Digital Systems

Introduction

R. Pacalet

Telecom Paris
Institut Mines-Telecom

August 20, 2024

Outline

Outline

DigitalSystems: goals

- Digital integrated systems overview
- Introduction to design methods
- Discover CAD tools
  - What they can do
  - Limitations
  - Cost
  - Future evolutions
- Allow you to
  - Join designers' team
  - Communicate with "hardware guys"
  - Join CAD tool vendor

DigitalSystems: resources

- 42 hours (lectures, labs, team work, project)
- 42 hours personal work
- PCs (GNU/Linux) equiped with:
  - CU-Boulder VIS tools: formal verification (or Mentor Graphics 0-in)
  - Mentor Graphics CAD tools: simulation, FPGA synthesis
  - Synopsys CAD tools: ASIC synthesis
  - Xilinx Zynq based Zybo by Digilent
  - Xilinx CAD tools: system level design, synthesis
- A WEB site: `http://soc.eurecom.fr/DS/`

DigitalSystems: organization

- Some theory (very practical): $2 \times 3 + 6 \times 1.5 = 15$ hours
  - Introduction, digital hardwared, Systems on Chip... (1×3 hours)
  - Validation, simulation, formal verification... (2×1.5 hours)
  - VHDL hardware modeling language ($1 \times 3 + 3 \times 1.5$ hours)
  - Team work (1×1.5 hours)
- A lot of practice: $4 \times 1.5 + 7 \times 3 = 27$ hours
  - Simulation-based and formal verification (2×1.5 hours)
  - Design, simulation, synthesis: "simple operator" (2×1.5 hours)
  - Project: specifications, digital hardware design, software design, integration, validation... (7×3 hours)

DigitalSystems: exam

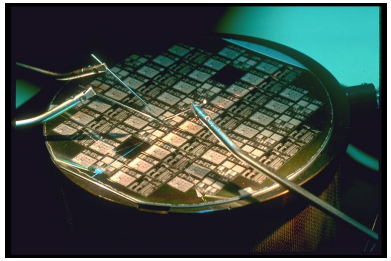
- A 2 hours exam
  - 75% of the final grade
  - Questions about lectures
  - Exercices, problems
  - All documents allowed
- Lab reports
  - 25% of the final grade
  - Understanding and results you got

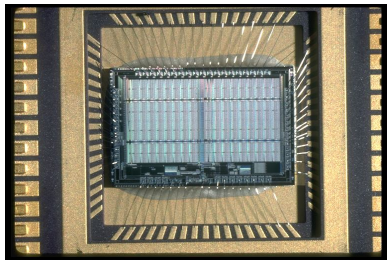
Outline

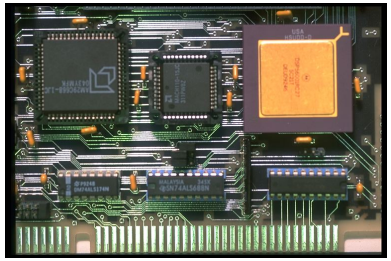
Outline

Integrated Circuits (ICs)

