent the cluster bounding box in `X_BOUNDS` and `Y_BOUNDS` constructs.

`-capacitance_unit` *`unit`*
: Sets the capacitance unit (specified by *`unit`*) in the output PDEF file.

`-cluster hier_cluster_name`
: Writes out a soft block cluster specified by *`hier_cluster_name`*. Example:
`write_pdef -cluster /clusterA /clusterB`

`depth`
: 0 means unlimited.
Default: unlimited if `-instance` is not specified; 1 level of `soft_block_depth` is specified using the `-instance` option.

`-distance_unit` *`unit`*
: Sets the distance unit (specified by *`unit`*) in the output PDEF file.

`-location_ieee_1481`
: Represents the object orientation and location defined by IEEE 1481-1998. The default value is to represent object orientation and location defined by PDEF 2.0.

`pdef_filename`
: Specifies the name of the PDEF file.

`-proprietary_avant_location`
: Specific to Avant! customers only. Represents the object orientation and location.

`-resistance_unit` *`unit`*
: Sets the resistance unit (specified by *`unit`*) in the output PDEF file.

`-version`
: Supports versions: `ieee`, `IEEE`, and `IEEE 1481-1999` in addition to all previously supported versions.

`-xoffset` *`float`*
: Adds coordinates to instance and pin locations, and x boundaries and RECT of clusters.

`-yoffset` *`float`*
: Adds coordinates to instance and pin locations, and y boundaries and RECT of clusters.

Example

`write_pdef` *`pdef file name`*

write_wdb

`write_wdb` *`wdb_name`*

Writes the wroute database (WDB) to *`wdb_name`*. This command writes LEF, netlist, placement, and final route (if applicable) information to the `.wdb` file.

This command is typically used as follows:

1. Write the `.wdb` file using the `write_wdb` command after final routing your design.
2. Read the `.wdb` file into PKS using the `read_wdb` command.
3. Perform in-place optimization (IPO).
4. Pass the optimization changes and previously stored final route data to a `.wdb` file using the `write_wdb` command.
5. Use the `.wdb` file to do an incremental global and final route of your design.

Options and Arguments

`wdb_name`

Specifies the name of the `.wdb` file.

Related Information

read_wdb

6

Test Synthesis Commands

This chapter describes the commands that you can use with the test synthesis tool.

check_dft_rules

```
check_dft_rules [> file_name | >> file_name]
     [-disable_clock_check]
```

Checks DFT rules for each flip-flop in the design. The flip-flops that pass `check_dft_rules` check are later automatically mapped to scan flip-flops and included in a scan chain during scan connection. The flip-flops that fail `check_dft_rules` are excluded from scan chains.

This command can be run on a generic database, structural database, or on mixed generic/structural designs.

The DFT rule violations which this command checks for are listed in Table 6-1 on page 693. To maximize fault coverage, you should try to fix any DFT rule violations so that all flip-flops can be included in a scan chain. To better understand the costs and benefits of fixing DFT rule violations, see the section Avoiding DFT Rule Violations in the *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)* manual.

DFT assertions which are used by `check_dft_rules` are:

- `set_scan_mode`
- `set_scan_style` (*Default*: `muxscan`)
- `set_must_scan`
- `set_dont_scan`
- `set_test_mode_setup`
- `set_dft_internal_clock_domain`
- `set_dft_transparent`
- `set_global dft_enable_combinational_loop_check`
- `set_global dft_enable_race_condition`

Important

You should run `check_dft_rules` whenever you make changes to the above assertions except for `set_scan_mode`. Otherwise, the changes are not propagated through the design.

Any error messages produced by `check_dft_rules` gives the RTL file name and line number at which the DFT violation occurred. If you are working in the GUI, you can triple-click

the error message itself to view that section of RTL code. You can fix the violation by editing the RTL code directly in the window.

This command has no effect on the design database.

Table 6-1 Checking for DFT Rule Violations

DFT Rule Violation	Check for Violation With
Combinational feedback loops	`set_global dft_enable_combinational_loop_check true`
	`check_dft_rules`
Race conditions	`set_global dft_enable_race_condition_check true`
	`check_dft_rules`
Uncontrollable asynchronous signals	`check_dft_rules`
Gated clocks and derived clocks	`check_dft_rules`
Register's clock port connected to tied lines	`check_dft_rules`
Multiple sequential control functions	`check_dft_rules`
Clocks feeding data path	Not checked
Bus conflicts or floating conditions	Not checked

Options and Arguments

If a file name is specified, then the output report is printed in the specified file.

`-disable_clock_check`

Prevents DFT from checking for clock violations. Clock violations can prevent the tool from looking at and reporting other violations.

`> file_name`

Creates or overwrites the specified file.

`>> file_name`

Appends to the specified file.

Example

Before checking the DFT rules in the design in the following example, `scan_en` is specified as the name of the input port that activates scan mode and the active polarity for the enable signal is set to active-high. The output of `check_dft_rules` is redirected to file `TDRCS`.

```
set_scan_mode scan_en 1
check_dft_rules > ./TDRCs
```

Related Information

`report dft assertions`

`report dft registers`

display_scan_chains

`display_scan_chains [-pks] [-chain` *`n`*`] [`*`module_name`*`]`

Displays the scan chains in the schematic viewer following test synthesis. This command is run on a scan-mapped netlist which was configured on chain mode. You should open the schematic viewer to the appropriate module prior to invoking this command.

If you want to display the scan chains for the current module, you can use the following Tcl command:

```
display_scan_chains [get_names[get_current_module]]
```

You can traverse the hierarchy in the schematic viewer by clicking on a module instance. Repeat the Tcl command to view the scan chains for the newly displayed module.

The `display_scan_chains` command has no effect on the design database.

Options and Arguments

`-chain` *`n`*

Displays only the specified scan chain. *n* corresponds to a chain number.
Default: All scan chains

`module_name`

Specifies the name of the module whose scan chains you want to display.
Default: Current module

Note: The *`module_name`* argument is ignored when you also specify the `-pks` argument because the PKS window shows only the flattened top level design.

`-pks`

Displays the scan chain in the PKS window. The design must be placed before using this option.
Default: Schematic viewer

Examples

- The following command displays the scan chain *2* in module `mymod` in the schematic viewer:

  ```
  display_scan_chains mymod - chain 2
  ```

- The following command displays all scan chains in the current module in the schematic viewer:

```
display_scan_chains [get_names [get_current_module]]
```

do_remove_scan_order_data

```
do_remove_scan_order_data
```

Removes explicit scan order data stored after reading the scan-order file (using the command `read_scan_order_file`) or SCANCHAINS section of the DEF file (using the command `read_def -scan_only`). This is necessary if you want to allow the scan configuration function to freely assign chains according to overall scan constraints instead of by explicit order from the above sources.

This command (`do_remove_scan_order_data`) is not needed to remove scan-order data after a connection cycle if read from the scan-order file. This is done automatically by the connection function.

Note: DEF file scan order data is *not* deleted automatically in this way, however, and will be retained by the system until deleted by this command (or by `do_remove_design`).

Examples

- The following commands remove the scan order data that were stored after reading the `new.scan_order` scan_order file:

```
read_scan_order_file new.scan_order
do_remove_scan_order_data
```

- The following commands removes the annotated scanDEF data after the reordering and connection cycle:

```
read_def -scan_only scan.def
do_place
do_xform_connect_scan -pks
do_remove_scan_order_data
```

Related Information

`do_remove_design`

`do_xform_connect_scan`

`get_dft_config_mode`

`read_def -scan_only`

`read_scan_order_file`

do_xform_connect_scan

```
do_xform_connect_scan [-scan_file file_name] [-pks] [-preserve_config]
```

Configures and connects scan flip-flops into scan chains. The command works at the `set_current_module` level and all lower hierarchies referred to in the current module. The design must be mapped to the target library before connecting scan chains in a design.

There are two methods of getting to scan-equivalent flip-flops:

- If starting with a generic netlist, all flip-flops that pass DFT rule checking are converted to scan flip-flops by the technology mapper, and are connected up into scan chains when the connection engine is run in chain mode.
- If starting with a structural netlist that is mapped to basic D-flops, all flip-flops that pass DFT rule checking are converted to scan-equivalent flops and connected up into scan chains when the connection engine is run in chain mode.

Important

The remapping is restricted to an **unplaced** design. If the design is placed, only those flip-flops that were mapped to scan flip-flops —by a `do_xform_connect_scan` command—prior to placement are connected in scan chains.

The flip-flops that do not pass DFT rule checking are automatically excluded from scan chains. The flip-flops in the lower module hierarchies are also automatically reconfigured and connected, unless the lower module has been attributed with a `dont_modify` or `dont_touch_scan` using commands `set_dont_modify` or `set_dont_touch_scan`, respectively. The scan style used for scan flip-flops is controlled by the `set_scan_style` command.
Default: Scan style is `muxscan`.

DFT assertions or commands which are used by `do_xform_connect_scan` are:

- `set_scan_mode`
- `check_dft_rules`
- `set_scan_data`
- `set_number_of_scan_chains`
- `set_max_scan_chain_length`
- `set_dft_compatible_clock_domains`
- `set_dont_touch_scan`

- `set_dont_modify [find -module]`
- `read_def -scan_only`
- `read_scan_order_file`
- `set_global dft_scan_path_connect`
- `set_global dft_scan_enable_connect`
- `set_global dft_scan_output_pref`
- `set_global dft_scan_avoid_control_buffering`
- `set_global dft_allow_scan_path_inv`
- `set_global dft_min_scan_wire_length`
- `set_global dft_scan_port_name_prefix`

Scan Configuration

When creating a configuration, all scan flip-flops are first configured into different chains based on the DFT assertions, such as

- Maximum scan chain length
- Number of scan chains
- Sets of compatible clock domains (when mixing different clock domains in a single chain in mux scan style)
- Fixed scan segments (that is, scan segments belonging to modules marked `dont_modify` or `dont_touch_scan`).

In the absence of specifying any of the above test assertions, the connection engine will create a single scan chain for each clock domain and clock phase identified by `check_dft_rules`. If the user reads in a specific scan chain configuration (using `read_scan_order_file` or `read_def -scan_only`), then the user-specified configuration is applied during chain synthesis.

To analyze and maintain the existing scan chain configuration of a structural netlist by PKS reordering, the scan connection engine must be run using the `-preserve_config` option.

The `-preserve_config` option is used in conjunction with the `-pks` option which instructs the connection engine to perform placement-based PKS reordering. Both options require a PKS license.

When using the `-pks` option, the placement information about scan flip-flops is used to configure the chains to minimize wire length due to routing the scan data shift path. The design must be placed before `-pks` option can be used in scan connection.

If scan chains contain multi-input gates on the scan path, then set the DFT global variable: `set_global dft_stop_analysis_at_complex_logic` prior to running the scan connection engine.

Scan Data Connection

Scan flip-flops are connected up according to the global setting on the `dft_scan_path_connect` global variable, which can be one of the following:

- `chain`—connects scan flip-flops in chain mode
- `tieback`—ties scan-in of a flip-flop to its own scan-out pin
- `floating`—leaves the scan-in of flip-flop unconnected.

  Default: `chain`

Under chain mode, the following two additional globals control how the chains will be connected.

- `dft_scan_output_pref`—controls whether to use normal or inverted output (Q or Q-bar) of scan flip-flop for scan connection. The possible settings are:
  - `non_inv`—uses non-inverted (Q) output
  - `min_load`—uses the output with least load.

  Default: `min_load`
- `dft_allow_scan path_inv`—controls if inverters should be inserted in scan path to compensate for logic inversion when connecting from inverted output or to inverted scan-input. The possible settings are:
  - on—inversion in scan path is allowed.
  - `off`—inversion in scan path is not allowed. When connecting from inverted output (Qbar) or to inverted scan-input, an inverter will be inserted to compensate for logic inversion on the data path.

  Default: on

Under chain mode, the chains get connected to the specified scan-in and scan-out pins (using `set_scan_data` command). If the scan-out of a chain is shared with a functional

output pin, a multiplexor, controlled by the scan-enable signal, is inserted to share the pin between functional output and scan output. For more information, refer to `-sharedout` option in the `set_scan_data` command.

In mux scan style, if scan flip-flops triggered by different edges of a clock or by different clock domains are to be connected in the same chain having specified the `set_dft_compatible_clock_domains` assertion, then lock-up latches would be automatically inserted to prevent clock-skew problems.

Scan Control Connection

When creating a scan chain configuration, the scan mode signal (scan-enable signal) is connected according to the setting of global variable `dft_scan_enable_connect`. The allowed settings are:

- on—Connect scan-enable pin of flip-flop to the specified scan-enable signal.
- `tieoff`—Connect scan-enable pin of flip-flop to its inactive value (power/ground).
- `floating`—Leave scan-enable unconnected.

  Default: on

If you do not specify scan mode pin (using `set_scan_mode` command) or an adequate number of scan data pins (using `set_scan_data` command), then scan synthesis will create new pins at the “current module” to drive scan mode and scan data nets.

The name prefix for these pins can be controlled with the following command: `set_global dft_scan_port_name_prefix`.

Default: `BG_scan`

When importing a structural netlist which had previously undergone test synthesis, the scan mode network will be preserved when the connection engine is run with the `-preserve_config` option in PKS mode.

Options and Arguments

`-pks`

Orders and connects each chain according to the proximity of flip-flops and the placement information from the PKS tool. Optionally, the user may specify the DFT global variable `dft_min_scan_wire_length` prior to invoking PKS reordering. This global instructs PKS to connect successive

registers along a scan chain according to a minimum wire length distance specified by the user which helps in minimizing hold time violations along the scan chain path.

Note: The `-pks` option requires a Cadence® Physically Knowledgeable Synthesis (PKS) software license. The design must be placed before performing placement-based scan chain ordering using the `do_place` command. The units of distance are specified in units consistent with the LEF file (typically microns).
Default: 0

`-preserve_config`

Preserves the analyzed configuration of existing scan chains in scan mapped netlists during PKS-based scan reordering (`-pks` option). Any additional configuration assertions that you specify, such as `set_number_of_scan_chains` or `set_max_scan_chain_length`, and TDRC data (output of `check_dft_rules`) are ignored. The `-preserve_config` option is the default mode used by the `do_place -scan_reorder` command.

`-scan_file` *`file_name`*

Specifies the output file name for the scan order file.
Default: Generates the flat scan order file: *`top_module`*`.scan.flat`

Examples

- The following commands create a configuration (two scan chains) prior to placement:

```
set_number_of_scan_chains 2
do_xform_conect_scan -scan_file ./SOF
```

- The following commands create a configuration (based on the maximum scan chain length) after placement:

```
set_max_scan_chain_length
do_place
do_xform_connect_scan
```

- The following commands order and connect the chains based on placement information using PKS while preserving the existing configuration:

```
do_place
set_global dft_min_scan_wire_length 5.0
do_xform_connect_scan -pks -preserve_config
```

```
do_xform_connect_scan
```

Related Information

check_dft_rules

do_place

write_scan_order_file

do_xform_fix_dft_violations

```
do_xform_fix_dft_violations
     [-preview | -dont_check_dft_rules]
     [> file_name | >> file_name]
```

Automatically fixes all asynchronous set and reset violations that you have specified to fix via the set_dft_fix_violations command. Any violation without a user-specified assertion is skipped.

Unless -preview is specified, this command transforms the DFT-fix assertions into actual netlist modifications. Consequently, after its successful execution, any existing DFT-fix assertions are cleared. This avoids re-applying a DFT fix more than once to a particular violation. In addition, the DFT-rule checker is automatically invoked after fixing the violations, unless -dont_check_dft_rules is specified. This allows the tagging of the fixed flip-flops as being feasible for scan insertion.

If you use do_xform_fix_dft_violations with the -dont_check_dft_rules option, you must eventually run check_dft_rules before running scan connection with the do_xform_connect_scan command to include all scannable flip-flops in scan chains.

Note: Currently, only DFT violations related to asynchronous set and reset are fixed. Clock violations cannot be fixed yet.

Options and Arguments

-dont_check_dft_rules
: Prevents do_xform_fix_dft_violations from checking DFT rules immediately after fixing the violations. You can use this option if you do not want to fix all violations at once and you want to use the DFT rule violation information, such as violation codes, provided by a previous execution of check_dft_rules.

> *file_name*
: Directs he report generated by the command to the specified file. If the file does not exist, it is created. If the specified file exist, it will be overwritten.

>> *file_name*
: Appends the report generated by the command to the specified file.

`-preview`

Displays the control points that will be inserted, but makes no modifications to the netlist.

Example

The following commands specify to fix the asynchronous reset violations on all instances with this type of violation using the test mode signal `TM`:

```
> set_dft_fix_violations -test_mode TM -async_reset -all
> do_xform_fix_dft_violations
     Successfully inserted common test-point at node C/<reset>. It covers 2 nodes:
          C/<reset>
          D/<reset>
     Successfully inserted test-point at node A/<reset>.
     Successfully inserted test-point at node B/<reset>.
     Total number of violation points: 4
     Total number of test points inserted: 3
```

Note: The report indicates the number of violation points, such as flip-flop pins, and not the number of violations reported when running `check_dft_rules`.

Related Information

`check_dft_rules`

`reset_dft_fix_violations`

`set_dft_fix_violations`

Tasks for Automatic Fixing of the Design Rule Violations in *Design for Test (DFT) Using BuildGates Synthesis and Cadence PKS*.

do_xform_insert_shadow_dft

```
do_xform_insert_shadow_dft -around instance
     [-share | -bypass] [-sharing_groups groups]
     [-inputs ports] [-outputs ports]
     [-exclude ports | -only ports] [-test_mode signal]
     [-test_clock clock [-rise|-fall]] [-preview]
```

Allows to insert two basic types of DFT shadow logic around a particular instance: bypass and scannable logic (see Figure 6-1 on page 709 and Figure 6-2 on page 710 for simple examples). The insertion of scannable logic requires three steps (two steps more than when you insert bypass logic):

- Insertion of the DFT shadow logic (performed by this command)
- Mapping the DFT shadow flip-flops to scannable flip-flops
- Connecting the scan-chains

Each shadow logic flip-flop can implement one control point and one observation point at the same time (see Figure 6-2 on page 710).

In general, sharing results in smaller gate count, but this not always translates into smaller area. If the points to observe and control are close, sharing them in a same flip-flop will give smaller area and less routing. However, if the points are far away, the additional routing and congestion result in an overall worse solution. Because deciding which points to share is not straightforward, the default behavior is not to share them. Choosing to insert bypass logic implicitly assumes sharing.

If you want to share observation and control points, either using `-share` or `-bypass`, the following sharing criteria are observed:

- If you specified `-sharing_groups`, the specified inputs and outputs are grouped together as indicated
- For the remaining inputs and outputs that are not listed with `sharing_groups`, the first input will share the flip-flop with or be connected to the first output, the second input with the second output, and so on. The order is that specified in the HDL interface declaration.

Options and Arguments

`-around` *`instance`*

Specifies the instance around which the DFT shadow logic must be inserted. Specify either a hierarchical instance name or an object identifier.

`-bypass`

Implements bypass logic instead of observation and control points. If you specify this option, you must balance the number of inputs and outputs.

`-exclude` *`ports`*

Prevents the specified ports from being considered for shadow logic insertion. Specify either port names or port identifiers.

`-inputs` *`ports`*

Specifies the bidirectional ports of the untestable module that must be considered as inputs.

Use this option when the instance specified by the `-around` option is a black box and, hence, all its interface ports are assumed to be bidirectional.

`-only` *`ports`*

Restricts the ports to be considered for shadow logic insertion to the specified ports. Specify either port names or port identifiers.

`-outputs` *`ports`*

Specifies the bidirectional ports of the untestable module that must be considered as outputs. Specify either port names or port identifiers.

Use this option when the instance specified by the `-around` option is a black box and, hence, all its interface ports are assumed to be bidirectional.

`-preview`

Shows the potential changes, without making any modifications to the netlist.

`-rise | -fall`

Specifies the edge of the `-test_clock` that is active during test mode operation. These options are only valid in conjunction with `-test_clock`.
Default: `-rise`

`-share`

Shares shadow flip-flops for control and observation points.

`-sharing_groups` *groups*

Specifies the inputs and outputs that must share an observation and control point. Each group can have multiple input pins and multiple output pins. Format the groups as follows:

{ {$input_i$ ... $output_i$...} ... }

Separate the pins by spaces and enclose each group by braces ({}). If you have more than one group, you must enclose the list of groups by braces or double quotes.

The inputs of a group are applied to an XOR. Groups with multiple outputs result in the insertion of one multiplexer per output.

Note: If you use the `-bypass` option, each group must have at least one input and one output. Otherwise, the number of inputs or outputs can be equal or larger than zero.

`-test_clock` *clock*

Specifies the signal that drives the clock pin of the flip-flops implementing the scannable observation points. If you do not specify this option, the tool issues a warning and connects the flip-flops to the first available clock domain.

`-test_mode` *signal*

Specifies the signal to use to control the multiplexers after the controlling points.Specify either a previously declared test-mode signal, a top-level port or a hierarchical pin. If the signal does not exist, a warning message will be issued. This option is not (necessarily) related to the test-mode signal used at a later stage by the scan-chain connection engine. scan-mode signal may later on be connected to the observation points.That test-mode (or scan-mode) signal might or might not be the same as the one specified by this `-test_mode` option.

Examples

In the following example, the logic before the ATPG-untestable module is not observable and the logic after it is not controllable. Following is the Verilog input code for the ATPG-untestable module and its instantiation:

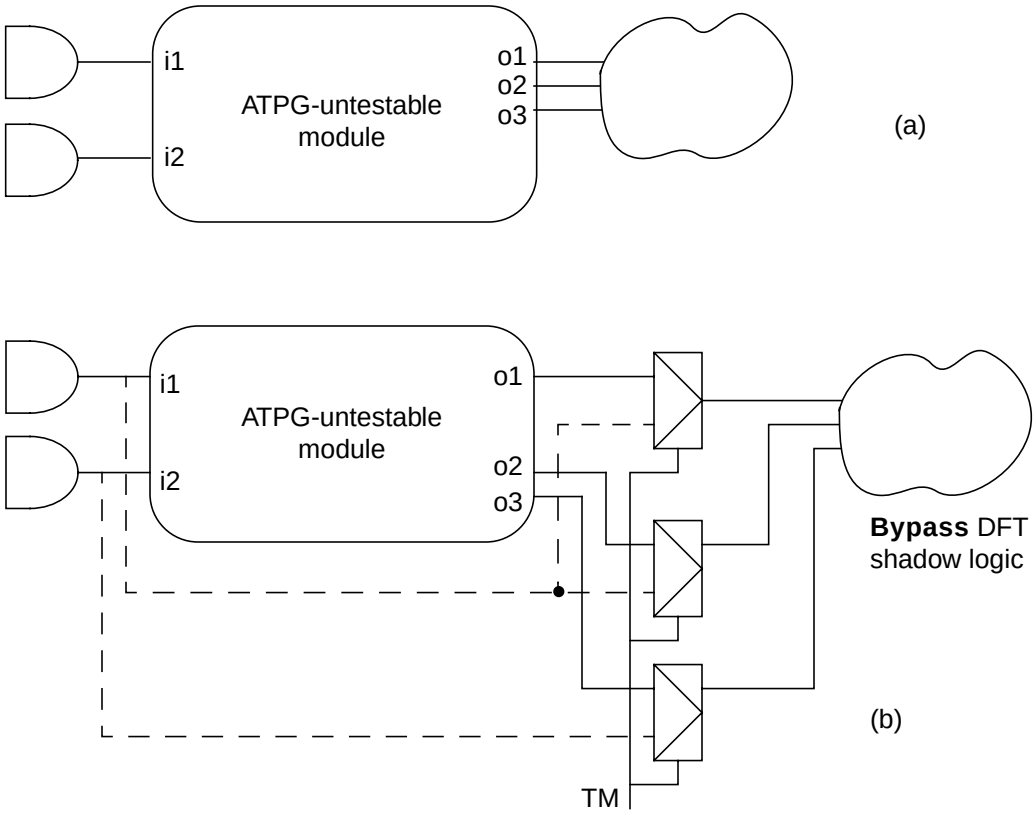
```
module Untestable (i1,i2, o1,o2,o3)
     input i1,i2;
```

```
     output o1,o2,o3;
...
Untestable U1 (.i1(n_1), .i2(n_2), .o1(n_3), .o2(n_4), .o3(n_5));
...
```

In the following example, bypass logic is used to make the two inputs observable and the three outputs controllable. The first command pairs input `i1` to output `o1,` and input `i2` to output `o3` (skipping `o2`).The second command pairs input `i1` and output `o2`. The result is shown in Figure 6-1 (b).

```
do_xform_insert_shadow_dft -around U1 -test_mode TM -bypass -exclude o2
do_xform_insert_shadow_dft -around U1 -test_mode TM -bypass -only {i1 o2}
```

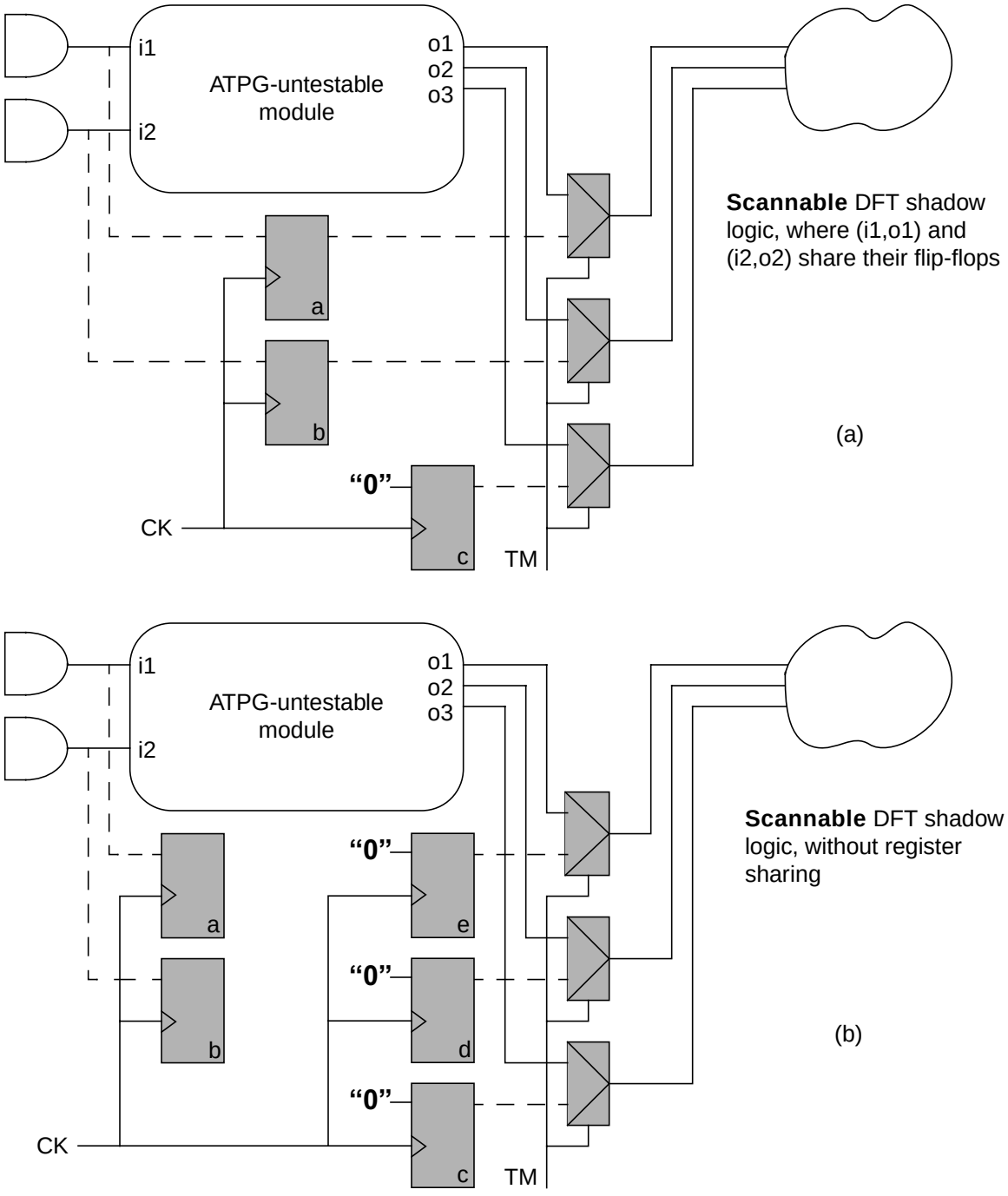
Figure 6-1 Insertion of Bypass DFT Shadow Logic



- The following example uses scannable registers and shares shadow flip-flops for control and observation points. Figure 6-2(a) shows that `i1` and `o1,` and `i2` and `o2` share the shadow flip-flop. In total, 3 flip-flops are needed.

  ```
  do_xform_insert_shadow_dft -around U1 -test_mode TM -test_clock CK -share
  ```

Figure 6-2 Inserting Scannable DFT Shadow Logic



- The following command uses scannable registers but does not share the shadow flip-flops for control and observation points. As a result, five flip-flops are needed.

```
do_xform_insert_shadow_dft -around U1 -test_mode TM -test_clock CK
```

Figure 6-2(b) shows the result.

- The following command uses scannable registers, shares shadow flip-flops for control and observation points, and controls which pins share a flip-flop. More specifically, `i1` and `o2` share a flip-flop, and `i2` and `o1`. In addition, no control point is inserted for the net driven by `U1/o3`.

```
do_xform_insert_shadow_dft -around U1 -test_mode TM -test_clock CK \
      -exclude o3 -share -sharing_groups {{i1 o2} {i2 o1}}
```

Related Information

Inserting DFT Shadow Logic in the *Design for Test (DFT) Using BuildGates Synthesis and Cadence PKS* manual.

do_xform_insert_testpoint

```
do_xform_insert_testpoint -location lsignal
     [-input | -output]
     [-observe_scan -test_clock clock [-rise|-fall] ]
     [-control_0 | -control_1 | -control csignal
     |-control_scan -test_clock clock [-rise|-fall] ]
     -test_mode tsignal [-preview]
```

Allows you to manually specify a control or observation test point to be added to the design. Control test points always require the specification of a test-mode signal. Test points that use scannable flip-flops to observe or control a node always require a test-clock signal.

If you insert a control or observation scan flip-flop, you need to run `check dft rules` to validate the control signals. You also need to run the scan connection engine to connect the flip-flops to a scan chain.

Options and Arguments

`-control_0`

Inserts a constant value `0` at the specified location when the signal specified by `-test_mode` is active.

`-control_1`

Inserts a constant value `1` at the specified location when the signal specified by `-test_mode` is active.

`-control` *`csignal`*

Inserts an arbitrary node, *`csignal`*, at the specified location when the signal specified by `-test_mode` is active.

`-control_scan`

Inserts a scan flip-flop to force a particular value at the specified location during test mode operation. The scan flip-flop is connected to a scan chain later on.

Note: This option requires you to specify the `-test_clock` option

`-input`

Specifies that the signal specified with `-location` should be considered as an input.

`-location` *`lsignal`*

Specifies the location of the control point or observation point. Specify an existing hierarchical pin name or an I/O (boundary port at the top or a lower level. For observation test points, the pin can be an input or output pin. For control test points, the result is different depending on the direction of the location. See the Examples on page 714.

Note: If you specify a bidirectional pin, no logic will be inserted unless you specify the direction of the pin.

`-observe_scan`

Inserts a scan flip-flop to observe the specified location. The scan flip-flop is connected to a scan chain later on.

Note: This option requires you to specify the `-test_clock` option

`-output`

Specifies that the signal specified with `-location` should be considered as an output.

`-preview`

Shows the potential changes, without making any modifications to the netlist.

`-rise | -fall`

Specifies the edge of the `-test_clock` that is active during test mode operation. These options are only valid in conjunction with `-test_clock`.
Default: `-rise`

Note: You must use the same clock edge for both the control scan and the observe scan.

`-test_clock` *`clock`*

Specifies the signal that drives the clock pin of the inserted scan flip-flops during test mode operation. This option is required when you specify either `-control_scan` or `-observe_scan`

Note: You must use the same clock signal for both the control scan and the observe scan.

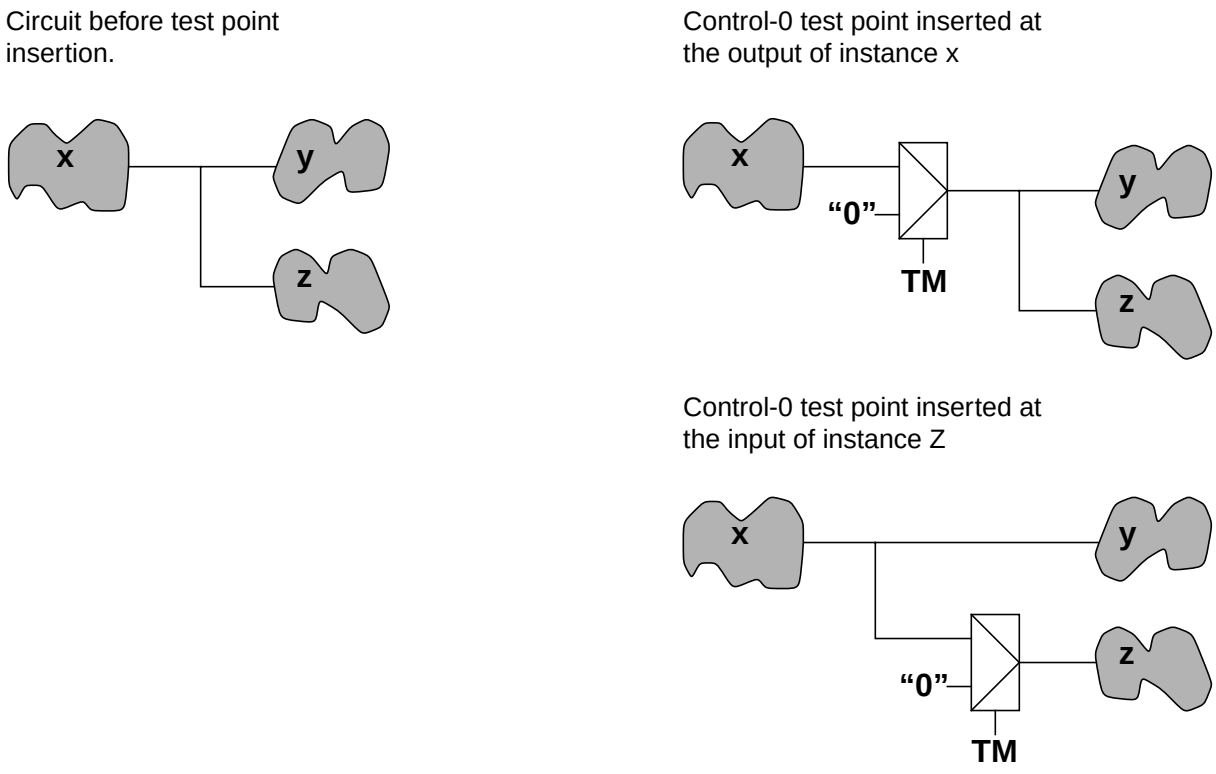
`-test_mode` *`tsignal`*

Specifies the signal that allows controlling or observing the specified location point. Specify either a previously declared test-mode or scan-mode signal, a top-level port, or a hierarchical pin. If *`tsignal`* does not exist, a warning message is issued.

Examples

Consider a net with multiple sinks. If the location signal is specified on an output pin, the test point is inserted at the beginning of the fork. If the location is specified on an input pin, the test point is inserted in such a way that only the end of the fork is controllable. Both situations are illustrated in Figure 6-3 on page 714.

Figure 6-3 Location of Test point Insertion Depends on Type of Used Pin



- The following command inserts a control-0 test point at hierarchical pin `X/out`:

```
do_xform_insert_testpoint -location X/out -test_mode TM -control_0
```

- The following command inserts a control-1 test point at hierarchical pin `X/out`:

```
do_xform_insert_testpoint -location X/out -test_mode TM -control_1
```

- The following command inserts a control signal `csignal` at hierarchical pin `X/out`:

```
do_xform_insert_testpoint -location X/out -test_mode TM -control csignal
```

- The following command inserts a scannable observation test point, using `CLK` to drive:

```
do_xform_insert_testpoint -location X/out -test_clock CLK -observe_scan
```

- The following command inserts a control-1 and scannable observation point:

```
do_xform_insert_testpoint -location X/out -test_mode TM \
-test_clock CLK -control_1 -observe_scan
```

- The following command inserts a scannable control point:

```
do_xform_insert_testpoint -location X/out -test_mode TM \
-test_clock CLK -control_scan
```

- The following command inserts a scannable control and observation test point:

```
do_xform_insert_testpoint -location X/out -test_mode TM \
-test_clock CLK -control_scan -observe_scan
```

Related Information

`check_dft_rules`

`do_xform_connect_scan`

`do_optimize`

Types of Test Points in the *Design for Test (DFT) Using BuildGates Synthesis and Cadence PKS* manual.

get_dft_config_mode

```
get_dft_config_mode
```

Returns the configuration mode established in the design database. The following keywords are returned:

- `scandef`—Indicates connection engine is using scanDEF data (from `read_def` command).
- `scanorder`—Indicates connection engine is using scan order file data (from `read_scan_order_file` command).
- `normal`—Normal BG configuration mode enabled (none of the above modes are active).

Related Information

`do_remove_scan_order_data`

`read_def`

`read_scan_order_file`

get_scan_chain_info

```
get_scan_chain_info [-test_setup | -scan_setup
     | -count | -enable | -clock [scan_chain_index]
     | -length [scan_chain_index]
     | -in [scan_chain_index]
     | -out [scan_chain_index] ] [-name]
```

Returns the information on scan chains in “current module” configured in a design database which has undergone test synthesis. This command is also used by the `write atpg info` command to generate an ATPG interface file.

Options and Arguments

`-clock` *scan_chain_index*
: Gets the pin ID of the system clock port of the specified chain and its active edge. The active edge is reported as `0` for negative edge, `1` for positive edge. If the `-name` option is used, the command returns the pin `name` rather than the ID.

  Valid values for *scan_chain_index* are 1 through the number of scan chains.
  Default: `1`

`-count`
: Gets the total number of scan chains.

`-enable`
: Gets the scan-enable pin ID and its active value. If the `-name` option is used, the command returns the pin `name` rather than the ID.

`-in` *scan_chain_index*
: Gets the pin ID of the scan input port of the specified chain. Valid values for *scan_chain_index* are 1 through the number of scan chains. If the `-name` option is used, the command returns the pin `name` rather than the ID.
  Default: `1`

`-length` *scan_chain_index*
: Gets the length of the specified chain. Valid values for *scan_chain_index* are 1 through the number of scan chains.
  Default: `1`

`-name`

Returns `name` of what you are looking for instead of pin ID.

`-out` *`scan_chain_index`*

Gets the pin ID of the scan output port of the specified chain. Valid values for *`scan_chain_index`* are 1 through the number of scan chains. If the `-name` option is used, the command returns the pin `name` rather than the ID.
Default: `1`

`-scan_setup`

Returns a list of sublists, where each sublist consists of a port ID and its active value that is set during the scan-shift cycle of the test mode.

`-test_setup`

Returns a list of sublists, where each sublist consists of a port ID and its active level that needs to be set during test mode (throughout the test session). If the `-name` option is used, the command returns the pin name rather than the ID.

Examples

- The following command returns the number of scan chains (including scan chain segments) configured in the design:

```
get_scan_chain_info -count
```

- The following commands return the length along scan chain 1:

```
get_scan_chain_info -length 1
set clockNameChain2 [lindex [get_scan_chain_info -clock 2 -name] 0]
set clockLogicValueChain2 [lindex [get_scan_chain_info -clock 2 -name] 1]
```

- The following command returns `sen` as the input port for the test mode signal and 1 as the value assigned to the test mode signal during the `scan_shift` cycle of test mode:

```
set_test_mode_setup sen 1 -scan_shift
Info: Test mode for module ‘test’ set to sen=1 during scan shift only.
get_scan_chain_info -scan_setup -name
```

Related Information

`do_xform_connect_scan`

`write_atpg_info`

read_scan_order_file

```
read_scan_order_file file_name
```

Reads a scan order data file and initializes the scan order chain configuration in the design database. The scan order file controls the configuration and specific ordering of instances and scan chains throughout the hierarchy. In this mode, all other scan configuration assertions, such as `set_number_of_scan_chains` and `set_max_scan_chain_length` will be ignored.

The test synthesis tool creates a scan order data file when you run the `do_xform_connect_scan`, `do_optimize`, or `write_scan_order_file` commands. *Default*: The file type is written in flat format using the file name *`top_module`*.scan.flat.

You can create or edit a scan order data file and define the order in which registers are to be connected along the scan chains by using the `read_scan_order_file` command.

If the connection engine is run with the `-pks` flag, PKS reordering of the flops will also occur. Scan Order File (SOF) data is automatically removed after a connection cycle.

Options and Arguments

`file_name`	Specifies the name of the scan order data file.

Examples

- The following commands first read in scan order data file `new.scan_order`, then configure and connect the scan flip-flops into scan chains before placement:

```
read_scan_order_file new.scan_order
do_xform_connect_scan
```

- The following commands read in scan order data file `new.scan_order`, place the design, then configure, connect and reorder the scan chains. After the reordering, the scan order file data is removed.

```
read_scan_order_file new.scan_order
do_place
do_xform_connect_scan -pks
```

Related Information

`do_remove_scan_order_data`

`do_xform_connect_scan`

remove_dft_assertions

```
remove_dft_assertions [-all] [-check_dft_rules_data]
     [-compatible_chains] [-compatible_clock_domains]
     [-config_constraints] [-configuration_constraints]
     [-dft_fix_violations] [-dft_transparent]
     [-dont_scan] [-dont_touch_scan] [-lockup_element]
     [-internal_clock_domains]
     [-must_scan] [-scan_chain] [-scan_chain_segment]
     [-scan_data_io] [-scan_mode] [-test_mode_setup]
```

Allows the user to selectively remove all or any of the DFT assertions specified on the design database.

Options and Arguments

`-all`
: Removes all DFT assertions.

`-check_dft_rules_data`
: Removes DFT rule check data generated by the `check_dft_rules` command.

`-compatible_chains`
: Removes all scan chain groupings set using the `set_dft_compatible_chains` command.

`-compatible_clock_domains`
: Removes the `-compatible_clock_domain` assertion set by `set_dft_compatible_clock_domains`.

`-config_constraints`
: Removes configuration constraints related to chain length and number of scan chains as set by `set_max_chain_length` and `set_number_of_scan_chains`.

`-configuration_constraints`
: Works the same as `-config_constraints` above.

`-dft_fix_violations`
: Removes all DFT assertions set with `set_dft_fix_violations`.

`-dft_transparent`
Removes the `-dft_transparent` assertion set by `set_dft_transparent`.

`-dont_scan`
Removes the `-dont_scan` assertion set by `set_dont_scan.`

`-dont_touch_scan`
Removes the `-dont_touch_scan` assertion set by `set_dont_touch_scan.`

`-internal_clock_domains`
Removes the `-internal_clock_domains` assertion set by `set_dft_internal_clock_domain`.

`-lockup_element`
Removes the `-lockup_element` assertion set by the `set_dft_lockup_element` command.

`-must_scan`
Removes the `-must_scan` assertion set by `set_must_scan.`

`-scan_chain`
Removes the `-scan_chain` assertions set by the `set_scan_chain` command. For consistency, it also removes all segments declared by the `set_scan_chain_segment` command.

`-scan_chain_segment`
Removes the `-scan_chain_segment` assertion set by `set_scan_chain_segment` command. It also removes the `-scan_chain` assertions set by the set_scan_chain command, as the assertions in the `set_scan_chain` command refer to the segments specified by set_scan_chain_segment.

`-scan_data_io`
Removes the `-scan_data_io` assertion set by `set_scan_data.`

`-scan_mode`
Removes the `-scan_mode` signal assertion set by `set_scan_mode.`

`-test_mode_setup`

Removes the `-test_mode_setup` attributes set by `set_test_mode_setup.`

In previous releases (with the exception of configuration assertions `set_number_of_scan_chains` and `set_max_scan_chain_length`, or `set_scan_mode`), the remaining DFT assertions could be selectively removed using the following commands:

```
reset_dft_compatible_clock_domains
reset_dft_transparent
reset_dont_scan
reset_dont_touch_scan
reset_dft_internal_clock_domain
reset_must_scan
reset_scan_data
reset_test_mode_setup
```

Example

The following commands remove configuration constraints related to the number of scan chains set by `set_number_of_scan_chains`:

```
set_number_of_scan_chains 5
report_dft_assertions
remove_dft_assertions -config_constraints
```

Related Information

`check_dft_rules`

`set_dft_compatible_chains`

`set_dft_compatible_clock_domains`

`set_dft_fix_violations`

`set_dft_internal_clock_domain`

`set_dft_lockup_element`

`set_dft_transparent`

`set_dont_scan`

report_dft_assertions

```
report_dft_assertions
     [-module {module_id | module_name}]
     [> file_name | >> file_name]
```

Displays the current settings of the DFT assertions and configuration constraints. When you do not use the `-module` option, `report_dft_assertion` displays the DFT assertions for the current module and all the modules below it, and it gives a report of the current settings.

Options and Arguments

`> file_name` Creates or overwrites the specified file.

`>> file_name` Appends to the specified file.

`-module {module_id | module_name}`
Reports DFT assertions for the specified module.
Default: Current module

Examples

- The following commands fix the asynchronous set and reset violations on instances R1 and R2 using the test mode signal `TM`. and then reports the current settings of the DFT assertions and configuration constraints for the current module:

```
> set_test_mode_setup TM 0
> set_dft_fix_violations -async_set -async_reset -test_mode TM -instance R1 R2
> report_dft_assertions
Info: Test mode for module 'top' set to 'TM = 0' during entire test session.
Info: Async. set/reset dft-fix violation assertions:
      R1/<set>: controlled by TM.
      R1/<reset>: controlled by TM.
      R2/<set>: controlled by TM.
      R2/<reset>: controlled by TM.
Info: Clock dft-fix violation assertions:
      None.
```

- The following commands specify the input port for the test mode signal and the input port that activates scan mode before reporting the current settings of the DFT assertions and configuration constraints for the module `top`:

```
set_current_module top
set_test_mode_setup u_jtag/test_mode_out 1
set_scan_mode u_jtag/scan_enable 1
report_dft_assertions
Info: Scan mode for module 'top' set to 'u_jtag/scan_enable = 1'
```

Related Information

set_dft_compatible_clock_domains

set_dft_compatible_chains

set_dft_fix_violations

set_dft_internal_clock_domain

set_dft_transparent

set_dont_scan

set_dont_touch_scan

set_lssd_aux_clock

set_lssd_scan_clock_a

set_lssd_scan_clock_b

set_max_scan_chain_length

set_must_scan

set_number_of_scan_chains

set_scan_chain

set_scan_chain_segment

set_scan_data

set_scan_mode

set_scan_style

set_test_mode_setup

set_test_scan_clock

report_dft_registers

```
report_dft_registers [-scan] [-non_scan] [-latch]
     [> file_name | >> file_name]
```

Reports whether all register instances are scanned or non-scanned, and the clock domain in which they exist. Use this command after running `check_dft_rules`.

- For scannable registers, it reports the clock domain.
- For non-scannable registers, it indicates the reason for their exclusion from scan (that is, the DFT rule that was violated).

You can request information about only the scannable registers or only the non-scannable registers.

Non-scannable latch instances can be reported using the `-latch` option.

Additionally, the `report_dft_registers` command reports the mapping/scan status of all register instances in the design. The following status flags apply:

- `<mapped to invalid scan register>`

  Scan style of registers does not match scan style specified using `set_scan_style` command.

- `<unmapped register>`

  Not yet mapped to a register from the target technology library.

- `<mapped to non-scan register>`

  Will be converted to scan equivalent register, if the register passes the DFT rule checking. The conversion of the register to its scan equivalent flop occurs when the scan chain configuration is being created in the design.

- `<mapped (for-DFT) to scan register>`

  Mapped to scan register from the target technology library

- `<mapped (NOT-for-DFT) to scan register>`

  Mapped to scan register but likely to be mapped to scan for functional logic purposes rather than for DFT. This status flag could also be obtained, if the user did not identify all scan mode signals and/or their correct active logic value specified using the `set_scan_mode` assertion.

Options and Arguments

`> file_name`
: Creates or overwrites the specified file.

`>> file_name`
: Appends to the specified file.

`-latch`
: Includes latches in the report.
Default: The latches are excluded from the report since the tool does not perform scan insertion on level-sensitive latches.

`-non_scan`
: Reports only the non-scan registers.

`-scan`
: Reports only the scan registers.

Example

The following commands report the scan registers.

```
check_dft_rules
report_dft_registers -scan
        Scan register instances:
        ========================
    ireg_reg_4  -> [clock-domain 0] <unmapped_register>
    ireg_reg_3  -> [clock-domain 0] <unmapped_register>
    ireg_reg_5  -> [clock-domain 0] <unmapped_register>
    ireg_reg_2  -> [clock-domain 0] <unmapped_register>
    ireg_reg_1  -> [clock-domain 0] <unmapped_register>
    submod2i/creg_reg_4         -> [clock-domain 0] <unmapped_register>
    submod2i/creg_reg_3         -> [clock-domain 0] <unmapped_register>
    submod2i/creg_reg_2         -> [clock-domain 0] <unmapped_register>
    submod2i/creg_reg_1         -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/breg_reg_0        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/breg_reg_1        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/breg_reg_2        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/breg_reg_3        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/breg_reg_4        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/breg_reg_5        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/breg_reg_6        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/areg_reg_0        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/areg_reg_1        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/areg_reg_2        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/areg_reg_3        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/areg_reg_4        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/areg_reg_5        -> [clock-domain 0] <unmapped_register>
    submod2i/bottom2i/areg_reg_6        -> [clock-domain 0] <unmapped_register>
    submod1i/do_reg_0   -> [clock-domain 0] <mapped (for-DFT) to scan register>
```

```
    submod1i/do_reg_1   -> [clock-domain 0] <mapped (for-DFT) to scan register>
    submod1i/do_reg_2   -> [clock-domain 0] <mapped (for-DFT) to scan register>
    submod1i/do_reg_3   -> [clock-domain 0] <mapped (for-DFT) to scan register>
    submod1i/do_reg_4   -> [clock-domain 0] <mapped (for-DFT) to scan register>
    submod1i/do_reg_5   -> [clock-domain 0] <mapped (for-DFT) to scan register>
    submod1i/do_reg_6   -> [clock-domain 0] <mapped (for-DFT) to scan register>

    Info:    Total Scannable register count: 30 <DFT-340>.

30
```

Related Information

check_dft_rules

reset_dft_compatible_clock_domains

```
reset_dft_compatible_clock_domains
```

Removes all the previous settings of the `set_dft_compatible_clock_domains` assertion.

You need to specify this command before you connect the scan chains in order to prevent the connection engine from creating chains with merged domains.

Examples

Reconfigure without domain merging.

- The following commands remove all the previous settings of the `set_dft_compatible_clock_domains` assertion, report the current settings of the DFT assertions and configuration constraints for the current module, and reconfigure the scan chains before placement:

  ```
  reset_dft_compatible_clock_domains
  report_dft_assertions
  do_xform_connect_scan
  ```

- The following commands first place the design, then removes all the previous settings of the `set_dft_compatible_clock_domains` assertion before connecting the scan chains:

  ```
  do_place
  reset_dft_compatible_clock_domains
  do_xform_connect_scan -pks
  ```

Related Information

`do_xform_connect_scan`

`remove_dft_assertions -compatible_clock_domains`

`report_dft_assertions`

`set_dft_compatible_clock_domains`

reset_dft_fix_violations

```
reset_dft_fix_violations [-async_set | -async_reset]
     [-clock] [-all | -instance instance_list]
```

Reverses the effects of set_dft_fix_violations on the specified instances.

Options and Arguments

`-all`
: Removes the specified type assertions from all the instances that have them.

`-async_reset`
: Removes the asynchronous reset violation assertions—associated with the specified instances—from the list of violations to be fixed.

`-async_set`
: Removes the asynchronous set violation assertions associated— with the specified instances—from the list of violations to be fixed.

`-clock`
: Removes the clock violation assertions—associated with the specified instances—from the list of violations to be fixed.

`-instance` *instance_list*
: Specifies the instances from which to remove the assertions. You can identify the instances using either their hierarchical instance names or their object identifiers (IDs).

Related Information

set_dft_fix_violations

Tasks for Automatic Fixing of the Design Rule Violations in the *Design for Test (DFT) Using BuildGates Synthesis and Cadence PKS* manual.

reset_dft_internal_clock_domain

```
reset_dft_internal_clock_domain
     { hier_pin_id | hier_pin_name }
```

Removes a previous setting of an internal clock as a separate DFT domain, previously set with the set_dft_internal_clock_domain assertion.

Clocks are specified cumulatively with the set_dft_internal_clock_domain assertion. To remove any of the previous settings, use the reset_dft_internal_clock_domain command. This data applies to the current module.

Before connecting the scan chains, you need to rerun check_dft_rules in order to propagate the changes through the circuit.

Options and Arguments

hier_pin_id
: Specifies an existing instance output pin identifier (ID).

hier_pin_name
: Specifies an existing hierarchical instance output pin name.

Example

The following commands remove the clock domain that was first created for internal clock u1/out:

```
set_dft_internal_clock_domain u1/out
reset_dft_internal_clock_domain u1/out
```

Related Information

check_dft_rules

remove_dft_assertions -internal_clock_domains

report_dft_assertions

set_dft_internal_clock_domain

reset_dft_transparent

```
reset_dft_transparent
     [-module { module_id | module_name }]
     -instance { instance_id | instance_name }
     { output_port_id | output_port_name} [-all]
```

Removes the virtual connectivity information specified for a black box module previously defined with the `set_dft_transparent` assertion.

You need to rerun `check_dft_rules` before you connect the scan chains in order to propagate the changes through the circuit.

Options and Arguments

`-all`
: Specifies that you want to remove the connectivity data for all modules and instances that were previous set using the `set_dft_transparent` assertion.

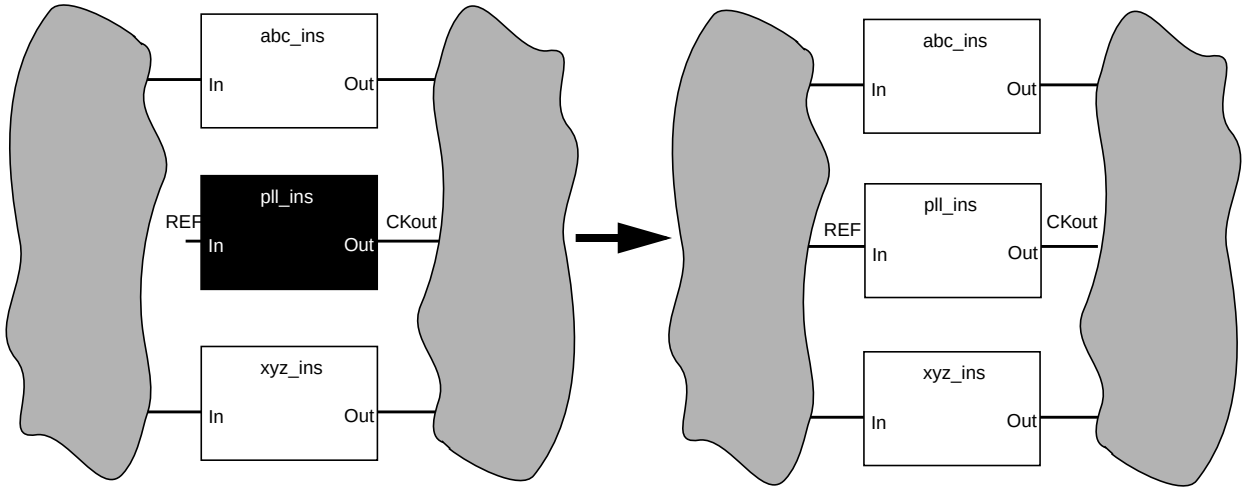
`-instance {`*`instance_id | instance_name`*`}` *`output_port_id | output_port_name`*
: Specifies the instance and its output port for which you want to remove the connectivity data.

`-module {`*`module_id | module_name`*`}`
: Specifies the module for the instance that you want to reset when you specify an instance name. You do not need to specify the module for an instance ID.
Default: Current module

Example

The following commands remove the virtual connectivity information that was specified for black box instance `pll_ins` and output port `Out`:

```
reset_dft_transparent -instance pll_ins Out
check_dft_rules
```



set_dft_transparent instance pll_ins -from In Out

reset_dft_transparent instance pll_ins Out

Black color (black box) indicates a `set_dft_transparent` assertion has been specified across the I/O pins of the `pll_ins` instance.

Related Information

`check_dft_rules`

`remove_dft_assertions -dft_transparent`

`report_dft_assertions`

`set_dft_transparent`

reset_dont_scan

```
reset_dont_scan [-module {module_id | module_name}]
     [-instance {list_of_inst_id | list_of_inst_name}]
```

Reverses the effect of the `set_dont_scan` assertion.

You need to rerun `check_dft_rules` before you connect the scan chain in order to propagate the changes through the circuit.

Options and Arguments

`-instance {list_of_inst_id | list_of_inst_name}`
Specifies one or more instances that you want to reset. All registers in the instance hierarchy that pass DFT rule checking are included in the scan chain.

`-module {module_id | module_name}`
Specifies the module that you want to reset. All registers in the module hierarchy that pass DFT rule checking are included in the scan chain.

Examples

- The following commands remove the `set_dont_scan` assertion from instance `clk_div_reg`, report the current settings of the DFT assertions and configuration constraints, then propagate the changes:

```
reset_dont_scan -instance clk_div_reg
report_dft_assertions
check_dft_rules
```

- The following commands remove the `set_dont_scan` assertion from module `lower1`, then propagate the changes:

```
reset_dont_scan -module lower1
check_dft_rules
```

Related Information

`check_dft_rules`

`remove_dft_assertions -dont_scan`

`report_dft_assertions`

`set_dont_scan`

reset_dont_touch_scan

```
reset_dont_touch_scan [module_id | module_name]
```

Reverses the effect of the `set_dont_touch_scan` assertion. This assertion must be set prior to running the scan connection engine.

After this assertion is issued, the connection engine may modify and reconfigure existing scan chains inside the module in subsequent scan connection runs (`do_xform_connect_scan`).

Options and Arguments

`module_id`
: Specifies the identifier (ID) of the module that you want to reset.

`module_name`
: Specifies the name of the module that you want to reset.

Example

The following commands remove the `dont_touch_scan` assertion from module `submod1` before running the connection engine:

```
reset_dont_touch_scan submod1
Info: module 'submod1' cleared of DONT_TOUCH_SCAN mode.
report_dft_assertions > ./DFTAssert
do_xform_connect_scan
```

Related Information

`do_xform_connect_scan`

`remove_dft_assertions -dont_touch_scan`

`report_dft_assertions`

`set_dont_touch_scan`

reset_must_scan

```
reset_must_scan
     [-instance {list_of_register_instance_id |
     list_of_register_instance_name}]
```

Reverses the effect of the `set_must_scan` assertion.

You need to rerun `check_dft_rules` before you connect the scan chain in order to propagate the changes through the circuit.

Options and Arguments

`-instance {`*`list_of_register_instance_id`* `|` *`list_of_instance_name`*`}`
: Specifies one or more instances that you want to reset. All registers in the instance hierarchy that pass DFT rule checking are included in the scan chain.

Example

The following commands remove the `set_must_scan` assertion from instances `u1/regA` and `u2/regB`, then propagate the changes through the circuit:

```
reset_must_scan -instance u1/regA u2/regB
check_dft_rules
```

Related Information

`check_dft_rules`

`remove_dft_assertions -must_scan`

`report_dft_assertions`

`set_must_scan`

reset_scan_data

```
reset_scan_data
     [-clock { clock_port_id | clock_port_name }]
```

Removes the scan data port assertions from the top-level module. The tool assigns default names or propagates the names applied at the current module (if specified) using the `set_scan_data` command.

Note: You need to specify the `reset_scan_data` command before you connect the scan chains.

Options and Arguments

`-clock {`*`clock_port_id`* `|` *`clock_port_name`*`}`
: Specifies the clock port ID or name of a top-level port, or an internal clock domain point set using the `set_scan_data` command.

Example

The following example resets all scan data port assertions:

```
reset_scan_data
```

Related Information

`do_xform_connect_scan`

`remove_dft_assertions -scan_data_io`

`report_dft_assertions`

`set_scan_data`

reset_test_mode_setup

```
reset_test_mode_setup
     {input_port_id | input_port_name }
```

Reverses the effect of a previous `set_test_mode_setup` assertion.

You need to rerun `check_dft_rules` before you connect the scan chain in order to propagate the changes through the circuit.

Options and Arguments

`input_port_id`
: Specifies an input port identifier (id) for the test mode signal.

`input_port_name`
: Specifies an input port name for the test mode signal.

Examples

The following commands reset the value for the test mode signal associated with input port `xyz`, then propagate the changes through the circuit:

```
set_top_timing_module top
set_test_mode_setup xyz 1
reset_test_mode_setup xyz
Info: Module 'top' cleared of test mode 'xyz = 1' during entire test session.
set_test_mode_setup xyz 0
check_dft_rules
```

Related Information

`remove_dft_assertions -test_mode_setup`

`report_dft_assertions`

`set_test_mode_setup`

set_dft_clock_waveform

```
set_dft_clock_waveform [RTZ|RT1]
```

Controls how the DFT clock waveforms are applied to all top-level DFT clock signals. The DFT rules checker and the scan configuration and connection engines will use this information to mix compatible clock edges in the proper order to minimize the number of lockup elements.

You must use this command prior to running `check_dft_rules` for it to be effective.
Default: `RTZ`

Options and Arguments

`RT1`

Specifies that for all clocks, the negative (falling) edge occurs first when applying the clock signals to the design. As a result the tool will connect the positive edge-triggered flip-flops before connecting the negative edge-triggered flip-flops to avoid insertion of a lockup element in the scan chains.

`RTZ`

Specifies that for all clocks, the positive (rising) edge occurs first when applying the clock signals to the design. As a result the tool will connect the negative edge-triggered flip-flops before connecting the positive edge-triggered flip-flops to avoid insertion of a lockup element in the scan chains.

Example

- The following example shows you where to use the `set_dft_clock_waveform` command in a flow that creates a scan chain configuration.

```
...
#specify the test setup
set_global dft_scan_avoid_control_buffering
set_scan_mode
set_scan_data
set_dft_clock_waveform
# run the DFT rule checks and report the registers
check_dft_rules
report_dft_registers > DFTregs
# optimize the design (map to scan flops and create the chains)
# uncomment compatible clock domains assertion if domain merging is desired.
# set_dft_compatible_clock_domains
do_optimize
...
```

- For examples of where to use the `set_dft_clock_waveform` command in flows that analyze pre-existing scan chains, refer to Reordering Scan Chains on an Imported Netlist with Preconfigured Scan Chain Architecture in the *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)* manual.

Related Information

`check_dft_rules`

`do_xform_connect_scan`

`set_dft_compatible_clock_domains`

set_dft_compatible_clock_domains

```
set_dft_compatible_clock_domains
     [-all | -all_rise | -all_fall | list_of_clocks
     |-same_edge | -same_clock | -same_root]
```

Specifies the compatible clock domains whose affected scan flip-flops can be merged into a single scan chain using lockup latches in between.
Default: No clock domains (including different phases of the same clock) are assumed compatible.

You need to specify this command before you create a scan chain configuration using the `do_xform_connect_scan` command.

Options and Arguments

Any command-line option, or a *`list_of_clocks`*, should be specified as:

`-all`

Specifies that all clocks are compatible. This setting supersedes all prior invocations of this command.

`-all_rise`

Specifies that all rising-edge clock domains are compatible.

`-all_fall`

Specifies that all falling-edge clock domains are compatible.

`list_of_clocks`

Specifies the clock domain. A clock domain can be specified as follows: *`clock`* `[-rise | -fall]`

`clock` should be an existing top level clock port name, hierarchical clock pin, or clock identifier.

`-rise` denotes rising edge.

`-fall` denotes falling edge.

When no edge is specified, both domains are assumed to be compatible, if present.

`-same_clock`

Specifies that both edges of each clock are compatible. For example, if both edges of clocks A and B are present (four domains total), this is equivalent to the following commands:

```
set_dft_compatible_clock_domains A
set_dft_compatible_clock_domains B
```

`-same_edge`

Specifies that all clock domains of the same edge are compatible.

`-same_root`

Causes all clocks/phases with the same root domain to be grouped together.

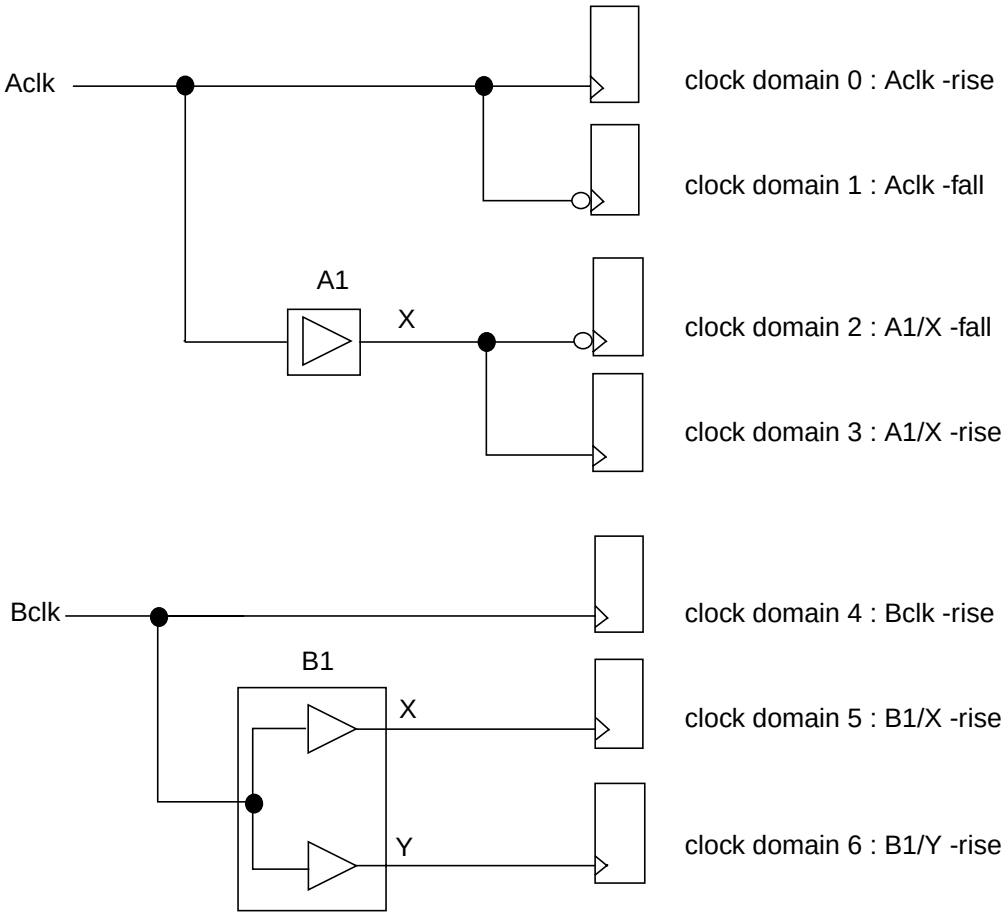
The settings of `set_dft_compatible_clock_domains` are cumulative. If clock domains `A` and `B` are specified as compatible, and subsequently `B` and `C` are set compatible, then clocks `A`, `B`, and `C` are compatible.

Use the `reset_dft_compatible_clock_domains` command to remove all prior compatibility settings.

Examples

For the following examples, consider the schematic shown in Figure 6-6. The design has 7 clock domains.

Figure 6-4 Schematic for the set_dft_compatible_clock_domains Examples



Note: Use the `set_dft_internal_clock_domain` command to identify domains 2, 3, 5, and 6.

- The following command declares that clock domains 2 and 5 can be merged:

  ```
  set_dft_compatible_clock_domains A1/X -fall B1/X -rise.
  ```

- The following command declares that clock domains 2 and 3 (both edges of A1/X) can be merged:

  ```
  set_dft_compatible_clock_domains A1/X
  ```

- The following command declares that clock domains 0 and 1 can be merged. No edge is specified; include both.

  ```
  set_dft_compatible_clock_domains Aclk
  ```

- The following command takes no action (same as default since only one edge specified for `Bclk`). To group all clocks with Bclk as the root (only), you would need to specify them explicitly, such as `Bclk B1/X B1/Y`.

```
set_dft_compatible_clock_domains Bclk
```

- The following command groups clock domains (`0`, `1`) and (`2`, `3`). Only alternate phases—when present for specific clocks, `Aclk` and `A1/X` in this case—are grouped together.

```
set_dft_compatible_clock_domains -sameclock
```

- The following command groups clock domains (`0`, `1`, `2`, `3`) and (`4`, `5`, `6`):

```
set_dft_compatible_clock_domains -sameroot
```

- The following example uses the cumulative effect of successive commands. As a result, the following groups are created: (`0`, `1`) and (`2`, `3`, `4`).

```
set_dft_compatible_clock_domains -sameclock
set_dft_compatible_clock_domains A1/X -rise Bclk
```

- The following command declares that clock domains `0`, `3, 4`, `5`, and `6` are compatible:

```
set_dft_compatible_clock_domains -allrise
```

- The following command declares that clock domains `1` and `2` are compatible:

```
set_dft_compatible_clock_domains -allfall
```

- The following command creates two groups of compatible clock domains: (`0`, `3, 4`, `5`, `6`) and (`1`,`2`):

```
set_dft_compatible_clock_domains -sameedge
```

Related Information

`do_xform_connect_scan`

`remove_dft_assertions -compatible_clock_domains`

`report_dft_assertions`

`reset_dft_compatible_clock_domains`

`set_dft_internal_clock_domain`

set_dft_compatible_chains

```
set_dft_compatible_chains [-partition name]
     {-all | chain_name1 chain_name2...}
```

Designates a set of scan chains in the scanDEF file as compatible swap candidates in placement-based reordering. When chains within a group are reordered or repartitioned, `FLOATING` and `ORDERED` segments can be swapped between them, but the original maximum sequential length of the chains in the group is retained.

Note: The number of chain elements may exceed the sequential length for any given chain because not all elements are required to be sequential.

Each use of this command defines a group of chains whose elements are swappable, allowing multiple groups of two or more chains to be defined. Chains may only be members of a single group, so references to chains previously-defined in other groups are ignored.

Chain groups defined with this command as swappable remain that way until removing them with `remove_dft_assertions -compatible_chains` or until the DEF scan data itself is removed using `do_remove_scan_order_data`.

Note: You must have a Cadence PKS license to use this functionality.

Options and Arguments

`-all`
: Specifies that all chains comprise a single group.

`chain_name1 chain_name2...`
: Specifies the names of the chains—as defined in the scanDEF file—that you want to designate as compatible swap candidates. You must specify at least two chain names.

`-partition` *`name`*
: Names the specific chain group. If you do not specify this option, a group name is given by default, such as `group_`*`n`*.

Examples

- The following commands designate scan chains `CLK_FB_F1`, `CLK_FB_F2`, and `CLK_FB_F3` (defined in the `scan.def` file) as compatible swap candidates, then assign the chains to group `group_1`:

```
read_def -scan_only scan.def
set_dft_compatible_chains CLK_FB_F1 CLK_FB_F2 CLK_FB_F3
```

- The following commands designate scan chains `CLK_FB_F1`, `CLK_FB_F2`, and `CLK_FB_F3` (defined in the `scan.def` file) as compatible swap candidates, then assign the chains to group `foo2`:

```
read_def -scan_only scan.def
set_dft_compatible_chains -partition foo2 CLK_FB_F1 CLK_FB_F2 CLK_FB_F3
```

Related Information

`do_remove_scan_order_data`

`do_xform_connect_scan`

`read_def -scan_only`

`remove_dft_assertions`

set_dft_fix_violations

```
set_dft_fix_violations
     [-async_set] [-async_reset] [-async_set_reset]
     -test_mode signal [-control signal]
     { -all | -instance instance_list}
     [{-clock|-observe_scan} -test_clock clock [-rise|-fall]]
```

Specifies which DFT rule violations must be fixed and how to fix them. DFT rule violations are detected by the `check_dft_rules` command. You must run the `do_xform_fix_dft_violations` command to actually fix the violations.

Note: Currently only DFT violations related to asynchronous set and reset can be fixed. Clock violations cannot be fixed yet.

To fix asynchronous set or reset violations, you need to specify a test-mode or scan-mode signal. Using a scan-mode signal makes it only controllable during the scan-shift operation. These signals can be either a previously specified test-mode or scan-mode signal or any existing port or hierarchical pin. If you use a hierarchical pin, a user-defined instance pin or flip-flop pin is preferred, because some design optimizations can change combinational logic.

Warning messages are issued whenever a strange assignment occurs. For example, when specifying instance names that do not have the particular violation to be fixed.

Options and Arguments

`-all`

Applies the assertion to all flip-flops that have the type of violation specified.

`-async_reset`

Fixes the asynchronous reset violations on either all instances, or just the instances specified with the `-instance` option.

`-async_set`

Fixes the asynchronous set violations on either all instances, or just the instances specified with the `-instance` option.

`-async_set_reset`

Fixes the asynchronous set and reset violations on either all instances, or just the instances specified with the `-instance` option.

`-clock`

Fixes the clock violations on either all instances, or just the instances specified with the `-instance` option.

Note: This option requires you to specify the `-test_clock` option and cannot be used together with the `-observe_scan` option.

`-control` *`signal`*

Specifies to fix violations by inserting control-variable test points rather than control-0 or control-1 test points. You must specify the *`signal`* value to be used during test mode using the `set_test_mode_setup` command.

`-instance` *`instance_list`*

Specifies the instances to which you want to apply the assertion. You can identify the instances using either their hierarchical instance names or their object identifiers (IDs).

`-observe_scan`

Inserts a scan flip-flop to observe the specified location. The scan flip-flop is connected to a scan chain later on.

Note: This option requires you to specify the `-test_clock` option and cannot be used together with the `-clock` option.

`-rise | -fall`

Specifies the edge of the `-test_clock` that is active during test mode operation. These options are only valid in conjunction with `-test_clock`.
Default: `-rise`

`-test_clock` *`clock`*

Specifies the test-mode clock signal used for fixing clock rule violations and also the signal that drives the clock pin of the inserted scan flip-flops during test mode operation. A warning is issued if the clock signal is not a primary input.

`-test_mode` *`signal`*

Specifies the particular signal to fix the violation. Specify either a previously declared test-mode or scan-mode signal, a top-level port, or a hierarchical pin. If you do not specify a *`signal`*, a warning message is issued.

Examples

- The following commands fix all asynchronous set and reset violations by using test mode signal TM active high:

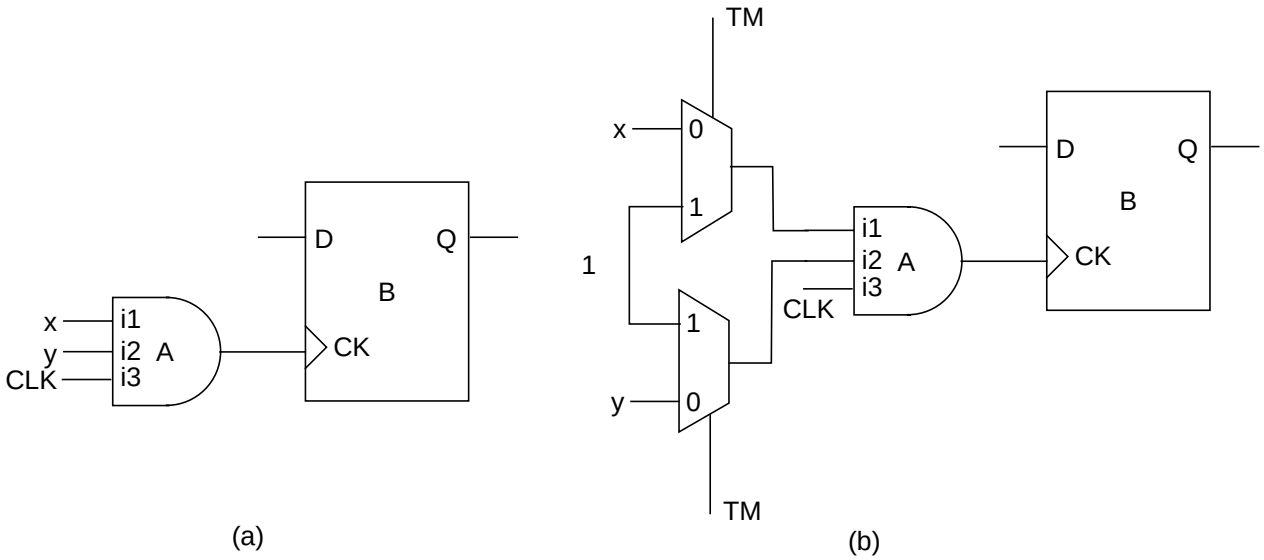
```
set_test_mode_setup TM 1
set_dft_fix_violation -async_set -async_reset -test_mode TM -all
```

- The following commands fix the clock rule violation caused by the gating of the clock pin of flip-flop B (see in Figure 6-5 on page 751 (a)).

```
set_test_mode_setup TM 1
set_dft_fix_violations -instance B -clock -test_mode TM -test_clock CLK
do_xform_fix_violations
```

  The result is shown in Figure 6-5 on page 751 (b).

Figure 6-5 Fixing a Clock Violation



Related Information

do xform fix dft violations

reset dft fix violations

Avoiding DFT Rule Violations in the *Design for Test (DFT) Using BuildGates Synthesis and Cadence PKS* manual.

Tasks for Automatic Fixing of the Design Rule Violations in the *Design for Test (DFT) Using BuildGates Synthesis and Cadence PKS* manual.

set_dft_internal_clock_domain

```
set_dft_internal_clock_domain
     { hier_pin_id | hier_pin_name }
```

Allows you to treat all the flip-flops driven by the specified pin as a separate domain for scan insertion purposes if there are different internal clock domains logically connected to the same clock input port in test mode.

This means that such flip-flops will be kept together on the same scan chain(s), and will be separated if necessary from other compatible domains on the same chain by lockup latches (see the `-sameroot` option for set_dft_compatible_clock_domains on page 743).

Such flip-flops must pass all DFT rules, including clock rules such that those flip-flops' clock ports can be driven by a primary input under test mode conditions.

Important

This command must be specified prior to issuing the `check_dft_rules` command.

Clocks are specified cumulatively by the `set_dft_internal_clock_domain` command. To remove any of the previous settings, use the `reset_dft_internal_clock_domain` command. This data applies to the current module.

Options and Arguments

`hier_pin_id`
: Specifies an existing instance output pin identifier (ID).

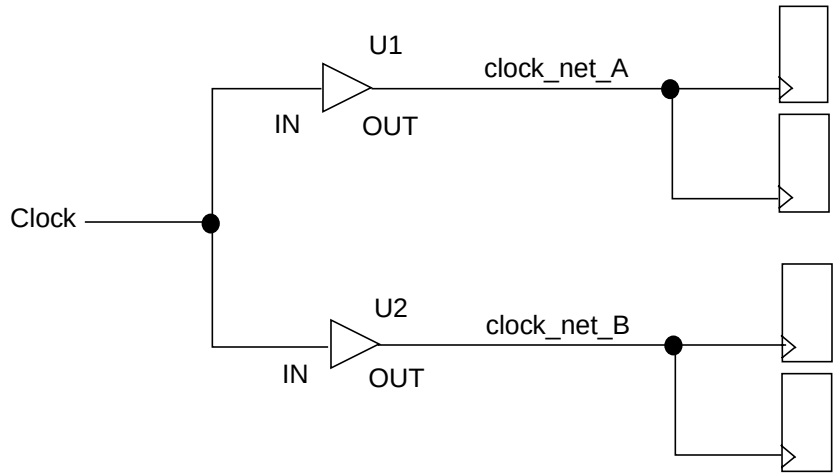
`hier_pin_name`
: Specifies an existing hierarchical instance output pin name.

Example

The following commands add a new clock domain for the internal clock nodes `U1/OUT` and `U2/OUT`. Consider the schematic shown in Figure 6-6 on page 753.

```
set_dft_internal_clock_domain U1/OUT
set_dft_internal_clock_domain U2/OUT
check_dft_rules
Info: Partitioning registers for scan based on clock domain. <DFT-325>
     Clock Domain 0 from pin 'clock' (Pos Edge)<Internal Pin: 'U1/OUT'> has 2 f/f
     Clock Domain 1 from pin 'clock' (Pos Edge)<Internal Pin: 'U2/OUT'> has 2 f/f
```

Figure 6-6 Schematic for set_dft_internal_clock_domain Example



Related Information

`check_dft_rules`

`remove_dft_assertions -internal_clock_domains`

`report_dft_assertions`

`reset_dft_internal_clock_domain`

`set_dft_compatible_clock_domains`

set_dft_lockup_element

```
set_dft_lockup_element instance
```

Identifies the specified instance as a lockup-element in an existing chain. This is required for lockup flip-flop elements, and is not necessary for lockup latches.

Currently, if you read in a design that has lockup flip-flops in the chain that were inserted by a third-party tool or even by PKS but in a different terminal session, DFT cannot recognize the element automatically, because it looks no different from a non-scan flip-flop feeding the scan-in pin of a scan flip-flop. Therefore, you must identify the lockup flip-flop so it can be correctly traced.

DFT can recognize lockup latches in chains, whether inserted by PKS or a third-party tool. DFT can also recognize lockup flip-flops if they are inserted by PKS and you stay in the same BG shell.

Options and Arguments

instance

Specifies an instance as a lockup-element in an existing chain.

Example

The following command identifies instance i_57 as a lockup flip-flop:

```
set_dft_lockup_element top/u2/i_57
```

Related Information

remove_dft_assertions -lockup_element

set_dft_transparent

```
set_dft_transparent
     -module {module_id | module_name}
     [-instance {instance_id | instance_name}]
     -from {input_port_id | input_port_name} [-invert]
     {output_port_id | output_port_name}
```

Specifies the logical connectivity within a black box module, from an input port to an output port. Otherwise, when your design has black box modules, the `check_dft_rules` command cannot detect whether a direct path exists from a primary input to the flip-flop's clock pin, set pins, or reset pins of the module, and it reports a DFT violation.

Important

This command should be used only for black box modules, or cellrefs for which a function definition has not been defined in the technology library. This command should not be used to propagate scan chain analysis data.

Options and Arguments

`-from {`*`input_port_id | input_port_name`*`}`
: Specifies an input port that is on a direct path through the module or instance. Specify the input port using its name or identifier (ID).

`-instance {`*`instance_id | instance_name`*`}`
: Specifies the black box instance for which you want to specify the connectivity.

`-invert`
: Specifies that the path is inverted.
Default: The `set_dft_transparent` command assumes that the path represents a direct connection.

`-module {`*`module_id | module_name`*`}`
: Specifies the black box module for the instance that you want to set, when you specify an instance name. You do not need to specify a module for an instance ID.
Default: Current module

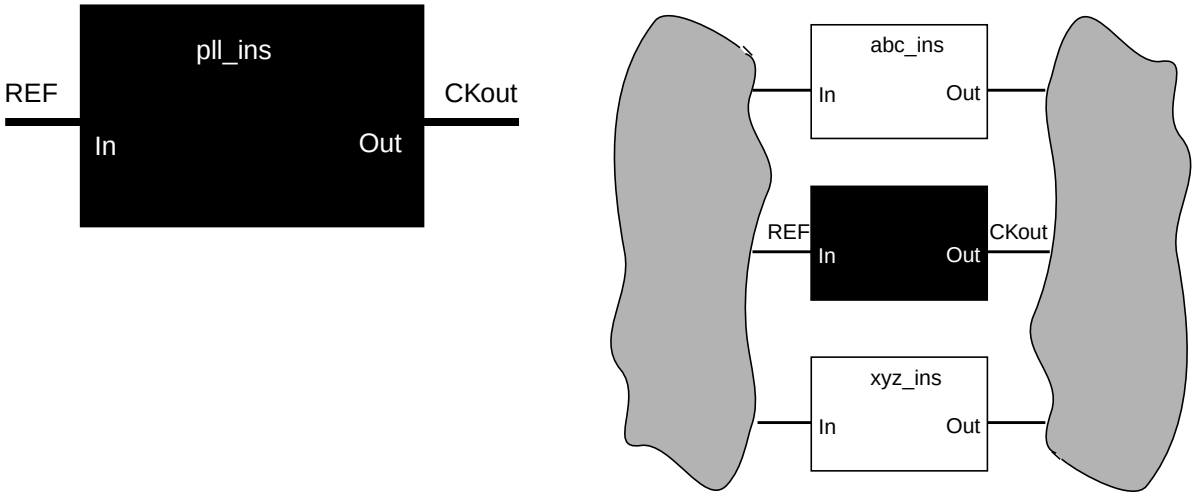
`{`*`output_port_id | output_port_name`*`}`
: Specifies an output port that is on a direct path through the

module or instance. Specify the output port using its name or identifier (ID).