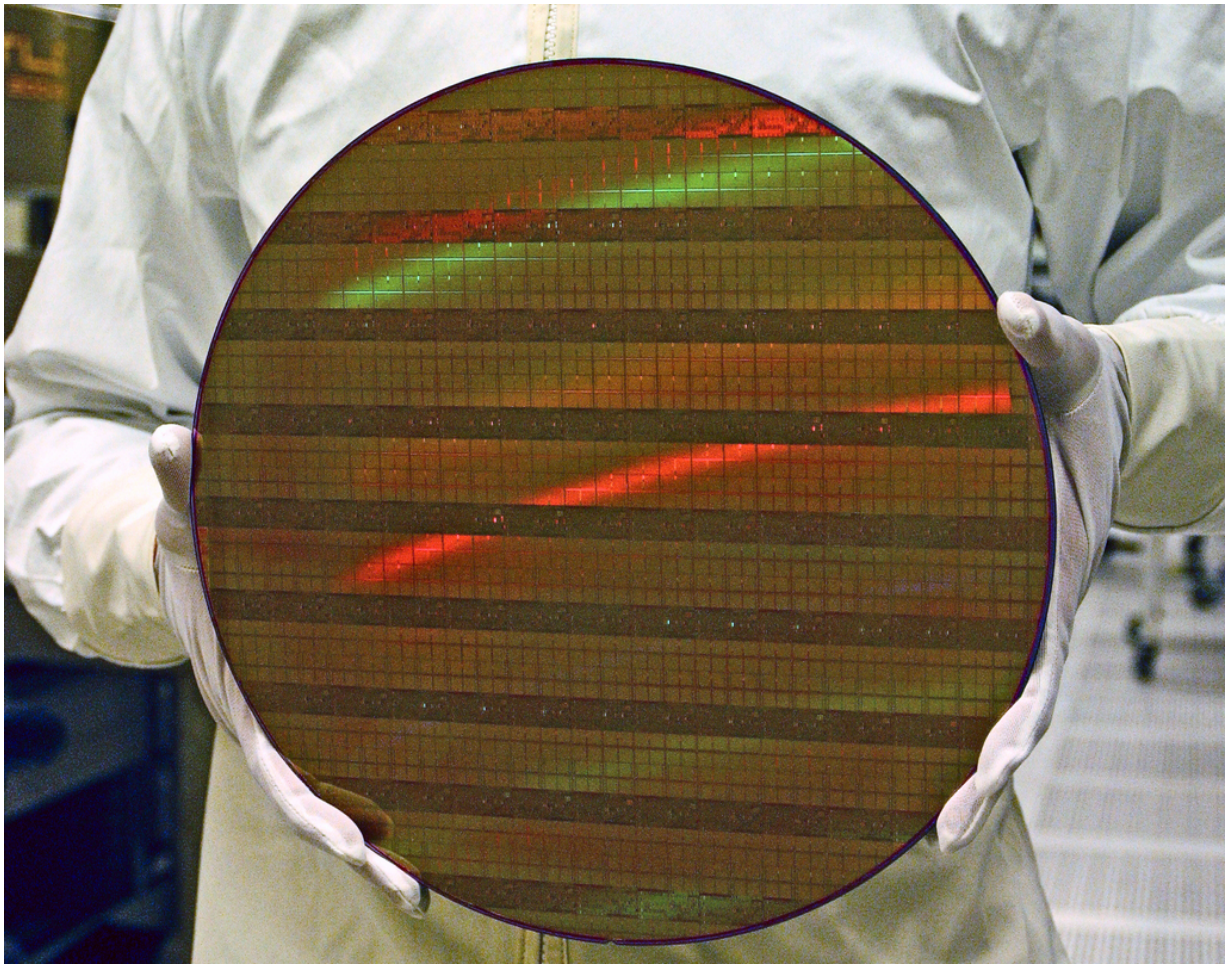
- Packages usually made of plastic or ceramic
- Silicon area: mm^2s to cm^2s; also measured in transistors or NAND2 gates equivalent
- Manufacturing process characterized by transistor's minimum channel length (e.g. 22 nm) or "lambda" (half channel length, e.g. 11 nm)
- 1970: bipolar transistors, a few dozens of gates (1971 Intel's 4004, 2250 tr., 10000 nm)
- 2015: CMOS, 14 nm, 3D, multi-billion transistors
- Example: Nvidia's Kepler GK110 GPU 7.1×10^9, 28 nm, 561 mm^2 (2012)