Example

The following commands specify the functional connectivity between input port `In` and output port `Out` of black box instance `pll_ins`, then propagate the changes:

```
set_dft_transparent -instance [find -hier -instance pll_ins] -from In Out
check_dft_rules
```



Black color (black box) indicates a `set_dft_transparent` assertion has been specified across the I/O pins of the `pll_ins` instance. The black box module has no function.

Related Information

`check_dft_rules`

`remove_dft_assertions -dft_transparent`

`report_dft_assertions`

`reset_dft_transparent`

set_dont_scan

```
set_dont_scan [-module {module_id | module_name}]
     [[-instance] {list_of_inst_id | list_of_inst_name}]
```

Excludes from the scan chain all of the registers in the specified module or instance hierarchy. Use this command before or after uniquification of your design.
Default: The test synthesis tool tries to include all registers that pass DFT rule checking in the scan chain.

This command determines whether a register is included in the scan chain, so it affects the wiring of the scan chain. This command can affect the area of the design because any registers excluded from the scan chain could be mapped to smaller, non-scannable cells.

Run `check_dft_rules` before you connect the scan chain in order to propagate the changes through the circuit.

Options and Arguments

`-instance {`*`list_of_inst_id | list_of_inst_names`*`}`
Excludes all registers in the specified instance hierarchy from the scan chain.

`-module {`*`module_id | module_name`*`}`
Excludes all registers in the specified module hierarchy from the scan chain.

Example

The following commands exclude registers 75939, 75027, and 77859 from the scan chain, then propagate the changes through the circuit:

```
set_dont_scan 75939 75027 77859
check_dft_rules
```

Related Information

`check_dft_rules`

`reset_dont_scan`

`remove_dft_assertions -dont_scan`

`report_dft_assertions`

set_dont_touch_scan

```
set_dont_touch_scan [module_id | module_name]
```

Prevents modification or reconfiguring of existing scan chains inside the specified module (a scan-specific equivalent to `set_dont_modify` on the module). All existing scan chains in the indicated module are analyzed for suitability and compatibility with scan chains being constructed from the top level and are connected accordingly. Any existing module chain containing a flip-flop that violates DFT rules will be excluded from connection to outer (top level) chains.

This command must be specified prior to running the scan connection engine.

Options and Arguments

module_id

Specifies the ID of the module that you want to leave unchanged.

module_name

Specifies the name of the module that you want to leave unchanged.

Example

The following commands prevent modification of the scan chains in module `submod1`:

```
set_dont_touch_scan submod1
Info: Module 'submod1' set to DONT_TOUCH_SCAN mode.
do_xform_connect_scan
```

Related Information

`do_xform_connect_scan`

`remove_dft_assertions -dont_touch_scan`

`report_dft_assertions`

`reset_dont_touch_scan`

set_lssd_aux_clock

```
set_lssd_aux_clock { clk_pin_name | clk_pin_id }
```

Sets the auxiliary clock pin for aux_clocked_lssd scan style. In the aux_clocked_lssd scan style, a regular edge-triggered D flip-flop is replaced by an aux_clocked_lssd scan cell that has one edge triggered System Clock input and level-sensitive scan clocks, scan_clock_a, scan_clock_b and aux_clock.

The *clock* active edge is assumed to be rising-edge, where applicable. This command overrides any previous command setting specified, and applies to the current module.

Note: You must specify this command along with aux_clock_lssd in the set_scan_style command before running check_dft_rules.

Options and Arguments

clk_pin_id

Specifies the identifier (ID) of a top-level input port or of an output pin on an instance.

clk_pin_name

Specifies the name of a top-level input port or the hierarchical name of an output pin on an instance. If the specified top-level pin does not exist, an input pin with that name is created.

Example

The following command specifies aux_test_clock as the auxiliary clock pin for aux_clocked_lssd scan style:

```
set_lssd_aux_clock aux_test_clock
```

Related Information

check_dft_rules

report_dft_assertions

set_lssd_scan_clock_a

set_lssd_scan_clock_b

set_scan_style aux_clock_lssd

set_lssd_scan_clock_a

```
set_lssd_scan_clock_a
     { clk_pin_name | clk_pin_id }
```

Sets the LSSD scan_clock_a pin for clocked_lssd or aux_clocked_lssd scan style. In the clocked LSSD scan style, a regular edge-triggered D flip-flop is replaced by a clocked_lssd scan cell that has one edge triggered System Clock input and level-sensitive scan clocks, scan_clock_a and scan_clock_b.

The *clock* active edge is assumed to be rising-edge, where applicable. This command overrides any previous command setting specified, and applies to the current module.

Note: You must specify this command along with the appropriate scan style using the set_scan_style command before running check_dft_rules.

Options and Arguments

clk_pin_id

Specifies the identifier (ID) of a top-level input port or of an output pin on an instance.

clk_pin_name

Specifies the name of a top-level input port or the hierarchical name of an output pin on an instance. If the specified top-level pin does not exist, an input pin with that name is created.

Example

The following command specifies clockA as the LSSD scan_clock_a pin for clocked_lssd or aux_clocked_lssd scan style:

```
set_lssd_scan_clock_a clockA
```

Related Information

check_dft_rules

report_dft_assertions

set_lssd_scan_clock_b

set_scan_style {clocked_lssd | aux_clock_lssd}

set_lssd_scan_clock_b

```
set_lssd_scan_clock_b
     { clk_pin_name | clk_pin_id }
```

Sets the LSSD scan_clock_b pin for clocked_lssd or aux_clocked_lssd scan style. In the clocked_lssd scan style, a regular edge-triggered D flip-flop is replaced by a clocked LSSD scan cell that has one edge triggered System Clock input and level-sensitive scan clocks, scan_clock_a and scan_clock_b.

The *clock* active edge is assumed to be rising-edge, where applicable. This command overrides any previous command setting specified, and applies to the current module.

Note: You must specify this command along with the appropriate scan style using the set_scan_style command before running check_dft_rules.

Options and Arguments

clk_pin_id

Specifies the identifier (ID) of a top-level input port or of an output pin on an instance.

clk_pin_name

Specifies the name of a top-level input port or the hierarchical name of an output pin on an instance. If the specified top-level pin does not exist, an input pin with that name is created.

Example

The following command specifies clockB as the LSSD scan_clock_b pin for clocked_lssd or aux_clocked_lssd scan style:

```
set_lssd_scan_clock_b clockB
```

Related Information

check_dft_rules

report_dft_assertions

set_lssd_scan_clock_a

set_scan_style {clocked_lssd | aux_clock_lssd}

set_max_scan_chain_length

```
set_max_scan_chain_length
     [-clock clock_name [-rise | -fall]]
     max_chain_length [-priority]
```

Specifies the maximum length of any scan chain in the current module. If necessary, the test synthesis tool creates additional scan chains to keep each scan chain at or below the required maximum length.
Default: There is no limit to the maximum length of a scan chain.

The `-priority` option specifies that this (maximum chain length) constraint takes precedence over `set_number_of_scan_chains` constraint when both are specified.

The `-clock` option allows the maximum chain length constraint to be applied to a specific clock domain or domain group (see the `set_dft_compatible_clock_domains` command). Such domain-specific max-length constraints always take precedence over non-specific constraints (either set_max_scan_chain_length or set_number_of_scan_chains), and may be combined with them to provide more detailed control over the scan configuration created. Also, the `-priority` option can be used with domain-specific forms of each command to specify which limit takes precedence. Warnings will be issued when constraint inconsistencies are specified (or inadvertently created).

Note: Applying a `set_max_scan_chain_length` constraint overrides chain balancing that would occur under a `set_number_of_scan_chains` constraint. This means that when a max length limit is applied within each domain group, the chains are packed to capacity, with the last chain created in each group getting the residual amount.

When the maximum length for a scan chain is exceeded by the length of an existing chain inside a `set_dont_modify` or `set_dont_touch_scan` module, a warning is issued and the `dont_touch` chain will be placed in a top level chain by itself.

For the recommended use model for this command, see *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Options and Arguments

`-clock` *`clock_name`* `[-rise | -fall]`
: Specifies a clock domain, or clock domain group for which the maximum length chain constraint applies (only). Specifying any member of a compatible clock domain group constitutes setting the constraint for the entire group. For example, if clocks A, B, and C are specified as compatible, specifying the option `-clock` A means that the length maximum applies to all chains

formed with registers in domains A, B, and C. Maximum limits specified with this `-clock` option take precedence over similar commands with no `-clock` option.

`max_chain_length`

Specifies the maximum number of registers allowed in a scan chain.

`-priority`

Specifies that the maximum length constraint specified takes precedence over the `set_number_of_scan_chains` limit when both commands have been specified (including with or without `-clock` options). A warning is issued whenever the non-priority limit is exceeded in order to enforce the priority limit. *Default*: The `set_number_of_scan_chains` limit has priority.

Examples

For the following examples, consider a design with 20 registers in clock domain `clkA`, and 30 registers in clock domain `clkB`.

- The following command limits the maximum chain length to 20. As a result, all registers in domain `clkA` are included in one chain, because it satisfies the limit of 20. Registers in `clkB` are assigned to two chains, one with length 20 and the other with length10:

  ```
  set_max_scan_chain_length 20
  ```

- The following commands limit the maximum chain length to 20 and the number of chains to 4. Because the number of chains constraint has priority (by default), clock domain `clkA` gets two chains of 10 registers each, and clock domain `clkB` gets two chains of 15 each:

  ```
  set_max_scan_chain_length 20
  set_number_of_scan_chains 4
  ```

- The following commands are similar to those in the previous example (same scan configuration created). However, in this case warnings are issued for clock domain `clkB`, because its two chains exceed the soft maximum chain limit of 10:

  ```
  set_max_scan_chain_length 10
  set_number_of_scan_chains 4
  ```

- The following commands give priority to the maximum length constraint. As a result, clock domain `clkA` has two chains (10 each), while clock domain `clkB` has 3 chains. A warning is issued since 5 total chains exceeds the soft number-of-chains limit of 4.

  ```
  set_max_scan_chain_length 10 -priority
  set_number_of_scan_chains 4
  ```

- The following command sets the maximum length constraint specifically for clock domain `clkA`, resulting in two chains of length 7 and one of length 6, respectively. Clock domain `clkB` gets a single chain of length 30, since no constraint was specified for it.

  ```
  set_max_scan_chain_length -clock clkA 7
  ```

- The following commands specify domain-specific constraints for `clkA.` Because the specific number-of-chains limit for `clkA` has default priority, two balanced chains of length 10 are created. A warning will be issued because the soft limit of 7 is exceeded.

  ```
  set_max_scan_chain_length -clock clkA 7
  set_number_of_scan_chains -clock clkA 2
  ```

- The following commands specify that the maximum chain length for clock domain `clkB` is 8, while the overall number of scan chains is limited to 6. As a result, four chains of length 8, 8, 8, and 6, respectively are created in clock domain `clkB`, while clock domain `clkA` gets the two remaining chains (to satisfy the overall limit of six) of length 10 each:

  ```
  set_max_scan_chain_length -clock clkB 8
  set_number_of_scan_chains 6
  ```

- The following commands limit the chain length for clock domain `clkB` to 8, and the chain length for the other domains to 5. As a result, four chains of length 8, 8, 8, and 6, respectively are created in clock domain `clkB`, while four chains of length 5, are created for clock domain `clkA`:

  ```
  set_max_scan_chain_length -clock clkB 8
  set_max_scan_chain_length 5
  ```

Related Information

`remove_dft_assertions [-config_constraints]`

`remove_dft_assertions [-configuration_constraints]`

`set_dft_compatible_clock_domains`

`set_number_of_scan_chains`

set_must_scan

```
set_must_scan
     [-clock {clock_port_id | clock_port_name}]
     [-rise | -fall] {list_of_register_instance_id |
     list_of_register_instance_name}
```

Forces a register to be included in a scan chain, even if the register does not pass all DFT rules. If the register fails the clock controllability rule check, you must also specify the clock domain to which the register belongs.

You must run `check_dft_rules` before you connect the scan chain in order to propagate the effects of this command.

Caution

When you use this command, some flip-flops that do not pass DFT rules may get placed in scan chains. As a result, the scan shifting operation may fail in simulation and on the tester. Please use this command rarely, and only when you understand the implications of its use.

Options and Arguments

`-clock {`*`clock_port_id | clock_port_name`*`}`
: Specifies the clock domain to which the register belongs. The *`clock_port_id`* or *`clock_port_name`* must be the top timing module's clock port. You use this option only if the register fails the clock controllability rule.

`list_of_register_instance_id`
: Specifies a list of identifiers (IDs) of the register instances that you want to include in the scan chain.

`list_of_register_instance_name`
: Specifies the hierarchical names of register instances that you want to include in the scan chain.

`-rise | -fall`
: Specifies the clock edge of the clock domain, when you also specify the `-clock` option.
Default: `-rise`

Example

The following commands force registers `u1`, `u3`, and `my_submodule/u7` to be included in a scan chain for the specified clock domain, then propagate the change:

```
set_must_scan -clock ck -rise u1 u3 my_submodule/u7
check_dft_rules
```

Related Information

`check_dft_rules`

`remove_dft_assertions -must_scan`

`report_dft_assertions`

`reset_must_scan`

set_number_of_scan_chains

```
set_number_of_scan_chains
     [-clock clock_name [-rise | -fall]]
     chain_count [-priority]
```

Specifies the number of scan chains to be created for the current module.

Default: Only one chain is created for each clock domain or compatible domain group (see `set_dft_compatible_clock_domains` command).

When this constraint is used to create multiple chains, an attempt is made to distribute registers from each domain or domain-group evenly (balance) across chains within that group, and apportion the number of chains across all applicable groups to meet the constraint in a way that minimizes differences in chain length as much as possible. For example, if a domain `clk1` had 30 registers, and `clk2` had 40 registers, a number-of-chains limit of 4 would result in `clk1` domain getting two chains of length 15, and `clk2` domain would have two chains of length 20.

The `-clock` option allows a number-of-chains constraint to be applied to a specific domain or domain group. This domain-specific form of the command always take precedence over non-specific ones, such that they can be combined to provide more detailed control over the scan configuration. When combined, the tool will attempt to meet the domain-specific constraint first, and then meet the non-specific one (see examples). Appropriate warnings are generated when this is not possible.

The `-priority` option allows you to specify that the limit being set takes precedence over a currently active max-chain-length constraint (set using `set_max_scan_chain_length` command). This is normally not needed unless the prior `set_max_scan_chain_length` command had itself used a `-priority` option, as it is the default for `set_number_of_scan_chains`.

For the recommended use model for this command, see *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Options and Arguments

chain_count
: Specifies the number of scan chains desired.
Default: One scan chain for each clock domain or domain group.

-clock *clock_name* [-rise | -fall]
: Specifies a clock domain for which the current limit applies (only). If the clock domain named is a member of a compatible

domain group, the setting is applied to the whole group. For example, if clocks A, B, and C are specified as compatible, specifying the option `-clock A` means that the number of chains applies to all chains formed with registers in domains A, B, and C. Commands specified with this option take precedence over commands without the option.

`-priority`

Specifies that the number-of-chains being set takes precedence over any max-chain-length limit currently active. Since this is the default for the `set_number_of_scan_chains` command, it is not needed unless you need to switch back from a prior `set_max_scan_chain_length` priority setting. This option can also be used similarly with domain-specific command forms (`-clock` options) to control the configuration within a specific domain or domain group.

Examples

For the examples below, consider a design with 20 registers in clkA domain, and 30 registers in clkB.

- The following command limits the overall number of chains to `4`. As a result, the tool creates two chains of length 10 for domain `clkA`, and two chains of length 15 for domain `clkB`:

  ```
  set_number_of_scan_chains 4
  ```

- The following commands limit the maximum chain length to `20` and the number of chains to `4`. Because by default the number of chains constraint has priority, the tool creates two chains of length10 for clock domain `clkA`, and two chains of length15 for clock domain `clkB`. So the additional `set_max_scan_chain_length` constraint has no effect:

  ```
  set_number_of_scan_chains 4
  set_max_scan_chain_length 20
  ```

- The following commands are similar to those in the previous example (same scan configuration created). However, in this case tool issues warnings for clock domain `clkB`, because its two chains exceed the soft maximum chain limit of `10`:

  ```
  set_number_of_scan_chains 4
  set_max_scan_chain_length 10
  ```

- The following command limits the number of chains for clock domain clkB to 3. As a result, one chain of length 30 is created for clock domain `clkA` (default, since no other length constraint was specified). Clock domain `clkB` gets three chains of length10 each:

  ```
  set_number_of_scan_chains -clock clkB 3
  ```

- The following commands limit the maximum chain length for clock domain clkB to 7, and the maximum number of scan chains for clock domain clkB to 6. As a result, the tool creates five chains of length 7, 7, 7, 7, and 2, respectively for clock domain clkB. Because the length constraint has priority, a warning is issued, since this exceeds the number-of-chains for this domain. Clock domain clkA is not constrained, and gets one chain of length 20.

```
set_max_scan_chain_length -clock clkB 7 -priority
set_number_of_scan_chains -clock clkB 4
```

- The following commands limit the maximum chain length for clock domain clkB to 8, and the overall number of scan chains to 6. As a result, the tool creates four chains of length 8, 8, 8, and 6, respectively for clock domain clkB, and two chains of length10 each for clock domain clkA (to satisfy the overall limit of six):

```
set_max_scan_chain_length -clock clkB 8
set_number_of_scan_chains 6
```

- The following commands limit the number of chains for clock domain clkB to 4, and the overall number of scan chains to 6. As a result, the tool creates four chains of length 7, 7, 8, and 8, respectively for clock domain clkB, and two chains of length 10 for clock domain clkA to satisfy the overall limit of 6:

```
set_number_of_scan_chains -clock clkB 4
set_number_of_scan_chains 6
```

Related Information

remove_dft_assertions [-config_constraints]

remove_dft_assertions [-configuration_constraints]

set_dft_compatible_clock_domains

set_max_scan_chain_length

set_scan_chain

```
set_scan_chain
     { -head head_segment_name | -tail tail_segment_name
     | -head head_segment_name -tail tail_segment_name }
```

Constrains segments previously defined with the `set_scan_chain_segment` command to be configured to specific chain positions.

Specific scan segments can be required to appear at the beginning (first shift-in position or *head*), or end (first shift-out position or *tail*) of a scan-chain. Also, both options can be used in the same command to specify when a head and tail must be placed in the same chain. Head and tail segments will always be inserted in chains with compatible domains (see `set_dft_compatible_clock_domains`). Head and tail segments applied to the same chain must themselves have compatible domains also.

Notes

1. You cannot use this command with the existing Scan Order File (SOF) (`read_scan_order_file`) or scanDEF file (`read_def -scan_only` or `write_def`) modes of configuration.
2. PKS scan-chain reordering treats defined segments as freely movable within chains (as a contiguous unit), unless you further constrain them with the `set_scan_chain` command. In this case, PKS ignores them in reordering.

Options and Arguments

`-head` *`head_segment_name`*
: Specifies a segment (an ordered set of scan registers) that must be placed at the head of a scan chain. The segment must have been previously defined with `set_scan_chain_segment`.

`-tail` *`tail_segment_name`*
: Specifies a segment (an ordered set of scan registers) that must be placed at the tail of a scan chain. The segment must have been previously defined with `set_scan_chain_segment`.

Examples

- The following commands define a segment S1 guaranteed to be kept at the head of a chain:

```
set_scan_chain_segment -name S1 \
      {C1/R_reg_0 C1/R_reg_1 C2/CC1/R_reg_3 C2/CC1/R_reg_0 C1/R_reg_2}
set_scan_chain -head S1
```

- The following commands define a segment S1 guaranteed to be kept at the tail of a chain:

```
set_scan_chain_segment -name S1 \
      {C1/R_reg_0 C1/R_reg_1 C2/CC1/R_reg_3 C2/CC1/R_reg_0 C1/R_reg_2}
set_scan_chain -tail S1
```

- The following commands define a pair of segments (S1 and S2) guaranteed to be kept at the head and tail of one chain:

```
set_scan_chain_segment -name S1 \
      {C1/R_reg_0 C1/R_reg_1 C2/CC1/R_reg_3 C2/CC1/R_reg_0 C1/R_reg_2
set_scan_chain_segment -name S2 \
      {C3/CCC2/CC0/R_reg_0 C3/CCC2/CC0/R_reg_1}
set_scan_chain -head S1 -tail S2
```

Related Information

remove_dft_assertions -scan_chain

report_dft_assertions

set_scan_chain_segment

Using Head and Tail Chain Constraints during Scan Chain Configuration and Scan Chain Reordering with PKS in the *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)* manual.

set_scan_chain_segment

```
set_scan_chain_segment
     {-from hier_pin_name -to hier_pin_name
     | [-name segment] instance_list }
```

This assertion has the following purposes:

- The `-from -to` specification identifies lower-level scan chain segments embedded in the hierarchy of the current module. This identification is used to correctly connect the embedded scan chains segments in a block that might have been marked as `dont_modify` or `dont_touch_scan` for example in the bottom-up flow.

  This identification is essential, if the tool can not distinguish the true scan-out pin of an embedded scan chain segment from a functional connection, or if the chain segment contains a terminal lockup element.

  Scan analysis uses this information as a hint to identify and trace the scan chain segments in the embedded module from its `sdi` to `sdo` pins, for connection of the segments into the top level chains.

- The second specification (`-name` *`segment instance_list`*) defines a collection of scan registers to be treated as a chain segment for scan configuration purposes.

  You can define multiple segments using multiple invocations of this command; one segment is defined per command invocation.

  You can later reference such segments in a `set_scan_chain` command to further constrain them and place them at the head or tail of a configured chain:

  ```
  set_scan_chain {-head segment_name|-tail segment_name
        |-head segment_name -tail segment_name}
  ```

  When the scan configuration is created from scratch (no pre-existing chains), DFT places the defined segments in scan-chains with consistent or compatible domains.

  When pre-existing chains are present, and the *preserve configuration* mode is used, DFT places the elements of each segment on the chain where the first element appeared originally. For more information, refer to the Running the Scan Connection Engine in preserve_config Mode section in the *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)* manual.

  Notes:

  - A scan element cannot be included in more than one segment.
  - You cannot use this command with the existing Scan Order File (SOF) or scanDEF file modes of configuration.

- PKS scan-chain reordering treats segments defined with this assertion as freely movable within chains (as a contiguous unit), unless you further constrained them with the `set_scan_chain` command. In this case, PKS ignores them in reordering.

Options and Arguments

`-from` *`hier_pin_name`*

Specifies the start pin for the scan chain segment. Specify the hierarchical name of the pin.

`instance_list`

Specifies the instances in the order they should appear in the scan chain (in shift order, that is, left-most corresponds to first bit shifted-in).

The instances must be scannable registers and must belong to the same clock domain, otherwise the segment is ignored. Do not specify scan registers that are contained in `dont_touch_scan` or `dont_modify` modules or they will be ignored. Specify either the hierarchical instance names or the instance identifiers.

`-name` *`segment`*

Defines a name for the segment that you can use to reference for the `set_scan_chain` command (head/tail specifications), and related diagnostics.

`-to` *`hier_pin_name`*

Specifies the end pin for the scan chain segment. Specify the hierarchical name of the pin.

Examples

- The following command identifies a scan chain segment that starts at pin `mod1/si` and ends at pin `mod1/so`:

```
set_scan_chain_segment -from mod1/si -to mod1/so
```

- The following commands define a segment `S1` guaranteed to be kept at the head of a chain:

```
set_scan_chain_segment -name S1 \
      {C1/R_reg_0 C1/R_reg_1 C2/CC1/R_reg_3 C2/CC1/R_reg_0 C1/R_reg_2}
set_scan_chain -head S1
```

- The following commands define a segment S1 guaranteed to be kept at the tail of a chain:

```
set_scan_chain_segment -name S1 \
      {C1/R_reg_0 C1/R_reg_1 C2/CC1/R_reg_3 C2/CC1/R_reg_0 C1/R_reg_2}
set_scan_chain -tail S1
```

- The following commands define a pair of segments (S1 and S2) guaranteed to be kept at the head and tail of one chain:

```
set_scan_chain_segment -name S1 \
      {C1/R_reg_0 C1/R_reg_1 C2/CC1/R_reg_3 C2/CC1/R_reg_0 C1/R_reg_2
set_scan_chain_segment -name S2 \
      {C3/CCC2/CC0/R_reg_0 C3/CCC2/CC0/R_reg_1}
set_scan_chain -head S1 -tail S2
```

- The following commands define two segments (S1 and S2) to be arbitrarily assigned to chains (instances are fixed relative to each other, but the segments are not required to be at the head or tail of any chain). Segment S3 must however appear at a tail of a chain.

```
set_scan_chain_segment -name S1 \
      {C1/R_reg_0 C1/R_reg_1 C2/CC1/R_reg_3 C2/CC1/R_reg_0 C1/R_reg_2}
set_scan_chain_segment -name S2 \
      {C3/CCC2/CC0/R_reg_0 C3/CCC2/CC0/R_reg_1}
set_scan_chain_segment -name S3 {C3/CCC1/R_reg_0 C3/CCC2/CC1/R_reg_3}
set_scan_chain -tail S3
```

Related Information

do_xform_connect_scan

remove_dft_assertions -scan_chain_segment

report_dft_assertions

set_scan_chain

set_global dft_insert_terminal_lockup_element

Inserting Terminal Data Lockup Latches at the End of the Scan Chain in *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*

Using Head and Tail Chain Constraints during Scan Chain Configuration and Scan Chain Reordering with PKS in *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*

set_scan_data

```
set_scan_data [-clock [clock [-rise | -fall]]]
     scan_in scan_out [-shared_out]
     [-enable scan_enable]
```

Specifies the names for the input and output scan data ports, as well as the following:

- Name for a port that the test synthesis tool creates
- Clock domain with which you want to associate the specified port names

If you want to use vectored ports for the scan data input and output ports, the scan data input and output ports must have the same bit widths. In this case the `set_scan_data` command is applied to each bit of the vectored ports.

Do not use `set_scan_data` during preliminary synthesis runs (that is, on lower-level modules). Let the test synthesis tool use default names for the scan data ports. In higher-level synthesis runs, the tool discards information that you have set with this command during previous synthesis runs.

Use the `set_scan_data` command during the final synthesis run of the entire design (when the tool connects the scan chains), to specify the desired scan data port names for each clock domain.

If a clock domain has multiple scan chains, you can invoke this command multiple times to give a different name to each scan chain data port. Otherwise, the tool uses the scan data ports specified without the `-clock` option or create default ports for the extra chains.

If you use the same port names for two different clock domains, the tool uses the port name for the clock domain specified first, and ignores the second invocation. For example:

```
set_scan_data -clock cka -fall sdia sdoa
set_scan_data -clock ckb -rise sdia sdoa *This assertion ignored*
```

If you do not use this command, the scan connection function creates or reuses the default scan port names which are derived from the setting of the `dft_scan_port_name_prefix` global. The settings from the `set_scan_data` command are cumulative, and apply to the current module set. To undo these settings, use the `remove_dft_assertions -scan_data_io` command.

If you want to create a configuration in a design database, specify the `set_scan_data` command prior to running the connection engine.

If you reorder scan chains in a scan-mapped structural netlist, specify the `set_scan_data` command prior to performing PKS reordering.

For more information on naming conventions see Naming Scan-Data Ports in *Design for Test (DFT) Using BuildGates Synthesis and Cadence PKS*.

Options and Arguments

`-clock [`*`clock`* `[-rise | -fall]]`

Specifies the clock domain. For a chain containing flip-flops driven by an internal clock domain, the tool first tries to match a `scan_data_port` command with the `-clock` option that matches the internal clock domain pin. If no match is found, then a second match will be made with the root clock pin (top level clock pin driving the internal clock signal).

`-enable` *`scan_enable`* `{`*`name`* `|` *`hiername`* `|` *`ID`*`}`

Designates a chain-specific scan-enable port/pin for the `muxscan` scan style. If you specify this option, you must use the `set_scan_mode` command to specify the active polarity of the scan-enable signal.

`name` must be a top-level port, if it exists. If not, the port will be created.

`hiername` must be an existing instance output pin (can be non-uniquified).

`ID` must be the ID of an existing top level input port, or uniquified instance output pin.

`scan_in` `{`*`name`* `|` *`hiername`* `|` *`ID`*`}`

Specifies the scan data input.

`name` should be top-level port/pin name, if it exists. If not, the port will be created.

`hiername` must be the name of an existing instance output pin (can be non-uniquified).

`ID` can be the ID of an existing top level input port, or hierarchical instance output pin (must be uniquified).

If you want to use a vectored scan data input port, the scan data input port must exist in the netlist and its bit width must correspond to the bit width of the specified scan data output port.

If the vectored port does not exist in the netlist, you can create it using the create_port command.

scan_out {*name* | *hiername* | *ID*}

Specifies the scan data output.

name should be top level port/pin name, if it exists. If not, the port will be created.

hiername must be the name of an existing instance input pin (can be non-uniquified).

ID can be the ID of an existing top level output port, or hierarchical instance input pin (must be uniquified).

If you want to use a vectored scan data output port, the scan data output port must exist in the netlist and its bit width must correspond to the bit width of the specified scan data input port. If the vectored port does not exist in the netlist, you can create it using the create_port command.

-shared_out

Specifies that the output port must be shared with a functional port using a mux.

Examples

- The following command specifies sdia and sdoa as scan data input and output ports for clock domain cka (falling edge):

```
set_scan_data -clock cka -fall sdia sdoa
```

- The following command specifies sin and sout as scan data input and output ports for all clock domains. If sin and sout are existing vectored ports with bit width 4, DFT associates sin[1] with sout[1], sin[2] with sout[2], sin[3] with sout[3], and sin[4] with sout[4].

```
set_scan_data sin sout
```

Related Information

check_dft_rules

dft_scan_port_name_prefix global

`report_dft_assertions`

`remove_dft_assertions -scan_data_io`

`set_number_of_scan_chains`

`set_scan_mode`

`write_scan_order_file`

Naming Scan-Data Ports in the *Design for Test (DFT) Using BuildGates Synthesis and Cadence PKS* manual.

set_scan_equivalent

`set_scan_equivalent` *`non_scan_cell scan_cell`*

Controls the scan-equivalent cell type that is used during the initial conversion of a non-scan flip-flop to a scan flop.

Initially, the specified scan-equivalent cell should match in functionality with the non-scan cell for the system mode operation. For example clock, set/reset, enable, data and so on. Later during optimization, the cell type can be replaced with a different scan instance to better meet timing and area constraints.

You can change the order of the arguments, because the tool automatically figures out which of the two cells is the scan cell and which one is the non-scan cell.

Note: You can also specify this information using the `scan_equivalent` construct for the non-scan cell in the TLF library.

Options and Arguments

`non_scan_cell`
Specifies the name of a non-scan cell.

`scan_cell`
Specifies the name of a scan cell.

Example

The following command specifies that cell `SCANREGD` can be used in the conversion of cell `REGD`. The tool figures out that `REGD` is the non-scan cell and that `SCANREGD` is the scan cell.

```
set_scan_equivalent  REGD  SCANREGD
```

set_scan_mode

`set_scan_mode [-default] scan_enable {0 | 1}`

Specifies the name of the input port that activates scan mode (scan shifting), and whether it is active-high (1) or active-low (0). A separate input port to activate scan mode is required by the muxed (`muxscan`) scan style. If you do not call this command, the tool creates a default port name using `set_global dft_scan_port_name_prefix`, which defaults to `BG_scan_enable`.

If the port already exists, you must ensure that any logic connected to the port is not needed in system mode. If the named port does not already exist, the test synthesis tool adds the scan mode port to the netlist when you call `do_xform_connect_scan`

This assertion is propagated to all (lower-level) child modules.

Options and Arguments

`{0 | 1}` Sets the active polarity desired for the scan enable signal.

`-default`
Indicates that the *`scan_enable`* specified is the default shift control for all scan chains. To specify chain-specific shift control pins, use `-enable` option in `set_scan_data` command. Optional.

`scan_enable`
Designates the port/pin used to control scan-shifting for the `muxscan` scan style. Follow the form:

{*`name`* | *`hiername`* | *`ID`*}.

`name` should be a top level input port, if it exists. If not, it will be created.

`hiername` must be an existing instance output pin name (can be non-uniquified).

`ID` must be the ID of an existing top level input port or internal instance output pin (but must be uniquified).

Examples

- The following command specifies that input port `scan_enable` activates scan mode when the applied scan enable signal is active high:

  ```
  set_scan_mode scan_enable 1
  ```

- The following command specifies that pin `jtag_instance/SE` activates scan mode when the signal is active high:

  ```
  set_scan_mode jtag_instance/SE 1
  ```

Related Information

`check_dft_rules`

`report_dft_assertions`

`remove_dft_assertions -scan_mode`

`set_scan_data`

`set_scan_style`

set_scan_style

```
set_scan_style [mux_scan | clocked_scan | clocked_lssd | aux_clocked_lssd]
```

Selects a scan style for the current module and all child modules. Currently, the test synthesis tool supports the muxed scan style, `muxscan, clocked_scan, clocked_lssd,` and `aux_clocked_lssd`.

You must set the appropriate scan style before running `check_dft_rules` since the DFT rules varies with different scan styles. If no scan style is set, the tool assumes `muxscan` style by default.

Options and Arguments

`aux_clocked_lssd`
Specifies the auxiliary scan style, using these commands:

```
set_lssd_aux_clock
set_lssd_scan_clock_a
set_lssd_scan_clock_b
```

`clocked_lssd`
Specifies the clocked lssd scan style, using these commands:

```
set_lssd_scan_clock_a
set_lssd_scan_clock_b
```

`clocked_scan`
Specifies clocked scan style, using this command:

```
set_test_scan_clock
```

`muxscan`
Specifies the muxed scan style.

Related Information

`check_dft_rules`

`report_dft_assertions`

Muxed Scan Style in the *Test Synthesis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)* manual.

set_test_mode_setup

```
set_test_mode_setup { input_port_id | input_port_name }
     { 0 | 1 } -scan_shift
     [-clock { clock_pin_name | clock_pin_id }]
```

Specifies the input port and constant value that is assigned during a test session. The test signal is asserted for the entire test session, unless you specify `-scan_shift` in which case it is assumed to be asserted only during the `scan_shift` cycle of test mode.

The `check_dft_rules` command ensures that this signal exists when checking for DFT rule violations.

Options and Arguments

`0 | 1`

Specifies the value for the test mode signal.

`-clock {`*`clock_pin_name | clock_pin_id`*`}`

Specifies that during low-power synthesis this test mode be used to bypass low-power clock gating logic for flip-flops clocked by the identified clock pin.

`input_port_id`

Specifies the input port identifier (ID) for the test mode signal.

`input_port_name`

Specifies the input port name for the test mode signal.

`-scan_shift`

Specifies that you want to assert the test signal only during the scan shift cycle of test mode. If you do not specify this option, the test signal is asserted for the entire test session, including the scan shift cycle.

Use this option for the test-mode setup of asynchronous signals, to keep them constant only during shifting of scan chains, and not during the capture stage of test application.

Examples

- The following commands specify that the active high value of test signal `u_jtag/test_mode` is asserted for the entire session, while the active high value of test signal `reset_L` is asserted during the scan shift cycle only:

```
set_test_mode_setup u_jtag/test_mode 1
set_test_mode -scan_shift reset_L 1
check_dft_rules
```

- The following commands specify that the active high value of test signal `scan_en` is asserted during the scan shift cycle, then propagates the change:

```
set_test_mode_setup scan_en -scan_shift 1
check_dft_rules
```

Related Information

`check_dft_rules`

`remove_dft_assertions -test_mode_setup`

`report_dft_assertions`

`reset_test_mode_setup`

set_test_scan_clock

```
set_test_scan_clock { clk_pin_name | clk_pin_id }
```

Sets the `scan_clock` pin for `clocked_scan` style.

You must specify this command along with the `clocked_scan` style (using the `set_scan_style` command) before running `check_dft_rules`.

Options and Arguments

`clk_pin_id`

Specifies the identifier (ID) of a top-level input port or of an output pin on an instance.

`clk_pin_name`

Specifies the name of a top-level input port or the hierarchical name of an output pin on an instance. If the specified top-level pin does not exist, an input pin with that name is created.

Example

The following commands specify `clockA` as the `scan_clock` pin for the `clocked_scan` scan style.

```
set_scan_style clocked_scan
set_test_scan_clock clockA
```

Related Information

`check_dft_rules`

`report_dft_assertions`

`set_scan_style clocked_scan`

write_atpg_info

```
write_atpg_info [-mentor | -syntest | -logicvision | -stil]
```

Creates an interface file containing the scan chain information in a format readable by the designated third-party ATPG tool. The third-party ATPG tool uses this interface file to generate appropriate test patterns. The file extension given to this interface file is determined by the selected third-party tool.

The interface file is useful only to the third-party tool if the test synthesis tool has connected the scan chain. Therefore, you should use this command only if the test synthesis tool connects the scan chains; that is, if you have set the `dft_scan_path_connect` global variable to `chain`.

The test synthesis tool does not currently add information about asynchronous controls or test mode ports to the interface file. If needed, you must add this information manually.

Note: You can also use the `get_scan_chain_info` command to get all necessary information about scan chains and generate necessary data for any ATPG tool.

Options and Arguments

`-logicvision`
: Creates an interface file in the format used by the Logic Vision ATPG tool. File generated:
`top_module``.LV`

`-mentor`
: Creates an interface file in the format used by Mentor Graphics ATPG tool. Files generated:
`top_module``.testproc`
`top_module``.dofile`

`-stil`
: Creates an interface file in the IEEE Standard Test Interface Language (STIL) format (IEEE format 1450). File generated:
`top_module``.stil`

`-syntest`
: Creates an interface file in the format used by Syntest Technologies ATPG tool. File generated:
`top_module``.dft`

write_scan_order_file

```
write_scan_order_file [-hier | -flat]
     [-verbosity {1 | 2 | 3}] filename
```

Generates a scan order file containing the scan chain information in the format specified. *Default*: A flat order file is created using the default file name: *set_current_module_name*.scan.flat.

Options and Arguments

`-hier` *filename*

Creates a hierarchical scan order file, organized by the modules with the design hierarchy.

`-flat` *filename*

Creates a flat scan order file, which contains a list of the complete scan chain in the top level module that uses full hierarchical path names, down to each register.

`-verbosity`

Sets the verbosity level where 1 = least verbose and 3 = most verbose.
Default: Off (`0`)

Examples

- The following command writes a hierarchical scan order file named `init.scan_order`:

  ```
  write_scan_order_file -hier init.scan_order
  ```

- The following command writes a flat scan order file named `./SO1`. By setting the verbosity level to 1, the number of messages generated is reduced to the minimum.

  ```
  write_scan_order_file -flat -verbosity 1 ./SO1
  ```

Related Information

`read_scan_order_file`

7

Common Timing Engine (CTE) Commands

This chapter contains information about the Common Timing Engine (CTE) commands used in timing analysis, as well as general information that applies to various commands:

Path Exception Priorities

The following are the path exception priorities if a path in the design matches more than one path exception:

1. `set_false_path`
2. `set_path_delay_constraint`
3. `set_cycle_addition`.

If there is more than one exception of a given type, for example the `set_cycle_addition` command, the path exception that is more specific has higher priority. A path exception is more specific if it specifies a longer path than the other. For example, the `-from -to` options would have priority over the `-from` option.

If the path has the same number of reference points:

- `-from` option has priority over the `-to` option
- `-to` option has priority over the `-through` option
- `-clock_from` option has priority over the `-clock_to` option

Note: To check for ignored path exceptions, use the `report_path_constraints` command.

Table 7-1 shows the priorities for path exceptions applied to the same path. See also "Examples of Path Exception Priorities" on page 798.

Table 7-1 Path Exception Priorities

Priority	Path Exception
1. (Highest)	`set_false_path`
2.	`set_path_delay_constraint -from` *`pin_list`*
3.	`set_path_delay_constraint -to` *`pin_list`*
4.	`set_path_delay_constraint -through` *`pin_list`* (The greatest number of throughs has the highest priority.)
5.	`set_path_delay_constraint -clock_from` *`clkwave_name`*
6.	`set_path_delay_constraint -clock_to` *`clkwave_name`*

Table 7-1 Path Exception Priorities

Priority	Path Exception
7.	`set_path_delay_constraint` (The most constraining adjustment has the higher priority over less constraining adjustments.)
8.	`set_cycle_addition -from` *`pin_list`*
9.	`set_cycle_addition -to` *`pin_list`*
10.	`set_cycle_addition -through` *`pin_list`*
11.	`set_cycle_addition -clock_from` *`clkwave_name`*
12.	`set_cycle_addition -clock_to` *`clkwave_name`*
13. (Lowest)	`set_cycle_addition` (The most constraining adjustment has the higher priority over less constraining adjustments.)

Examples of Path Exception Priorities

The following are examples of the path exception priorities for a path starting at A, going through B, and ending with C. The pairs of examples are ordered such that the highest priority constraint in one pair has precedence over the highest priority constraint in the next pair.

- In the following pair of constraints, the false path overrides the cycle addition for the path from A through B to C. For all other paths from A, the cycle addition applies.

```
set_false_path -from A -through B -to C
set_cycle_addition -from A 1
```

- In the following pair, the first constraint is applied (two cycles added) because the `-from` option has priority over the `-to` option.

```
set_cycle_addition -from A 2
set_cycle_addition -to C 1
```

- In the following pair, the first constraint is applied (two cycles added) because it specifies a `-through` pin as well as a `-to` pin.

```
set_cycle_addition -through A -to B 2
set_cycle_addition -to B 1
```

- In the following pair, the first constraint is applied (two cycles added) because the `-clock_from` option has precedence over the `-clock_to` option.

```
set_cycle_addition -clock_from CLKA 2
set_cycle_addition -clock_to CLKB 1
```

- In this pair of path exceptions, one cycle is added because it is the most constraining value.

```
set_cycle_addition -from A 1
set_cycle_addition -from A 2
```

Bidirectional Pin Defaults

Bidirectional pins have both an input pin and an output pin type. This section explains the default pin type that the system uses for bidirectional pins.

Defaults for Path Exceptions and Timing Reports

Table 7-2 shows the defaults for bidirectional pins used in path exceptions and timing reports.

Table 7-2 Bidirectional Pin Type Defaults for Path Exceptions

Pin Type	-from	-through	-to
Primary ports	`bidi_input_from`	`bidi_input_through`	`bidi_output_to`
Hierarchical ports	`bidi_input_from`	`bidi_input_through`	`bidi_output_to`
Instance pins	`bidi_output_from`	`bidi_output_through`	`bidi_input_to`
Blackbox pins	`bidi_output_from`	`bidi_output_through`	`bidi_input_to`

Tip

Override the default by specifying these options:

```
[-bidi_input_from | -bidi_output_from]
[-bidi_input_through | -bidi_output_through]
[-bidi_input_to | -bidi_output_to]
```

For more information, see the following:

- report_timing on page 990
- set_disable_timing on page 1153 and reset_disable_timing on page 1035
- set_false_path on page 1172
- set_path_delay_constraint on page 1230
- set_cycle_addition on page 1136

- reset_path_exception on page 1057
- Path Exception Priorities on page 797

Defaults for Pin (or Port) Assertions

Table 7-3 defines the pin and port types that are used in Table 7-4.

Table 7-3 Bidirectional Pin (or Port) Type Definitions

Pin Type	Type 1	Type 2
Blackbox pins	out	in
Instance pins	out	in
Netlist (primary) ports	in	out
Hierarchical ports	in	out

Table 7-4 Bidirectional Pin Type Defaults

Pin or Port Constraint	Type	Hierarchical Port
`set_external_delay`	2	ignored
`set_clock_insertion_delay`	1	ignored
`set_generated_clock`	1	used
`set_clock_info_change`	1	used
`set_slew_time`	1	ignored
`set_drive_cell`	1	ignored
`set_drive_resistance`	1	ignored
path exception (from)	1	used
path exception (through)	1	used
path exception (to)	2	used

Defaults for Port Assertions

Table 7-5 defines the port types that are used in Table 7-6.

Table 7-5 Bidirectional Port Type Definitions

Pin Type	Type 1	Type 2
Netlist (primary) ports	in	out
Hierarchical ports	in	out

Table 7-6 Bidirectional Port Type Defaults

Port Constraint	Type	Hierarchical Port
set_fanout_load_limit	2	ignored
set_fanout_load_limit	2	ignored
set_num_external_sources	2	ignored
set_num_external_sinks	2	ignored
set_port_capacitance	2	ignored
set_port_capacitance_limit	2	ignored
set_port_wire_load	2	ignored
set_slew_time_limit	2	ignored

Command Descriptions

This section contains descriptions of the commands used in the Common Timing Engine (CTE) for timing analysis.

check_timing

```
check_timing [-pins pin_list] [-type type_list] [-verbose]
     [-sort {pin | warning}] [-exclude_warning warning_list]
     [-early | -late] [-tcl_list] [-old] [{> | >> } file_name]

     Valid types are: clocks clock_clipping constant_collision endpoints inputs
     loops
```

Performs a variety of consistency and completeness checks on the timing constraints specified for a design. Use the `check_timing` command after setting all constraints but before using the `do_optimize` command or any CTE commands, such as `report_timing` or `get_timing`, to verify that the timing environment is complete and self-consistent.

The checks include arrival time and external delay (or required time) for each clock in a multiple clock system. In addition, clock connectivity and data connectivity are checked to make sure the clock or data is propagated as expected. Clock gating points are also reported. For more information about gated clock checks, see "Using Clock Gating" in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Use this command on generic or mapped netlists. The `check_timing` command considers constants applied to a pin while reporting warnings.

Important

Use the `check_timing` command before using any commands for timing analysis. Warnings are displayed when problems occur. Table 7-7 on page 806 lists the `check_timing` warnings. For best results, rerun the `check_timing` command after resolving each warning. Redirecting the output to a file is the quickest run method. For example: `check_timing filename`

Options and Arguments

`-early|-late`

Reports early or late timing inconsistencies.
Default: The late timing inconsistencies are reported.

`-exclude_warning warning_list`

Disables reporting of the warnings specified in the *`warning_list`*. The list of warnings and their descriptions is given in Table 7-7 on page 806.

file_name

Specifies the name of the output file.
Default: stdout

-old

Reports in the 4.0 format.

-pins *pin_list*

Limits the checks to pins in the *pin_list*.

-sort {pin | warning}

Sorts the Timing Check Detail table by pin name or warning.

-tcl_list

Reports the warnings in a Tcl list.

-type *type_list*

Limits the checks to the specified type.
Default: All check types are reported.

The valid types are as follows:

clock_clipping
Warns when there is a problem with clock gating that can result in clock clipping.

clocks
Warns if no clock arrives at register clock pins, if data arrives on a pin where a clock signal is expected, and if the master of a generated clock is not driven by a clock source.

constant_collision
Warns when there is a constant collision on a net connected to the pin or if there is a contradiction on a pin.
Constant collision occurs when a net is driven simultaneously by conflicting values and a constant contradiction occurs when a constant assertion on a pin conflicts with the driven value.

endpoints
For output ports, warns if no external delay (or required time) assertion is applied to the port or if a specific (late/early rise/fall) external delay (or required time) is missing. It also warns if an

external delay (or required time) assertion is applied with no clock information.

For all endpoints (output ports and register data pins), it warns if any incoming signal is unconstrained.

Also warns if a clock arrives where clock is not expected or if multiple signals arrive at an endpoint.

`inputs`
Issues warnings if no input delay (or arrival time) assertion is applied to an input port or if a specific (late/early rise/fall) input delay (or arrival time) is missing on that port.

Warns if an input delay (or arrival time) assertion is applied with no clock information.

Warns if no drive assertion is applied to an input port or if a specific (late/early rise/fall) drive assertion is missing on that port.

`loops`
Warns when a combinational loop is detected. Reports the loop and the arc used to break the loop. The `-pin` option is ignored in this case.

Also reports loops caused by generated clocks.

`-verbose`

Gives detailed information about the warnings.
Default Gives a summary of how many warnings of each type exists.

Note: This option replaces the `-detail` option, which is obsolete.

Warning Messages

Table 7-7 shows the `check_timing` warning messages.

Table 7-7 check_timing Warning Messages

Warning	Description	Comments
`async_arr`	Input delay or arrival time assertion with no clock information (asynchronous).	Input delay or arrival time assertion applied without the `-clock` option. Do not mix an asynchronous clock (@) with a regular clock.
`async_ext`	External delay or required time assertion with no clock information (asynchronous).	External delay or required time assertion is applied without the `-clock` option. Do not mix an asynchronous clock (@) with a regular clock.
`clock_but_data`	Clock signal found where data is expected.	See the `reset_clock_root` and `set_generated_clock`.
`clock_clipping_gate`	Clock clipping possible due to wrong gate type or wrong trigger for data input.	
`clock_clipping_freq`	Clock clipping possible due to incompatible clock signal and data signal frequencies.	
`clock_expected`	Clock not found where clock is expected.	Indicates that no clock signal is defined at the clock pin. Typically, accompanies `data_but_clock` warnings. See the `set_clock_root`.
`clock_not_propagated`	Clock not propagated.	Indicates that the clock is not propagated further because it is connected to a black box or signal pin of a timing check.
`const_collision`	Constant collision.	See `set_constant_for_timing`.

Table 7-7 check_timing Warning Messages , *continued*

Warning	Description	Comments
const_contradiction	Constant contradiction.	See set constant for timing.
data_but_clock	Data signal found where clock is expected.	
data_gating_clock	Data gating clock.	A warning that the clock is a gated clock.
loop	Timing loop found in the design.	Indicates that there are timing loops in the design. It also includes loops caused by generated clocks.
missing_arr	Missing specific (rise/fall) input delay or arrival time assertion.	Either a rise or fall input delay or arrival time assertion is applied and the other one is missing.
missing_drive	Missing specific (rise/fall) drive assertion.	Either a rise or fall drive assertion is applied and the other one is missing.
missing_ext	Missing specific (rise/fall) external delay or required time assertion.	Either rise or fall external delay or required time assertion is applied and the other one is missing.
multiple_signal	Multiple signals arriving at end points.	Indicates that there are signals triggered by different clocks arriving at the pin.
no_drive	No drive assertion.	See set_drive_cell and set drive resistance
no_ext	No external delay or required time assertion.	See set external delay.
no_gen_clock_source	No clock source found for generated clock.	Indicates that the source pin of the clock generated by the set generated clock command is not driven by a clock source.

Table 7-7 check_timing Warning Messages , *continued*

Warning	Description	Comments
specific_async_arr	Specific (rise/fall) input delay or arrival time assertion with no clock information (asynchronous).	Either a rise or fall input delay or arrival time assertion is applied without the -clock option. Do not mix an asynchronous clock (@) with a regular clock.
specific_async_ext	Specific external delay or required time assertion with no clock information (asynchronous).	Either rise or fall external delay or required time assertion is applied without the -clock option. Do not mix asynchronous clock (@) with regular clock.
uncons_signal	Unconstrained signal arriving at end point.	Signal arriving at an end point is not constrained. Use the set external delay command to constrain it.
unmatched_req	Unmatched required time assertion.	Required time does not match any incoming signal. Check for bad constraints.

Examples

- The following command displays the default summary report:

```
check_timing
```

```
+-------------------------------------------------------------------------+
|                           TIMING CHECK SUMMARY                          |
|-------------------------------------------------------------------------|
|    Warning     |             Warning Description              |  Number  |
|                |                                              |    of    |
|                |                                              | Warnings |
|----------------+----------------------------------------------+----------|
| clock_expected | Clock not found where clock is expected      |        2 |
| loop           | Timing loop found in the design              |        1 |
| no_arr         | No input delay or arrival time assertion     |        6 |
| no_ext         | No external delay or required time assertion |        1 |
+-------------------------------------------------------------------------+
```

- The following command displays a specified type of warning using the -type option:

```
check_timing -type loops
```

```
+-----------------------------------------------------+
|                 TIMING CHECK SUMMARY                |
|-----------------------------------------------------|
| Warning |       Warning Description       |  Number  |
|         |                                 |    of    |
```

```
|         |                                    | Warnings |
|---------+------------------------------------+----------|
| loop    | Timing loop found in the design |        1 |
+-------------------------------------------------------+
```

- The following command provides more information about problems and prints a summary using the `-detail` option:

```
check_timing -detail
```

```
+-------------------------------------------------------------------------+
|                           TIMING CHECK SUMMARY                          |
|-------------------------------------------------------------------------|
|    Warning     |             Warning Description              |  Number  |
|                |                                              |    of    |
|                |                                              | Warnings |
|----------------+----------------------------------------------+----------|
| clock_expected | Clock not found where clock is expected      |        2 |
| loop           | Timing loop found in the design              |        1 |
| no_arr         | No input delay or arrival time assertion     |        6 |
| no_ext         | No external delay or required time assertion |        1 |
+-------------------------------------------------------------------------+
+-----------------------------------------------------+
|                  TIMING CHECK DETAIL                |
|-----------------------------------------------------|
|  Pin   |                  Warning                   |
|--------+--------------------------------------------|
|  clk1  | No input delay or arrival time assertion    |
|  clk2  | No input delay or arrival time assertion    |
| enable | No input delay or arrival time assertion    |
| ff1/CP | Clock not found where clock is expected     |
| ff2/CP | Clock not found where clock is expected     |
|  in1   | No input delay or arrival time assertion    |
|  in2   | No input delay or arrival time assertion    |
|  in3   | No input delay or arrival time assertion    |
|  out1  | No external delay or required time assertion |
+-----------------------------------------------------+
+------------------------------+
|      TIMING CHECK LOOP       |
|------------------------------|
|   Snipped Arcs    |  Timing  |
|                   |  Loop    |
|                   | Details  |
|-------------------+----------|
|  From   |  To Pin |          |
|   Pin   |         |          |
|---------+---------+----------|
|  g1/Z   |  g1/A   |  g1/A    |
|         |         |  g1/Z    |
+------------------------------+
```

create_mp_constraint_arc

```
create_mp_constraint_arc [-name constraint_name]
     [-setup | -hold | -recovery | -removal | -no_change_setup | -no_change_hold
     | -skew | -mpw | -period] [-from port_name_list] [-edge {rise | fall}]
     [-to port_name_list]
     [{-path_type path_type|-path_factor factor|-value extra_constraint_time}]
```

Specifies the value for a timing check. Most checks relate a transition on a clock input port to a transition on a data input port. The minimum pulse width (mpw) and period checks have only a clock pin.

The constraint arc starts with a base value found with the set_mp_global_parameter command with a matching -param option. To this base constraint value, an internal delay is added. This delay is either given explicitly or from a path_type.

Options and Arguments

-edge

Specifies the edge of the clock, *rise* or *fall*, relevant to the timing check.

-from *port_name_list*

Specifies the name of the clock input port to use for the timing check.

The *port_name_list* argument can be either a single name or a list of names in curly braces. Each name can be a simple name, such as A, a bus element name, such as Data[3], a bus name, such as Data, or a bus range, such as Data[3:6]. The current name space is used for special character escaping. See Port Names in the Using the Module Prototyper (MP) section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

-name *constraint_name*

Specifies the constraint arc name, which is not used.

-hold

Specifies a hold constraint arc.

-mpw

Specifies a minimum pulse width constraint arc.

`-no_change_hold`
: Specifies a `no_change_hold` constraint arc.

`-no_change_setup`
: Specifies a `no_change_setup` constraint arc.

`-path_factor` *`factor`*
: Specifies a multiplicative factor for the path type.

`-path_type` *`path_type`*
: Specifies the extra constraint time based on a path type. The extra constraint time is the delay time from the path type times the path factor.

`-period`
: Specifies a period constraint arc.

`-recovery`
: Specifies a recovery constraint arc.

`-removal`
: Specifies a removal constraint arc.

`-setup`
: Specifies a setup constraint arc.

`-to` *`port_name_list`*
: Specifies the input port name to use for the timing check. For slew arcs, this should be another clock. This argument is not used for mpw and period arcs.

`-value` *`extra_constraint_time`*
: Specifies an explicit extra constraint time, which can be either a single number or an SDF style triplet.

Examples

- The following command creates a hold constraint arc from `cp` to `D`. The base constraint value is set by the `set_mp_global_parameters`. To that base, 2 is added as an extra constraint time:

```
create_mp_constraint_arc -hold -from cp -to D -edge rise -value 2
```

- The following command creates a setup constraint arc. The base constraint value is set by the `set_mp_global_parameters`. To that base, an extra constraint time is added, which is found by multiplying the path delay of `path1` by the path factor:

```
create_mp_constraint_arc -setup -from cpn -to D -edge rise -path_type path1
-path_factor 2
```

Related Information

`create_mp_path_type`

`set_mp_global_parameter`

create_mp_delay_arc

```
create_mp_delay_arc [-name arc_name]
     [-from port_name_list]
     [-to port_name_list]
     [-edge {rise | fall}| {-set | -clear}]
     [-value delay_time] [-path_type path_type] [-path_factor factor]
```

Creates a delay arc. By default, it is a purely combinational delay arc from all input ports to all output ports.

All delay arcs created by the Module Prototyper (MP) are non-inverting except for the edge arcs.

The delay and slew tables for the arc are based on the tables implied by the `set_mp_port_drive` command on the `-to` pin. An internal delay time is added to the delay table to create the delay table for the specific arc. This internal delay time can be found either by explicitly stating the time using the `-value` option or by using the `path_type` option. In the later case, the delay associated with the path type is multiplied by the path factor.

Edge arcs also add the `clk_to_output` delay set by the `set_mp_global_parameter -param clk_to_output` command.

If there are multiple calls to `create_mp_delay_arc` for the same pair of `from` and `to` ports, the model will have multiple delay arcs between these ports.

Options and Arguments

`-edge {rise | fall}`

Specifies the delay arc is either a rising edge or falling edge arc. The from ports for this arc must be clock pins.

`-from` *`port_name_list`*

Specifies the name of the input port or ports. These ports must be either input or inout pins. Edge arcs must have clock pins as the from pin. If this option is not specified, then delays arcs will come from all appropriate ports.

The `port_name_list` variable can be either a single name or a list of names in curly braces. Each name can be a simple name, such as A, a bus element name, such as Data[3], a bus name, such as Data, or a bus range, such as Data[3:6]. The current name space is used for special character escaping. See Port Names in the “Using the Module Prototyper (MP)” section of

the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-name` *`arc_name`*

Specifies an optional name for the from-to arc to which the delay applies. The arc name can be specified but is not used.

`-path_type` *`path_type`*

Specifies the path type to be used for the delay. The default for path factor is 1. The name of the basic path, *`path_type`*, must be previously defined using the `create_mp_path_type` command.

`-path_factor` *`factor`*

Specifies the delay based on the path type multiplier by a factor. The default for the path factor is `1`.

`-set | -clear`

Specifies that the delay arc is an asynchronous set or clear arc. The `-set` option creates a `set` asynchronous delay path. This arc will have a `SET QUALIFIER` if stored in a TLF file or a `timing_type` of `preset` in a Liberty file.

The `-clear` option creates a `clear` asynchronous delay path. This arc will have a `CLEAR QUALIFIER` if stored in a TLF file or a `timing_type` of `clear` in a Liberty file.

`-to` *`port_name_list`*

Specifies the name of the output port or ports, which must be an output or an inout port. If this option is not specified, delays will be created to all output and inout ports.

`-value` *`delay_time`*

Specifies the internal delay as an explicit value, which can be either be a single number or an SDF style triplet.

Examples

- The following command creates a delay arc from port `cp` to `Q`. It is a rising edge arc and has a constant delay of 2 for the minimum corner and 3 for the maximum corner (there is no typical corner for this prototype, otherwise the `typ` value must be provided):

```
create_mp_delay_arc -from cp -edge rise -value 2::3 -to Q
```

- The following command creates two delay arcs from port A to port Z and from port B to port Z. The delay and slew data is based on the driver type port Z was set to (if any). An extra delay is added, found by multiplying the path delay of `path1` by `1`:

  ```
  create_mp_delay_arc -from {A B} -path_type path1 -path_factor 1 -to Z
  ```

Related Information

`create_mp_path_type`

`set_mp_global_parameter`

create_mp_drive_type

```
create_mp_drive_type [-lib_cell cell_name][-input_pin pin_name]
     [-output_pin pin_name] drive_type_name
```

Specifies a drive type that describes a drive type that describes drives. The drive type can then be applied to an out or inout port using the set_mp_port_drive command. All delay arcs to ports set to this drive type uses the reference delay arc as a base time. If the `inherit_port_limits` option was set to on using the set_mp_technology command, then any port limits on the reference's cell driver pin are copied to the driver port.

A delay arc is found in the reference arc for the matching cell name and pin names. If multiple arcs match, the first one is used. A capacitive load found for that arc is either given explicitly or based on the fanout count and the current Module Prototyper (MP) wire load table. A delay is then found for the output rising arc using that capacitance for the load and zero for all other axes. This delay is used whenever this path type is used.

The *pin_name* variable is the name of a single pin the reference cell.

Bus names or bus ranges are not allowed.The current name space is used for special character escaping. See Pin Names in the "Using the Module Prototyper (MP)" section of the *Common Timing Engine* (*CTE) User Guide*.

Options and Arguments

drive_type_name
: Specifies the name for the drive type. The drive type is stored using this name,

-input_pin *pin_name*
: Specifies the input pin for the arc. If missing, the first input or inout will be used.

-lib_cell *cell_name*
: Specifies the cell in the library for which the logic type is defined.

-output_pin *pin_name*
: Specifies the output pin for the arc. If this option is not specified, the first output or inout pin will be used.

Examples

- The following command sets the `drive1` drive type to the delay arc from cell `INVX1`'s first input pin to the cell's first output pin:

  ```
  create_mp_drive_type drive1 -lib_cell INVX1
  ```

- The following command sets the `drive2` drive type to the delay arc from pin `A` to pin `Z` in the reference cell `INXV1`:

  ```
  create_mp_drive_type drive2 -lib_cell INVX1 -input_pin A -output_pin Z
  ```

Related Information

`create_mp_constraint_arc`

`create_mp_delay_arc`

create_mp_load_type

`create_mp_load_type [-lib_cell` *`cell_name`*`] [-input_pin` *`pin_name`*`]` *`load_type_name`*

Specifies a load type that describes the load of a input or inout reference pin. The load type created by this command can be used with the `set_mp_port_load` command where it is used to set the port's capacitance and optionally other port limits.

Options and Arguments

`-input_pin` *`pin_name`*

Specifies the input pin for the arc. If this option is not specified, the first input or inout will be used.

The *`pin_name`* argument specifies the name of a single pin for a reference cell. Bus names or bus ranges are not allowed.The current name space is used for special character escaping. See Pin Names in the "Using the Module Prototyper (MP)" section of the *Common Timing Engine (CTE) User Guide*.

`-lib_cell` *`cell_name`*

Specifies the cell in the library for which the logic type is defined.

`load_type_name`

Specifies the name for the load type. The load type will be stored using this name.

Examples

- The following command creates a `load1` load type from the pin information on the first input or inout pin in the `INVX1` reference cell:

  ```
  create_mp_load_type load1 -lib_cell  INVX1
  ```

- The following command creates a `load2` load type from the pin information on pin `Z` in the `INVX1` reference cell:

  ```
  create_mp_load_type load1 -lib_cell  INVX1 -input_pin A -output_pin Z
  ```

Related Information

`set_mp_port_load`

create_mp_model

```
create_mp_model [-copy_ports_from_module_or_inst_name] cell_name
```

Starts the definition of a new module prototype. The *cell_name* will be the name for both the model cell and the library it is stored in. If the `-copy_ports_from` option is specified, the Module Prototyper (MP) will look at the current design to find a module or an instance with that name. If found, the model will be created with the same ports and names as in the design module or instance. The name and numbers of ports will then be fixed, but you can use the `create_mp_port` command to change the port type.

The `create_mp_model` command starts an MP edit session and the `save_mp_model` command ends the session. The prototype is visible only to MP commands until it is saved.

Options and Arguments

cell_name
: Specifies the name of the module that is being prototyped. The same name will be used both as the name of the library containing the MP prototype, and the name of the prototype cell within the library.

`-copy_ports_from_module_or_inst_name`
: Specifies a module or an instance in the current design whose port number and names the prototype model should match.

Examples

- The following command creates a new MP prototype with no pins:

  ```
  create_mp_model invert
  ```

- The following command creates a new MP prototype, which has the same pin names and pin order as `block5` in the current netlist:

  ```
  create_mp_model -copy_ports_from block5 mp_block5
  ```

Related Information

`save_mp_model`

create_mp_path_type

```
create_mp_path_type [-lib_cell cell_name] [-input_pin pin_name]
     [-output_pin pin_name] [-fanout fanout_count | -capacitance cap]
     path_type_name
```

Specifies a basic path type that describes delays through one or more levels of combinational logic. The information in the command line is converted to a simple delay value (the delay time for different PVT corners may differ). The path type is referred to in the create_mp_delay_arc and create_mp_constraint_arc commands to refer to an additional delay time to be added to a delay or constraint time.

A delay arc is found in the reference arc for the matching cell name and pin names. If multiple arcs match, the first one is used. A capacitive load is found for that arc either given explicitly or based on the fanout count and the current Module Prototyper (MP) wire load table. A path delay is then found for the output rising arc using that capacitance for the load and zero for all other axes. This path delay is used whenever this path type is used to create an arc.

Options and Arguments

-capacitance *cap*

Specifies an explicit load for the arc when calculating the delay for the path. It may either be a single number or an SDF style triplet.

-fanout *fanout_count*

A load is calculated using the fanout_count and the *wireload_model* specified in the set_mp_technology command. This capacitance is used as the load for the delay arc when calculating the delay of the path.

-input_pin *pin_name*

Specifies the input pin for the arc. If missing, the first input or inout will be used.

The pin_name is the name of a single pin in the reference cell. Bus names or bus ranges are not allowed.The current name space is used for special character escaping. See Pin Names in the “Using the Module Prototyper (MP)” section of the *Common Timing Engine* (*CTE) User Guide.*

-lib_cell *cell_name*

Specifies the cell in the library for which the logic type is defined.

`-output_pin` *`pin_name`*

Specifies the output pin for the arc. If this option is not specified, the first output or inout pin will be used.

`path_type_name`

Specifies a name for the path type. The path type will be stored using this name.

Examples

- The following command creates a path type based on the delay arc between the first input and the first output pin in the `INVX1` reference cell. The Module Prototyper's current wireload table is used to find a capacitance for a fanout of `5` and this load is used to calculate the path delay:

  ```
  create_mp_path_type path1 -lib_cell INVX1 -fanout 5
  ```

- The following command creates a path type based on the delay arc between pin `A` and pin `Z` in the `INXV1` reference cell. The path delay is calculated using `0.09` as the load:

  ```
  create_mp_path_type path12 -lib_cell  INVX1 -input_pin A -output_pin Z
  -capacitance 0.09
  ```

Related Information

`create_mp_constraint_arc`

`create_mp_delay_arc`

create_mp_port

`create_mp_port [-type {input | -output | -inout | -clock}] port_name_list`

Adds ports of the specified type to the prototype or changes the type of existing ports. How it is used depends on how the Module Prototype (MP) was created.

If the `create_mp_model` command used to start this MP session specified the `-copy_ports_from` option, then the port names and order are already set. The ports names cannot be changed. If the ports were copied from a Verilog or VHDL module, the port types are also set and this command is not necessary. However, if the ports were copied from a black box instance, the port types are set to inout and this command should be used to set the port types.

If the `create_mp_model` command used to start this MP session did not specify the `-copy_ports_from` option, then the `create_mp_port` command must be used to create all the ports. When you use this command, make sure that the connections to the pins of an instance are correct by order (rather than by name). To do this, you must issue the command in the same order in which the ports appear in the Verilog or VHDL definition of the module. Similarly, bus ports must be created using the bus port notation. The order of the indices (ascending or descending) must match that appearing in the Verilog or VHDL definition of the module.

Options and Arguments

`port_name_list`
: Specifies the name of the created or modified ports.

  The *`port_name_list`* argument can be either a single name or a list of names in curly braces. Each name may be a simple name, such as A, a bus element name, such as Data[3], a bus name, such as Data, or a bus range, such as Data[3:6]. The current name space is used for special character escaping. See Port Names in the “Using the Module Prototyper (MP)” section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-type (input | -output | -inout | -clock)`
: Specifies the type of port to be added to the module prototype.

Examples

- The following command creates two input ports A and B in that order:

```
create_mp_port -type input { A  B}
```

- The following command creates a clock port named cp:

```
create_mp_port -type clock cp
```

- The following command changes the pin type of a port previously created by copying from an existing `inst5` black box in the current design:

```
create_mp_model -copy_ports_from inst5 inst5_mp
create_mp_port -type input A
```

Related Information

`create_mp_model`

do_analyze_crosstalk

```
do_analyze_crosstalk [-engine {xmc | celtic} [-coupling_file coupling_file]
     [-dont_read_coupling_file] [-config configfile] [-twf_based {net | pin}]
     [-no_timing] [-which] [-spef parasitics.spef] [-spf parasitics.spf]
     [-keep_inc sdf][-exefile exe_full_pathname] [-echo file_list]
     [-exename exe_filename] [-cdb cdb_filename] [-text] [-outprefix prefix]
```

Runs the crosstalk analyzer engine on input from RC extraction (HyperExtract) and produces a crosstalk repair file used by the `do_xform_run_repair_file` command or downstream tools. The engine is CeltIC or SE5.4 or the Crosstalk Magnitude Calculator (XMC) for SE5.3. This command supports the SE-PKS flow.

You cannot use the `do_analyze_crosstalk` command if you only have a license for Cadence PKS. You must have the appropriate licenses to run HyperExtract and Celtic or XMC. Run RC extraction on a fully routed design before you analyze crosstalk. For more information, see the *Synthesis Place and Route (SP&R) Flow Guide*.

If the Celtic license (FEATURE Celtic in license file) is not available, then the CTE `do_analyze_crosstalk` command asks CeltIC to check out the `Envisia_SE_SI_place_route` license. If the SESI license is not available, then CTE exits CeltIC. Otherwise, crosstalk analysis starts with the SESI license checked out.

Output Files

When the `do_analyze_crosstalk` command completes successfully, it produces the following files in the run directory:

- `xtalk.log` —Crosstalk analysis log file.
- `xtalk.gcf` — Intermediate General Constraint Format (GCF) timing constraint file required for boundary timing, drive, and load information (XMC only).
- `xtalk.twf` — Timing windows file generated by the `do_analyze_crosstalk` command. This is an internal handshake from PKS to the crosstalk analyzer only. You do not need to do anything with this file, the tools handle the handshake during each run.
- `xtalk.incsdf` — Incremental Standard Delay Format (SDF) file containing the delay effects of the induced crosstalk. Unless the `-no_timing` option is specified, this SDF is loaded when the crosstalk analysis is completed.

Additional files are produced for analysis and repair.

CeltIC produces the following files:

- `xtalk.vg`
- `xtalk_eco.html` — Repair file used with the `do_xform_run_repair_file` PKS command.
- `xtalk_noise_html`
- `xtalk_sens_html`
- `celtic.tcl` —Tcl file containing the runtime commands to CeltIC. Specify this file in a later run using the `-config` option.

For information on what the files represent, see the *CeltIC User Guide*.

XMC produces the following files:

- `xtalk.repair` — ASCII file containing the nets to be repaired due to crosstalk induced logic problem along with the method used for the repair. For example:

  ```
  ReduceNetC net1 0.5
  ```

- `xtalk.rpt` — Output report file
- `xtalk.err` — Crosstalk errors file

Options and Arguments

`-cdb` *`cdb_filename`*

Specifies a CeltIC-specific binary database file used for signal integrity (SI) analysis. The file, *`cdb_filename`*, is passed to `wrap.celtic.pl`.

Note: Coupling files that are suffixed with `.spf` or `.spef` are automatically read in to CeltIC using the `do_analyze_crosstalk` command.

`-config` *`configfile`*

Specifies an optional configuration file for running CeltIC (or XMC). Recommended for advanced users only. Use this option to pass additional information or settings (CeltIC constraints) that CeltIC will take into account when doing crosstalk analysis.

For information on CeltIC specific settings, see the *CeltIC User Guide*.

`-coupling_file` *`coupling_file_list`*

Specifies the input for CeltIC (or XMC) from HyperExtract. Required argument.

If you are using CeltIC, you can pass multiple coupling files in a Tcl list. CeltIC can read cross-coupling information specified in Standard Parasitic Format (SPF), Standard Parasitic Exchange Format (SPEF), or HE-xcoup file format. The coupling files must be in the same format. See Examples on page 829.

If you are using XMC, you cannot pass multiple coupling files. XMC only accepts HE-xcoup file format, not SPF or SPEF.

If the *`coupling_file_list`* contains files in SPF or SPEF format, the `do_analyze_crosstalk` will read the *`coupling_file_list`* and take into account the RCs specified in the file. When reading coupling files, coupling capacitances will be treated as grounded capacitance.

`-dont_read_coupling_file`

Does not read the file specified in the `-coupling_file` option. Without this option, the `do_analyze_crosstalk` will read the `coupling_file_list`. This option is useful when you want to do timing analysis with parasitics being annotated before calling the `do_analyze_crosstalk` command. In this case, you will read the parasitics file, analyze the timing report, then use the `do_analyze_crosstalk` command with the `-dont_read_coupling_file` option.

`-echo` *`file_list`*

Allows the `do_analyze_crosstalk` command to take ECHO or UDN files. The file names are passed to CeltIC.

`-engine xmc | celtic`

Specifies the engine to use to compute crosstalk effects. Use `xmc` to specify the XMC engine.
Default: `celtic`

When using the celtic engine, CeltIC does not support the signal integrity (SI) analysis of two mixed corner-case timing libraries. For example:

```
read_tlf -min $rootDir/links/tsmc13_ff.tlf -max
$rootDir/links/tsmc13_ss.tlf -name tsmc13
```

CeltIC only supports the use of single corner-case timing libraries each time SI is performed.

Note: Supply the proper license and executable for the crosstalk calculator you intend to use.

`-exefile` *`exe_full_pathname`*

Specifies a non-default executable file using the complete path name.
Default: The executable for XMC is `xtalk_file` and the executable for CeltIC is `wrap_celtic.pl`.

`-exename` *`exe_filename`*

Specifies a non-default executable file using only the file name. CTE searches for the executable in your `$path`.
Default: The executable for XMC is `xtalk_file` and the executable for CeltIC is `wrap_celtic.pl`.

`-keep_incsdf`

Preserves the incremental SDF file that is generated by CeltIC.

`-no_timing`

Prevents Celtic from doing incremental delay analysis so it will not generate incremental SDF. If specified, Celtic completes noise and glitch analysis.

`-outprefix` *`prefix`*

Specifies a prefix for the output files. The default prefix of `xtalk` names the files as shown in Output Files on page 824. This is useful when you want to do multiple crosstalk analysis runs using different sets of data. For example, the results of the first run could have a prefix of `xtalk1`, and second run results a prefix of `xtalk2`. You can specify a directory name in the prefix, for example: `/net/projects/mydesign/analysis1/xtalk1`.

`-spef $path/parasitics.spef`

Provides an alternative to using the `read_spef` command before the `do_analyze_crosstalk` command. Reading net

parasitics data is a requirement before crosstalk analysis can be done.

Note: If the SPEF file contains cross-coupling information, you must pass the file to CeltIC using the `-coupling_file` option. Files that are passed to PKS using `-spef` option are not automatically passed to CeltIC unless they have suffix `.spef`.

Timing windows produced for CeltIC crosstalk analysis are effected by the parasitics file. The tool treats cross-coupling capacitance in a parasitics file as grounded capacitance.

`-spf $path/parasitics.spf`

Provides an alternative to issuing the `read_spf` command before the `do_crosstalk_analysis` command. Reading net parasitics data is a requirement before crosstalk analysis can be done.

Note: If the SPF file contains cross-coupling information, pass the file to CeltIC using the `-coupling_file` option. Files that are passed to PKS using the `-spf` option are not automatically passed to CeltIC unless they have suffix `.spf`.

Timing windows produced for CeltIC crosstalk analysis are effected by the parasitics file. The tool treats cross-coupling capacitance in a parasitics file as grounded capacitance.

`-text`

Specifies that output files are to be written out as text rather than HTML.

`-twf_based net | pin`

Lets you choose between pin-based timing windows or net-based timing windows when running crosstalk analysis.
Default: `net`

`-which`

Gives a message indicating which `exe` file is executed. This option is useful because `do_analyze_crosstalk` tries to find the executable in several different paths.

Examples

- The following command shows the usage if the parasitics file has not been previously read by `read_spf:`

  ```
  do_analyze_crosstalk -coupling_file routed.xcoup -spf routed.rspf
  ```

- The following command shows how to use multiple SPEF files with CeltIC.

  ```
  do_analyze_crosstalk -spef {parasit1.spef parasit2.spef} -coupling xcoup.spef
  ```

  PKS reads in `parasit1.spef` and `parasit2.spef` before it computes timing windows. Since the coupling file `xcoup.spef` is suffixed with `.spef`, PKS also reads this before producing timing windows. However, the only parasitics file that is passed to CeltIC is the coupling file, `xcoup.spef`.

- The following command reads in all parasitics files before it computes timing windows. All parasitics files are passed files to CeltIC:

  ```
  do_analyze_crosstalk -coupling "parasit1.spef parasit2.spef xcoup.spef"
  ```

Related Information

Synthesis Place and Route (SP&R) Flow Guide

CeltIC User Guide

do_cppr_analysis

`do_cppr_analysis [-slack_limit slack_limit] [-critical] [-early | -late]`

Removes delay pessimism from paths that have a portion of the clock network in common.

Removes both clock networks as well as clock source pessimism. Common path pessimism removal (CPPR) is the process of computing delay adjustments through a clock network and at clock source to remove the pessimism introduced at various checks. All timing checks compare a latest arriving signal against an earliest arriving signal. If both signals share a portion of the clock network, then a pessimism equal to the difference in late and early arrival times at nearest common pin, is introduced.

The results of this command are reflected in the reports generated by the `report_timing` command.

For more information, see Removing Common Path Pessimism" in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Note: Currently only simple clock trees are supported. If there are any parallel buffers or clock muxes, CPPR calculation stops with an error message.

Options and Arguments

`-critical`
: Stops the common path pessimism removal process at each endpoint, as soon as the most critical path to that endpoint has its pessimism removed. Use this option as an efficiency measure to reduce CPU and memory usage.

`-early | -late`
: Specifies pessimism removal from either early or late data paths. *Default*: The command removes pessimism from both early and late paths.

`-slack_limit slack_limit`
: Removes clock pessimism only from paths whose slack is worse than the specified slack limit. Use this option as an efficiency measure to reduce CPU and memory usage.

Example

The following command removes clock pessimism from paths whose slack is worse than the `3.2`:

```
do_cppr_analysis -slack_limit 3.2
```

Related Information

`set_global pvt_early_path`

`set_global pvt_late_path`

`set_operating_conditions`

do_derive_context

`do_derive_context [`*`list_of_instances`*`]`

Builds the characteristics of the design from its environment constraints. A context-based optimization approach requires that constraints for lower level modules be derived from the higher level modules. This is one of the fundamental capabilities for performing top-down hierarchical synthesis. The constraints set at the top level of the design are pushed down the hierarchy so that each of the lower level modules are optimized using the correct set of constraints. Later, as the higher level modules are grouped together, the optimized modules are all connected correctly with respect to each other.

Tip

Use either the `do_time_budget` or `do_derive_context` command in CTE. For PKS, use the `do_push` command instead. Budgeted constraints cannot be used in a third party tool.

The `do_derive_context` command derives the worst condition for modules among all the instances specified in the list of instances in a similar manner to that performed by the time budgeting procedure.

The context is derived by extracting three types of constraints from the surrounding environment of the instances in the instance list. The name of the instances can be hierarchical and are related to the current module:

- Interface timing constraints

  Includes timing information about clock definitions, arrival time on input ports, required time on output ports, and identification of multicycle and false paths.

- Electrical constraints

  Includes port capacitance and resistance, number of drivers, and load for every port of every instance.

- Logic value constraints

  Includes the logic constants that block timing analysis.

Note: The derived context information is stored in the database. Use the `write assertions` command to store the constraints in a file.

Options and Arguments

list_of_instances

Specifies the list of instance names for which the context is derived. The instance name can be specified by a hierarchical path name.

The instance name is relative to the current module.

If the instance list is not provided, the default is the current module.

The current module must be unique in the context of the top timing module. If not, run the `do_uniquely_instantiate` command.

Examples

```
do_derive_context [find -inst *]
do_derive_context blkA/xbar
```

Related Information

See "Using the do_derive_context Command with Backannotated Timing and RC Information" in the *Cadence Common Timing Engine (CTE) User Guide* for information on how to buid the characteristics of the design from its environment constraints.

`do_optimize`

`do_uniquely_instantiate`

`do_time_budget`

`write_assertions`

do_extract_model

```
do_extract_model [-lib_name lib_name] [-cell_name cell_name]
     [-clock_slews {clk_slew1 clk_slew2 ...}]
     [-input_slews {input_slew1 input_slew2 ...}]
     [-output_loads {output_load1 output_load2 ...}]
     [-precision integer] [-resolution float] [-tolerance float]
     [-unroll_loops] [-no_insertion_delay]
     [-keep_trigger_arcs] [-force] [-power] [-blackbox] [-blackbox_2d]
     [-verilog_shell_file filename]
     [-verilog_shell_module top_module_name] [-remove_boundard_parasitics]
     [-max_num_slews integer] [-max_num_loads integer] tlf_filename
```

Builds a Timing Library Format (TLF) model (*tlf_filename*) and Liberty (.lib) format for the block specified by the set_top_timing_module and set_current_module commands.

The model is extracted for the given input vectors, PVT conditions, input slews and load capacitance. If any one of these parameters change, then you must extract the model again.

Tip

For model extraction, your design must be properly constrained. For example, no clock or data conflicts can be present. It is a good idea to use the check_timing command and fix all errors prior to model extraction.

For information about the use model, the limitations, and example TLF and .lib output, see Extracting the Timing Model and Writing Extracted Timing Models in Liberty Format in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Note: After binding a model to an instance, use the -check_clocks option with the report_timing command to produce the same timing violations as the original netlist. The reason is that the model extractor moves timing checks associated with sequential elements to timing checks at port pins, usually clock pins. The report_timing command -check_clocks option considers clock paths that end at the reference end of a timing check, as well as the paths that end at the signal end.

Options and Arguments

-blackbox

Produces a TLF file for a black box model. These models can be used by tools that do not understand internal pins. Use black box models primarily in IP characterization, not in a top-down hierarchical design.

For more information, see "Producing Blackbox Models" in *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

`-blackbox_2d`

Transforms three dimensional tables to two dimensional tables by fixing the capacitance at the secondary output to the current value. This is necessary if extracted models are used by tools that do not support three dimensional tables such as First Encounter®.

`-cell_name` *`cell_name`*

Specifies the cell name used in the TLF model.
Default: The name of the module to be extracted.

`-clock_slews {`*`clk_slew1 clk_slew2...`*`}`

Specifies a range of slew values used at all clock pins. The list of values must be enclosed by curly braces. If this option is not specified, appropriate values are computed automatically by the model extractor.

`-force`

Forces, for black box models, the model extraction to continue even though the model extractor has detected assertions on internal pins or hierarchical pins. Such assertions will not be preserved on the model.