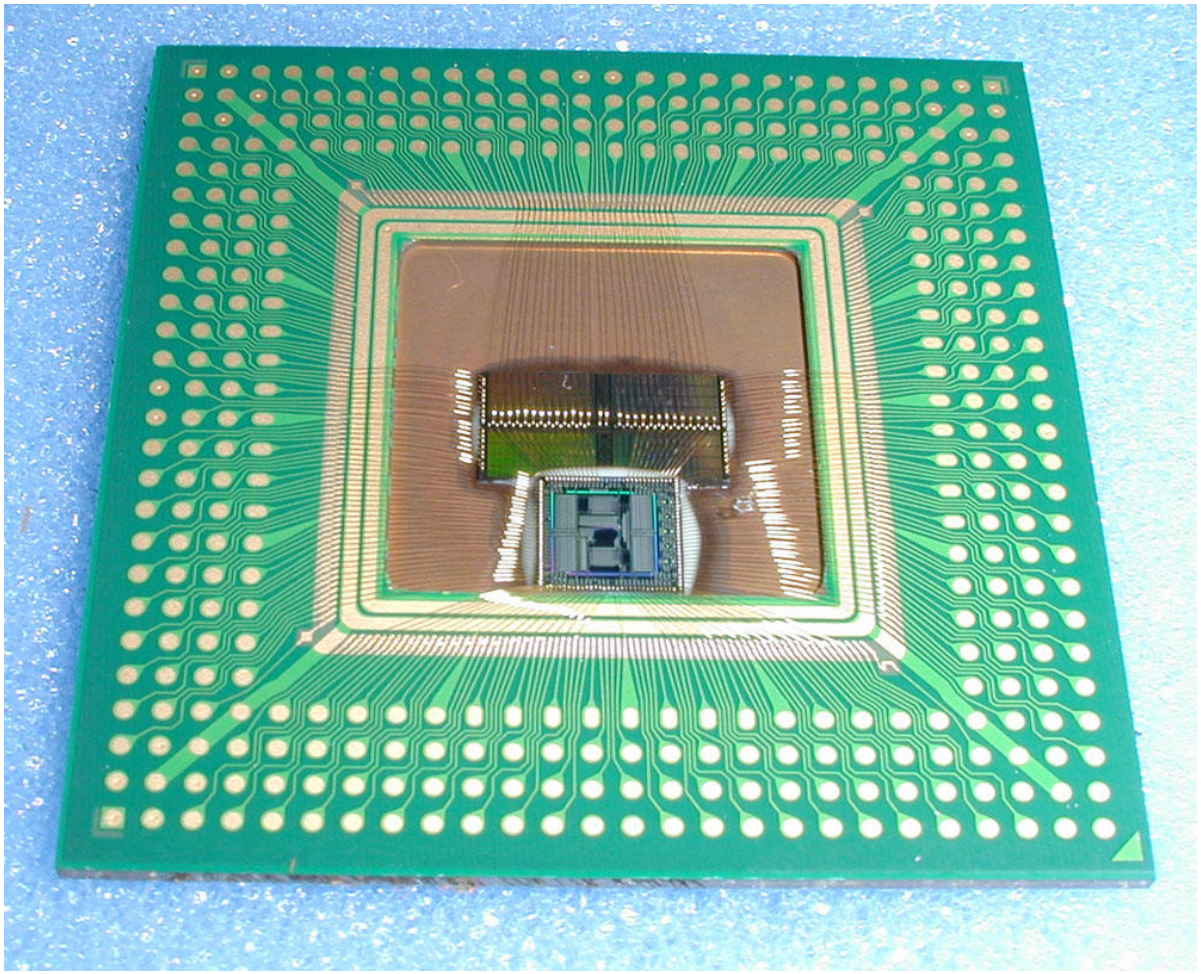


Wafer



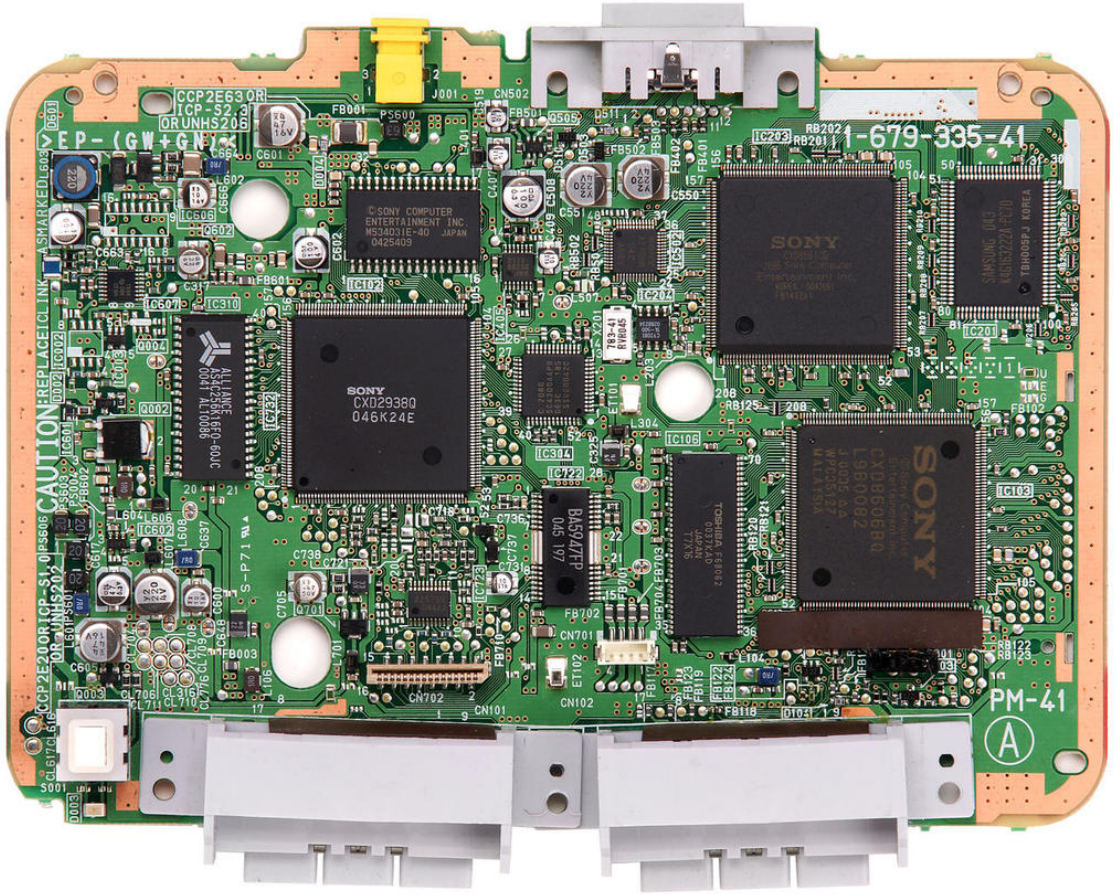
Source: Moe Adham (Intel 300mm wafer)

Dies in package



Source: Emulation Technology Inc. (multi-die BGA package)