For gray box models, the `-force` option, when used with the option `-keep_trigger_arcs`, forces the model extractor to preserve all clock and data signal conflicts that arise, because data signals arrive at pins where clock signals are expected. See the `-keep_trigger_arcs` option.

Note: Do not use the `-force` option for gray box models unless the model extractor issues a warning message regarding too many data signals arriving at control input pins.

 *Tip*

Instead of using the `-force` option, define the clocks properly prior to model extraction such that data and clock conflicts are minimized.

`-input_slews {`*`input_slew1 input_slew2...`*`}`
Specifies a range of slew values used at all input pins. Enclose the list of values with curly braces. If this option is not specified, appropriate values are computed automatically by the model extractor.

`-keep_trigger_arcs`
Models clock and data signal conflicts by preserving trigger arcs (arcs from clock pin to register or latch output pin) that arise because a data signal arrives at a pin where a clock signal is expected. This option only works for gray box models. For black box models, this option is ignored with a warning. This option is useful for characterizing incompletely constrained designs where clock definitions are incomplete or missing. Clocks can be defined after model extraction and this will yield timing results as if the clocks had been defined prior to model extraction. Without the `-keep_trigger_arcs` option, the model extractor expects clock definitions to be complete and removes relevant trigger arcs and timing checks if clock and data signal conflicts occur. This prevents data signals from propagating to clock paths. The drawback of this option is that the model size may increase significantly for a large number of clock and data conflicts. The model extractor will abort if the model size is expected to increase beyond a certain limit. Use the `-force` option to force the model extraction to continue regardless of the increase in model size.

`-lib_name` *`lib_name`*
Specifies the library name used in the TLF model.
Write extracted timing models in the liberty format by specifying the `.lib` file suffix. If you do not specify the file suffix, by default, timing models are written out in the TLF format. See Writing Extracted Timing Models in Liberty Format for details.
Default: The name of the module to be extracted.

`-max_num_loads` *`integer`*
Specifies the maximum number of output load values used for delay arc characterization. A low value of `1` or `2` improves runtime and memory usage but may significantly degrade model accuracy. Use this option only if the model size or model extraction time is critical.
Default: `64`

`-max_num_slews` *`integer`*

Specifies the maximum number of input slews or clock slews used for delay and check arc characterization. Specifying a low value of 1 or 2 improves runtime and memory but may significantly degrade model accuracy. Using the `-max_num_slews` option has a larger impact on runtime and memory than the `-max_num_loads` option. Use this option only if the model size or the model extraction time is critical.
Default: 64

Note: The max slew and max loads options specify only the maximum limits on how many slew and load points are chosen. The tool will look at the gates connected to the input and output ports to come up with the number of points.

`-no_insertion_delay`

Suppresses writing insertion delays in the output TLF file. Insertion delays characterize the embedded clock tree and are used by clock tree generator. The model extractor determines insertion delay by tracing combinational paths back to clock source pins from register clock pins and latch enable pins. Using the `-no_insertion_delay` option saves run time, but results in a model without clock tree characterization.
Default: The insertion delays are written out to TLF.

`-output_loads {`*`output_load1 output_load2...`*`}`

Specifies a range of load values used at all output pins. Enclose the list of values by curly braces. If this option is not specified, appropriate values are computed automatically by the model extractor.

`-power`

Produces a TLF file that includes a power model (see Examples on page 839). Low-Power Synthesis (LPS) provides a constant power number as the model for power analysis in the SE-PKS flow.

This option requires the LPS feature.
Default: No power model is included in the output.

`-precision` *`integer`*

Specifies the floating point precision used for the TLF model
Default: 4

For example, the floating point precision for 3.0089 can be changed as follows:

Precision	Data Value in TLF
4	3.0089
3	3.009
2	3.01
1	3.0

`-remove_boundary_parasitics`

Changes output port capacitances by removing the parasitic resistance (R_s and R_pi) at output port boundaries and asserting C1 and C2 as wire caps. A new set of external loads are then applied to characterize delays at output ports. The model extractor takes into account the possible change in external loading at the expense of losing RC net delay at boundary nets. See Figure 2-6: Time-Varying Thevenin-Equivalent Gate Delay Model in *Common Timing Engine (CTE) User Guide*.

Useful when reduced parasitics data (RSPF) is present. Since parasitics are reduced for only one set of output port capacitances, the extracted model is valid only for the capacitance values for which the reduction was done.

Note: The boundary parasitics will not be restored after model extraction. This option is not needed for detailed parasitics data (DSPF), since a reduction is performed automatically for each new external output port capacitance.

`-resolution` *`float`*

Specifies the magnitude of delay difference in user units that can be ignored by the extractor.
Default: `0.001`

`tlf_filename`

Specifies the name of the TLF output file.

Tip

Avoid problems with rebind by using the `-lib_name` and `-cell_name` options instead of using the default names.

`-tolerance` *`float`*

Specifies the accuracy level of extractor in percent. For example, 5.0 represents a plus or a minus 5.0 percent allowable error in the extracted model. The higher the tolerance, the faster the extraction time.
Default: 1.0

`-unroll_loops`

Unrolls, for gray box models, self-loop setup/hold timing checks or direct clock-to-clock constraints by introducing dummy internal pins and zero-delay arcs. This is useful for timing analysis tools that do not understand self-loop constraints. For CTE, use the `report_timing -check_clocks` command to see violations associated with self-loop checks or direct clock-to-clock constraints.

This option is ignored if used with the `-blackbox` option because black box models have no internal pins.

`-verilog_shell_file` *`filename`*

Instructs the model extractor to generate a Verilog shell file that instantiates the TLF model. This option is useful for validating the model. Use the `-verilog_shell_module` option to specify the name of the top module in this Verilog file.

`-verilog_shell_module` *`top_module_name`*

Specifies the top module name of the Verilog shell file.
Default: `test_shell`

Examples

- The following command builds a TLF model (*tlf_filename*) for the block:

```
do_extract_model min_max.tlf
```

- The following commands show a flow for generating the power model for instance `foo`, module `foo_m`:

```
read_tlf lib.tlf
read_verilog netlist.v
```

```
do_build_generic
set_top_timing_module top
set_current_module top
source assertions
read_tcf netlist.tcf
do_derive_context foo
set_top_timing_module foo_m
set_current_module foo_m
do_extract_model -power -lib_name new.lib -cell_name new_cell
set_top_timing_module top
set_current_module top
read_tlf model.tlf
```

- The following shows what the resulting TLF looks like (model contents deleted):

```
Cell(new_cell
  Timing_Model(Model0
    ...
  )
  Timing_Model(Model1
    ...
  )
  Cell_SPower(0.000732)
)
```

- The following command creates the timing model for the `test_model` cell belonging to the `test_lib` library, and writes the model into the `test.lib` liberty file:

```
do_extract_model -cell_name test_model -library_name test_lib test.lib
```

  If you do not specify the file suffix, by default, timing models are written out in the TLF format.

Related Information

`do_derive_context`

`do_pull`

`do_push`

`do_rebind`

`do_time_budget`

`read_tlf`

`set_current_module`

`set_top_timing_module`

do_signalstorm

```
do_signalstorm { -process process_name | -setup setupfile}
     { -dspf dspffile | -spef speffile | -parasitics parasiticsfile }
     [ -lib ecsm_library] [-include filelist] [-user_inlude_only]
     [-max_slew max_slew_limit] [-outprefix outprefix]
```

Lets you run SignalStorm™ for accurate delay calculation on gate and interconnect segments. The `do_signalstorm` command takes your input and creates the default constraint and setup files necessary to run the SignalStorm tool.

Note: The `do_signalstorm` command is a utility that runs the SignalStorm software with the necessary input files to generate an SDF file. This file is back-annotated to the BG/PKS run after it is generated. For information on how to use the utility or the input files needed to generate the SDF file, use the `do_signalstorm -help` command after you load a design. Make sure the SignalStorm hierarchy and license file are present, the SIMPLEX ENV variables are set, and that `signalstorm` is part of the PATH ENV variable as follows:

```
SIMPLEX_HOME
SIMPLEX_VERSION
set path = (<path to bin directory of signalstorm> $path)
```

Options and Arguments

`-dspf` *`dspffile`* `|` `-spef` *`speffile`* `|` `-parasitics` *`parasiticsfile`*

Specifies only one parasitic file, which covers the whole design. The parasitic file can either be in dspf format, spef format, or a compressed .gz file of either format. The `do_signalstorm` command errors out if you specify more than one parasitics file.

Use the `-parasitics` option if you prefer to let the `do_signalstorm` command detect the parasitics file and decide whether it is a spef or a dspf file.

`-include` *`filelist`*

Specifies a list of files to be included in the SignalStorm run. Unless the `-user_include_only` option is used, the specified files are included, in addition to the default constraint information that the `do_signalstorm` command generates. The default constraint information includes the following:

– Load of all output ports in the design
– Slew of all input ports in the design stretched to a 0-100 percent threshold
– Timing constraints that effect propagation of slews

This option is only for advanced users.

`-lib` *`ecsm_library`*

Specifies the path to the ECSM library. This is a required option. For information on how to create a ECSM library, see the “ECSM Library Database” section in the *SignalStorm Manual*.

`-max_slew` *`max_slew_limit`*

Writes out a `outprefix.slew` file that contains nets having a slew greater than the specified *`max_slew_limit`*.

`-outprefix` *`outprefix`*

Specifies the outprefix file name. If you do not specify this option, the default outprefix is derived from the top module name and the processed module name ID. There are four main files that are generated as a result of running the `do_signalstorm` command:

`outprefix.cmd`: The command file that the `do_signalstorm` command generates and uses to run the SignalStorm tool.

`outprefix.log`: The log file that contains error and warning messages displayed during the SignalStorm run.

`outprefix.sdf`: The SDF file that SignalStorm generates. This file is automatically back-annotated.

`outprefix.slew`: The file that lists any part of the design, which exceeds the slew time specified in the `-max_slew` option.

`-process` *`process_name`*

Specifies the process name that has been characterized and saved in the ECSM library for computing delay, which refers to a specific process, voltage, and temperature parameter for timing analysis. For information on how to create an ECSM library from a .lib library, see “Generating the Library Database from Synopsys `.lib` Files” in the *SignalStorm Manual*.

Make sure you specify the process name that is related to the timing analysis setting used in other commands, such as the `timing_analysis_type` and the `pvt_late_path` globals,

and the set_operating_conditions command where you specify the process, voltage, and temperature values.

For example, if in CTE you set the following commands:

```
set_global pvt_late_path max
set_global pvt_early_path min
set_global timing_analysis_type min-max
set_operating_condition -pvt min fast
set_operating_condition -pvt max slow
```

When running the do_signalstorm command as follows:

```
do_signalstorm -process min_process:max_process
-dspf test.dspf -lib /lib/dtmf_chip.ipdb
```

The process name that you specify for min_process, should be the process name that you use to characterize the library dtmf_chip.ipdb with a fast operating condition. Consequently, the max_process setting should be the process name that you use to characterize the library dtmf_chip.ipdb with a slow operating condition.

Various process names at which the ECSM library has been characterized are found in the library directory: *cell*.design/analysis/process.

The do_signalstorm command errors out if the process name you specify cannot be found in the ECSM library. When you want to specify more than one process, separate each process with a colon. For example, -process min:typ:max.

`-setup` *setupfile*

Looks at the setupfile and uses the process name defined in the set process setup command, which consequently ignores the -process option if specified. If there are multiple set process commands specified in the setupfile, the do_signalstorm command chooses the last one.

`-user_include_only`

Prevents the do_signalstorm command from creating the default constraints. See the -include option description for an explanation of what the default constraints contain. If this option is specified, the do_signalstorm command uses the list of

include files in the order you specify. This option is for advanced users.

Examples

- The following command uses the `technology.ipdb` library at a single pvt corner, which the process name `max` refers to. The design and its connectivity is described in the spef file `chip.spef.gz`:

```
do_signalstorm -process "max" -spef "chip.spef.gz" -lib
"technology.ipdb"
```

- The following command includes the user include file. Since the `-user_include_only` option is specified, the `do_signalstorm` command will not generate the default constraints. The user include file captures information, such as slew and load at input and output ports in the design, as well as timing constraints that may effect the slew propagation.

```
do_signalstorm -process "max" -spef "chip.spef.gz" -lib "technology.ipdb"
-include "signalstorm.constraints"  -user_include_only
```

- The following command shows how to create your own setup file. The setup file must contain the “set process” setup command, which the `do_signalstorm` command uses to decide which process name the delay needs to be computed.

```
do_signalstorm -setup "top.setup" -spef "chip.spef.gz" -lib "technology.ipdb"
```

- The following command shows how to run the `do_signalstorm` command at multiple process names. The description of the `-process` option, shown in the above example, explains how the process names you choose should correspond to the settings used in CTE.

```
do_signalstorm -process "min:max" -spef "chip.spef.gz" -lib "technology.ipdb"
```

do_time_budget

`do_time_budget [-top` *`list_of_top_instances`*`] | [`*`list_of_instances`*`]`

Generates realistic time estimates for the modules specified in the instance list.Time budgeting is the process of splitting the combinational delay requirements across module boundaries to allocate slack across all modules in a fair manner. The segments of paths entirely contained in the instance list are considered to be changeable, while all other paths are considered to be fixed.

Tip

Use either the `do_time_budget` or the `do_derive_context` command in CTE. For PKS, use the `do_push` command instead. Budgeted constraints cannot be used in a third party tool.

Note: The following two paragraphs apply to PKS users only.

The `do_time_budget` command comes up with assertions to use for a block, assuming the block is run using wireloads. The PKS `do_push` command, on the other hand, works around soft blocks.The PKS `do_push` command does the budgeting assuming the block timing is completed using the physical information that is pushed (by default). The `do_time_budget` and `do_derive_context` commands do not push physical information to the budgeted block and therefore, will not capture your intent in a PKS session.

The budgeting results are slightly different between the `do_time_budget` (or `do_derive_context`) and the `do_push` commands. This is because of a delay correction that the `do_push` command does on the boundary arrival and required times based on the capacitance and delay calculated in the lower context. The correction is done so that design constraints and physical design characteristics match to the top level context.

The `do_time_budget` command does the following:

- Writes assertions on the instances in the database. Use the `write_assertions` or the `write_adb` command to save the result of time budgeting.
- Finds the worst constraints for a module from all those available (if the module is used in many contexts under the top timing module).
- Accepts the `dont_modify` flags on instances and modules in the computation of constraints.

The reference of the hierarchical instances starts from the current module as specified by the `set_current_module` command. The current module must be unique in the scope of the

top timing module as set by the `set_top_timing_module` command. See Setting the Current Module in the *Common Timing Engine (CTE) User Guide* for more information.

Note: The time budgeting algorithm handles both positive and negative slacks.

A “changeable” path or block is one that can be modified during synthesis to meet timing constraints, whereas a “fixed” one cannot be modified. Normally, you specify as fixed, the hard macros or other blocks that you do not want the optimizer to touch. There are two ways to specify such fixed blocks:

- Use the `dont_modify` switch for such instances.
- Use the `-top` option only for blocks that can be modified.

If you issue the command without any options, for example, `do_time_budget`, then all instances (except those having the `dont_modify` switch) are assumed to be changeable. BuildGates Synthesis creates budgets for all the instances.

If you want to create the budgets only for specific modules, use the following command:

```
do_time_budget {inst1 inst2 ...}
```

Note: The above command still assumes that all the instances except those marked `dont_modify` are changeable.

If you want only a specific instance (`inst1`) to be changeable and all others fixed, use the following command:

```
do_time_budget -top inst1
```

Options and Arguments

`list_of_instances`
: Specifies a list of instances for time budgeting. The instances in the *`instance_list`* must all be hierarchical entities. If the *`instance_list`* is not specified, then the default is to generate budgets for all instances within the current module. The time budget assumes that only the delay arcs (paths) entirely within the specified instances can be optimized. The others are considered to be fixed.

  Depending on the number of instances in the current module, omitting the instance list can result in long run times.

`-top` *`list_of_top_instances`*
: Creates a top timing module from which to start the synthesis.

The listed instances are considered part of the top context. Only those sub-designs that exist under the list of top instances are synthesized. The time budget assumes that only the delay arcs (paths) within and between the specified top instances can be optimized. The others are considered to be fixed.
Default: Assumes that everything under the top timing module can be changed. Use the `dont_modify` option for instances and modules to overrides this.

Examples

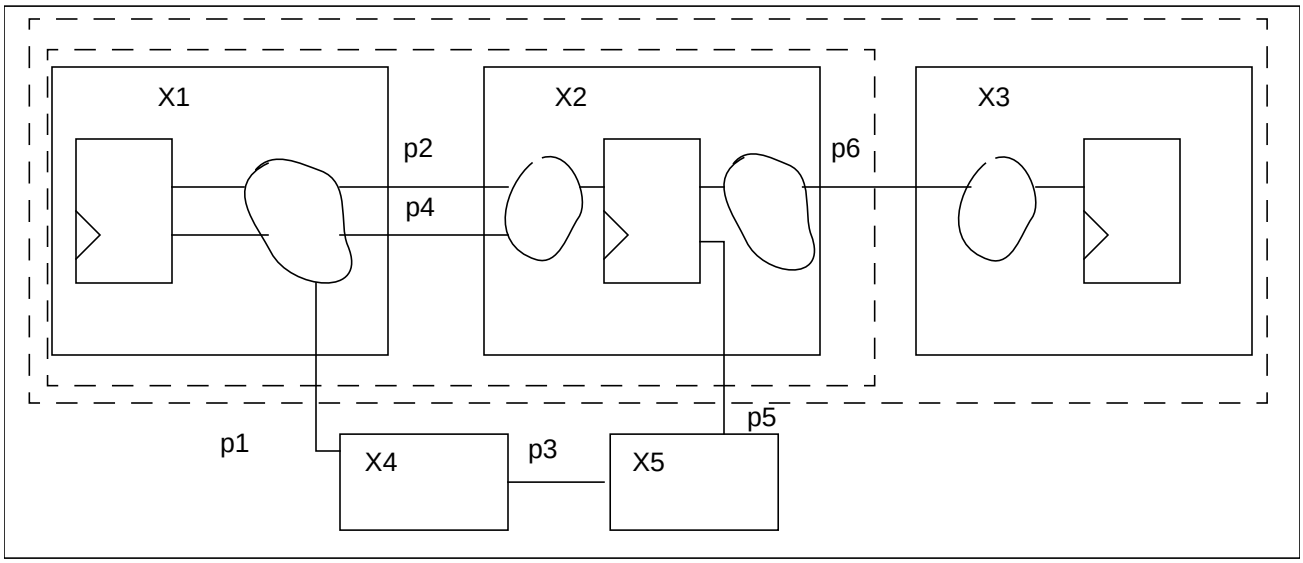
- The following command performs time budgeting on the instances `X1`, `X2`, and `X3` for the circuit shown in Figure 7-1, assuming that all delay arcs contained within these instances are changeable and all others are fixed. The paths `p2`, `p4`, and `p6` are changeable and the others are treated as fixed. This is because the top contains only `p2`, `p4`, and `p6`, while `p1`, `p3`, `p5` are outside the boundaries of the top context.

  ```
  do_time_budget -top {X1 X2 X3}
  ```

- The following command assumes only paths in and between `X1` and `X2` instances to be changeable and `X3` to be fixed. As a result, the time budgeting algorithm derives a new set of constraints since only the paths `p2` and `p4` are changeable while the others are fixed.

  ```
  do_time_budget -top {X1 X2}
  ```

Figure 7-1 do_time_budget Example



Related Information

do_derive_context

do_push

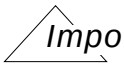
set_current_module

set_dont_modify

set_top_timing_module

do_xform_timing_correction

```
do_xform_timing_correction [-footprint] [-resize] [-upsize_only] [-buffer]
     [-clone] [-dont_modify_children]
     [-design_rule_only | {-no_design_rule | -dont_fix_design_rule}]
     [-min_hold | -fix_hold] [-no_pinswap | -pinswap_only] [-fix_clock_net]
     [-max_area float] [-quick] [-incremental][-critical_ratio float]
     [-critical_offset float] [-dont_reclaim_area | -minimize_area]
     [-change_limit integer] [-change_file filename]
```



Important

This command is obsolete and will be removed in the next full release of the software. Use the `do_optimize -effort low` and `do_xform_fix_hold` commands for comparable results.

get_capacitance_unit

```
get_capacitance_unit
```

Returns the session capacitance unit in picoFarads.

Example

The following command returns the session capacitance unit:

```
get_capacitance_unit
1.0
```

Related Information

`set_capacitance_unit`

`reset_capacitance_unit`

get_cell_drive

```
get_cell_drive [-library library_name] [-cell cell_name ]
     [-from_pin from_pin_name] [-from_pin_rise] [-from_pin_fall]
     [-rise] [-fall] [-early | -late] [-pin pin_name]
```

Returns the resistance value at the specified output port of a cell in the library. For a library having nonlinear delay models, the returned value is the resistance as seen at the specified output pin if the output pin were to drive the input pin having the worst capacitance of the same cell. (In other words, the resistance is linear at the point as if the cell is driving itself.)

The returned value is a real (floating point) number. The units of resistance are the same as the units used in the technology library.
Default: The worst resistance found in rising, falling, from-pin rising, from-pin falling, early and late modes is returned.

Options and Arguments

`-cell` *cell_name*
: Specifies the name of the cell whose output port is specified by the `-pin` option.

`-early | -late`
: Returns the resistance corresponding to early (data hold check) or late (data setup check) modes only.

`-fall`
: Returns the resistance corresponding to falling edge only.

`-from_pin_fall`
: Returns the resistance corresponding to the falling signal at the from pin only.

`-from_pin` *from_pin_name*
: Specifies the input port name as the starting point of the path for which the resistance is returned.

`-from_pin_rise`
: Returns the resistance corresponding to the rising signal at the from pin only.

`-library` *library_name*
: Specifies the name of the library where the cell can be found.
Default: The target library.

`-pin` *`pin_name`*

Specifies the name of the output port whose resistance is returned. If there is more than one path terminating at the *`pin_name`*, the largest resistance seen among all paths is returned.

`-rise`

Returns the resistance corresponding to rising edge only.

Examples

```
get_cell_drive -cell BUFA -from_pin A -pin Z
1.920189

get_cell_drive -cell BUFA -from_pin A -from_pin_fall -pin Z
1.368688
```

Related Information

`get_cell_pin_load`

`set_global target_technology`

get_cell_pin_load

`get_cell_pin_load [-library` *`library_name`*`] [-cell` *`cell_name`*`] [-pin` *`pin_name`*`]`

Returns the value of the capacitive load at the specified pin of a cell in the library. The returned value is a real (floating point) number. The units of the load are the same as the units used in the library.

Options and Arguments

`-cell` *`cell_name`*

Specifies the name of the cell whose pin is specified by the `-pin` option.

`-library` *`library_name`*

Specifies the name of the library where the cell can be found.
Default: The target library is searched for the cell.

`-pin` *`pin_name`*

Specifies the name of the pin whose capacitive load is returned.

Example

- The following command returns the value of the capacitive load of pin `z` of cell `IV` in the library. See the `set_cell_pin_load` command for the related library cell description used for these examples.

  ```
  get_cell_pin_load -cell IV -pin z
  ```

- The following command gets the capacitance values for pin A:

  ```
  get_cell_pin_load -library cell_load_example -cell AN2 -pin A
  0.042000
  ```

- The following command gets the capacitance values for pin B:

  ```
  get_cell_pin_load -library cell_load_example -cell AN2 -pin B
  0.041000
  ```

  An error message is returned if you are getting values for a pin that is not defined for a cell.

Related Information

`get_cell_drive`

set_cell_pin_load

set_global target_technology

get_clock

```
get_clock [clock_name] [-period | -waveform | -lead | -trail]
```

Returns the information of clocks defined in the database through the `set_clock` or the `set_generated_clock` command.

Tip

If you want information about a generated clock, you must run CTE first. This is because the generated clock waveforms are created by timing analysis. Without timing analysis, the system cannot report the name of the generated clock.

See Specifying Clock Information in the *Common Timing Engine (CTE User Guide*.

Options and Arguments

clock_name
: Returns information for the specified clock.
Default: A list of all currently defined clocks is returned and all other command line options are ignored.

-lead
: Returns the leading edge of the specified clock.

-period
: Returns the period of the specified clock.

-trail
: Returns the trailing edge of the specified clock.

-waveform
: Returns the waveform of the specified clock in the form of {*lead*, *trail*}.

Examples

- The following commands define the clock named CLOCK:

```
set_clock CLK -period 10.0 -wave {0 5}
set_clock_root -clock CLK pinX
```

- The following command gets clock-related information for CLK:

```
get_clock -period CLK
```

```
10.000000
```

- The following commands get the name of all the defined clocks:

```
get_clock
CLK
```

- The following commands define a generated clock:

```
set_generated_clock -name my_clock -from pinX -divide_by 2 pinY
get_clock
CLK
```

- The following commands report only the ideal clock because you have not performed timing analysis yet. Use the `report_timing` command followed by the `get_clock` command:

```
report_timing
# report deleted
get_clock
CLK my_clock
get_clock -period my_clock
20.000000
```

- Using the `report_clocks` command with the `-generated` option runs analysis and reports the same information that is available with the `get_clock` command except in a different format. See Examples for the `report_clocks` command report format.

Related Information

`get_clock_source`

`report_clocks`

`set_clock`

`set_generated_clock`

get_clock_propagation

```
get_clock_propagation
```

Returns the value of the clock propagation mode, `ideal` or `propagated`, associated with the current top timing module.

Note: Because this command returns the clock propagation mode associated with a design, load a design before using this command.

If the `get_clock_propagation` command returns an `ideal` clock propagation mode, use the `get_propagated_clock` to determine the status of individual clocks.

See Setting the Clock Propagation Mode for Analysis in the *Common Timing Engine (CTE) User Guide* for more information.

Example

The following command returns the value of the clock propagation mode:

```
get_clock_propagation
ideal
```

Related Information

`get_propagated_clock`

`reset_propagated_clock`

`set_clock_propagation`

`set_propagated_clock`

get_clock_source

```
get_clock_source [-port_only] clock_name
```

Returns the list of clock names (sources) for the defined clock using the *clock_name* argument in the database. Also returns pins where a generated clock has been asserted. Specify the clock sources through the `set_clock_root` and `set_generated_clock` commands.

Options and Arguments

clock_name

Returns the clock sources (input ports or instance output pins) set on the named clock. If this option is omitted, all clock sources specified in the current database are returned.

`-port_only`

Returns only the clock sources that are input ports.

Examples

- The following command defines the clock `CLK`:

  ```
  set_clock CLK -period 4 -waveform {0 2}
  ```

- The following command defines the clock source `clk_in`:

  ```
  set_clock_root -clock CLK clk_in
  ```

- The following command gets the clock source for clock `CLK`:

  ```
  get_clock_source CLK clk_in
  ```

Related Information

`get_clock`

`get_derived_clock`

`set_clock`

`set_clock_root`

get_constant_for_timing

```
get_constant_for_timing [-bidi_input | -bidi_output] pin_name
```

Queries the design database for the state of the specified pin or the state propagated through the combinational logic cone to that pin. The returned value is `0` for logic 0, `1` for logic 1, or `X` if no constant has been set.

Note: Constants on hierarchical ports are not computed.

See Setting Constant Values for Timing in the *Common Timing Engine (CTE) User Guide* for more information.

Tip

The `get_constant_for_timing` command returns the state for one pin only. If you want to know the state on all pins that have a constant set, use the `-type constant` option with the `report_port` command. For example:
`report_port -type constant.`
Use the `-tcl_list` option to process the list in Tcl.

Options and Arguments

`-bidi_input | -bidi_output`
Returns the constant that is set on the input or output part of bidirectional pins.
Default: Returns the constant on the input part.

`pin_name`
Specifies the name of the single pin to query for pin state. The pin must be an instance pin or primary IO port.

Examples

- The following command sets the value of the specified pin(s) to either a 1 or 0 for use by the timing engine:

  ```
  set_constant_for_timing 0 J_block/zbuf0/A
  ```

- The following command queries the design database for the state of the specified pin or the state propagated through the combinational logic cone to that pin:

  ```
  get_constant_for_timing J_block/zbuf0/A
  ```

Related Information

report_ports

reset_constant_for_timing

set_constant_for_timing

get_dcl_calculation_mode

```
get_dcl_calculation_mode
```

Returns the current DCL calculation mode.

Calculation modes defined in the DCL library allow the vendor library developer to model three process conditions (`best_case`, `nominal_case`, and `worst_case`) within one library. Choose the mode with the `set_dcl_calculation_mode` command.

Note: Temperature, process multiplier, and voltage are set independently of the calculation mode using the `set_operating_parameter` command.

Example

The following command returns the current DCL calculation mode:

```
get_dcl_calculation_mode
worst_case
```

Related Information

`get_operating_parameter`

`load_dcl_rule`

`set_dcl_calculation_mode`

`set_operating_parameter`

get_dcl_functional_mode

`get_dcl_functional_mode` *`instance`*

Returns the current DCL functional mode on the specified hierarchical instance. The value returned is a Tcl list where the current functional mode group for the instance is the first element in the list, and the current functional mode value is the second element in the list.

Functional modes defined in a DCL library let the library developer model the timing arcs and timing checks for a technology cell in a variety of ways. For instance, a scan flip-flop could have functional modes defined for normal flip-flop behavior and for scan behavior where, for each mode, different timing arcs and checks are defined.

See Using IEEE 1481 Delay and Power Calculation System (DPCS) Libraries in the *Common Timing Engine (CTE) User Guide* for more information.

Options and Arguments

`instance`

Specifies the hierarchical instance for which the DCL functional mode needs to be obtained.

Example

The following command shows that the current functional mode group for instance `A1/B3/C2` is `latch_type` and the value is `transparent`:

```
get_dcl_functional_mode A1/B3/C2
{latch_type transparent}
```

Related Information

`get_dcl_functional_mode_array`

`load_dcl_rule`

`set_dcl_functional_mode`

get_dcl_functional_mode_array

```
get_dcl_functional_mode_array [-cell cellname]
```

Returns the DCL functional mode array on the specified library cell. This array lists all of the functional modes available for the library cell by group. The array is returned as a hierarchical Tcl list.

Functional modes defined in a DCL library let the vendor library developer model the timing arcs and timing checks for a technology cell in a variety of ways. For instance, a scan flip-flop could have functional modes defined for normal flip-flop behavior and for scan behavior where, for each mode, different timing arcs and checks are defined.

Options and Arguments

`-cell` *`cellname`*

Specifies the library cell for which the DCL functional mode array needs to be obtained.

Example

The following command returns the DCL functional mode array on the specified library cell:

```
get_dcl_functional_mode_array -cell ram_8_8
{
{rw { read write}} {latch_type {latching transparent}}
}
```

Related Information

`load_dcl_rule`

`set_dcl_functional_mode`

get_dcl_level

```
get_dcl_level [-perfLevel]
```

Returns the current performance level used in DCL based delay and slew calculations.

DCL supports two performance levels for delay calculation. One performance level targets faster run time with less accurate timing, while the other targets more accurate timing with slower run time.

Refer to the DCM library vendor documentation for information about the characteristics of each performance level in their library and whether or not they are both supported. Use the faster performance in terms of run time for initial synthesis. For final timing correction, use the more accurate level.

Options and Arguments

`-perfLevel`

Returns the current performance level.

Note: The DCL standard contains other types of levels which, if supported in the future, are accessible via other arguments. These other types include `temperatureScope`, `voltageScope`, `functionalModeScope`, and `wireloadModelScope`.

Example

The following command returns the current performance level used in DCL based delay and slew calculations:

```
get_dcl_level -perfLevel
1
```

Related Information

`load_dcl_rule`

`set_dcl_level`

get_derived_clock

`get_derived_clock [-full_path_name] [`*`instance_path_or_id`*`]`

Reports the derived clock sources for all sequential instances or for the specified instance.

Note: This command is not related to the `set_derived_clock` command.

The derived clocks are found by traversing backwards from the clock pin of sequential instances until reaching an output of a non-single-input-single-output combinational instance, or a sequential output (divided clock, for example), or a primary input port.

Note: If the `clock`: `true` attribute is not present in the library, the input pin on a RAM/ROM with setup/hold timing check is also treated as clock pin.

Options and Arguments

`-full_path_name`
: Returns a list of full path names for derived clocks. If this option is not specified, pin object identifiers are returned.

`instance_path_or_id`
: Specifies a sequential instance by path or object identifier. If no *`instance_path_or_id`* is specified, derived clocks for all sequential instances (`ff` and `latch`) are reported.

Example

The following command reports the derived clock sources for all sequential instances or for the specified instance:

```
get_derived_clock -full_path_name
TCLK3
COREP_P2/BEP_P2/I14P/Q
COREP_P2/BEP_P2/I25P/Q
CS
```

Related Information

`set_clock_info_change`

get_drive_pin

`get_drive_pin [-hierarchical] [-full_path_name] net_path_or_id`

Returns a list of drive pins that drive a given net within the current module.

Note: For multiple driver situations, the `get_drive_pin` command returns a list of pin identifiers.

Options and Arguments

`-full_path_name`
: Returns the full path name for the drive pin, rather than the pin object identifier.
Default: Object identifiers are returned.

`-hierarchical`
: Enables traversal across a hierarchy if the given net crosses a hierarchy boundary.

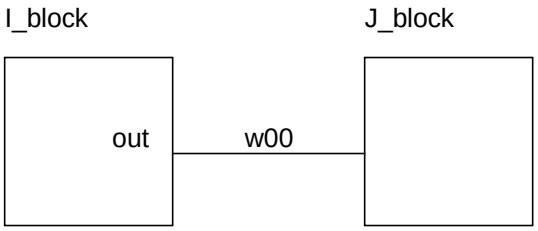
`net_path_or_id`
: Specifies the net for which you want to know the driver pin. Specify the net by the full path name or the object ID.

Example

The following command shows how to get names for the circuit shown in Figure 7-2:

```
get_names [get_drive_pin [find -net w00]]
out
get_drive_pin -full_path_name w00
I_block/out
```

Figure 7-2 get_drive_pin Example



Related Information

`get_load_pin`

get_fanin

```
get_fanin [-level integer] [-tristate] [-sequential] [-hierarchical]
     [-full_path_name] list_of_pin_path_or_id
```

Returns a Tcl list of all begin points (output pins and input ports) in the fanin cone.

Use the `-tristate` and `-sequential` options for finer control of the gate type to be included in the fanin cone.

The command can take a hierarchical path name as well as a pin object identifier. Specifying a hierarchical name is useful when a module is not uniquely instantiated.

Options and Arguments

`-full_path_name`
: Returns the hierarchical path name for the starting points, starting from the current module.
Default: Pin object identifiers are returned.

`-hierarchical`
: Lets you search cross hierarchy boundaries.
Default: Search stops at input ports and returns the port as the starting point.

`-level` *`integer`*
: Limits the search to the number of logic levels specified.

`list_of_pin_path_or_id`
: Starts the fanin cone search from the specified Tcl list of pins or ports specified by path names or object identifiers.

`-sequential`
: Includes sequential cells in the fanin path.
Default: Sequential cells are excluded.

`-tristate`
: Includes tristate cells in the fanin path.
Default: Tristates are excluded.

Examples

- The following command returns a Tcl list of all begin points (output pins and input ports) in the fanin cone:

```
get_fanin -hierarchical -full_path_name out*
J_block/C_reg/Q J_block/D_reg/Q
```

- The following Tcl script iterates over all the begin points of the fanin cone of net `req`:

```
foreach i [get_fanin [find -net req]] {
puts "Found fanin [get_names $i]"
}
```

Related Information

`get_fanout`

`do_extract_fanin`

get_fanout

```
get_fanout [-level integer] [-tristate] [-sequential] [-hierarchical]
     [-full_path_name] list_of_pin_path_or_id
```

Returns a Tcl list of all endpoints (input pins and output ports) in the fanout cone.

The `-tristate` and `-sequential` options give you finer control of the gate type to be included in the fanout cone.

The command can take a hierarchical path name as well as pin object identifier. Specifying a hierarchical name is useful when a module is not uniquely instantiated.

Options and Arguments

`-full_path_name`
: Returns the hierarchical path name for the starting points, starting from the current module.
Default: The pin object identifiers are returned.

`-hierarchical`
: Searches cross hierarchy boundaries.
Default: Search stops at output ports and returns the port as the ending point.

`-level` *`integer`*
: Limits the search to the number of logic levels specified.

`list_of_pin_path_or_id`
: Starts the fanout cone search from the specified Tcl list of pins or ports specified by path names or object identifiers.

`-sequential`
: Includes sequential cells in the fanout path.
Default: Sequential cells are excluded.

`-tristate`
: Includes tristate cells in the fanout path.
Default: Tristates are excluded.

Examples

- The following command returns the names of endpoints of the fanout cone of any ports whose names begin with characters `in`:

```
get_fanout -full_path_name [find -port in*]
```

- The following Tcl script Iterates over all the endpoints of the fanout cone of net `req`:

```
foreach i [get_fanout [find -net req]] {
    puts "Found fanout[get_names $i]"
}
```

Related Information

`get_fanin`

`do_extract_fanin`

`do_extract_fanout`

get_flow_compatible_mode

`get_flow_compatible_mode`

Returns the constraint interpretation mode, CTE or Synopsys PrimeTime style, for timing analysis.

Related Information

`set_flow_compatible_mode`

get_load_pin

`get_load_pin [-hierarchical] [-full_path_name] net_path_or_id`

Returns a list of load (sink) pins for the given net within the current module and in the sub-modules.

Note: For multiple load situations the `get_load_pin` command returns a list of pin identifiers.

Options and Arguments

`-full_path_name`
: Returns a list of full path names of load (sink) pins.
Default: The object identifiers are returned.

`-hierarchical`
: Traverses the design hierarchy when the given net connects to an output port.

`net_path_or_id`
: Specifies the net for which you want to know the load pin. Specify the net using the full path name or the object ID.

Examples

- The following command returns a list of load (sink) pins for the given net within the current module and in the sub-modules as shown in Figure 7-3.

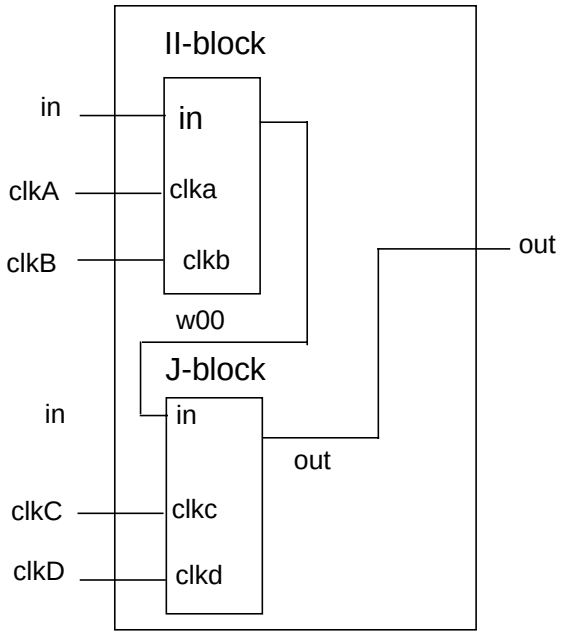
  ```
  get_load_pin -full_path_name [find -net out]
  out
  ```

  If you do not use the `-hierarchical` option, only the sink pins of the net in the current module are listed as IDs.

  Use the `-full_path_name` option to list the ID names.

  Use the `-hierarchical` option to list the sink pins in the sub-modules and the sink pins of the instances this net connects to.

Figure 7-3 Current Module and Sub-Modules Example




```
module I_block (out, in, clka, clkb);
input clka, clkb, in;
output out;
FD1 A_reg(.Q(w01), .CP(clka), .D(in));
FD1 B_reg(.Q(w02), .CP(clkb), .D(in));
BUF8A  abuf (.Z(PA), .A(w01));
BUF8A  bbuf (.Z(PB), .A(w02));
AN2 u000 (.Z(out), .A(PA), .B(PB));
endmodule

module J_block (out, in, clkc, clkd);
input clkc, clkd, in;
output out;
IV iC(.Z(inC), .A(in));
BUF8A  cbuf (.Z(PC), .A(inC));
IV iD(.Z(inD), .A(in));
FD1 A_reg(.Q(w03), .CP(clkc), .D(PC));
FD1 B_reg(.Q(w04), .CP(clkd), .D(PD));
BUF8A  dbuf0 (.Z(w041), .A(inD));
BUF8A  dbuf2 (.Z(w042), .A(w041));
BUF8A  dbuf (.Z(PD), .A(w042));
BUF8A  zbuf0 (.Z(w03prime), .A(w03));
B BUF8A  zbuf1 (.Z(Z), .A(w03prime));
AN2 u000 (.Z(out), .A(Z), .B(w04));
endmodule
```

- The following commands show how to get the load pin for the current module and sub-modules shown in Figure 7-3:
  - `get_load_pin -hier in`

    ```
    74006 73878
    ```
  - `get_load_pin -full in`

    ```
    I_block/in
    ```
  - `get_load_pin -hier -full in`

    ```
    I_block/B_reg/D I_block/A_reg/D
    ```
  - `get_load_pin -hier w00`

    ```
    74614 74486
    ```
  - `get_load_pin -full w00`

    ```
    J_block/in
    ```
  - `get_load_pin -hier -full w00`

    ```
    J_block/iD/A J_block/iC/A
    ```

- `get_load_pin -full J_block/w041`
  `J_block/dbuf2/A`
- `get_load_pin -full J_block/w042`
  `J_block/dbuf/A`

Related Information

`get_drive_pin`

`get_fanout`

`get_fanin`

get_module_worst_slack

```
get_module_worst_slack [-clknet_too] [-early | -late]
```

Returns the worst slack time for the module and all of its children. The module that is set as the current module is searched for the worst slack time on its paths.

If there are negative slack paths in the module, the returned value is the most negative value. If there are no negative slack paths in the module, the returned value is the least positive slack value.

Options and Arguments

`-clknet_too`

Includes clock nets while searching for the worst slack time on all paths in the module. By default, slack time on clock nets is not considered. For clock pins the late arrival is determined by the hold time check and the early arrival by the setup time check.

Note: If the global variable `clock_gating_regardless_of_downstream_logic` is set to `true`, the results from using the `get_module_worst_slack` command without the `-clknet_too` option may not match the slack numbers from using the `report_timing` command without the `-check_clocks` option.

`-early | -late`

Indicates whether the worst slack time should be considered on early arrivals (data hold) or late arrivals (data setup) of signals. If neither option is specified, the default is to obtain worst slack time for late arrivals on signals.

Example

```
get_module_worst_slack -early
```

Related Information

`set_current_module`

get_operating_conditions

```
get_operating_conditions [-pvt {min | typ | max}]
```

Returns the operating conditions asserted by the `set_operating_condition` command.

Examples

The following commands query a single operating condition previously set using the `set_operating_condition` command:

```
get_operating_conditions -pvt min
fast_0_3.60
get_operating_conditions -pvt typ
slow
```

Related Information

`get_operating_parameter`

`read_alf`

`read_tlf`

`report_library`

`reset_operating_condition`

`set_operating_conditions`

`set_operating_parameter`

get_operating_parameter

```
get_operating_parameter {-process | -voltage | -temperature}
     [-pvt {min | typ | max}]
```

Queries the database for the process, voltage, or temperature used for delay calculation.

Specify only one of the parameter options per command.

Options and Arguments

`-process`
: Returns the process multiplier.

`-pvt {min | typ | max}`
: Returns the operating parameter for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The `typ` value is returned.

`-temperature`
: Returns the operating temperature.

`-voltage`
: Returns the operating voltage.

Examples

- The following command queries the database for the voltage used for delay calculation:

```
get_operating_parameter -voltage -pvt {min max}
3.100000 3.500000
```

- The following command queries the database for the process used for delay calculation:

```
get_operating_parameter -process
1.000000
```

- The following command queries the database for the temperature used for delay calculation:

```
get_operating_parameter -temperature
25.000000
```

Related Information

`load_dcl_rule`

`set_operating_parameter`

`set_operating_conditions`

get_operating_voltage

```
get_operating_voltage [-pvt {min | typ | max}] leaf_level_instance
```

Returns the current operating voltage on the specified hierarchical instance.

Options and Arguments

leaf_level_instance
: Specifies the leaf level instance for which the operating voltage needs to be obtained. Specify only one instance, not a list.

-pvt {min | typ | max}
: Returns the value for the environment corner of interest. Specify only one PVT corner per command.
Default: max

Examples

- The following command returns by default only the max PVT value:

```
get_operating_voltage ld1
5.500000
```

- The following example shows all PVT values using three commands:

```
get_operating_voltage -pvt min ld1
4.500000
get_operating_voltage -pvt typ ld1
5.000000
get_operating_voltage -pvt max ld1
5.500000
```

Related Information

read_irdrop

read_rrf

reset_operating_voltage

set_operating_voltage

get_propagated_clock

```
get_propagated_clock [-clock clock_list] [-pin pin_list]
```

Returns the clock propagation mode, either `ideal` or `propagated`, for the given *`clock_list`* and *`pin_list`* arguments.

Note: The return value for the `get_propagated_clock` command is only accurate when `get_clock_propagation` command returns an `ideal` clock propagation mode. If the `get_clock_propagation` command returns a `propagated` clock propagation mode, the values returned by the `get_propagated_clock` command are not relevant.

Options and Arguments

`-clock` *`clock_list`*

Specifies one or more clock waveforms for which you want to get the clock propagation mode information.

`-pin` *`pin_list`*

Specifies the pin where you want to get the clock propagation mode.

Specifies one or more pins or ports for which you want to get the clock propagation mode information. No hierarchical pins are allowed in the `pin_list`.

Either the `clock_list` option or the `pin_list` option must be specified.You can also specify both the `clock_list` and the `pin_list` options.

Example

The following command gets the clock propagation mode information for all clock waveforms and the pin `ff1/CP`:

```
get_propagated_clock -clock {*} -pin {ff1/CP}
```

Related Information

`get_clock_propagation`

`reset_propagated_clock`

set_clock_propagation

set_propagated_clock

get_scale_delays

```
get_scale_delays [-net_delay | -cell_delay | -cell_check] [-pvt {min | typ | max}]
```

Returns the value of the delay scaling factor for a specific PVT corner (min, typ, or max) for cells or nets.

Options and Arguments

`-cell_check`
: Returns scaling on timing checks.

`-cell_delay`
: Returns the delay scaling factor on cells.
If neither the `-cell_delay` nor the `-net_delay` option is specified, the scaling factor for cells is returned. You cannot specify both `-cell_delay` and `-net_delay` in the same command.

`-net_delay`
: Returns the delay scaling factor on nets.

Note: Specifying `-cell` is equivalent to using both the `-cell_delay` and the `-cell_check` options.

`-pvt {min | typ | max}`
: Returns the delay scaling factor for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The `max` PVT corner delay scaling factor is returned.

Examples

- The following commands return delay scaling factor for nets:

```
get_scale_delays -net_delay -pvt min
0.950000
get_scale_delays -net_delay -pvt max
1.050000
```

- The following command returns cell delay scaling factor for the `max` PVT corner:

```
get_scale_delays
1.050000
```

Related Information

`set_scale_delays`

get_slack

`get_slack [-early | -late] net_or_pin_id`

Returns the slack of the given net or pin. The late mode slack is the worst case required time minus the worst case arrival time.

Options and Arguments

`-early`

Reports the amount of time that the data signal misses or passes the hold check. If the signal changes before the required hold time, it is early. Early slack is calculated as arrival time minus required time. If the signal arrives after the required hold time, there is no violation and the slack is positive. This is the opposite of late slack.

For clock nets or pins, using the `-early` option reports the time that the clock signal misses or passes the setup check.

`-late`

Reports the amount of time that the data signal misses or passes the setup check. If the signal changes after the required setup time, it is late. Late slack is calculated as required time minus arrival time. If the signal arrives before the required setup time, there is no violation and the slack is positive.
Default: Late check reports.

For clock nets or pins, using the `-late` option reports the amount of time that the clock signal misses or passes the hold check.

`net_or_pin_id`

Specifies the net or pin for which you want to get the slack.

Examples

- The following command returns the slack of net `out` within the current module and sub-modules as shown in Figure 7-3 on page 874:

```
get_slack [find -net out]
-2.000000
```

- The following command returns an error message if multiple net or pin ID arguments are specified:

```
find -registers -full -hier
J_block/B_reg J_block/A_reg I_block/B_reg I_block/A_reg
```

  Only one net ID or pin ID can be given as an input. If more than one ID is given, the following error message is displayed:

```
==> ERROR: Invalid object id'Id1 Id2....' <TCLCMD-117>.
```

  The following command reports the fanin cone of the specified design objects in the current module:

```
report_fanin J_block/A_reg/*
+-------------------------------------------------------------------+
| Level | From(Instance/Pin) | To(Instance/Pin) | Net          |
|-------+--------------------+------------------+--------------|
|   1   | J_block/A_reg/Q    |                  | J_block/w03  |
+-------+--------------------+------------------+--------------+
|   1   | J_block/A_reg/QN   |                  |              |
+-------+--------------------+------------------+--------------+
|   1   | J_block/in         | J_block/iC/A     | J_block/in   |
|   2   | J_block/iC/Z       | J_block/cbuf/A   | J_block/inC  |
|   3   | J_block/cbuf/Z     | J_block/A_reg/D  | J_block/PC   |
+-------+--------------------+------------------+--------------+
|   1   | J_block/clkc       | J_block/A_reg/CP | J_block/clkc |
+-------------------------------------------------------------------+
```

- The following commands get the slack based on the example report:
  - `get_slack [find -pin J_block/A_reg/D]`

    `1.285744`
  - `get_slack [find -net J_block/clkc]`

    `0.561901`
  - `get_slack -early [find -net J_block/clkc]`

    `1.081700`

Related Information

`get_module_worst_slack`

`get_area`

get_slew_thresholds

```
get_slew_thresholds
```

Returns a list containing the lower and upper slew threshold values in percentage used for the entire design.

Example

The following command returns the lower and upper slew threshold values used for the design:

```
get_slew_thresholds
10.000000 90.000000
```

Related Information

reset_slew_thresholds

set_tech_info

get_tech_info

```
get_tech_info {([-library list_of_library_names]
     {[-default_wire_load] | [-default_wire_load_selection] |
     [-default_operating_conditions] |
     [-default_fanout_load] | [-default_max_capacitance] |
     [-default_max_fanout] | [-default_max_transition] |
     [-default_min_capacitance] [-default_min_fanout] [-default_min_transition]
     [-input_threshold_pct_rise] | [-input_threshold_pct_fall] |
     [-output_threshold_pct_rise] | [-output_threshold_pct_fall] |
     [-slew_lower_threshold_pct_rise] | [-slew_lower_threshold_pct_fall] |
     [-slew_upper_threshold_pct_rise] | [-slew_upper_threshold_pct_fall]
     [-slew_lower_meas_threshold_pct_rise] |
     [-slew_lower_meas_threshold_pct_fall] |
     [-slew_upper_meas_threshold_pct_rise] |
     [-slew_upper_meas_threshold_pct_fall]}
     [-pvt {min | typ | max}]}
     |
     {[-library list_of_library_names]
     -cell list_of_cell_names
     {[-dont_modify] | [-dont_utilize] | [-scaling_factors] |
     [-input_threshold_pct_rise] | [-input_threshold_pct_fall] |
     [-output_threshold_pct_rise] | [-output_threshold_pct_fall] |
     [-slew_lower_threshold_pct_rise] | [-slew_lower_threshold_pct_fall] |
     [-slew_upper_threshold_pct_rise] | [-slew_upper_threshold_pct_fall]
     [-slew_lower_meas_threshold_pct_rise] |
     [-slew_lower_meas_threshold_pct_fall] |
     [-slew_upper_meas_threshold_pct_rise] |
     [-slew_upper_meas_threshold_pct_fall]}
     [-pvt {min | typ | max}]}
     |
     ([-library list_of_library_names]
     -cell list_of_cell_names
     -pin list_of_pin_names
     {[-fanout_load] | [-max_fanout] | [-min_fanout]
     [-max_transition] | [-min_transition] |
     [-max_capacitance] | [-min_capacitance]}
     [-pvt {min | typ | max}])}
```

Returns data for the specified library parameters in the named target libraries. The data can come either from the library itself or the overrides (assertions) as previously specified with the `set_tech_info` command.

Options and Arguments (any level)

`-library` *`list_of_library_names`*
: Reports data for the named libraries.

Default: All libraries specified with the `set_global target_technology` command.

`-pvt {min | typ | max}`
Returns the value for the environment corner of interest. Specify one PVT corner at a time.
Default: `typ`

Options and Arguments (library level)

`-default_fanout_load`
Returns the value of the default fanout load for all pins on all cells in the library.

`-default_max_capacitance`
Returns the value of the default maximum capacitance for all pins on all cells in the library.

`-default_max_fanout`
Returns the value of the default maximum fanout for all pins on all cells in the library.

`-default_max_transition`
Returns the value of the default maximum transition time for all pins on all cells in the library.

`-default_min_capacitance`
Returns the value of the default minimum capacitance for all pins on all cells in the library.

`-default_min_fanout`
Returns the value of the default minimum fanout for all pins on all cells in the library.

`-default_min_transition`
Returns the value of the default minimum transition time for all pins on all cells in the library.

`-default_operating_conditions`
Returns the default operating conditions for the library.

`-default_wire_load`
: Returns the default wire-load model for the library.

`-default_wire_load_selection`
: Returns the default wire-load selection table for the library.

`-input_threshold_pct_fall`
: Returns the value of the default input threshold percent for the falling edge for all cells in the library.

`-input_threshold_pct_rise`
: Returns the value of the default input threshold percent for the rising edge for all cells in the library.

`-output_threshold_pct_fall`
: Returns the value of the default output threshold percent for the falling edge for all cells in the library.

`-output_threshold_pct_rise`
: Returns the value of the default output threshold percent for the rising edge for all cells in the library.

Note: The following slew threshold options refer to the section of the waveform where the slew is nearly linear. This is measured and extended like a linear waveform. The linear waveform, a result of extending the near-linear slew waveform, is measured at certain points. The points where this extended waveform is measured are called slew thresholds. Measured slew thresholds are determined at the time of library characterization.

Slew Thresholds

`-slew_lower_threshold_pct_fall`
: Returns the value of the default lower threshold percent for the slew time of the falling transition for all cells in the library.

`-slew_lower_threshold_pct_rise`
: Returns the value of the default lower threshold percent for the slew time of the rising transition for all cells in the library.

`-slew_upper_threshold_pct_fall`
: Returns the value of the default upper threshold percent for the slew time of the falling transition for all cells in the library.

`-slew_upper_threshold_pct_rise`
Returns the value of the default upper threshold percent for the slew time of the rising transition for all cells in the library.

Measured Slew Thresholds

`-slew_lower_meas_threshold_pct_fall`
Returns the measured lower threshold percent of the slew time for the falling transition for all cells in the library.

`-slew_lower_meas_threshold_pct_rise`
Returns the measured lower threshold percent of the slew time for the rising transition for all cells in the library.

`-slew_upper_meas_threshold_pct_fall`
Returns the measured upper threshold percent of the slew time for the falling transition for all cells in the library.

`-slew_upper_meas_threshold_pct_rise`
Returns the measured upper threshold percent of the slew time for the rising transition for all cells in the library.

Options and Arguments (cell level)

`-cell` *`list_of_cell_names`*
Reports assertions on the named cells. Required option for cell and pin level assertions.

`-dont_modify`
Returns `true` or `false`. If `true`, the cell is not modified in optimization or time budgeting.

`-dont_utilize`
Returns `true` or `false`. If `true`, the cell is not used in optimization.

`-input_threshold_pct_fall`
Returns the value of the default input threshold percent for the falling edge for the listed cells.

`-input_threshold_pct_rise`
: Returns the value of the default input threshold percent for the rising edge for the listed cells.

`-output_threshold_pct_fall`
: Returns the value of the default output threshold percent for the falling edge for the listed cells.

`-output_threshold_pct_rise`
: Returns the value of the default output threshold percent for the rising edge for the listed cells.

`-scaling_factors`
: Returns the name of the derating table used for scaling.

Note: The following slew threshold options refer to the section of the waveform where the slew is nearly linear. This is measured and extended like a linear waveform. The linear waveform, a result of extending the near-linear slew waveform, is measured at certain points. The points where this extended waveform is measured are called slew thresholds. Measured slew thresholds are determined at the time of library characterization.

Slew Thresholds

`-slew_lower_threshold_pct_fall`
: Returns the value of the default lower threshold percent for the slew time of the falling transition for the listed cells.

`-slew_lower_threshold_pct_rise`
: Returns the value of the default lower threshold percent for the slew time of the rising transition for the listed cells.

`-slew_upper_threshold_pct_fall`
: Returns the value of the default upper threshold percent for the slew time of the falling transition for the listed cells.

`-slew_upper_threshold_pct_rise`
: Returns the value of the default upper threshold percent for the slew time of the rising transition for the listed cells.

Measured Slew Thresholds

`-slew_lower_meas_threshold_pct_fall`
Returns the measured lower threshold percent of the slew time for the falling transition for the listed cells.

`-slew_lower_meas_threshold_pct_rise`
Returns the measured lower threshold percent of the slew time for the rising transition for the listed cells.

`-slew_upper_meas_threshold_pct_fall`
Returns the measured upper threshold percent of the slew time for the falling transition for the listed cells.

`-slew_upper_meas_threshold_pct_rise`
Returns the measured upper threshold percent of the slew time for the rising transition for the listed cells.

Options and Arguments (pin level)

`-fanout_load`
Returns the value of the fanout load constraint for the named pins. Specify this option only for an input pin.

`-max_capacitance`
Returns the value of the maximum capacitance constraint for the named pins. Specify this option only for an output pin.

`-max_fanout`
Returns the value of the maximum fanout constraint for the named pins. Specify this option only for an output pin.

`-max_transition`
Returns the value of the maximum transition time constraint for the named pins. Specify this option for both input and output pins.

`-min_capacitance`
Returns the value of the minimum capacitance constraint for the named pins. Specify this option only for an output pin.

`-min_fanout`

Returns the value of the minimum fanout constraint for the named pins. Specify this option only for an output pin.

`-min_transition`

Returns the value of the minimum transition time constraint for named pins. Specify this option for both input and output pins.

`-pin` *`list_of_pin_names`*

Reports assertions on the named pins. Required option for pin level assertions.

Note: The pin assertions apply to a specific pin type. If the correct pin type is not specified, an error is issued.

Examples

These examples follow those for the `set_tech_info` command:

- The following command returns the value of the default fanout load for all pins on all cells in the library:

```
get_tech_info -library lib1 -default_fanout_load
1.411100
```

- The following command returns `true` or `false`. If `true`, the cell is not used in optimization:

```
get_tech_info -cell FD2ESSA -dont_utilize
true
```

- The following command reports data for the named libraries.

```
get_tech_info -library lca300kv -cell B2I -pin Z1 Z2 -pvt min -max_fanout
3.000000
```

Default: All libraries specified with the `set_global target_technology` command

Related Information

`reset_tech_info`

`set_global target_technology`

`set_tech_info`

`write_library_assertions`

get_time_borrow_limit

`get_time_borrow_limit [-pin` *`pin_list]`* `[-clock` *`list_of_clksigs`*`]`

Returns the limit on time that can be borrowed by one cycle from the next cycle as previously specified with the `set_time_borrow_limit` command. Query for borrow limits on pins or waveforms.

Note: This command only returns an assertion value when one has been set by the `set_time_borrow_limit` command. Nothing is returned if no assertion has been set.

Options and Arguments

`-clock` *`list_of_clksigs`*

Returns the borrow limit for the specified list of ideal (or generated) clock names.

`-pin` *`pin_or_instance_list`*

Returns the borrow time limit for a single pin on a latch.

You can alternatively give one latch instance name, which returns the assertion for the clock pin of the instance.

Examples

The following commands return the limit on time that can be borrowed by one cycle from the next cycle as previously specified with the `set_time_borrow_limit` command on a pin:

```
set_time_borrow_limit -pin l1/q 3.0
get_time_borrow_limit -pin l1/q
3.0
```

Related Information

`reset_time_borrow_limit`

`set_time_borrow_limit`

Analyzing Latch-Based Designs in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

get_time_unit

```
get_time_unit
```

Returns the time unit for the session as previously set by the `set_time_unit` command.

Example

The following command returns the time unit for the session:

```
get_time_unit
0.001
```

Related Information

`reset_time_unit`

`set_time_unit`

get_timing

```
get_timing [-early | -late] [-rise | -fall] pin_path_or_id
     {arrival | required | edge | slack | slew | stolen| cell | net | instance
     | pin | clkordata | clkordatapin | phase | load | pinload | wireload | fanout
     | hpin}
```

Retrieves timing information for the specified pin property on the given pin. Specify the pin by the object ID or the hierarchical name. Only one pin is allowed per command.
Default: Return information about the most critical path.

Note: If no information is found, this command does not return anything.

Slack is a function of required time and arrival time, both must be present in order for the `get_timing` command to return a value for `slack`, `arrival`, `required`, `edge`, and `stolen`.

Tip

Run `check timing` first and resolve all warnings that cause paths to pins to be unconstrained. For example:
`--> WARNING: No required or external delay assertion found at port DOUT.`
A constraint consists of a required time that can be compared to an arrival time. Because port `DOUT` does not have a required time, any path to it is unconstrained.

Options and Arguments

`arrival`

Returns the arrival time of the path causing the worst slack at the given pin.

`cell`

Returns the cell of the given pin's instance.

`clkordata`

Tells you whether the specified path is of type clock or of type data. Change this behavior using the `-rise` or `-fall` option.

`clkordatapin`

Tells you whether the specified pin expects a clock or data signal. This information is currently derived from the library, except in the case of output ports. Output ports are considered of type `DATA` if there is a delay constraint asserted on the port. This behavior

is consistent with the `check_timing` command. This option is not path dependent.

`-early | -late`

Specifies analysis type, either early or late.
Default: `late`

`edge`

Returns the edge of the worst slack causing path at the given pin. Rising edge is ^, falling edge is v.

`fanout`

Returns the total fanout on a given pin.

`hpin`

Returns the full hierarchical name of the pin. Equivalent to using the `-full_path_name` option with the `get_fanin`, `get_fanout`, or `find` commands.

`instance`

Returns the hierarchical name of the given pin's instance.

`load`

Returns the total capacitive load on a given pin.

`net`

Returns the hierarchical name of the net connected to given pin.

`phase`

Returns the phase of a pin in the same format as using the `report_cell_instance -timing` command.

`pin`

Returns the pin name of the given hierarchical pin.

`pinload`

Returns the total capacitive load from pins on a given pin, including its own load.

`pin_path_or_id`

Specifies the port or pin for which timing information is to be returned. Specify the port or pin by the object ID or the hierarchical name.

`required`

Returns the required time of the path causing the worst slack at the given pin.

`-rise | -fall`

Specifies the edge, either rise or fall.
Default: The edge with the worst timing.

`slack`

Returns the worst slack at the given pin.

`slew`

Returns the propagated slew at the given pin.

`stolen`

Returns the slack stolen at the given pin (only visible on the output of transparent latches).

`wireload`

Returns the total capacitive load from the wire on a given pin.

Examples

- The following command returns the arrival time of the path causing the worst slack at the given pin:

```
get_timing i00/Z arrival
1.23
```

- The following command returns the hierarchical name of the net connected to given pin:

```
get_timing i00/Z net
n001
```

- The following command returns the edge of the worst slack causing path at the given pin. Rising edge is ^, falling edge is v:

```
get_timing in edge
^
```

- The following command tells whether the specified path is of type clock or of type data:

```
get_timing clkC clkordata
CLOCK
```

- The following command returns the phase of a pin:

```
get_timing clkC phase
CLK2(C)(P)
```

Related Information

report_cell_instance

report_net

report_ports

report_timing

set_global slew_propagation_mode

get_top_timing_module

```
get_top_timing_module
```

Returns the object ID of the module that was designated as the top timing module by the `set_top_timing_module` command.

Examples

- The following command returns the object ID of the module that was designated as the top timing module:

  ```
  get_top_timing_module
  67841
  ```

- The following command returns the names of specified objects in the design:

  ```
  get_names 67841
  top
  ```

Related Information

`get_names`

`set_top_timing_module`

libcompile

```
libcompile [-force] (UNIX command) [-help] [-version] [-expire] [-queue]
     [-ipformat] [-debug] [-verbosity level] [-logfile file_name]
     [library_file_name] [output_file_name]
```

Support for `libcompile` is being phased out. Use the `read_dotlib` or the `read_tlf` command to read `.lib` files directly. Some `.lib` constructs are supported in `read_dotlib` and `syn2tlf`, but not in `libcompile`.

Note: The following `.lib` constructs are supported by the `read_dotlib` command and `syn2tlf` only:

- Interface Timing Specification (ITS) constructs such as generated clocks, functional modes, interface timing, and so forth.
- Conditional expressions involving bus pins.
- Rise/fall capacitance.

See Using Synopsys Liberty (.lib) Libraries in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)* for more information.

load_dcl_rule

```
load_dcl_rule
```

Loads a binary Delay and Power Calculation Module (DPCM) into memory. The DPCM is an executable shared library that is linked to an application (like BuildGates Synthesis) at runtime. DPCM is often abbreviated as DCM. After loading the DCM, all timing calculations are performed using delay tables and equations defined in the DCM.

Note: Before loading a DCM, set environment variables before running `bg_shell`. For example:

```
setenv DCMRULEPATH /dcl_libs/sample_dcm
setenv DCMRULESPATH /dcl_libs/%RULENAME
setenv DCMTABLEPATH /dcl_libs/
```

Refer to the DCL library vendor documentation for information about how to set these variables.

Example

The following command loads a binary Delay and Power Calculation Module (DPCM) into memory:

```
load_dcl_rule
Info:     Library 'MY_DCM' was loaded from file './DCL_OLA_Libs/DCMinterface_SOL'
         <TCLCMD-701>.
```

Related Information

`get_dcl_functional_mode_array`

`read_ola`

`set_dcl_calculation_mode`

`set_dcl_functional_mode`

`set_dcl_level`

`set_operating_parameter`

Using IEEE 1481 Delay and Power Calculation System (DPCS) Libraries in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

read_alf

```
read_alf [-force]{alf_library_name | {{-min filename -max filename -typ
     filename [-name merged_filename]} | alf_library_filename |
     {list_of_alf_filenames}}
```

Reads the Advanced library format (ALF) file. The binary .alf files are previously compiled from the .lib format using the libcompile utility. For more details, see "Using Synopsys Liberty (.lib) Libraries" in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Use this command in two ways:

- To read an ALF library into the database, use this form:

  ```
  read_alf alf_filename
  ```

  The above command loads the library data from *alf_filename*. CTE uses the library data for all min, typ, and max process, voltage, and temperature (pvt).

  Note: When you are not merging libraries, you can only read one library at a time. If you try to read in more than one library, for example, read_alf *1.alf 2.alf*, without using the following merged library form, you will receive an error.

- To create and read a merged library from single operating point libraries, use this form:

  ```
  read_alf [-min filename] [-typ filename] [-max filename]
  [-name merged_filename]
  ```

  Use the -min, -typ, -max options to create a merged library from libraries different operating points. Use the min library for min pvt analysis, and use the typ library and max library for typ pvt and max pvt respectively. When merging two libraries or more, the libraries have to define the same cell names with the same number and type of timing arcs. See Examples on page 905.

  The values from each library are scaled to the current session units using the units from that library. The first library read_ command will provide the default session units. The library for the max corner is used for the default units if it is present, otherwise the typ corner is used, or finally, the min corner.

  Use the -name option to change the name of the merged library to a new name. For example, -name merged_lib_name changes the merged library name to merged_lib_name.

  When merging libraries, specify libraries for at least two pvt corners.

Options and Arguments

alf_library_name
: Specifies the ALF library file name. This name can be a fully qualified file name or a relative file name. If the AMBIT_SLIB_PATH environment variable is set, the search is performed in the order specified.

-force
: Overrides invalid 0-100 percent measured threshold values. If you do not use this option, CTE will not read libraries that define 0-100 percent threshold values.
Default: Delay threshold value is 50-50, and measured slew threshold is 20-80.

list_of_alf_filenames
: Lets you specify multiple library names. Use curly braces to specify multiple library filenames, for example:
{lib1 lib2 lib3}

-max *filename*
: Uses data from library *max_filename* to populate the max field of the merged library.

-min *filename*
: Uses data from library *min_filename* to populate the min field of the merged library.

-name *merged_filename*
: Changes the new merged library name to *merged_lib_name*.

-typ *filename*
: Uses data from the library *filename* to populate the typ field of the merged library.

Examples

- The following command reads the LSI Logic G11 library:

```
read_alf lca_G11.alf
```

- The following command shows that ALF files can contain data for multiple operating points:

```
read_alf library_with_min_and_typ_and_max_data.alf
```

Suppose `min.alf` contains `0.02`, `typ.alf` contains `0.03`, and `max.alf` contains `0.04`, the resultant triplet in `merged_lib` is `(0.02:0.03:0.04)`.

Related Information

`read_adb`

`read_dotlib`

`read_library_update`

`read_tlf`

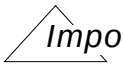
`read_verilog`

`read_vhdl`

See Using Advanced Library Format (ALF) Libraries in *Common Timing Engine (CTE) User Guide*.

read_ctlf

```
read_ctlf
     OBSOLETE: Use read_tlf instead.
```



Important

The `read_ctlf` command is no longer supported. Obtain a `.tlf` source file with version 4.4 and use `read_tlf`.

Related Information

`read_tlf`

"Using Timing Library Format (TLF) Libraries" in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

read_dc_script

```
read_dc_script  [-dc] [-group_path_effort {weight_boundary1 weight_boundary2}]
     [-group_path_for_clocks] [-tcl] [-verbose] [-help] write_script_file

     The options -ambit ambit_output_file, -apply_only, -write_only, -no_write,
     and -no_apply are obsolete. To enable the old options, set the following tcl
     variable:
     set syn2ambit_flow_control_support 0
```

Translates Synopsys PrimeTime and Design Compiler constraints to the CTE constraint format automatically. Also automatically detects whether the `write_script` file is in Tcl or `dcsh` format. See the *SDC Constraint Support Guide* for details about this command.

Important

Do not mix CTE and PrimeTime constraints. Mixing CTE and PT constraints causes different interpretations of constraints. Once you use the `read_dc_script` command to read constraints, all other constraints must be read before using this command.

Using the `read_dc_script` command automatically sets the `set_flow_compatible_mode` command to `on`.

The `-through` option on a net name will apply on a driver pin/port in CTE, for the `set_false_path` command and the `set_multicycle_path` command.

Note: When using the `set_false_path`, `set_multicycle_path`, or the `set_path_delay` commands in your `read_dc_script`, you may see the following error with both the `-sparc` and the `-64` options:

```
==> ERROR:   An unrecoverable exception has occurred (SEGV).
--> Info:    Probably due to a stack overflow.
--> Info:    Consider a higher stacksize limit; see limit(1).
```

This occurs only on Solaris 32 and 64 platforms when you have a large number of the above commands, or if you use a large number of pins as arguments to the `-to`, `-from`, or the `-through` options with the these commands. To prevent this error, before starting PKS or Timing Analysis, enter the following command from the UNIX prompt:

```
limit stacksize unlimited
```

Options and Arguments

-dc

Specifies that the Synopsys Synthesis constraints file is in the `dc_shell` format. To read a dc-format file, specify the `-dc`

option with the `read_dc_script` command and set the `syn2ambit_no_tcl_detection tcl` variable to `true`.

Note: Without the `-dc` option, the `read_dc_script` command assumes that the input file is in tcl format.

`-group_path_effort {`*`weight_boundary1 weight_boundary2`*`}`

The mapping between Synopsys `weight` and CTE `effort` is handled as follows:

weight	effort
Between 0 and 1.0	`low`
Between 1.0 and 10.0	`medium`
above 10.0	`high`

The `-group_path_effort` option lets you change the upper boundary values (by default, 1.0 and 10.0). For example, `read_dc_script -group_path_effort {2.0 15.0}` changes the mapping to the following:

weight	effort
Between 0 and 2.0	`low`
Between 2.0 and 15.0	`medium`
above 15.0	`high`

`-group_path_for_clocks`

Changes the default behavior. By default, the `create_clock` and `create_generated_clock` commands do not create any group path. When you use this option, `create_clock` and `create_generated_clock` add the following constraint:

`set_path_group -name syn2ambit_`*`clock_name`*
`-clock_to `*`clock_name`*

`-help`

Prints the `read_dc_script` usage message.

`-verbose`

Specifies that the translator should write the line number and the

corresponding Synopsys Synthesis command in the Ambit Synthesis constraints output along with the translated command.

write_script_file

Specifies the name of the Synopsys Synthesis constraints input file. This is a required argument and must be the last specified argument.

Related Information

write_sdc

set_flow_compatible_mode

read_dotlib

```
read_dotlib {{-min filename -max filename -typ filename
     [-name merged_filename]} | dotlib_library_filename |
     {list_of_dotlib_filenames}} [-nopower] [-nonoise] [-notiming]
```

Reads the Synopsys Liberty file for timing data. For more details, see "Using Synopsys Liberty (.lib) Libraries" in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Use this command in the following two ways.

- To read the .lib library into the database, use this form:

  ```
  read_dotlib dotlib_library_filename
  ```

  The above command loads the library data from the *dotlib_library_filename*. CTE uses the library data for all min, typ, and max pvt.

- To create and read a merged library from multiple single operating point libraries, use this form:

  ```
  read_dotlib [-min filename] [-typ filename]
  [-max filename] [-name merged_filename]
  ```

  Use the -min, -typ, -max options to create a merged library. Use the min library for min pvt analysis, and use the typ library and max library for typ pvt and max pvt respectively. When merging libraries, the libraries have to define the same cell names with the same number and type of timing arcs. See Examples on page 912.

  Use the -name option to change the name of the merged library to a new name. For example, -name *merged_filename* changes the merged library name to *merged_filename*.

Options and Arguments

dotlib_library_filename
: Specifies the name of the .lib library. This name can be a fully qualified file name or a relative file name. If the AMBIT_SLIB_PATH environment variable is set, the search is performed in the order specified.

list_of_dotlib_filenames
: Specifies multiple dotlib library filenames. Use curly braces to specify multiple library filenames, for example:
  {lib1 lib2 lib3}

`-max` *`filename`*

Uses data from library *`max_filename`* to populate the `max` field of the merged library.

`-min` *`filename`*

Uses data from library *`filename`* to populate the `min` field of the merged library.

`-name` *`merged_filename`*

Changes the new merged library name to a *`merged_filename`*.

`-nonoise`

Does not read the Liberty SI data, such as `noise_immunity` and `noise_propagation` from the `.lib` file.

`-nopower`

Does not read the power data from the `.lib` file.

`-notiming`

Does not read the timing data present in the library.

Note: You can use the `-nonoise`, `-nopower`, and the `-notiming` options together in any combination. Using these options saves time and memory while reading a `.lib` file.

`-typ_`*`filename`*

Uses data from library *`filename`* to populate the `typ` field of the merged library.

Examples

- The following command reads the *`dotlib_library_filename`* library:

  ```
  read_dotlib dotlib_library_filename
  ```

- The following command populates data from each respective single operating point library into the `min`, `typ`, `max` fields of the merged library named `merged_lib`:

  ```
  read_dotlib -min min.lib -typ typ.lib -max max.lib -name merged_lib
  ```

  Suppose that the pin capacitance value for `min.alf`,`typ.alf`, and `max.alf` are 0.02, 0.03, and 0.04 respectively. The resultant triplet for pin capacitance in `merged_lib` is `(0.02:0.03:0.04)`.

- The following command reads only the timing data from a.lib file and skips the power and noise data:

```
read_dotlib -nopower -nonoise a.lib
```

- The following command will not load constructs such as `internal_power` from the `.lib` file:

```
read_dotlib -nopower a.lib
```

Related Information

`read_adb`

`read_alf`

`read_tlf`

`read_verilog`

`read_vhdl`

read_irdrop

`read_irdrop [-pvt {min|typ|max}] filename`

Reads an irdrop report file generated by VoltageStorm and asserts the operating voltage accordingly.

Options and Arguments

filename

Specifies the name of the irdrop file that has the voltage drop data.

-pvt {min | typ | max}

Applies an operating voltage specified in the filename for a specific PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces. If you do not specify a -pvt value, by default, the operating voltage specified in the filename is applied to all three PVT corners.

Example

The following command applies an operating voltage specified in the filename into operating voltages for a min PVT corner:

`read_irdrop -pvt min voltage_storm_ir_drop_report_file.ir`

Related Information

get_operating_voltage

read_rrf

reset_operating_voltage

set_operating_voltage

read_library_update

```
read_library_update [-library library_name] [-pvt {min | typ | max}]
     [-wireloads] [-scaling_factors] [-operating_conditions][-cells]
     [-spice spice_subckt_file] filename
```

Replaces existing technology library data with new data, and updates wire load, operating conditions, and cell-specific information in the library.

Note: The `read_library_update` command is not a substitute for directly reading a complete `.lib` file. This command updates wireloads and a limited set of globals from the `.lib` file.

If none of the three options (`wireloads`, `operating_conditions`, `cells`) are specified, all wire-load models, operating conditions, and cells from *`libname`* are used to update the library.

Note: The `read_library_update` command updates scaling factor groups to the target library similar to a wire-load models update. Use the `-scaling_factors` option to set a new scaling factors group for a cell or to change the scaling factors group of a cell within a library.

Tip

You can only update a library with linear models with data from another library with linear models. The same is true for nonlinear models. The defaults and k-factors are significantly different for linear and nonlinear library types. A forced overwrite of one over the other is not desirable. To use both linear and nonlinear libraries, read them in with separate `read_tlf` or `read_alf` statements, then set the global `target_technology` to the list of libraries. In this flow, without overwriting data, the cells and timing information from the corresponding libraries is used.

Options and Arguments

`-cells`

Adds cells from the file *`filename`* to the library. If the cell already exists in the library, an error is reported.

`filename`

Specifies the name of the ASCII file that contains the new data. The file can be `.tlf`, `.alf`, or `.lib` file types. Required argument.

`-library` *`library_name`*

Specifies the name of the library to be updated. If this option is

not used, the target library specified by the `set_global target_technology` is updated.

`-operating_conditions`

Indicates that the operating conditions specified in the file *`filename`* are used to update the library.

`-pvt {min | typ | max}`

Replaces library data for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The library is updated for all three PVT corners.

`-scaling_factors`

Uses the scaling factor groups from the file for updating the scaling factor groups in the library. If a scaling factor group already exists in the library, then it is overwritten by the one from the file.

`-spice` *`spice_subckt_file`*

Reads the vendor provided *`spice_subckt_file`* that contains Spice subcircuit definitions of cells in the technology library. CTE reads the pin order of each SUBCKT defined and uses the pin order when it generates the Spice netlist of a path.

For cells that do not have a Spice SUBCKT definition, CTE uses the cell pin order specified in the timing library.

For more information about generating a Spice netlist for a path, see the `report_timing` command Options for Creating Spice Output.

`-wireloads`

Uses the wire-load models from the file for updating the wire-load models in the library. If a wire-load model already exists in the library then it is overwritten by the one from the file.

Examples

- The following command adds cells in library file `memgen.lib` to the library previously specified by `set_global target_technology`.

```
read_library_update -cells memgen.lib
```

- The following command loads TLF library `lib1.tlf`:

  ```
  read_tlf lib1.tlf
  ```

- The following command appends all information from TLF library `lib2.tlf` to `lib1.tlf`:

  ```
  read_library_update lib2.tlf
  ```

- The following command updates the target technology library with data from `corelib.lib` and adds the Spice SUBCKT definitions from `corelib.spf`:

  ```
  read_library_update -spice corelib.spf corelib.lib
  ```

Related Information

`read_adb`

`read_alf`

`read_tlf`

`read_verilog`

`read_vhdl`

`set_global target_technology`

read_ola

```
read_ola
```

Loads an OLA into memory in the form of a DCM (compiled module) and initializes it for use in BuildGates Synthesis. Upon loading the OLA library, all timing calculations are performed using the OLA delay tables and equations. The synthesis properties and functions are also derived from the OLA library.

Note: Before loading an OLA library, set certain environment variables before running the `ac_shell`.The following list shows an example:

```
setenv DCMRULEPATH /dcl_libs/sample_dcm
setenv DCMRULESPATH /dcl_libs/%RULENAME
setenv DCMTABLEPATH /dcl_libs/
```

Refer to the OLA library vendor documentation for information about how to set these variables.

See "Using Open Library API (OLA) Libraries" in the *Common Timing Engine (CTE) User Guide* for more information.

Example

This example shows the proper sequence for loading OLA libraries when using a DPCM:

- At the UNIX prompt, set the library path:

  ```
  setenv LD_LIBRARY_PATH "/libdata/lib"
  ```

- At the `shell` prompt:

  - Set the UNIX environment variables:

    ```
    set env(DCMRULEPATH) "/libdata/OLA/myDPCM"
    set env(DCMTABLEPATH) "/libdata/OLA/tables:/libdata/OLA/%RULENAME"
    ```

  - Load the library:

    ```
    read_ola
    ```

  - Merge (backannotate) wire-load information from a floorplanner:

    ```
    read_library_update my_wire_load.pdef
    ```

Related Information

`load_dcl_rule`

`set_dcl_calculation_mode`

set_dcl_functional_mode

set_dcl_level

set_operating_parameter

read_rrf

`read_rrf [-pvt {min | typ | max}] rrf_filename`

Reads the SE power analyzer output Rail Result File (RRF), which contains the power and ground `IR` drop information. The relevant information is read into CTE and converted internally into operating voltages on leaf level instances.

See Performing Voltage Drop Analysis in the *Common Timing Engine (CTE) User Guide* for more information.

Read in a compressed RRF file in GNU zip format if the specified input file ends in the .gz suffix. No special option is needed to read a gzipped file. For example: `read_rrf rrf_filename.rrf.gz`. However, you cannot create a gzipped file with the `read_rrf` command.

This command is used in the SE-PKS flow. Run the SE power analyzer before using the `read_rrf` command.

Options and Arguments

`-pvt {min | typ | max}`
: Converts the data into operating voltages for a particular PVT (process, voltage, temperature) corner. You can choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces (`{}`) and separate the values by spaces.
Default: The data is converted into operating voltages for all three PVT corners.

`rrf_filename`
: Specifies the name of the RRF file that has the voltage drop data.

Examples

- The following command converts the data into operating voltages for a min PVT corner:

  ```
  read_rrf -pvt min route1.rrf
  ```

- The following command converts voltage drops from this file into operating voltages on leaf level instances used to derate delays and slews of cells during timing analysis:

  ```
  read_rrf pa_1.rrf
  ```

Related Information

`get_operating_voltage`

`read_irdrop`

`reset_operating_voltage`

`set_operating_voltage`

read_sdf

```
read_sdf [-continue_on_error] [-ignore_sdf_cond] [-min |-typ | -max]
     [-store_as_min | -store_as_typ | -store_as_max] [-scale float]
     [-worst_case_mismatched_conds] sdf_filename
```

Reads a Standard Delay Format (SDF) file. After the physical design is complete, backannotate the post-layout delay data for further timing analysis, and use the `read_sdf` command to read SDF files into the design. See Reading Standard Delay Format (SDF) Files in the *Common Timing Engine (CTE) User Guide* for more information.

For supported SDF constructs, see Supported SDF Constructs in the *Common Timing Engine (CTE) User Guide*.

Read in a compressed SDF file in GNU zip format if the specified input file ends in the `.gz` suffix. No special option is needed to read a gzipped file. For example: `read_sdf sdf_filename.gz`. However, you cannot create a gzipped file with the `read_sdf` command.
Default: Read all the MTM information in the SDF, and store it as it is presented in the SDF. For example:

```
read_sdf SDF
```

SDF	Command	CTE
`(1:2:3)`	`read_sdf` (no options)	`(1:2:3)`

Use the `-min` | `-typ` | `-max` options for pulling the values only from a given MTM field. For example:

```
read_sdf -max SDF1
read_sdf -typ SDF2
read_sdf -min SDF3
```

SDF1	SDF2	SDF3	Commands	CTE
`(2.8:2.9:3)`	N/A	N/A	`read_sdf -max`	`(::3)`
N/A	`(2.0:2:2.1)`	N/A	`read_sdf -typ`	`(:2:3)`
N/A	N/A	`(1:1.1:1.2)`	`read_sdf -min`	`(1:2:3)`

In the example session, the first `read_sdf` loads the SDF1 max value into the CTE `max` field, and leaves the `min` and `typ` fields unannotated. The next command gets the `typ` data from another SDF, which still leaves the `min` field unannotated. The third command completes the annotation by picking up the `min` value from yet another SDF.

Caution

Partially annotated SDFs can give unexpected results. Make sure you fully annotate min/typ/max.

Tip

The `-min` | `-typ` | `-max` and `-store_as_min` | `-store_as_typ` | `-store_as_max` options provide a powerful mechanism for reading and storing the `-min` | `-typ` | `-max` values from multiple SDF files. However, use these options carefully (see Caution above).

Options and Arguments

`-continue_on_error`

Continues to read a file if an error occurs. Usually when an error occurs, the `read_sdf` command stops reading the file. By specifying this option, you can attempt to continue reading the file if possible. Use caution if you continue because some annotations could be missing due to the errors. Use the `report_annotations` command to find missing annotations. If the file is corrupted, obtain a new SDF file.

`-ignore_sdf_cond`

Allows the SDF reader to support SDF files with dummy conditions used to speedup simulation. This option is distinguished from the `-worst_case_mismatched_conds` option that lets the SDF reader match the SDF assertions to timing and check arcs even when their conditions do not match. See Figure 7-4 on page 926.
With the `-ignore_sdf_cond` option, the SDF delay and check assertion is matched with a timing and check arc only if the condition of the SDF assertion matches the condition of the timing/check arc, or the SDF assertion has a condition (dummy condition) and the library timing/check arc does not have one.

`-min` | `-typ` | `-max`

Specifies only one field, either the `min`, `typ`, or `max` numbers from the SDF file to be applied to the arc.
Default: This number is applied to all three delay values in the timing system. To override this behavior, use one of the `-store_as_*` options.

`-scale` *`float`*

Scales all values in the SDF file by the given number.

`sdf_filename`

Specifies the name of the SDF file that has the backannotation data.

`-store_as_min` | `-store_as_typ` | `-store_as_max`

Tells the system to store the specified value into a specific value in the timing system. Must be used with one of the `-min` | `-typ` | `-max` options.These options must be compatible with `pvt_early_path` and `pvt_late_path` globals.

For example, to load the minimum values from an SDF file and ignore the other two values, use one of the following sets of commands:

```
set_global pvt_early_path min
set_global pvt_late_path min
read_sdf -min -store_as_min
```

or:

```
set_global pvt_early_path typ
set_global pvt_late_path typ
read_sdf -min -store_as_typ
```

or simply:

```
read_sdf -min -store_as_max
```

since `pvt_early_path` and `pvt_late_path` are set to `max` by default.

If you want to use the minimum SDF values for early paths and the typical SDF values for late paths, use the following commands:

```
set_global pvt_early_path min
set_global pvt_late_path typ
read_sdf -min -store_as_min
read_sdf -typ -store_as_typ
```

or:

```
set_global pvt_early_path min
set_global pvt_late_path max
```

```
read_sdf -min -store_as_min
read_sdf -typ -store_as_max
```

Note: This option assumes that you have previously annotated the design from another SDF file with all three values or that you will eventually fill in all values with subsequent `read_sdf -store_as_*` commands. The behavior of the timing analysis is not defined unless all three fields are annotated.

`-worst_case_mismatched_conds`

Specifies that the SDF file contains conditional delays that do not match the conditional delays in the library. When this option is specified, the mismatched condition is compared to the default conditional delay and the worst delay is used.
Default: Mismatched conditional delays in the SDF file are ignored. See Figure 7-4 on page 926.

Examples

- The following command scales all delay values in the file `typ.sdf` by 2:

  ```
  read_sdf -scale 2 typ.sdf
  ```

- The following commands store the `min` values from `min.sdf` in the `min` delay of the timing system, and so on for `typ` and `max` values. This method is useful when setting up a model for simultaneous worst case, best case analysis:

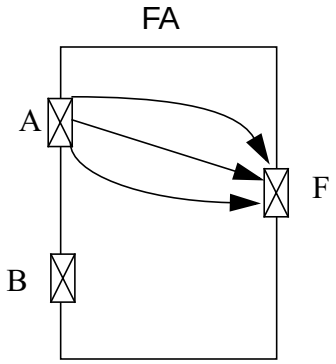
  ```
  read_sdf -min -store_as_min min.sdf
  read_sdf -typ -store_as_typ typ.sdf
  read_sdf -max -store_as_max max.sdf
  ```

  Default: The default timing analysis type is `min_max`.

  Note: For best-case/worst-case, you must use `set_global timing_analysis_type bc_wc`. For more information about best-case/worst-case analysis, see Analyzing On and Off Chip Variation in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

- Example 7-1, 2-2, and 2-3 show how to use the `-worst_case_mismatched_conds` and the `-ignore_sdf_cond` options. Assume a timing model as shown in Figure 7-4.

Figure 7-4 Timing Model



The timing arcs between pin A and pin F are described in the following TLF code:

```
// timing arc (1)
Path( A => F 01 10 COND(B) SDF_COND(B == 1'b1) DELAY(ioDelayFallModel0)
SLEW(SlopeFallModel0) )
Path( A => F 10 01 COND(B) SDF_COND(B == 1'b1) DELAY(ioDelayRiseModel2)
SLEW(SlopeRiseModel2) )
// timing arc (2)
Path( A => F 10 01 COND(~B) SDF_COND(B == 1'b0) DELAY(ioDelayRiseModel1)
SLEW(SlopeRiseModel1) )
```

Assume the SDF description of instance `I1` of the library cell `FA` shown in Example 7-1, where the SDF description matches the TLF description.

Example 7-1 SDF Description Matches the TLF Description

```
(CELL ( CELLTYPE "FA" )
(INSTANCE I1 )
   (DELAY
      (ABSOLUTE
       ( COND B == 1'b1 (IOPATH A  F  (1.5:1.5:1.5) (3:3:3) ) )// SDF assertion (1)
        (COND  B == 1'b0 (IOPATH A  F (1:1:1) () ) ) // SDF assertion (2)
       )
   )
)
```

SDF assertion (1) is matched with timing arc (1) and SDF assertion (2) is matched with timing arc (2), therefore, you do not need to use the `-ignore_sdf_cond` or the `-worst_case_mismatched_conds` options.

Example 7-2 shows a SDF description where you need to use the `-worst_case_mismatched_conds` option.

Example 7-2 SDF Description Using the -worst_case_mismatched_conds Option

```
(CELL ( CELLTYPE “FA” )
(INSTANCE I1 )
   (DELAY
      (ABSOLUTE
       ( COND B == 1’b1 (IOPATH A  F  (1.5:1.5:1.5) (3:3:3) ) )// SDF assertion (1)
        (COND dummyCond (IOPATH A  F (1:1:1) (2:2:2) ) ) // SDF assertion (2)
        (IOPATH A  F (0.8:0.8:0.8) (1.2:1.2:1.2) ) ) // SDF assertion (3)
       )
   )
)
```

SDF assertion (1) is matched with timing arc (1). However, SDF assertions (2) and (3) cannot be matched with a timing arc, because these conditions do not match a timing arc condition. *Default*: An error message is issued and the delays from these SDF assertions are discarded. If the -`worst_case_mismatched_conds` option is specified, the condition SDF assertion (2) is ignored, and this assertion is merged with SDF assertion (3). Both assertions are matched with the timing arc (2).

Using the -`ignore_sdf_cond` option will not have any affect as this option will not change the default behavior.

Assume that the timing arcs are described by the following TLF code:

```
// timing arc (1)
Path( A => F 01 10 COND(B) SDF_COND(B == 1’b1) DELAY(ioDelayFallModel0)
SLEW(SlopeFallModel0) )
// timing arc (2)
Path( A => F 10 01 COND(~B) SDF_COND(B == 1’b0) DELAY(ioDelayRiseModel1)
SLEW(SlopeRiseModel1) )
// timing arc (3)
Path( A => F 10 01 DELAY(ioDelayRiseModel2) SLEW(SlopeRiseModel2) )
Path( A => F 01 10 DELAY(ioDelayFallModel2) SLEW(SlopeFallModel2) )
```

Assume now that the SDF description of instance `I1` of the library cell FA, after specifying the -`ignore_sdf_cond` option, shown in Example 7-3.

Example 7-3 SDF Description After Using the -ignore_sdf_cond Option

```
(CELL ( CELLTYPE “FA” )
(INSTANCE I1 )
   (DELAY
      (ABSOLUTE
        ( COND B == 1’b1 (IOPATH A  F () (3:3:3) ) )// SDF assertion (1)
```

```
        (COND  B == 1'b0 (IOPATH A  F (1:1:1) () ) ) // SDF assertion (2)
        (COND Dummy (IOPATH A  F (1.3:1.3:1.3) (2:2:2) ) ) // SDF assertion (3)
       )
   )
)
```

SDF assertion (1) is matched with timing arc (1) and SDF assertion (2) is matched with timing arc (2). SDF assertion (3) cannot be matched to any timing arc, because its condition does not match any timing arc condition.
Default: An error message is issued and the delay from SDF assertion (3) is discarded. However, if the -ignore_sdf_cond option is specified, condition SDF assertion (3) is assumed to be a dummy condition and is ignored. The SDF assertion (3) is matched with the timing arc (3).

The -worst_case_mismatched_conds option has the same effect as the -ignore_sdf_cond in this example. However if you want to allow the mismatches shown in this example, but catch the type of mismatches shown in Example 7-2, then use the -ignore_sdf_cond option.

Related Information

read_adb

read_alf

read_pdef

read_verilog

read_vhdl

read_spef

`read_spef [-instance` *`instance_name`*`] [-reduce_for_user_pvt] [-verbose]` *`file_name`*

Lets you read in a parasitics file in Standard Parasitics Exchange Format (SPEF). The SPEF file can be in detailed (DSPEF) or reduced (RSPEF) form. Internally a detailed SPEF is reduced. The reduced file is stored in the database and used for delay calculation.

If the OLA library implements all post-layout APIs, CTE passes each RLC network to the OLA library for reduction to obtain the pi model for each driver pin and to get the poles and residues for each net segment in the netlist.

Read in a compressed SPEF file in GNU zip format if the specified input file ends in the `.gz` suffix. No special option is needed to read a gzipped file. For example: `read_spef` *`spef_filename`*`.gz`. However, you cannot create a gzipped parasitics file with the `read_spef` command.

For flow information and details about delay calculation with parasitics, see Reading Parasitics File Formats in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Options and Arguments

`file_name`
: Specifies the name of the SPEF file to be read.

`-instance` *`instance_name`*
: Reads multiple parasitics files associated with different logical or physical hierarchical blocks. The SPEF file being read is associated with the instance specified by *`instance_name`*.

`-reduce_for_user_pvt`
: Reduces detailed parasitics only for the corner specified with the `pvt_early_path` and `pvt_late_path` globals. This option makes runtime faster.

`-verbose`
: Performs various checks such as resistance loops in detailed parasitics and the completion of parasitics specified and reports them.

Example

The following performs various checks such as resistance loops in detailed parasitics and the completion of parasitics specified and reports them:

```
read_spef -verbose spiffy.spef
```

Related Information

read_spf

write_spf

read_spf

```
read_spf [-instance instance_name] [-reduce_for_user_pvt]
     [-left_bus_delimiter char][-right_bus_delimiter char] [-verbose] file_name
```

Reads in a parasitics file in Standard Parasitics Format (SPF). The SPF file can be in detailed (DSPF) or reduced (RSPF) form. Internally a detailed SPF is reduced. The reduced file is stored in the database and used for delay calculation. Write out the reduced file using the `write_spf` command.

Read in a compressed SPF file in GNU zip format if the specified input file ends in the .gz suffix. No special option is needed to read a gzipped file. For example: `read_spf` *`spf_filename`*`.spf.gz`. However, you cannot create a gzipped file with the `read_spf` command.

For flow information and details about delay calculation with parasitics, see Reading Parasitics File Formats in *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Options and Arguments

`file_name`
: Specifies the name of the SPF file to read.

`-instance` *`instance_name`*
: Reads multiple parasitics files associated with different logical or physical hierarchical blocks. The SPF file being read is associated with the instance specified by *`instance_name`*.

`-left_bus_delimiter` *`char`* `-right_bus_delimiter` *`char`*
: Specifies the left bus delimiter and the right bus delimiter. Both of these options are required for SPF versions 1.3 and older. These options are ignored for SPF versions newer than 1.3.

`-reduce_for_user_pvt`
: Reduces detailed parasitics only for the corner specified with the `pvt_early_path` and `pvt_late_path` globals. This option makes runtime faster.

`-verbose`
: Performs various checks such as resistance loops in detailed parasitics and the completeness of parasitics data and reports them. This option lets you check the correctness of the parasitics file.

Examples

- The following command reads in a version 1.4 detailed SPF file named `new.dspf`:

  ```
  read_spf new.dspf
  ```

- The following command reads in a version 1.3 reduced SPF file named `old.rspf`:

  ```
  read_spf old.rspf -left_bus_delimiter \[ -right_bus_delimiter \]
  ```

Tip

Some delimiter characters like square brackets and curly braces must be escaped characters (preceded by \) for proper Tcl interpretation. See above example.

Related Information

`read_spef`

`write_spf`

read_stamp

```
read_stamp (-model model_file model_data_file) |
     ({[-min min_model_data_file] | [-typ typ_model_data_file] |
     [-max max_model_data_file]} -model model_file [-name merged_name])
```

Reads in a timing model in the Synopsys Stamp format. Use this command in two ways:

- The first usage associates one model data file with the model. In this case, the model data file is assumed to contain `typ` data:

  ```
  read_stamp -model model_file model_data_file
  ```

- The second usage lets you associate data from one or more model data files to the same model:

  ```
  read_stamp {[-min min_model_data_file] | [-typ typ_model_data_file] |
  [-max max_model_data_file]} -model model_file [-name merged_name]
  ```

Options and Arguments

`-max` *`max_model_data_file`*
: Specifies the name of the `max` data file associated with the Stamp model.

`-min` *`min_model_data_file`*
: Specifies the name of the `min` data file associated with the Stamp model.

`model_data_file`
: Specifies the name of the data file associated with the Stamp model. This model data file is assumed to contain `typ` data.

`-model` *`model_file`*
: Specifies the name of the Stamp model to load.

`-name` *`merged_lib_name`*
: Changes the new merged library name to *`merged_lib_name`*.

`-typ` *`typ_model_data_file`*
: Specifies the name of the `typ` data file associated with the Stamp model.

Example

The following command loads the data from `cpu.data` into the typical field of the library:

```
read_stamp -mod cpu.mod cpu.data
```

Related Information

`reset_functional_mode`

`report_functional_mode`

`set_functional_mode`

Using Synopsys Stamp Models in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

read_tlf

```
read_tlf [-force] [-silent] [-password password] {{-min filename -max filename
     -typ filename [-name merged_filename]} | tlf_library_filename |
     {list_of_tlf_filenames}}
```

Reads in an encrypted (.etlf) or unencrypted (.tlf) Timing Library Format (TLF) library format. TLF version 4.3 or higher is supported. For information about the latest version supported, see "Using Timing Library Format (TLF) Libraries" in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Read in a compressed TLF file in GNU zip format if the specified input file ends in the .gz suffix. No special option is needed to read a gzipped file. For example: `read_tlf tlf_filename.tlf.gz`. However, you cannot create a gzipped file with the `read_tlf` command.

Use this command in two ways.

- To read a TLF library into the database, use this form:

  ```
  read_tlf [-silent] [-password password] tlf_library_filename
  ```

  If the library has complete triplets of data (min/typ/max values), then all three fields are loaded simultaneously.

- To create and read a merged TLF library from multiple single operating point libraries, use this form:

  ```
  read_tlf [-silent] [-password password]
  {{-min filename -max filename -typ filename -name merged_filename}}
  ```

  Use the `-min`, `-typ`, and `-max` options to create a merged library from libraries containing one or more operating point data. The libraries to be merged must contain data in the appropriate operating point field or the `typ` field. For example, if the library for the `-min` (`-max`) option contains multiple operating point data, it must contain data in the `min` (`max`) or `typ` field. The merged library is read into the database immediately. See Example on page 937.

  The values from each library are scaled to the current session units using the units from that library.The first library `read_` command will provide the default session units. The library for the `max` corner is used for the default units if it is present, otherwise the `typ` corner is used, or finally, the `min` corner.

  Use the `-name` option to change the name of the merged library to a new name. For example, `-name merged_filename` changes the merged library name to `merged_filename`.

  Note: When performing crosstalk analysis using the `do_analyze_crosstalk -engine celtic` command, CeltIC™ does not support

the signal integrity (SI) analysis of two mixed corner-case timing libraries. For example:

```
read_tlf -min $rootDir/links/tsmc13_ff.tlf -max $rootDir/links/tsmc13_ss.tlf
-name tsmc13
```

CeltIC only supports the use of single corner-case timing libraries each time SI is performed.

Options and Arguments

`-force`

Overrides invalid 0-100 percent measured threshold values. If you do not use this option, CTE will not read libraries that define 0-100 percent threshold values. The measured delay threshold value is 50-50, while the measured slew threshold is 40-60. Also lets older TLF versions lower than 4.3 be accepted.

`list_of_tlf_filenames`

Allows reading multiple tlf library filenames. Use curly braces to specify multiple library filenames, for example:
`{lib1 lib2 lib3}`

`-max` *`filename`*

Uses `max` data from the library *`filename`* to populate the `max` field of the merged library. If the `max` data is not present, the `typ` data is used. This option is required when using the `-name` option.

`-min` *`filename`*

Uses `min` data from the library *`filename`* to populate the `min` field of the merged library. If the `min` data is not present, the `typ` data is used. This option is required when using the `-name` option.

`-name` *`merged_filename`*

Changes the name of the merged library to *`merged_filename`*. This option requires the use of the `-min`, `-typ`, `-max` options.

`-password` *`password`*

Passes the key *`password`* for an encrypted TLF library.

`-silent`

Suppresses the message which says that the library has been successfully loaded.

`tlf_library_filename`

Specifies the name of the tlf library filename.

`-typ` *`filename`*

Uses `typ` data from the library *`filename`* to populate the `typ` field of the merged library. This option is required when using the `-name` option.

Example

- The following command picks up `min` values from the `min.tlf` library and populates the `min` fields of the merged library with name `merged_lib`. If `min` values are not present in `min.tlf` then `typ` values are used. The `typ` values are obtained from the `typ.tlf` library. Likewise `max` values of the merged library are obtained from the `max.tlf` library. If `max` values are not present in `max.tlf,` then `typ` values are used:

  ```
  read_tlf -min min.tlf -typ typ.tlf -max max.tlf -name merged.tlf
  ```

  Suppose `min.tlf` contains `(0.01:0.02:0.03)`as the (*`min`*`:`*`typ`*`:`*`max`*`)` pin-to-pin delay for some cell, `typ.tlf` contains `(0.04:0.05:0.06)`for the corresponding delay, and `max.tlf` contains `(0.07:0.08:0.09)`, the resultant triplet in `merged.tlf` is `(0.01:0.05:0.09)`.

Related Information

`read_alf`

`read_dotlib`

"Using Timing Library Format (TLF) Libraries" in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

remove_assertions

```
remove_assertions [-bidi_input] [-bidi_output]
     [-type {input_delay | external_delay | arrival | required
     | clock_root | clock_insertion_delay | clock_info_change }]
     list_of_pins
```

Removes all (or specified) assertions on the given list of pins. This command removes assertions on a pin in a more general way than the `reset_*` commands.

The impact on the database is to remove the assertions; there is no change to the netlist.

By contrast, the `reset_*` commands provide specific ways to remove assertions. The rules for resetting an assertion are the same for setting one. The following example shows the difference between `reset_*` assertion and `remove_assertions`. First, the command below sets a late `external_delay` assertion on pin `out` with a clock `CLK`:

```
set_external_delay -clock CLK out 0
```

Then the following command resets the late `external_delay` assertion on pin `out`:

```
reset_external_delay -clock CLK out
```

However, this command would not:

```
reset_external_delay out
```

The original assertion is not reset because the `-clock` option is not specified (the default clock is `@`, which is different from `CLK`). To remove general assertions, use the following command:

```
remove_assertions -type external_delay out
```

The above command removes all `external_delay` assertions on pin `out`, regardless of rise or fall, early or late, or clock.

Note: Currently, the `remove_assertions` command does not remove assertions that are set on waveforms. Use the `reset_*` commands to remove assertions from waveforms. For example, `reset_clock_uncertainty` or `reset_clock_insertion_delay`.

Options and Arguments

`list_of_pins`
: Specifies the pins where the assertion is to be removed.

`-type` *`assertion_type`*
: Removes only the assertion types specified. When no `-type` option is provided, the default is to remove all assertions listed

below. The following are the allowable *assertion_type* values:

arrival
Removes assertions previously given by the set_data_arrival_time command. Also removes assertions given by the obsolete set_clock_arrival_time command.

clock_arrival_time
Removes only assertions given by the obsolete set_clock_arrival_time command.

clock_gating_check
Removes only assertions previously given by the set_clock_gating_check command.

clock_info_change
Removes only assertions previously given by the set_clock_info_change command.

clock_insertion_delay
Removes only assertions previously given by the set_clock_insertion_delay command for both source and network pins. Waveform insertion delays are not removed.

clock_required_time
Removes only assertions given by the rarely used set_clock_required_time command.

clock_root
Removes only assertions previously given by the set_clock_root command.

clock_uncertainty
Removes only assertions previously given by the set_clock_uncertainty command. Waveform uncertainty is not removed.

constant_for_timing
Removes only assertions previously given by the set_constant_for_timing command.

data_arrival_time
Removes only assertions previously given by the set_data_arrival_time command.

data_required_time
Removes only assertions previously given by the set_data_required_time command.

drive_cell
Removes only assertions previously given by the set_drive_cell command.

drive_resistance
Removes only assertions previously given by the set_drive_resistance command.

external_delay
Removes only assertions previously given by the set_external_delay command.

fanout_load
Removes only assertions previously given by the set_fanout_load_limit command.

fanout_load_limit
Removes only assertions previously given by the set_fanout_load_limit command.

generated_clock
Removes only assertions previously given by the set_generated_clock command.

num_external_sinks
Removes only assertions previously given by the set_num_external_sinks command.

num_external_sources
Removes only assertions previously given by the set_num_external_sources command.

port_capacitance
Removes only assertions previously given by the set_port_capacitance command.

port_capacitance_limit
Removes only assertions previously given by the set_port_capacitance_limit command.

port_wire_load
Removes only assertions previously given by the set_port_wire_load command.

required
Removes assertions previously given by the set_data_required_time command. Also removes assertions given by the rarely used set_clock_required_time command.

slew_time
Removes only assertions previously given by the set_slew_time command.

slew_time_limit
Removes only assertions previously given by the set_slew_time_limit command.

time_borrow_limit
Removes only assertions previously given by the set_time_borrow_limit command.

Examples

- The following command writes the existing assertions, then removes some, and writes a new assertions file:

```
write_assertions assert.txt
cat assert.txt
set_current_module {top}
set_top_timing_module {top}
set_clock_propagation ideal
set_input_delay -clock clkA -lead -early -rise   0.000  {in}
set_external_delay -clock clkA -lead -late -rise   0.000  {out}
#end of assertions for module top
```

- The following command removes assertions on ports in and out:

```
remove_assertions [find -port in out]
write_assertions assert2.txt
```

```
cat assert2.txt
set_current_module {top}
set_top_timing_module {top}
set_clock_propagation ideal
#end of assertions for module top
```

Related Information

reset_clock_info_change

reset_clock_gating_check

reset_clock_root

reset_clock_uncertainty

reset_constant_for_timing

reset_drive_cell

reset_external_delay

reset_fanout_load

reset_fanout_load_limit

reset_generated_clock

reset_input_delay

reset_port_capacitance

reset_port_capacitance_limit

reset_port_wire_load

reset_num_external_sinks

reset_num_external_sources

reset_time_borrow_limit

set_clock_arrival_time

set_clock_required_time

report_analysis_coverage

```
report_analysis_coverage [-pins pin_list] [-check_type check_type_list]
     [-verbose check_status_list] [-sort {pin | refpin | checktype | slack |
     reason}] [-tcl_list] [{> | >> } file_name]

     Valid check types are clock_gating_hold clock_gating_pulse_width
     clock_gating_setup clock_period clock_separation external_delay hold
     no_change_hold no_change_setup path_delay pulse_width recovery removal setup
     skew time_borrow
     Valid status types are: met untested violated
```

Provides information about the timing checks in the design. For each type of timing check, the command reports the number of checks that meet constraints, violated constraints, or that are untested in the SUMMARY section. The DETAILS section includes information about the signal pin, reference pin, check type, slack, and the reason for being untested.

A `met` check shows as a positive value in the `Slack` column. A `violated` check shows as a negative value in the `Slack` column. An `untested` check shows as `UNTESTED` in the `Slack` column and the reason is given in the `Reason` column. The `Reason` column is blank when a `Slack` number is given because the check has been tested.

Note: If there are no timing checks in the design of the type specified by `-check_type`, the report shows “No timing checks found.”


Tip

Use `check timing` to look for missing assertions.

Options and Arguments

`{> | >>} filename`
: Stores the generated report in the file specified by *`filename`*. If no file name is specified, the report is displayed on standard output. The file name must be the last argument in the list.

`-check_type check_type_list`
: Limits the report to the type of check(s) in the *`check_type_list`*.

  Valid check types are: `setup, hold, pulse_width, clock_period, clock_gating_setup, clock_gating_hold, clock_gating_pulse_width, recovery, removal, clock_separation, skew, no_change_setup, no_change_hold, time_borrow,`

`external_delay, and path_delay.`
Default: All `check_types` are reported.

`-pins` *`pin_list`*
Limits the report to timing checks on pins in *`pin_list`*.

`-sort {pin | refpin | checktype | slack | reason}`
While reporting detailed information (`-detail` option), checks are sorted by the specified method.
Default: The checks are sorted by `pin`. This option requires the `-detail` option.

The `-sort` option does not change the order of the columns. When you sort by `slack`, the slack value is reported in order of least negative to most negative (worst slack last).

 *Tip*

Improve usability of the report for debugging purposes by using the `-detail` and `-sort` options as shown in these examples:

```
report_analysis_coverage -detail untested -sort reason
report_analysis_coverage -detail violated -sort slack
```

`-tcl_list`
Produces a report in a tcl list, not a human-readable report. This is useful for integrating timing with custom Tcl functions and also for customizing report generation.

`-verbose` *`check_status_list`*
Limits the detailed information to only the checks with the status specified.
Default: The details are shown regardless of met, violated, or untested status. A default report is given in the Examples section.

Valid status types are: `met, violated,` and `untested.`

Note: The `-verbose` option replaces the `-detail` option, which is obsolete.

Tip

Reduce the number of lines in the `DETAILS` section and improve usability of the report by using the `-detail` option to filter out the `met` checks as shown in this example:

```
report_analysis_coverage -detail {violated untested}
```

Example

The following command provides information about the timing checks in the design:

```
 report_analysis_coverage
+-----------------------------------------------+
|         TIMING CHECK COVERAGE SUMMARY         |
|-----------------------------------------------|
| Check | No. of |   Met   | Violated | Untested |
|  Type | Checks |         |          |          |
|-------+--------+---------+----------+----------|
| Hold  | 7      | 2 (28%) | 1 (14%)  | 4 (58%)  |
| Setup | 7      | 4 (58%) | 0 (0%)   | 3 (42%)  |
+-----------------------------------------------+
+-------------------------------------------------------------------------+
|                     TIMING CHECK COVERAGE DETAILS                       |
|-------------------------------------------------------------------------|
|   Pin    | Reference | Check |  Slack   |          Reason              |
|          |    Pin    |  Type |          |                              |
|----------+-----------+-------+----------+------------------------------|
| reg1/D   | reg1/CP ^ | Setup | UNTESTED | No data signal               |
| reg1/D   | reg1/CP ^ | Hold  | UNTESTED | No data signal               |
| reg2/D   | reg2/CP ^ | Setup | UNTESTED | No data signal               |
| reg2/D ^ | reg2/CP ^ | Hold  | 1.05     |                              |
| reg2/D v | reg2/CP ^ | Hold  | UNTESTED | Data signal with @  clock    |
| reg3/D   | reg3/CP ^ | Hold  | UNTESTED | Data signal with @  clock    |
| reg3/D ^ | reg3/CP ^ | Setup | 1.86     |                              |
| reg3/D v | reg3/CP ^ | Setup | 1.90     |                              |
| reg4/D ^ | reg4/CP ^ | Hold  | 0.05     |                              |
| reg4/D ^ | reg4/CP ^ | Setup | 2.86     |                              |
| reg4/D v | reg4/CP ^ | Hold  | -0.07    |                              |
| reg4/D v | reg4/CP ^ | Setup | 2.90     |                              |
| reg5/D   | reg5/CP   | Setup | UNTESTED | No reference signal          |
| reg5/D   | reg5/CP   | Hold  | UNTESTED | No reference signal          |
+-------------------------------------------------------------------------+
```

Related Information

`check_timing`

`report_timing`

`set_table_style`

Note: Setting the `clock_gating_regardless_of_downstream_logic` global can effect timing related report commands. For example, the `report_analysis_coverage` command may or may not report clock gating checks.

report_annotated_check

```
report_annotated_check [-missing_delays] [-missing_setup] [-missing_hold]
     [-missing_recovery] [-missing_removal] [-missing_no_change]
     [-missing_period] [-missing_skew] [-missing_mpw] [-max_missing integer]
     [-tcl_list]
```

Reports coverage of annotated timing checks. To limit the search for annotations to a specific area of a design, use the `set_current_module` and `set_current_instance` commands before using the `report_annotated_check` command.

Options and Arguments

`-max_missing` *`integer`*
: Limits the number of missing annotations to the number specified.
Default: 1000

`-missing_delay`
: Reports all the timing checks that are missing SDF annotations, in addition to the total coverage report.

`-missing_hold`
: Provides a detailed report of the missing annotations on hold timing checks.

`-missing_mpw`
: Provides a detailed report of the missing annotations on both mpwl and mpwh timing checks

`-missing_no_change`
: Provides a detailed report of the missing annotations on the `no_change` timing checks.

`-missing_period`
: Provides a detailed report of the missing annotations on period timing checks.

`-missing_recovery`
: Provides a detailed report of the missing annotations on recovery timing checks.

`-missing_removal`
: Provides a detailed report of the missing annotations on removal timing checks

`-missing_setup`
: Provides a detailed report of the missing annotations on setup timing checks.

`-missing_skew`
: Provides a detailed report of the missing annotations on skew timing checks.

`-tcl_list`
: Formats the information for use in a Tcl program.

Examples

- The following command reports missing annotated setups in SDF:

  ```
  report_annotated_check -missing_setup
  ```

- The following command reports all the missing annotated timing checks in SDF:

  ```
  report_annotated_check -missing_delays
  ```

Related Information

`set_current_module`

`set_current_instance`

report_annotations

```
report_annotations [-missing_resistances] [-missing_capacitances] [-missing_rc]
     [-missing_delays] [-missing_spf] [-max_missing integer] [-tcl_list]
     [-ignore_floating_nets] [-ignore_tied_low_high_nets]
```

Reports the coverage of the annotations on a design. These annotations include SDF delay arc annotations, SPF annotations, wire resistances, and wire capacitances.

Limit the search for annotations to a specific area of the design by specifying the `set_current_module` and `set_current_instance` commands before issuing the `report_annotations` command.

Use the `-missing_*` options to get a detailed report of all delay arcs and wires that are missing annotations. You can use the information in a Tcl program.

Options and Arguments

`-ignore_floating_nets`
: Ignores nets without drivers.

`-ignore_tied_low_high_nets`
: Ignores timing arcs of power and ground nets.

`-max_missing` *`integer`*
: Limits the number of missing annotations to the number specified.
Default: 1000

`-missing_capacitances`
: Reports the nets that are missing capacitances.
Default: Only a summary of the coverage is reported.

`-missing_delays`
: Reports the arcs that are missing SDF annotations.
Default: Only a summary of the coverage is reported.

`-missing_rc`
: Reports the nets that are missing resistances and capacitances.
Default: Only a summary of the coverage is reported.

`-missing_resistances`
: Reports the nets that are missing resistances.
Default: Only a summary of the coverage is reported.

`-missing_spf`

Reports the nets that are missing SPF data.
Default: Only a summary of the coverage is reported.

`-tcl_list`

Formats the information for use in a Tcl program.

Examples

- The following command reports the nets that are missing capacitances:

  ```
  report_annotations -missing_capacitances
  ```

  Default: Only a summary of the coverage is reported.

- The following commands show the difference between the summary and the `-missing_spf` reports:

  - `report_annotations`

    ```
    +-------------------------------------------------------------------------------------+
    | Total |  Total C  | Percentage |  Total R  | Percentage | Total SPF | Percentage |
    |  Nets | Annotated|     C      | Annotated |     R      | Annotated |    SPF     |
    |       |          |  Annotated |           |  Annotated |           |  Annotated |
    |-------+----------+------------+-----------+------------+-----------+------------|
    |   21  |        0 |   0.000000 |         0 |   0.000000 |        15 |   0.714286 |
    +-------------------------------------------------------------------------------------+
    +---------------------------------+
    | Total |   Total   | Percentage |
    |  Arcs |   Arcs    |    Arcs    |
    |       | Annotated |  Annotated |
    |-------+-----------+------------|
    |   272 |         0 |   0.000000 |
    +---------------------------------+
    ```

  - `report_annotations -missing_spf`

    ```
    +-------------------------------------------------------------------------------+
    |          |         Min          |         Typ          |         Max          | |
    |----------+----------------------+----------------------+----------------------|
    |  Net Name |   R |   C   |  SPF  |   R   |   C   |  SPF  |   R   |   C   |  SPF  |
    |----------+-------+-------+-------+-------+-------+-------+-------+-------+-------|
    |  bar/n019 |   - |   -   |   N   |   -   |   -   |   N   |   -   |   -   |   N   |
    |  bar/n018 |   - |   -   |   N   |   -   |   -   |   N   |   -   |   -   |   N   |
    |  bar/n017 |   - |   -   |   N   |   -   |   -   |   N   |   -   |   -   |   N   |
    +-------------------------------------------------------------------------------+
    ```

- If your SDF has only cell arcs, and net delays are included in the cell delays, you can report the status of annotation for cell arcs, internal net arcs, net arcs from primary inputs, and net arcs to primary outputs, and all timing arcs separately as shown below:

  ```
  report_annotations
  ```

  ```
  +-------------------------------------------------------------------+
  |  Delay Arc Type   |  Total  | Annotated |    Not    | Percentage |
  |                   |         |           | Annotated |  Annotated |
  ```

```
|-------------------+---------+-----------+-----------+------------|
|         Cell arcs |  551319 |    353103 |    198216 |  64.046951 |
|    Input net arcs |      18 |         7 |        11 |  38.888889 |
| Internal net arcs |  501156 |    304341 |    196815 |  60.727798 |
|   Output net arcs |     597 |       456 |       141 |  76.381912 |
+-------------------+---------+-----------+-----------+------------+
|   All timing arcs | 1053090 |    657907 |    395183 |  62.473957 |
+-----------------------------------------------------------------+
```

Related Information

report_net

report_cell_instance

```
report_cell_instance list_of_cell_instances [-timing | -power]
     [-early | -late] [-tcl_list] filename
```

Reports information about the timing context or power specifications of a library cell instance. *Default*: `-timing`.

Timing information on all pins of the instance are reported, followed by all the delay arcs between those pins.

Note: Internal pins (pins inside the instance boundary) are also reported.

The timing report contains three tables:

- The first table in the report contains information about the pins of the instance.

  The propagated slew is the slew that is propagated to the pin, based on the slew propagation mode.

- The second table contains information about the arcs through the instance.

  The slew out is the slew calculated at the arc output from the arc input slew, using the timing model.

- The third table contains information about the timing checks, if present, on the cell.

  Only checks that are from the library are shown in this table. Automatic checks, such as clock gating setup and hold, are not reported.

Power information includes three tables showing the cell internal power, leakage power, and net power. You must have a license that includes low power synthesis to get this information.

The following table indicates the table names to be specified with the `set_table_style` command if you want to adjust the format of the tables generated with `report_cell_instance -power`.

Table Name for set_table_style	Corresponds to
`pa_inst_leak_pwr_rep_table`	leakage power table
`pa_inst_net_pwr_rep_table`	net power table
`pa_inst_int_pwr_rep_table`	first table for internal power
`pa_inst_int_pwr_rep_table1`	second table for internal power

Options and Arguments

`-early|-late`

Reports the early timing (data hold/clock setup) or late timing (data setup/clock hold).
Default: `-late`

`filename`

Specifies the name of the output file where the tables are printed out.
Default: `stdout`

`list_of_cell_instances`

Specifies a flat (non-hierarchical) cell instance or list of cell instances. Use curly braces (`{}`) to enclose the list and separate the instance names with white space.

`-power`

Reports information about the power specifications of a library cell instance, as shown in the Examples section below. Without the `-power` option, the timing information is reported. You must read in a TCF file before using this option.

`-tcl_list`

Produces the report in a Tcl list instead of the default table report. Use this option to integrate timing or power analysis with custom Tcl functions or to customize report generation.

`-timing`

Reports information about the timing context of a library cell instance. This is the default report. See the Examples section below.

Examples

- Reports information about the timing context of instance `reg01`:

```
report_cell_instance -timing reg01
+-------------------------------------------------------------+
|                    Instance reg01 of LD1                    |
+-------------------------------------------------------------+
|-------------------------------------------------------------|
|  Pin  | Propagated | Arrival | Required | Slack |   Phase  |
|       |    Slew    |         |          |       |          |
|-------+------------+---------+----------+-------+----------|
|   Q ^ |       0.09 |    0.66 |     1.78 |  1.12 | CLK(D)(P)|
```

```
|   Q v |       0.06 |    0.65 |     1.77 |  1.12 | CLK(D)(P)|
|  QN ^ |       0.02 |    0.66 |          |       | CLK(D)(P)|
|  QN v |       0.03 |    0.71 |          |       | CLK(D)(P)|
|   D ^ |       0.03 |    1.25 |     0.08 | -1.17 | CLK(D)(P)|
|   D v |       0.03 |    1.31 |     0.07 | -1.24 | CLK(D)(P)|
|   G ^ |       0.00 |    0.04 |     1.16 |  1.12 | CLK(C)(P)|
|   G v |       0.00 |    2.03 |     5.25 |  3.22 | CLK(C)(P)|
+-------------------------------------------------------------+
```

```
+-------------------------------------------------------+
|                  Instance reg01 of LD1                |
|-------------------------------------------------------|
|      Arc      |     Slew      |       |       |       |
|---------------+---------------+-------+-------+-------|
|  From |  To   |  In   |  Out  |  Load | Delay |  Phase  |
|-------+-------+-------+-------+-------+-------+---------|
|   D ^ |  QN v |  0.03 |  0.03 |       |  0.62 | CLK(D)(P) |
|   D v |  QN ^ |  0.03 |  0.02 |       |  0.59 | CLK(D)(P) |
|   D ^ |   Q ^ |  0.03 |  0.09 |  0.12 |  0.58 | CLK(D)(P) |
|   D v |   Q v |  0.03 |  0.07 |  0.12 |  0.57 | CLK(D)(P) |
|   G ^ |  QN ^ |  0.00 |  0.02 |       |  0.62 | CLK(C)(P) |
|   G ^ |  QN v |  0.00 |  0.03 |       |  0.67 | CLK(C)(P) |
|   G ^ |   Q ^ |  0.00 |  0.09 |  0.12 |  0.62 | CLK(C)(P) |
|   G ^ |   Q v |  0.00 |  0.06 |  0.12 |  0.61 | CLK(C)(P) |
+-------------------------------------------------------+
```

```
+-------------------------------------------------------------------------+
|                          Instance reg01 of LD1                          |
|-------------------------------------------------------------------------|
|       |     Pins      |       |       |        |       Phase           |
|-------+---------------+-------+-------+--------+-----------------------|
| Check |  Sig  |  Ref  | Slack | Delay | Adjust |    Sig    |    Ref    |
|-------+-------+-------+-------+-------+--------+-----------+-----------|
| HOLD  |   D ^ |   G v |  3.22 |  0.01 | 4.00   | CLK(D)(P) | CLK(C)(P) |
| HOLD  |   D v |   G v |  3.29 |  0.01 | 4.00   | CLK(D)(P) | CLK(C)(P) |
| SETUP |   D ^ |   G v |  0.73 |  0.05 | 0.00   | CLK(D)(P) | CLK(C)(P) |
| SETUP |   D v |   G v |  0.53 |  0.19 | 0.00   | CLK(D)(P) | CLK(C)(P) |
+-------------------------------------------------------------------------+
```

- If a constant is asserted or propagated to the instance pin, it is reported in the *Phase* column of the pin table:

```
report_cell_instance -timing xor2
```

```
+-------------------------------------------------------------------------+
|                          Instance xor2 of XOR2                          |
|-------------------------------------------------------------------------|
|  Pin  |  Dir  | Propagated | Arrival | Required | Slack | Phase        |
|       |       |    Slew    |         |          |       |              |
|-------+-------+------------+---------+----------+-------+--------------|
|   Y ^ |   OUT |       1.00 |    1.00 |     0.00 | -1.00 |  clk(C)(P)   |
|   Y v |   OUT |       1.00 |    6.00 |          |       |  clk(C)(P)   |
|   A ^ |    IN |       0.00 |    0.00 |    -1.00 | -1.00 |  clk(C)(P)   |
|   A v |    IN |       0.00 |    5.00 |          |       |  clk(C)(P)   |
|     B |    IN |            |         |          |       | Constant 0   |
+-------------------------------------------------------------------------+
```

```
+-------------------------------------------------------+
|                  Instance xor2 of XOR2                |
|-------------------------------------------------------|
|      Arc      |     Slew      |       |       |       |
|---------------+---------------+-------+-------+-------|
|  From |  To   |  In   |  Out  |  Load | Delay |  Phase  |
|-------+-------+-------+-------+-------+-------+---------|
```

```
|   A ^ |   Y ^ |  0.00 |  1.00 |  1.00 |  1.00 | clk(C)(P) |
|   A ^ |   Y v |  0.00 |       |  1.00 |       | clk(C)(P) |
|   A v |   Y ^ |  0.00 |       |  1.00 |       | clk(C)(P) |
|   A v |   Y v |  0.00 |  1.00 |  1.00 |  1.00 | clk(C)(P) |
|   B ^ |   Y ^ |       |       |  1.00 |       |           |
|   B ^ |   Y v |       |       |  1.00 |       |           |
|   B v |   Y ^ |       |       |  1.00 |       |           |
|   B v |   Y v |       |       |  1.00 |       |           |
 +-------------------------------------------------------------+
```

- In the power report, a “-” in the column implies that the value is not present for those fields.

```
report_cell_instance -power i_53 ex.rpt
+-----------------------------------------------------+
| Report                       | report_cell_instance  |
|------------------------------+-----------------------|
| Options                      | -power i_191 > ex.rpt |
+------------------------------+-----------------------+
| Date                         | 20021018.173603       |
| Tool                         | pks_shell             |
| Release                      | v5.0-s005             |
| Version                      | Oct 15 2002 15:12:07  |
+------------------------------+-----------------------+
| Module                       | AWDP_MULT_0           |
| Timing Operating Condition   | slow                  |
| Power Operating Library      | Typical               |
| Process                      | 1.000000              |
| Voltage                      | 1.620000              |
| Temperature                  | 125.000000            |
| time unit                    | 1.00 ns               |
| capacitance unit             | 1.00 pF               |
| power unit                   | 1.00mW                |
+-----------------------------------------------------+
```

Internal Cell Power for i_1/i_191									
Source Pin					Sink Pin				
name	dir	slew	cap	td	name	dir	slew	cap	td
B	^v	0.2002	-	2.655e-03	Y	v	-	5.729e-03	5.067e-03
B	^v	0.2002	-	2.655e-03	Y	^	-	5.729e-03	5.067e-03
B	^v	0.2002	-	2.655e-03	Y	v	-	5.729e-03	5.067e-03
B	^v	0.2002	-	2.655e-03	Y	^	-	5.729e-03	5.067e-03
A	^v	0.3145	-	2.412e-03	Y	v	-	5.729e-03	5.067e-03
A	^v	0.3145	-	2.412e-03	Y	^	-	5.729e-03	5.067e-03
A	^v	0.3145	-	2.412e-03	Y	v	-	5.729e-03	5.067e-03
A	^v	0.3145	-	2.412e-03	Y	^	-	5.729e-03	5.067e-03
-	-	-	-	-	-	-	-	-	-

```
+-------------------------------------------------------------------+
| Internal Cell Power for i_1/i_53 (cont.)                          |
|-------------------------------------------------------------------|
|       Other Pin       |  When |   effTD   |  arcPo |  effPower  |
|                       | Condi |           |  wer   |            |
|-----------------------+-------+-----------+--------+------------|
|  name |  slew |  cap  |       |           |        |            |
|-------+-------+-------+-------+-----------+--------+------------|
| -     | -     | -     | !(A)  | 2.008e-04 | 0.0405 | 8.129e-06  |
| -     | -     | -     | !(A)  | 2.008e-04 | 0.0235 | 4.726e-06  |
| -     | -     | -     | A     | 1.127e-03 | 0.0262 | 2.948e-05  |
| -     | -     | -     | A     | 1.127e-03 | 0.0386 | 4.354e-05  |
| -     | -     | -     | !(B)  | 2.650e-04 | 0.0277 | 7.332e-06  |
| -     | -     | -     | !(B)  | 2.650e-04 | 0.0116 | 3.085e-06  |
| -     | -     | -     | B     | 9.409e-04 | 0.0215 | 2.021e-05  |
| -     | -     | -     | B     | 9.409e-04 | 0.0300 | 2.822e-05  |
|                                           Total Power | 1.447e-04  |
+-------------------------------------------------------------------+

+-------------------------------------------------+
|              Leakage Power for                  |
|                  i_0/i_0                        |
|-------------------------------------------------+
|    When    |  When  |   Leak    |  Eff Leak     |
|  Condition |  Prob  |   Power   |   Power       |
|------------+--------+-----------+---------------+
| -          | 1.0000 | 1.458e-07 |  1.458e-07    |
|                     Total Power |  1.458e-07    |
+-------------------------------------------------+

+-----------------------------------------------------------+
|     Net Power for i_0/i_0                                 |
|-----------------------------------------------------------|
|    Name     |  Cap   |    TD     |  Prob  |  Power      |
|-------------+--------+-----------+--------+-------------|
| n_272       | 0.0000 | 2.412e-03 | 0.1059 | 0.0000      |
| n_201       | 0.0000 | 2.655e-03 | 0.2485 | 0.0000      |
| O0[10]      | 0.0057 | 5.067e-03 | 0.5026 | 3.809e-05   |
|                      Total Power | 3.809e-05           |
+-----------------------------------------------------------+
```

Related Information

get_timing

report_net

report_timing

set_global slew_propagation_mode

set_table_style

report_cell_instance_timing

```
report_cell_instance_timing
     This command is obsolete.Use the report cell instance -timing command
     instead.
```

report_clocks

```
report_clocks [-description] [-arrival_points] [-clock_to_data] [-data_to_clock]
     [-phase_shift_table] [-total_shift_table] [-uncertainty_table]
     [-adjustment_table] [-delay_adjustment_table] [-source_insertion]
     [-insertion] [-generated] [-clocks clk_signame | clk_signame_list]
     [-tcl_list] [{> | >>} filename]
```

Reports information about ideal (or generated) clocks and assertions with respect to those clocks. It reports information about all clock waveforms, clock arrival points, clock uncertainties, where clocks are converted to data, and where data is converted to clocks.

Use the `-description`, `-arrival_points`, `-clock_to_data`, `-data_to_clock`, `-uncertainty_table`, and the `*_table` options to specify different parts of the report. If one of these options is supplied, only the section that the option corresponds to is reported. If other options are supplied, those sections are also reported.

Use the `-tcl_list` option to format the information for use in a Tcl program.

Tip

The `report_clocks` command only reports assertions with respect to clock waveforms. Use the `report_port -type` command for assertions applied to clock pins (except for the `set_clock_root` command, which can be reported by either command). If you use the `set_clock_info_change` command, use the `report_port -type arrival` command, not the `report_clocks -arrival` command. For more information, see `report_ports`.

Options and Arguments

`{> | >>} filename`

Specifies the name of the file in which the report is saved. The *`filename`* argument must be the last argument in the list. If the file name is not specified, the report displays on the standard output without being saved. Since the report is often large, text may be lost if you do not save the report to a file.

`-adjustment_table`

Reports two tables that detail the early and late cycle adjustments between clock waveforms as set by the `set_cycle_addition` command.

Note: If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.

`-arrival_points`

Reports only the clock arrival points. A clock arrival point is the pin where the `set_clock_root` or the `set_clock_arrival_time` command has been asserted.

The data is reported differently if the `set_clock_root` command is used instead of the `set_clock_arrival_time` command. If the `set_clock_arrival_time` command is used, the early/late rise/fall fields are displayed along with the Ideal (or generated) clock name and the place where the clock is attached. If the `set_clock_root` command is used, only the Ideal clock name and the attachment point are displayed; the rise/fall early/late fields are left empty.

`-clock_to_data`

Reports only the clock to data change points. A clock to data change point is where the `set_clock_info_change` command has been used to convert any clock signals to data signals.

`-clocks` *`clock_name`*`|`*`clock_name_list`*

Requests information on a specific ideal (or generated) clock. If a list is given, only information about those clocks is reported. You can use wildcards in the names.
Default: All ideal clocks are reported. Use the `-generated` option to also report generated clocks.

`-data_to_clock`

Reports only the data-to-clock change points. A data-to-clock change point is where the `set_clock_info_change` command has been used to convert any data signals to clock signals.

`-delay_adjustment_table`

Reports two tables that detail the early and late path delay adjustments between clock waveforms as set by the `set_path_delay_constraint` command with `-clock_from` `-clock_to` options.

Note: If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.

`-description`

Reports only the description of the specified clock waveform(s).

`-generated`

Reports generated clocks created by the `set_generated_clock` command in addition to the ideal clocks. The report uses the internal clock names created by the system, if you did not specify new clock names while creating these clocks. See Examples on page 962.

`-insertion`

Reports the network insertion delays specified on clock waveforms. The values in the table are in `min:typ:max` format.

`-phase_shift_table`

Reports two tables that detail the phase shift between clock waveforms. One table shows the phase shifts in late mode and the other in early mode.

Note: If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.

`-source_insertion`

Reports the source insertion delays specified on clock waveforms.

`-tcl_list`

Produces the report in a Tcl list, not in a readable report. This is useful for integrating timing with custom Tcl functions and also for customizing report generation.

`-total_shift_table`

Reports two tables (late mode and early mode) that detail the total relationship between clock waveforms.

Note: If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.

`-uncertainty_table`

Reports two tables (late mode and early mode) that detail the uncertainty between clock waveforms as specified by the `set_clock_uncertainty` command.

Note: If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.

Examples

- The following command displays the default report:

```
report_clocks
```

```
+--------------------------------+
|       Clock Descriptions       |
|--------------------------------|
| Clock | Period |  Lead | Trail |
|  Name |        |       |       |
|-------+--------+-------+-------|
|   A   |  15.00 |  0.00 |  7.50 |
|   B   |  10.00 |  2.00 |  8.00 |
+--------------------------------+
+-----------------------------------------------+
|       Total Clock To Clock Relationship       |
|-----------------------------------------------|
|  From |  To   |  L->L |  L->T |  T->L |  T->T |
|-------+-------+-------+-------+-------+-------|
|     A |     A | 14.76 | -0.48 | 14.28 | 14.04 |
|     A |     B | -1.20 | -6.44 |  8.32 | -1.92 |
|     B |     A |  2.84 | -7.40 |  7.36 |  2.12 |
|     B |     B |  6.88 | -3.36 |  6.40 |  6.16 |
+-----------------------------------------------+
```

- The following command displays only the uncertainty information:

```
report_clocks -uncertainty_table
```

```
+-----------------------------------------------+
|       Clock To Clock Uncertainty (Late)       |
|-----------------------------------------------|
|  From |  To   |  L->L |  L->T |  T->L |  T->T |
|-------+-------+-------+-------+-------+-------|
| iclk1 | iclk1 |  0.00 |  0.00 |  0.00 |  0.00 |
| iclk1 | iclk2 |  1.70 |  1.70 |  1.70 |  1.70 |
| iclk2 | iclk1 |  0.00 |  0.00 |  0.00 |  0.00 |
| iclk2 | iclk2 |  0.00 |  0.00 |  0.00 |  0.00 |
+-----------------------------------------------+

+-----------------------------------------------+
|      Clock To Clock Uncertainty (Early)       |
|-----------------------------------------------|
|  From |  To   |  L->L |  L->T |  T->L |  T->T |
|-------+-------+-------+-------+-------+-------|
| iclk1 | iclk1 |  0.00 |  0.00 |  0.00 |  0.00 |
| iclk1 | iclk2 | -2.70 | -2.70 | -2.70 | -2.70 |
| iclk2 | iclk1 |  0.00 |  0.00 |  0.00 |  0.00 |
```

```
| iclk2 | iclk2 |  0.00 |  0.00 |  0.00 |  0.00 |
+-----------------------------------------------+
```

- The following command displays a report from a design with a generated clock (created in the example for `get_clock`):

```
report_clocks -generated
+-----------------------------------+
|         Clock Descriptions        |
|-----------------------------------|
|  Clock   | Period |  Lead |  Trail |
|   Name   |        |       |        |
|----------+--------+-------+--------|
|   CLK    | 10.000 | 0.000 |  5.000 |
| my_clock | 20.000 | 0.000 | 10.000 |
+-----------------------------------+
+-----------------------------------------------------------+
|        Total Clock To Clock Relationship (Late)           |
|-----------------------------------------------------------|
|   From   |    To    |  L->L  |  L->T |  T->L  |  T->T  |
|----------+----------+--------+-------+--------+--------|
|      CLK |      CLK | 10.000 | 0.000 | 10.000 | 10.000 |
|      CLK | my_clock | 10.000 | 0.000 | 10.000 |  0.000 |
| my_clock |      CLK | 10.000 | 0.000 | 20.000 | 10.000 |
| my_clock | my_clock | 20.000 | 0.000 | 20.000 | 20.000 |
+-----------------------------------------------------------+
+-------------------------------------------------------------+
|        Total Clock To Clock Relationship (Early)            |
|-------------------------------------------------------------|
|   From   |    To    |  L->L |  L->T   |  T->L  |  T->T   |
|----------+----------+-------+---------+--------+---------|
|      CLK |      CLK | 0.000 | -10.000 |  0.000 |   0.000 |
|      CLK | my_clock | 0.000 | -10.000 |  0.000 | -10.000 |
| my_clock |      CLK | 0.000 | -10.000 | 10.000 |   0.000 |
| my_clock | my_clock | 0.000 | -20.000 |  0.000 |   0.000 |
+-------------------------------------------------------------+
```

Related Information

`get_clock`

`set_clock_info_change`

`set_clock_insertion_delay`

`set_clock_root`

`set_clock_uncertainty`

`set_generated_clock`

`set_table_style`

report_fanin

```
report_fanin [-level integer] [-tristate] [-sequential] [-hierarchical]
     [{> | >>} filename] [-maxline integer] list_of_pin_path_or_id
```

Generates a report on the fanin cone of the specified design objects in the current module. The report generated contains all the pertaining information on fanin cone, for example, from pin, to pin, level in the module and the module. All pins and their associated nets on the fanin path are reported with hierarchical names (starting at the current module).

Options and Arguments

`-hierarchical`
: Allows the command to traverse backwards across hierarchy boundaries to generate a report on the fanin cone.
Default: Stops at the input port and it will report on the fanin path starting from that port (this is treated as the start point).

`-level` *integer*
: Specifies the number of levels that need to be traversed for generating the report.

list_of_pin_path_or_id
: Specifies the Tcl list of pin or port path names or object identifiers for which to start the fanin cone search.

`-maxline` *integer*
: Specifies the number of lines which need to be traversed for generating the report.

`-output` *filename*
: Specifies that a report be written out to a file that can be reviewed later. The name of the file written out is designated by *filename*.

`-tristate`
: Includes the tristates in the fanin path.

`-sequential`
: Includes the sequential cells in the fanin path.
Default: The reporting stops at the encounter of the first sequential cell in the fanin path.

Related Information

get_fanin

get_fanout

report_fanin

set_table_style

report_fanout

```
report_fanout [-level integer] [-tristate] [-sequential] [-hierarchical]
     [{> | >>} filename] [-maxline integer] list_of_pin_path_or_id
```

Generates a report on the fanout cone of the specified design objects in the current module. The report generated contains all the pertaining information on fanout, for example, from pin, to pin, level in the module and the module. All pins and their associated nets on the fanout path are reported with hierarchical names (starting at the current module).

Options and Arguments

`-hierarchical`
: Allows the command to traverse forward across hierarchy boundaries to generate a report on the fanout cone.
Default: Stops at output ports and report the fanout path ending at that port (which is treated as the end point).

`-level` *`integer`*
: Specifies the number of levels that need to be traversed for generating the report.
Default: Value is infinity.

`list_of_pin_path_or_id`
: Specifies the Tcl list of pin or port path names or object identifiers for which to start the fanin cone search.

`-maxline` *`integer`*
: Specifies the number of lines which need to be traversed for generating the report.

`-output` *`filename`*
: Specifies that a report be written out to a file that can be reviewed later. The name of the file written out is designated by *`filename`*.

`-sequential`
: Includes the sequential cells in the fanout path.
Default: The reporting stops at the encounter of the first sequential cell in the fanout path.

`-tristate`
: Includes the tristates in the fanout path.

Related Information

get_fanin

get_fanout

report_fanout

set_table_style

report_functional_mode

```
report_functional_mode [-cell cell_name] instance_list
```

Reports the status (active or inactive) of all modes in the instance list.

The following form of the command reports a list of mode groups and modes for *cell_name*:

```
report_functional_mode -cell cell_name
```

Without arguments, the command reports status of all modes in all instances:

```
report_function_mode
```

Options and Arguments

`-cell` *cell_name*
: Specifies the name of the cell.

instance_list
: Specifies the list of instances for which the functional mode status needs to reported.

Examples

- The following command reports the status of all modes in all instances:

  ```
  report_functional_mode
  ```

- The following command reports the status of all modes in cell named `RAM`:

  ```
  report_functional_mode -cell RAM
  ```

- The following command reports the status of all modes in instances `A1/B3/C2` and `A1/B2/C2`:

  ```
  report_functional_mode {A1/B3/C2 A1/B2/C2}
  ```

Related Information

`reset_functional_mode`

`set_functional_mode`

report_inactive_arcs

```
report_inactive_arcs [-from {instance | pin}] [-to {instance | pin}]
     [-delay_arcs_only | -check_arcs_only] [-type disable_type]
     [{> | >>} filename]
```

Reports information about disabled timing and timing check arcs in the design. Reports all arcs disabled due to specified exceptions using commands such as `set_disable_timing`, `set_disable_clock_gating_check` or `set_constant_for_timing`, as well as information about arcs disabled by constant propagation during timing analysis, snipped loop arcs, and arcs disabled in the timing library.

Options and Arguments

`{> | >>} filename`
: Specifies the name of the file in which the report is saved. The *`filename`* argument must be the last argument in the list. If the file name is not specified, the report displays on the standard output without being saved. Since the report is often large, text may be lost if you do not save the report to a file.

`-check_arcs_only`
: Lists only disabled check arcs. The `check_arcs` are disabled by using the `set_disable_timing` and the `set_disable_clock_gating_check` commands.
Default: Both disabled timing delay arcs and check arcs are listed.

`-delay_arcs_only`
: Lists only disabled timing delay arcs.

`-from instance`
: Reports all disabled timing arcs in the fanout logic cone of instance output pins.

`-from pin`
: Reports all disabled timing arcs in the fanout logic cone of a specified pin.

`-from -to`
: Reports all disabled timing arcs in the cone of logic between the from and to pins.

`-to` *`instance`*

Reports all disabled timing arcs in the fanin logic cone of instance input pins.

`-to pin`

Reports all disabled timing arcs in the fanin logic cone of a specified pin.

`-type` *`disable_type`*

Lists only the disabled arcs that belong to this category of disabled arcs. The `disable_type` option restricts the report to only those arcs which are disabled due to a particular timing analysis step, such as constant propagation, specified disables, or loop arc disables. The following are valid *`disable_type`* values:

`check_types`:
Setup, Hold, Recovery, Removal, PulseWidth, ClockGatingSetup, ClockGatingHold, ClockGatingPulseWidth, ClockPeriod, Skew, ClockSeparation, NoChange. These types apply to the disabled check arcs and are listed under the “CheckType” field if the disabled arc is a timing check arc. For delay arcs the “CheckType” field is empty.

`const`
Propagated constant — arcs disabled due to a constant value. The constant value can apply to one of the arc nets/pins, to side inputs which disable a condition for the arc, to nets that disable an active mode for the arc, or disable a default conditional arc.

`disable`
Defined by using the `set_disable_timing` command.

`disable_clock_gating`
Defined by using the `set_disable_clock_gating_check` command. These arcs are only listed if the `-check_arc` option is specified.

`library_disable`
Delay arc disabled due to library disables. These disables may be caused by the following conditions: Disables specified using the `set_disable_cell_timing` command; The arc is an asynchronous arc and the global `lib_build_asynch_arc` is

set to `false`; the arc is a conditional default timing arc and the global `lib_build_timing_cond_default_arc` is set to `false`.

`snipped`
Loop snipped arcs — arcs automatically disabled by CTE as part of a feedback loop. The arcs can be enabled by using the `reset_feedback_loop_snipped_arcs` command.

Example

The following command reports all disabled arcs in the design:

```
report_inactive_arcs
```

```
From             To                    Summary             CheckType   Reason
-------------------------------------------------------------------------------------------
U1/U2/reg[0]/E ^ U1/U2/reg[0]/D ^      const                           U1/U2/reg[0]/D = 0
U1/U2/reg[0]/E ^ U1/U2/reg[0]/D v      const                           U1/U2/reg[0]/D = 0
U1/RAM1/A ^      U1/RAM1/D v           const                           mode (RW) = 0
M0/M1/D ^        M0/M1/Q ^             const                           (EN) == 0
A0/D v           A0/Q ^                disable                         User Disable
N0/A v           N0/Q ^                snipped                         loop arc
DREG1/CP ^       DREG1/D ^             const setup                     DREG1/D = 1
DREG1/CP ^       DREG1/D V             const Setup                     DREG1/D = 1
N0/A ^           N0/B ^                disable_clock_gating    setup   N0/A = 1
N0/A v           N0/B ^                disable_clock_gating    setup   N0/A = 1
D0/A ^           D0/Z v                library_disable                 library disable
-------------------------------------------------------------------------------------------
```

report_library

```
report_library[-library library_name] [-cells [pattern]] [-wireloads]
     [-operating_conditions] [-pvt {min | typ | max}] [-tcl_list]
     [{> | >>} filename]
```

Generates a report about the technology library used in the design.

Options and Arguments

`{> | >>} filename`
: Specifies the name of the file in which the report is saved. The *filename* argument must be the last argument in the list. If the file name is not specified, the report is displayed on the standard output without being saved. Since the report is often large, text may be lost if you do not save the report to a file.

`-cells [pattern]`
: Reports information about all cells in the library. If a *pattern* is specified, only the cell names in the library that match the pattern are reported. See Examples on page 973.

`-library library_name`
: Reports contents of the specified library.
Default: Contents of the library set by `set_global target_technology` are reported.

`-operating_conditions`
: Reports information on all the operating conditions in the library.

  If none of the three options (`-cells`, `-wireloads`, `-operating_conditions`) are specified, the report contains information about all the cells, wire-load models and operating conditions.

`-pvt {min | typ | max}`
: Reports library data for a particular PVT (process, voltage, temperature) corner. Specify only one PVT corner per command.
Default: The library data is reported for the `max` PVT corner only. If the library is a merged library, the order of reporting is `max` if it exists, then `typ` if it exists, and finally `min` if neither `max` or `typ` data exists.

Tip

If you have a merged library, use multiple commands with different `-pvt` options to report all the data. See Examples on page 973.

`-tcl_list`

Produces the report in a Tcl list, not a readable report. This is useful for integrating timing with custom Tcl functions and also for customizing report generation.

`-wireloads`

Reports information on all the wire-load models in the library.

Examples

- The following command reports the contents of the technology library to the file named `library.rep`:

```
report_library library.rep
```

- The following command reports all cells matching the string `AO*` in library `lca300kv`:

```
report_library -library lca300kv -cells AO*
```

- The following commands use a merged library defined by:

```
read_tlf -min lib_bc.tlf -max lib_wc.tlf -name merge.tlf
```

  - This command shows only the operating conditions for the `max` PVT (those from the `-max` library):

```
report_library -operating_condition
```

```
+-------------------------------------------------------------+
|  Operating | process | temperature | voltage |       OC type |
|  Condition |         |             |         |               |
|------------+---------+-------------+---------+---------------|
|      WCMIL |    1.50 |      125.00 |    1.65 | balanced_tree |
|      WCCOM |    1.50 |       85.00 |    1.65 | balanced_tree |
+-------------------------------------------------------------+
```

  - This command shows the operating conditions that come from the `-min` library:

```
report_library -operating_condition -pvt min
```

```
+-------------------------------------------------------------+
|  Operating | process | temperature | voltage |        OC type |
|  Condition |         |             |         |                |
|------------+---------+-------------+---------+----------------|
|      BCMIL |    0.50 |      -55.00 |    1.95 | best_case_tree |
|      WCMIL |    1.50 |      125.00 |    1.65 |  balanced_tree |
|      BCCOM |    0.50 |        0.00 |    1.95 | best_case_tree |
|      WCCOM |    1.50 |       85.00 |    1.65 |  balanced_tree |
+-------------------------------------------------------------+
```

Related Information

`report_timing`

`set_table_style`

`set_global target_technology`

report_net

```
report_net [-min_fanout int] [-max_fanout int]
     {-net list_of_net_name_or_id | -pin list_of_pin_name_or_id}
     [-old] [-tcl_list] [-hier] [-output filename | {> | >>} filename]
```

Reports the net information on the current module. The information includes the net name, the number of source pins, sink pins and bidirectional pins on the net, the wireload model, the wire capacitance, wire resistance, and total capacitance.

Electrical information such as pin capacitance, wire capacitance, wire resistance, and the wireload model with which the net is associated are also included. If wire capacitance or wire resistance are backannotated from post-layout information, such as an SDF file, they are indicated in the Wireload Model (`WLM`) column accordingly. If any attributes are set on the net, they are also indicated in the attribute column.

Use the `-pin` option to report the net that is connected to the specified pin (or port). Both `-net` and `-pin` cannot be specified at the same time. The net (or pin) names can be hierarchical names that are unique relative to the top timing module.

Note: Set the top timing module with the `set_top_timing_module` command before using the `report_net` command.

This command only searches for the defined nets and does not modify any object in database.

Options and Arguments

`-hier`
: Reports all of the nets in the design hierarchy. Without this argument, only the nets in the current level of hierarchy are reported.

`-max_fanout` *`integer`*
: Reports the net whose number of fanouts is less than the maximum number specified by *`integer`*.

`-min_fanout` *`integer`*
: Reports the net whose number of fanouts is greater than the minimum number specified by *`integer`*.

`-net` *`list_of_net_name_or_id`*
: Reports the set of nets on the list of net names. This argument

cannot be specified at the same time as `-pin`. If neither `-net` or `-pin` is specified, all nets are listed by default.

Note: This argument also reports the capacitance of the net and the input pins.

`-old`

Produces the table in 3.0 format (without information about the interconnect delay model).

`-output` *`filename`* | {> | >>} *`filename`*

Dumps the report to the specified file instead of to the standard output. The *`filename`* argument must be the last argument in the list. Cannot be used simultaneously with the `-tcl_list` option.

`-pin` *`list_of_pin_name_or_id`*

Reports the set of nets to which the pins on the list *`list_of_pin_name_or_id`* are connected. This argument cannot be specified at the same time as `-net`.

`-tcl_list`

Produces the report in Tcl list form instead of a tabular format. This is useful for extracting information from the report in a Tcl program. Cannot be used simultaneously with the `-output` *`filename`* option.

Examples

The format of the report is the same for all examples, as shown in the first example.

- The following command reports information about net named `in`:

```
report_net -net in
```

```
Net Name                : in
Number of Sources       : 1
Number of Sinks         : 2
Number of Bidis         : 0

Source of parasitics    : Wireload model(.lib B0X0 lca300kv)
Net Capacitance         : 0.00
Total Capacitance       : 0.02
```

```
+-------------------------------------------------------------------------------------+
|                                        Source                                       |
|-------------------------------------------------------------------------------------|
|            Pin             |  Reduced Net |   Driver Load   |     Slew      |       |
|----------------------------+--------------+-----------------+---------------+-------|
```

```
|  Name |  Dir  |  Cell |  Cap  |    Model     |  Model | Ctotal |  Rise |  Fall | Phase |
|-------+-------+-------+-------+--------------+--------+--------+-------+-------+-------|
|  in   |  IN   |  top  |  0.00 | Elmore Delay | CTOTAL |  0.02  |  0.05 |  0.04 |@(D)(P)|
+-------------------------------------------------------------------------------------------+

+-------------------------------------------------------------------------------------------+
|                                        Sinks for pin in                                   |
|-------------------------------------------------------------------------------------------|
|                  Pin                | Reduced Ne |     Delay    |        Slew           |
|-------------------------------------+------------+--------------+-----------------------|
|      Name       | Dir |  Cell |  Cap | Parameters | Rise | Fall | Rise | Fall | Phase   |
|-----------------+-----+-------+------+------------+------+------+------+------+---------|
| I_block/B_reg/D | IN  | FD1QA | 0.01 |    0.00    | 0.00 | 0.00 | 0.05 | 0.04 | @(D)(P)|
| I_block/A_reg/D | IN  | FD1QA | 0.01 |    0.00    | 0.00 | 0.00 | 0.05 | 0.04 | @(D)(P)|
+-------------------------------------------------------------------------------------------+
```

- The following command reports the net to which the pin `interesting_pin` is connected:

  ```
  report_net -pin interesting_pin
  ```

- The following command reports all the nets in the current module:

  ```
  report_net -net *
  ```

- The following command reports all nets with a fanout greater than 16:

  ```
  report_net -net * -min_fanout 16
  ```

The following commands report nets without wire capacitance or resistance assertions using the `report_annotations` command:

```
report_annotations -missing_capacitances
report_annotations -missing_resistances
```

Related Information

`report_path_exceptions`

`report_timing`

`set_current_module`

report_path_exceptions

```
report_path_exceptions {-ignored] [-tcl_list] [{> | >>} filename]
```

Generates a report about the path exceptions that you have specified using the set_false_path, set_path_delay_constraint and set_cycle_addition commands.

A path can have more than one path exception. Multiple path exceptions that match a given path are prioritized. The adjustment on the path is from the path exception with the highest priority. Table 7-1 on page 797 ranks path exception priorities from highest to lowest.

Options and Arguments

`{ > | >> filename }`

Redirects the output to the named file. If this option is omitted, the default is `stdout`.

`-ignored`

Reports only the path exceptions that have been ignored by timing analysis. Useful for debugging assertions that may be ineffective or incorrectly entered. A path exception may be ineffective if it is covered by, or over-ridden by another path exception. For example:

```
set_false_path -to OUT
set_cycle_addition -to OUT 2
```

The `set_cycle_addition` command would be ignored because the false path has a higher priority.

One example of such an ineffective or incorrectly entered assertion is the `set_cycle_addition` command that is ignored due to the `set_false_path` command shown in the Examples section below.

`-tcl_list`

Produces the report in Tcl list form instead of a tabular format. This is useful for extracting information from the report in a Tcl program.

Examples

- The following commands shows that a false path has priority over cycle addition on the same path:

```
set_false_path -from I_block/A_reg/Q -to J_block/D_reg/D
set_cycle_addition -from I_block/A_reg/Q -to J_block/D_reg/D 2
report_path_exceptions
```

```
+-------------------------------------------------------------------------+
|      From       |       To        |     Early     |      Late       |
|-----------------+-----------------+---------------+-----------------|
| I_block/A_reg/Q | J_block/D_reg/D | false         | false           |
| I_block/A_reg/Q | J_block/D_reg/D |               | add 2 ignored   |
+-------------------------------------------------------------------------+
```

The term `false` means that the path is treated as a false path due to the path exception, and `false ignored` means that there was a false path assertion but that it is ignored. This may be because a path does not exist that matches the path exception. If you notice `ignored` path exceptions in your design, it is a good idea to check why they are ignored. The problem may be a typo in the script or some other reason

- The following command shows how edges are handled:

```
set_false_path -from_rise {in[0]} -through {out[0]}
set_false_path -from_rise {in[0] in[1]} -to_fall {out[0]}}
set_false_path -from_rise {in[0]}
```

```
+-------------------------------------------------------------------------+
|     From      | Through |    To    |     Early     |     Late      |
|---------------+---------+----------+---------------+---------------|
| ^ in[0]       | out[0]  |          | false ignored | false ignored |
| ^ in[0] in[1] |         | v out[0] | false ignored | false ignored |
| ^ in[0]       |         |          | false         | false         |
+-------------------------------------------------------------------------+
```

- The following commands show the path exception specified by the `set_disable_timing` command:

```
set_disable_timing -from foo/i/A
report_path_exceptions
```

```
+-----------------------------+
|  From   |  Early  |  Late   |
|---------+---------+---------|
| foo/i/A | disable | disable |
+-----------------------------+
```

Related Information

`report_timing`

`report_area`

`report_design_rule_violations`

`report_hierarchy`

`report_fsm`

`reset_path_exception`

`set_current_module`

See Generating Path Exception Reports in the *Common Timing Engine (CTE) User Guide*.

report_path_groups

```
report_path_groups [-name group_name] [-tcl_list] [{> | >>} filename]
```

Lists the names of all path groups with the corresponding paths.

Options and Arguments

`{ > | >>` *`filename }`*

Redirects the output to the named file. If this option is omitted, the default is `stdout`.

`-name` *`group_name`*

Limits the report to the path specifications that define the given group.
Default: All path groups are reported.

`-tcl_list`

Produces the report in Tcl list form instead of a tabular format. This is useful for extracting information from the report in a Tcl program.

Example

The following command reports the group defined in the example for `set_path_group`:

```
report_path_groups
+------------------------+
|  From  |  To   | Group |
|        |       |  Name |
|--------+-------+-------|
| "CLKA" | "*"   | GRPA  |
+------------------------+
```

Related Information

`report_path_group_options`

`reset_path_group`

`set_path_group`

report_path_group_timing

```
report_path_group_timing [-tcl_list] [{> | >>} filename]
```

Produces a report of the timing of all active path groups in the design. The report includes the name of the path group, the number of critical (out of the total) endpoints, the worst slack of the group, and the Total Endpoint Negative Slack (TENS), which is the sum of the negative slack of all endpoints in the group.

Options and Arguments

`> | >>` *filename*
: Redirects the output to the named file. If this option is omitted, the default is `stdout`.

`-tcl_list`
: Produces the report in Tcl list form instead of a tabular format. Useful for extracting information from the report in a Tcl program.

Related Information

report_path_group_options

report_path_group_options

set_path_group

set_port_capacitance

report_poles_residues

```
report_poles_residues {[-source pin_list -sink pin_list] | [-net netname_list]}
     [-early | -late] [-tcl_list] [-output filename | {> | >>} filename]
```

Reports poles and residues from interconnect RC networks in an OLA library. The PI model is reported by `report_net`.

Options and Arguments

`-early|-late`
: Reports the early timing (data hold/clock setup) or late timing (data setup/clock hold).
Default: `-late`.

`-net` *`netname_list`*
: Reports the poles and residues for the named nets only.
Default: Poles and residues for all nets are reported. This option cannot be used with `-source -sink` pair options.

`-output` *`filename`* `|` `{>` `|` `>>}` *`filename`*
: Dumps the report to the specified file instead of to the standard output. The *`filename`* argument must be the last argument in the list.

`-source` *`pin_list`* `-sink` *`pin_list`*
: Specifies the list of source-sink pairs of a net for which the poles and residues are displayed. The specified sources and sinks must be connected to the same net.

`-tcl_list`
: Produces the report in Tcl list form instead of a tabular format. This is useful for extracting information from the report in a Tcl program.

Example

The following command reports the poles and residues for the named nets only:.

```
report_poles_residues -net n1
```

```
+---------------------------------+
| Report  | report_poles_residues  |
|---------+------------------------|
| Options | -net n1 > n1_poles.rpt |
+---------+------------------------+
```

```
| Date    | 20020305.094704        |
| Tool    | bg_shell               |
| Release | v5.0-b025              |
| Version | Mar  4 2002 07:53:50   |
+---------+------------------------+
| Module  | msinks_yhn             |
+----------------------------------+

Net Name                 : n1
Maximum number of poles  : 2
+-------------------------------------------------------------------+
| Source |  Sink |          Poles           |       Residues       |
|--------+-------+--------------------------+----------------------|
|  Name  |  Name |     Real     |    Img    |   Real    |   Img    |
|--------+-------+--------------+-----------+-----------+----------|
| drv1/Z | ld2/A | -1430.454346 | -0.028014 |  0.033253 | 0.000000 |
|        |       |  -36.852859  |  0.000761 | -1.033253 | 0.000000 |
| drv1/Z | ld1/A |  -36.300274  | -0.027548 | -1.000000 | 0.000000 |
+-------------------------------------------------------------------+
```

Default: Poles and residues for all nets are reported.

Related Information

read_ola

read_spef

report_net

report_ports

```
report_ports [-type ([input] | [source_insertion] | [insertion] | [clock_root] |
     [uncertainty] | [arrival] | [required] | [external] | [clk_arrival] |
     [clk_required] | [port_cap] | [fanout_load] | [num_external_sinks] |
     [num_external_sources] | [fanout_load_limit] | [port_wire_load] |
     [drive_resistance] | |[drive_cell] | [slew_time] | [slew_limit] |
     [constant] | external_detail | drive_resistance_detail)] [-tcl_list]
     [-pins port_name_id_list]
```

Reports timing assertions on ports and pins. Specify a specific port using either an object ID, a name, or a list of IDs or names.
Default: If no parameters are supplied) reports all timing assertions on all ports of the current module.

Use the `-type` option to query specific assertions.

Use the `-tcl_list` option to format the information for use in a Tcl program.

Options and Arguments

`-pins` *`port_name_id_list`*
Specifies the pins or ports to be reported. Pins can be specified by name or object ID in a Tcl list.

`-tcl_list`
Produces the report in a Tcl list, not a human-readable report. This is useful for integrating timing with custom Tcl functions and also for customizing report generation.

`-type {`*`assertion_type | assertion_type_list`*`}`
Specifies the assertions to be reported.
Default: All assertions are reported. The following are valid *`assertion_type`* values:

`arrival`
Reports arrival time assertions as set by (obsolete) `set_data_arrival_time`.

`clk_arrival`
Reports clock arrival time assertions as set by `set_clock_insertion_delay`.

clk_required
Reports clock required time assertions as set by set_clock_required_time.

clock_root
Reports the clock root as set by set_clock_root.

constant
Reports constant assertions as set by set_constant_for_timing.

drive_cell
Reports drive cell assertions as set by set_drive_cell.

drive_resistance
Reports drive resistance assertions as set by set_drive_resistance.

drive_resistance_detail
Reports detail about the input ports with drive resistance assertions.

external
Reports external delay assertions as set by set_external_delay.

external_detail
Reports detail about the output ports with external delay assertions.

fanout_load
Reports fanout load assertions as set by set_fanout_load_limit.

fanout_load_limit
Reports fanout load limit assertions as set by set_fanout_load_limit.

insertion
Reports the network insertion delay as set by set_clock_insertion_delay.

num_external_sinks
Reports external sink assertions as set by set_num_external_sinks.

num_external_sources
Reports external source assertions as set by set_num_external_sources.

port_cap
Reports port capacitance assertions as set by set_port_capacitance.

port_wire_load
Reports port wire-load model assertions as set by set_port_wire_load.

required
Reports required time assertions as set by set_data_required_time.

slew_limit
Reports slew time limit assertions as set by set_slew_time_limit.

slew_time
Reports slew time assertions as set by set_slew_time.

source_insertion
Reports the source insertion delay as set by set_clock_insertion_delay -source.

uncertainty
Reports the clock uncertainty as set by set_clock_uncertainty.

Examples

- The following command shows a port report:

```
report_port
```

```
+-------------------------------------------------------------------------------------+
|                |      |                 |           |     Early     |     Late     |
|----------------+------+-----------------+-----------+---------------+--------------|
|    Pin Name    |  Dir |    Assertion    | Clock Name |  Rise |  Fall |  Rise |  Fall|
|----------------+------+-----------------+-----------+-------+-------+-------+------|
```

```
|  I_block/u000/A |  IN  |    slew_time     | CLK1(C)(P) |  1.60 |  1.60 |  1.60 |  1.60 |
| J_block/D_reg/D |  IN  |     arrival      | CLK1(D)(P) |       |  0.07 |       |       |
| J_block/D_reg/D |  IN  | drive_resistance | CLK1(D)(P) |  1.20 |  1.20 |  1.20 |  1.20 |
|      clkA       |  IN  |   clk_arrival    | CLK2(C)(N) |  2.00 |  0.00 |  2.00 |  0.00 |
|      clkA       |  IN  |    slew_time     | CLK1(C)(P) |  1.30 |  1.30 |  1.30 |  1.30 |
|      clkB       |  IN  |   clk_arrival    | CLK2(C)(P) |  0.00 |  2.00 |  0.00 |  2.00 |
|      clkB       |  IN  |    slew_time     | CLK1(C)(P) |  1.30 |  1.30 |  1.30 |  1.30 |
|      clkC       |  IN  |   clk_arrival    | CLK2(C)(P) |  0.05 |  0.07 |  0.05 |  0.07 |
|      clkD       |  IN  |   clk_arrival    | CLK2(C)(N) |  3.00 |  0.50 |  3.00 |  0.50 |
|       in        |  IN  |    drive_cell    |  *(D)(P)   |  BUFA |  BUFA |  BUFA |  BUFA |
|       out       |  OUT |     external     | CLK1(C)(N) |       |       |  1.00 |  1.00 |
+-----------------------------------------------------------------------------------------+

+-----------------------------------------------------------+
|  Pin  |  Dir  |     Assertion      |         Value          |
|  Name |       |                    |                        |
|-------+-------+--------------------+------------------------|
|  out  |  OUT  |      port_cap      | ( 3.20 : 3.20 : 3.2 )  |
|  out  |  OUT  | num_external_sinks |                      4 |
|  out  |  OUT  |   port_cap_limit   |                   4.00 |
|  in   |  IN   |     slew_limit     |                   1.50 |
|  in   |  IN   |   port_cap_limit   |                   2.00 |
|  clkA |  IN   |   port_cap_limit   |                   2.00 |
|  clkB |  IN   |   port_cap_limit   |                   2.00 |
|  clkC |  IN   |   port_cap_limit   |                   2.00 |
|  clkD |  IN   |   port_cap_limit   |                   2.00 |
+-----------------------------------------------------------+
```

- The following command shows, from a different design, that `input`, `source_insertion` and `clock_root` assertion type options report the assertions in the same format as `arrival`.

```
report_port -type clock_root source_insertion input -pin clk in
```

```
+-------------------------------------------------------------------------+
|      |      |                 |          |    Early     |     Late     |
|------+------+-----------------+----------+--------------+--------------|
|  Pin | Dir  |    Assertion    |   Clock  | Rise | Fall  | Rise | Fall  |
|  Name|      |                 |   Name   |      |       |      |       |
|------+------+-----------------+----------+------+-------+------+-------|
|  clk |  IN  |    clock_root   | CLK(C)(P) |      |       |      |       |
|  clk |  IN  | source_insertion |  *(D)(P) |  2.50 |  2.50 |  2.50 | 2.50 |
|  in  |  IN  |      input      | CLK(D)(P) |  3.00 |  3.00 |  3.00 | 3.00 |
+-------------------------------------------------------------------------+
```

- The following command shows that the insertion and uncertainty are reported as one value in the following format:

```
insertion              : RISEmin FALLmin : RISEtyp FALLtyp : RISEmax FALLmax
uncertainty            : EARLYrise EARLYfall LATErise LATEfall
```

- The following command shows that the `-to` uncertainty is denoted by `(T)` next to the number. If a value is not specified, it is denoted by a dash (`-`):

```
report_port -type insertion uncertainty -pin clk
```

```
+-----------------------------------------------------------------------+
|  Pin  |  Dir  |  Assertion           |              Value             |
|  Name |       |                      |                                |
|-------+-------+----------------------+--------------------------------|
|  clk  |  IN   |  insertion           | ( 1.50 3.50 : - - : 2.50 4.50 ) |
```

```
|  clk  |  IN   |  uncertainty          |          1.00(T) 1.10 - 1.20(T) |
+-------------------------------------------------------------------------+
```

Related Information

report_clocks

report_net

report_timing

set_table_style

report_timing

```
report_timing [-clock_from clk_signame_list] [-clock_to clk_signame_list]
     [-edge_from {lead | trail}] [-edge_to {lead | trail}]
     [-rise | -fall] [-early | -late] [-max_slack float] [-min_slack float]
     [-max_paths integer | -max_points npoint] [-nworst integer]
     [{-from | -from_rise | -from_fall} pin_list]
     [{-through | -through_rise | -through_fall} pin_list]
     [{-to | -to_rise | -to_fall} pin_list]
     [-bidi_input_from | -bidi_output_from] [-bidi_input_to | -bidi_output_to]
     [-bidi_input_through | -bidi_output_through]
     [-unconstrained [-delay_limit float]] [-check_clocks]
     [-latch] [-path_group groupname_list]
     [-false_path_analysis {static | robust}] [-justify] [-true]
     [-tclfile tclfile_name] [-gcffile gcffile_name]
     [-hdl_sim_file hdlfile_name]
     [-spice_output filename [-spice_power_node power_rail_voltage_name_list]]
     [-net] [-unique_pins] [-path_type {end | summary | full | full_clock}]
     [-check_type {setup | hold | pulse_width | clock_period | clock_gating_setup
     | clock_gating_hold | clock_gating_pulse_width | recovery | removal |
    clock_separation | skew | no_change_setup | no_change_hold}]
     [-format column_list] [-tcl_list] [{> | >>} filename]

     Valid columns are: addition arc arrival bottle cell clkordata clkordatapin
     delay delay_ast direction dont_modify_instance dont_modify_net edge eslack
     fanin fanout fpin from_edge hpin instance instance_location load lslack net
     phase pin pinload pin_location required slack slew stolen to_edge tpin
     wireload wlmodel
```

Generates a timing report that provides information about the various paths in the design. The reports typically contain data on the delay through the entire path. The start node and the end node of each path is identified.