Printed circuit board

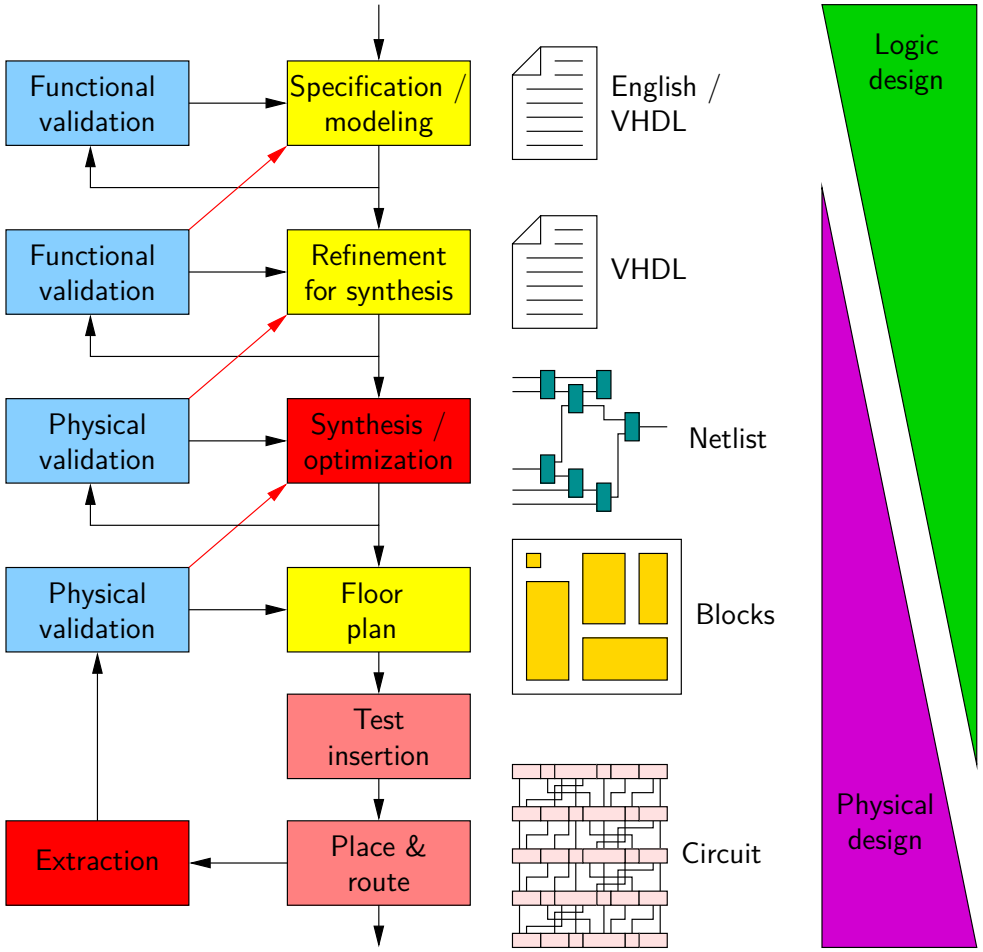


Source: Evan-Amos (Sony PSone motherboard)

Types of ICs

- Full custom
- Standard cells
- Gate array
- Programmable logic devices
  - EPLD
  - FPGA

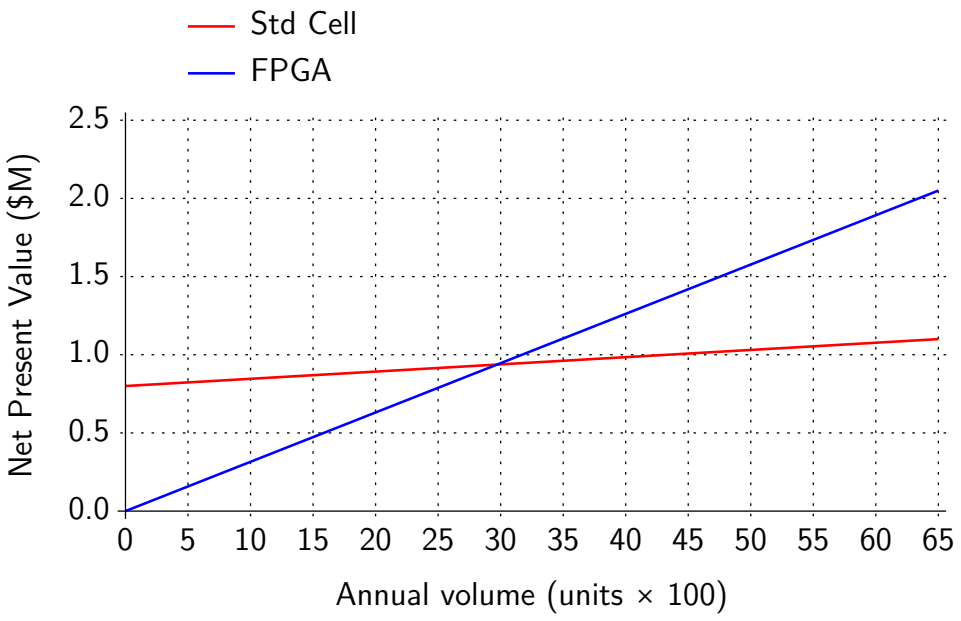
The Std Cells IC design flow: global view



Economics of ICs

Example of a 250 k-gates product, 200 MHz, 250 pins BGA package, 5 years life cycle, 8.5% discount rate.
(source: Arun Kottolli, Open-Silicon - EDN, 3/16/2006)