Note: Timing reports show different slack times because of slew merging. For more information on timing differences due to slew merging, see Explaining Timing Differences Caused by Slew Merging When Path Exceptions are Applied in the *Common Timing Engine (CTE) User Guide*.

The report contains the following information:

- The slack times of the arriving signal at the end node
- The start node
- The associated transitions
- The signal required times and the actual signal arrival times
- Any phase shifts applied when evaluating timing checks

- Any CPPR values applied to timing check evaluation

If the *filename* is specified, then the report is written out to the file. Otherwise, the report is displayed on the standard output.
Default: The detailed timing information is reported.

Use the `-format` option to customize the reports to your needs by requesting the exact fields in which you have an interest. Use a combination of the `-format` and `-tcl_list` options to integrate the timing reports into your Tcl scripts.

Use the `-from` option to limit the number of paths reported, and to find specific paths in the design.

Options and Arguments

`-bidi_input_from | -bidi_output_from`
Specifies that the bidirectional pins in the `-from`/`-from_rise`/`-from_fall` pin list refer to the input or output part of the bidirectional pins. For default value, see Bidirectional Pin Defaults on page 799.

`-bidi_input_through | -bidi_output_through`
Specifies that the bidirectional pins in the `-through`/`-through_rise`/`-through_fall` pin list refer to the input or output part of the bidirectional pins. For default value, see Bidirectional Pin Defaults on page 799.

`-bidi_input_to | -bidi_output_to`
Specifies that the bidirectional pins in the `-to`/`-to_rise`/`-to_fall` pin list refer to the input or output part of the bidirectional pins. For default value, see Bidirectional Pin Defaults on page 799.

`-check_clocks`
Generates reports based on timing paths on the clock network instead of the standard timing to data endpoints. This report includes clock paths that end at the reference end of a check or a clock gating end point.
Default: Data paths and all the other clock paths (for example, a clock path ending at a D pin of a register) are reported.

A clock path ending at a D pin of a register is considered a data path by default because the global

`clock_gating_regardless_of_downstream_logic` is `false` by default.

Note: If the `clock_gating_regardless_of_downstream_logic` global is set to `true`, the `report_timing` command slack results without the `-check_clocks` option may not match the slack numbers from the `get_module_worst_slack` command without the `-clknet_too` option.
Default: Without the `-clknet_too` option, the `get_module_worst_slack` command does not consider clock networks at all.

`-check_type` *`check_type`*

Reports only the paths that end at the specified timing check. The supported check types are: `setup`, `hold`, `pulse_width`, `clock_period`, `clock_gating_setup`, `clock_gating_hold`, `clock_gating_pulse_width`, `recovery`, `removal`, `clock_separation`, `skew`, `no_change_setup`, `no_change_hold`.

Do not use this option with the `-unconstrained` or `-early|-late` options.

`-clock_from` *`clk_signame_list`*

Generates reports based on source clock waveform(s). Reports only those paths whose source clocks are the clock signals in *`clk_signame_list`*.

`-clock_to` *`clk_signame_list`*

Generates reports based on target clock waveform(s). Reports only those paths whose target clocks are the clock signals in *`clk_signame_list`*.

`-delay_limit` *`float`*

Specifies the path delay limit for unconstrained paths (`-unconstrained` option).

For early paths (`-early` option), reports only those paths with path delay less than the delay limit. For late paths (`-late` option) reports only those paths with path delay more than the delay limit.

The `-delay_limit` option can only be used in conjunction with the `-unconstrained` option.

`-early | -late`

Generates the timing report for early paths (data hold checks/ clock setup checks) or late paths (data setup checks/clock hold checks).
Default: `-late`.

The `-late` and `-early` options in `report_timing` are meant for the data path, not the clock path. CTE will check late data against early clock in the `-late` option, and early data against late clock in the `-early` option.

For clock signals, there is another timing criteria called the `PulseWidth Check` in the report. Specifying `report_timing` with the `-late` option will report the worst slack path between the pulsewidth check and the hold check. Similarly, the `-early` option reports the worst slack path between the pulsewidth check and the setup check. See Examples on page 1004 for some sample reports.
Default: `-late`.

`-edge_from {lead | trail}`

Generates reports based on source clock edge, either leading or trailing.
Default: Both

`-edge_to {lead | trail}`

Generates reports based on the target clock edge, either leading or trailing.
Default: Both

`-from | -from_rise | -from_fall` *`pin_list`*

Reports paths starting from the pin(s) specified by the *`pin_list`*. Using `-from_rise` (or `-from_fall`) specifies that the rising (or falling) edge of the signals on the pins in *`pin_list`* are the start of the paths.

Note: Use this option with the `-through` and `-to` options for specifying particular paths in the design.

`-max_paths` *`integer`*

Reports the specified number of worst paths in the design, regardless of the endpoint. This is useful, but can be time consuming if a large number of paths are requested. The paths are always sorted based on slack.
Default: worst path

The `-max_paths` option cannot be used with the `-max_points` option.

Note: You cannot use this option when the following CPPR analysis global is set to `true`:
`timing_remove_clock_reconvergence_pessimism`

`-max_points` *`integer`*

Reports the worst path it finds to each endpoint up to the number specified by the `-max_points` option. If `-path_type end` option is specified, it reports worst path to each endpoint. This is the most frequently used report.
Default: Only the worst path to one endpoint is reported.

The `-max_points` option cannot be used with the `-max_paths` option.

`-max_slack` *`float`*

Reports only those paths with slack equal to or less than the value of *`float`* are reported. The `-max_slack` option limits the report to paths that fall into the specified range. A positive slack value indicates that timing was met. A negative value for slack indicates a timing violation.

The `-max_slack` option cannot be used with the `-unconstrained` option.

Typically, you can report all violating endpoints by using:
`-max_slack 0.0 -max_points 1000`

Note: This still limits the report to 1000 endpoints. The endpoint limitation can be adjusted to a reasonable number.

`-min_slack` *`float`*

Reports only those paths whose slack is greater than the value of *`float`*.

The `-min_slack` option cannot be used with the `-unconstrained` option.

Generate a timing report containing any path with greater than the specified slack by using the `-max_paths` option. For example, to report all paths with slack greater than 2ns: `-min_slack 2.0 -max_paths 1000`

Note: This still limits the report to 1000 paths. The path limitation can be adjusted to a reasonable number.

`-nworst` *`integer`*

Specifies the number of paths to be enumerated for each endpoint. Use the `-nworst` option to report all the checks at an end point or use the `-check_type` option to report a specific check.
Default: Only the worst path to each endpoint is reported.

The `-nworst` option cannot be used with the `-max_paths` option.

Note: You cannot use this option when the following CPPR analysis global is set to `true` because it removes common path pessimism (CPPR) from slack calculation: `timing_remove_clock_reconvergence_pessimism`. For more information, see "Removing Delay Pessimism from Reported Paths" in the *Cadence Common Timing Engine (CTE) User Guide*.

`-path_group` *`groupname_list`*

Reports only paths contained in the groups specified in *`groupname_list`*.

Report paths belonging to the default path group by using `-path_group default`. The paths that do not belong to any user-specified path group belong to the default path group named `default`.

`-rise | -fall`

Reports the path with the specified edge on the endpoint.

If an endpoint is specified using `-to_rise` (or `-to_fall`) option, the `-rise` (or `-fall`) option is ignored and paths with edge specified by `-to_rise` (or `-to_fall`) are reported.

`-through | -through_rise | -through_fall` *`pin_list`*

Reports paths that pass through the pin(s) specified by the *`pin_list`*. Any number of `-through` pins can be specified. Using `-through_rise` (or `-through_fall`) specifies that the paths go through the rising (or falling) edge of the signals on the pins in *`pin_list`*.
The *`pin_list`* is a logical OR function. The resulting path may pass through any of the pins in the `-through` *`pin_list`*. To force the report to pass through multiple pins, separate `-through` statements are needed.

`-to | -to_rise | -to_fall` *`pin_list`*

Reports paths leading to the pin(s) specified by the *`pin_list`*. Pins in the *`pin_list`* can be either pins on the design boundary (ports) or pins on an instance. Only one list of `-to` pins can be specified per report. Using `-to_rise` (or `-to_fall`) specifies that the rising (or falling) edge of the signals on the pins in *`pin_list`* are at the end of the paths.

`-unconstrained`

Reports only the unconstrained paths (paths with no slack). If no paths are found, there may be constrained paths or false paths or the path may not exist. Each signal arriving at the path end node which does not have a matching required time, results in an unconstrained path. If there is an asynchronous signal (specified with'@' clock) arriving at the path end node, the path is always reported as an unconstrained path.

Note: An asynchronous signal is applied to all the unconstrained primary inputs during this mode.

The `report_timing` command without the `-unconstrained` option reports only constrained paths. If no constrained path is found, there may be unconstrained paths or false paths or the path may not exist. If no constrained or unconstrained path is found, the path is either false or does not exist structurally.

Note: The `-min_slack` and `-max_slack` options cannot be specified with the `-unconstrained` option. See the `-delay_limit` option described below.

See for the conditions under which a path is reported as constrained or unconstrained.

`-unique_pins` Reports paths through unique set of pins. Only one path (the worst) is reported for unique set of pins. This option suppresses multiple paths that differ only because of transition polarity or conditional arcs.

Note: Using this option may significantly affect runtime and memory usage if the number of paths to be reported is high.

Option for Latch Analysis Type

`-latch`

Reports the paths through latches in maximum time borrow latch mode. With this option, `latch_time_borrow_mode` is automatically set to `max_borrow`, if not already set. After `report_timing` the global is automatically reset back to its previous mode. Only the worst path is reported for each end point.

The `-latch` option cannot be used with the `-max_paths`, `-nworst`, or any of the `-from*` or `-through*` options.

For more information, see Analyzing Latch-Based Designs in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Options for Reporting False Paths

In addition to the analysis that is performed by default, you can use options to perform false path analysis. The following five options (`-false_path_analysis` through `-gcffile`) deal with false path analysis. For more information, see Identifying and Eliminating False Paths in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

`-false_path_analysis {static | robust}`
Determines the status of a path using static (or robust) analysis.

You must always specify the analysis type, either `static` or `robust`, with the `-false_path_analysis` option.

Unless the `-true` option is used to disable printing of false paths, a false path is indicated in the report as shown in this example:
Path 1: FALSE PATH

`-gcffile` *`gcffilename`*

Generates a General Constraint Format (GCF) file, *`gcffilename`*, containing GCF `DISABLE` constraints which correspond to the false paths identified. An error message is issued if *`gcffilename`* is not specified.

Use *`gcffilename`* to pass the `DISABLE` constraints (false paths) to backend tools for operations like place and route. These backend tools will not consider the timing on the false paths while doing placement and routing.

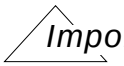
Note: This option can only be used with the `-false_path_analysis` option.

`-hdl_sim_file` *`hdlfile_name`*

Generates HDL simulation models for combinational TRUE paths. The HDL simulation models are appended to the file specified by *`hdlfile_name`*.

Use the `-hdl_sim_file` option to pass the result of false path analysis to dynamic simulation tools for the purpose of verifying true paths under sequential operation.

Note: This option can only be used with the `-false_path_analysis` option.



Before you use the `-hdl_sim_file` option, you must set two globals:
`set_global cfpv_testbench_name` *`testbench_name`*
`set_global cfpv_design_instance_name` *`instance_name`*

The first specifies the name of the simulation testbench, and the second specifies the design instance name within the testbench. These parameters are needed to produce the HDL output file for simulation. For more information and an example

output file, see Producing Simulation Vectors for Sequential Verification in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

`-justify`

Gets one test vector for which the reported paths become true. This option also displays a test pattern for each true path.

Note: Use this option only with the `-false_path_analysis` option.

`-tclfile` *`tclfilename`*

Generates a Tcl file, *`tclfilename`*, containing `set_false_path` commands corresponding to the false paths identified. An error message is issued if *`tclfilename`* is not specified.

Source *`tclfilename`* before running the `report_timing` command again. This filters out the false paths from the report and next available true paths are reported. The Tcl file can also be sourced before performing finer timing optimizations so that the false paths are not optimized.

Note: Use this option only with the `-false_path_analysis` option.

`-true`

Displays only identified true paths. The false paths identified are not printed. Specify the sensitization criterion using the `-false_path_analysis` option.

Note: Use this option only with the `-false_path_analysis {static | robust}` option.

Options for Creating Spice Output

Use the following two options to generate Spice files.

For the prerequisites, limitations, and example Tcl procedure, see Generating a Spice Netlist in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

`-spice_output` *`filename`*

Creates output in Spice format for the path reported. The file can be simulated in a Spice simulator for transient analysis of the path waveform.

CTE appends the suffix `.sp` to the given *`filename`*. If `report_timing` gives more than one path, then CTE generates more than one output file with the following pattern: `filename.sp filename2.sp filename3.sp`. In this example, `filename.sp` is the Spice representation of Path 1 in `report_timing`, `filename2.sp` corresponds to Path 2 and so on.
See Generating a Spice Netlist in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)* for more information.

`-spice_power_node` *`power_rail_voltage_name_list`*

Ties any node whose name is in the list of power rail names to its instance operating voltage. The instance-specific operating voltage can be updated using the `read_rrf` command when `rrf` file is available. Use this option only in conjunction with the `-spice_output` option.

When the instance operating voltage value is not specified, the rail voltage value from the operating conditions is used.

Options for Formatting and Redirecting Reports

`{> | >>}` *`filename`*

Stores the generated report in the file specified by *`filename`*. If the file name is not specified the report is displayed on standard output. The file name must be the last argument in the list.

`end`

Generates an end point report for each path consisting of an endpoint, cause, slack, arrival time, required time, and phase. This option generates a very fast report. The `-format` option does not have any impact on this report format.

Note: The `-path_type end` format is identical to the format produced by the obsolete `-summary` option.

`-format column_list`

Formats the report according to the *column_list*. The *column_list* specifies which columns to display in the timing report and the order in which they appear. For example:

```
-format {hpin cell delay required arrival
required edge}
```

See Table 7-8 on page 1002 for a list of valid options.

Default: {`instance arc cell delay arr req`}

The default net format (with `-net` option) for the full path is: `{hpin edge net cell delay arr req}`.

If the `-unconstrained` option is specified, the `req` column is not displayed.

The `-format` option cannot be used with the `-path_type end` or `-path_type summary` options.

Note: To control the report_timing table fields, such as the min/max column width, see the `set_table_style` command. For example, the following command:
`set_table_style -name report_timing -max_width {100,,,,,}` resizes the first column to a maximum range of (100) and specifies a minimum width for that column based on the data in that column.
Default: For the minimum width is 5, and for the maximum width is 50.

Note: Do *not* include a blank space between commas "`,`" when specifying widths. If you are specifying a value for the width(s), make sure there is *no* leading or trailing blank space.

`full`

Generates a report that displays the full path with accompanying required time and slack calculation. This is the default path type. Control the report format using the `-format` option.

`full_clock`

If the path reported ends at a timing check, this option also reports the full clock path (`Other End Path`) in addition to the

full data path (`Timing Path`). If the reported path is a clock path, it also reports the full data path. By default, without this option, either the full data path or the full clock path is reported depending on the endpoint (data or clock).

`-net`

Adds a row for the net arc.
Default: The net arc is not shown, and the net delay is added to the following delay. This option also separates the cell delay from the wire delay.

Note: The `report_timing -net` command displays the net delays separate from the cell delays. However, the net delay is sometimes shown as 0.0.To get the net delays to show up, increase the report precision using the following global:
`set_global report_precision 5`

`-path_type {end | summary | full | full_clock}`

The `path_type` option lets you choose the format of the report by path type. The default format, if the `-path_type` option is not specified is `full`. See path type examples.

`summary`

Generates a summary report for each path consisting of a start point, endpoint, cause, slack, arrival time, required time, and phase. The `-format` option does not have any impact on this report format.

Note: The `-path_type summary` format differs from that produced by the obsolete `-summary` option. To generate a report in the same format as `-summary`, use `-path_type end`.

`-tcl_list`

Produces a report in a tcl list. This is useful for integrating timing with custom Tcl functions and customizing report generation.

Table 7-8 Report Timing—Column List Options

Option	Description
`addition`	Delay addition on pin. This delay typically comes from cycle addition or clock uncertainty. The column is labeled `Delay Addition`.

Table 7-8 Report Timing—Column List Options, *continued*

Option	Description
`arc`	The arc as described by the from pin, from pin edge, to pin, and to pin edge. For example, the arc from the rising edge of pin A to the falling edge of pin Z is reported as: `A ^->Z v`
`arrival`	Arrival time on the pin.
`bottle`	Number of paths in the report that pass through the pin.
`cell`	Cell name of the given pin's instance.
`clkordata`	Reports whether the given pin is on a clock path or a data path. The column is labeled `Clock/Data`.
`clkordatapin`	Reports whether the given pin expects a clock signal or a data signal. The column is labeled `Clock Expected`.
`delay`	Arc delay. If stolen slack at the arc output pin (output of transparent latches) is not zero, the `stolen` column is also displayed along with this column.
`delay_ast`	Reports if a delay has been asserted on the arc.
`direction`	Pin direction (IN, OUT).
`dont_modify_instance`	Reports whether the instance is `dont_modify`. If it is, YES appears in the column. Otherwise, this column is blank. This argument displays the `instance` column automatically.
`dont_modify_net`	Reports whether the net is `dont_modify`. If it is, YES appears in the column. Otherwise, this column is blank. This argument displays the `net` column automatically.
`edge`	Edge on the pin (^=rise, `v`=fall).
`eslack`	Worst early slack at the given pin.
`fanin`	Total fanin on the net.
`fanout`	Total fanout on the net.
`fpin`	From or source pin of the arc.
`from_edge`	Edge on the from, or source, pin.
`hpin`	Hierarchical pin name.
`instance`	Hierarchical name of the given pin's instance.

Table 7-8 Report Timing—Column List Options, *continued*

Option	Description
instance_location	Location (x,y) of the instance. If the design is not placed, the column is blank. This argument displays the `instance` column automatically.
`load`	Total capacitive load on a given pin.
`lslack`	Worst late slack at the given pin.
`net`	Hierarchical name of the net connected to given pin.
`phase`	Phase name on pin.
`pin`	Pin name of the given hierarchical pin.
`pinload`	Total capacitive load from pins on a given pin, including its own load.
`pin_location`	Location (x,y) of the pin. If the design is not placed, the column is blank.
`required`	Required time on the pin.
`slack`	Slack on the pin (this is the same for the entire path).
`slew`	Propagated slew at the given pin
`stolen`	Slack stolen (or the time given to previous stage) at the given pin. If the slack is not zero and the delay column is specified, this column is displayed by default. (only visible on the output of transparent latches.)
`to_edge`	Edge on the to, or sink, pin.
`tpin`	To, or sink, pin of the arc.
`wireload`	Total capacitive load from the wire on a given pin.
`wlmodel`	Wireload model used to calculate load on the net.

Examples

- The following command reports both late data paths, such as setup, recovery and so on, and late clock paths, such as hold, removal, pulse width and so on, in the same report.

  ```
  report_timing -late -check_clocks
  ```

  Note: This is the only way to report multiple checks on clock end points. Using the `-check_type` option reports only one type of check.

- The following command reports paths leading to the pin(s) specified by the *pin_list*:

```
report_timing -to O
Path 1: VIOLATED External Delay Assertion
Endpoint:   O     (v) checked with  leading edge of 'CK1'
Beginpoint: fd3/Q (v) triggered by  leading edge of 'CK1'
Other End Arrival Time      1.00
- External Delay            4.70
+ Phase Shift               5.00
= Required Time             1.30
- Arrival Time              1.70
= Slack Time               -0.40
     Clock Rise Edge                  1.00
     = Beginpoint Arrival Time        1.00
     +-------------------------------------------------------------+
     | Instance |     Arc     |  Cell | Delay | Arrival | Required |
     |          |             |       |       |  Time   |   Time   |
     |----------+-------------+-------+-------+---------+----------|
     |          | CK ^        |       |       |    1.00 |     0.60 |
     | an2      | B ^ -> Z ^  | AN2   |  0.00 |    1.00 |     0.60 |
     | fd3      | CP ^ -> Q v | FD1   |  0.70 |    1.70 |     1.30 |
     |          | O v         |       |  0.00 |    1.70 |     1.30 |
     +-------------------------------------------------------------+
```

- The following command shows a report similar to the first with the addition of net arc information.

```
report_timing -to O -net
Path 1: VIOLATED External Delay Assertion
Endpoint:   O     (v) checked with  leading edge of 'CK1'
Beginpoint: fd3/Q (v) triggered by  leading edge of 'CK1'
Other End Arrival Time      1.00
- External Delay            4.70
+ Phase Shift               5.00
= Required Time             1.30
- Arrival Time              1.70
= Slack Time               -0.40
     Clock Rise Edge                  1.00
     = Beginpoint Arrival Time        1.00
     +-------------------------------------------------------------+
     |  Pin   |  Edge |  Net  |  Cell | Delay | Arrival | Required |
     |        |       |       |       |       |  Time   |   Time   |
     |--------+-------+-------+-------+-------+---------+----------|
     | CK     |   ^   | CK    |       |       |    1.00 |     0.60 |
     | an2/B  |   ^   | CK    | AN2   |  0.00 |    1.00 |     0.60 |
     | an2/Z  |   ^   | CLK1  | AN2   |  0.00 |    1.00 |     0.60 |
     | fd3/CP |   ^   | CLK1  | FD1   |  0.00 |    1.00 |     0.60 |
     | fd3/Q  |   v   | O     | FD1   |  0.70 |    1.70 |     1.30 |
```

```
| O      |   v   | O     | latch |  0.00 |    1.70 |     1.30 |
+-------------------------------------------------------------+
```

- The following command displays the worst late path in the design. The format of the report is similar to the first example:

  ```
  report_timing
  ```

- The following command displays the worst late path to each violating endpoint that has a slack less than -1.0:

  ```
  report_timing -max_slack -1.0
  ```

- The following command displays all the late paths that end at port `out[2]` and that have negative slack up to a maximum of 1000 worst paths. If there are more than 1000 paths, only the 1000 worst paths are reported:

  ```
  report_timing -to out[2] -max_paths 1000 -max_slack 0.0
  ```

- The following command reports the worst late path that starts at in`[0]` and ends at `out[1]`:

  ```
  report_timing -from in[0] -to out[1]
  ```

- The following command displays the three worst paths that start at `in[1]` and end at `out[3]`. With reconvergent fanout, more than one path may exist:

  ```
  report_timing -from in[1] -to out[3] -max_paths 3
  ```

- The following command displays the ten worst paths. Only the paths between the specified pins are enumerated.With reconvergent fanout, more than one path may exist:

  ```
  report_timing -from i102/Z -to i123/A -max_paths 10
  ```

- The following command enumerates the worst ten paths through the given two pins, starting at the beginning points in the design and ending at the endpoints, similar to using the `-from` option:

  ```
  report_timing -through i102/Z -through i123/A -max_paths 10
  ```

- The following command reports the ten worst paths through blocks A and C, or blocks B and C. The path only has to satisfy one element in a through list, but all through lists must be satisfied. The example can be thought of as ((A or B) and C):

  ```
  report_timing -through {A/*B/*} -through {C/*} -max_paths 10
  ```

- The following command forces a path through pin A and pin B. Use the syntax shown in the example. Do not use `-through {A B}`:

  ```
  report_timing -through -through A -through B
  ```

- The following command reports the worst endpoint and the ten worst paths to that endpoint. Similar to `report_timing -max_paths 10 -to out[1]`, if `out[1]` is the worst endpoint:

  ```
  report_timing -max_points 1 -nworst 10
  ```

- The following command displays the clock path through the clock root to Launch flop:

  ```
  report_timing -through aBC_bs/bs_mex1/r2_00_q1_reg_10_/CK -to .... -path_type
  full_clock
  ```

  You will not see the path for the launch clock if you specify:

  ```
  report_timing -from aBC_bs/bs_mex1/r2_00_q1_reg_10_/CK
  ```

  Because you specified to report timing to start *from* the CK pin, that is exactly what it will report. The first command displays the path leading to the CK pin.

- The following command shows the `-unconstrained` option:

```
report_timing -unconstrained
```

```
+-----------------------------------------------+
| Report               | report_timing          |
|----------------------+------------------------|
| Options              | -unconstrained         |
+----------------------+------------------------+
| Date                 | 20010122.181608        |
| Tool                 | ac_shell               |
| Release              | v4.1-eng               |
| Version              | Jan 22 2001 15:25:02   |
+----------------------+------------------------+
| Module               | scid                   |
| Timing               | LATE                   |
| Slew Propagation     | WORST                  |
| Operating Condition  | NOM                    |
| PVT Mode             | max                    |
| Tree Type            | balanced               |
| Process              | 1.00                   |
| Voltage              | 5.00                   |
| Temperature          | 25.00                  |
| time unit            | 1.00 ns                |
| capacitance unit     | 1.00 pF                |
| resistance unit      | 1.00 kOhm              |
+-----------------------------------------------+
Path 1:Endpoint:   O2     (^)
Beginpoint: reg2/Q (v) triggered by trailing edge of 'vclk'
+-------------------------------------------------------+
| Instance |     Arc     |  Cell | Delay | Arrival |
|          |             |       |       |  Time   |
|----------+-------------+-------+-------+---------|
|          | clk1 v      |       |       |   10.00 |
| nand1    | B v -> Z ^  | ND2   |  0.17 |   10.17 |
| buf1     | A ^ -> Z ^  | BUF8A |  0.18 |   10.35 |
| reg2     | CP ^ -> Q v | FD1   |  0.74 |   11.09 |
| nand4    | A v -> Z ^  | ND2   |  0.07 |   11.16 |
|          | O2 ^        |       |  0.00 |   11.16 |
+-------------------------------------------------------+
```

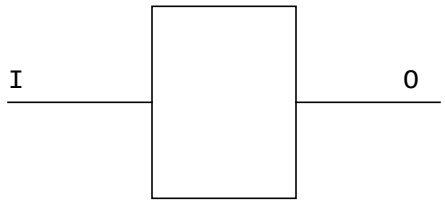
```
report_timing -unconstrained -summary
```

```
+-----------------------------------------------+
| Report               | report_timing          |
|----------------------+------------------------|
| Options              | -unconstrained         |
|                      | -summary               |
+----------------------+------------------------+
| Date                 | 20010122.181053        |
```

```
| Tool                | ac_shell              |
| Release             | v4.1-eng              |
| Version             | Jan 22 2001 15:25:02 |
+---------------------+-----------------------+
| Module              | scid                  |
| Timing              | LATE                  |
| Slew Propagation    | WORST                 |
| Operating Condition | NOM                   |
| PVT Mode            | max                   |
| Tree Type           | balanced              |
| Process             | 1.00                  |
| Voltage             | 5.00                  |
| Temperature         | 25.00                 |
| time unit           | 1.00 ns               |
| capacitance unit    | 1.00 pF               |
| resistance unit     | 1.00 kOhm             |
+-------------------------------------------+
+-------------------------------------------------------------+
|       Pin | Cause               | Arrival |         Phase |
|-----------+---------------------+---------+---------------|
|      O2 ^ | Unconstrained Path |   10.82 |        vclk N |
|      O1 v | Unconstrained Path |    0.70 |        vclk P |
|  reg2/D ^ | Unconstrained Path |    0.24 | Unconstrained |
|  reg3/D ^ | Unconstrained Path |    0.17 | Unconstrained |
|  reg1/D v | Unconstrained Path |    0.15 | Unconstrained |
|  reg4/D v | Unconstrained Path |    0.15 | Unconstrained |
| nand1/A v | Unconstrained Path |    0.00 | Unconstrained |
+-------------------------------------------------------------+
```

The following examples refer to Figure 7-5.

Figure 7-5 Conditions for Unconstrained and Constrained Paths



- Figure 7-5 shows the path from `I` to `O` is reported as an unconstrained path under the following conditions (constraints on `I` and `O`):
  - No constraints on either `I` or `O`
  - `set_input_delay -clock ck1 1.0 I`
    No `set_external_delay` constraint on `O`
  - `set_external_delay -clock ck1 1.0 O`
    No `set_input_delay` constraint on `I`
  - `set_input_delay 1.0 I`
    `set_external_delay 1.1 O`

- ❑
```
set_input_delay -clock ck1 1.0 I
set_data_required_time -clock ck2 1.0 O
```

- Shows the path from `I` to `O` is reported as a constrained path under the following conditions (constraints on `I` and `O`):

  - ❑
```
set_input_delay -clock ck1 1.0 I
set_external_delay -clock ck1 1.0 O
```

  - ❑
```
set_input_delay -clock ck1 1.0 I
set_external_delay -clock ck2 1.0 O
```

  - ❑
```
set_data_arrival_time -clock ck1 1.0 I
set_data_required_time -clock ck1 1.0 O
set_data_required_time -clock ck2 1.0 O
```

- Shows two paths from `I` to `O` and reports one as a constrained path and the other one as an unconstrained path under the following conditions (constraints on `I` and `O`):

  - ❑
```
set_input_delay -clock ck1 1.0 I
set_input_delay -clock ck2 1.0 I
set_data_required_time -clock ck2 1.0 O
```

  - ❑
```
set_input_delay 1.0 I
set_data_required_time 1.1 O
```

- Explains the `-late` and `-early` reports for clock signals. Consider the following instance of a flip-flop. First the `report cell instance -timing` command is used to show all of the checks for the instance. The results are given below:

```
+-------------------------------------------------------------------------------+
|                          Instance tmp_reg of FD1                              |
|-------------------------------------------------------------------------------|
|   Check    |   Sig |   Ref | Slack | Delay   |  Adjust |   Sig    |     Ref   |
|------------+-------+-------+-------+---------+---------+----------+-----------|
| PULSEWIDTH |  CP ^ |  CP v | 14.58 |    0.42 |    0.00 | c1(C)(P) |  c1(C)(P) |
| PULSEWIDTH |  CP v |  CP ^ | 14.63 |    0.37 |  -30.00 | c1(C)(P) |  c1(C)(P) |
|       HOLD |   D ^ |  CP ^ |  1.05 |   -0.05 |    0.00 | c1(D)(P) |  c1(C)(P) |
|       HOLD |   D v |  CP ^ |  1.93 |    0.07 |    0.00 | c1(D)(P) |  c1(C)(P) |
|      SETUP |   D ^ |  CP ^ | 28.86 |    0.14 |  -30.00 | c1(D)(P) |  c1(C)(P) |
|      SETUP |   D v |  CP ^ | 27.90 |    0.10 |  -30.00 | c1(D)(P) |  c1(C)(P) |
+-------------------------------------------------------------------------------+
```

  Since the hold check has worse slack than pulsewidth check, `report_timing` shows:

```
Path 1: MET Hold Check with Pin tmp_reg/D
Endpoint:   tmp_reg/CP (^) checked with  leading edge of 'c1'
Beginpoint: c1         (^) triggered by  leading edge of 'c1'
Other End Arrival Time    1.00
+ Hold                     0.05
+ Phase Shift              0.00
= Required Time            1.05
- Arrival Time             0.00
= Slack Time               1.05
```

And similarly, since pulsewidth check has worse slack than setup check, `report_timing -early` shows:

```
Path 1: MET PulseWidth Check with Pin tmp_reg/CP
Endpoint:   tmp_reg/CP (v) checked with  leading edge of 'c1'
Beginpoint: c1         (v) triggered by trailing edge of 'c1'
Other End Arrival Time    0.00
+ PulseWidth              0.42
+ Phase Shift             0.00
= Required Time           0.42
- Arrival Time           15.00
= Slack Time            14.58
```

Default: `-late`

- Shows the transparent latch reports using the default global setting `latch_time_borrow_mode budget`. For a comparison of reports using `max_borrow` analysis mode, see Analyzing Latch-Based Designs in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

```
report_timing -to ld2/D
Path 1: MET Latch Borrowed Time Check with Pin ld2/G
Endpoint:   ld2/D (v) checked with  leading edge of 'CK1'
Beginpoint: I     (^) triggered by  leading edge of 'CK1'
Other End Arrival Time      1.00
+ Time Borrowed             1.81
+ Phase Shift               0.00
= Required Time             2.81
- Arrival Time              2.10
= Slack Time                0.71
     Clock Rise Edge                  1.00
     + Input Delay                    1.00
     = Beginpoint Arrival Time        2.00
     +-------------------------------------------------------------+
     | Instance |    Arc     |  Cell | Delay | Arrival | Required |
     |          |            |       |       |  Time   |   Time   |
     |----------+------------+-------+-------+---------+----------|
     |          | I ^        |       |       |    2.00 |     2.71 |
     | inv0     | A ^ -> Z v | IV    |  0.10 |    2.10 |     2.81 |
     | ld2      | D v        | LD1   |  0.00 |    2.10 |     2.81 |
     +-------------------------------------------------------------+
report_timing -through ld2/Q
Path 1: MET External Delay Assertion
Endpoint:   O2    (^) checked with  leading edge of 'CK1'
Beginpoint: ld2/Q (^) triggered by  leading edge of 'CK1'
Other End Arrival Time      1.00
- External Delay            1.00
+ Phase Shift               5.00
```

```
= Required Time             5.00
- Arrival Time              3.40
= Slack Time                1.60
     Clock Rise Edge                 1.00
     = Beginpoint Arrival Time       1.00
     +-------------------------------------------------------------------+
     | Instance |    Arc     |  Cell | Delay |  Time  | Arrival | Required |
     |          |            |       |       | Given  |  Time   |   Time   |
     |          |            |       |       |  To    |         |          |
     |          |            |       |       | Start  |         |          |
     |          |            |       |       | Point  |         |          |
     |----------+------------+-------+-------+--------+---------+----------|
     |          | CK ^       |       |       |        |    1.00 |     2.60 |
     | an2      | B ^ -> Z ^ | AN2   |  0.00 |        |    1.00 |     2.60 |
     | ld2      | D ^ -> Q ^ | LD1   |  0.46 |  1.94  |    3.40 |     5.00 |
     |          | O2 ^       |       |  0.00 |        |    3.40 |     5.00 |
     +-------------------------------------------------------------------+
```

- Changes the format of the report. Compare the output to that shown in the first example (`report_timing -to out`):

```
report_timing -to O -net -format {instance arc net load delay arrival}
Path 1: VIOLATED External Delay Assertion
Endpoint:   O     (v) checked with  leading edge of 'CK1'
Beginpoint: fd3/Q (v) triggered by  leading edge of 'CK1'
Other End Arrival Time      1.00
- External Delay            4.70
+ Phase Shift               5.00
= Required Time             1.30
- Arrival Time              1.70
= Slack Time               -0.40
     Clock Rise Edge                 1.00
     = Beginpoint Arrival Time       1.00
     +-----------------------------------------------------------+
     | Instance |     Arc     |  Cell | Delay | Arrival | Required |
     |          |             |       |       |  Time   |   Time   |
     |----------+-------------+-------+-------+---------+----------|
     |          | CK ^        |       |       |    1.00 |     0.60 |
     | an2      | B ^ -> Z ^  | AN2   |  0.00 |    1.00 |     0.60 |
     | fd3      | CP ^ -> Q v | FD1   |  0.70 |    1.70 |     1.30 |
     |          | O v         |       |  0.00 |    1.70 |     1.30 |
     +-----------------------------------------------------------+
```

Related Information

`report_net`

`report_ports`

reset_capacitance_limit

```
reset_capacitance_limit [-min | -max] [-instance instance_list]
     [-port port_list]
```

Resets instance-specific or port-specific design rule constraints. Design rules set on a hierarchical instance applies to all the pins contained in that hierarchical instance.

Use the `set_global max_capacitance_limit` or the `set_global min_capacitance_limit` to set context specific design rules. If either one of these globals has also been set, the most constraining value is used.

Options and Arguments

`-instance` *`instance_list`*
: Specifies the list of hierarchical instances where the limit is reset.

`-min | -max`
: Resets the minimum (or maximum) instance-specific or port-specific design rule constraints.

`-port` *`port_list`*
: Specifies the list of ports where the limit is reset.

Example

The following command resets the maximum instance-specific or port-specific design rule constraints:

```
reset_capacitance_limit -max -instance [I1}
```

Related Information

`reset_port_capacitance_limit`

`reset_slew_time_limit`

`set_capacitance_limit`

`set_global max_capacitance_limit`

`set_global min_capacitance_limit`

`set_port_capacitance_limit`

`set_slew_time_limit`

reset_capacitance_unit

```
reset_capacitance_unit
```

Resets the session capacitance unit. The default target technology units become applicable for the session.

Example

The following command resets the session capacitance unit to the default target technology units:

```
reset_capacitance_unit
```

Related Information

`reset_time_unit`

`set_capacitance_unit`

`set_time_unit`

reset_clock_gating_check

```
reset_clock_gating_check [-setup | -hold] [-rise | -fall]
     [-pin instance_or_pin_list] [-clock clock_list [-lead | -trail]]
```

Resets the timing margin specified with `set_clock_gating_check` back to the default value.

Options and Arguments

`-clock` *`clock_list`*

Specifies the clock signals for which the command resets a clock-gating assertion to the default. The timing margins for clock-gating checks for any instance that gates a clock signal having such an assertion are reset.

`-lead | -trail`

Resets the specified timing margin for only those clock-gating checks for which the reference clock edge is the leading (or trailing) edge. If neither option is specified, the timing margins are reset on both leading and trailing edges.

Use the `-lead` or `-trail` options only with the `-clock` option.

`-pin` *`instance_or_pin_list`*

Lists the pins for which the timing margins (setup/hold values) for clock-gating checks are reset. If the name of a clock-gating instance is given in the argument list, the command resets all inputs of the instance. Likewise, if the clock input pin of the instance is given, the command again resets all the instance inputs.

`-rise | -fall`

Resets the specified timing margin for the rising (or falling) delay only. If neither option is specified, the default resets both rising and falling.

`-setup | -hold`

Resets the timing margin (setup value or hold value) for the clock-gating setup check back to the default setup or hold value.

Example

The following command resets the setup value for the clock-gating setup check and the clock signal `CLK1` back to the default value:

```
reset_clock_gating_check -setup -clock CLK1
```

Related Information

`set_clock_gating_check`

Setting Clock Gating Setup and Hold Checks in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

reset_clock_info_change

```
reset_clock_info_change
     To reset a data signal use: reset_clock_info_change
     [-lead | -trail] [-early | -late] [-rise | -fall]
     -clock clock_name
     pin_list
     ||
     To reset a clock signal use: reset_clock_info_change
     [-pos | -neg] [-early | -late] [-rise | -fall]
     -clock clock_name
     pin_list
```

Changes the clock/data information for paths going through the specified pins for the downstream logic. This command removes assertions previously given by the `set_clock_info_change` command.

Tip

Remove all clock info assertions on pin Z regardless of clock name or current signal type with the following command:

```
remove_assertions -type clock_info_change Z
```

Options and Arguments

`-clock` *`clock_name`*

Specifies the clock waveform to apply or remove from the pin.

`-early | -late`

Resets early or late arrival of the signal.
Default: Both

`-lead | -trail`

Indicates that the signal at the specified pin is currently (before the reset) recognized as a data signal with the specified association (`lead` or `trail`) with respect to the specified clock *`clock_name`*.
Default: `-lead`

`pin_list`

Associates paths from the pin list with the specified clock *`clock_name`*. The pin list can be ports or instance pins. These can be object IDs or hierarchical names relative to the current module.

`-pos | -neg`

Indicates that paths are recognized as clock paths prior to the reset and that the clock has the specified polarity. This option is required to reset a clock path. If neither `-pos` or `-neg` is specified, the path is recognized as a data path prior to the reset.

Note: Because the signal cannot be both a clock signal and a data signal, the `-pos` | `-neg` options cannot be used with the `-lead` | `-trail` options.

`-rise | -fall`

Resets the rising or falling edge of the signal.
Default: Both

Example

The following command changes the output of an AND gate from a clock signal type back to the data signal type that it had before being changed to clock:

```
reset_clock_info_change -clock clk -pos U1/A1/O
```

Related Information

`remove_assertions`

`set_clock_info_change`

reset_clock_insertion_delay

```
reset_clock_insertion_delay {([-source] [-lead | -trail] [-early | -late]
     [-pvt {min | typ | max}] list_of_clocks) | ([-source] [-clock clock_name]
     [-early | -late] [-pvt {min | typ | max}] [-rise | -fall] -pin list_of_pins)}

     HINT: Use [-early | -late] only with -source. Use -pvt only for network delay
     (no -source).
```

Resets the clock insertion delay (either source or network) as previously specified with the `set_clock_insertion_delay` assertion.

Tip

Alternatively, you can use the following command to remove both source and network clock insertion delay assertions from pin `in1`:

```
remove_assertions -type clock_insertion_delay in1
```

Options and Arguments

`-clock` *`clock_name`*

Resets only the source insertion delay that is caused by the named clock waveform. Used when more than one clock generator is connected to the port.

`-early | -late`

Specifies clock arrival time with respect to the early (setup) or the late (hold) time of the clock signal. If neither option is specified, the default is both `-early` and `-late`. Use these options only with the `-source` option.

`-lead | -trail`

Resets insertion delay at the leading or trailing edge of the clock waveform.
Default: Both edges

`list_of_clocks`

Lists clock waveform names to associate with the clock insertion delay reset.

`-pin` *`list_of_pins`*

Lists pins to associate with the clock insertion delay reset. The pin can be a port or an instance pin. These can either be object IDs or hierarchical names relative to the current module.

For internal pins, both the source and network delay must be specified.

`-pvt {min | typ | max}`
Resets the clock insertion delay for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces (`{}`) and separate the values by spaces.
Default: The specified clock insertion delay applies to all three PVT corners. This option cannot be used with the `-source` option.

`-rise | -fall`
Resets the insertion delay at the rising or falling edge of the clock.

`-source`
Specifies that the delay being reset is a source delay. Without this option, the delay being reset is considered a network delay.

Example

The following command resets the insertion delay at the falling edge of the clock and lists the pins `clkB clkD` associated with the clock insertion delay reset:

```
reset_clock_insertion_delay -fall -pin {clkB clkD}
```

Related Information

`remove_assertions`

`set_clock_insertion_delay`

reset_clock_root

```
reset_clock_root [-clock clk_signame] [-pos] [-neg] pin_list
```

Removes previously given clock root assertions. The command behavior follows that of `set_clock_root`, except both `-pos` and `-neg` can be given at the same time. If the `-clock` option is absent, all clock root assertions are removed.

Tip

Alternatively, you can use the following command to remove all clock root assertions on a pin `x`:

```
remove_assertions -type clock_root x
```

Options and Arguments

`-clock` *`clk_signame`*

Specifies the name of the ideal clock signal as specified by the previous `set_clock_root` assertion.

`-neg`

Removes a negative ideal clock signal. If neither `-pos` nor `-neg` is specified, the default is `-pos` (rising edge precedes falling edge).

`pin_list`

Lists the pins to remove from association with an ideal clock signal. The pin can be a port or an instance pin. These can either be object IDs or hierarchical names relative to the current module.

`-pos`

Removes a positive ideal clock signal. If neither `-pos` nor `-neg` is specified, the default is `-pos` (rising edge precedes falling edge).

Example

The following command removes a previously given clock root assertion for the `master` ideal clock signal and removes a negative `negclk` and a positive `posclk` ideal clock signal:

```
reset_clock_root -clock master -neg -pos {negclk posclk}
```

Related Information

`remove_assertions`

`set_clock_root`

reset_clock_uncertainty

```
reset_clock_uncertainty [-ideal | -propagated] [-early | -late]
     {([-clock_from clk_from] [-clock_to clk_to]
     [-edge_from {leading | trailing}] [-edge_to {leading | trailing}]) |
     (-pin list_of_clock_tree_pins [-rise] [-fall] [-to])}
```

Resets the assertions that were made by previous `set_clock_uncertainty` commands.

Options and Arguments

`-clock_from` *`clk_from`*

Specifies the starting point of the paths for which the uncertainty is reset. The *`clk_from`* argument is the name of clock signal with respect to which the uncertainty was specified.

If the `-clock_from` option is omitted then the uncertainty reset applies to all the paths ending at the registers clocked by the `-clock_to` signal.

`-clock_to` *`clk_to`*

Specifies the endpoint of the paths for which the uncertainty is reset. The uncertainty reset applies to all paths starting from the `-clock_from` signal and ending at the registers clocked by `-clock_to` signal.

If the `-clock_to` option is omitted, then the uncertainty reset applies to all paths starting at the `-clock_from` signal.

If both the options `-clock_from` and `-clock_to` are omitted, then the uncertainty reset applies to all clocked paths in the design, including combinational paths constrained by the `set_input_delay` and `set_external_delay` times associated with the clock.

`-early | -late`

Specifies whether the uncertainty is reset with respect to the early times or the late times. In other words, whether the reset applies to hold (`-late`) or setup (`-early`) checks on the clock signals
Default: Both `-early` and `-late`

`-edge_from {leading | trailing}`

Specifies whether the triggering (launching) edge of the clock is

a leading or a trailing edge.
Default: The `leading` edge.

`-edge_to {leading | trailing}`
Specifies whether the capturing edge at the destination register is a leading or a trailing edge.
Default: The `leading` edge

`-ideal | -propagated`
Specifies whether the uncertainty applies to the `ideal` clock propagation mode or the `propagated` mode of analysis. If neither option is given, the uncertainty applies in both ideal and propagated modes by default.

`-pin` *`list_of_clock_tree_pins`*
Resets the uncertainty for the given list of clock pins. Each pin can be a port or an instance pin. It can either be given as an object ID or a hierarchical name relative to the current module. Cannot be used with `-clock_from` or `-clock_to` options.

`-rise | fall`
Specifies the rising (or falling) edge of the clock pins on which the uncertainty is reset. For each edge, the assertion is reset only if that edge fans out to the data and the clock pin of a register or latch. With the `-to` option, the assertion is reset if that edge fans out to the clock pin of a register or latch.
Default: Both edges.

`-to`
Specifies that the uncertainty reset applies to all the registers and latches whose clock pin is in the transitive fanout of a clock pin from the *`list_of_clock_tree_pins`*.

Example

The following command resets an uncertainty on a clock pin (`clk_in`) only for propagated mode, which is how the uncertainty was originally set:

```
reset_clock_uncertainty -propagated -pin clk_in
```

Related Information

`remove_assertions`

set_clock_uncertainty

reset_constant_for_timing

`reset_constant_for_timing {-bidi_input | -bidi_output}` *`pin_list`*

Removes previously asserted constants on the specified pin(s). Once a constant has been reset, CTE returns to analyzing the system by calculating and propagating the arrival times (and in turn computing the required times) at the pin.

Options and Arguments

`-bidi_input | -bidi_output`
: Resets the constant that is set on the input or output part of bidirectional pins.
Default: Resets the constant on the input part.

`pin_list`
: Specifies a single pin or multiple pins to reset. To specify multiple pins, enclose the list with curly braces ({}) and separate the pin information with white spaces.

Example

The following command removes previously asserted constants on pin `X20/Z`:

```
reset_constant_for_timing X20/Z
```

Related Information

`get_constant_for_timing`

reset_dcl_calculation_mode

```
reset_dcl_calculation_mode
```

Resets the current Delay Calculation Library (DCL) calculation mode to the default mode of `worst_case`.

The library developer defined the calculation modes in the DCL library to model three process conditions, `best_case`, `nominal_case`, and `worst_case` within one library. Refer to the vendor documentation for more information.

Note: Process, temperature, and voltage are reset independently of the calculation mode using the `reset_operating_parameter` command.

Example

The following command resets the DCL calculation mode to `worst_case`:

```
reset_dcl_calculation_mode
```

Related Information

`get_dcl_calculation_mode`

`get_operating_parameter`

`load_dcl_rule`

`set_dcl_calculation_mode`

`set_operating_parameter`

reset_dcl_functional_mode

`reset_dcl_functional_mode [-group` *`group_name`*`] [-mode` *`mode_name`*`]` *`instance_list`*

Resets the current DCL functional mode to the library default.

Options and Arguments

`-group` *`group_name`*
: Specifies the functional mode group name.

`instance_list`
: Lists the instances for which the functional mode needs to be reset.

`-mode` *`mode_name`*
: Specifies the functional mode name.

Example

The following command resets the current DCL functional mode to the library default for group `rw`, functional mode `read`, for instances `A1/B3/C2 A1/B2/C2`:

```
reset_dcl_functional_mode -group rw -mode read {A1/B3/C2 A1/B2/C2}
```

Related Information

`get_dcl_functional_mode`

`get_dcl_functional_mode_array`

`load_dcl_rule`

`set_dcl_functional_mode`

reset_dcl_level

```
reset_dcl_level
```

Resets the performance level for DCL-based delay calculations to the default level of 0 (lower accuracy but faster run time).

Related Information

set_dcl_level

reset_default_slew_time

```
reset_default_slew_time [-early | -late] [-rise | -fall]
```

Resets the default slew time for the ports to a value of 0.0. The system uses this slew value for any input or bidirectional port for which slew or drive assertions are not specified.

The `reset_default_slew_time` command is ignored for a port if `set_slew_time`, `set_drive_cell`, or `set_drive_resistance` constraints have been set for that port.

Options and Arguments

`-late | -early`
: Indicates whether the slew times being reset are with respect to the late (data setup) or the early (data hold) checks. If neither option is specified the default is to use both.

`-rise | -fall`
: Specifies that the slew time should be reset on the rising edge or the falling edge of the signal. If neither edge is specified, the default is to use both.

Example

The following command resets the default slew time with respect to early (data hold) checks on the rising edge of the signal previously set with the `set_default_slew_time` command:

```
reset_default_slew_time -rise -early 10.0
```

Related Information

`set_default_slew_time`

`set_drive_cell`

`set_drive_resistance`

`set_slew_time`

reset_disable_cell_timing

```
reset_disable_cell_timing [-library library_name] [-cell cell_name]
     [{-from_pin | -from_rise | -from_fall} from_pin_name ]
     [{-to_pin | -to_rise | -to_fall} to_pin_name]
```

Re-enables timing arcs that were disabled by the `set_disable_cell_timing` command. The timing arcs from or to internal pins inside IP cells can also be re-enabled.

Options and Arguments

`-cell` *`cell_name`*
: Specifies the cell in the library for which timing arcs are re-enabled.

`-from_pin|-from_rise|-from_fall` *`from_pin_name`*
: Re-enables timing arcs originating from the specified `from_pin` *`from_pin_name`* option. Re-enable timing arcs that have been disabled using either the `from_rise` option or the `from_fall` option. All arcs from input pins specified with the `-from_pin` *`from_pin_name`* option are re-enabled.

`-library` *`library_name`*
: Specifies the library name which contains the cell whose arcs are to be re-enabled. If not specified, the default is the target technology library.

`-to_pin|-to_rise|-to_fall` *`to_pin_name`*
: Re-enables disabled timing arcs specified for the source pin of the timing arc. You can re-enable disabled cell arcs with specific edges using either the `to_rise` option or the `to_fall` option. All arcs to output pins specified with the `-to_pin` *`to_pin_name`* option are re-enabled.

Example

The following command re-enables arcs with a specific transition. Arcs originating from rising pin A and falling to pin Y are re-enabled:

```
reset_disable_cell_timing -cell XORS -from_rise A -to_fall Y
```

Related Information

set_constant_for_timing

set_disable_cell_timing

set_disable_timing

set_false_path

reset_disable_clock_gating_check

```
reset_disable_clock_gating_check [-from from_pins] [-to to_pins]
     [-bidi_input_from | -bidi_output_from] [-bidi_input_to | -bidi_output_to]
```

Resets all disabled clock gating checks to, from, or between the specified pins.

Options and Arguments

`-bidi_input_from | -bidi_output_from`
: Specifies the assertion on the input or output part of the `from` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_to | -bidi_output_to`
: Specifies the assertion on the input or output part of the `to` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-from` *`from_pins`*
: Specifies the sink pin (clock pin) of the clock gating check to be disabled. If the `-from` option is used without the `-to` option, all clock gating checks that reference *`from_pins`* are disabled. Either `-from` or `-to` must be used with this command. Specifying both `-from` and `-to` identifies a check with specific start and endpoints.

`-to` *`to_pins`*
: Specifies the source (data enable signal) pin of the clock gating check to be disabled.

  If the `-to` option is used without the `-from` option, all clock gating checks with *`to_pins`* for data enable signal (clock gating pin) are disabled.

Related Information

`reset_clock_gating_check`

`set_clock_gating_check`

`set_disable_clock_gating_check`

reset_disable_timing

```
reset_disable_timing -from from_pins -to to_pins
     [-bidi_input_from | -bidi_output_from] [-bidi_input_to | -bidi_output_to]
```

Deletes the assertion created by `set_disable_timing` with the exact same pin(s). For example, you cannot disable all the arcs to a pin individually and then use `reset_disable_timing -to` *`pin`* to reset all the arcs.

Deletes assertions applied to check arcs and delay arcs for the pin or arc.

Tip

To reset all of the arcs, use the same number of commands with the exact same pin(s), only replacing the `set_disable_timing` command with the `reset_disable_timing` command.

Options and Arguments

`-bidi_input_from | -bidi_output_from`
: Specifies the assertion on the input or output part of the `from` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_to | -bidi_output_to`
: Specifies the assertion on the input or output part of the `to` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-from` *`from_pins`*
: Specifies the source pin of the timing arc or the clock reference pin of the check arc to be re-enabled. If the `-from` option is used without the `-to` option, all paths originating from the *`from_pins`* are enabled. All check arcs with the specified from pin as reference pin are also re-enabled.

  Use either the `-from` or `-to` option with this command. Specifying both the `-from` and `-to` options identifies a path with specific start and endpoints.

  The `-from` and `-to` options can be intermediate hierarchical boundaries.

`-to` *`to_pins`*

Specifies the sink pin of the timing arc or the data pin of the check arc to be re-enabled.

If the `-to` option is used without the `-from` option, all paths ending in *`to_pins`* are enabled.

Examples

- The following command re-enables all timing arcs from the `Q` pin of instance `I1`:

  ```
  reset_disable_timing -from I1/Q
  ```

- The following command re-enables the timing arc from the input part of bidirectional pin `A` to the output part of bidirectional pin `B`:

  ```
  reset_disable_timing -from A -to B -bidi_input_from -bidi_output_to
  ```

- The following command re-enables the timing arc from the `A` pin to `Z` pin of instance `NAND2`:

  ```
  reset_disable_timing -from NAND2/A -to NAND2/Z
  ```

- The following command re-enables all timing arcs which pass through the hierarchical port, `port1`, of module `mod` and end at an input or bidirectional pin:

  ```
  reset_disable_timing -from mod/port1
  ```

- The following command re-enables all timing check arcs with `r/D` as data signal and `r/CP` as check reference pin:

  ```
  reset_disable_timing -from r/CP to r/D
  ```

Related Information

`set_disable_cell_timing`

`set_disable_timing`

`set_false_path`

reset_drive_cell

```
reset_drive_cell [-early | -late] [-clock clock_signame]
     [-lead | -trail | -pos | -neg] port_list
```

Removes the drive cell assertion, previously set by the `set_drive_cell` command, from the named ports.

Tip

You can more easily remove all drive cell assertions from port `IN` with this command:

```
remove_assertions -type drive_cell IN
```

Options and Arguments

`-clock` *`clock_signame`*

Specifies the ideal clock that was controlling the signal.

`-early`

Specifies that the driver provided a signal for early mode analysis (data hold check).

`-late`

Specifies that the driver provided a signal for late mode analysis (data setup check).

`-lead | -trail | -pos | -neg`

Specifies that the slew time was for a data signal triggered by the leading or the trailing edge of the ideal clock.
Default: `-lead`.

For a clock signal, `-pos` and `-neg` specify that the slew time was applied to an actual clock that has positive or negative polarity with respect to the ideal clock.

`port_list`

Specifies the list of ports for which the driver cell is to be removed.

Example

The following command finds the previously set drive cell `input1`, previously set by the `set_drive_cell` command, then removes the drive cell assertion from the named ports:

```
reset_drive_cell -clock clk [ find -input -port * ]
reset_drive_cell input1
```

Related Information

`remove_assertions`

`set_drive_cell`

reset_drive_resistance

```
reset_drive_resistance -clock clk_signame [-lead | -trail | -pos | -neg]
     [-rise | -fall] [-early | -late] port_list
```

Removes assertions previously specified by the `set_drive_resistance` command.

Options and Arguments

`-clock` *`clk_signame`*
: Specifies the name of the clock signal.

`-early | -late`
: Specifies that the drive resistance should be removed from the early arrival time (data hold time) or late arrival time (data setup time).

`-lead | -trail | -pos | -neg`
: Indicates that the signal at the specified port must be recognized as a data signal with the specified association (`lead` or `trail`) with respect to the specified clock *`clock_name`*.
Default: `-lead`.

  For a clock signal, `-pos` and `-neg` specify that the slew time should be applied to an actual clock that has positive or negative polarity with respect to the ideal clock.
Default: `-pos`.

`-rise | -fall`
: Specifies that the drive resistance is removed only from the rising edge or the falling edge transition at the input port. If neither rise nor fall options are specified then the resistance is removed from both transitions at the input port.

`port_list`
: Specifies the list of ports for which drive resistance is removed.

Example

```
reset_drive_resistance [-input -port * ]
```

Related Information

`set_drive_cell`

`set_drive_resistance`

`set_slew_time`

reset_external_delay

```
reset_external_delay [-clock clk_name] [-lead] [-trail] [-early] [-late]
     [-rise] [-fall] pin_list
```

Resets previously given external delay assertions. Its behavior follows that of the `set_external_delay` command, except both the `-lead` and `-trail` options can be given at the same time.

Tip

Remove all external delay assertions (without specifying the clocks) for a pin x using the following command:

```
remove_assertions -type external_delay x
```

Options and Arguments

`-clock` *`clock_name`*

Specifies the name of the clock.
Default: The asynchronous (@) clock.

`-early | -late`

Specifies that the constraint refers to the early (data hold) or late (data setup) times. If both `-late` and `-early` options are omitted, then the external delay is reset for both early and late times.

`-lead`

Resets the external delay for the leading edge of the clock. If neither `-lead` nor `-trail` is specified, the default is `-lead`.

`pin_list`

Specifies a single pin or multiple pins for which the external delay is reset. To specify multiple pins, enclose the list with curly braces ({}) and separate the pin information with white space.

`-rise | -fall`

Resets the external delay for the rising edge or falling edge at the input port. If both `-rise` and `-fall` options are omitted, the external delay is reset on both the edges.

`-trail`

Resets the external delay for the trailing edge of the clock. If neither option is provided, the default is `-lead`.

Example

The following command resets the external delay on the `out` pin of the clock (`clk1`) that was previously set by a `set_external_delay` constraint:

```
reset_external_delay -clock clk1 -lead out
```

Related Information

`remove_assertions`

`set_external_delay`

reset_fanout_load

```
reset_fanout_load port_list
```

Resets the fanout load on the ports of a cell to the default values specified in the library. The fanout load previously specified by the `set_fanout_load` assertion is removed. Fanout loads are only used to enforce the design rule checks and have no effect on timing. Setting (or resetting) the port capacitance affects timing analysis.

Options and Arguments

port_list

Specifies the list of ports for which the assertion is removed.

Example

```
reset_fanout_load [find -port -output data*]
```

Related Information

`remove_assertions`

`set_fanout_load_limit`

reset_fanout_load_limit

```
reset_fanout_load_limit port_list
```

Resets the fanout load limit (maximum value) on the ports of a cell. Fanout load limits are only used to enforce the design rule checks; they do not affect timing analysis. Setting (or resetting) port capacitance limit affects timing analysis.

The design rule requirement of a maximum fanout load value is set using the `set_global fanout_load_limit` global. For the specified ports, the `reset_fanout_load_limit` command restores the global default fanout load limit.

Options and Arguments

`port_list`

Specifies the list of ports for which the assertion is removed.

Examples

```
reset_fanout_load_limit [find -port -output data*]
reset_fanout_load_limit [find -port -input reset]
```

Related Information

`remove_assertions`

`set_fanout_load_limit`

reset_feedback_loop_snipped_arcs

```
reset_feedback_loop_snipped_arcs {[-from from_pin] | [-to to_pin]}
```

Restores the arcs that the tool disabled in order to time the loops in the design. The tool chooses the arc at which to break a loop somewhat arbitrarily. This command lets you reactivate the timing arcs that were previously snipped by the tool. If there was no snipped arc, a warning message is issued.

After issuing this command, you must disable (snip) a different arc to break the loop where you want it broken. Use the `set_disable_timing` command to break feedback loops. if the loop is not broken, a warning message is issued by the `check_timing` command.

Options and Arguments

The arcs to be reactivated must be chosen using `-from`, or `-to`, or both options.

`-from` *`from_pin`*

Resets all arcs out of the *`from_pin`* which were snipped by the tool.

`-to` *`to_pin`*

Resets all arcs into the *`to_pin`* which were snipped by the tool.

Example

```
reset_feedback_loop_snipped_arcs -from g1/Z
```

Related Information

`check_timing`

`set_disable_timing`

reset_functional_mode

`reset_functional_mode [-group` *`group_name`*`] [-mode` *`mode_name`*`]` *`instance_list`*

Resets the DCL or the STAMP functional mode group and name on hierarchical instances. To determine what modes are available for an instance, use the `report_functional_mode` command.

This command has no effect if the mode is already inactive. If you reset a mode which is the only active mode, then all modes become active. For example, consider a mode group `rw` with two modes `read` and `write`. If you set `read` and then reset it, both `read` and `write` modes become active again, as if no modes had been set. This behavior differs from PrimeTime which leaves all modes inactive if one is set and then later reset.

Note: Currently there is no command to make all modes inactive.

Options and Arguments

`-group` *`group_name`*
: Specifies the string value of the functional mode group name.

`instance_list`
: Specifies the list of instances for which the functional mode needs to be reset.

`-mode` *`mode_name`*
: Specifies the string value of the functional mode name.

Example

The following command resets the DCL or the STAMP functional mode group `rw` and name `read` on hierarchical instances `A1/B3/C2`:

```
reset_functional_mode -group rw -mode read A1/B3/C2
```

Related Information

`report_functional_mode`

`set_functional_mode`

reset_generated_clock

```
reset_generated_clock target_pin_list
```

Removes generated clock assertions previously created using the `set_generated_clock` command. This command removes all the generated clock assertions on each pin in the *`target_pin_list`*. However, the generated clock waveforms themselves are not removed from the system, because they may be referenced by other generated clocks in the design. A generated clock can be reused by a subsequent `set_generated_clock` command.

Options and Arguments

`target_pin_list`

Specifies the list of pins where the generated clock assertion is removed. The signal propagating downstream from the pins is no longer associated with the generated clock.

Example

The following command removes generated clock assertions on `pinY` previously created using the `set_generated_clock` command:

```
reset_generated_clock pinY
```

Related Information

`set generated clock`

reset_ideal_net

`reset_ideal_net` *`net_name_or_id`*

Removes the ideal assertion from the net. The net delays revert to their Steiner, annotated, or wire-load model (WLM) values.

Options and Arguments

`net_name_or_id`

Specifies one or more net names (or ids) that are not to be considered ideal nets anymore.

Example

The following command removes the ideal assertion from nets `clk1 clk2`:

```
reset_ideal_net {clk1 clk2}
```

Related Information

`set ideal net`

reset_input_delay

```
reset_input_delay [-clock clk_name] [-lead] [-trail] [-early] [-late] [-rise]
     [-fall] pin_list
```

Resets previously given input delay assertions. Its behavior follows that of the `set_input_delay` command, except both the `-lead` and `-trail` options can be given at the same time.

Tip

To remove all input delay assertions (without specifying the clocks) for a pin `x`, use:

```
remove_assertions -type input_delay x
```

Options and Arguments

`-clock` *`clock_name`*

Specifies the name of the clock.
Default: The asynchronous (@) clock.

`-early | -late`

Specifies that the constraint refers to the early (data hold) or late (data setup) times. If both `-late` and `-early` options are omitted, then the input delay is reset for both early and late times.

`-lead`

Resets the input delay for the leading edge of the clock. If neither `-lead` nor `-trail` is specified, the default is `-lead`.

`pin_list`

Specifies a single pin or multiple pins for which the input delay is reset. To specify multiple pins, enclose the list with curly braces (`{}`) and separate the pin information with white space.

`-rise | -fall`

Resets the input delay for the rising edge or falling edge at the input port. If both `-rise` and `-fall` options are omitted, the input delay is reset on both the edges.

`-trail`

Resets the input delay for the trailing edge of the clock. If neither option is provided, the default is `-lead`.

Example

```
reset_input_delay -clock clk1 -lead -trail in
```

Related Information

remove_assertions

reset_num_external_sinks

`reset_num_external_sinks` *`port_list`*

Resets the number of external sinks previously specified by the `set_num_external_sinks` command to the default value of `1`.

Options and Arguments

`port_list`

Specifies the list of ports to have 1 external sink.

Example

The following command resets the number of external sinks to the default value of 1 for port `out`:

```
reset_num_external_sinks out
```

Related Information

`remove assertions`

`set num external sinks`

reset_num_external_sources

`reset_num_external_sources` *`port_list`*

Resets the constraint for the number of external sources previously specified by the `set_num_external_sources` command to the default value of `1`.

Options and Arguments

`port_list`

Specifies the list of ports to have 1 external source.

Example

The following command resets the constraint for the number of external sources to the default value of 1 for port `in`:

```
reset_num_external_sources in
```

Related Information

`remove assertions`

`set num external sources`

reset_operating_condition

```
reset_operating_condition [-library library_name] [-pvt {min | typ | max}]
     operating_condition_name
```

Removes the operating condition (process, temperature, voltage, and tree-type) from the current design. No operating condition is used by the mapping and optimization commands.

Each technology library contains one or more operating conditions. Each condition is identified by name, which specifies a set of process, temperature, voltage, and tree-type conditions as the operating condition. This information is used to calculate accurate cell delays from the nominal cell delays and the k-factors (also called derating factors) from either linear or nonlinear models.

Get a list of all operating conditions in a technology library along with their PVT and OC type values by using the `report_library` command.

Options and Arguments

`-library` *`library_name`*
: Specifies the technology library containing the named operating condition.

`operating_condition_name`
: Specifies the name of the operating condition in the library.

`-pvt {min | typ | max}`
: Resets the operating condition for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified operating condition is removed for all three PVT corners.

Example

```
reset_operating_condition
```

Related Information

`get_operating_parameter`

`report_library`

reset_operating_parameter

```
reset_operating_parameter {-process | -voltage | -temperature}
     [-pvt {min | typ | max}]
```

Resets the process, voltage, and temperature (PVT) operating parameters for the design used in calculating accurate cell delays from the nominal cell delays and the k-factors (also called derating factors) from either linear or non-linear models.

Options and Arguments

Note: Only one of the parameters (process, voltage, or temperature) can be reset per command.

`-process`
: Resets the process multiplier.

`-pvt {min | typ | max}`
: Resets the operating parameter for a particular PVT (process, voltage, temperature) corner. You can choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces. *Default*: The specified operating parameter is reset for all three PVT corners.

`-temperature`
: Resets the temperature at which the circuit operates.

`-voltage`
: Resets the voltage at which the circuit operates.

Examples

```
reset_operating_parameter -voltage -typ
reset_operating_parameter -process
reset_operating_parameter -temperature -typ
```

Related Information

set_operating_parameter

reset_operating_voltage

`reset_operating_voltage [-pvt {min | typ | max}] list_of_leaf_level_instances`

Resets the operating voltages asserted on the list of leaf level instances specified. The value is reset to the operating voltage defined in the library.

Options and Arguments

`list_of_leaf_level_instances`
Resets the operating voltage on the specified list of leaf level instances.

`-pvt {min | typ | max}`
Resets the voltage for a particular PVT corner.
Default: The command applies to all three PVT corners.

Example

The following commands reset the `max` operating voltage for `ld1` to the value in the library (5.61):

```
reset_operating_voltage -pvt max ld1
get_operating_voltage -pvt max ld1
5.610000
```

Related Information

`get_operating_voltage`

`read_irdrop`

`read_rrf`

`set_operating_voltage`

reset_path_exception

```
reset_path_exception [-exact] [-type {false_path | cycle_addition |
     path_delay_constraint}] [{-from | -from_rise | -from_fall} from_list]
     [{-through | -through_rise | -through_fall} through_list]
     [{-to | -to_rise | -to_fall} to_list] [-clock_from list_of_clocks]
     [-clock_to list_of_clocks] [-edge_from {leading | trailing}]
     [-edge_to {leading | trailing}] [-early | -late]
     [-bidi_input_from | -bidi_output_from] [-bidi_input_to | -bidi_output_to]
     [-bidi_input_through | -bidi_output_through] [-all]
```

Removes any previously set path exceptions (or path exception of the specified type) for the given paths. To remove all path exceptions, use the `-all` option with the `reset_path_exception` command.

Options and Arguments

`-all`
: Removes all path exceptions. This option cannot be used with the `-from`, `-through`, `-to`, `-clock_from`, `-clock_to`, and `-name` options.

`-bidi_input_from | -bidi_output_from`
: Specifies the assertion on the input or output part of the `from` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_through | -bidi_output_through`
: Specifies the assertion on the input or output part of the `through` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_to | -bidi_output_to`
: Specifies the assertion on the input or output part of the `to` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-clock_from` *`list_of_clocks`*
: Specifies the clocks at the start of the path exceptions to be reset.

`-clock_to` *`list_of_clocks`*
: Specifies the clocks at the end of the path exceptions to be reset.

`-early | -late`

Resets only the early (or late) paths.
Default: Both `-early` and `-late` are reset.

`-edge_from {leading | trailing}`

Specifies an edge for the `-clock_from`.
Default: Both edges.

`-edge_to {leading | trailing}`

Specifies an edge for the `-clock_to`.
Default: Both edges.

`-exact`

The path specifications must match exactly. By default, the given paths to the `reset_path_exception` command must be a subset or equal to the original paths given to any path exception command. The `-exact` option disallows subsets.

Note: Only the path specifiers and the pin list must match exactly. The `-early` and `-late` options do not need to match in order to reset the exception. However, if you use the `-early` and `-late` options, only the one specified is reset. For example:

```
#default is both -early and -late
set_false_path -to Z
#removes only the early exception
reset_path_exception -exact -early -to Z
```

This leaves a false path on the late portion, something that would look like:

```
set_false_path -late -to Z
```

To reset both `-early` and `-late`, use this:

```
#removes both early and late exceptions
reset_path_exception -exact -to Z
```

`{-from | -from_rise | -from_fall}` *`from_list`*

Specifies pins at the start of the path exceptions that are to be reset.

`{-through | -through_rise | -through_fall}` *`through_list`*

Specifies the pins that the path to be reset goes through.

`{-to | -to_rise | -to_fall}` *`to_list`*
Specifies pins at the end of the path exceptions that are to be reset.

`-type {false_path | cycle_addition | path_delay_constraint}`
Removes only path exceptions of the specified type.
Default: All path exception assertions are reset.