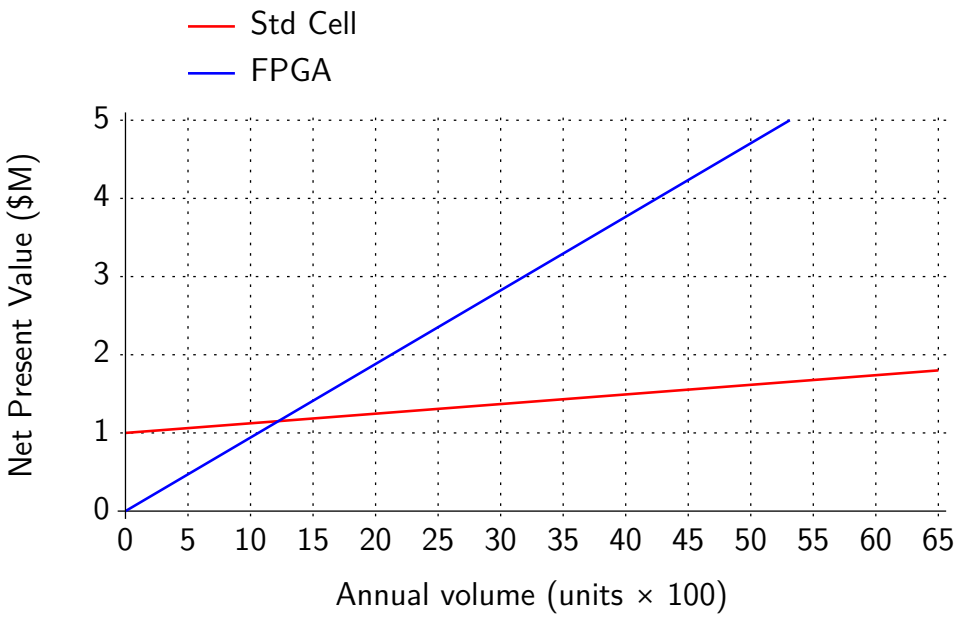
Type	NRE	Unit
FPGA	$M 0	$80
Std cell	$M 0.8	$12



Economics of ICs

Example of a 5 M-gates product, 3 Mbits of internal memory, one high-speed SERDES interface, 5 years life cycle, 8.5% discount rate. (source: Arun Kottolli, Open-Silicon - EDN, 3/16/2006)

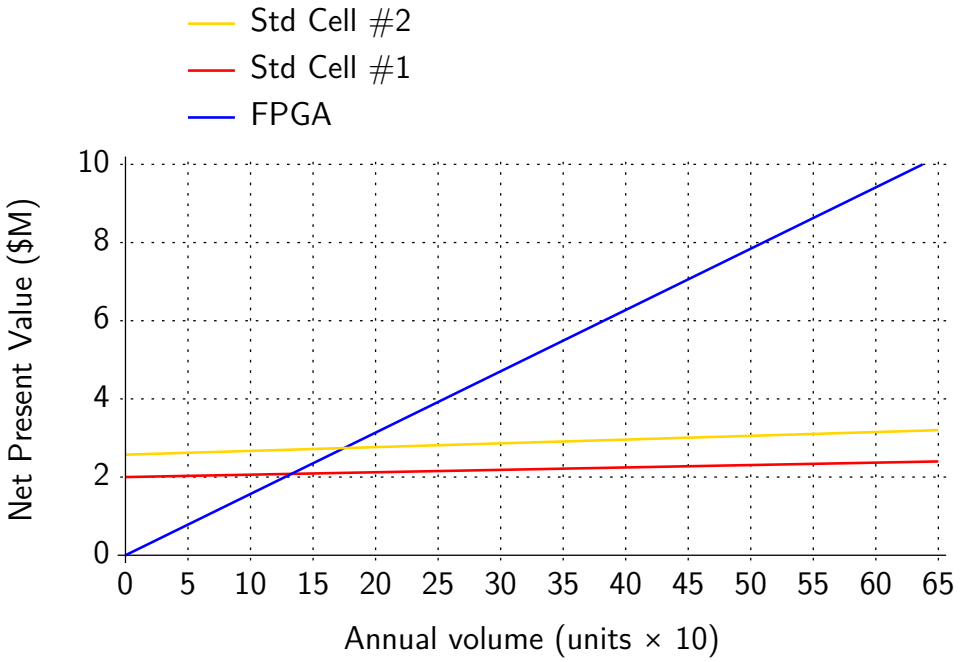
Type	NRE	Unit
FPGA	$M 0	$240
Std cell	$M 1.0	$32



Economics of ICs

Example of a 12 to 20 M-gates product, 10 to 20 Mbits of internal memory, multiple SERDES interfaces, a high-speed DDR interface, 5 years life cycle, 8.5% discount rate. Does not fit in a single FPGA.
(source: Arun Kottolli, Open-Silicon - EDN, 3/16/2006)

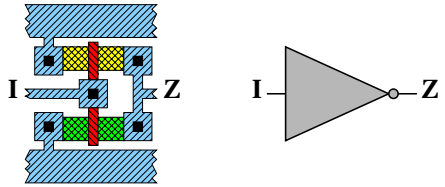
Type	NRE	Unit
FPGA	$M 0	$4000
Std cell #1	$M 2	$150
Std cell #2	$M 2.5	$320