Examples

- 1.0 Common Example Setup

  The examples 1.1 through 1.11 all start with the following exceptions. The examples are not successive, that is, the `reset_path_exception` command in 1.2 does not follow the `reset_path_exception` command in 1.1.

```
set_false_path -from { A B C } -to { X Y Z }
set_cycle_addition -from { A B C } 1
set_cycle_addition -to { X Y Z } 1
report_path_exceptions
```

```
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
| A B C | X Y Z | false | false |
| A B C |       |       | add 1 |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.1 Complete from/through/to Match

```
reset_path_exception -from { A B C } -to { X Y Z }
report_path_exceptions
```

```
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
| A B C |       |       | add 1 |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.2 Partial from/through/to Match

```
reset_path_exception -from { A B C }
report_path_exceptions
```

```
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.3 Partial from/through/to Match

```
reset_path_exception -to { X Y Z }
```

```
report_path_exceptions
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
| A B C |       |       | add 1 |
+-------+-------+-------+-------+
```

- 1.4 Exact Match

```
reset_path_exception -exact -from { A B C }
report_path_exceptions
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
| A B C | X Y Z | false | false |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.5 Exact Match

```
reset_path_exception -exact -to { X Y Z }
report_path_exceptions
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
| A B C | X Y Z | false | false |
| A B C |       |       | add 1 |
+-------+-------+-------+-------+
```

- 1.6 No Match

```
reset_path_exception -from { P D Q } -to { X Y Z }
report_path_exceptions
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
| A B C | X Y Z | false | false |
| A B C |       |       | add 1 |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.7 Partial List Match

```
reset_path_exception -from { A } -to { X Y Z }
report_path_exceptions
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
| B C   | X Y Z | false | false |
| A B C |       |       | add 1 |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.8 Partial List Match

```
reset_path_exception -from { A }
report_path_exceptions
```

```
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
|  B C  | X Y Z | false | false |
|  B C  |       |       | add 1 |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.9 Partial List Match

```
reset_path_exception -from { A } -to { Y Z }
report_path_exceptions
```

```
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
|   A   |   X   | false | false |
|  B C  | X Y Z | false | false |
| A B C |       |       | add 1 |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.10 Partial List Match

```
reset_path_exception -from { A } -to { Z }
report_path_exceptions
```

```
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
|   A   |  X Y  | false | false |
|  B C  | X Y Z | false | false |
| A B C |       |       | add 1 |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

- 1.11 Partial List Match w/ Early/Late

```
reset_path_exception -late -from { A } -to { Z }
report_path_exceptions
```

```
+-------+-------+-------+-------+
| From  |   To  | Early |  Late |
|-------+-------+-------+-------|
| A B C | X Y Z | false |       |
|   A   |  X Y  |       | false |
|  B C  | X Y Z |       | false |
| A B C |       |       | add 1 |
|       | X Y Z |       | add 1 |
+-------+-------+-------+-------+
```

Related Information

report_path_exceptions

set_cycle_addition

set_false_path

`set_path_delay_constraint`

reset_path_group

```
reset_path_group [-name group_name]
     [-from from_list] [-through through_list] [-to to_list]
     [-clock_from clk_from_list] [-clock_to clk_to_list] [-exact]
```

Removes the specified path from the named group or completely removes the group.

- To remove the entire group, use:

  ```
  reset_path_group [-name group_name]
  ```

- To remove specific paths, use the same options that were used to add the paths to the group.

Options and Arguments

`-clock_from` *`clk_from_list`*

Removes paths in the group whose start points are related to the listed clocks as specified in `set_path_group` command. This removes paths from flip-flops in the transitive fanout of the clock source pin or port, and paths from ports that have input delay related to the clock.

`-clock_to` *`clk_to_list`*

Removes paths in the group whose endpoints are related to the listed clocks as specified in `set_path_group` command. This removes paths to flip-flops in the transitive fanout of the clock source pin or port, and paths to ports that have external delay related to the clock.

`-exact`

Specifies that the path specifications must match exactly. By default, the paths specified by the `reset_path_group` command must be a subset or equal to the original paths given to any path exception command. The `-exact` option disallows subsets.

`-from` *`from_list`*

Specifies the path start points. The *`from_list`* is a list of port, pin, or leaf names as specified in `set_path_group`. Paths starting from *`from_list`* are removed from the group.

`-name` *`group_name`*

Removes the named group completely when no path options are specified.

When the path options are a subset of those given in the `set_path_group` command, the specified paths are removed from the group. When the path options exactly match those given in `set_path_group` command, the specified group is removed completely. Partially specified paths or path segments are not removed.

`-through` *`through_list`*

Specifies the path through points. The *`through_list`* is a list of port or pin names, or leaf cell names as specified in `set_path_group` command, Only those paths that pass through one of the points in the *`through_list`* are removed.

If the `-through` option is used with the `-from` option as specified in `set_path_group` command, only paths that start in the *`from_list`* and pass through the *`through_list`* are removed.

If the `-through` option is used with the `-to` option as specified in `set_path_group` command, the group includes only paths that pass through the *`through_list`* and end at the *`to_list`* are removed.

If the `-through` option is used with both the `-from` and `-to` options as specified in the `set_path_group` command, only paths that start in the *`from_list`* and pass through the *`through_list`* and end at the *`to_list`* are removed.

If the `-through` option is specified multiple times as specified in `set_path_group`, only those paths that pass through one or more of the points in each *`through_list`* in the order specified are removed. See Example on page 1065.

`-to` *`to_list`*

Specifies the path endpoints.The *`to_list`* is a list of port or pin names as specified in `set_path_group`. Paths ending at the *`to_list`* are removed from the group.

Example

```
reset_path_group -name GRPA
```

Related Information

report_path_groups

set_path_group

reset_port_capacitance

```
reset_port_capacitance port_list [-pvt {min | typ | max}]
```

Removes the port capacitance assertion previously set by the `set_port_capacitance` command.

Options and Arguments

`port_list`
: Specifies the list of ports for which port capacitance is reset.

`-pvt {min | typ | max}`
: Resets the port capacitance for a particular PVT (process, voltage, temperature) corner. You can choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces (`{}`) and separate the values by spaces. *Default*: The port capacitance reset applies to all three PVT corners.

Example

```
reset_port_capacitance [find -port -output dbus*]
```

Related Information

`remove assertions`

`set port capacitance`

reset_port_capacitance_limit

```
reset_port_capacitance_limit [-min | -max] port_list
```

Resets the maximum capacitance allowed to be driven by the port.

This design rule constraint can be reset on top level input and output ports. The limit, by default, is the maximum (or minimum) value for the total capacitances (wire capacitance and pin capacitance) of nets attached to the ports in the port list. Reset a minimum value limit using the `-min` option.

If `set_global max_capacitance_limit` (or `set_global min_capacitance_limit`) has also been set, the most constraining value is used.

Options and Arguments

`-min | -max`
: Resets minimum (or maximum) design rule constraints. If neither option is given.
Default: `-max` (for backward compatibility)

`port_list`
: Specifies the list of ports for which port capacitance limit is reset.

Example

```
reset_port_capacitance_limit [find -port dbus*]
```

Related Information

`remove_assertions`

`set_capacitance_limit`

`set_port_capacitance_limit`

reset_port_wire_load

```
reset_port_wire_load [-pvt {min | typ | max}] port_list
```

Resets the wire load for an input or output top level port of a design to the default wire-load model. The net connected to the port is associated with the default wire-load model, which is used for wire capacitance and resistance estimation.

Options and Arguments

port_list
: Specifies the list of ports.

-pvt {min | typ | max}
: Resets the port wire-load model for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified port wire-load model applies to all three PVT corners.

Example

```
reset_port_wire_load out
```

Related Information

remove_assertions

set_port_wire_load

reset_propagated_clock

```
reset_propagated_clock [-clock clock_list] [-pin pin_list]
```

Resets the clock propagation mode for the given clock waveforms or pins. There are two modes of clock propagation: `ideal` and `propagated`. See the `set_propagated_clock` command for more information.

Note: If the `set_clock_propagation` command is set to `propagated`, the `reset_propagated_clock` command has no effect until the `set_clock_propagation` command is set to `ideal`.

Options and Arguments

`-clock` *`clock_list`*

Specifies one or more clock waveforms for which you want to reset the clock propagation mode.

`-pin` *`pin_list`*

Specifies one or more pins or ports for which you want to reset the clock propagation mode. No hierarchical pins are allowed in the `pin_list`. When a *`pin_list`* is specified, it affects the propagation mode for all the registers in the transitive fanout (TFO) of the pin or port.

Either the `clock_list` option or the *`pin_list`* option must be specified.You can also specify both the *`clock_list`* and the *`pin_list`* options.

Examples

- The following command sets clock waveform `CLK1` to propagated mode:

  ```
  set_propagated_clock -clock CLK1
  ```

- The following command resets clock waveform `CLK1` back to ideal mode:

  ```
  reset_propagated_clock -clock CLK1
  ```

Related Information

`get_clock_propagation`

`get_propagated_clock`

`set_propagated_clock`

`set_clock_propagation`

reset_scale_delays

```
reset_scale_delays [-clock | -data] [-net_delay | -cell_delay | -cell_check]
     [-incl_sdf] [-pvt {min | typ | max}]
```

Unscales `min`, `typ`, or `max` delays from the library and optionally from SDF assertions. This command removes the scaling as previously specified by the `set_scale_delays` command.

Options and Arguments

`-cell_check`
: Removes scaling on timing checks.

  Note: Specifying the `-cell` option is equivalent to both the `-cell_delay` and `-cell_check` options.

`-cell_delay`
: Removes scaling on cell delays.
  If neither the `-cell_delay` nor the `-net_delay` option is specified, the specified scaling factor is removed for both types of delays.

`-clock`
: Removes scaling on clock networks only.

`-data`
: Removes scaling on data networks only.

  Note: If neither the -`clock` option or the -`data` option is specified, the scaling is removed on both the clock and data networks.

`-incl_sdf`
: Removes scaling on SDF assertions.

`-net_delay`
: Removes scaling on net delays.

`-pvt {min | typ | max}`
: Associates a min, typ, or max PVT (process, voltage, or temperature) corner with the scale delay to be removed. Three PVT corners are allowed, namely, min, typ and max.

Default: The specified delay scaling is removed from all PVT corners.

Examples

```
reset_scale_delays -incl_sdf -pvt min
reset_scale_delays -incl_sdf -pvt max
```

Related Information

get_scale_delays

set_scale_delays

reset_slew_limit

`reset_slew_limit [-min | -max] port_list - Renamed to reset_slew_time_limit`

Note: The `reset_slew_limit` command has been renamed, because it resets the limit on the slew time parameter. See reset_slew_time_limit on page 1077.

reset_slew_thresholds

```
reset_slew_thresholds
```

Resets the lower and upper slew threshold values to the default values.

Default: Slew thresholds are taken from the default target technology library. If the target technology library does not have thresholds defined, the defaults of 20-80 percent are used (some library formats use defaults of 10-90 percent, see Using Different Default Thresholds for Different Library Formats in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*).

Examples

```
reset_slew_thresholds
get_slew_thresholds
20.000000 80.000000
```

Related Information

`remove_assertions`

`set_slew_thresholds`

reset_slew_time

```
reset_slew_time [-clock clkname] [-early | -late] [-rise | -fall]
     [-lead | -trail] |-pos | -neg] port_list
```

Removes the slew time assertion previously set by the `set_slew_time` command.

Options and Arguments

`-clock` *`clk_name`*

Specifies the name of the ideal (or generated) clock waveform that is associated with the slew time to be removed. If this option is not specified, the default is @, a signal without an associated clock (asynchronous).

`-early | -late`

Indicates whether the slew times to be removed are with respect to the early (hold) or the late (setup) time checks of the data signal. If neither time is specified the default is to use both.

`-lead | -trail`

Specifies whether the slew time to be removed was asserted with respect to the leading edge or trailing edge of the clock signal.
Default: `-lead`

`port_list`

Specifies the list of pins or leaf cell instance pins for which the slew time assertion is to be removed.

`-pos | -neg`

Specifies whether the slew to be removed applied to an actual clock having a positive or negative polarity with respect to the clock. Only one option either `-pos` or `-neg` can be specified per command.
Default: `-pos`.

`-rise | -fall`

Specifies whether the slew time to be removed applied to the rising edge or the falling edge of the signal (data or clock). If neither time is specified, the default is to use both.

Examples

```
reset_slew_time input3
reset_slew_time -clock B0 -pos clk_b
```

Related Information

set_drive_cell

set_drive_resistance

set_global slew_propagation_mode

set_global slew_time_limit

set_slew_time

set_slew_time_limit

reset_slew_time_limit

`reset_slew_time_limit [-min | -max] port_list`

Resets the limit (maximum or minimum) for slew time at the input and output ports of a module to the default slew time limit placed on specific ports by the `set_global max_slew_time_limit` or `set_global min_slew_time_limit` globals.

Options and Arguments

`-min | -max`
: Specifies minimum (or maximum) design rule constraints. If neither option is given, the default is `-max` for backward compatibility.

`port_list`
: Specifies the list of ports to which the limit to be removed applies.

Example

The following command resets the maximum slew time limits for `bus1` and `bus2`:

```
reset_slew_time_limit {bus1 bus2}
```

Related Information

`set_capacitance_limit`

`set_global max_slew_time_limit`

`set_global min_slew_time_limit`

`set_global slew_propagation_mode`

`set_slew_time`

`set_slew_time_limit`

reset_tech_info

```
reset_tech_info
     ({[-library list_of_library_names]
     [-default_wire_load] [-default_wire_load_selection]
     [-default_operating_conditions]
     [-default_fanout_load] [-default_max_capacitance]
     [-default_max_fanout] [-default_max_transition]
     [-default_min_capacitance] [-default_min_fanout]
     [-default_min_transition][-input_threshold_pct_rise]
     [-input_threshold_pct_fall]
     [-output_threshold_pct_rise] [-output_threshold_pct_fall]
     [-slew_lower_threshold_pct_rise] [-slew_lower_threshold_pct_fall]
     [-slew_upper_threshold_pct_rise] [-slew_upper_threshold_pct_fall]
     [-slew_lower_meas_threshold_pct_rise] |
     [-slew_lower_meas_threshold_pct_fall] |
     [-slew_upper_meas_threshold_pct_rise] |
     [-slew_upper_meas_threshold_pct_fall]}
     [-pvt {min | typ | max}])
     |
     ([-library list_of_library_names] [-cell list_of_cell_names]
     [-dont_modify] [-dont_utilize][-scaling_factors]
     [-input_threshold_pct_rise] [-input_threshold_pct_fall]
     [-output_threshold_pct_rise] [-output_threshold_pct_fall]
     [-slew_lower_threshold_pct_rise] [-slew_lower_threshold_pct_fall]
     [-slew_upper_threshold_pct_rise] [-slew_upper_threshold_pct_fall]
     [-slew_lower_meas_threshold_pct_rise]
     |[-slew_lower_meas_threshold_pct_fall]
     |[-slew_upper_meas_threshold_pct_rise]
     |[-slew_upper_meas_threshold_pct_fall]}
     [-pvt {min | typ | max}])
     |
     ([-library list_of_library_names] [-cell list_of_cell_names]
     [-pin list_of_pin_names] [-fanout_load] [-max_fanout] [-min_fanout]
     [-max_transition] [-min_transition][-max_capacitance] [-min_capacitance]
     [-pvt {min | typ | max}])}
```

Resets the original value of the library data for the specified parameters in the named target libraries. This overrides values previously specified with the `set_tech_info` command with the values from the library itself.

Options and Arguments (any level)

`-library` *`list_of_library_names`*

Specifies the libraries to which the reset is applied.
Default: All libraries specified with the `set_global target_technology` global.

`-pvt {min | typ | max}`
Resets the value for the environment corner of interest. Sets the value for the environment corner of interest. Set one PVT corner per command.
Default: `typ`

Options and Arguments (library level)

`-default_fanout_load`
Resets the fanout load for all pins on all cells in the library to the default value.

`-default_max_capacitance`
Resets the maximum capacitance for all pins on all cells in the library to the default value.

`-default_max_fanout`
Resets the maximum fanout for all pins on all cells in the library to the default value.

`-default_max_transition`
Resets the maximum transition time for all pins on all cells in the library to the default value.

`-default_min_capacitance`
Resets the minimum capacitance for all pins on all cells in the library to the default value.

`-default_min_fanout`
Resets the minimum fanout for all pins on all cells in the library to the default value.

`-default_min_transition`
Resets the minimum transition time for all pins on all cells in the library to the default value.

`-default_operating_conditions`
Resets the operating conditions for the library to the default values.

`-default_wire_load`
Resets the wire-load model for the library to the default.

`-default_wire_load_selection`
Resets the wire-load selection table for the library to the default.

`-input_threshold_pct_fall`
Resets the input threshold percent for the falling edge for all cells in the library to the default value.

`-input_threshold_pct_rise`
Resets the input threshold percent for the rising edge for all cells in the library to the default value.

`-output_threshold_pct_fall`
Resets the output threshold percent for the falling edge for all cells in the library to the default value.

`-output_threshold_pct_rise`
Resets the output threshold percent for the rising edge for all cells in the library to the default value.

Note: The following slew threshold options refer to the section of the waveform where the slew is nearly linear. This is measured and extended like a linear waveform. The linear waveform, a result of extending the near-linear slew waveform, is measured at certain points. The points where this extended waveform is measured are called slew thresholds. Measured slew thresholds are determined at the time of library characterization.

Slew Thresholds

`-slew_lower_threshold_pct_fall`
Resets the lower threshold percent for the slew time of the falling transition for all cells in the library to the default value.

`-slew_lower_threshold_pct_rise`
Resets the lower threshold percent for the slew time of the rising transition for all cells in the library to the default value.

`-slew_upper_threshold_pct_fall`
Resets the upper threshold percent for the slew time of the falling transition for all cells in the library to the default value.

`-slew_upper_threshold_pct_rise`
Resets the upper threshold percent for the slew time of the rising transition for all cells in the library to the default value.

Measured Slew Thresholds

`-slew_lower_meas_threshold_pct_fall`
Returns the measured lower threshold percent of the slew time for the falling transition for all cells in the library to the default value.

`-slew_lower_meas_threshold_pct_rise`
Resets the measured lower threshold percent of the slew time for the rising transition for all cells in the library to the default value.

`-slew_upper_meas_threshold_pct_fall`
Returns the measured upper threshold percent of the slew time for the falling transition for all cells in the library to the default value.

`-slew_upper_meas_threshold_pct_rise`
Returns the measured upper threshold percent of the slew time for the rising transition for all cells in the library to the default value.

Options and Arguments (cell level)

`-cell` *`list_of_cell_names`*
Specifies the cells to which the reset is applied. Required argument for cell and pin level assertions.

`-dont_modify`
Resets to the original value in the library. If `true`, the cell is not modified in optimization or time budgeting.

`-dont_utilize`
Resets `true` or `false`. If `true`, the cell is not used in optimization.

`-input_threshold_pct_fall`
Resets the input threshold percent for the falling edge for the listed cells to the default value.

`-input_threshold_pct_rise`
Resets the input threshold percent for the rising edge for the listed cells to the default value.

`-output_threshold_pct_fall`
Resets the output threshold percent for the falling edge for the listed cells to the default value.

`-output_threshold_pct_rise`
Resets the output threshold percent for the rising edge for the listed cells to the default value.

`-scaling_factors`
Resets the name of the derating table used for scaling.

Note: The following slew threshold options refer to the section of the waveform where the slew is nearly linear. This is measured and extended like a linear waveform. The linear waveform, a result of extending the near-linear slew waveform, is measured at certain points. The points where this extended waveform is measured are called slew thresholds. Measured slew thresholds are determined at the time of library characterization.

Slew Thresholds

`-slew_lower_threshold_pct_fall`
Resets the lower threshold percent for the slew time of the falling transition for the listed cells to the default value.

`-slew_lower_threshold_pct_rise`
Resets the lower threshold percent for the slew time of the rising transition for the listed cells to the default value.

`-slew_upper_threshold_pct_fall`
Resets the upper threshold percent for the slew time of the falling transition for the listed cells to the default value.

`-slew_upper_threshold_pct_rise`
Resets the upper threshold percent for the slew time of the rising transition for the listed cells to the default value to the default value.

Measured Slew Thresholds

`-slew_lower_meas_threshold_pct_fall`
Returns the measured lower threshold percent of the slew time for the falling transition for the listed cells to the default value.

`-slew_lower_meas_threshold_pct_rise`
Resets the measured lower threshold percent of the slew time for the rising transition for the listed cells to the default value.

`-slew_upper_meas_threshold_pct_fall`
Returns the measured upper threshold percent of the slew time for the falling transition the listed cells to the default value.

`-slew_upper_meas_threshold_pct_rise`
Returns the measured upper threshold percent of the slew time for the rising transition the listed cells to the default value.

Options and Arguments (pin level)

`-fanout_load`
Resets the value of the fanout load constraint for the named pins.

`-max_capacitance`
Resets the value of the maximum capacitance constraint for the named pins.

`-max_fanout`
Resets the value of the maximum fanout constraint for the named pins.

`-max_transition`
Resets the value of the maximum transition time constraint for the named pins.

`-min_capacitance`
Resets the value of the minimum capacitance constraint for the named pins.

`-min_fanout`
Resets the value of the minimum fanout constraint for the named pins.

`-min_transition`
Resets the value of the minimum transition time constraint for the named pins.

`-pin` *`list_of_pin_names`*
: Specifies the pins to which the reset is applied. Required argument for pin level assertions.

Examples

These examples follow those of the `get_tech_info` command:

```
reset_tech_info -library lib1 lib2 -default_fanout_load
reset_tech_info -cell FD2ESSA -dont_utilize
reset_tech_info -library lca300kv -cell B2I -pin Z1 -pvt min -max_fanout
```

Related Information

`get_tech_info`

`set_tech_info`

`write_library_assertions`

reset_time_borrow_limit

```
reset_time_borrow_limit [-pin pin_list] [-clock list_of_clocks]
```

Resets the maximum borrow time as set by the `set_time_borrow_limit` command to the default value (Max = pulse width - setup). Set the borrow limit on pins, clocks, or waveforms.

Options and Arguments

`-clock` *`list_of_clocks`*

Resets the borrow limit for the named clocks.

If neither `-pin` nor `-clock` option is specified, the borrow limit is placed on all clocks.

`-pin` *`pin_or_instance_list`*

Resets the borrow limit on a single pin or multiple pins of latches. To specify multiple pins, enclose the list with curly braces (`{}`) and separate the pin information with white space.

You can alternatively give a list of latch instances. In which case, the default limit applies to the clock pins of the instances listed.

Example

```
reset_time_borrow_limit -pin l1/en
```

Related Information

`get_time_borrow_limit`

`set_time_borrow_limit`

Analyzing Latch-Based Designs in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

reset_time_unit

```
reset_time_unit
```

Resets the time units for the session to the default value. The default time unit is the one given in the first library in the target technology library list.

The data from all libraries are descaled appropriately to conform to the default units.

Example

```
reset_time_unit
```

Related Information

`get_time_unit`

`set_time_unit`

reset_wire_capacitance

```
reset_wire_capacitance net_names [-pvt {min | typ | max}]
```

Removes any wire capacitance assertions previously set by the `set_wire_capacitance` command.

Options and Arguments

net_names

Specifies the list of nets for which the wire capacitance assertions are to be removed.

`-pvt {min | typ | max}`

Removes the wire capacitance assertions for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces. *Default*: Wire capacitance assertions are removed for all three PVT corners.

Related Information

`reset_wire_resistance`

`set_wire_capacitance`

reset_wire_load

```
reset_wire_load [-pvt {min | typ | max}] list_of_instances
```

Resets the wire-load model to the default wire-load model on the specified instances. The default model is the default specified in the library, unless overridden by the `set_tech_info -default_wire_load` command. If the list of instances is omitted, the wire-load model is reset for the current instance.

Options and Arguments

`list_of_instances`
: Specifies the instances to which the default wire-load model is re-applied.
Default: The current instance.

`-pvt {min | typ | max}`
: Resets the wire-load model for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified wire-load model is reset for all three PVT corners.

Example

The following command resets the wire-load model for pvt `min` to the default wire-load model on instance `J_block:`

```
reset_wire_load -pvt min J_block
```

Related Information

`get_tech_info -default_wire_load`

`set_tech_info -default_wire_load`

`set_wire_load`

`write_assertions`

reset_wire_load_mode

```
reset_wire_load_mode
```

Resets the wire-load mode to the default mode of `top`.

Related Information

`set_wire_load_mode`

reset_wire_load_selection_table

```
reset_wire_load_selection_table [-library library_name] [-pvt {min | typ | max}]
```

Resets the wire-load selection table entry previously specified by the `set_wire_load_selection_table` command. The default wire load selection table specified in the library *`library_name`* becomes the selection table used for wire load calculations.

Options and Arguments

`-library` *`library_name`*
: Specifies the name of the technology library to search for the wire load selection table.
Default: The target technology library.

`-pvt {min | typ | max}`
: Resets the wire-load model selection table for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces (`{}`) and separate the values by spaces.
Default: The wire-load model selection table is reset for all three PVT corners.

Example

The following command resets the wire-load selection table to the one specified in the `wireloads2` library for the `min` PVT corner:

```
reset_wire_load_selection_table -library wireloads2 -pvt min
```

Related Information

`report_library`

`set_global target_technology`

`set_wire_load_selection_table`

`set_wire_load`

reset_wire_resistance

```
reset_wire_resistance [-pvt {min | typ | max}] net_names
```

Removes wire resistance assertions previously specified by the `set_wire_resistance` command.

Options and Arguments

net_names

Specifies the list of nets for which the wire resistance assertions are to be removed.

-pvt {min | typ | max}

Removes wire resistance assertions for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces. *Default*: Removes wire resistance assertions for all three PVT corners.

Related Information

`reset_wire_capacitance`

`set_wire_resistance`

save_mp_model

```
save_mp_model
     [-min min_file_name] [-typ typ_file_name] [-max max_file_name] file_name
```

Saves the prototype either to one or more files or to memory. It also ends an Module Prototyper (MP) session. If no parameters are given, the model is saved to memory where it can be used for binding. If there are any parameters, the model will be written to one or more files. One file will be written for each PVT corner present. If the ending of the file name ends in `.lib,` a Liberty file will be written. Otherwise, a TLF file will be written.

Options and Arguments

`file_name`
: Creates a file name if a file name is not specified using the `-min`, `-typ`, or `-max` options.The `typ` corner will use the name as is. The `min` and `max` corners will have a prepended `min_` or `max_`.

`-max` *`max_file_name`*
: Specifies the file name used for the maximum PVT corner.

`-min` *`min_file_name`*
: Specifies the file name used for the minimum PVT corner.

`-typ` *`typ_file_name`*
: Specifies the file name used for the typical PVT corner.

Examples

- The following command saves the Module Prototype to memory, which can be used by `do_bind_generic` command or the `do_rebind` command, but is not saved to a file:

  ```
  save_mp_model
  ```

- The following command saves the Module Prototype to one or more tlf files. If present, the typical data will be saved to `my_block.tlf,` the minimum data will be saved to `“”`, and the maximum data will be saved to `“”`. After using this command, the MP data is not present in memory and cannot be used for binding until it is read using the `read_tlf` command:

  ```
  save_mp_model my_block.tlf
  ```

- The following command is similar to the proceeding command except the data is saved to a `.lib` Liberty format file:

  ```
  save_mp_model my_block_max.lib
  ```

- The following command saves the Module Prototype to Liberty files: the -max data is saved in my_block_max.lib and the -min data is saved in my_block_min.lib:

```
save_mp_model -min my_block_min.lib -max my_block_max.lib
```

Related Information

create_mp_model

set_annotated_check

```
set_annotated_check  [-clock clock_check] [-cond sdf_condtion] check_value
     [-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low]
     [-rise] [-fall] [-min] [-max] [-from from_pins][-to to_pins]
```

Annotates the setup, hold, recovery, or removal timing check value between two or more pins on a cell in the current design.

Options and Arguments

check_value

Specifies the timing check value.

-clock *clock_check*

Specifies clock rising or falling.
Default: Sets checks for both clock rise and fall.
Use the *clock_check* argument to specify rise or fall.

-cond *sdf_expression*

Specifies the condition only if the library has a condition attached to the specified timing check arc. The *sdf_expression* must match the syntax of the condition specified in the library.

-fall

Specifies the data fall transition for the timing check.
Default: Sets both -rise and -fall.

-from *from_pins*

Specifies startpoints of the timing arcs for which checks are annotated. Use the *from_pins* argument to specify a list of leaf cell clock pins.

-max

Specifies the maximum timing check for both data rise and data fall transitions. Use this option only if the design uses minimum and maximum (min_max) operating conditions.

-min

Specifies the minimum timing check for both data rise and data

fall transitions. Use this option only if the design uses minimum and maximum (`min-max`) operating conditions.

`-rise`

Specifies the data rise transition for the timing check.
Default: Sets both rise and fall.

`-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low`

Specifies the timing check type. Make sure there is a corresponding timing arc between the *`from_pins`* and the *`to_pins`* arguments.

`-to` *`to_pins`*

Specifies endpoints of the timing arcs for which checks are annotated. Use the *`to_pins`* argument to specify a list of leaf cell clock pins.

Examples

- The following commands annotates a `nochange check` between data low and clock low. The nochange margin before the clock is `1.3` and the nochange margin after the clock is `2.4`:

```
set_annotated_check -nochange_low 1.3 -clock fall -fall -from {I1/CP} -to {I1/EN}
set_annotated_check -nochange_low 2.4 -clock fall -rise -from {I1/CP} -to {I1/EN}
```

- The following command annotates a `2.0` setup time between the `CP` clock pin and the `A` data pin of the `instancename` instance:

```
set_annotated_check -setup 2.0 from instancename/CP -to instancename/A
```

Related Commands

set_annotated_delay

See “Timing Checks” in the *Cadence Common Timing Engine (CTE) User Guide* for more information.

set_annotated_delay

```
set_annotated_delay [-net | -cell] [-rise | -fall] [-pvt {min |typ | max}]
     [-from from_pins] [-to to_pins] [-sdf_cond expression] delay_value
```

Annotates delay to timing arcs.

Options and Arguments

delay_value
: Specifies the delay value to be annotated to the selected timing arcs.

[-from *from_pins*]
: Specifies timing arcs to which the delay applies. The selected timing arcs are those that have their source pins specified using this option and the sink pins specified using the -to option. If this option is not specified, only timing arcs whose sink pins are specified using the -to option are selected.

[-net | -cell]
: Specifies whether the delay should be annotated to cell arcs or net arcs. The tool automatically detects if the arcs specified with the -from and -to options are cell arcs or net arcs. This option is ignored if it conflicts with the -from and -to options. Use this option if the pin specified by the -from option or the -to option is a bidirectional pin. See Figure 7-6.

[-pvt {min | typ | max}]
: Specifies the PVT corner to which the delay applies.
Default: The delay applies to all three PVT corners.

[-rise | -fall]
: Specifies the transition to which the delay applies. For cell arc, this is the output transition.
Default: The delay applies to both transitions.

[-sdf_cond expression]
: Specifies the sdf condition of the selected timing arcs.
Default: The delay applies to all the timing arcs selected using the -from and the -to options. The expression is ignored for net arcs.

Note: Using the `set_annotated_delay` command without `sdf_cond` option will overwrite all previously specified annotated delays with the `sdf_cond` option. So, annotate delays without the condition first before annotating delays with the condition.

`[-to` *`to_pins`*`]`

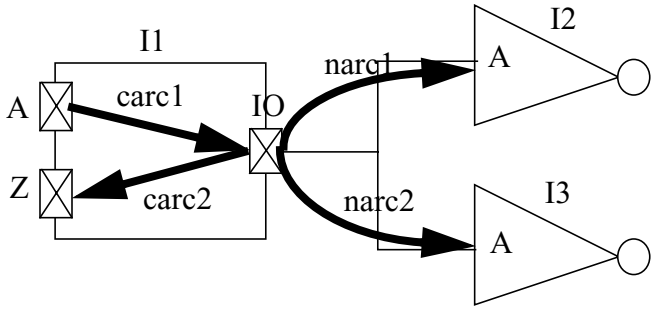
Specifies timing arcs to which the delay applies. The selected timing arcs are those that have their sink pins specified using this option and the source pins specified using the `-from` option. If this option is not specified, only timing arcs whose source pins are specified using the `-from` option are selected.

Examples

- The following command affects three timing arcs: the cell arc `carc2` and the net arcs `narc1` and `narc2` as shown in Figure 7-6:

  ```
  set_annotated_delay 3.0 -from {I1/IO}
  ```

Figure 7-6 Example Figure to Show How to Use the -cell and -net Options



  - Use the `-cell` option to select the cell arc only:

    ```
    set_annotated_delay 3.0 -cell -from {I1/IO}
    ```

  - Use the `-net` arc option to select net arcs only:

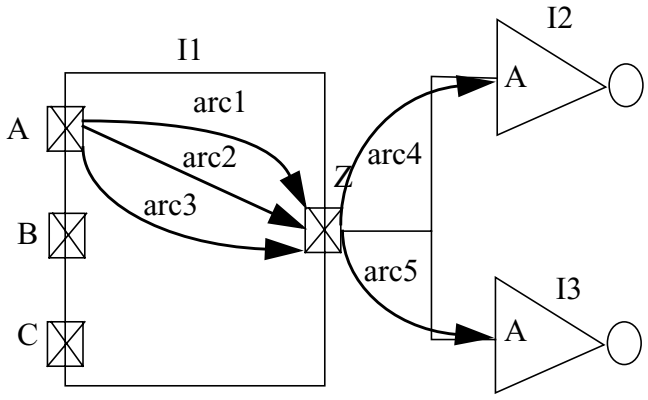
    ```
    set_annotated_delay 3.0 -net -from {I1/IO}
    ```

- This TLF description is used for the following examples:

  ```
  CELL (NAND3
  . . . .
  // arc 1
  Path( A => Z 10 01 DELAY(ioDelayRiseModel0) SLEW(SlopeRiseModel0) )
  Path( A => Z 01 10 DELAY(ioDelayFallModel0) SLEW(SlopeFallModel0) )
  ```

```
// arc2
Path( A => Z 10  01 COND(B & C) SDF_COND(B & C) DELAY(ioDelayRiseModel1)
SLEW(SlopeRiseModel1) )
Path( A => Z 01 10 COND(B & C) SDF_COND(B & C) DELAY(ioDelayFallModel1)
SLEW(SlopeFallModel1) )
arc3
Path( A => Z 10 01 COND(~B | ~C) SDF_COND(!B | !C) DELAY(ioDelayRiseModel2)
SLEW(SlopeRiseModel2) )
Path( A => Z 01 10 COND(~B | ~C) SDF_COND(!B | !C) DELAY(ioDelayFallModel2)
SLEW(SlopeFallModel2) )
. . .
}
```

Figure 7-7 Example Figure to Show How to Use the -from and -to Options



- The following command annotates a delay of `3.0` to arcs `arc1`, `arc2`, and `arc3` for the rise and fall transition at the pin I1/Z, for the three pvt corners `{min typ max}`:

  ```
  set_annotated_delay  3.0 -from A -to Z
  ```

- The following command annotates a delay of 2.5 to net arc `arc4` for both the rising and the falling transition and for the three pvt corners `{min typ max}`:

  ```
  set_annotated_delay 2.5 -from I1/Z -to I2/A
  ```

- The following command annotates a delay of 1.5 to `arc2` for the rising transition and the min pvt corner:

  ```
  set_annotated_delay 1.5 -from I1/A -to I1/Z -rise -pvt min -sdf_cond “B & C”
  ```

- The following commands annotate a delay of 3.0 to `arc1` and `arc2`, and a delay of 2.4 to the `arc3`:

  ```
  set_annotated_delay 3.0 -from I1/A -to I1/Z
  set_annotated_delay 2.4 -from I1/A -to I1/Z -sdf_cond {!B | !C}
  ```

- The following commands first annotate a delay of 3.0 to `arc1`, `arc2` and `arc3`, then overwrite the value set by the first `set_annotated_delay` command:

```
set_annotated_delay 2.4 -from I1/A -to I1/Z -sdf_cond {!B | !C}
set_annotated_delay 3.0 -from I1/A -to I1/Z
```

Related Commands

set_annotated_check

set_capacitance_limit

```
set_capacitance_limit [-min | -max] float [-instance instance_list]
     [-port port_list] [-module list_of_modules]
```

Sets instance-specific or port-specific design rule constraints. Design rules set on a hierarchical instance applies to all the pins contained in that hierarchical instance.

Note: If neither the `-instance` or the `-port` argument is specified, the design rule constraint is applied to the top timing module.

Use the `set_global max_capacitance_limit` or the `set_global min_capacitance_limit` to set context specific design rules. If either one of these globals has also been set, the most constraining value is used.

See Extracting Design Rules on the Timing Model in the *Common Timing Engine (CTE) User Guide* for supported design rules and more information.

Options and Arguments

float
: Specifies the minimum or maximum capacitance value.

-instance *instance_list*
: Specifies the list of hierarchical instances where the limit is applied.

-max
: Sets the maximum instance-specific design rule constraints on all cells.

-min
: Sets the minimum instance-specific capacitance on all cells.

-module *list_of_modules*
: Specifies the list of modules where the limit is applied.

  Note: The `-instance` option takes precedence over the `-module` option. For example, if a module and its instance have a capacitance limit assertion, the instance assertion is used. Instances of a module inherit the capacitance limit assertion of the module unless the instance has its own capacitance limit assertion.

`-port` *`port_list`*

Specifies the list of ports where the limit is applied.

Examples

- Shows equivalent commands. The min capacitance limit `0.05` applies to the top timing module:

```
set_capacitance_limit 0.05 -min
set_capacitance_limit 0.05 -min -instance { }
```

- Uses the last value set if several values are set to the same design rule using the same command or global:

```
set_capacitance_limit 0.05 -max -instance {I1}
set_capacitance_limit 0.07 -max -instance {I1}
set_capacitance_limit 1.1 -max -instnace {I1}
```

  The maximum capacitance limit used is `1.1`.

- Uses the most constraining value (smallest value for the max limit) (largest value for the min limit) if multiple values are set to the same design rule using different commands and globals. For example, if the capacitance limit of pin Z in the library is `0.7`, and the context capacitance limit is set using the global:

```
set_global max_capacitance_limit 1.0
```

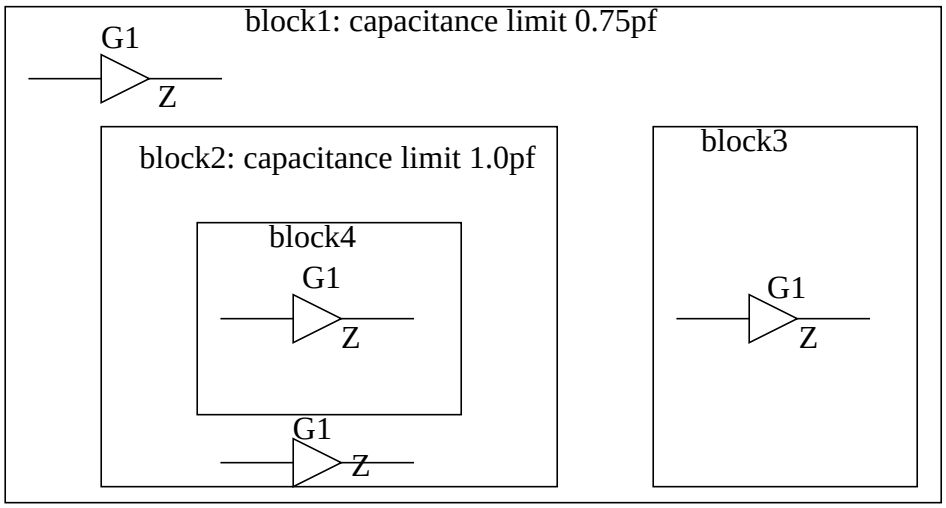
  and the capacitance limit is set on hierarchical instance `I1` containing an instance of the pin Z:

```
set_capacitance_limit 0.8 -max -instance {I1}
```

  The maximum capacitance limit used for the instance of pin Z in `I1` is 0.7 = min {0.7, 1.0, 0.8}.

Note: Design rules asserted on a hierarchical block is inherited by the hierarchical subblock contained in the blocks. Design rules set on a sub-block overwrite design rules set on a block containing the subblock as shown in Figure 7-8.

Figure 7-8 Design Rules Asserted on a Hierarchical Block



The following lists the pin block capacitance limits:

- The capacitance limit of the pin block 1/G1/Z is `0.75pf`.
- The capacitance limit of the pin block1/block2/G1/Z is `1.0pf` because the capacitance limit set on block1/block2 (`1.0pf`) overwrites the capacitance limit set on block1 (`0.75pf`).
- The capacitance limit of the pin block1/block3/Z is `0.75pf`, because block1/block3 inherits the capacitance limit set on block1.
- The capacitance limit of the pin block1/block2/block4/G1/Z is `1.0pf` (capacitance limit inherited from block1/block2).

Related Information

`reset_capacitance_limit`

`reset_port_capacitance_limit`

`reset_slew_time_limit`

`set_global_max_capacitance_limit`

`set_global_min_capacitance_limit`

`set_port_capacitance_limit`

`set_slew_time_limit`

set_capacitance_unit

`set_capacitance_unit` *`float_unit`*

Sets the capacitance unit in picoFarads used for the entire session. Without this command, capacitance units for the session are decided by the first library in the target technology list.

All reports use these session units. All design assertions you make (such as `set_port_capacitance`) are also expected to be in these units. Any assertion you make that modifies library data (`read_library_update` and `set_tech_info`) is expected to provide data in units of the library (not session units).

The data from all libraries are scaled appropriately to conform to the session units. Units are for the entire session and are not removed by the `do_remove_design -all` command.

Options and Arguments

`float_unit`

Specifies the capacitance unit for this session in picoFarads.

Example

The following command sets the capacitance unit to 1 picoFarad:

```
set_capacitance_unit 1
```

Related Information

`get_capacitance_unit`

`reset_capacitance_unit`

`set_time_unit`

set_case_analysis

```
set_case_analysis {0 | 1} {-bidi_input | -bidi_output} {-rising | -falling}
     list_of_ports_or_pins
```

Sets constant or transitional values to a list of pins/ports to either a 1, 0, rising, falling value for use by the timing engine.

Options and Arguments

`0 | 1`

Specifies a logic value of 0 or 1, which is propagated through combinational logic to disable or enable parts of logic.

`-bidi_input | -bidi_output`

Specifies that the constant is set on the input or output part of bidirectional pins. The default places the constant on the input part.

`list_of_ports_or_pins`

Specifies a list of ports or pins on which to apply the value. To specify multiple pins, enclose the list with curly braces (`{}`) and separate the pin names with white space.

`-rising | -falling`

Specifies a valid transition value to be considered by the timing engine. For example, If a rising value is specified at a certain pin or port, only the rising signal transition from the pin or port is considered for timing analysis. The opposite transition type (falling) is ignored.

Examples

- The following command sets the value of pin `i4/A` to 1 for use by the timing engine:

  ```
  set_case_analysis 1 i4/A
  ```

- The following example shows that the pins P1/P2/B and P1/P3/CI are considered only for a rising transition. The falling transition on these pins is ignored.

  ```
  set_case_analysis rising {P1/P2/B P1/P3/CI}
  ```

Related Information

check_timing -type

const_collision

const_contradiction

get_constant_for_timing

reset_constant_for_timing

set_constant_for_timing

set_disable_timing

set_false_path

Setting Constant Values for Timing in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

set_cell_pin_load

```
set_cell_pin_load [-library library_name] [-cell cell_name]
     [-pin pin_name] load_val
```

Changes the pin capacitance value specified in the library. The assertion uses the capacitance unit specified in the first library of the target technology list.

Options and Arguments

`-cell` *`cell_name`*

Specifies the name of the cell whose pin is specified by the `-pin` argument.

`-library` *`library_name`*

Specifies the name of the library where the cell can be found.
Default: The target library is searched for the cell.

Note: If the `target_technology` is set to all loaded libraries, then the `-library` option is not needed. If you are querying the capacitance value of a cell or pin for a loaded library, but it is not part of the target technology list, an ERROR is returned. Use the `-library` option to find the cell or pin that is queried from the library name specified by the `-library` option.

`load_val`

Specifies the float value to be set.

`-pin` *`pin_name`*

Specifies the name of the pin whose capacitive load is returned.

Example

- The following example describes a library cell. The timing arcs have been deleted for simplicity:

```
library(cell_load_example) {
  cell ( AN2 ) {
    area : 2.000 ;
        cell_footprint : "dpand2c" ;
    pin ( Z ) {
        direction : output;
        capacitance : 0.000000;
```

```
            ...
            ...
            function : "(A & B)";
            max_capacitance :  1.14574 ;
        }

        pin ( A ) {
            direction : input;
            capacitance : 0.042000;
            }
        pin ( B ) {
            direction : input;
            capacitance : 0.041000;
        }
      }
    }
```

- The following command changes the capacitance value of pin B for cell AN2to the value 0.063450:

  ```
  set_cell_pin_load -library cell_load_example -cell AN2 -pin B 0.063450
  ```

  Note: An ERROR is returned if you are setting or getting values for a pin that is not defined for a cell.

- The following command gets the capacitance value for pin A:

  ```
  get_cell_pin_load -library cell_load_example -cell AN2 -pin A
  0.042000
  ```

- The following command gets the capacitance value for pin B:

  ```
  get_cell_pin_load -library cell_load_example -cell AN2 -pin B
  0.063450
  ```

Related Information

get_cell_drive

get_cell_pin_load

set_global target_technology

set_clock

`set_clock` *`clock_signame`* {[`-period` *`period`*] | [`-waveform` {*`lead_time trail_time`*}]}

Defines an ideal clock signal. An ideal clock must be defined as a global reference signal for all the data signals in the design. The ideal clock is defined by its name, period, and position of the leading edge and trailing edge.

In a single-clock design, only one `set_clock` command is used. In a multiphase clock system, several `set_clock` commands are used to define each phase of the clock. For purely combinational design, an ideal clock (or any clock) definition is not necessary.

The scope of this command is the entire session of `ac_shell`. Once an ideal clock is defined, it remains accessible to all subsequent commands.

Options and Arguments

`clock_signame`

Specifies the name of the ideal clock signal.

`-period` *`period`*

Specifies the period of the ideal clock signal is set to *`period`*. If the period is not specified, it is derived from the waveform specification such that a symmetric clock (50 percent duty cycle) is generated. Units are specified in the library.

`-waveform {`*`lead_time trail_time`*`}`

Specifies the clock waveform is defined by a leading edge and a trailing edge. The braces are used to create a list. A Tcl list can also be created using a pair of quotes (“”), so braces can be replaced by quotes.

The *`lead_time`* is the time at which the first transition on the clock occurs. The *`trail_time`* is the time that the second transition on the clock occurs.

Note: The *`trail_time`* should be greater than the lead time. For example:

```
set_clock {rfclk_ideal} -waveform
{38.400002 76.800003} -period 76.800003
```

If the `-waveform` option is not specified, the leading edge is placed at 0 and the trailing edge is placed at the midpoint of the period such that a symmetric clock is generated.

Specify either the period or the waveform option.

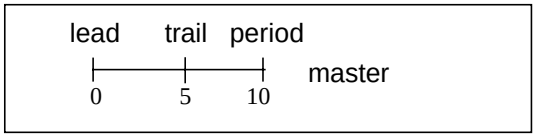
Examples

- The following command defines an ideal clock signal called `master` with a period of `10`, with leading transition at `0` and trailing transition at `5`:

  Note: An ideal clock signal has no polarity.

  ```
  set_clock master -period 10 -waveform {0 5}
  ```

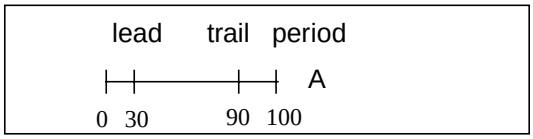
Figure 7-9 Ideal Cock Example 1



- The following command defines an ideal clock called A with a period of `100`, with leading transition at `30` and trailing transition at `90`:

  ```
  set_clock A -period 100 -waveform {30 90}
  ```

Figure 7-10 Ideal Cock Example 2



Related Information

`set_clock_root`

`set_clock_insertion_delay`

`set_clock_uncertainty`

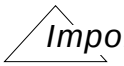
`set_data_required_time`

`set_external_delay`

See Specifying Clock Information in the *Common Timing Engine (CTE) User Guide*.

set_clock_arrival_time

```
set_clock_arrival_time
     OBSOLETE: Use set_clock_root and set_clock_insertion_delay instead.
```



Do not use the `set_clock_arrival_time` command, use the `set_clock_root` and the `set_clock_insertion_delay` commands instead. See set_clock_root on page 1127 and set_clock_insertion_delay on page 1120.

Cadence timing analysis still writes out `set_clock_arrival_time` during time budgeting. The following description is for reference only.

```
set_clock_arrival_time
-clock clock_name [-early | -late] [-rise rise_time] [-fall fall_time]
[-pos | -neg] [-bidi_input | -bidi_output] pin_list
```

The internally generated `set_clock_arrival_time` command associates the ideal clock to physical ports of the design in the database. It defines the actual clock signals arriving at the input port of the design in relation to the ideal clock signal defined by the `set_clock` command.

set_clock_gating_check

```
set_clock_gating_check [-setup | -hold] [-clock clock_list [-lead | -trail]]
     [[-pin instance_or_pin_list] [-rise | -fall]] [-high | -low] float
```

Creates or overrides the default setup and hold values for clock-gating timing checks.

Note: This command has no effect on the `ClockGatingSetup` (or `ClockGatingHold`) check defined as a non-sequential setup (or hold) check in the library.

You cannot use this command to create clock-gating assertions for hierarchical pins or clock ports. If you use this command on pins where clock gating occurs, these assertions are not propagated along the clock tree.

If neither the `-pin` nor the `-clock` option is specified, then the timing margin specified becomes a global assertion and is used for all clock gating checks. However, pin-based timing check assertions have higher priority and, if they exist, they override the global assertion.

The timing analyzer uses the following priority rules to determine the timing margin to use for a particular clock gating check:

1. Use the timing check assertion on the data pin if it exists
2. Otherwise, use the timing check assertion on the clock pin if it exists
3. Otherwise, use the timing check assertion on the clock waveform (for the clock pin) if it exists
4. Otherwise, use the global timing check assertion if it exists
5. If no timing check assertion is present, use the gate delay. This value is as follows:
   - For setup checks: the delay from the data pin to the output
   - For hold checks: the delay from the clock pin to the output

These rules are demonstrated in Examples on page 1114.

Options and Arguments

`-clock clock_list`

Specifies the clock waveforms for which the command creates a clock-gating assertion. The timing margins for clock-gating checks for any instance that gates a clock signal having such an assertion are overridden.

float

The timing margin (setup value or hold value) for the clock-gating setup check.

The default values when this command is not used depends on the analysis mode. In ideal propagation mode, the clock network is assumed to have zero delays, so CTE uses zero setup and zero hold time. In propagated mode, for setup time, the analysis uses the maximum arc delay from the data pin to the output pin. For hold time, it uses the minimum arc delay from the clock pin to the output pin.

`-high | -low`

Specifies that the non-controlling value of the clock is high or low. By default, it determines the non-controlling value of the clock using information from the gate's logic. For complex gates such as XOR and MUX, the clock is never controlling, and it does not perform checks unless you specify either the `-high` option or the `-low` option. The specified value takes precedence over the default values. Use this option only with the `-pin` option.

`-lead | -trail`

Applies the specified timing margin to only those clock-gating checks for which the reference clock edge is the leading (or trailing) edge. If neither option is specified, the timing margins apply to both leading and trailing edges.

Use the `-lead` or `-trail` options only with the `-clock` option.

`-pin` *pin_or_instance_list*

If a pin is specified, the clock-gating instance connected to this pin will use the specified setup and hold values for this pin.

If a clock-gating instance is specified, the assertion applies to all input pins of the instance.

`-rise | -fall`

Applies the specified timing margin to the rising (or falling) signals on the specified pins. If neither option is specified, the default is both rising and falling. Use the `-rise` or `-fall` options only with the `-pin` option.

`-setup | -hold`

Specifies the type of clock-gating check, either setup or hold. If neither option is present, the timing margin is applied to both setup and hold checks.

The setup check is performed with respect to the edge of the clock signal that changes the state of the clock pin from controlling to non-controlling. For example, for AND and NAND gates, the setup check is performed with respect to the rising edge of the clock input. For OR and NOR gates, the setup check is performed with respect to falling edge of the clock input.

The hold check is performed with respect to the edge of the clock signal that changes the state of the clock pin from non-controlling to controlling. For example, for AND and NAND gates, the hold check is performed with respect to the falling edge of the clock input. For OR and NOR gates, the hold check is performed with respect to rising edge of the clock input.

If the clock pin is never controlling, no clock gating checks are performed. For example, there will be no clock gating check performed for XOR and MUX gates.

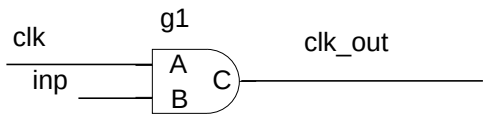
If the gate logic in unknown, setup and hold checks are performed with both the rising edge and the falling edge of the clock signal.

Note: The clock pin is controlling when the gate output is independent of the data input with which the clock gating check is performed.

Examples

- The following example show the priority rules for multiple timing check assertions for the gated clock shown in Figure 7-11.

Figure 7-11 Gated Clock



Tcl script:

```
# ideal clock
set_clock CLK1 -period 12 -waveform "0 6"
set_clock_root -clock CLK1 clk
...
# clock-gating timing check assertions

# on ideal clock waveform
set_clock_gating_check -setup 0.44 -clock CLK1

# on clock pin
set_clock_gating_check -hold 0.11 -pin g1/A

#on data pin
set_clock_gating_check -setup 0.66 -rise -pin g1/B
set_clock_gating_check -hold 0.22 -fall -pin g1/B
```

Given the previous commands, the following timing margins are used for slack calculation:

SETUP g1/B rise: 0.66 — from assertion on data pin g1/B (rule 1.)

SETUP g1/B fall: 0.44 — from assertion on ideal clock CLK1 (rule 3.)

HOLD g1/B rise: 0.11 — from assertion on clock pin g1/A (rule 2.)

HOLD g1/B fall: 0.22 — from assertion on data pin g1/B (rule 1.)

Related Information

`reset_clock_gating_check`

`set_global_clock_gating_to_be_checked`

`set_clock_gating_check`

Setting Clock Gating Setup and Hold Checks in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

set_clock_info_change

```
set_clock_info_change
     To change to a data signal use: set_clock_info_change
     [-lead | -trail] [-early | -late] [-rise | -fall]
     -clock clock_name
     pin_list
     ||
     To change to a clock signal use: set_clock_info_change
     [-pos | -neg] [-early | -late] [-rise | -fall]
     -clock clock_name
     pin_list
```

Changes the clock or data information for paths going through the specified pins for the downstream logic. This command is useful for modeling frequency dividers and clock-shaping circuits. Use this command to change:

- Clock to data
- Data to clock
- Clock to clock (derived clock)
- The associated clock information

Calculations

This section describes how the rising and the falling edges of the final (mapped) clock signal are computed when the output of a gate register is defined as a clock.

Here RA_{old} is the arrival time of the rising edge of the original signal before mapping. Similarly, FA_{old} is the arrival time of the falling signal.

For a signal mapped to a positive clock:

Rising edge = RA_{old}

Falling edge = FA_{old} + (ClockTrailingEdge - ClockLeadingEdge)

ClockLeadingEdge and ClockTrailingEdge are times at which the leading and the trailing edges of the ideal clock signal occur. Here the rising edge of the mapped clock corresponds to the leading edge of the ideal clock waveform and falling edge corresponds to the trailing edge.

Similarly, if the signal is mapped to a negative clock:

Rising edge = RA_{old} + (ClockTrailingEdge - ClockLeadingEdge)

Falling edge = FA_{old}

Here the falling edge of the mapped clock corresponds to the leading edge of the ideal clock waveform and rising edge corresponds to the trailing edge.

Derived Clocks

A clock to clock `set_clock_info_change` command on a pin results in starting a new clock signal from the pin. The new clock signal is derived from the old. Source and network clock insertion delays from the source clock are transferred to the derived clock. Here, the clock coming into the pin is being called the source clock, and the new clock created on the pin is called the derived clock.

Details of how source and network insertion delays are transferred to the derived clock are as follows:

First, the edge correspondence is determined from the source and derived clock polarities. Edge correspondence means which edge of the derived clock is derived from which edge of the source clock. The clock polarity (positive or negative) of the source clock refers to the polarity of the clock signal arriving at the pin. Similarly, clock polarity of the derived clock refers to the polarity of the clock signal being generated from the pin.

- Edge correspondence is determined as follows:

  If source and derived clocks are both of positive polarity, or both of negative polarity, then rising (falling) edge of the derived clock is assumed to be derived from the rising (falling) edge of the source clock. Similarly, if source and derived clocks are of opposite polarity, the rising (falling) edge of the derived clock is assumed to be derived from the falling (rising) edge of the source clock.

Then, the source and network insertion delays from the source clock edges are transferred to the corresponding derived clock edges.

- Source and network insertion delays from the source clock edges are transferred to the corresponding derived clock edges if the following additional constraints are satisfied:
  - Derived clock period must be an integral multiple (n) of the source clock period, with n >= 2.
  - For the corresponding edges of the source and derived clocks (as determined above), following relation must be satisfied:

    Edge_time_of_derived_clock = Corresponding_edge_time_of_source_clock + (n * period_of_source_clock)

    where n is an integer.

Options and Arguments

`-clock` *`clock_name`*

Specifies the clock waveform to apply or remove from the pin.

`-early | -late`

Specifies early or late arrival of the signal.
Default: Both

`-lead | -trail`

Indicates that the signal at the specified pin must be recognized as a data signal with the specified association (`lead` or `trail`) with respect to the specified clock *`clock_name`*.
Default: `-lead`

`pin_list`

Associates paths from the pin list with the specified clock *`clock_name`*. The pin list can be ports or instance pins. These can be object IDs or hierarchical names relative to the current module.

`-pos | -neg`

Indicates that paths must be recognized as clock paths and that the clock has the specified polarity. This option is required to set the path as a clock path. If neither `-pos` or `-neg` is specified, the path is treated as a data path.

Note: Because the signal cannot be both a clock signal and a data signal, the `-pos | -neg` options cannot be used with the `-lead | -trail` options.

`-rise | -fall`

Maps the rising or falling edge of the signal.
Default: Both.

Examples

- The following command changes the output of an AND gate to a clock signal type:

```
set_clock_info_change -clock clk -pos U1/A1/O
```

- The following command changes the output back to data signal type:

```
set_clock_info_change -clock clk U1/A1/O
```

Related Information

set_clock

set_clock_insertion_delay

set_clock_root

report_clocks

reset_clock_info_change

set_clock_insertion_delay

```
set_clock_insertion_delay {{[-source] [-lead | -trail] [-early | -late]
     [-min | -max] insertion_delay list_of_clocks}
     | {[-source] [-clock clock_signame] [-early | -late] [-min | -max]
     [-rise | -fall] insertion_delay -pin list_of_pins}}
```

Important

Use this command in conjunction with the `set_clock_root` command to replace the `set_clock_arrival_time` command.

Specifies clock insertion delay in the ideal and propagated modes. Clock insertion delay has two components:

- Source latency

  Specified by the `-source` option, source latency is the delay from the external clock generator to the clock port of the design. Source insertion delay is considered in both ideal and propagated modes.

- Network latency (clock tree insertion delay)

  The default delay type. Network delay is the delay from the clock port to the register. The network delay specified by the `set_clock_insertion_delay` command is only considered in ideal mode. In propagated mode, this delay is replaced by the actual clock tree delays.

  Network delay on the clock waveform or the clock port is ignored in the propagated mode. Only the source insertion delay is honored. Source insertion delay specified on the clock port has higher precedence than one on clock waveform. In addition, a source and network insertion delay pair specified on internal circuit pins (including hierarchical ports) is ignored in propagated mode. To summarize:

- Source (or network) insertion delay specified on a timing pin has higher precedence than clock waveform specification.
- Source insertion delay is honored in both ideal and propagated modes.
- Network insertion delay is ignored in the propagated mode, and the actual circuit delays are used.
- If both source and network insertion delays are specified on internal circuit pins (or hierarchical ports), they are used in the fanout cone of the pin in Ideal clock propagation mode.

- If the set_flow_compatible_mode command is set to off, the network insertion delay on an internal pin is ignored if specified without a source insertion delay on the same pin.
- If the set_flow_compatible_mode command is set to on, the source insertion delay on an internal pin is ignored, which is similar to Synopsys PrimeTime (PT does not allow setting a source insertion delay on an internal pin). If the network insertion delay is specified, it is applied to the pins in the fanout cone in ideal mode.
- Any source and/or network insertion delay specified on clock waveforms is used with both the set_input_delay and the set_external_delay command in both ideal and propagated modes.

Use the set_clock_insertion_delay command in two ways:

- To specify clock insertion delay for a clock waveform:
  - Use this form for source delay associated with a clock signal:

    ```
    set_clock_insertion_delay -source [-lead | -trail] [-early | -late]
    [-min | -max] insertion_delay list_of_clocks
    ```

  - Use this form for network delay associated with a clock signal:

    ```
    set_clock_insertion_delay [-pvt {min | typ | max}] [-lead | -trail]
    [-min | -max] insertion_delay list_of_clocks
    ```

    Source and network insertion delays specified on the clock waveforms are picked up for both the set_input_delay and the set_external_delay assertions. Insertion delay specification on a port/pin overrides any previous insertion delay on the port/pin/waveform. The *list_of_clocks* is the list of the clock waveform names as specified by the set clock or the set generated clock command.
- To specify clock insertion delay on clock pins:
  - Use this form for source delay on a clock pin:

    ```
    set_clock_insertion_delay -source [-early | -late] [-rise | -fall]
    insertion_delay -pin list_of_pins
    ```

  - Use this form for network delay on a clock pin:

    ```
    set_clock_insertion_delay [-pvt {min | typ | max}] [-rise | -fall]
    insertion_delay -pin list_of_pins
    ```

Note: In case of assertions on both the clock waveform and the clock pin, the specification on the clock tree pin has priority over that on the clock waveform. The specification on the clock tree pin furthest down the clock tree decides the insertion delay to a register. A balanced clock tree is assumed.

During ideal propagation, the last specification along a path wins. This is useful when you want to intentionally skew the clock tree and model the physical hierarchical clock tree.

In rare cases, more than one clock generator is connected to the input port., for example CLK1 and CLK2 both connect to clock port clkA. The -clock option makes the assertion for the named clock. Use the -clock option only with -source option. Only source delay can be specified from the clock generator to the input port.

```
set_clock_insertion_delay -source -clock clock_name [-early | -late]
[-rise | -fall] insertion_delay -pin list_of_pins
```

Options and Arguments

`-clock` *`clock_signame`*

Specifies that the source insertion delay is caused by the named clock waveform. Used when more than one clock generator is connected to the port.

`-early | -late`

Specifies clock arrival time with respect to the early (setup) or the late (hold) time of the clock signal. If neither option is specified, the default is both early and late. Use these options only with the `-source` option.

`insertion_delay`

Specifies the insertion delay value. Floating point.

`-lead | -trail`

Specifies insertion delay at the leading or trailing edge of the clock waveform.
Default: Both edges

`list_of_clocks`

Specifies the list of clock waveform names to associate with the clock insertion delay.

`-pin` *`list_of_pins`*

Specifies the list of pins to associate with the clock insertion delay. The pin can be a port or an instance pin. These can either be object IDs or hierarchical names relative to the current module.

For internal pins, both the source and network delay must be specified.

`-min | -max`

Specifies the clock insertion delay for a particular operating

corner. Apply the `-min` option to the minimum operating corner, and `-max` to the maximum operating corner.

`-rise | -fall`

Specifies insertion delay at the rising or falling edge of the clock.

`-source`

Specifies that the delay is a source delay. Without this option, the delay is considered a network delay.

Examples

```
set_clock CLK1 -period 10 # Define the ideal clock CLK1
set_clock CLK2 -period 20 # Define the ideal clock CLK2
set_clock_root -pos -clock CLK1 clkA
set_clock_root -pos -clock CLK2 {clkA clkB clkD}
set_clock_insertion_delay -source 2 -pin {clkB clkD}
set_clock_insertion_delay -rise 5 -pin {clkB clkD}
set_clock_insertion_delay -fall 7 -pin {clkB clkD}
set_clock_insertion_delay -source -clock CLK2 5 -pin clkA
```

Related Information

reset_clock_insertion_delay

reset_clock_uncertainty

set_clock

set_clock_root

set_clock_propagation

set_clock_uncertainty

set_data_required_time

set_external_delay

See Specifying Clock Insertion Delay in the *Common Timing Engine (CTE) User Guide*.

set_clock_propagation

```
set_clock_propagation
     {ideal | propagated}
```

Tells the timing analyzer to treat all clock signals associated with the current timing top module as either `ideal` or `propagated` signals for the purpose of timing analysis. One of the signal types must be specified when using this command. If this command is not used, the default is `ideal`.

Important

if you backannotate delays to the clock network, use the `propagated` mode to enable them in CTE.

Options and Arguments

`ideal`

Ignores actual clock tree network latency and uses network latency asserted by the `set_clock_insertion_delay` command. The total delay is the sum of the clock source latency and the asserted network latency. This is the default when the command is not issued.

`propagated`

Uses the actual clock tree delays in the timing of the circuit. The total delay is the sum of the clock source latency and the backannotated clock tree delay. Use propagated mode for postlayout analysis when you have accurate clock tree information in the form of an SDF file, or backannotated net capacitance and resistance.

Example

```
set_clock_propagation propagated
```

Related Information

`get_clock_propagation`

`get_propagated_clock`

`reset_propagated_clock`

set_clock

set_clock_insertion_delay

set_clock_root

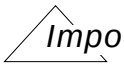
set_propagated_clock

set_top_timing_module

See Setting the Clock Propagation Mode for Analysis in the *Common Timing Engine (CTE) User Guide*.

set_clock_required_time

```
set_clock_required_time
      OBSOLETE: Use set_external_delay instead.
```



Important

Do not use the `set_clock_required_time` command, use the `set_external_delay` command instead. Cadence timing analysis still writes out `set_clock_arrival_time` during time budgeting. The following description is for reference only.

```
set_clock_required_time
-clock clock_name [-pos | -neg] [-late | -early] [-rise | -fall]
time pin_list
```

The internally generated `set_clock_required_time` command specifies the required time that the output clock ports must be stable.

set_clock_root

`set_clock_root [-clock` *`ideal_clock_name`*`] [-pos | -neg]` *`list_of_pins`*

Assigns a polarity to a previously specified ideal clock and associates a list of clock pins or ports with the ideal clock signal.

When a clock root assertion is present on an internal circuit pin, one that is not an input port, all incoming signals are blocked. Setting the `set_clock_root` command stops slew propagation, so the slew at this point is one of the following:

- The value specified using the `set_slew_time` command.
- The default slew value specified using the `set_default_slew` command.
- 0

Options and Arguments

`-clock` *`ideal_clock_name`*
: Specifies the name of the ideal clock as specified by the `set_clock` command.

`list_of_pins`
: Specifies the list of pins to associate with an ideal clock waveform. The pin can be a port or an instance pin. These can either be object IDs or hierarchical names relative to the current module. Pins can be associated with more than one ideal clock signal.

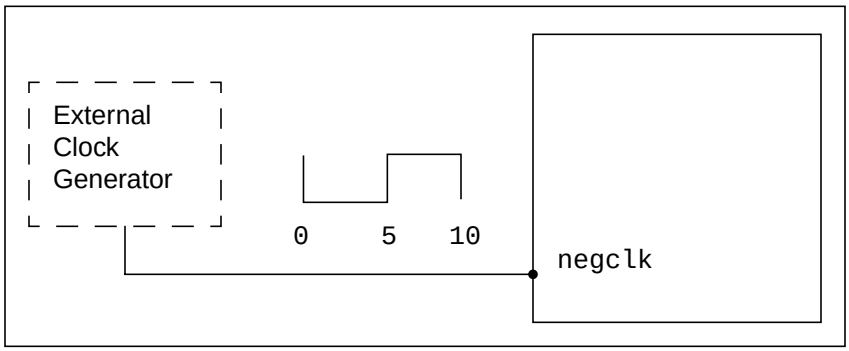
`-pos | -neg`
: Specifies a positive or negative ideal clock waveform. *Default*: `-pos` (rising edge precedes falling edge).

Example

The following command puts the negative waveform of ideal clock named `master` on the input port named `negclk` shown in Figure 7-12. The ideal clock `master` is shown in Figure 7-9:

```
set_clock_root -clock master -neg negclk
```

Figure 7-12 Defining a Clock Pin



Related Information

set_clock

set_clock_insertion_delay

set_clock_uncertainty

set_external_delay

set_clock_transition

`set_clock_transition [-rise | -fall] [-min |-max] slew_time clock_list`

Specifies the slew time of register clock pins. This is useful for pre-layout static timing analysis before clock tree synthesis (CTS) is performed since net delays can be inaccurate with large fanout nets.

Use this command only with ideal clocks. In propagated mode, CTE calculates slew times on register clock pins.

Options and Arguments

`-rise |-fall`
: Specifies the rising (or falling) edge of the register clock pins on which the slew time is asserted.

`-min | -max`
: Specifies the clock slew time for minimum or maximum process corner slew time.

`slew_time`
: Specifies the slew time of the clock pins.

`clock_list`
: Specifies a list of clocks only with ideal mode in the design, which affects the slew times on all register clock pins in the transitive fanout of the specified clocks. Specified `set_clock_transition` values on register clock pins in the fanout of a clock with propagated latency are ignored.

Example

The following example specifies a `2.0` rise slew time and a `1.0` fall slew time for register clock pins clocked by `CLKA`:

```
set_clock_transition -rise 2.0 [get_clocks CLKA]
set_clock_transition -fall 1.0 [get_clocks CLKA]
```

set_clock_uncertainty

```
set_clock_uncertainty [-ideal | -propagated] [-early | -late]
     {([-clock_from clksig_from] [-clock_to clksig_to]
     [-edge_from {leading | trailing}] [-edge_to {leading | trailing}]) |
     (-pin list_of_clock_tree_pins [-rise] [-fall] [-to])} float
```

Specifies the clock uncertainty (skew) between two clock edges for one or more paths. The command has two mutually exclusive forms, one for uncertainty on the clock waveform and one for uncertainty on clock pins.

- Specify uncertainty for clock waveforms using this set of options:

  ```
  set_clock_uncertainty [-ideal | -propagated] [-early | -late]
  [-clock_from clk_from] [-clock_to clk_to]
  [-edge_from {leading | trailing}] [-edge_to {leading | trailing}] float
  ```

  The specified uncertainty applies to paths starting from the `clock_from` signal and ending at registers clocked by the `clock_to` signal.

  The `edge_from` parameter is the type of edge for the launching clock of the path. The `edge_to` parameter is the capturing clock edge type of the destination register.

  Use this command to add optimism by using negative values for the clock uncertainty value. This is convenient when the design has paths starting in one clock domain and ending in another clock domain, and there is no synchronous relationship between the two clocks.

- Specify clock uncertainty on a clock pin using this set of options:

  ```
  set_clock_uncertainty [-ideal | -propagated] [-early | -late]
  -pin list_of_clock_tree_pins [-rise] [-fall] [-to]
  float
  ```

  Uncertainty on a clock pin affects paths downstream from that pin. By default, an uncertainty specification on pin `X` is honored only at those registers or latches where there is a path from pin `X` to both the data pin and the clock pin of the register or latch. However, the `-to` option lifts this restriction and makes it applicable to all registers and latches whose clock pin is in the transitive fanout of pin `X`. Pin-based uncertainty has priority over clock-based uncertainty. Among all applicable pin-based uncertainty specifications, the one furthest down the clock tree has the highest priority.

- Priority of setting multiple assertions using the `set_clock_uncertainty` command is as follows:

  - `set_clock_uncertainty -clock_from -clock_to`
  - `set_clock_uncertainty -clock_from`
  - `set_clock_uncertainty -clock_to`

- `set_clock_uncertainty`

When setting multiple assertions with the same options, the latest command is applied and the previous value is overwritten.

Options and Arguments

`-clock_from` *`clksig_from`*

Specifies the starting point of the paths for which the uncertainty is specified. The *`clksig_from`* argument is the name of clock signal with respect to which the uncertainty is specified.

If the `-clock_from` option is omitted then the uncertainty applies to all the paths ending at the registers clocked by the `-clock_to` signal.

`-clock_to` *`clksig_to`*

Specifies the endpoint of the paths for which the uncertainty is specified. The specified uncertainty applies to all paths starting from the `-clock_from` signal and ending at the registers clocked by `-clock_to` signal.

If the `-clock_to` option is omitted then the uncertainty applies to all paths starting at the `-clock_from` signal.

If both the options `-clock_from` and `-clock_to` are omitted, then the specified uncertainty applies to all clocked paths in the design, including combinational paths constrained by the `set_input_delay` and `set_external_delay` times associated with the clock.

`-early | -late`

Specifies whether the clock uncertainty is with respect to early times or late times. The `-late` option applies to setup, and the `-early` option applies to hold checks on the clock signals.
Default: `-early` and `-late`

`-edge_from {leading | trailing}`

Specifies whether the triggering (launching) edge of the clock is a leading or a trailing edge. It must match the edge used in the timing check for the destination register.
Default: `leading` edge.

`-edge_to {leading | trailing}`
Specifies whether the capturing edge at the destination register is a leading or a trailing edge.
Default: `leading` edge

`float`
Specifies the uncertainty value.

`-ideal | -propagated`
Specifies whether the uncertainty applies to the `ideal` clock propagation mode or the `propagated` mode of analysis. If neither option is given, the uncertainty applies in both ideal and propagated modes by default.

`-pin` *`list_of_clock_tree_pins`*
Applies the uncertainty to the given list of clock pins. Each pin can be a port or an instance pin. It can either be given as an object ID or a hierarchical name relative to the current module. Cannot be used with `-clock_from` or `-clock_to` options.

`-rise | fall`
Specifies the rising (or falling) edge of the clock pins on which the uncertainty is asserted. For each edge, the assertion is used only if that edge fans out to the data and the clock pin of a register or latch. With the `-to` option, the assertion is used if that edge fans out to the clock pin of a register or latch.
Default: Both edges.

`-to`
Specifies that the uncertainty applies to all the registers and latches whose clock pin is in the transitive fanout of a clock pin from the *`list_of_clock_tree_pins`*.

Examples

- The following command gives all paths starting from registers clocked by `c1` to registers clocked by `c2` a hold (early) uncertainty of 2.0 when the triggering edge is the leading edge and the capturing edge is the trailing edge:

```
set_clock_uncertainty -clock_from c1 -clock_to c2 \
-edge_from leading -edge_to trailing -early 2.0
```

  In case of multiple assertions, the actual uncertainty applied by CTE depends on which constraint matches the specific path in the question. CTE may also apply the worst case uncertainty in some situations, as shown in the following example.

- The following commands cause a path starting at a register clocked by `c1` and ending at a register clock by `c2` to have an uncertainty of 3.0 (which is the worse of the two).

```
set_clock_uncertainty -clock_from c1 2.0 -late
set_clock_uncertainty -clock_to c2 3.0 -late
```

- The following command sets an uncertainty on a clock pin (`clk_in`) only for propagated mode.

```
set_clock_uncertainty -propagated -pin clk_in 1.2
```

Related Information

`set_clock`

`reset_clock_uncertainty`

Specifying Clock-Based Uncertainty in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

set_constant_for_timing

```
set_constant_for_timing {0 | 1} { -rising | -falling} {-bidi_input | -bidi_output}
     list_of_ports_or_pins
```

Sets the value of the specified pin(s) to either a 1 or 0 for use by the timing engine. The applied state value propagates through the corresponding combinational logic cone(s) and the enable or disable sections of logic. Constants do not propagate through sequential logic, however you can set the constant at the sequential output. Constants can also be applied at primary IO and hierarchical ports.

Controls the transparency of a latch. Constants can also be applied to clear or preset a latch. For more information, see Specifying Latch Transparency in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

Note: Any paths considered to be false paths because of `set_constant_for_timing` will not be optimized. To set a constant for synthesis, use the `set_logic0` or `set_logic1` command.

Options and Arguments

`0 | 1`

Specifies a logic value of 0 or 1, which is propagated through combinational logic to disable or enable parts of logic.

`-bidi_input | -bidi_output`

Specifies that the constant is set on the input or output part of bidirectional pins. The default places the constant on the input part.

`list_of_ports_or_pins`

Specifies a list of ports or pins on which to apply the value. To specify multiple pins, enclose the list with curly braces (`{}`) and separate the pin names with white space.

`-rising | -falling`

Specifies a relevant transition value to be considered by the timing engine. For example, if rising is specified, the opposite transition type (falling) is ignored.

Example

The following command sets the value of pin `i4/A` to 1 for use by the timing engine:

```
set_constant_for_timing 1 i4/A
```

Related Information

`check_timing -type`

`const_collision`

`const_contradiction`

`get_constant_for_timing`

`reset_constant_for_timing`

`set_case_analysis`

`set_disable_timing`

`set_false_path`

Setting Constant Values for Timing in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

set_cycle_addition

```
set_cycle_addition [{-from | -from_rise | -from_fall} pin_list]
     [{-through | -through_rise | -through_fall} pin_list]
     [{-to | -to_rise | -to_fall} pin_list] [-clock_from list_of_clksigs]
     [-edge_from {leading | trailing}] [-clock_to list_of_clksigs]
     [-edge_to {leading | trailing}] [-early | -late]
     [-bidi_input_from | -bidi_output_from]
     [-bidi_input_through | -bidi_output_through]
     [-bidi_input_to | -bidi_output_to]
     [-target][-reset_path] float
```

Identifies multicycle paths in the design for timing analysis. A path is a multicycle path when the signal propagation from the start point to the endpoint of the path takes more than one clock cycle. Normally, timing analysis takes place over one clock cycle, and by default all paths are considered one cycle paths. This command lets you add a certain number of cycles to the path under consideration. Timing analysis is then done for the total number of cycles, thereby eliminating violations caused by inadequate analysis time. The information about number of cycles for a path is attached to the pins of the path.

All paths starting in one clock domain and ending in another clock domain are taken as multicycle paths. You cannot have the `-clock_from` and `clock_to` options in the same command with the `-from` and `to` options.

If both the `-from` and `-clock_from` options are specified, the path exception applies to all paths starting from the `-from` pin and all the paths starting from the `-clock_from` signal. If both the `-to` and `-clock_to` options are specified, the path exception applies to all the paths ending at the `-to` pin and all the paths ending at the `-clock_to` signal.

If both the `-through` and `-clock_from` options are specified, the path exception applies to all paths starting from the `-clock_from` signal and going through the `-through` pin (AND operation). If both the `-through` and `-clock_to` options are specified, the path exception applies to all paths going through the `-through` pin and ending at the `-clock_to` path can have more than one path exception. Multiple path exceptions that match a given path are prioritized. The adjustment on the path is from the path exception with the highest priority. Table 7-1 on page 797 ranks path exception priorities from highest to lowest. Examples are provided in “Examples of Path Exception Priorities” on page 798.

The cycle addition values are stored at the `-to` pins if the `-to` option is specified, otherwise values are stored at the `-from` pins. If neither the `-to` nor `-from` option is specified, values are stored at the `-through` pins.

Specify a group of pins by enclosing the group with curly braces (`{}`) within one `set_cycle_addition` command.

Tip

To add cycles for any path beginning at `p1` or `p2` and terminating at `t1` or `t2`, group the pins as follows:

```
set_cycle_addition -from {p1 p2} -to {t1 t2} 2.2
```

Note: This method of pin grouping reduces the number of exceptions. Having large numbers of exceptions adversely impacts the run time of `report_timing`.

Options and Arguments

`-bidi_input_from | -bidi_output_from`
: Specifies that the assertion applies to the input or output of bidirectional pins in the `-from` pin list. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_through | -bidi_output_through`
: Specifies that the assertion applies to the input or output of bidirectional pins in the `-through` pin list. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_to | -bidi_output_to`
: Specifies that the assertion applies to the input or output of bidirectional pins in the `-to` pin list. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-clock_from` *`list_of_clksigs`*
: Adjusts the cycle count for all paths originating from the specified ideal (or generated) clock signals.

`-clock_to` *`list_of_clksigs`*
: Adjusts the cycle count for all paths going to the specified ideal (or generated) clock signals.

`-early | -late`
: Specifies whether the cycles added are with respect to the early (hold) or the late (setup) times of the data signal. If neither option is specified, the `-late` option is the default.

`-edge_from {leading | trailing}`
: Specifies an edge for the `-clock_from`. This option can only be

used with the `-clock` options.
Default: Both edges

`-edge_to {leading | trailing}`
Specifies an edge for the `-clock_to` option.
Default: Both edges

`float`
Specifies the number of extra cycles that should be added to the path. The number of cycles to be added for an *`N`* cycle path is *`N`*`-1`, because all paths by default are single cycle paths. The number of cycles can be negative, thus reducing the number of cycles for the path.

Note: Although a floating point number is allowed, if you pass the constraint to downstream tools with GCF, the value must be an integer.

`{-from | -from_rise | -from_fall}` *`pin_list`*
Specifies the pins that are at the start of the paths. If the `-from` option is used without the `-to` option, all paths originating from the *`pin_list`* are considered as multicycle paths.

Use either the `-from`, `-to`, or `-through` option with the `set_cycle_addition` command to identify the paths. Specifying both the `-from` and `-to` options identifies a path with specific starting and ending points.

The *`pin_list`* variable can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified as the object for the `-from*` options.

Use only one `-from` (or `-from_rise` or `-from_fall`) option per command.
Default: The `-from` option applies the assertion to both the rising and the falling edges.

Using the `-from_rise` option applies the assertion at the rising edge of the signal on the `-from` pins.

Using the `-from_fall` option applies the assertion at the falling edge of the signal on the `-to` pins.

`-reset_path`

Changes the constraint on the specified path to the new value. See Examples.

`-target`

The cycle adjustment is calculated using the cycle time of the capturing (target) clock.
Default: The no `-target` option, which uses the launching clock cycle time.

`{-through | -through_rise | -through_fall}` *`pin_list`*

Specifies the pins that the path goes through. When used without the `-from` or `-to` option, all paths that go through the `-through` pins are multicycle paths.

When used with both the `-from` and `-to` options, any path starting at any pin on the `-from` list *and* going through any point on the `-through` pin list *and* going to any point on the `-to` list is affected by the assertion.

The *`pin_list`* can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries.There can be only one `-from` and one `-to` option but many `-through` options in the same command.

By default, the `-through` option applies the assertion to both the rising and the falling edges.

Using the `-through_rise` option applies the assertion at the rising edge of the signal on the through pins.

Using the `-through_fall` option applies the assertion at the falling edge of the signal on the through pins.

`{-to | -to_rise | -to_fall}` *`pin_list`*

Specifies the pins that are at the end of the paths. If the `-to` option is used without the `-from` option, all paths ending in the `-to` pin list are considered as multicycle paths.

The *pin_list* can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified as the object for the `-to*` options.

Use only one `-to` (or `-to_rise` or `-to_fall`) option per command.

By default, the `-to` option applies the assertion to both the rising and the falling edges.

Using the `-to_rise` option applies the assertion at the rising edge of the signal on the through pins.

Using the `-to_fall` option applies the assertion at the falling edge of the signal on the through pins.

Examples

- The following command specifies all paths from port `Q` of instance `I1` as 3-cycle paths (additional 2 cycles). The paths can end anywhere in the design.

  ```
  set_cycle_addition -from I1/Q 2
  ```

- The following command sets all paths ending at port `A` of instance `I2` as 2-cycle paths. The paths can start anywhere in the design.

  ```
  set_cycle_addition -to I2/A 1
  ```

- The following command sets all paths from port `Q` of instance `I1` to port `B` of instance `I3` as 2-cycle paths. The paths can go through any pins in the design.

  ```
  set_cycle_addition -from I1/Q -to I3/B 1
  ```

- The following command sets only the path from port `Q` of instance `I1` through port `Q` of instance `I2` to port `B` of instance `I3` as a 3-cycle path.

  ```
  set_cycle_addition -from I1/Q -through I2/Q -to I3/B 2
  ```

- The following command sets a different number of cycles on the early and late paths using these commands:

  ```
  set_cycle_addition -late -from I1/Q 2
  set_cycle_addition -early -from I1/Q 1
  ```

Tip

Be careful of multiple cycle additions on the same path. The priority rules say that the most constraining value is used. For example:

```
set_cycle_addition -late -from I1/Q 2
report_path_exceptions -from I1/Q
```

```
+-----------------------------------------------+
|             From             | Early |  Late |
|------------------------------+-------+-------|
| I1/Q                         |       | add 2  |
+-----------------------------------------------+
```

```
set_cycle_addition -late -from I1/Q 3
report_path_exceptions -from I1/Q
```

```
+-------------------------------------------------------+
|               From               | Early |     Late      |
|----------------------------------+-------+---------------|
| I1/Q                             |       | add 3 ignored |
| I1/Q                             |       | add 2         |
+-------------------------------------------------------+
```

- Using the -reset_path option changes the constraint on the specified path to the new value. However, if there is only one path and both paths are equivalent, using this option resets a path only when there is an exact match in the arguments. For example, the following command specifies the delay as U14/A 5:

  ```
  set_cycle_addition -through U14/A 5
  ```

  The following shows the report using this command:

  ```
  +---------------------------+
  | Through | Early |  Late   |
  |---------+-------+---------|
  | U14/A   |       | delay 5 |
  +---------------------------+
  ```

  If you then change the constraint to a new value using the -reset_path option on the same path, but use a different argument as follows:

  ```
  set_cycl_addition -through U12/Z 7 -reset_path
  ```

  The report using the report_path_exceptions command or the report_timing command shows the following:

  ```
  +-----------------------------------+
  | Through | Early |      Late       |
  |---------+-------+-----------------|
  | U12/Z   |       | delay 7 ignored |
  | U14/A   |       | delay 5         |
  +-----------------------------------+
  ```

  Using the -reset_path option resets a path only when there is an exact match in the arguments, not when both the constraints represent the same path.

Related Information

`reset_path_exception`

`report_timing -format addition`

`set_false_path`

set_data_arrival_time

```
set_data_arrival_time
     OBSOLETE: Use set_input_delay instead.
```

Do not use the `set_data_arrival_time` command, use the `set_input_delay` command instead. Cadence timing analysis still writes out `set_data_arrival_time` during time budgeting. The following description is for reference only.

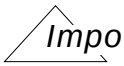
```
set_data_arrival_time
[-clock clock_name]
[-lead | -trail] [-early | -late] [-rise | -fall] [-bidi_input | -bidi_output]
time pin_list
```

The internally generated `set_data_arrival_time` command attaches arrival times to input ports in the database.

Note: The arrival time is always specified as an absolute time with respect to the ideal clock time zero, even if the pertinent edge of the ideal clock is not at time zero. When you use the `set_input_delay` command, the time is relative to the clock.

set_data_required_time

```
set_data_required_time
     OBSOLETE: Use set_external_delay.
```



Do not use `set_data_required_time` command, use the `set_external_delay` command instead. See set_external_delay on page 1167 for more details.

Cadence timing analysis still writes out `set_data_required_time` during time budgeting. The following description is for reference only.

```
set_data_required_time
[-clock clock_name]
[-lead | -trail] [-early | -late] [-rise | -fall] [-bidi_input | -bidi_output]
time pin_list
```

The internally generated `set_data_required_time` command specifies the required time that the output data ports must be stable.

The required time is always specified as an absolute time with respect to the ideal clock time zero, even if the pertinent edge of the ideal clock is not at time zero. When you use the `set_external_delay` command, the time is relative to the clock.

set_dcl_calculation_mode

```
set_dcl_calculation_mode [-mode {best_case | nominal_case | worst_case}]
```

Sets the current Delay Calculation Language (DC)L calculation mode. If this command is not specified, the default is `worst_case`. Load the Delay Calculation Module (DCM) first with the `load_dcl_rule` command.

The library developer defined the calculation modes in the DCL library to model three process conditions, `best_case`, `nominal_case`, and `worst_case`, within one library. Refer to the vendor documentation for more information.

Note: Process, temperature, and voltage are set independently of the calculation mode using the `set_operating_parameter` command.

Options and Arguments

`-mode {best_case | nominal_case | worst_case}`
Specifies which calculation mode to use for analysis.
Default: `worst_case`

Example

The following command lets the current Delay Calculation Language (DCL) calculation mode to `nominal_case:`

```
set_dcl_calculation_mode -mode nominal_case
```

Related Information

`get_dcl_calculation_mode`

`get_operating_parameter`

`load_dcl_rule`

`reset_dcl_calculation_mode`

`set_operating_parameter`

See Using IEEE 1481 Delay and Power Calculation System (DPCS) Libraries in the *Common Timing Engine (CTE) User Guide* for more information.

set_dcl_functional_mode

`set_dcl_functional_mode [-group` *`group_name`*`] [-mode` *`mode_name`*`]` *`instance_list`*

Sets the Delay Calculation Language (DCL) functional mode by group and name on hierarchical instances.To determine what modes are available for an instance, use the command `get_dcl_functional_mode_array` on the underlying library cell of that instance.

Functional modes defined in a DCL library let the vendor library developer model the timing arcs and timing checks for a technology cell in a variety of ways. For example, a scan flip-flop could have functional modes defined for normal flip-flop behavior and for scan behavior with different timing arcs and checks for each mode.

Options and Arguments

`-group` *`group_name`*
: Specifies the functional mode group name.

`instance_list`
: Specifies a list of instances for which the functional mode needs to be set.

`-mode` *`mode_name`*
: Specifies the functional mode name.

Example

```
set_dcl_functional_mode -group rw -mode read {A1/B3/C2 A1/B2/C2}
```

Related Information

`get_dcl_functional_mode`

`get_dcl_functional_mode_array`

`load_dcl_rule`

`reset_dcl_functional_mode`

See Using IEEE 1481 Delay and Power Calculation System (DPCS) Libraries in the *Common Timing Engine (CTE) User Guide* for more information.

set_dcl_level

```
set_dcl_level [-perfLevel {0 | 1}]
```

Sets the desired performance level for Delay Calculation Language (DCL) based delay calculations. A value `0` indicates lower accuracy in favor of faster run time while a value of `1` indicates calculations for maximum accuracy. If this command is not issued, the default performance level is `0`.

Options and Arguments

`-perfLevel {0 | 1}`

Specifies the performance level. DCL supports two performance levels for delay calculation, where one performance level targets faster run time with less accurate timing, while the other targets more accurate timing with slower run time.

The library vendor who provided the Delay Calculation Module (DCM) also documents the characteristics of each performance level in their library and whether or not both 0 and 1 levels are supported and which is faster (usually `0`). In general, the faster performance in terms of run time should be used for initial synthesis and the more accurate level should be used in final timing correction.

Example

The following command sets the desired performance level to 1:

```
set_dcl_level -perfLevel 1
```

Related Information

`get_dcl_level`

`load_dcl_rule`

See Using IEEE 1481 Delay and Power Calculation System (DPCS) Libraries in the *Common Timing Engine (CTE) User Guide* for more information.

set_default_slew_time

```
set_default_slew_time [-early | -late] [-rise | -fall] float
```

Specifies the default slew time for the ports. The system uses this slew value for any input or bidirectional port for which slew or drive assertions are not specified. If this command is not used, the system uses a value of `0.0`.

The `set_default_slew_time` command is ignored for a port if `set_slew_time`, `set_drive_cell`, or `set_drive_resistance` constraints have been set for that port.

Options and Arguments

`-early | -late`

Indicates whether the slew times are with respect to the late (data setup) or the early (data hold) checks. If neither option is specified the default is to use both.

`float`

Specifies the default slew time for input and bidirectional ports that do not have slew or drive assertions on them.

`-rise | -fall`

Specifies that the slew time should be applied to the rising edge or the falling edge of the signal. If neither edge is specified, the default is to use both.

Examples

```
set_default_slew_time -rise -early 10.0
set_default_slew_time -fall -early 20.0
set_default_slew_time -rise -late 30.0
set_default_slew_time -fall -late 40.0
```

Related Information

`reset_default_slew_time`

`set_drive_cell`

`set_drive_resistance`

`set_slew_time`

set_disable_cell_timing

```
set_disable_cell_timing [-library library_name][ -cell cell_name]
     [-from_pin | -from_rise | -from_fall from_pin_name]
     [-to_pin | -to_rise | -to_fall to_pin_name]
```

Disables delay arcs at the cell level. The `set_disable_cell_timing` command effects all instantiations of the specified cell, and lets you disable delay arcs from and to internal pins.

Options and Arguments

`-cell` *`cell_name`*
: Specifies the cell in the library for which timing arcs are disabled.

`-from_pin`|`-from_rise`|`-from_fall` *`from_pin_name`*
: Specifies the source pin of the timing arc. You can disable cell arcs with specific edges using either the `from_rise` option or the `from_fall` option. All arcs specified using the `-from_pin` *`from_pin_name`* option are disabled.

`-library` *`library_name`*
: Specifies the library that contains the cell for which arcs are disabled. If not specified, the default is the target technology library.

`-to_pin`|`-to_rise`|`-to_fall` *`to_pin_name`*
: Specifies the sink pin of the timing arc. You can disable cell arcs with specific edges using either the `to_rise` option or the `to_fall` option. All arcs specified using the `-from_pin` *`from_pin_name`* option are disabled.

Examples

- The following command disables all arcs in cell `FD1`. Any instantiation of this cell in your design will not propagate timing:

  ```
  set_disable_cell_timing -cell FD1
  ```

- The following command disables all arcs originating from rising input pin `A`:

  ```
  set_disable_cell_timing -library lca300kv -cell AN2 -from_rise A
  ```

- The following command disables arcs with a specific transition. Arcs originating with rising from pin `A` and falling to pin `Y` are disabled.

  ```
  set_disable_cell_timing -cell XORS -from_rise A -to_fall Y
  ```

Related Information

reset_disable_cell_timing

set_constant_for_timing

set_disable_timing

set_false_path

set_disable_clock_gating_check

```
set_disable_clock_gating_check [-from from_pins] [-to to_pins]
     [-bidi_input_from | -bidi_output_from] [-bidi_input_to | -bidi_output_to]
     [-check_type check_type_list]

     Valid check types are: clock_gating_hold clock_gating_setup
```

Disables any clock gating check from, to, or between the specified pins. Disables only the inferred clock gating checks, which result when a data signal meets a clock signal through a combination gate. (See section describing inferred clock gating checks.)

Use the -check_type option to selectively disable the clock gating setup and clock gating hold.

Options and Arguments

`-bidi_input_from | -bidi_output_from`
: Specifies the assertion on the input or output part of the `from` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_to | -bidi_output_to`
: Specifies the assertion on the input or output part of the `to` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-check_type` *`check_type_list`*
: Specifies the check arc being disabled. If not specified, all arcs from, to, or between the specified pins are disabled. Valid check types are `clock_gating_setup` and `clock_gating_hold`.

`-from` *`from_pins`*
: Specifies the clock pin of the clock gating check to be disabled. If the `-from` option is used without the `-to` option, all clock gating checks that reference the *`from_pins`* are disabled. Either the `-from` or `-to` option must be used with this command. Specifying both the `-from` and `-to` options identifies a path with specific start and endpoints.

`-to` *`to_pins`*
: Specifies the data enable pin of the clock gating check to be disabled.

If the `-to` option is used without the `-from` option, all clock gating checks with *`to_pins`* as data signal are disabled.

Related Information

`reset_clock_gating_check`

`reset_disable_clock_gating_check`

`set_clock_gating_check`

set_disable_timing

```
set_disable_timing [-from from_pins] [-to to_pins]
     [-bidi_input_from | -bidi_output_from] [-bidi_input_to | -bidi_output_to]
     [-check_type check_type_list]

     Valid check types are: clock_gating_hold clock_gating_pulse_width
     clock_gating_setup clock_period clock_separation hold no_change_hold
     no_change_setup recovery removal setup skew
```

Disables (snips) timing arc(s) of an instance or interconnect and is used for breaking combinational loops.

Use the `set_disable_timing` command to control the transparency of a latch. For more information, see Specifying Latch Transparency in the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS)*.

This command differs from the `set_false_path` command as follows:

- `set_disable_timing` command snips a timing arc such that neither arrival times nor constant values propagate through it.
- `set_false_path` command selectively disables the arrival time information depending upon the precedence rules given in Table 7-1 on page 797. Constant values are not affected by the `set_false_path` command.

Options and Arguments

`-bidi_input_from | -bidi_output_from`
: Specifies the assertion on the input or output part of the `from` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_to | -bidi_output_to`
: Specifies the assertion on the input or output part of the `to` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-check_type` *`check_type_list`*
: Specifies the check arc being disabled. If not specified, all arcs from, to, or between the specified pins are disabled. Valid check types are `setup`, `hold`, `recovery`, `removal`, `clock_separation`, `clock_gating_setup`, `clock_gating_hold`, `clock_gating_pulse_width`,

`clock_period`, `pulse_width`, `no_change_setup`, `no_change_hold`, and `skew`.

`-from` *`from_pins`*

Specifies the source pin of the timing arc to be disabled.
If the `-from` option is used without the `-to` option, all paths originating from the *`from_pins`* are disabled.
Either the `-from` or `-to` option must be used with this command. Specifying both the `-from` and `-to` options identifies a path with specific start and endpoints.

The `-from` and `-to` options can be intermediate hierarchical boundaries.

If disabling check arcs, the `-from` pin must be a clock pin, and the `-to` pin must be a data signal pin.

`-to` *`to_pins`*

Specifies the sink pin of the timing arc to be disabled.

If the `-to` option is used without the `-from` option, all paths ending in the *`to_pins`* option are disabled.

Examples

- The following command disables all check arcs between `reg01/CP` and `reg01/D`:

```
set_disable_timing -from reg01/CP -to reg01/D
```

- The following command disables all check arcs and delay arcs from reg01/CP:

```
set_disable_timing -from reg01/CP
```

- The following command disables only setup check arcs attached to `reg01/CP`. Any delay arcs from `reg01/CP` will not be disabled:

```
set_disable_timing -from reg01/CP -check_type setup
```

Related Information

`reset_feedback_loop_snipped_arcs`

`set_disable_cell_timing`

`set_false_path`

set_drive_cell

```
set_drive_cell [-early | -late] [-library library_name]
     [-library_rise library_name] [-library_fall library_name]
     [-cell cell_name] [-cell_rise cell_name] [-cell_fall cell_name]
     [-pin pin_name] [-pin_rise pin_name] [-pin_fall pin_name]
     [-from_pin from_pin_name] [-from_pin_rise from_pin_name]
     [-from_pin_fall from_pin_name]
     [-rise_source_edge {R | F}] [-fall_source_edge {R | F}]
     [-clock clock_name] [-lead | -trail | -pos | -neg]
     [-source_slew slew_value] [-rise_source_slew rise_slew_value]
     [-fall_source_slew fall_slew_value] [-no_design_rule] port_list
```

Models the drive capability of an external driver connected to the input port. Use this command only for timing analysis. The command does not affect the electrical properties of the design, which means that the capacitance of the output pin of the driving cell does not add to the total capacitance of the input port.

The timing analyzer computes an offset to the arrival time of an input and also changes the slew time used to compute the delay of the cell on the sink of the net. The offset to data arrival time is computed by taking the total delay of the cell for C seen at the input (not including its own C_{driver}) and subtracting the total delay of the cell given C_{load} of zero. The total delay means the propagation and transition delays, if applicable. The transition delay itself is used as the slew value for the delay calculation of the next cell. Identify the library cell that drives the input of the design, and designate the cell as the driving cell for the ports in context using `set_drive_cell` command.

This command overrides the `set_drive_resistance` command, per port or per net, if it is the last command applied. Also, if `set_drive_cell` is set for a port, the `set_slew_time` command is ignored for that port.

Options and Arguments

`-cell` *`cell_name`*
: Specifies the name of the driver cell that provides both a rising edge transition driver and a falling edge transition driver.

`-cell_fall` *`cell_name`*
: Specifies the name of the cell that provides a falling transition driver.

`-cell_rise` *`cell_name`*
: Specifies the name of the cell that provides a rising transition driver.

`-clock` *`clock_name`*

Specifies the clock signal that is controlling the data signal.

`-early`

Specifies that the driver provides a signal for early mode analysis (data hold check).

`-fall_source_edge R | F`

Specifies whether the rising edge (`R`) or the falling edge (`F`) at the *`from_pin_name`* option is controlling the falling edge transition at the input port of the design. This option is required for non-unate timing arcs.

`-fall_source_slew` *`fall_slew_value`*

Specifies the slew value for the falling signal at the input of the drive cell.

`-from_pin_fall` *`from_pin_name`*

Specifies the name of the input port of the driver cell with the controlling timing arc to the output port of the driver cell that provides the falling edge transition drive to the input port of the design.

`-from_pin` *`from_pin_name`*

Specifies the name of the input port of the driver cell that has an arc to the output port of the driver cell, which is connected to the input port of the design. This option is required only if there are multiple timing arcs in the driver cell from its input ports to its output ports, and the arcs from those pins have different drive characteristics.

The `-from_pin` option assumes that the same timing arc between the input port and the output port of the driver cell is driving both the rising edge transition and the falling edge transition on the input port of the design. To specify one edge, use the `-from_pin_rise` or the `-from_pin_fall` option.

Without the `-from_pin*` option, the default is to use the first timing arc found on the library cell.

`-from_pin_rise` *`from_pin_name`*

Specifies the input port name of the driver cell with the controlling

timing arc to the output port of the driver cell that provides the rising edge transition drive to the input port of the design.

`-late`

Specifies that the driver provides a signal for late mode analysis (data setup check).

`-lead | -trail | -pos | -neg`

Specifies that the slew time is for a data signal triggered by the leading or the trailing edge of the clock waveform.
Default: `-lead`

For a clock signal, `-pos` and `-neg` specify that the slew time should be applied to an actual clock that has positive or negative polarity with respect to the ideal clock waveform.

`-library` *`library_name`*

Specifies the name of the library that contains the driving cell. This option is used if the same cell provides both the rising and the falling edge transitions, or if different cells provide the rising edge and falling edge transitions but are in the same library.

`-library_fall` *`library_name`*

Specifies the name of the library that contains the falling edge transition driver.

`-library_rise` *`library_name`*

Specifies the name of the library that contains the rising edge transition driver.

`-no_design_rule`

Prevents drive cell design rules from being used as design rules for ports where the drive cell is asserted. This option is valid only if the global `use_drive_cell_design_rules` is set to `true`.

`-pin_fall` *`pin_name`*

Specifies the name of the output port of the driver cell that provides the falling edge transition drive to the input port of the design.

`-pin` *`pin_name`*

Specifies the name of the output port of the driver cell that drives the signal at the input port of the design. This option is required

if there are multiple output ports on the driver. If there is only one output port on the driver cell, then using this option is not necessary.

`-pin_rise` *`pin_name`*

Specifies the name of the output port of the driver cell that provides the rising edge transition drive to the input port of the design.

`port_list`

Specifies the list of ports driven by the driver cell.

`-rise_source_edge R | F`

Specifies whether the rising (R) edge or the falling edge (F) at the *`from_pin_name`* option is controlling the rising edge transition at the input port of the design.

`-rise_source_slew` *`rise_slew_value`*

Specifies the slew value for the rising signal at the input of the drive cell. This option is required for non-unate timing arcs

`-source_slew` *`slew_value`*

Specifies the slew value for the signal at the input of the drive cell.

Examples

- The Current Module and Sub-Modules Example figure on page 874 shows an example module to help show how to use the `set_drive_cell` command. The following are the constraints that are specified on the example module and submodules:

```
set_clock CLK1 -period 3
set_clock_root -clock CLK1 "clkA clkB clkC clkD"
set_input_delay -clock CLK1 0.0  in
set_external_delay -clock CLK1 1.0  out
```

  The following commands model the drive capability of an external driver connected to the input port.The timing analyzer computes an offset to the arrival time of an input and also changes the slew time used to compute the delay of the cell on the sink of the net:

```
set_drive_cell -cell IV [ find -input -port * ]
set_drive_cell -cell AN3 -from_pin I0 input1
```

  The following command provides a detailed timing report based on this example:

```
check_timing -verbose
```

```
+---------------------------+
|    TIMING CHECK DETAIL    |
|---------------------------|
|  Pin  |      Warning      |
|-------+-------------------|
|  clkA | No drive assertion |
|  clkB | No drive assertion |
|  clkC | No drive assertion |
|  clkD | No drive assertion |
|  in   | No drive assertion |
+---------------------------+
```

The following commands show the timing results:

```
get_timing -rise in load                =>    0.0820
get_timing -fall in load                =>    0.0820
get_timing -rise in slew                =>    0.0000
get_timing -fall in slew                =>    0.0000
get_timing -rise in arrival             =>    0.0000
get_timing -fall in arrival             =>    0.0000
get_timing -rise I_block/A_reg/D slack  =>    2.8584
get_timing -fall I_block/A_reg/D slack  =>    2.8992
```

The following command sets the drive cell on port `in` for `rise`:

```
set_drive_cell -cell_rise AN2 -clock CLK1 -from_pin_rise A -pin_fall Z
-rise_source_slew 0.43 in
```

For this example, a two input one output gate is used. Make sure you specify the pin. If an input pin is not specified, the tool picks up the first defined pin in the library for this cell. The `drive_cell` assertion specifies the input pin as `A` and the specific edges that should be used for both the input and output pins to calculate the delay and slew values from the library tables.To understand how slew and delay values are calculated using library tables, see the Calculating Slew and Propagation Delay Values from Non-Linear 2D Tables example in the *Common Timing Engine (CTE) User Guide*.

Because the `set_drive_cell` command was set only on the `rise`, the following command displays a warning for the missing `drive_assertion` on the `fall`:

```
check_timing -verbose
```

```
+-------------------------------------------------------------+
|                     TIMING CHECK DETAIL                     |
|-------------------------------------------------------------|
|  Pin  |                       Warning                       |
|-------+-----------------------------------------------------|
|  clkA | No drive assertion                                  |
|  clkB | No drive assertion                                  |
```

```
|  clkC | No drive assertion                                      |
|  clkD | No drive assertion                                      |
|  in v | Missing specific drive assertion for signal CLK1 lead |
+-------------------------------------------------------------+
```

The slew value changes from `0.0000` to `0.0720` after setting the `drive_cell` assertion. These values will be used only for the `rise` signal; `fall` is still `0`.These new slew and delay values are calculated from the characterization tables for this cell in the library.

The following commands retrieves the timing information for the specified pin property on the given pin:

```
get_timing -fall in slew                =>    0.0000
get_timing -fall in arrival             =>    0.0000
```

The following commands show the changes in the arrival time, which leads to a change in `slack` but only on the `rise` because the `set_drive_cell` command was asserted only for the rising edge:

```
get_timing -rise I_block/A_reg/D slack   =>    2.7431
get_timing -fall I_block/A_reg/D slack   =>    2.8992
```

The following command sets the drive cell on port `in` for `fall`. If an inverter is used, you do not need to specify the pins. Specific edges are used to avoid any overrides by tool:

```
set_drive_cell -cell_fall IV -clock CLK1 -fall_source_edge R -fall_source_slew
0.35 in
```

Because the `set_drive_cell` command was set for the `fall` edge, the following `check_timing` command will not issue a `Missing drive assertion` for `in`:

```
check_timing -verbose
     +---------------------------+
     |    TIMING CHECK DETAIL    |
     |---------------------------|
     |  Pin  |      Warning      |
     |-------+-------------------|
     |  clkA | No drive assertion |
     |  clkB | No drive assertion |
     |  clkC | No drive assertion |
     |  clkD | No drive assertion |
     +---------------------------+
```

The following commands show that the `rise` values did not change but the `fall` values did:

```
get_timing -rise in slew                =>    0.0720
get_timing -rise in arrival             =>    0.0832
get_timing -fall in slew                =>    0.0256
```

```
get_timing -fall in arrival              =>    0.0487
get_timing -rise I_block/A_reg/D slack   =>    2.7431
get_timing -fall I_block/A_reg/D slack   =>    2.8421
```

Related Information

set_drive_resistance

set_slew_time

set_drive_resistance

```
set_drive_resistance [-clock clk_name] [-lead | -trail | -pos | -neg]
     [-rise | -fall] [-early | -late] [-slew_res slew_res_value]
     [-slew_intrinsic slew_intrinsic_value] value port_list
```

Provides a simpler version of the `set_drive_cell` command used in situations where the drive resistance is specified. Use this command only for timing analysis. This command does not affect the electrical properties of the design, and is also used to specify the drive resistance of a cell. The command computes an offset to the arrival time of an input and also changes the slew time used to compute the delay of the cell on the sink of the net.

The arrival time at the input port is modified by adding the RC constant to the specified arrival time at the input port. The RC constant is the capacitance (C) seen at the input port multiplied by the drive resistance (R). The RC value is used as the slew value for the delay calculation of the next cell.

Note: This command overrides a `set_drive_cell` command previously applied to the same port. Also, if the `set_drive_resistance` command is set for a port, the `set_slew_time` command is ignored for that port.

Options and Arguments

`-clock` *`clk_name`*

Specifies the name of the clock.

`-early | -late`

Specifies that the drive resistance should be applied to the early arrival time (data hold time) or late arrival time (data setup time) for timing analysis.

`-lead | -trail | -pos | -neg`

Indicates that the signal at the specified port must be recognized as a data signal with the specified association (`lead` or `trail`) with respect to the specified clock *`clock_name`*.
Default: `-lead`

Specifies for a clock signal, `-pos` and `-neg` that the slew time should be applied to an actual clock that has positive or negative polarity with respect to the ideal clock waveform.
Default: `-pos`.

`port_list`

Lists the ports for which drive resistance is specified.

`-rise | -fall`

Specifies that the drive resistance is applicable to only the rising edge or the falling edge transition at the input port. If neither rise nor fall options are specified then the resistance is applied to both transitions at the input port.

`-slew_intrinsic` *`slew_intrinsic_value`*

Specifies the intrinsic value of slew used in the slew computation at the input port of the module. The slew value at the input port of the module is computed as follows, assuming cap is the capacitance at this port:

```
slew = slew_intrinsic_value + slew_res_value *
cap
```

`-slew_res` *`slew_res_value`*

Specifies the value of resistance used in the slew computation at the input port of the module.

`value`

Specifies the resistance value.

Examples

- The `set_drive_resistance` command provides a simpler version of the `set_drive_cell` command used only in STA. The command computes an offset to the arrival time of an input and also changes the slew time used to compute the delay of the cell on the sink of the net.

  Note: This command overrides a `set_drive_cell` command previously applied to the same port. Also the `set_slew_time` command is ignored for that port.

  Current Module and Sub-Modules Example figure on page 874 provides an example of how to use the `set_drive_resistance` command and related commands.

  The following are the constraints applied to the design shown in Figure 7-3:

  ```
  set_clock CLK1 -period 3
  set_clock_root -clock CLK1 "clkA clkB clkC clkD"
  set_input_delay -clock CLK1 0.0  in
  set_external_delay -clock CLK1 1.0  out
  ```

  The following command provides a detailed timing report, which shows that no specified drive assertions:

  ```
  check_timing -verbose
  ```

```
+---------------------------+
|    TIMING CHECK DETAIL    |
|---------------------------|
|  Pin  |      Warning      |
|-------+-------------------|
|  clkA | No drive assertion |
|  clkB | No drive assertion |
|  clkC | No drive assertion |
|  clkD | No drive assertion |
|  in   | No drive assertion |
+---------------------------+
```

The following commands gets the timing information for the specified pin property on the given pin. Specify the pin by the object ID or the hierarchical name:

```
get_timing -rise in load                =>    0.0820
get_timing -fall in load                =>    0.0820

get_timing -rise in slew                =>    0.0000
get_timing -fall in slew                =>    0.0000
get_timing -rise in arrival             =>    0.0000
get_timing -fall in arrival             =>    0.0000

get_timing -rise I_block/A_reg/D slack   =>    2.8584
get_timing -fall I_block/A_reg/D slack   =>    2.8992
```

The following command sets the drive resistance on port `in`:

```
set_drive_resistance -clock CLK1 -rise -slew_res 0.03 -slew_intrinsic 0.08 0.9
in
```

Because the `set_drive_resistance` command is set using the `-rise` option, the `check_timing` command shows a warning because the `drive_assertion` is not also set using the-`fall` option, as shown in the following report:

```
check_timing -verbose
```

```
 +-------------------------------------------------------------+
 |                    TIMING CHECK DETAIL                      |
 |-------------------------------------------------------------|
 |  Pin  |                       Warning                       |
 |-------+-----------------------------------------------------|
 |  clkA | No drive assertion                                  |
 |  clkB | No drive assertion                                  |
 |  clkC | No drive assertion                                  |
 |  clkD | No drive assertion                                  |
 |  in v | Missing specific drive assertion for signal CLK1 lead |
 +-------------------------------------------------------------+
```

The slew value is calculated using the `slew_res`, `slew_intrinsic`, and `load` values:

= slew_intrinsic + (slew_res * load)

= 0.08 + (0.03 * 0.0820)

= 0.082460 rounded to 0.0825

The port delay value is calculated using the `load` and the `set_drive_resistance` values:

= load * set_drive_resistance_value

= 0.0825 * 0.9

= 0.0738

The slew value changes from `0.0000` to `0.0825` after setting the `drive_assertion`. This is only for the rise signal; fall is still `0`.

The following commands gets the values, whose calculations are shown above:

```
get_timing -rise in slew                =>    0.0825
get_timing -rise in arrival             =>    0.0738
```

The following commands show that for `fall` the value of slew is still `0.0000` because the `drive_resistance` is specifically set on `rise`:

```
get_timing -fall in slew                =>    0.0000
get_timing -fall in arrival             =>    0.0000
```

The following commands show the changes in the arrival time leads to a change in `slack` but only on the `rise`, since the `set_drive_resistance` command was asserted only for the rising edge:

```
get_timing -rise I_block/A_reg/D slack   =>    2.7478
get_timing -fall I_block/A_reg/D slack   =>    2.8992
```

Related Information

See Explaining Drive Adjustment in the Common Timing Engine (CTE) User Guide.

`set_drive_cell`

`set_slew_time`

set_external_delay

```
set_external_delay [-rise | -fall] [-late | -early] [-sig | -ref] [-lead | -trail]
     [-clock clock_name] [-arrival float] [-source_insertion_delay_included]
     [-network_insertion_delay_included] [-worst_case] external_delay
     port_or_pin_list
```

Provides an alternate way to specify the required time that the output ports must be stable by using external delay requirements. This constraint considers the timing requirement of the configuration of the output port and the connection to the register input of the external block.

When you know the critical path from the output port of the current block to the register of the external block, you can specify required time at the output with this command. The required time is derived from the port and path configuration.

A successive `set_external_delay` assertion on the same output port with respect to the same clock overrides a previous assertion. If multiple `set_external_delay` assertions are used on the same output port, but with respect to different clocks, then the worst case required time is used for timing analysis.

Options and Arguments

`-arrival` *`float`*

Specifies the actual arrival time of the edge of the clock waveform used to determine the external delay. This is used if the arrival time of the capturing clock signal is different than the arrival time of the corresponding edge of the ideal (or generated) clock. This option is similar to specifying the actual clock arrival time using the `set_clock_insertion_delay` command in relation to the ideal (or generated) clock signal defined by the `set_clock` (or `set_generated_clock`) command.

The `-arrival` option is an archaic method of specifying latency for the clock network for the fictitious flops connected to the output ports. It is used to set the absolute clock arrival time for those flops, which requires you to calculate the clock reference time plus latency along the clock path. With the `set_clock_insertion_delay` command, only the network latency needs to be given for those clocks. You do not need to do any arithmetic. The `-arrival` option is still available, and if it is given, this option overrides the `set_clock_insertion_delay` specification.

`-clock` *`clock_name`*

Specifies the ideal clock signal that is controlling the external register. A generated clock cannot be used with this option. *Default*: Asynchronous (@) clock

`-early | -late`

Specifies that the constraint refers to the early (data hold/clock setup) or late (data setup/clock hold) times. If both `-late` and `-early` options are omitted, then the external delay is considered to apply to both early and late times.

`external_delay`

Specifies the external delay, including data setup time for the external register.

`-lead | -trail`

Specifies that the leading or trailing edge of the clock waveform controls the external register. If both `-lead` and `-trail` options are omitted, the default is `lead`. If the external register is a latch, the controlling edge is the capturing edge.

`-network_insertion_delay_included`

Specifies that network insertion delays are already reflected in the specified input and external delay values.

When used with the `-worsecase` option, previously specified network insertion delays on associated clocks are used for the worse casing decision.

`-source_insertion_delay_included`

Specifies that source insertion delays are already reflected in the specified input and external delay values.

When used with the `-worsecase` option, previously specified source insertion delays on associated clocks are used for the worse casing decision.

`port_or_pin_list`

Specifies the list of output or bidirectional ports or pins. When an external delay is asserted on an internal circuit pin (one that is not an output port), all outgoing signals from the pin are blocked.

`-rise | -fall`

Specifies that the external delay refers to the rising edge or falling edge at the output port. If both `-rise` and `-fall` options are omitted, the external delay applies to both the edges at the output port.

`-sig | -ref`

Determines whether the external delay assertion is being applied on a data path (`-sig`) or a clock network (`-ref`). If neither option is given, the default is `-sig`.

`-worst_case`

Takes the worst-case delay when multiple external delays for the same clock are specified. Without this option, successive external delays on the same clock overwrite previously asserted values.

Worst-case calculation includes clock arrival time as specified by `-arrival` option. See Examples.

Note: The `-bidi_input | -bidi_output` options are obsolete for this command. Without the `-bidi_output` option, the external delay assertion is made on the output part of bidirectional pins in the *`port_list`* by default. An assertion made with the `-bidi_input` option is ignored by `set_external_delay`.

Examples

- The following command sets an external delay of 4.1 ns on the output port `pout`, which is driving the input to a register that is controlled by the leading edge of the clock signal `clk`.

```
set_external_delay -lead -clock clk 4.1 pout
```

- The following command shows that the `set_external_delay` has no effect because `set_data_required_time` was previously specified for the same port:

```
set_data_required_time -clock "*" -lead -late -rise   2.000  {out}
set_external_delay -clock "@" -lead -arrival   0.000 -sig -late -rise   4.000
{out}
```

- The following command shows the behavior of `-worst_case` option when the `-arrival` option is specified and when it is not:

```
set_external_delay -clock CLK -worst_case -arrival 3 5 out
```

- For late analysis, if 5-3 (external delay - arrival) is greater than what was previously given (or stored), then these values will replace the stored values. Otherwise they are ignored for late analysis.
- For early analysis, if 5-3 (external delay - arrival) is less than what was previously given (or stored), then these values will replace the stored values. Otherwise they are ignored for early analysis.

  Without the `-arrival` option, worst-case calculation is more straightforward:

  ```
  set_external_delay -clock CLK -worst_case 5 out
  ```

- For late analysis, if the given external delay is greater than the stored value, then it replaces the stored value.
- For early analysis, if the given external delay is less than the stored value, than it replaces the stored value.

Tip

Without the `-early` or `-late` options, the above worstcasing applies to both early and late analysis. So be careful. Avoid confusion by specifying `-early` or `-late` options whenever you specify the `-worst_case` option. See the next example.

- The following command shows the result of using `-early` and `-late` options with the `-worst_case` option. The initial value is set as follows:

  ```
  set_external_delay 2 -clock clk port1
  ```

  By default, the value 2 applies to both early and late analysis.

  If the subsequent assertion is applied to late analysis:

  ```
  set_external_delay 1 -late -clock clk port1 -worst_case
  ```

  The stored value (2) is retained because the given value (1) is not greater than the stored value.

  However, if the subsequent assertion is applied to early analysis:

  ```
  set_external_delay 1 -early -clock clk port1 -worst_case
  ```

  The stored value (2) is replaced by the given value (1) because the given value is less than the stored value.

Related Information

`set_clock`

`set_data_required_time`

`remove_assertions`

See Setting External Delay at Clock Output Ports in the *Common Timing Engine (CTE) User Guide* for more information.

set_false_path

```
set_false_path [{-from | -from_rise | -from_fall} pin_list]
     [{-through | -through_rise | -through_fall} pin_list]
     [{-to | -to_rise | -to_fall} pin_list]
     [-clock_from clk_signame]
     [-edge_from {leading | trailing}] [-clock_to clk_signame]
     [-edge_to {leading | trailing}] [-early | -late]
     [-bidi_input_from | bidi_output_from]
     [-bidi_input_through | -bidi_output_through]
     [-bidi_input_to | bidi_output_to] [-reset_path] [-hold | -setup]
```

Identifies false paths in a design, and breaks or disables specific instance timing arcs in a design. Path exceptions between clock domains are accepted.

If both the `-from` and `-clock_from` options are specified, the path exception applies to all paths starting from the `-from` pin and all the paths starting from the `-clock_from` signal. If both the `-to` and `-clock_to` options are specified, the path exception applies to all the paths ending at the `-to` pin and all the paths ending at the `-clock_to` signal. As shown in the following example, when the output of the `ff1` flip-flop is connected to the inputs of two other flip-flops (`ff2` & `ff3`) all paths for the `-clock_from clk1` will be set as `false`:

```
set_false_path -from ff1/Q -to ff2/D -clock_from clk1
```

If both the `-through` and `-clock_from` options are specified, the path exception applies to all paths starting from the `-clock_from` signal and going through the `-through` pin (AND operation). If both the `-through` and `-clock_to` options are specified, the path exception applies to all paths going through the `-through` pin and ending at the `-clock_to` signal.

Important

To break combinational loops, use the `set_disable_timing` command (see set_disable_timing on page 1153) instead of the `set_false_path` command.

The `set_false_path` command differs from the `set_disable_timing` command as follows:

- `set_false_path` command selectively disables the arrival time information depending upon the precedence rules.
- `set_false_path` command does not affect constant values.
- `set_disable_timing` command physically snips a timing arc such that neither arrival times nor constant values propagate through it.

The `set_false_path -from fpin` command disables all timing arcs whose source pin is an `fpin`. Similarly, the `set_false_path -to tpin` command disables all timing arcs

whose sink pin is a `tpin`. Using the `set_false_path -from fpin -to tpin` command disables all timing arcs whose source is `fpin` and sink is `tpin`.

Specify a group of pins by enclosing the group with curly braces (`{}`) within one `set_false_path` command.

Tip

To set a false path for any path beginning at `p1` or `p2` and terminating at `t1` or `t2`, group the pins as follows:

```
set_false_path -from {p1 p2} -to {t1 t2}
```

Note: This method of pin grouping reduces the number of exceptions. Having large numbers of exceptions adversely impacts the run time of the `report_timing` command.

Options and Arguments

`-bidi_input_from | -bidi_output_from`
: Specifies the assertion on the input or output part of the `from` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_through | -bidi_output_through`
: Specifies the assertion on the input or output part of the `through` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_to | -bidi_output_to`
: Specifies the assertion on the input or output part of the `to` pin. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-clock_from` *`clk_signame`*
: Specifies the starting point of the false paths. The *`clk_signame`* argument is the name of the ideal (or generated) clock signal (waveform).

  If the `-clock_from` option is omitted then the path exception applies to all the paths ending at the registers clocked by the `-clock_to` signal.

`-clock_to` *`clk_signame`*

Specifies the endpoint of the false paths. The *`clk_signame`* argument is the name of the clock signal (waveform).The path exception applies to all paths starting from the `-clock_from` signal and ending at the registers clocked by `-clock_to` signal.

If the `-clock_to` option is omitted then the path exception applies to all paths starting at the registers clocked by the `-clock_from` signal.

Do not use the `-to` and `clock_to` options together.

`-early | -late`

If neither the `-early` nor the `-late` options are specified then it is assumed to be both early and late (hold and setup for data paths). This only applies to the `-to` side if the `-to` *`pin_list`* is present. It applies to the `-from` side in other case.

`-edge_from {leading | trailing}`

Specifies whether the triggering (launching) edge of the clock is a leading or a trailing edge. It should match the edge used in the timing check for the destination register.
Default: `leading` edge

`-edge_to {leading | trailing}`

Specifies whether the capturing edge at the destination register is a leading or a trailing edge.
Default: `leading` edge

`{-from |-from_rise | -from_fall}` *`pin_list`*`}`

Specifies the pins that are at the start of the false paths. If the `-from` option is used without the `-to` option, all paths originating from the *`pin_list`* are false paths.

Either `-from`, `-to`, or `-through` must be used with the `set_false_path` command to identify the paths. Specifying both `-from` and `-to` identifies a path with specific starting and ending points.

The *`pin_list`* can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells

(instances) can also be specified as the object for the `-from*` options.

Use only one `-from` (or `-from_rise` or `-from_fall`) option per command.

By default, `-from` applies the assertion to both the rising and the falling edges.

Using `-from_rise` applies the assertion at the rising edge of the signal on the from pins.

Using `-from_fall` applies the assertion at the falling edge of the signal on the from pins.

`-hold | -setup`

Specifies the early (hold) or late (setup) mode with PT syntax.

`-reset_path`

Changes the constraint on the specified path to the new value. See Examples.

`{-through | -through_rise | -through_fall}` *`pin_list`*

Specifies the pins that the path goes through. When used without the `-from` or `-to` options, all paths that go through the `-through` pins are false paths.

When used with both the `-from` and `-to` options, any path starting at any pin on the `-from` list *and* going through any point on the `-through` pin list *and* going to any point on the `-to` list is affected by the assertion.

The *`pin_list`* variable can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries.There can be only one `-from` and one `-to`, option but many `-through` options in the same command.

By default, the `-through` option applies the assertion to both the rising and the falling edges. The `-through` *`pin_list`* can be a list of pin names, ID, or a collection of pins.

Using the `-through_rise` option applies the assertion at the rising edge of the signal on the through pins.

Using the `-through_fall` option applies the assertion at the falling edge of the signal on the through pins.

`{-to | -to_rise | -to_fall}` *`pin_list`*

Specifies the pins that are at the end of the false paths. If the `-to` option is used without the `-from` option, all paths ending in the `-to` pin list are false paths.

The *`pin_list`* variable can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified as the object for the `-to*` options.

Use only one `-to` (or `-to_rise` or `-to_fall`) option per command.

By default, the `-to` option applies the assertion to both the rising and the falling edges.

Using the `-to_rise` option applies the assertion at the rising edge of the signal on the to pins.

Using the `-to_fall` option applies the assertion at the falling edge of the signal on the to pins.

Note: Only internally generated scripts such as the `write_assertions` command uses `-rise` and `-fall` as options. Use [`-from_rise` | `-from_fall`] and [`-to_rise` | `-to_fall`] options.

Examples

- The following command sets all paths from port `Q` of instance `I1` to false paths:

  ```
  set_false_path -from I1/Q
  ```

- The following command specifies that all paths ending in port `A` of instance `I2` are considered false paths:

  ```
  set_false_path -to I2/A 1
  ```

- The following command makes the path from port `Q` of instance `I1` to port `A` of instance `I2` into a false path:

```
set_false_path -from I1/Q -to I2/A 2
```

- The following command resets all path exceptions on the specified path, then applies the new assertion:

```
set_false_path -reset_path -from A =
reset_path -from A
set_false_path -from A
```

Using the -reset_path option changes the constraint on the specified path to the new value. However, if there is only one path and both paths are equivalent, using this option resets a path only when there is an exact match in the arguments. For example, the following command specifies the delay as U14/A 5:

```
set_fale_path -through U14/A 5
```

The following shows the report using this command:

```
 +---------------------------+
 | Through | Early |  Late   |
 |---------+-------+---------|
 | U14/A   |       | delay 5 |
 +---------------------------+
```

If you then change the constraint to a new value using the -reset_path option on the same path, but use a different argument as follows:

```
 set_false_path -through U12/Z 7 -reset_path
```

The report using the report_path_exceptions command or the report_timing command shows the following:

```
 +-----------------------------------+
 | Through | Early |      Late       |
 |---------+-------+-----------------|
 | U12/Z   |       | delay 7 ignored |
 | U14/A   |       | delay 5         |
 +-----------------------------------+
```

Using the -reset_path option resets a path only when there is an exact match in the arguments, not when both the constraints represent the same path.

Related Information

reset_path_exception

set_constant_for_timing

set_cycle_addition

set_disable_cell_timing

set_disable_timing

`set_path_delay_constraint`

set_fanout_load_limit

`set_fanout_load_limit` *`float`* [`-module` *`module_list`*] [`-port` *`port_list`*]

Specifies the fanout load limit (maximum value) on the output ports and the input ports of a top level module. Fanout load limits are only used to enforce the design rule checks, they do not affect timing analysis. Setting port capacitance limit affects timing analysis.

The design rule requirement of a maximum fanout load value is set using the `set_global fanout_load_limit` global. For the specified ports, the `set_fanout_load_limit` global overrides the global default fanout load limit.

Options and Arguments

`float`
: Specifies the fanout load limit on the ports.

`-module` *`module_list`*
: Specifies the list of modules to which the limit applies. All instances of the specified modules will inherit the limit value.

`-port` *`port_list`*
: Specifies the list of top level ports for which the fanout load limit applies.

Examples

```
set_fanout_load_limit 4 [find -port -output data*]
set_fanout_load_limit 3 [find -port -input reset]
```

Related Information

`get_fanout`

`set_current_module`

`set_global max_fanout_load_limit`

set_flow_compatible_mode

```
set_flow_compatible_mode [ on | off ]
```

Controls the interpretation of constraints during timing analysis. Setting this command to on makes the interpretation of constraints compatible with Synopsys PrimeTime. The read_dc_script command automatically sets the set_flow_compatible_mode command to on. The write_assertions and write_adb commands writes the set_flow_compatible_mode command as part of the assertions.

Default: off

Important

Constraints must be interpreted in one style from start to finish. Do not change the interpretation style in the middle of a run. Decisions that are made when using the read_dc_script command are not reversible by simply changing the set_flow_compatible_mode command. One example is multiple constraints (such as the set_input_delay command) on the same ports. In PT style, the latest constraints override any previous ones and overwritten constraints are thrown away. In CTE style, the worst case is taken among all the constraints. Once the original constraints are gone, you can no longer do the worst casing in CTE style. Therefore, it's best to describe constraints in one style.

Options and Arguments

off
: Interprets constraints in the CTE style.

on
: Interprets constraints in the Synopsys PrimeTime style.

PrimeTime Interpretation of Constraints

- False paths and multicycle paths originating from or terminating at invalid path start and end points are ignored.
- Multicycle paths on late paths will imply a hold constraint.
- The last assertion overwrites previous ones when multiple assertions of the same type exist on the same start and end points.

Valid Start Points for False Paths and Multicycle Paths

- Input port
- Input part of bidirectional port
- Clock pin of sequential cell
- Pin associated with the `set_input_delay` command
- Pin associated with the `set_path_delay -from` command

Valid End Points for False Paths and Multicycle Paths

- Output port
- Output part of bidirectional port
- Data pin of sequential cell
- Pin associated with `set_external_delay`
- Pin associated with `set_path_delay -to`

Examples

Use the `set_flow_compatible_mode` command before reading in the constraints and do not change it in the middle of a run.

- The following command loads libraries:

```
read_tlf lib.tlf
```

- The following command reads a netlist:

```
read_verilog design.v
do_build_generic
```

- The following command interprets constraints in Synopsys PrimeTime style:

```
set_flow_compatible_mode on
```

- The following command reads constraints:

```
source design.asrt
```

- The following command verifies timing:

```
report_timing
```

- The following flow is not supported:

```
read_tlf lib.tlf
```

```
read_verilog design.v
do_build_generic
```

- The following command interprets constraints in PrimeTime style:

```
set_flow_compatible_mode on
source pt.asrt
```

- The following command interprets constraints in CTE style:

```
set_flow_compatible_mode off
source bgpks.asrt
report_timing
```

Related Information

`get_flow_compatible_mode`

set_functional_mode

`set_functional_mode [-group` *`group_name`*`] [-mode` *`mode_name`*`]` *`instance_list`*

Sets the DCL or the STAMP functional mode by group and name on hierarchical instances. Determine what modes are available for an instance using the `report_functional_mode` command.

Once the *`mode_name`* variable is set, all other modes in the *`group_name`* variable become inactive and stay inactive unless they are explicitly set later.

The following form of the command makes all modes in all mode groups in the *`instance_list`* variable active.

`set_functional_mode` *`instance_list`*

While this form of the command makes all modes in the *`group_mode`* variable in the *`instance_list`* variable active.

`set_functional_mode -group` *`group_mode instance_list`*

Options and Arguments

`-group` *`group_name`*
: Specifies the string value of the functional mode group name.
Default: All groups

`instance_list`
: Specifies the list of instances for which the functional mode needs to be set.

`-mode` *`mode_name`*
: Specifies the string value of the functional mode name.
Default: All modes

Example

```
set_functional_mode -group rw -mode read {A1/B3/C2 A1/B2/C2}
```

Related Information

`report_functional_mode`

`reset functional mode`

set_generated_clock

```
set_generated_clock [-name clock_signame] [-from source_pin]
     ([-multiply_by integer] | [-divide_by integer]) [-edges edge_list]
     [-edge_shift edge_shift_list] [-duty_cycle percent] [-neg] target_pin_list
     [-master_clock] [-bidi_input_from | -bidi_output_from bidi port]
     [-bidi_input | -bidi_output bidi port] [-add] [-invert]
```

Creates a new clock signal from the clock waveform of a given pin in the design, or an existing virtual (ideal) clock signal, and binds it with the pins in the *target_pin_list* argument. Whenever the source clock changes, the derived clock(s) change automatically.

Tip

If you need to change the default for the `generated_clocks_scale_edge` global, set it at the beginning of the session. Otherwise existing clocks are not updated.

Generate the new clock waveform by one of three ways:

- Multiplying the frequency of the source clock
- Dividing the frequency of the source clock
- Selecting the edges of the source clock to be mapped on to the edges of the new clock

For examples and information about assertions on generated clocks, Specifying Generated Clocks in the *Common Timing Engine (CTE) User Guide*.

Note: A `set_generated_clock` assertion on a pin overrides any existing `set_generated_clock` assertion if both the assertions have the same from pin or if the from pins are not specified (no `-from` option). For example, in the first and second command pairs below, the second assertion overwrites the first, while in the third and fourth pairs, both assertions are applied:

```
set_generated_clock -name GCK1 -from CK -divide_by 2 U_FD1/Q
set_generated_clock -name GCK2 -from CK -multiply_by 2 U_FD1/Q


set_generated_clock -name GCK1 -divide_by 2 U_FD1/Q
set_generated_clock -name GCK2 -multiply_by 2 U_FD1/Q


set_generated_clock -name GCK1 -from CK -divide_by 2 U_FD1/Q
set_generated_clock -name GCK2 -multiply_by 2 U_FD1/Q


set_generated_clock -name GCK1 -divide_by 2 U_FD1/Q
set_generated_clock -name GCK2 -from CK -multiply_by 2 U_FD1/Q
```

Options and Arguments

`-add`

Models multiple generated clocks on the same source when multiple clocks must fan into the source pin. Ideally, one generated clock must be specified for each clock that fans into the master pin. Specify this option with the `-name` and `-master_clock` options.

By default, CTE creates one generated clock at the pin for each clock present on the source pin. However, this option lets you specify a different clock name for each generated clock when used with the `-master_clock` option. Subsequently, you can use this clock name for setting other constraints such as the `set_false_path` command and the `set_input_delay` command.

`-bidi_input | -bidi_output` *`bidi port`*

Places the assertion on the input or the output bidirectional port of the target pin.
Default: `-bidi_input`

`-bidi_input_from | -bidi_output_from` *`bidi port`*

Places the assertion on the input or the output bidirectional port of the source pin.
Default: `-bidi_output_from`

`-divide_by` *`integer`*

Determines the frequency of the new clock by dividing the frequency of the source clock by *`dfactor`*. For example, if *`dfactor`*=5, the new frequency is 1/5 of the source frequency, and the new clock period is 5 times the source clock period.

`-duty_cycle` *`percent`*

Sets the duty cycle (high pulse width or clock period).

Use the `-duty_cycle` option only with the `-multiply_by` option.

The number (*`percent`*) is between 0 and 100. For example, if *`percent`*=50, the high pulse width of the new clock waveform is the same as its low pulse width.

`-edges` *`edge_list`*

Selects a list of edges from the source clock that form the edges of the derived clock. Currently, three edge numbers are allowed. See the frequency division example below.

`-edge_shift` *`edge_shift_list`*

Specifies the amount of shift for each edge in the *`edge_list`* *`option`*. See Example on page 1192.

`-from` *`source_pin`*

Specifies the name of the source pin from which the new clock is to be derived. If no from pin is specified, the clock is derived from the clock arriving on the target pin.

`-invert`

Inverts the source clock. Use with the `-from` option and the `-multiply_by` or `-divide_by` options. Since virtual clocks do not have polarity, the `-invert` option has no impact when used with the `-clock_from` option.

`-master_clock`

Generates a target clock. If you specify the `-master_clock` option and multiple signals arrive at the target pin, the clock specified by this option is used to generate the target clock. If you do not specify this option, multiple target clocks are generated.

Specifying the -name option lets the `set_input_delay` command and the `set_external_delay` command use a `generated_clock` as a reference clock. For example:

```
set_generated_clock -from IV/A -divide_by 1 -name
gclk IV/Y

set_input_delay -clock gclk 3.0 in1
set_external_delay -clock gclk 2.0 out1
```

Use the name option to pick the source of the generated clock. For example, consider a 2 input mux input with clk1 on A and clk2 on B. Let the mux drive an inverter with a generated clock at the output. Selecting the `-master` option creates a generated clock based on the selected master clock. If the `-master` option is not specified, then two clocks are generated at the output of the inverter. For example, `-master clk1 -divide_by 2` creates

a `generated_clock` on the inverter output based on clk1. If a master clock is not specified, then two clocks are generated at the output: one is derived from clk1 and the other from clk2.

`-multiply_by` *`integer`*

Determines the frequency of the new clock by multiplying the frequency of the source clock by *`mfactor`*. For example, if *`mfactor`*=4, the new frequency is 4 times the source frequency, and the new clock period is 1/4 of the source clock period.

`-name` *`clock_signame`*

Specifies the name for the generated clock.
Default: The system creates a name for you.


Tip

Always specify a name for the generated clock. Some commands issue an error if you use the system-generated name. See" Creating a Generated Clock" of `set_generated_clock` in the *Common Timing Engine (CTE) User Guide*.

`-neg`

Inverts the source clock. Use with the `-from` option and the `-multiply_by` or `-divide_by` options. Since virtual clocks do not have polarity, the `-neg` option has no impact when used with the `-clock_from` option.

`target_pin_list`

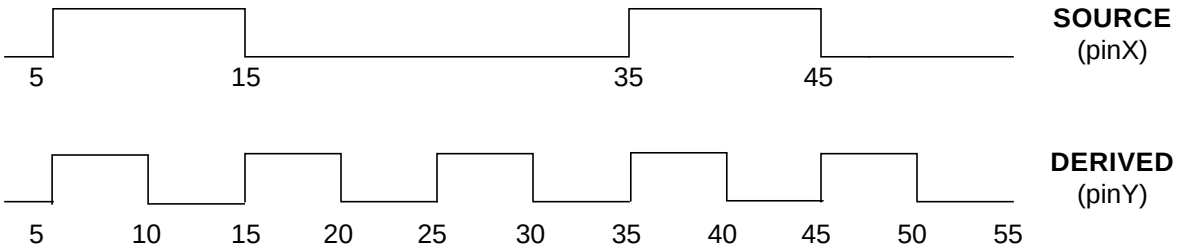
Specifies the list of pins where the generated clock assertion is applied. The clock signal propagating downstream from each of these pins becomes associated with the generated clock.

Examples

- The following command generates the derived clock waveform, as shown in Figure 7-13:

```
set_generated_clock -name mult_3 -from pinX -multiply_by 3 -duty_cycle 50 pinY
```

Figure 7-13 Derived Clock (Frequency Multiplication)

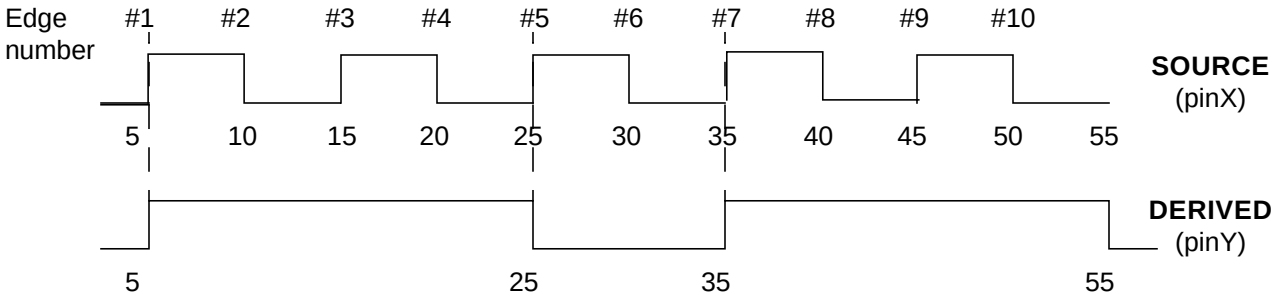


This derives a clock with a clock period =1/3 of the source period, and duty cycle=50 percent as shown in Figure 7-15 on page 1189.

- The following command shows one way to create a divide-by-3 clock by using edge numbers of the source waveform to specify the new waveform, as shown in Figure 7-14,:

```
set_generated_clock -name genclk -from pinX -edges {1 5 7} pinY
```

Figure 7-14 Derived Clock (Frequency Division)

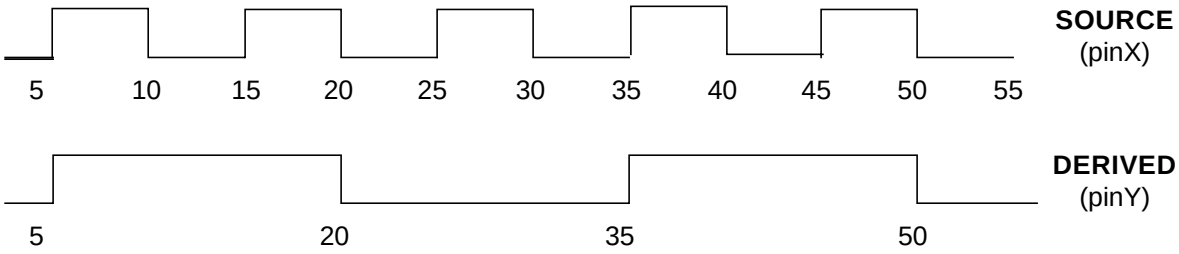


- The following command uses the `-divide_by` option to get the same clock period; the difference is in the duty cycle:

```
set_generated_clock -name div_3 -from pinX -divide_by 3 pinY
```

With the `-edges` option, the above example creates a clock with a duty cycle of 67 percent (20/30). But `-divide_by` 3 would preserve the 50 percent duty cycle of the source clock as shown in Figure 7-15.

Figure 7-15 Derived Clock (Frequency Division, 50 Percent Duty-cycle)



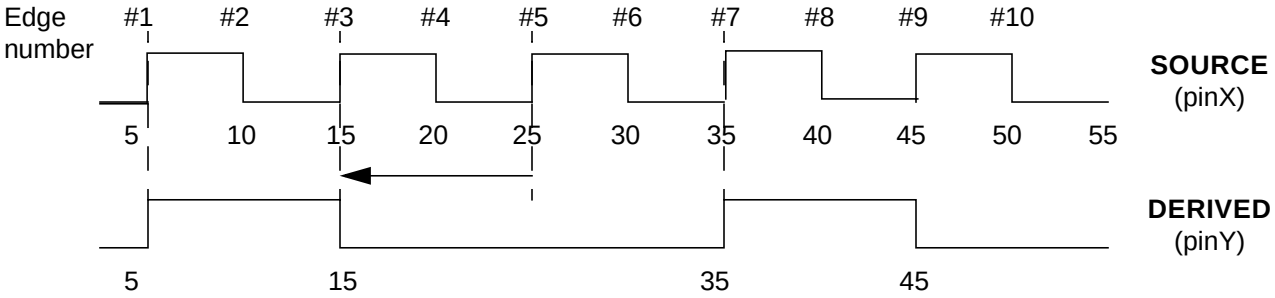
Tip

Use the `-duty_cycle` option only with the `-multiply_by` option. To set the duty cycle to any other number, use the `-edge_shift` option to exactly determine where the fall edge should be as shown in the following example.

- The following command uses edge shifts to create the 33 percent (10/30) duty-cycle waveform, as shown in Figure 7-16:

```
set_generated_clock -name foo -from pinX -edges {1 5 7} -edge_shift {0 -10 0}
pinY
```

Figure 7-16 Derived Clock (Frequency Division, 33% Duty-cycle)



- Using the following commands, when there is more than 1 source clock, the tool creates the generated clock based on the source clock specified with the `-master` option:

```
set_clock c1 -p 10
set_clock c2 -p 10
set_clock_root -clock c1 ck
set_clock_root -clock c2 ck
set_generated_clock -name newclk -from ck -divide_by 2 -pos -master c1 reg/Q
```

Using the `report_clocks` command, the output waveform of the `newclk` clock is a divide-by-2 of the `c1` clock, and is reported as shown in Example 7-4.

Example 7-4 report_clocks after using the -master option

```
+-------------------------------------+
|         Clock Descriptions          |
|-------------------------------------|
|  Clock |  Period |  Lead  |  Trail  |
|  Name  |         |        |         |
|--------+---------+--------+---------|
|  c1    | 10.0000 | 0.0000 |  5.0000 |
|  c2    | 20.0000 | 0.0000 | 10.0000 |
| newclk | 20.0000 | 0.0000 | 10.0000 |
+-------------------------------------+
```

Related Information

get_clock

report_clocks

reset_generated_clock

set_clock

set_clock_root

set_ideal_net

```
set_ideal_net [-delay net_delay] [-load net_load] [-slew_degradation sink_slew]
     net_name_or_id
```

Specifies the delay and load that CTE uses for the net.

An ideal net is a net with no delay and no load. The `set_ideal_net` command expands that idea to let you specify the delay and load that CTE uses for the net. That is, the net load and delay seen by all pins on the net.

There are times in the physical implementation that the routing cannot be accurately modeled. An example of the type of routed net that cannot be modeled accurately is a spine with uniform length connections from the spine to all leaf pins. For high fanout nets (most commonly along a clock spine) the delays can be 300 percent or more off if you do not specify them as ideal nets. Use the `set_ideal_net` command for both preplacement and post placement.

CTE does not perform wire-load model (WLM) or Steiner lookup on ideal nets. The ideal net specification wins over parasitics. Uniquify copies net level assertions.

The `set_ideal_net` command has no effect on ideal clock networks. For ideal clock networks, the delays of gates and interconnects are idealized (delays = 0, slewout = slewin). On data networks and on clock networks in propagated mode, use this command to model the desired behavior.

Options and Arguments

`-delay` *`net_delay`*

Specifies the delay associated with the net. Every source-sink pin pair has the same delay. The *`net_delay`* variable is a floating number.
Default: `0.0`

`-load` *`net_load`*

Specifies the total capacitance that each driver of the net sees, including all pin capacitances. The *`net_load`* variable is a floating number.
Default: `0.0`

`net_name_or_id`

Specifies one or more net names (or ids) that are to be considered ideal nets.

`-slew_degradation` *`sink_slew`*

Specifies the slew degradation at every sink pin. The specified slew value is added to the slew arriving at the source pin to yield the slew at the sink pin. If this option is not used, the slew at the sink pin is equal to the slew at the source pin.The *`sink_slew`* variable is a floating point number.
Default: `0.0`

Example

The following command specifies the delay associated with the net:

```
set_ideal_net -delay 0.2 {clk1 clk2}
```

Related Information

`report_net`

`reset_ideal_net`

See Specifying Ideal Nets in the *Common Timing Engine (CTE) User Guide*.

set_input_delay

```
set_input_delay [-early | -late] [-rise |-fall] [-clock clock_name]
     [-lead |-trail] [-source_insertion_delay_included]
     [-network_insertion_delay_included] [-worst_case] float port_or_pin_list
```

Sets input path delay values on input ports or internal input pins. This input path delay models the delay from an external register to an input port of the module. The `set_input_delay` command differs from `dc_shell` in that the `-min` and `-max` options are replaced by the `-early` and `-late` options. The justification is that there could be multiple paths between the fictitious register and the input port. The `-early` and `-late` options properly capture the difference in delays for these paths.

Note: The `set_input_delay` command replaces the `set_data_arrival_time` command.

Options and Arguments

`-clock` *`clock_name`*
: Specifies the name of the ideal clock. A generated clock cannot be used with this option. Default is the asynchronous clock.

`-clock` *`clock_name`*
: Specifies the name of the ideal clock. A generated clock cannot be used with this option.
Default: Asynchronous (@) clock.

`-early | -late`
: Specifies that the constraint refers to the early (hold) or late (setup) times of the data signal. If both `-late` and `-early` options are omitted, then the input delay is considered to apply to both early and late times.

`float`
: Specifies the delay value.

`-lead | -trail`
: Specifies that the input delay is with respect to the leading or the trailing edge of the clock. If neither option is provided, the default is `-lead`.

`-network_insertion_delay_included`
: Specifies that network insertion delays are already reflected in the specified input and external delay values.

When used with the -worst_case option, previously specified network insertion delays on associated clocks are used for the worse casing decision.

port_or_pin_list

Specifies a single input port or pin or multiple ports or pins to which the input delay applies. To specify multiple ports or pins, enclose the list with curly braces ({}) and separate the port or pin information with white space.

When an input delay assertion is present on an internal circuit pin (one that is not an input port), all incoming signals to the pin are blocked.

-rise | -fall

Specifies that the input delay refers to the rising edge or falling edge at the input port/pin. If both -rise and -fall options are omitted, the input delay applies to both the edges.

-source_insertion_delay_included

Specifies that source insertion delays are already reflected in the specified input and external delay values.

When used with the -worst_case option, previously specified source insertion delays on associated clocks are used for the worse casing decision.

-worst_case

Takes the worst-case delay when multiple input delays for the same clock are specified. Without this option, successive input delays on the same clock overwrite previously asserted values.

The -bidi_input | -bidi_output options are obsolete for this command. Without the -bidi_input option, the input delay assertion is made on the input part of bidirectional pins in the *port_list* by default. An assertion made with the -bidi_output option is ignored by set_input_delay.

Examples

- The following command specifies the name of the ideal clock, specifies that the constraint refers to the early (hold) time of the data signal, and specifies that the input delay refers to the rising edge at the input port/pin:

```
set_input_delay -clock CLK -early -rise -0.05 [ find -inputs ]
```

- The following command specifies the name of the ideal clock, specifies that the constraint refers to the late (setup) time of the data signal, and specifies that the input delay refers to the falling edge at the input port/pin:

```
set_input_delay -clock CLK -late  -rise 0.05 [ find -inputs ]
```

- The following command specifies the name of the ideal clock, specifies that the constraint refers to the early (hold) time of the data signal, and specifies that the input delay refers to the falling edge at the input port/pin:

```
set_input_delay -clock CLK -early -fall 0.15 [ find -inputs ]
```

- The following command specifies the name of the ideal clock, specifies that the constraint refers to the late (setup) time of the data signal, and specifies that the input delay refers to the falling edge at the input port/pin:

```
set_input_delay -clock CLK -late  -fall 0.25 [ find -inputs ]
```

- The following commands reference an `input_delay` to a generated clock:

```
set_clock sysclk -waveform "0 0.5" -period 1.0
set_clock_root -clock sysclk -pos clk
set_generated_clock -from clk -divide_by 2 -name genclk clkbuf/Y
set_input_delay -clock sysclk -worst_case 0 in
set_input_delay -clock genclk -worst_case 0 in
```

Related Information

`remove_assertions`

`reset_input_delay`

`set_clock_insertion_delay`

`set_external_delay`

set_max_delay

```
set_max_delay
     Not an ac_shell command, use set_path_delay_constraint instead.
```

See set_path_delay_constraint on page 1230.

set_min_delay

```
set_min_delay
     Not an ac_shell command, use set_path_delay_constraint instead.
```

See set_path_delay_constraint on page 1230.

set_mp_area

`set_mp_area` *`value`*

Specifies the estimated area of the module in square microns. The estimate represents a composite result for the block as a whole.

Options and Arguments

`value`

Specifies the area of the module in square microns, which must be a number.

Example

The following command sets the area of the Module Prototype to `10.7`:

```
set_mp_area 10.7
```

set_mp_global_parameter

```
set_mp_global_parameter
     [-param {clk_to_output | setup | hold | recovery | removal |no_change_setup
     |no_change_hold | period | min_pulse_width_low | min_pulse_width_high |
     skew}] [-lib_cell cell_name] [-pin pin_name] [-clock pin_name]
     [value parameter_value]
```

Sets base values for sequential delays and constraint arcs (all other parameter types). These values are used by the create_mp_delay_arc command for edge based arcs (`clk_to_output param` type) or by the create_mp_constraint_arc command (all other parameter types). The base value is either found by identifying an arc in the reference cell or by a given, explicit value. Use the `create_mp_model` command to reset the global parameters.

Options and Arguments

`-clock` *`pin_name`*

Specifies the reference pin for a sequential delay or a constraint arc.

`-lib_cell` *`cell_name`*

Specifies a cell in the reference library used for sequential information.

`-param`

Specifies which global parameter will be set. If it is the `clk_to_output` option, the global parameter will be added to all edge delay arcs. Other `param` types select a specific constraint arc.

`-pin pin_name`

Specifies the non-reference pin for a sequential delay or a constraint arc.

The *`pin_name`* is the name of a single pin the a reference cell. Bus names or bus ranges are not allowed.The current name space is used for special character escaping. See Pin Names in the “Using the Module Prototyper (MP)” section in the *Common Timing Engine* (*CTE) User Guide*.

`-value parameter_value`

Specifies a constant sequential value, which can be a single number or an SDF style triplet.

Examples

- The following command sets a delay time of 1 that will be added to all edge delay arcs:

  ```
  set_mp_global_parameter -param clk_to_output -value 1
  ```

- The following command sets the base setup check time to 4:

  ```
  set_mp_global_parameter -param setup -value 4
  ```

- The following command sets the base hold time to be the same as for the hold time between pins D and cp in the jk01d2 reference cell:

  ```
  set_mp_global_parameter -param hold -lib_cell jk01d2 -pin D -clock cp
  ```

Related Information

create_mp_path_type

create_mp_constraint_arc

set_mp_max_fanout_limit

```
set_mp_max_fanout_limit
     [-value port_limit_value] [-port port_list]
```

Sets the maximum fanout limit for ports.

Options and Arguments

`-port port_list`

Specifies the name of the affected ports. They must be output or inout ports. If the *port_list* argument is not specified, all output or inout ports will be affected.

The `port_name_list` can be either a single name or a list of names in curly braces. Each name can be a simple name, such as A, a bus element name, such as Data[3], a bus name, such as Data, or a bus range, such as Data[3:6]. The current name space is used for special character escaping. See Port Names in the “Using the Module Prototyper (MP)” section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-value port_limit_value`

Specifies the maximum fanout value, which can be either a single number or an SDF triplet.

Examples

- The following command sets the maximum fanout limit on port `Z` to `8`:

  ```
  set_mp_max_fanout_limit -value 8 Z
  ```

- The following command sets the maximum fanout limit on ports `Q` and `QN` to `8`:

  ```
  set_mp_max_fanout_limit -value 8 {Q QN}
  ```

Related Information

`set_mp_port_drive`

`set_mp_technology`

set_mp_min_fanout_limit

```
set_mp_min_fanout_limit
     [-value port_limit_value] [-port port_list]
```

Defines the minimum fanout for ports.

Options and Arguments

`-port` *`port_name_list`*

Specifies the name of the affected ports. They must be output or inout ports. If this option is not specified, all output and inout ports will be affected.

The `port_name_list` can be either a single name or a list of names in curly braces. Each name may be a simple name, such as A, a bus element name, such as Data[3], a bus name, such as Data, or a bus range, such as Data[3:6]. The current name space is used for special character escaping. See Port Names in the “Using the Module Prototyper (MP)” section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-value` *`port_limit_value`*

Specifies the minimum fanout value, which can be either a single number or an SDF triplet.

Examples

- The following command sets the minimum fanout limit on port `Z` to 8:

  ```
  set_mp_min_fanout_limit -value 8 Z
  ```

- The following command sets the minimum fanout limit on ports `Q` and `QN` to 8:

  ```
  set_mp_min_fanout_limit -value 8 {Q QN}
  ```

Related Information

`set_mp_port_drive`

`set_mp_technology`

set_mp_port_drive

```
set_mp_port_drive
     [-type drive_type] [-value port_limit_value] port_name_list
```

Specifies the driver strength (resistance) for each of the specified output or inout ports. If you do not specify any ports, this command applies to all output or inout ports. All delay arcs to these ports will be affected. If you specify more than one `set_mp_port_drive` command for the same port, the last one specified before using the `create_mp_delay_arc` command creates a given timing arc to the port is used in the model for that arc.

By default, only the delay arcs to this port are affected. If a reference type is used and the `set_mp_technology -inherit_port_limits on` command was used, then any port limits on the reference pin will be copied to the ports.

Options and Arguments

`port_name_list`

Specifies the name of the ports. If missing, then all output and inout ports are affected.

The `port_name_list` can be either a single name or a list of names in curly braces. Each name can be a simple name, such as A, a bus element name, such as Data[3], a bus name, such as Data, or a bus range, such as Data[3:6]. The current name space is used for special character escaping. See Port Names in the “Using the Module Prototyper (MP)” section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-type` *`drive_type`*

If a drive type is specified, then the drive information will come from the path information from a reference cell delay arc.

`-value` *`port_limit_value`*

Specifies an explicit drive resistance for the ports.The drive resistance is not in library resistance units. Instead, the output slew time and the base delay will equal this value times the output load. The value can be either a single number or an SDF style triplet.

For output slew, if the `set_mp_port_drive` command is not set on the output, then the output slew is the same as the input

slew. If the `set_mp_port_drive` command is used with the `-value` option, then the output slew is `value*Cload`. If the `set_mp_port_drive` command is used with a drive type, then the output slew is copied from reference cell data.

Examples

- The following command specifies that all subsequently created delay arcs to port `Z` have timing based on the arc used to create the `drive1` drive type:

  ```
  set_mp_port_drive -type drive1 Z
  ```

- The following command specifies that for all subsequently created delay arcs to ports `Q` and `Qn`, the output slew will be twice the output load and the base delay time is also twice the output load:

  ```
  set_mp_port_drive -value 2 {Q QN}
  ```

Related Information

`create_mp_delay_arc`

`create_mp_path_type`

`set_mp_global_parameter`

`set_mp_technology`

set_mp_port_load

```
set_mp_port_load
     {-type load_type | -value port_cap} port_name_list
```

Specifies the port capacitance for each of the specified input or inout ports. If you do not specify any ports, this command applies to all input or inout ports.

By default, only the port capacitance of this port are affected. If a reference type is used and the set_mp_technology -inherit_port_limits on command was used, then any maximum or minimum transition port limits on the reference pin will be copied to the ports.

Options and Arguments

port_name_list

Specifies the name of the affected ports. They must be output or inout ports. If the *port_name_list* argument is not specified, all output or inout ports will be affected.

The *port_name_list* can be either a single name or a list of names in curly braces. Each name can be a simple name, such as A, a bus element name, such as Data[3], a bus name, such as Data, or a bus range, such as Data[3:6]. The current name space is used for special character escaping. See Port Names in the "Using the Module Prototyper (MP)" section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

-type *load_type*

If a load type is specified, then the port capacitance will come from a reference cell pin.

-value *port_cap*

Specifies an explicit port capacitance, which can be either a single number or an SDF style triplet.

Examples

- The following command sets the load on port A of the Module Prototype to 1.5 times the load of the pin used in defining the load1 load type:

```
set_mp_port_load -type load1 -factor 1 A
```

- The following command sets the load on the Module Prototype's port to B max load = 0.8, typ load = 0.85, max load = 0.9

```
set_mp_port_load -value 0.8::0.9 {A B}
```

Related Information

create_mp_delay_arc

set_mp_max_fanout_limit

set_mp_min_fanout_limit

set_mp_port_max_capacitance

set_mp_port_min_capacitance

set_mp_port_max_transition

set_mp_port_min_transition

set_mp_port_max_capacitance

`set_mp_port_max_capacitance [-value` *`port_limit_value`*`] port_name_list`

Specifies an explicit constraint on the maximum external capacitance that can be driven by each of the output or bidirectional ports. This command overwrites any previous maximum port capacitance.

Options and Arguments

`port_name_list`

Specifies the name of the ports. The ports must be and output or inout type. If this argument is not specified, all output and inout ports will be affected.

The *`port_name_list`* can be either a single name or a list of names in curly braces. Each name can be a simple name, such as A, a bus element name, such as `Data[3]`, a bus name, such as `Data`, or a bus range, such as `Data[3:6]`. The current name space is used for special character escaping. See Port Names in the "Using the Module Prototyper (MP)" section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-value` *`port_limit_value`*

Specifies a constraint on the external capacitance for the ports, which can be either a single number or an SDF style triplet.

Examples

- The following command sets the maximum capacitance for port `Z` to `0.9`:

  ```
  set_mp_max_capacitance -value 0.9 Z
  ```

- The following command sets the maximum capacitance for ports `Q` and `QN`. The minimum value is set to `0.8` and the maximum value is set to `0.9`. No typical value is supplied, which is legal only if the Module Prototype has no `typ` corner.

  ```
  set_mp_max_capacitance -value 0.8::0.9 {Q QN}
  ```

Related Information

`set mp port drive`

`set_mp_technology`

set_mp_port_max_transition

```
set_mp_port_max_transition [-value port_limit_value] port_name_list
```

Specifies an explicit constraint on the maximum transition time measured at each of the input, output, or inout ports.

Options and Arguments

port_name_list

Specifies the name of the affected ports. If this argument is not specified, then this command applies to all ports.

The *port_name_list* argument can be either a single name or a list of names in curly braces. Each name can be a simple name, such as A, a bus element name, such as `Data[3]`, a bus name, such as `Data`, or a bus range, such as `Data[3:6]`. The current name space is used for special character escaping. See Port Names in the "Using the Module Prototyper (MP)" section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-value` *port_limit_value*

Specifies a constraint on the maximum transition time measured at each of the ports, which can be either a single number or an SDF style triplet.

Examples

- The following command sets the maximum transition time for port `Z` to `0.5`:

  ```
  set_mp_port_max_transition -value 0.5 Z
  ```

- The following command sets the maximum transition time for ports `Q` and `QN`. The minimum value is set to `0.8` and the maximum value is set to `0.9`. No typical value is supplied, which is legal only if the MP prototype has no `typ` corner:

  ```
  set_mp_port_max_transition -value 0.8::0.9 {Q QN}
  ```

Related Information

`set_mp_port_drive`

`set_mp_port_load`

set_mp_technology

set_mp_port_min_capacitance

```
set_mp_port_min_capacitance [-value port_limit_value] port_name_list
```

Specifies an explicit constraint on the minimum external capacitance that can be driven by each of the output or bidirectional ports. This command overwrites any previous minimum port capacitance.

Options and Arguments

port_name_list

Specifies the name of the affected ports. They must be output or inout ports. If the *port_name_list* argument is not specified, all input or input ports will be affected.

The *port_name_list* argument can be either a single name or a list of names in curly braces. Each name can be a simple name, such as A, a bus element name, such as `Data[3]`, a bus name, such as `Data`, or a bus range, such as `Data[3:6]`. The current name space is used for special character escaping. See Port Names in the "Using the Module Prototyper (MP)" section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-value` *port_limit_value*

Specifies a constraint on the external capacitance for the ports, which can either be a single number or an SDF style triplet.

Examples

- The following command sets the minimum capacitance for port `Z` to `0.9`:

  ```
  set_mp_min_capacitance -value 0.9 Z
  ```

- The following command sets the minimum capacitance for ports `Q` and `QN`. The minimum value is set to `0.8` and the maximum value is set to `0.9`. No typical value is supplied, which is legal only if the Module Prototype has no `typ` corner:

  ```
  set_mp_min_capacitance -value 0.8::0.9 {Q QN}
  ```

Related Information

`set_mp_port_drive`

set_mp_technology

set_mp_port_min_transition

`set_mp_port_min_transition [-value` *`port_limit_value`*`] port_name_list`

Specifies an explicit constraint on the minimum transition time measured at each of the input, output, or inout ports.

Options and Arguments

`port_name_list`

Specifies the name of the affected ports. If this option is not specified, then this command applies to all ports.

The *`port_name_list`* argument can be either a single name or a list of names in curly braces. Each name may be a simple name, such as A, a bus element name, such as `Data[3]`, a bus name, such as `Data`, or a bus range, such as `Data[3:6]`. The current name space is used for special character escaping. See Port Names in the "Using the Module Prototyper (MP)" section of the *Timing Analysis for BuildGates Synthesis and Cadence Physically Knowledgeable Synthesis (PKS).*

`-value` *`port_limit_value`*

Specifies a constraint on the minimum transition time measured at each of the ports, which can be either a single number or an SDF style triplet.

Examples

- The following command sets the minimum transition time for port `Z` to `0.5`:

  ```
  set_mp_port_max_transition -value 0.5 Z
  ```

- The following command sets the minimum transition time for ports `Q` and `QN`. The minimum value is set to `0.8` and the maximum value is set to `0.9`. No typical value is supplied which is legal only if the Module Prototype has no `typ` corner:

  ```
  set_mp_port_max_transition -value 0.8::0.9 {Q QN}
  ```

Related Information

`set_mp_port_drive`

`set_mp_port_load`

`set_mp_technology`

set_mp_technology

```
set_mp_technology
     [-library library_name][-max_transition trans_value]
     [-min_transition trans_value] [-max_capacitance cap_value]
     [-min_capacitance cap_value][-inherit_port_limits {on | off}]
     [-wire_load_model wlm_name]
```

Sets technology parameters. These values should be set at the beginning of an Module Prototyper (MP) session. Use the `create_mp_model` command to reset the technology parameters.

Options and Arguments

`-inherit_port_limits {`*`on`*`|`*`off`*`}`
: If the `-inherit_port_limits` option is set to on, then the `set_mp_port_drive` and `set_mp_port_load` commands copies port limits from reference pins to the affected ports.
Default: `off`

`-library` *`library_name`*
: Sets the technology library, which is used for all library level quantities (PVT corners, kfactors, defaults, and units). Also, all pin, delay, and constraint information used to create the prototype come from this library. Once set, the reference library should not be changed.

`-max_capacitance` *`cap_value`*
: Specifies the default maximum capacitance, which can be either a single number or an SDF style triplet.

`-min_capacitance` *`cap_value`*
: Specifies the default minimum transition time, which can be either a single number or an SDF style triplet.

`-max_transition` *`trans_value`*
: Specifies the default maximum transition time, which can be either a single number or an SDF style triplet.

`-min_transition` *`trans_value`*
: Specifies the default minimum transition time, which can be either a single number or an SDF style triplet.

`-wireload_model` *`wlm_name`*
: Specifies the wire load model used for calculating the delays for path types.

Examples

- The following command sets the reference library to library `fast`:

  ```
  set_mp_technology -library fast
  ```

- The following command sets the wire load model used in creating path types to `tsmc18_wl10`:

  ```
  set_mp_technology -wire_load_model tsmc18_wl10
  ```

- The following command sets the default maximum capacitance to `2.0`:

  ```
  set_mp_technology max_capacitance 2.0
  ```

Related Information

`create_mp_path_type`

`set_mp_port_drive`

`set_mp_port_load`

`set_mp_max_fanout_limit`

`set_mp_min_fanout_limit`

`set_mp_port_max_capacitance`

`set_mp_port_min_capacitance`

`set_mp_port_max_transition`

`set_mp_port_min_transition`

set_num_external_sinks

`set_num_external_sinks` *`non_neg_integer port_list`*

Specifies top level input and output ports only, and sets the constraint for the number of external sinks that are connected to the ports in the port list. This number is factored into the wire capacitance and the wire resistance estimation for the port nets using wire-load models.

It does not add to the total fanout count for Design Rule Violation (DRV) checks. For more information on design rule violations, see the description for `max_fanout_load_limit` in "Command Arguments for `set_global`."

If you do not specify the `set_num_external_sinks` command, the default fanout is `1` for the port (and any other internal sinks of the wire). If you specify the `set_num_external_sinks` command, then you get that exact number added to the count of internal sinks of the wire.

Options and Arguments

`non_neg_integer`
Specifies the number of external sinks.
Default: 1

`port_list`
Specifies the list of top level ports.

Example

```
set_num_external_sinks 3 out
```

Related Information

`do_derive_context`

`do_time_budget`

`set_fanout_load_limit`

`set_num_external_sources`

`set_port_capacitance`

`set_port_wire_load`

set_num_external_sources

```
set_num_external_sources non_neg_integer port_list
```

Specifies top level input and output ports only, and sets the constraint for the number of external sources that are connected to the ports in the port list. This number is factored into the wire capacitance and the wire resistance estimation done for the port nets using the wire-load models.

It does not add to the total fanin count for Design Rule Violation (DRV) checks. If you do not use the `set_num_external_sources` command, then the fanin is 1 for the port (and any other internal sources on the wire). If you specify the `set_num_external_sources` command, then you get that exact number added to the count of internal sources of the wire.

Options and Arguments

non_neg_integer
: Specifies the number of external sources.
Default: 1

port_list
: Specifies the list of top level ports.

Example

```
set_num_external_sources 2 in
```

Related Information

`do_derive_context`

`do_time_budget`

`set_fanout_load_limit`

`set_num_external_sinks`

`set_port_capacitance`

`set_port_wire_load`

set_operating_conditions

```
set_operating_conditions [-library library_name] [-pvt {min | typ | max}]
     operating_condition_name
```

Specifies the operating condition (process, temperature, voltage, and tree-type) for the design from a given library. The named operating condition is used by the mapping and optimization commands to appropriately select cells.

Each technology library contains one or more operating conditions. Each condition is identified by name, which specifies a set of process, temperature, voltage, and tree-type conditions as the operating condition. This information is used to calculate accurate cell delays from the nominal cell delays and the k-factors (also called derating factors) from either linear or nonlinear models.

Get a list of all operating conditions in a technology library along with their PVT and OC type values using the `report_library` command.

Note: If you specify an operating condition, for example: `set_operating_conditions -library` *`library_name operating_condition_name`* `-pvt min`, but the `pvt_early_path` global is set to the `max` corner, the `set_operating_conditions` assertion will be ignored. To apply the assertion, set the `pvt_early_path` global to `min`. This also applies to the `max` operating condition. Set the `pvt_late_path` global to the `max` corner before setting the `set_operating_conditions -pvt max` command. Using the `report_timing -early` command or the `report_timing -late` command will list the specified operating condition in the report.

Options and Arguments

`-library` *`library_name`*
: Specifies the technology library containing the named operating condition.

`operating_condition_name`
: Specifies the name of the operating condition in the library.

`-pvt {min | typ | max}`
: Sets the operating condition for a particular PVT corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces (`{}`) and separate the values by spaces.
Default: The specified operating condition applies to all three PVT corners.

Examples

- The following command sets the operating conditions for the min PVT corner to `fast_0_3.60`:

  ```
  set_operating_conditions -pvt min fast_0_3.60
  ```

- The following timing report shows the operating conditions listed in the library:

```
+-------------------------------------------------------------+
| Operating | process | temperature | voltage |       OC type |
| Condition |         |             |         |               |
|-----------+---------+-------------+---------+---------------|
|       NOM |  1.0000 |     25.0000 |  5.0000 | balanced_tree |
|     WCCOM |  1.2800 |     70.0000 |  4.7500 | balanced_tree |
|     WCIND |  1.2800 |     85.0000 |  4.7500 | balanced_tree |
|     WCMIL |  1.2800 |    125.0000 |  4.5000 | balanced_tree |
|     BCCOM |  0.7000 |      0.0000 |  5.2500 | balanced_tree |
|     BCIND |  0.7000 |    -40.0000 |  5.2500 | balanced_tree |
|     BCMIL |  0.7000 |    -55.0000 |  5.5000 | balanced_tree |
|       TST |  1.2800 |     25.0000 |  5.0000 | balanced_tree |
+-------------------------------------------------------------+
```

  You can set different operating conditions for different PVT corners. If no operating conditions were set, the `report_timing` command will list the default operating condition besides the `Operating Condition` field, shown in the example above. The report also lists the Operating Condition (`OC`) `Type`, and the `process`, `temperature`, and `voltage` values for the `operating_condition` used.

  For more information, see" Explaining How the Operating Condition Type Effects Delay Calculation" in the *Common Timing Engine (CTE) User Guide*.

  To set a `min` operating condition, make sure the global `pvt_early_path` is set to `min`. If you set the corner using the `pvt_early_path` and `pvt_late_path` globals, the operating condition set to that corner will be picked. If you specify the `-pvt` option with the `set_operating_conditions` command, the operating condition applies to all three PVT corners.

  The following examples display only the timing report header to show what operating condition is used by the timer to calculate delays and slews. The `pvt_early_path` and `pvt_late_path` globals are pointing to the `max` corner, and the default operating condition is `NOM`, which is for both the `min` and `max` corners.

  - `report_timing -late`

    ```
    ..
    | Timing              | LATE                |  <==
    ..
    ```

```
    | Operating Condition | NOM                 |  <==
    | PVT Mode            | max                 |  <==
    | Tree Type           | balanced            |
    | Process             | 1.0000              |
    | Voltage             | 5.0000              |
    | Temperature         | 25.0000             |
..
```

- `report_timing -early`

```
    ..
    | Timing              | EARLY               |  <==
    ..
    | Operating Condition | NOM                 |  <==
    | PVT Mode            | max                 |  <==
    | Tree Type           | balanced            |
    | Process             | 1.0000              |
    | Voltage             | 5.0000              |
    | Temperature         | 25.0000             |
    ..
```

- Setting the `pvt_early_path` global to `min` results in the following timing report:

```
    ..
    | Timing              | EARLY               |  <==
    ..
    | Operating Condition | NOM                 |  <==
    | PVT Mode            | min                 |  <==
    | Tree Type           | balanced            |
    | Process             | 1.0000              |
    | Voltage             | 5.0000              |
    | Temperature         | 25.0000             |
    ..
```

- The following commands set the operating conditions to `max`, `typ`, and `min`:
  - `set_operating_conditions -pvt max WCCOM`
  - `set_operating_conditions -pvt min BCCOM`
  - `set_operating_conditions -pvt typ TST`
- The following command shows the timing report using the `-late` option:

```
    report_timing -late
    ..
    | Timing              | LATE                |  <==
```

```
..
| Operating Condition | WCCOM                |  <==
| PVT Mode            | max                  |  <==
| Tree Type           | balanced             |
| Process             | 1.2800               |  <==
| Voltage             | 4.7500               |  <==
| Temperature         | 70.0000              |  <==
..
```

- The following command show the timing report using the `-early` option:

```
report_timing -early
..
| Timing              | EARLY                |  <==
..
| Operating Condition | BCCOM                |  <==
| PVT Mode            | min                  |  <==
| Tree Type           | balanced             |
| Process             | 0.7000               |  <==
| Voltage             | 5.2500               |  <==
| Temperature         | 0.0000               |  <==
..
```

- The following global sets the PVT corner to `typ`. The `typ` value is not used since neither the `pvt_early_path`, nor the `pvt_late_path` globals are pointing to `typ`:

```
set_global pvt_early_path typ
report_timing -early
..
| Timing              | EARLY                |  <==
..
| Operating Condition | TST                  |  <==
| PVT Mode            | typ                  |  <==
| Tree Type           | balanced             |
| Process             | 1.2800               |  <==
| Voltage             | 5.0000               |  <==
| Temperature         | 25.000               |  <==
```

- The following example explains how setting the operating condition effects delay calculation.

```
in --->INV1-->INV2--- out
```

This example has two inverters in series. `INV1` is from library1, while `INV2` is from library2. Library1 and library2, as shown below, have a `scaled_cell` for `INV1` and

INV2 for all three operating conditions.This means in library1 for INV1 a scaled_cell is defined for all three operating conditions: MAXIMUM, MINIMUM, TYPICAl. This also applies to INV2 in library2.

```
library1:
+-------------------------------------------------------------+
| Operating |  process | temperature |  voltage |       OC type |
| Condition |          |             |          |               |
|-----------+----------+-------------+----------+---------------|
|       NOM | 1.000000 |   25.000000 | 2.750000 | balanced_tree |
|   MAXIMUM | 0.780000 |  100.000000 | 2.250000 | balanced_tree |
|   MINIMUM | 0.950000 |    0.000000 | 3.750000 | balanced_tree |
|   TYPICAL | 1.500000 |   35.000000 | 3.000000 | balanced_tree |
+-------------------------------------------------------------+
library2:
+-------------------------------------------------------------+
| Operating |  process | temperature |  voltage |       OC type |
| Condition |          |             |          |               |
|-----------+----------+-------------+----------+---------------|
|       NOM | 1.000000 |   25.000000 | 5.000000 | balanced_tree |
|   MAXIMUM | 1.280000 |   70.000000 | 4.750000 | balanced_tree |
|   MINIMUM | 0.700000 |    0.000000 | 5.250000 | balanced_tree |
|   TYPICAL | 1.280000 |   25.000000 | 5.000000 | balanced_tree |
+-------------------------------------------------------------+
```

- The following command sets an operating condition from library2:

  ```
  set_tech_info -library library2 -default_operating_conditions NORM
  ```

  Even if the target_technology is specified as "library1 library2", the default operating condition from library1 is used for timing analysis. The default operating condition from library2 is used only if library2 is the first library in the target_technology list.