Outline

Standard cell libraries

- A key factor
- Cell model = several views:
  - Layout
  - Schematic icon
  - Functional model
  - Propagation delay
  - Simulation (VHDL/Verilog)
  - *Test information*
  - *Transistor level netlist*
- Some views are private
- Foundry choice (technology, standard cell libraries) essential



Inverter models

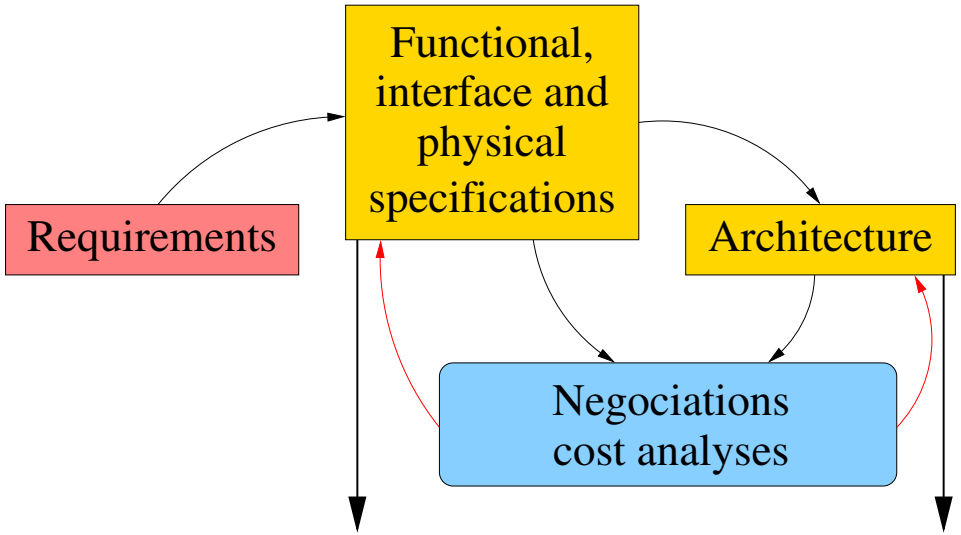
$$Z = \bar{I}$$

$$tp_{lh} = tp0_{lh} + \Delta tp_{lh} \times C_l$$

```
module IN01D1(I, Z)
  input I; output Z;
  assign Z = !I;
endmodule
```

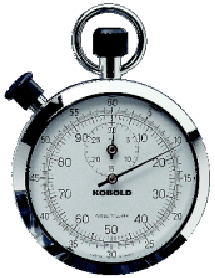
Specification, Architecture

- Functional
- Interface
- Physical
- Validated by cost study (clients, CAD, foundry, other providers)