  ```
  get_tech_info -default_operating_conditions
  NOM
  set_global target_technology "library2 library1"
  get_tech_info -default_operating_conditions
  NORM
  ```

- The following command sets the operating condition to all three corners because the -pvt option is not used:

  ```
  set_operating_condition -library library2 MAXIMUM
  ```

- The following global sets the target technology:

  ```
  set_global target_technology "library1 library2"
  ```

Since a `scaled_cell` is defined for `INV1` for the `MAXIMUM` operating condition, the values defined for this `scaled_cell` for that operating condition are picked up. This is also true for `INV2`. No derating is done - means the multiplication factor is `1`. The different PVT values between the operating conditions has no effect, so the `scaled_cell` is used. If a `scaled_cell is` not defined for this operating condition, the value picked up from the table defined for the cell is derated to the PVT values of `MAXIMUM` from library2.

The `set_operating_condition` command overrides any previous settings by the `set_tech_info -default_operating_conditions` command. Once you set the `set_operating_condition` command, any further settings using the `set_tech_info -default_operating_conditions` command will have no effect.

- The `report_library` command prints a list of all the operating conditions in a technology library along with their PVT and OC type values, as shown in the following table:

```
+-------------------------------------------------------------------------+
|  PVT |     Operating | process | temperature | voltage |         OC type |
|      |     Condition |         |             |         |                 |
|------+---------------+---------+-------------+---------+-----------------|
|  max | slow_125_3.00 |    1.00 |      125.00 |    3.00 | worst_case_tree |
|  typ | slow_125_3.00 |    1.00 |      125.00 |    3.00 | worst_case_tree |
|  min |   fast_0_3.60 |    0.69 |        0.00 |    3.60 |  best_case_tree |
+-------------------------------------------------------------------------+
```

  Any operating condition that is not set is picked up from the first library in the target technology list.

- The following command reads in another library and sets the `typ` condition to that library:

```
set_operating_conditions -library library_name slow -pvt typ
```

  The `report_library` command prints the following table, assuming the OC type is `balanced_tree`, and the numbers are for that condition from the library:

```
+-------------------------------------------------------------------------+
|  PVT |     Operating | process | temperature | voltage |         OC type |
|      |     Condition |         |             |         |                 |
|------+---------------+---------+-------------+---------+-----------------|
|  max | slow_125_3.00 |    1.00 |      125.00 |    3.00 | worst_case_tree |
|  typ |          slow |    1.00 |      125.00 |    1.08 |   balanced_tree |
|  min |   fast_0_3.60 |    0.69 |        0.00 |    3.60 |  best_case_tree |
+-------------------------------------------------------------------------+
```

  Instead of using the `report_library` command with the `-operating_conditions` option, query a single operating condition using the `get operating conditions` command:

```
get_operating_condition -pvt min fast_0_3.60
get_operating_condition -pvt typ slow
```

Related Information

get_operating_conditions

get_operating_parameter

read_alf

read_tlf

report_library

reset_operating_condition

set_operating_parameter

See the Reading Timing Libraries chapter in the *Common Timing Engine (CTE) User Guide* for information on setting operating conditions for timing libraries.

set_operating_parameter

```
set_operating_parameter {-process float | -voltage float | -temperature float}
     [-pvt {min | typ | max}]
```

Sets the PVT operating parameters individually for the design used in calculating accurate cell delays from the nominal cell delays and the k-factors (also called derating factors) from either linear or non-linear models.

If a DCM has been loaded, the delay and slew equations in the DCM use the operating parameters specified here.

The `set_operating_parameter` command overrides the parameter value from the default operating condition of the technology library. It also overrides the parameter value from any operating condition you previously selected with the `set_operating_condition` command.

Any operating parameter value specified using the `set_operating_parameter` command has a higher precedence than the `set_operating_condition` command. However, other parameters not specified using the `set_operating_parameter` command will continue to be picked up from specified operating condition. Even if all operating parameters are specified, the wire load tree type is still picked up from operating condition.

Note: The `do_remove_design -all` command does not remove operating conditions, regardless of how the operating conditions are set in the design.

Options and Arguments

Note: Set only one of the parameters (process, voltage, or temperature) per command.

`-process` *float*
: Specifies the process multiplier.

`-pvt {min | typ | max}`
: Sets the operating parameter for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified operating parameter applies to all three PVT corners.

`-temperature` *float*
: Specifies the temperature at which the circuit operates.

`-voltage` *`float`*

Specifies the voltage at which the circuit operates.

Examples

```
set_operating_parameter -voltage 3.3 -pvt typ
set_operating_parameter -process 1.0 -pvt typ
set_operating_parameter -temperature 25 -pvt typ
```

Tip

This is equivalent to the following command:

```
read_tlf -process 1.0 -voltage 3.3 -temperature 25 -pvt typ zippy.tlf
```

However, without the `-pvt typ` option, the `read_tlf` command would apply the parameters to all three PVT corners by default. In that case, results would differ from the set of `set_operating_parameter` commands with the `-pvt typ` option because their `min` and `max` values are taken from the library.

Note: If the temperature or voltage k factors are undefined in the library, or are set to zero, the `set_operating_parameter` will have *no* scaling effect on timing.

Related Information

`get_operating_parameter`

`load_dcl_rule`

`reset_operating_parameter`

`set_operating_conditions`

See the Reading Timing Libraries chapter in the *Common Timing Engine (CTE) User Guide* for information on setting operating parameters for timing libraries.

set_operating_voltage

```
set_operating_voltage [-ground] [-pvt {min | typ | max}] voltage
     list_of_instances
```

Models operating voltage for hierarchical or leaf level instances. This command is intended for use in voltage drop analysis. For more information about IR drop, see "Using Timing Analysis with Voltage Drops on Power and Ground Rails" in the *Common Timing Engine (CTE) User Guide*.

Note: Full support for multiple supply voltages is not available in this release.

The operating voltage of a leaf level instance is determined in the following manner:

- If there is an operating voltage assertion on the instance, then that specifies its operating voltage.
- Otherwise, the voltage of the lowest level hierarchical module containing this instance, and with an operating voltage assertion is the operating voltage of the instance.
- If no such module is found, then the operating voltage is determined by the operating condition command, or by the default operating condition (or operating voltage) setting in the first library in target technology list.

Note: The `set_operating_parameter -voltage` command can be set for the entire design and is used in cell derating. Set an operating voltage on a leaf level instance using the `set_operating_voltage` command, which is also used in derating, but overrides the value set by `set_operating_parameter -voltage` command only for this instance.

For interconnect delay and slew calculations, the operating voltage of the driver is used. The voltage drop of the instance (if any) connected to the receiver will not affect its delay or slew (other than affecting receiver pin capacitance).

For cell delay and slew calculation, `.lib` and TLF allow only one set of derating factors. Therefore, the instance and its inputs are expected to be operating at the same voltage, and this voltage is used to derate the cell timing arcs.

Options and Arguments

`-ground`
: Specifies the ground bounce-voltage of an instance.

`list_of_instances`
: Specifies the list of instances includes hierarchical and leaf level instances. Specify the instance list after the voltage value.

`-pvt {min | typ | max}`

Sets the operating voltage for a particular PVT corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces (`{}`) and separate the values by spaces.
Default: The specified operating voltage applies to all three PVT corners.

`voltage`

Specifies the operating voltage of the instance. This floating number is a required argument.

Examples

- The following commands set the operating voltage triplet to `4.5:5.0:5.5` volts for instance `ld1`:

```
set_operating_voltage -pvt min 4.5 ld1
set_operating_voltage -pvt typ 5.0 ld1
set_operating_voltage -pvt max 5.5 ld1
```

- The following commands sets the ground voltage on instance il:

```
set_operating_voltage 1.8 i1
set_operating_voltage -ground 0.1 i1
```

- The following command provides the effective voltage operating on instance `i1`, which is `1.7`:

```
get_operating_voltage i1
```

Related Information

`get_operating_voltage`

`read_rrf`

`read_irdrop`

`reset_operating_voltage`

`set_operating_conditions`

`set_operating_parameter`

set_path_delay_constraint

```
set_path_delay_constraint [{-from | -from_rise | -from_fall} pin_list]
     [{-through | -through_rise | -through_fall} pin_list]
     [{-to | -to_rise | -to_fall} pin_list] [-early | -late]
     [-clock_from list_of_clk_signames] [-edge_from {leading | trailing}]
     [-clock_to list_of_clk_signames_] [-edge_to {leading | trailing}]
     [-bidi_input_from | bidi_output_from] [-bidi_input_through |
     -bidi_output_through] [-bidi_input_to | bidi_output_to] [-reset_path] float
```

Specifies a delay that must be met (`-late`) or exceeded (`-early`).
Default: `false`

Path exception constraint calculations are different depending on type of start or end point:

- Sequential start or end point

  Considers the clock insertion delay, uncertainty, and arrival times in the path delay constraint calculations. Effectively, the delay *through* the pin is calculated, considering the clock network delay as well.

  The interpretation of a path delay statement on a hierarchical port is that of a `-through` constraint. Since a hierarchical port is a logical pin and does not represent a real circuit pin, constraints on such pins are treated as through constraints. If you want the constraint to be such that it is a path start or end point, move the constraint to a previous or later leaf level pin.

- Combinational start or end point

  Does not consider any clock delay, and the delay path constraint is measured from or to the pin.

Note: The global `timing_allow_register_output_as_delay_through` controls whether register outputs are seen as sequential or combinational start points. The default is `false`, meaning register outputs are combinational start points and the delay path constraint is measured from the pin.

If both the `-from` and `-clock_from` options are specified, the path exception applies to all paths starting from the `-from` pin and all the paths starting from the `-clock_from` signal. If both the `-to` and `-clock_to` options are specified, the path exception applies to all the paths ending at the `-to` pin and all the paths ending at the `-clock_to` signal.

If both the `-through` and `-clock_from` options are specified, the path exception applies to all paths starting from the `-clock_from` signal and going through the `-through` pin (AND operation). If both the `-through` and `-clock_to` options are specified, the path exception applies to all paths going through the `-through` pin and ending at the `-clock_to` signal.

Options and Arguments

`-bidi_input_from | -bidi_output_from`
Specifies that the assertion applies to the input or output of bidirectional pins in the `-from` pin list. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_through | -bidi_output_through`
Specifies that the assertion applies to the input or output of bidirectional pins in the `-through` pin list. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-bidi_input_to | -bidi_output_to`
Specifies that the assertion applies to the input or output of bidirectional pins in the `-to` pin list. Default value is shown in "Bidirectional Pin Defaults" on page 799.

`-clock_from` *`list_of_clk_signames`*
Sets the delay for all paths originating from the specified clock signal(s).

`-clock_to` *`list_of_clk_signames`*
Sets the delay for all paths going to the specified clock signal(s).

`-edge_from {leading | trailing}`
Specifies an edge for the `-clock_from` option.
Default: Both edges

`-edge_to {leading | trailing}`
Specifies an edge for the `-clock_to` option.
Default: Both edges

`float`
Specifies the delay value for the constraint.

`{-from | -from_rise | -from_fall}` *`pin_list`*
Specifies the pins that are at the start of the paths. If the `-from` option is used without the `-to` option, all paths originating from the *`pin_list`* are constrained.

Either the `-from`, `-to`, or `-through` option must be used with the `set_path_delay_constraint` command to identify the

paths. Specifying both the `-from` and `-to` options identifies a path with specific starting and ending points.

The *`pin_list`* argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Use only one `-from` (or `-from_rise` or `-from_fall`) option per command.

Specifying `set_path_delay_constraint -from` a hierarchical pin treats the hierarchical pin as a `-through`. A `-from` pin must be an instance pin in order to place a constraint on a combinational path.

By default, the `-from` option applies the assertion to both the rising and the falling edges.

Using the `-from_rise` option applies the assertion at the rising edge of the signal on the through pins.

Using the `-from_fall` option applies the assertion at the falling edge of the signal on the through pins.

`-late | -early`

Specifies the maximum delay (`-late`) or minimum delay (`-early`).
Default: `-late`

`-reset_path`

Changes the constraint on the specified path to the new value. Using this option resets a path only if the path description using the `-from`, `-through`,`-to` options match. See Examples.

`{-through | -through_rise | -through_fall}` *`pin_list`*

Specifies the pins that the path goes through. When used without the `-from` or `-to` options, all paths that go through the `-through` pins have the specified delay.

When used with both the `-from` and `-to` options, any path starting at any pin on the `-from` list *and* going through any point on the `-through` pin list *and* going to any point on the `-to` list is affected by the assertion.

The *pin_list* argument can contain either object IDs or hierarchical names relative to the current module. Use only one `-from` and one `-to` option, but many `-through` options in the same command.

By default, using the `-through` option applies the assertion to both the rising and the falling edges.

Using the `-through_rise` option applies the assertion at the rising edge of the signal on the through pins.

Using the `-through_fall` option applies the assertion at the falling edge of the signal on the through pins.

`{-to | -to_rise | -to_fall}` *pin_list*

Specifies the pins that are at the end of the paths. If the `-to` option is used without the `-from` option, all paths ending in the `-to` pin list have the specified delay.

The *pin_list* argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Use only one `-to` (or `-to_rise` or `-to_fall`) option per command.

Specifying `set_path_delay_constraint -to` a hierarchical pin treats the hierarchical pin as a `-through`. A `-to` pin must be an instance pin in order to place a constraint on a combinational path.

By default, the `-to` option applies the assertion to both the rising and the falling edges.

Using the `-to_rise` option applies the assertion at the rising edge of the signal on the through pins.

Using the `-to_fall` option applies the assertion at the falling edge of the signal on the through pins.

Examples

- The following command constrains the delay on the path from `in` to `out` to a maximum of 3:

```
set_path_delay_constraint -from in -to out 3
```

- The following command shows the `-reset_path` option. The second example sets the constraint to 6.0 and resets the 4.0 constraint to o3:

```
set_path_delay_constraint 4.00 -to { o1 o2 o3 o4 o5 }
set_path_delay_constraint 6.00 -reset_path -to o3
```

  Using the `-reset_path` option changes the constraint on the specified path to the new value. However, if there is only one path and both paths are equivalent, using this option resets a path only when there is an exact match in the arguments. For example, the following command specifies the delay as `U14/A 5`:

```
set_path_delay_constraint -through U14/A 5
```

  The following shows the report using this command:

```
+--------------------------+
| Through | Early |  Late   |
|---------+-------+---------|
| U14/A   |       | delay 5 |
+--------------------------+
```

  If you then change the constraint to a new value using the `-reset_path` option on the same path, but use a different argument as follows:

```
set_path_delay_constraint -through U12/Z 7 -reset_path
```

  The report using the `report_path_exceptions` command or the `report_timing` command shows the following:

```
+-----------------------------------+
| Through | Early |      Late       |
|---------+-------+-----------------|
| U12/Z   |       | delay 7 ignored |
| U14/A   |       | delay 5         |
+-----------------------------------+
```

  Using the `-reset_path` option resets a path only when there is an exact match in the arguments, not when both the constraints represent the same path.

Related Information

`reset_path_exception`

`set_cycle_addition`

See Constraint Tips and Specifying Path Delay Constraints in the *Common Timing Engine (CTE) User Guide*.

set_path_group

```
set_path_group [-name group_name] [-from from_list] [-through through_list]
     [-to to_list] [-clock_from clksig_from_list] [-clock_to clksig_to_list]
```

Specifies path groups for use in optimization. A path group is a collection of timing paths in the design, for which you can set different optimization objectives. See set_port_capacitance on page 1238 for more details on how to set these values once you've specified the groups.

Specifying a path group is similar to specifying a path exception, such as a false path. You can give multiple path specifications the same name, which results in all the specified paths being placed in the same path group. You can also give path groups different names (or a default name) and placed in different groups.

Initially, there is only one group, which is named "default." You cannot create any other groups with that name.

To optimize each waveform or clock in a different group, use the following script:

```
foreach clk [get_clock] { set_path_group -clock_to $clk -name "group_$clk" }
```

Note: Timing reports show different slack times when defining and without defining path groups because of slew merging. For more information on timing differences due to slew merging, see Explaining Timing Differences Caused by Slew Merging When Path Exceptions are Applied in the *Common Timing Engine (CTE) User Guide*.

Options and Arguments

`-clock_from` *`clksig_from_list`*
: Includes paths in the group whose start points are related to the listed clock signals. This includes paths from flip-flops in the transitive fanout of the clock source pin or port, and paths from ports that have input delay related to the clock waveform.

`-clock_to` *`clksig_to_list`*
: Includes paths in the group whose endpoints are related to the listed clock signals. This includes paths to flip-flops in the transitive fanin of the clock output pin or port, and paths to ports that have output delay related to the clock waveform.

`-from` *`from_list`*
: Specifies the path start points. The *`from_list`* is a list of port or pin names. Paths starting from *`from_list`* are included in the group.

Note: Do not use `-from` option with the `-from_clock` option.

`-name` *`group_name`*

Creates a new group with name *`group_name`*. If a group with this name already exists, the paths or endpoints are added to that group.

The name `default` is an illegal name for a path group. It is reserved for the default path group (paths not belonging to any specified path group).

Note: If you do not specify the `-name` option, the system creates a name. This new name is unique, and different from all other group names. Once it has been created, you can use the new name just like any other group name. Be advised, however, that the created name may not be the same from run to run, and therefore, it is better to specifically assign names to your path groups.

Tip

Avoid problems by always assigning a group name with the `-name` option. Remember that `default` is not a legal name.

`-through` *`through_list`*

Specifies the path through points. The *`through_list`* is a list of port or pin names, or leaf cell names. The path group includes only those paths that pass through one of the points in the *`through_list`*.

If the `-through` option is used with the `-from` option, the group includes only paths that start in the *`from_list`* and pass through the *`through_list`*.

If the `-through` option is used with the `-to` option, the group includes only paths that pass through the *`through_list`* and end at the *`to_list`*.

If the `-through` option is used with both the `-from` and `-to` options, the group includes only paths that start in the *`from_list`* and pass through the *`through_list`* and end at the *`to_list`*.

If the `-through` option is specified multiple times, the group includes only those paths that pass through one of the points in each *`through_list`* in the order specified.

`-to` *`to_list`*

Specifies the path endpoints.The *`to_list`* is a list of port or pin names. Paths ending at *`to_list`* are included in the group. **Note**: Do not use the `-to` option with the `-to_clock` option.

Examples

- The following command specifies path groups for use in optimization, and includes the path in the group whose start point is related to `CLKA -name GRPA` signal:

  ```
  set_path_group -clock_from CLKA -name GRPA
  ```

- The following command sets a `G_PiClk` path group:

  ```
  set_path_group -name G_PiClk -clock_from {PiClk} -clock_to {PiClk}
  ```

  Using the `report_path_groups` command or the `report_timing -path_group G_PiClk` command displays the following report:

```
+-------------------------------+
|  From   |    To    |  Group   |
|         |          |   Name   |
|---------+----------+----------|
| "PiClk" | "PiClk"  | G_PiClk  |
| "PiClk" | "VdoClk" | test_pjj |
+-------------------------------+
```

Related Information

`report_timing -path_group`

`reset_path_group`

`set_port_capacitance`

See Using Path Groups for Optimization in the *Common Timing Engine (CTE) User Guide*.

set_port_capacitance

`set_port_capacitance` *`float port_list`* `[-pvt {min | typ | max}] [-wire_cap]`

Specifies top level input and output ports only, and specifies the capacitance external to the design based on input and output loading from other ports and nets connected to the ports of the module.

The capacitance at any port is the sum of the external port capacitances that are connected to the port. For an input port, the port capacitance refers to the capacitance of all the ports that drive the net and the capacitance of the other loads on the net. For output ports, the port capacitance refers to the capacitance of all the external sinks, (and external drivers, for the case of a net with multiple drivers) that are connected to the net, as well as the capacitance of the input port of any other external drivers.

Options and Arguments

`float`

Specifies the capacitance value. The unit of capacitance must be the same as the unit of capacitance used in the library.

`port_list`

Specifies the list of top level ports.

`-pvt {min | typ | max}`

Sets the port capacitance for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified port capacitance applies to all three PVT corners.

`-wire_cap`

Specifies the external wire capacitance.The default specified value is the external pin capacitance.

Note: The external wire capacitance is ignored if you specify the external number of source pins or the external number of sink pins, and the external wire capacitance on the same port.

Example

The following command sets a 3.2 capacitance on all output ports of the current module whose names start with the string `dbus`:

```
set_port_capacitance 3.2 [find -port -output dbus*]
```

Related Information

`reset_port_capacitance`

`set_current_module`

`set_global max_capacitance_limit`

`set_port_capacitance_limit`

`set_top_timing_module`

set_port_capacitance_limit

```
set_port_capacitance_limit [-min | -max] float port_list
```

Specifies top level input and output ports only, and sets the limit on the capacitance external to the design based on input and output loading from other ports and nets connected to the specified ports of the current module.

Tip

Make sure the current module is the top timing module before applying this command.

Specify this design rule constraint on top level input and output ports. The limit, by default, is the maximum value for the total capacitances (wire capacitance and pin capacitance) of nets attached to the ports in the port list. Specify a minimum value limit with the `-min` option.

If the `set_global max_capacitance_limit` (or `set_global min_capacitance_limit`) has also been set, the most constraining value is used.

Options and Arguments

`float`
: Specifies the capacitance value. Units must match the units used in the library.

`-min | -max`
: Specifies minimum (or maximum) design rule constraints. If neither option is given, the default is `-max` for backward compatibility.

`port_list`
: Specifies the list of top level ports.

Example

```
set_port_capacitance_limit 5.0 [find -port dbus*]
```

Related Information

`do_derive_context`

`do_time_budget`

set_port_wire_load

```
set_port_wire_load [-library library_name] wire_load_model port_list
     [-pvt {min | typ | max}]
```

Specifies the wire load for an input or output top level port of a design. The net connected to the port is associated with the specified wireload model, which is used for wire capacitance and resistance estimation.

If a net is connected to more than one port with a `set_port_wire_load` assertion, then the worst wireload model on the net is computed and used for timing analysis.

Options and Arguments

`-library` *`library_name`*
: Indicates the library containing the specified wire-load model. If this option is not specified, the default is the target technology library.

`port_list`
: Lists the top level ports.

`-pvt {min | typ | max}`
: Sets the port wire-load model for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified port wire-load model applies to all three PVT corners.

`wire_load_model`
: Specifies the name of the wire-load model to use for estimating wire capacitance and resistance.

Example

```
set_port_wire_load B0X0 out
```

See Specifying Wire Loads in the *Common Timing Engine (CTE) User Guide* for more examples and detailed information.

Related Information

set_num_external_sinks

set_wire_load

set_propagated_clock

```
set_propagated_clock [-clock clock_list] [-pin pin_list]
```

Changes the clock propagation mode to propagated mode for the given clock waveforms or pins. There are two modes of clock propagation: `ideal` and `propagated`.

- In `ideal` mode, the delay from a clock port to a register clock pin (network insertion delay) comes from design constraints (`set_clock_insertion_delay`).
- In `propagated` mode, the network insertion delay is computed from the actual gates and interconnects in the clock network.

Note: Use the `set_clock_propagation` command to globally change the clock propagation mode for all clock networks. If the `set_clock_propagation` command is set to `propagated` (`set_clock_propagation propagated`), then all clock networks are assumed to be in `propagated` mode, and the `set_propagated_clock` and `reset_propagated_clock` commands will have no impact. If the `set_clock_propagation` command is set to `ideal` mode (`set_clock_propagation ideal`), then you can change some clock networks to `propagated` mode by using the `set_propagated_clock` command.

Options and Arguments

`-clock` *`clock_list`*

Specifies the clock where you want to change the clock propagation mode to propagated mode.

`-pin` *`pin_list`*

Specifies one or more pins for which you want to change the clock propagation mode to propagated mode. No hierarchical pins are allowed in the *`pin_list`*. When a *`pin_list`* is specified, it effects the propagation mode for all the registers in the transitive fanout (TFO) of the pin or port.

Either the *`clock_list`* argument or the *`pin_list`* argument must be specified.You can also specify both the *`clock_list`* and the *`pin_list`* arguments.

Examples

- The following command sets clock waveform `CLK1` to propagated mode:

```
set_propagated_clock -clock CLK1
```

- The following command changes clock waveform `CLK1` back to ideal mode:

```
reset_propagated_clock -clock CLK1
```

Related Information

get_clock_propagation

get_propagated_clock

reset_propagated_clock

set_clock

set_clock_insertion_delay

set_clock_propagation

set_clock_root

set_clock_uncertainty

set_propagated_clock

set_scale_delays

```
set_scale_delays [-clock | -data]
     [-net_delay | -cell_delay | -cell_check]
     [-incl_sdf] [-early | -late]
     [-pvt {min | typ | max}] scale_factor
```

Scales `min`, `typ`, or `max` delays from the library and optionally from SDF assertions by the specified scaling factor. To specify unique scaling factors and PVT for cell and net, issue the command multiple times. Use this command to remove or add a certain degree of pessimism.

Options and Arguments

`-cell_check`

Specifies scaling on timing checks.

Note: Specifying the `-cell` option is equivalent to both the `-cell_delay` and `-cell_check` options.

`-cell_delay`

Specifies scaling on cell delays.
If neither the `-cell_delay` nor the `-net_delay` option is specified, the specified scaling factor applies to both types of delays.

`-clock`

Applies scaling only to clock networks.

`-data`

Applies scaling only to data networks.

Note: If neither the `-clock` option or the `-data` option is specified, the scaling applies to both the clock and data networks.

`-early | -late`

Specifies the variation around best case (BC) and worst case (WC) corners. Use these options to specify four corner data as follows:

```
-early -min
-late -min
-early -max
-late -max
```

`-incl_sdf`
: Specifies scaling on SDF assertions.

`-net_delay`
: Specifies scaling on net delays.

`-pvt {min | typ | max}`
: Associates a min, typ, or max PVT corner with the scale delay. Three PVT corners are allowed, namely, `min`, `typ`, and `max`. *Default*: The specified delay scaling is applied to all PVT corners.

`scale_factor`
: Specifies the factor used to scale the delay. Scaling is specified by the decimal notation: `x.n` where *x* = 0 for negative scaling (reducing) or 1 for positive scaling (increasing), and *n* = 0 to 99.
For example: 0.95 scales delays by -5 percent
1.1 scales delays by 10 percent
1.5 scales delays by 50 percent

Examples

- The following commands scales the library delays to 95 percent of the original value for the min PVT corner. For the max PVT corner, the library delays are scaled to 105 percent of the original value:

```
set_scale_delays -incl_sdf -pvt min 0.95
set_scale_delays -incl_sdf -pvt max 1.05
```

- The following commands set different delay scaling for cells and nets:

```
set_scale_delays -cell_delay -pvt min 0.90
set_scale_delays -cell_delay -pvt max 1.1
set_scale_delays -net_delay -pvt min 1.1
set_scale_delays -net_delay -pvt max 1.3
```

Related Information

`get_scale_delays`

`reset_scale_delays`

See Analyzing BC-WC Set-Up and Hold Checks" in the *Common Timing Engine (CTE) User Guide* for more information.

set_slew_limit

`set_slew_limit` *`float port_list`* `- Renamed to set_slew_time_limit`

The `set_slew_limit` command has been renamed because it limits the slew time parameter. See set_slew_time_limit on page 1253.

set_slew_thresholds

`set_slew_thresholds [-lower` *`lower_float`*`] [-upper` *`upper_float`*`]`

Sets the lower and upper slew threshold values used for the entire design, which is called a single-threshold mode. By default, the rising input slew thresholds from the default target technology library are used for reporting all rise and fall slews. If the target technology library does not have thresholds defined, the defaults of 20-80 percent are used (some library formats use defaults of 10-90 percent, see Using Different Default Thresholds for Different Library Formats in the *Common Timing Engine (CTE) User Guide*). To override these default thresholds, specify the single-threshold values for the whole design using the `set_slew_thresholds` command.

The `set_slew_thresholds` command is used only for reporting purposes in the user interface. It is still your responsibility to specify slew values that conform to the design level thresholds when you use the commands `set_slew_time`, `set_slew_time_limit`, `set_default_slew_time`, and `set_global slew_time_limit`.

When you use the `set_slew_thresholds` command, the delay or slew calculation scales the incoming slews to the appropriate thresholds specified in the library for this cell. After the delay or slew calculation, the output slews are scaled back to the design slew threshold values. Also, the max and min transition limits at pin level and the default `max` and `min` transition limits at library level are scaled appropriately.

Options and Arguments

`-lower` *`lower_float`*
: Specifies the value of the lower slew threshold value in percentage between 0 to 100 percent.

`-upper` *`upper_float`*
: Specifies the value of the upper slew threshold value in percentage between 0 to 100 percent.

Specify both the upper and lower values together. The values must conform to *`upper_float`* > *`lower_float`*.

Note: If you set the slew threshold value at a percentage of *less* than one, you will get an error message. For example: `set_slew_threshold -lower 0.1 -upper 0.9`. You will also get an error message if the specified upper threshold is the same as the lower threshold.

Example

The following command sets the lower and upper slew threshold values used for the entire design:

```
set_slew_thresholds -lower 10 -upper 90
```

Related Information

`get_slew_thresholds`

`reset_slew_thresholds`

`set_global_lib_cell_thresholds_for_reporting`

set_slew_time

```
set_slew_time [-clock clk_signame] [-early | -late] [-rise | -fall]
     [-lead | -trail] |-pos | -neg] time port_list
```

Specifies the slew time for a port on the top level module of a design or for a pin on an instance of a technology cell anywhere in the hierarchy. This command is useful only in specifying a hard ramp signal as if the pin were driven by a very powerful driver. The command does not change the arrival time of an input, but changes the slew time used to compute the delay of the cell on the sink of a net.

Note: The `set_slew_time` command is ignored for a port if the `set_drive_cell` or `set_drive_resistance` constraints have been set for that port.

Options and Arguments

`-clock` *`clk_signame`*

Specifies the name of the ideal (or generated) clock waveform that is associated with the slew time. If this option is not specified, the assertion applies to all clocks.

`-early | -late`

Indicates whether the slew times are with respect to the early (hold) or the late (setup) time checks of the data signal. If neither time is specified, the default is to use both.

`-lead | -trail`

Specifies whether the slew time data signal is triggered by the leading edge or trailing edge of the clock signal.
Default: `-lead`

`port_list`

Specifies the list of top level nets or technology cell instance pins for which the slew time is specified.

`-pos | -neg`

Specifies, for a clock signal, that the slew time should be applied to an actual clock having a positive or negative polarity with respect to the clock. Specify only one option, either `-pos` or `-neg` per command.
Default: `-pos`

`-rise | -fall`

Specifies that the slew time should be applied to the rising edge

or the falling edge of the signal (data or clock). If neither time is specified, the default is to use both.

time

Specifies the slew time.

Examples

```
set_slew_time 0.5 input3
set_slew_time -clock B0 -pos 0.1 clk_b
```

Related Information

reset_slew_time

set_drive_cell

set_drive_resistance

set_global slew_propagation_mode

set_global slew_time_limit

set_slew_time_limit

set_slew_time_limit

```
set_slew_time_limit [-min | -max] float [-port port_list] [-clock clock_list]
     [-module module_list]
```

Specifies the limit (maximum or minimum) for slew time at the input and output ports of a module. The slew limit is usually derived from the design rule constraints placed on the cells that an output port is driving or an input port is driven by.

Use this command to override the default slew time limit placed on specific ports by the `set_global max_slew_time_limit` or the `set_global min_slew_time_limit` globals.

See Specifying Slew Rates in the *Common Timing Engine (CTE) User Guide* for more information.

Options and Arguments

`-clock clock_list`
: Specifies the maximum slew limit for one or more clock waveforms.

`float`
: Specifies the maximum (or minimum) slew allowed.

`-min | -max`
: Specifies minimum (or maximum) design rule constraints. If neither option is given, the default is `-max` for backward compatibility.

`-module module_list`
: Specifies the list of modules to which the limit applies. All instances of the specified modules will inherit the limit value.

`-port port_list`
: Specifies the list of ports to which the limit applies.

Example

The following command sets the maximum slew time limits for `bus1` and `bus2` to `2.3`:

```
set_slew_time_limit 2.3 {bus1 bus2}
```

Related Information

reset_slew_time_limit

set_capacitance_limit

set_global max_slew_time_limit

set_global min_slew_time_limit

set_global slew_propagation_mode

set_slew_time

set_tech_info

```
set_tech_info
     {([-library list_of_library_names]
     [-default_wire_load wlm_name]
     [-default_wire_load_selection wireload_selection_table]
     [-default_operating_conditions op_cond_name]
     [-default_fanout_load float] [-default_max_capacitance float]
     [-default_max_fanout value] [-default_max_transition value]
     [-default_min_capacitance value] [-default_min_fanout value]
     [-default_min_transition value]
     [-input_threshold_pct_rise float] [-input_threshold_pct_fall float]
     [-output_threshold_pct_rise float] [-output_threshold_pct_fall float]
     [-slew_lower_threshold_pct_rise float]
     [-slew_lower_threshold_pct_fall float]
     [-slew_upper_threshold_pct_rise float]
     [-slew_upper_threshold_pct_fall float]
     [-slew_lower_meas_threshold_pct_rise] |
     [-slew_lower_meas_threshold_pct_fall] |
     [-slew_upper_meas_threshold_pct_rise] |
     [-slew_upper_meas_threshold_pct_fall]}
     [-pvt {min | typ | max}])
     |
     ([-library list_of_library_names]
     -cell list_of_cell_names
     {[-dont_modify true | false] [-dont_utilize true | false]
     [-scaling_factors value]
     [-input_threshold_pct_rise float] [-input_threshold_pct_fall float]
     [-output_threshold_pct_rise float] [-output_threshold_pct_fall float]
     [-slew_lower_threshold_pct_rise float]
     [-slew_lower_threshold_pct_fall float]
     [-slew_upper_threshold_pct_rise float]
     [-slew_upper_threshold_pct_fall float]
     [-slew_lower_meas_threshold_pct_rise] |
     [-slew_lower_meas_threshold_pct_fall] |
     [-slew_upper_meas_threshold_pct_rise] |
     [-slew_upper_meas_threshold_pct_fall]}
     [-pvt {min | typ | max}])
     |
     ([-library list_of_library_names]
     -cell list_of_cell_names
     -pin list_of_pin_names
     [-fanout_load value] [-max_fanout value] [-min_fanout value]
     [-max_transition value] [-min_transition value]
     [-max_capacitance float] [-min_capacitance float]
     [-pvt {min | typ | max}])}
```

Makes assertions (overrides values) for the specified parameters in the named target technology libraries. Apply assertions to the library, cell, or pin level and specify one PVT value at a time.

Use this command to loosen overly pessimistic values in the library. Be careful when applying overrides to vendor determined values.

Note: The technology library must contain the attributes for which you want to overwrite the default values.

Options and Arguments (any level)

`-library` *`list_of_library_names`*
Overrides data for the named libraries.
Default: If this option is not specified, this command affects only the first library specified with the `set_global target_technology` global.

`-pvt {min | typ | max}`
Sets the value for the environment corner of interest. Set one PVT corner per command.
Default: `typ`.

Note: If you use the `set_tech_info` command without the `-pvt` option, the command will set the information on all pvt corners that are already loaded in the library.

Options and Arguments (library level)

`-default_fanout_load` *`float`*
Specifies the value of the default fanout load for all pins on all cells in the library.

`-default_max_capacitance` *`float`*
Specifies the value of the default maximum capacitance for all pins on all cells in the library.

`-default_max_fanout` *`float`*
Specifies the value of the default maximum fanout for all pins on all cells in the library.

`-default_max_transition` *`float`*
Specifies the value of the default maximum transition time for all pins on all cells in the library.

`-default_min_capacitance` *`float`*
Specifies the value of the default minimum capacitance for all pins on all cells in the library.

`-default_min_fanout` *`float`*
Specifies the value of the default minimum fanout for all pins on all cells in the library.

`-default_min_transition` *`float`*
Specifies the value of the default minimum transition time for all pins on all cells in the library.

`-default_operating_conditions` *`op_cond_name`*
Specifies *`op_cond_name`* as the default operating conditions for the library.

`-default_wire_load_selection` *`wireload_selection_table`*
Specifies the default wire-load selection table for the library.

`-default_wire_load` *`wlm_name`*
Specifies the default wireload for the library.

`-input_threshold_pct_fall` *`float`*
Specifies the value of the default input threshold percent for the falling edge for all cells in the library.

`-input_threshold_pct_rise` *`float`*
Specifies the value of the default input threshold percent for the rising edge for all cells in the library.

`-output_threshold_pct_fall` *`float`*
Specifies the value of the default output threshold percent for the falling edge for all cells in the library.

`-output_threshold_pct_rise` *`float`*
Specifies the value of the default output threshold percent for the rising edge for all cells in the library.

Note: The following slew threshold options refer to the section of the waveform where the slew is nearly linear. This is measured and extended like a linear waveform. The linear waveform, a result of extending the near-linear slew waveform, is measured at certain points. The points where this extended waveform is measured are called slew thresholds. Measured slew thresholds are determined at the time of library characterization.

Slew Thresholds

`-slew_lower_threshold_pct_fall` *`float`*
Specifies the value of the default lower threshold percent for the slew time of the falling transition for all cells in the library.

`-slew_lower_threshold_pct_rise` *`float`*
Specifies the value of the default lower threshold percent for the slew time of the rising transition for all cells in the library.

`-slew_upper_threshold_pct_fall` *`float`*
Specifies the value of the default upper threshold percent for the slew time of the falling transition for all cells in the library.

`-slew_upper_threshold_pct_rise` *`float`*
Specifies the value of the default upper threshold percent for the slew time of the rising transition for all cells in the library.

Measured Slew Thresholds

`-slew_lower_meas_threshold_pct_fall`
Specifies the measured lower threshold percent of the slew time for the falling transition for all cells in the library.

`-slew_lower_meas_threshold_pct_rise`
Specifies the measured lower threshold percent of the slew time for the rising transition for all cells in the library.

`-slew_upper_meas_threshold_pct_fall`
Specifies the measured upper threshold percent of the slew time for the falling transition for all cells in the library.

`-slew_upper_meas_threshold_pct_rise`
Specifies the measured upper threshold percent of the slew time for the rising transition for all cells in the library.

Options and Arguments (cell level)

`-cell` *`list_of_cell_names`*
Specifies the cells to which the assertions are applied. Required argument for cell and pin level assertions.

`-dont_modify {true | false}`
When set to `true` this option puts the `dont_modify` property on the named cells. These cells are not modified by optimization or time budgeting.
Default: `false`.

`-dont_utilize {true | false}`
Puts the `dont_utilize` property on the named cells, when set to `true`. These cells are not used in optimization.
Default: `false`

`-input_threshold_pct_fall` *`float`*
Specifies the value of the default input threshold percent for the falling edge for the listed cells.

`-input_threshold_pct_rise` *`float`*
Specifies the value of the default input threshold percent for the rising edge for the listed cells.

`-output_threshold_pct_fall` *`float`*
Specifies the value of the default output threshold percent for the falling edge for the listed cells.

`-output_threshold_pct_rise` *`float`*
Specifies the value of the default output threshold percent for the rising edge for the listed cells.

`-scaling_factors` *`derate_table_name`*
Specifies a derating table containing scaling factors for the cell delay of the named cells.

Note: The `read_library_update` command has been enhanced to update scaling factor groups to the target library similar to a wire-load models update. Use the `-scaling_factors` option with the `read_library_update` command to set a new scaling factors group for a cell or to change the scaling factors group of a cell within a library.

Note: The following slew threshold options refer to the section of the waveform where the slew is nearly linear. This is measured and extended like a linear waveform. The linear waveform, a result of extending the near-linear slew waveform, is measured at certain points. The points where this extended waveform is measured are called slew thresholds. Measured slew thresholds are determined at the time of library characterization.

Slew Thresholds

`-slew_lower_threshold_pct_fall` *`float`*
Specifies the value of the default lower threshold percent for the slew time of the falling transition for the listed cells.

`-slew_lower_threshold_pct_rise` *`float`*
Specifies the value of the default lower threshold percent for the slew time of the rising transition for the listed cells.

`-slew_upper_threshold_pct_fall` *`float`*
Specifies the value of the default upper threshold percent for the slew time of the falling transition for the listed cells.

`-slew_upper_threshold_pct_rise` *`float`*
Specifies the value of the default upper threshold percent for the slew time of the rising transition for the listed cells.

Measured Slew Thresholds

`-slew_lower_meas_threshold_pct_fall`
Specifies the measured lower threshold percent of the slew time for the falling transition for the listed cells.

`-slew_lower_meas_threshold_pct_rise`
Specifies the measured lower threshold percent of the slew time for the rising transition for the listed cells.

`-slew_upper_meas_threshold_pct_fall`
Specifies the measured upper threshold percent of the slew time for the falling transition for the listed cells.

`-slew_upper_meas_threshold_pct_rise`
Specifies the measured upper threshold percent of the slew time for the rising transition for the listed cells.

Options and Arguments (pin level)

`-fanout_load` *`float`*
Specifies the value of the fanout load for the named pins.

Overrides the library default value. Specify this option only for an input pin.

`-max_fanout` *`float`*
Specifies the value of the maximum fanout for the named pins. Overrides the library default value. Specify this option only for an output pin.

`-max_capacitance` *`float`*
Specifies the value of the maximum capacitance for the named pins. Overrides the library default value. Specify this option only for an output pin.

`-max_transition` *`float`*
Specifies the value of the maximum transition time for the named pins. Overrides the library default value. Specify this option for both input and output pins.

`-min_capacitance` *`float`*
Specifies the value of the minimum capacitance for the named pins. Overrides the library default value. Specify this option only for an output pin.

`-min_fanout` *`float`*
Specifies the value of the minimum fanout for the named pins. Overrides the library default value. Specify this option only for an output pin.

`-min_transition` *`float`*
Specifies the value of the minimum transition time for the named pins. Overrides the library default value. Specify this option for both input and output pins.

`-pin` *`list_of_pin_names`*
Specifies the pins to which the assertions are applied. Required argument for pin level assertions.

Note: The pin assertions apply to a specific pin type. If the correct pin type is not specified, an error is issued.

Examples

- The following command overrides data for the named libraries:

```
set_tech_info -library lib1 lib2 -default_fanout_load 1.4111
```

- The following command specifies the cell to which the assertions are applied, and puts the `dont_utilize` property on the named cell:

```
set_tech_info -cell FD2ESSA -dont_utilize true
```

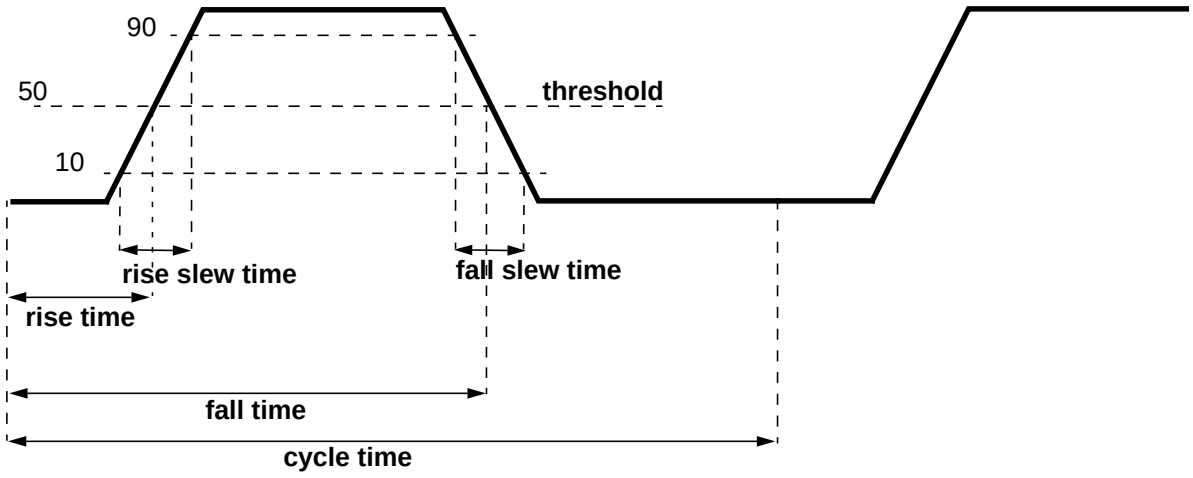
- The following command overrides data for the named library, specifies the cell and pins to which the assertions are applied, sets the value for the min environment corner, and specifies the value of the maximum fanout for the named pins:

```
set_tech_info -library lca300kv -cell B2I -pin Z1 Z2 -pvt min -max_fanout 3
```

- The following commands set the values shown in Figure 7-17 for all cells in the target technology library:

```
set_tech_info -input_threshold_pct_rise 50 -input_threshold_pct_fall 50
set_tech_info -output_threshold_pct_rise 50 -output_threshold_pct_fall 50
set_tech_info -slew_lower_threshold_pct_rise 10
-slew_upper_threshold_pct_rise 90
set_tech_info -slew_lower_threshold_pct_fall 10
-slew_upper_threshold_pct_fall 90
```

Figure 7-17 Thresholds Example



Related Information

`get_tech_info`

`reset_tech_info`

`write_library_assertions`

set_time_borrow_limit

`set_time_borrow_limit [-pin` *`pin_list`*`] [-`*`clock list_of_clksigs`*`] `*`borrow_limit`*

Specifies the maximum time that can be borrowed by one cycle from the next cycle in order to meet timing constraints. Time borrowing is sometimes called cycle stealing. Time borrowing is performed automatically when performing timing analysis.

Use the `set_time_borrow_limit` assertion on pins or waveforms. If the assertion is placed on an ideal (or generated) clock waveform, all latches triggered by this clock signal get the specified maximum time borrow limit. If both the clock waveform arriving at the clock pin of a latch and the clock pin itself have time borrow limit assertions, the maximum borrow limit on the latch is decided by the assertion on the latch clock pin.

Options and Arguments

`borrow_limit`

Specifies a non- negative real number, which is the maximum amount of time that can be borrowed. The minimum value allowed is 0, which disables time borrowing. The maximum allowed is equal to the pulse width minus the setup time: *`Max = pulse width - setup`*

`-clock` *`list_of_clksigs`*

Specifies the ideal (or generated) clock names to which the borrow limit applies.

If neither the `-pin` or `-clock` option is specified, the borrow limit is placed on all clocks.

`-pin` *`pin_or_instance_list`*

Specifies a single pin, multiple pins, or instances to which the borrow limit applies. To specify multiple pins, enclose the list with curly braces ({}) and separate the pin information with white space.

Use the `-pin` option with the clock pin of the latch (`L31/G`) as shown in the example below.

Example

```
set_time_borrow_limit -pin L31/G 0.1
```

Related Information

`get_time_borrow_limit`

`reset_time_borrow_limit`

Analyzing Latch-Based Designs in the *Common Timing Engine (CTE) User Guide*.

set_time_unit

```
set_time_unit float_unit
```

Specifies the time unit for the session. If you do not include this command, the time unit specified in the first library on the target technology list is used as the session time unit.

All timing assertions (such as the `set_input_delay` and the `set_slew_time` commands) for the session must use this time unit. All reports will also use this time unit. To find out the time unit for the session, use the `get_time_unit` command.

The `set_time_unit` command resets all assertions in the new unit that you specify, so expect the delay and slew numbers to be changing based on the new assertions. For example, after the following commands are used, the slew time asserted to all input ports is (1 * 10) ns:

```
set_time_unit 10ns
set_slew_time 1 [find -input *]
```

If you then set the time unit to something else in the middle of the session, CTE will reset timing and recompute delays and slews. For instance, if you use `set_time_unit 100ns`, the original slew assertion is changed to (1 * 100)ns, because the `set_time_unit` command reapplies the timing assertions with the new unit.

Tip

Use the `set_time_unit` command when reading multiple libraries with different units. If you do not specify the `set_time_unit` command, you must load the libraries in the same order for all sessions to ensure that CTE uses the correct time unit. Also, it is safer to use the `set_time_unit` command in the beginning of a session before applying any assertions, and not change the units afterwards.

Options and Arguments

`float_unit`

Specifies the time unit for this session in nanoseconds.

Example

The following command sets the time unit to 0.001ns which is equivalent to 1ps:

```
set_time_unit 0.001
```

Related Information

`get_time_unit`

`reset_time_unit`

set_top_timing_module

`set_top_timing_module` *`module_name`*

Identifies the module to be used by subsequent commands as a context for setting timing constraints. This command does not reset the current module or instance.

All the constraints are set with reference to the module specified as the top timing module. The optimization commands operate on the module (and its hierarchy) set by the top timing module. If you set a new top timing module, change the context for the subsequent timing constraints and the optimization steps. The constraints applied to the previous top timing module are preserved but do not affect the steps carried out in the new top timing module.

Note: If a different module needs constraints set on its ports as part of the `top_timing_module` context, then set them using the `set_current_module` command.

See Deriving the Timing Context in Synthesis in the *Common Timing Engine (CTE) User Guide* for information and examples on how to set the timing context for synthesis, how to compile single or multiple modules, and how to compile lower-level time-budgeted modules using the `set_top_timing_module` command.

Important

If you subsequently set the current module or instance, make sure that the specified module or instance is below the top timing module in the hierarchy.

Options and Arguments

`module_name`

Specifies the name of the module being set as the timing context.

Example

The following command sets the `control_block` module as the context for all timing constraints. Specify timing constraints for the `control_block` or any instances in the downward hierarchical path:

```
set_top_timing_module control_block
```

Related Information

`do_optimize`

`do_xform_timing_correction`

get_top_timing_module

set_current_module

set_wire_capacitance

`set_wire_capacitance` *`float net_names`* `[-subtract_pin_caps] [-pvt {min | typ | max}]`

Overrides the wire capacitance value used during timing analysis. This command is not written out by the `write_assertions` or the `write sdc` commands, but wire capacitance annotations are saved in the `.adb` file.

By default, the total capacitance seen by the driver of the net is the specified wire capacitance, plus the sum of pin and port capacitance on the net. If you use the `-subtract_pin_caps` option, only the specified wire capacitance value is used.

For a PAD cell, the net PAD/out to output port exists only in a logical design but not in the physical design. It is for this reason the `set_wire_capacitance` assertion on the net that connects PAD/out to output port is not honored, and a 0 cap value is asserted. The same holds true if the PAD/input is connected directly to an input port. For example, if a design has two PAD instances, where the cell has one input, and one output, and the PAD instances outputs are connected to output ports, the `set_wire_capacitance` settings will not be annotated for (1 + 1) = 2 nets.

Note: The `set_wire_resistance` command is not affected, which can be successfully set on any net, including port-to-pad nets.

Options and Arguments

`float`
: Specifies the wire capacitance to be backannotated. Units are determined by the library.

`net_names`
: Specifies the list of nets having this capacitance.

`-pvt {min | typ | max}`
: Sets the capacitance for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified capacitance applies to all three PVT corners.

`-subtract_pin_caps`
: Calculates wire capacitance by subtracting the capacitance on pins and ports of the net from the specified wire capacitance

value. The wire capacitance that is backannotated does not include pin capacitance.

Examples

- The following commands backannotate a wire capacitance of 2 on the net named `foo` for `min` and `typ` environment corners, then a wire capacitance of 2.2 for `max` PVT:

```
set_wire_capacitance 2 foo -pvt {min typ}
set_wire_capacitance 2.2 foo -pvt max
```

- The following command backannotates a wire capacitance of 1.5, factoring that the total pin capacitance on net `foo` is 0.5:

```
set_wire_capacitance 2 foo -pvt {min max} -subtract_pin_caps
```

Related Information

`report_net`

`reset wire capacitance`

`set wire resistance`

set_wire_load

```
set_wire_load [-library library_name] [-pvt {min | typ | max}] [-hier]
     wireload_model list_of_instances
```

Specifies the wire-load model to be used from the technology library, and sets the wire-load model on the current instance, if the list of instances is omitted. Otherwise, the wire-load model is set on all specified instances.

Use wire-load models for estimating delays before the actual wire loads are backannotated. A technology library contains different wire-load models, previously computed based on the analysis of several designs differing in area. Use the `report_library` command to list all the wire-load models in a library. To set a wire load on individual ports, use the `set_port_wire_load` command.

Options and Arguments

`-hier`

Affects wire-load assertions in the `enclosed` wire-load mode. (See set_wire_load_mode on page 1274). By default (without the `-hier` option), in the `enclosed` wire load mode, the asserted wire-load model applies to only those nets for which the specified hierarchical instance is the lowest enclosing module.

If the `-hier` option is set, then for all nets fully contained within the hierarchical instance (or any of its lower level hierarchical instances), the asserted wire-load model is used, unless another such assertion is present on one of the lower level hierarchical instances.

In the `top` wire-load mode, a wire-load assertion on a hierarchical instance applies to all the nets fully enclosed within the hierarchical instance (unless there is another such wire load assertion present on one of the lower level hierarchical instances) meaning the `-hier` switch is always on.

The following describes how the wire load for a net is looked up. The search process starts with the lowest hierarchical instance completely enclosing the net and recursively traverses up the hierarchy chain, until it reaches (and includes) the top module.

Enclosed wire load mode

If a wire-load model is asserted on the lowest enclosing hierarchical instance (with or without the `-hier` option), that wire-load model is used. Otherwise, the search travels up the hierarchy looking for an asserted wire-load with the `-hier` option. If one is found, that wire load is used. Otherwise, area-based lookup is used.

Note: As the search traverses up the hierarchy chain, if a wire load without a `-hier` option is found, that assertion is ignored. Thus, the assertion for wire-load model without the `-hier` option applies only to the nets fully enclosed within that specific hierarchical instance (and not in any of its lower level hierarchical instances).

Top mode
In this mode, the `-hier` option on the wireload assertion is completely ignored. Again, start from the lowest enclosing hierarchical instance and travel up the hierarchy chain. As soon as a wire-load assertion is found on an instance (with or without the `-hier` option), that wire-load model is used. If no wire-load assertion is found, then the area-based lookup is used.

`-library` *`library_name`*
Specifies the technology library to use when searching for the wire load model.

`library_name`
Specifies the name of the technology library.

`list_of_instances`
Specifies the list of instances.

`-pvt {min | typ | max}`
Sets the wire-load model for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified wire-load model applies to all three PVT corners.

`wireload_model`
Specifies the name of the wire-load model in the library.

Example

```
set_wire_load B5X5 {I_block J_block}
```

Related Information

do_derive_context

report_library

set_current_module

set_global target_technology

set_port wire_load

set_wire_load_mode

See Specifying Wire Loads in the *Common Timing Engine (CTE) User Guide*.

set_wire_load_mode

```
set_wire_load_mode {top | enclosed}
```

Controls the determination of the wire-load model.

Options and Arguments

`enclosed`

Selects the wire-load model using an area lookup table for the module that is at the lowest level in the design hierarchy that contains the entire net.

`top`

Selects the wire-load model using the area lookup table of the top level module (as set by the `set_top_timing_module` command).
Default: `top`

Example

The following command sets the wire-load model to `enclosed`:

```
set_wire_load_mode enclosed
```

Related Information

`report_library`

`reset_wire_load_mode`

`set_current_module`

`set_global_target_technology`

`set_top_timing_module`

`set_wire_load`

See Specifying Wire Loads in the *Common Timing Engine (CTE) User Guide*.

set_wire_load_selection_table

```
set_wire_load_selection_table [-library library_name] [-pvt {min | typ | max}]
     [-min_area float] wireload_selection_table
```

Overrides the wire-load selection table in the library specified by the *library_name* argument. When *library_name* is the target technology library, the specified *wire_load_selection_table* is used in analysis.

Options and Arguments

`-library` *library_name*
: Specifies the name of the technology library to search for the wire-load selection table.
Default: The `target_technology` file.

`-min_area` *float*
: Selects the area wire-load model for all area values smaller than *float*.

`-pvt {min | typ | max}`
: Controls the choice of a wire-load selection table for different PVT values.
Default: Selects the table for all PVTs.

wireload_selection_table
: Specifies the name of the wire-load selection table.

Example

```
set_wire_load_selection_table TLM
```

Related Information

`report_library`

`reset_wire_load_selection_table`

`set_global target_technology`

`set_wire_load`

set_wire_resistance

```
set_wire_resistance [-pvt {min | typ | max}] float net_names
```

Overrides the wire resistance value used during timing analysis. This command is not written out by the `write_assertions` or `write_sdc` commands, but wire resistance annotations are saved in the `.adb` file.

Options and Arguments

`float`
: Specifies the resistance value

`net_names`
: Specifies the list of nets with the resistance

`-pvt {min | typ | max}`
: Sets the resistance for a particular PVT (process, voltage, temperature) corner. Choose one, two, or three PVT corners. If you choose more than one corner, enclose the list in curly braces ({}) and separate the values by spaces.
Default: The specified resistance applies to all three PVT corners.

Example

The following command sets the resistance for the nets `bus1` and `bus2` at `4.6`:

```
set_wire_resistance 4.6 {bus1 bus2}
```

Related Information

`report_net`

`reset_wire_resistance`

`set_wire_capacitance`

unload_dcl_rule

```
unload_dcl_rule
```

Removes a previously loaded binary Delay and Power Calculation Module (DPCM) from memory. The DPCM is an executable shared library that is linked to an application (such as BuildGates Synthesis) at runtime. DPCM is often abbreviated as DCM. After unloading the DCM, all timing calculations are performed without information from the DCM.

Example

```
unload_dcl_rule
```

Related Information

load_dcl_rule

write_assertions

```
write_assertions [-no_internal_arr_req] [-no_timing] [-no_tag] [file_name]
```

Writes out the assertions made on the design prior to timing analysis for future reference and use. The assertions for the module may have been asserted by the user or created by the `do_derive_context`, `do_time_budget`, or `do_push` commands.

The assertions for wire capacitance and wire resistance are not written out by the `write_assertions` command. Wire capacitance and wire resistance are stored in the ADB file.

Options and Arguments

`file_name`

Specifies the name of the file in which to write the assertions. If the *`file_name`* argument is not specified, assertions are written to `stdout`.

`-no_internal_arr_req`

Specifies the internal arrival times are not written to the file.

`-no_tag`

Removes all proprietary information (`set_begin_tag`, `-tag`, `-from_tag`, and `-info`) from the output file. The resulting assertions file is in a form that is easier for a script to translate to a third-party language.

Note: The `-no_tag` option works only for chip-level constraints. If you have completed time budgeting, then using the `write_assertions -no_tag` command for block level constraints does not work. Removing tags from constraints is not the solution either. This is similar to dropping constraints from the design. Therefore, you can only use budgeted block constraints in CTE.

`-no_timing`

Specifies the timing assertions, such as clock assertions, arrival and required time assertions are not written to the file. Only those assertions that are not related to timing checks, such as load, drive, wire-load model, operating conditions, and so on, are written out.

Example

The following command saves all the assertions, except timing assertions, in `my_assert_file`:

```
write_assertions -no_timing my_assert_file
```

Related Information

`do_derive_context`

`do_push`

`do_time_budget`

`remove_assertions`

write_constraints

```
write_constraints [-max_slack slack_value] [-max_paths num_paths]
     [-coverage_ratio float] [-min_constraint float] file_name
```

Writes the constraints of the current `top_timing_module` context to the named file. The file is written in a Standard Delay Format (SDF) for timing-driven place and route using path constraint constructs to cover all nets in the design.

The `write_constraints` command outputs constraints for use by place and route tools. Therefore, this command must provide a set of paths that cover the design. The paths included are only those whose slack is less than the value specified by the `-max_slack` option.

Options and Arguments

`-coverage_ratio` *`float`*
: Specifies a coverage ratio that is the number of arcs covered in the constraints divided by the total number of arcs in the design. Limit the number of paths by specifying a `coverage_ratio`.

`file_name`
: Specifies the name of the resulting SDF file.

`-max_paths` *`num_paths`*
: Specifies an optional limit to the number of paths produced.
Default: `1000`

`-max_slack` *`slack_value`*
: Limits the number of paths by specifying the maximum slack of the net for which to write a path.
Default: `+infinity`

`-min_constraint` *`float`*
: Relaxes the written path constraints by the specified amount. Otherwise, if the paths in the design are overconstrained, the resulting constraint file is also similarly overconstrained.

Examples

```
set_top_timing_module top
set_current_module top
write_constraints constraints.sdf
```

See Example SDF Constraint File in the *Common Timing Engine (CTE) User Guide*.

Related Information

`write sdf`

write_gcf_assertions

```
write_gcf_assertions [-version {1.3 | 1.4}] [file_name]
     [-tlf_pvt { min | typ | max }]
```

Automatically translates the CTE Tcl constraints into Cadence General Constraint Format (GCF). When using timing analysis, SE-DSM, or SE-Ultra for ASIC design, there are two timing analysis engines that interact between the synthesis and the place-and-route phase. The frontend of the flow uses the Common Timing Engine (CTE) and the backend uses Pearl® from the place-and-route tool suite. Pearl also reads the TLF libraries to calculate the delays in the ASIC during floorplanning and place & route. Pearl utilizes the GCF constraints file in conjunction with the TLF libraries or the Verilog gate-level netlist to create the file to drive all the floorplanning and place-and-route phases of Cadence Design Planner and Silicon Ensemble. This combination yields a very tight correlation between the timing reports generated by Pearl and those generated by Cadence timing analysis. This combination also speeds up the conversion process to final silicon. Other flows like the SE-PKS flow use CTE throughout and do not require any constraint translation to GCF.

Options and Arguments

file_name
: Specifies the file name. If it is not specified, the default is the standard output.

-tlf_pvt {min | typ | max}
: Writes out the min, typ, and max TLF library. The typ library defaults (if not specified) to the first library read by the read_tlf command.
Default: Writes out the typ library

-version {1.3 | 1.4}
: Specifies the version of GCF to write out. The version shows in the header of the GCF file and is used by the tools that read GCF. Version 1.4 has more constructs for greater inter-tool compatibility.
Default: 1.4

Example

The following command writes out the file cpu.gcf:

```
write_gcf_assertions cpu.gcf
```

Related Information

`read_tlf`

`read_verilog`

`read_vhdl`

`report_timing`

`write_verilog`

See Generating GCF Files in the *Common Timing Engine (CTE) User Guide*.

write_library_assertions

```
write_library_assertions [-library list_of_library_names] file_name
```

Writes out assertions you have made to the named libraries using the `set_tech_info` command.

Options and Arguments

`file_name`
: Specifies the name of the output file. Required argument.

`-library` *`list_of_library_names`*
: Reports data for the named libraries.
Default: All libraries specified with the `set_global target_technology` global.

Example

- The following command writes out all assertions previously defined by the `set_tech_info` command:

  ```
  write_library_assertions assert.txt
  ```

  The contents of `assert.txt` shows that two default values in the `cb` library have been overwritten:

  ```
  set_tech_info -library cb -default_max_transition 5.000000 -pvt typ
  set_tech_info -library cb -default_max_capacitance 2.500000 -pvt min
  ```

Related Information

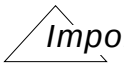
`get_tech_info`

`reset_tech_info`

`set_tech_info`

write_rspf

```
write_rspf
     OBSOLETE: Use write_spf instead.
```



The `write_rspf` command is no longer supported, use `write_spf` instead.

Related Information

`write_spf`

write_sdc

```
write_sdc [-version {sdc1.1 | sdc1.2 | sdc1.3}] [-include_wire_rc]
     [-leading_edge_falling clock_name] [-comment_non_integer_multicycle]
     [-exclude_ignored_exceptions] [-format {sdc|pttcl}] file_name
```

Uses equivalent timing constraints to perform RTL synthesis and gate level timing analysis in PKS and PrimeTime respectively. See *SDC Constraint Support Guide* for more details about this command.

Note: The `write_sdc` command does not support design budgeting constraints.

To write out the `set_operating_conditions` command with the `-analysis_type` option, use the `-format` option with the `write_sdc` command to write out the PrimeTime tcl format.

Options and Arguments

`-comment_non_integer_multicycle`
: By default, `write_sdc` rounds up the setup or hold multiplier value. In CTE, the value is of type `float` (for instance, you can assign a cycle addition of 2.3). Synopsys supports only integer values. By default, `write_sdc` rounds the float value to the nearest integer. Use the following to override this default:

  `write_sdc -comment_non_integer_multicycle`

`-exclude_ignored_exceptions`
: By default, `write_sdc` writes out all ignored exceptions. Use this option if you do not want these exceptions to be written out.

  Note: Using this option may increase the `write_sdc` runtime.

`file_name`
: Specifies the name of the Synopsys design constraint (SDC) output file.

`-format {sdc|pttcl}`
: Writes output in the PrimeTime tcl format.

  Use the SDC format to write out only the constraints and options that are standardized in the SDC specifications.

Use the `pttcl` format to write out a non-standard option. Currently, only the non-standard options of the `set_operating_condtions` constraint is written out.

`-include_wire_rc`

Allows the translation of the CTE `set_wire_capacitance` and `set_wire_resistance` commands to the Synopsys `set_load` and `set_resistance` commands. By default, this translation is disabled, because translating these commands for large designs can take a long time.

`-leading_edge_falling` *`clock_name`*

By default, `write_sdc` considers that the leading edge of a virtual clock is rising. Modify this behavior using this option.

`-version`

By default, `write_sdc` writes out the translations in sdc1.3 format. Use this option if you want output in the SDC1.1 or SDC1.2 format.

Related Information

`read_dc_script`

`sdc_write_unambiguous_names`

write_sdf

```
write_sdf [-version  { 2.1 | 3.0 }] [-precision non_neg_integer]
     [-scale float] [-delimiter char] [ {-early | -late} ]
     [-transform_out_to_out_arcs] [-celltiming {all | none | nochecks}]
     [-interconn {all | none | noport | noinport | nooutport}]
     [-edges {edged | library | noedge | check_edge}] [-remashold] [-splitrecrem]
     [-splitsetuphold] [-condelse] [-nonegchecks] [-force_calculation]
     [file_name] [-compute_clock_network_delays]
```

Writes delays to a Standard Delay Format (SDF) file.
Default: The `write_sdf` command uses the delays already cached by the system and writes only values of the triplets specified by the `pvt_early_path` and `pvt_late_path` globals. Using the `-force_calculation` option calculates each PVT with a slew depth of 2.

By default, this command writes 0 delays for the clock network when the clock propagation mode is ideal.

Note: The `-compute_clock_network_delays` option only works with the `-force_calculation` option.

For supported SDF constructs, see Supported SDF Constructs in the *Common Timing Engine (CTE) User Guide*.

Options and Arguments

`-celltiming {all | none | nochecks}`
Specifies which cell delays and timing checks to write out.
Default: `all`

`all`
Writes all cell delays and timing checks to the SDF file. This is the default.

`nochecks`
Disables the writing of timing checks only.

`none`
Prevents cell delays and timing checks from being written into the SDF file.

`-compute_clock_network_delays`
Use the `-compute_clock_network_delays` option with the `-force_calculation` option to write the actual delays for

clock network elements (gates and interconnections) when the clock propagation mode is ideal.

`-condelse`

Writes `CONDELSE` constructs with the default value when a `COND` construct is written.

Note: Use this argument with the `-version 3.0` option (see page 1292).

`-delimiter` *`char`*

Specifies the hierarchy divider character to use in the SDF output file. This option sets the hierarchy divider of only the `write_sdf` command. If omitted, the default `ac_shell` hierarchy divider character (`/`) is used as the `write_sdf` default.

`-early | -late`

Specifies an early mode or late mode analysis for computing SDF delays.
Default: `-late`

Using these options only affects the input slew used to compute delays when the `-force_calculation` option is specified, or when the `write_sdf_force_calculation` global is set to `true`.

If you specify the `-early` option, the slews associated with the early path are used to compute timing arc delays. If you specify the `-late` option, the slews associated with the late path are used to compute timing arc delays. These options are ignored if the `-force_calculation` option is not used. For more information, see Writing Out SDF Delay Files in the *Common Timing Engine (CTE) User Guide*.

`-edges {edged | library | noedge |check_edge}`

Specifies the edges values. Refer to Table 7-9 on page 1292 for full qualifications of the possible values for the edges option.
Default: `edged`

`check_edge`
Keeps edge specifiers on timing check arcs but does not add edge specifiers on combinational arcs.

edged
Writes the SDF `IOPATH` and timing checks with appropriate edges.

library
Assumes that the *`sdf_edges`* attributes are specified in the `.lib` file or the library. The default *`sdf_edges`* value of a timing arc is `noedge`.

noedge
Defined in Table 7-9 on page 1292.

`file_name`

Specifies the name of the SDF output file. If omitted, this argument defaults to `stdout`.

`-force_calculation`

Forces the recomputation of delays during `write_sdf` and writes triplets (`min:typ:max`). This was the default behavior prior to 4.0.8. Use this option for backward compatibility with 4.0, but be aware that this option significantly increases the run time of `write_sdf`.

See Writing Out SDF Delay Files in the *Common Timing Engine (CTE) User Guide* for more information.

`-interconn {all | none | noport | noinport | nooutport}`
Specifies which interconnect delays to write.
Default: `all`

all
Writes all `INTERCONN` delays into the SDF file. This is the default if the `-interconn` option is not specified.

none
No `INTERCONN` delays are written into the SDF file.

noport
No top level port `INTERCONN` delays are written into the SDF file.

noinport
Disables only the input port `INTERCONN` delays.

nooutport
Disables only the output port INTERCONN delays.

-nonegchecks
Converts all negative timing check values to 0.0.

-precision *non_neg_integer*
Controls the number of digits appearing after the decimal point in the output SDF file. The default precision is the value of the report_precision.

-remashold
Converts the SDF v2.1 unsupported REMOVAL (and the REMOVAL portion of the RECREM constructs) to HOLD checks when used with the -version 2.1 option.
Default: Not to write out the unsupported checks.

Converts the REMOVAL checks to HOLD checks when combined with the -version 3.0 option.The RECREM checks are split into a RECOVERY check and a HOLD check. This option automatically splits the RECREM checks.
Default: To maintain the version 3.0 checks.

-scale *float*
Specifies a multiplier to the delay values used in the SDF file. This option does not change the timescale setting of the SDF file. The delay values are scaled, but the units are the same.
Default: 1

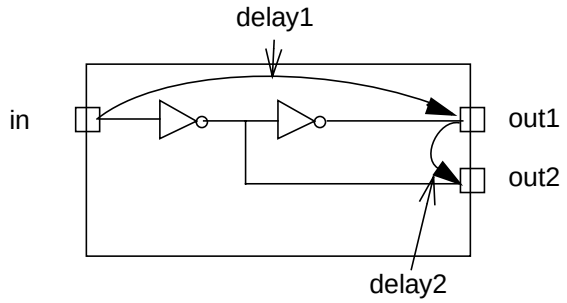
-splitrecrem
Splits the RECREM checks into a RECOVERY check and a REMOVAL check.
Default: To write the combined RECREM check.

Note: Use this option only with the version 3.0 option (see page 1292).

-splitsetuphold
Splits the SETUPHOLD timing checks into separate SETUP and HOLD checks.
Default: To write the combined SETUPHOLD checks.

`-transform_out_to_out_arcs`

Writes out all output-to-output timing arcs as input-to-output arcs. Input-to-output arc delays are computed by adding the delay from the input pin to one of the output pins plus the delay from that output pin to the other output pin. For instance, in the following figure, the delay between `in` and `out2` would be the sum of `delay1` and `delay2`.



`-version {2.1 | 3.0}`

Specifies whether to generate the SDF V2.1 or V3.0. The default SDF version is 3.0. Used in combination with the `remashold` option.

Table 7-9 Definition of `-edges` Option Values

	edged Value	library Value	noedge Value
IOPATH from a unate input pin to output pin	No edge specifier	No edge specifier	No edge specifier
IOPATH from a non-unate input pin to output pin	Edge specifier at the input pin	Edge specifier at the input pin when sdf_edges is either “both_edges” or “start_edge”	No edge specifier When merging edged paths, the maximum values are used for MAX and TYP fields; minimum values are used for MIN fields.

Table 7-9 Definition of `-edges` Option Values, *continued*

	edged Value	library Value	noedge Value
IOPATH of a clock/enable pin to data out pin	Edge specifier at the input pin	Edge specifier at the input pin when sdf_edges is either “both_edges” or “start_edge”	No edge specifier
Timing Checks	Edge specifiers match the check arcs specified in the timing library	Overridden by value of the `sdf_edges` attribute For `both_edges`, the timing checks will contain edge specifier on both starting and ending pins For `start_edge`, the starting pin (for HOLD type check) or ending pin (for SETUP type check) is edge-specified For `end_edge`, the ending pin (for HOLD style check) or the starting pin (for SETUP type check) is edge-specified The timing checks are duplicated or combined appropriately. The `start_edge` and `end_edge` are interpreted as first and second events, respectively, of the Verilog-XL’s $hold() and $setup system tasks	Only the edge specifiers of the reference pin are written

Examples

- The following command writes out an SDF 2.1 compliant output file named `my.sdf` with 4 digits after the decimal point:

```
write_sdf -version 2.1 -precision 4 my.sdf
```

- The following example SDF files compare the output of `-edges edged` to the output of `-edges noedge`.

  Resultant SDF with `-edges` set to `edged`:

```
(HOLD (posedge D) (posedge CK) (-0.14:-0.14:-0.14))
(HOLD (negedge D) (posedge CK) (-0.06:-0.06:-0.06))
```

```
(SETUP (posedge D) (posedge CK) (0.22:0.22:0.22))
(SETUP (negedge D) (posedge CK) (0.41:0.41:0.41))
```

With `-edges` set to `noedge` only the edge specifiers of the clock pin are written.

```
(HOLD D  (posedge CK) (-0.14:-0.06:-0.06))
(SETUP D  (posedge CK) (0.22:0.41:0.41))
```

Related Information

`write_pdef`

`write_verilog`

See Writing Out SDF Delay Files in the *Common Timing Engine (CTE) User Guide*.

For supported SDF constructs, see Supported SDF Constructs.

write_spf

```
write_spf [-add_pks_rspf] [-pvt {min | typ | max}] [-no_instance_section]
     file_name
```

Writes out the reduced parasitics model to the named file. Reduced parasitics models can come from reduced or detailed parasitics previously loaded using the `read_spf` or `read_spef` commands, or from parasitics extracted from a PKS Steiner tree.

Note: This command replaces the obsolete `write_rspf` command.

Options and Arguments

`-add_pks_rspf`
: Writes out a reduced model from PKS Steiner tree into the SPF file. Writes out reduced models (Pi-Elmore or Ctotal-Elmore) regardless of whether they were created from global or final routes.

`file_name`
: Specifies the name of the file to which the reduced SPF is written.

`-no_instance_section`
: Tells the system not to write the instance section of the SPF file. Some tools require the instance section, you should check before using this option.

`-pvt {min | typ | max}`
: Specifies which PVT value to write. SPF format supports only one value (no pairs or triplets). Choose `min`, `typ`, or `max`.

Example

```
write_spf -pvt min min.rspf
```

Related Information

`read_spf`

`read_spef`

write_timing_windows

`write_timing_windows [-sstorm] [-pin] output_file`

Generates a Timing Window File (TWF) used by crosstalk analysis tools such as Celtic™ or delay calculator tools such as SignalStorm™. The TWF file mainly contains the earliest and the latest possible arrival times that a signal may arrive on a net or a pin.

For an example of the Timing Window File format see the *CeltIC manual*.

Note: SignalStorm requires a pin-based timing windows file. Using the `-sstorm` option, PKS writes out pin-based timing windows for SignalStorm.

Options and Arguments

filename
: Specifies the TWF filename.

-pin:
: Generates a TWF for each pin instead of each net.
Default: Generates a TWF for the CeltIC crosstalk analysis tool, which contains timing windows on each net.

Note: The -pin option is ignored when writing out a timing windows file for SignalStorm.

-sstorm
: Generates a Timing Windows File (TWF) for the SignalStorm delay calculator tool.

1

Temporary Commands

Important

This chapter contains descriptions for temporary commands, which usually begin with an underscore (_), intended for use only in the situations described. These commands are often untested, unsupported, and intended only as temporary workarounds. Also, some of these functions are considered high risk, in that they may cause serious problems with your design if not used with extreme caution.

If you are using a hidden command in your flow and would like it documented, please let us know by using the *CDSDoc Feedback* button or by e-mailing *synthesis_hidden@cadence.com*.

This chapter contains the following sections:

Timing Analysis on page 1310Datapath Synthesis

This section describes the hidden commands associated with Datapath Synthesis in BuildGates Synthesis and Cadence PKS.

write_datapath_pdef

```
write_datapath_pdef [-hier] [-module <module_name>]
```

Provides you with access to the hierarchical relative placement for DP partitions. The command outputs the hierarchical relative placement (PDEF 3.0) for the DP partitions in the current module or the module specified. If the `-hier` option is specified, PDEF is output for all instances of DP partitions contained in the hierarchy below the current module. You can then use this PDEF with other third party tools or use it to manually floorplan a design with multiple DP partitions instead of using the automated flow.

Options and Arguments

`-hier`

Outputs all instances of DP partitions contained in the hierarchy below the current module in the PDEF.

`-module <module_name>`

Outputs only the module specified in the PDEF.

Turning on the Prototype DP GUI

We are in the process of developing a graphical user interface for datapath synthesis. To see and try the prototype version available in this release, launch BuildGates Extreme Synthesis or Cadence PKS as follows:

```
bgx_shell -gui -set show_dp_tab=1
pks_shell -gui -set show_dp_tab=1
```

Note: Because this is a prototype version, functionality is not complete or stable.

Optimization

This section describes the hidden commands associated with optimization used in BuildGates Synthesis and Cadence PKS.

get_cell_area

`get_cell_area` *`cell_id`*

Returns the area of the specified cell.

Options and Arguments

`cell_id`

Specifies the object identifier associated with a cell.

Related Information

`set_cell_area`

Examples

```
>get_cell_area 60979
16.0
```

get_cell_pin_load

`get_cell_pin_load [-library` *`library_name`*`] -cell` *`cell_name`* `-pin` *`pin_name`*

Returns the capacitive load at the specified pin of a cell in the library. The units of the load are the same as the units used in the library.

Options and Arguments

`library` *`library_name`*
: Specifies the name of the library.

`-cell` *`cell_name`*
: Specifies the name of the cell. If this option is omitted the pin load for all cells in the named libraries are returned.

`-pin` *`pin_name`*
: Specifies the name of the pin. If this option is omitted the pin load for all pins on the named cells in the named libraries are returned.

Related Information

`set_cell_pin_load`

Examples

```
>get_cell_pin_load -cell IV -pin z
2.0
```

set_cell_area

`set_cell_area` *`cell_id float`*

Sets the area of the specified cell.

Options and Arguments

`cell_id`

Specifies the object identifier associated with a cell.

`float`

Specifies the value of the cell area.

Examples

```
set_cell_area 60979 16.0
```

set_cell_pin_load

`set_cell_pin_load [-library` *`library_name`*`] -cell` *`cell_name`* `-pin` *`pin_name float`*

Sets the value of the capacitive load at the specified pin of a cell in the library. The units of the load are the same as the units used in the library.

Options and Arguments

`float`
: Specifies the load value to be set on the pin.

`library` *`library_name`*
: Specifies the name of the library where the cell can be found. By default, the target library is searched for the named cell.

`-cell` *`cell_name`*
: Specifies the name of the cell for which pin load is applied.

`-pin` *`pin_name`*
: Specifies the name of the pin.

Related Information

`get_cell_pin_load`

PKS

This section describes hidden commands associated with Cadence PKS.

_generate_lut_from_routes

```
_generate_lut_from_routes str [-two_luts]
```

After the design has been routed (global or detail), issuing this command completes a statistical analysis of the real routes and vias, including route layer assignment and length and number of vias used. From the statistics, one or two luts will be generated. You can then restart the run using this lut for steiner RC calculation. See the `set_layer_usages_table` and the `set_net_physical_attribute` commands for more information. The purpose of this command is to bring better pre- and post- route timing correlation without requiring you to know the lut.

Options and Arguments

str

Represents the file name of the generated lut. This is a required option.

`-two_luts`

Generates two luts, one for signal nets and one for clock nets. The file name of the signal net lut is specified as *str*`_signal` while the clock lut is specified as *str*`_clock`. If this option is not specified, which is the default, one lut will be generated, for both signal and clock nets.

Examples

- The following command creates a file named `gen_lut` in the current directory:

  ```
  _generate_lut_from_routes gen_lut
  ```

- The following command creates two files in the current directory, one is `gen_lut_signal` and the other is `gen_lut_clock`.

  ```
  _generate_lut_from_routes gen_lut -two_luts
  ```

report_steiner_route

`report_steiner_route` *`net_name`*

Output to stdout:

Outputs a list of all vertices (pins are named with a <cB> for boundary pins, < I> for instance pins), and a list of edges connecting the vertices.

Examples

```
pks_shell[10]>report_steiner {tdigit_flag}
Net tdigit_flag ......
NET: tdigit_flag
CbPIN:Test7/tdigit_flag LOC:92000,692000
IPIN:DIGIT_REG_INST/flag_out_reg/Q LOC:319600,114500
Total Lumped C: 0.106609  Total Lumped R: 0.303157  Source of parasitics: Steiner
tree

    >>>>>>> Steiner Route Summary
v1: 319600,692000 (C=0.0533045)
v2: 92000,692000 (C=0.0152007) (<cB>tdigit_flag)
v3: 319600,114500 (C=0.0381038) (< I>DIGIT_REG_INST/flag_out_reg/Q)
e1: V 1: v3->v1 (R=0.220976) (L=577500)
e2: H 2: v2->v1 (R=0.0821811) (L=227600)
Summary for Net tdigit_flag :
Number Of Pins = 2   Number of StPoints = 1
Number of Edges = 2
StRoute Length = 805100  BBOX = <92000,114500,319600,692000> HP Length = 805100
Total Lumped C: 0.106609
Total Lumped R: 0.303157
```

Timing Analysis

This section describes the hidden commands associated with timing analysis in BuildGates Synthesis and Cadence PKS.

_convert_delays_to_assertions

```
-convert_delays_to_assertions
```

Forces the delay system to convert all delays into sdf assertions. This is needed when you want to load in an incremental sdf.

_get_cell_area

`get_cell_area`

Gets the cell area in the timing library.

Using these commands has several limitations:

- By default, cell areas (as in `report_area`) are taken from physical cells, not the timing library area cell data. To use the area reported by `get_cell_area` or set by `set_cell_area`, set the global `use_lef_area` to `false`.
- The actual meaning of the timing library cell area is somewhat vague. The `read_tlf` command will either read the gate count (if present) or the transistor count (if present) as a gate area if the area property is missing.

If the timing library file is incorrect, it is usually better to correct the file rather than using the `set_cell_area` command to correct the area after loading.

_set_cell_area

`set_cell_area`

Sets the cell area in the timing library.

_write_design_timing

`_write_design_timing` *`outfile`*

Writes out names in the DEF naming style. This is the file format used by Nanoroute.

The precision follows the `report_precision` global variable setup. By default the precision is 2.

The following shows the file format:

```
CAPACITANCE_UNIT <multi> <unit>
TIME_UNIT <multi> <unit>

PORT <boundary_pin_name>
    CAP      <cap>           #  input pin -> pin cap
                             #  output pin -> pin cap + total wire cap
    SLACK    <late_rise> <late_fall> <early_rise> <early_fall>
             #  worst timing slack of this pin
    SLEW     <late_rise> <late_fall> <early_rise> <early_fall>
             #  transition time of this pin
    DELAY    <late_rise> <late_fall> <early_rise> <early_fall>
             #  input pin -> inc. delay from driver to this pin
             #  output pin -> max delay of the timing arc which end at this pin
    ARRIVAL  <late_rise> <late_fall> <early_rise> <early_fall>
             #  arrival time of this pin, if it's available
    REQUIRED <late_rise> <late_fall> <early_rise> <early_fall>
             #  required time of this pin, if it's available
;            # ; at end of PORT

PORT <boundary_pin_name>
         :
         :
;            # ; at end of PORT


INST <inst_name>
  PIN  <pin_name>
    CAP      <cap>           #  input pin -> pin cap
                             #  output pin -> pin cap + total wire cap
    SLACK    <late_rise> <late_fall> <early_rise> <early_fall>
             #  worst timing slack of this pin
    SLEW     <late_rise> <late_fall> <early_rise> <early_fall>
```

```
             #  transition time of this pin
    DELAY    <late_rise> <late_fall> <early_rise> <early_fall>
             #  input pin -> inc. delay from driver to this pin
             #  output pin -> max delay of the timing arc which end at this pin
    ARRIVAL  <late_rise> <late_fall> <early_rise> <early_fall>
             #  arrival time of this pin, if it's available
    REQUIRED <late_rise> <late_fall> <early_rise> <early_fall>
             #  required time of this pin, if it's available
  PIN <pin_name>
         :
         :
;            # ; at end of INST

INST <inst_name>
         :
         :
;
```

_write_timing_windows_clock_arrival

```
_write_timing_windows_clock_arrival {true|false}
```

Set to false by default, so that a clock arrival time assertion set using `set_clock_arrival_time` command will not be listed in the timing windows file. Setting this global to `false` will also improve runtime.

_write_timing_windows_fast_mode

```
_write_timing_windows_fast_mode {true|false}
```

Set to `true` by default, so that the `write_timing_windows` command will not create hash tables to sort out the timing windows per clock phase. You will see significant runtime improvement when this global is set to `true`.

When the global is set to `false`, the `write_timing_windows` command behaves in the old mode where it sorts the timing windows per clock phase and takes longer time.

_write_timing_windows_slack_info

Set to `true` by default because CeltIC™ uses the slack information for better accuracy. However, setting this global to `false` improves runtime.

Index

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

P

Q

R

S

U

V

W