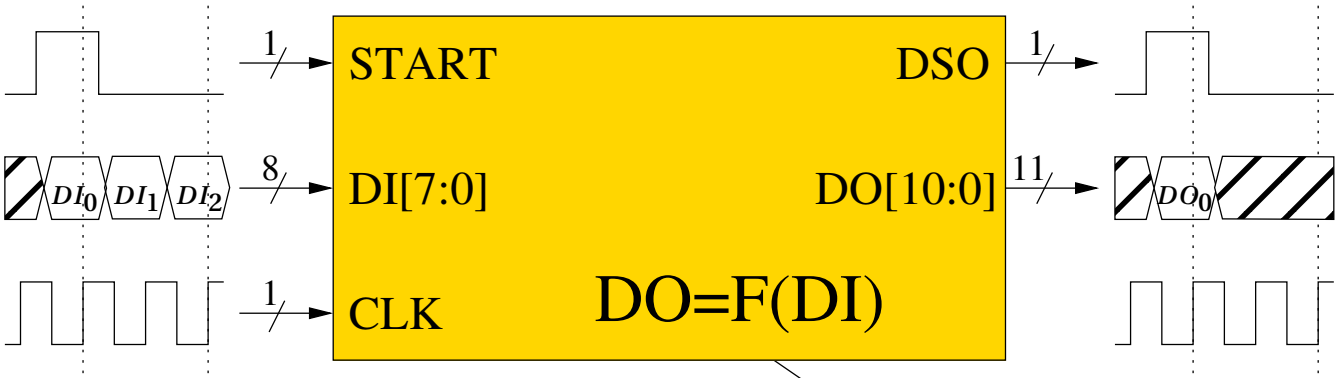


Specification: function, interface, physical

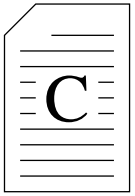
$P < 500\ \mu W$



$F > 2$ GHz



$S < 150\ \mu^2$

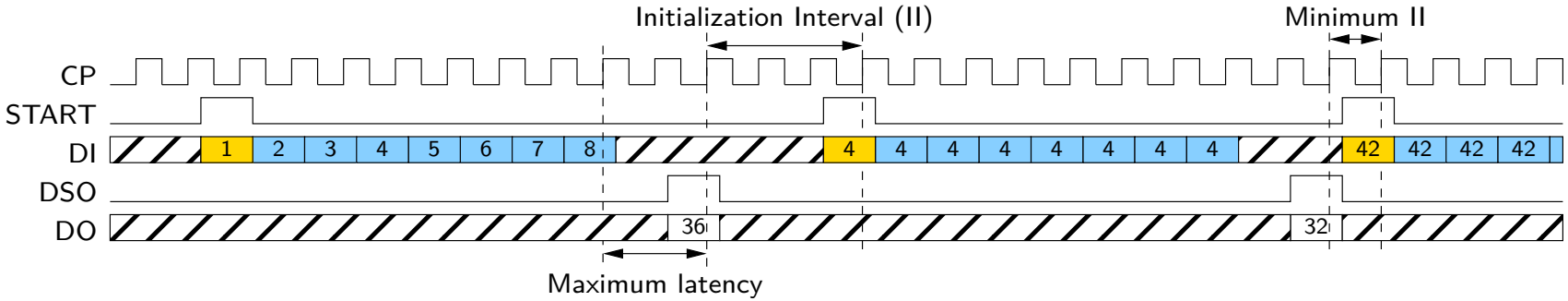


Example of specification

- Function
  - Accumulator on series of 8 unsigned integers
  - $F(DI_1, ...DI_8) = \sum_{i=1}^{i=8} DI_i$
  - Integers in the $[0..255]$ range $\Rightarrow$ result in $[0..8 \times 255]$
- Physical
  - Manufacturing process: 28 nm
  - Operating conditions
    - Process: typical
    - Power supply: 1V
    - Temperature: 25 °C
  - Clock frequency > 2 GHz
  - Silicon area $< 150\ \mu^2$
  - Power $< 500\ \mu W$

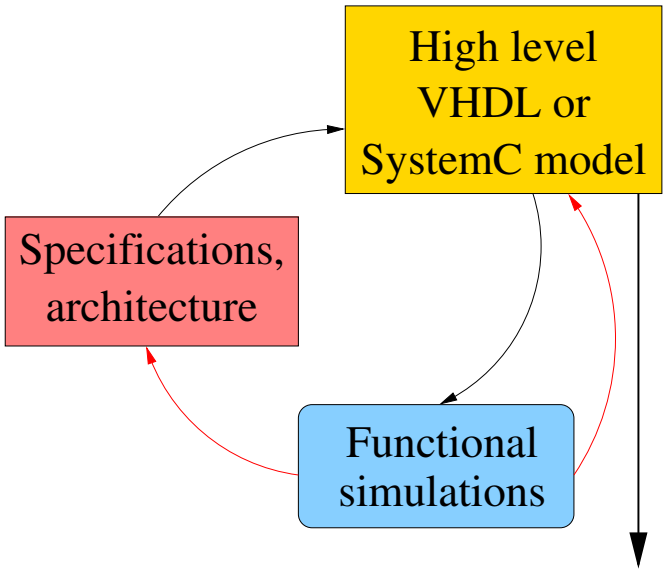
Example of specification

- Interface
  - First DI of series valid on CLK rising edges for which START is set
  - Next DI valid on 7 next CLK rising edges
  - DO $= F(DI_1, ...DI_8)$ valid on CLK rising edges for which DSO is set
  - Maximum latency between last DI and DO: 2 clock cycles
  - Minimum latency between DO and next START: 1 clock cycle



High level modelling

- Reference golden model
  - (Timed) transaction level model
  - Bus-cycle accurate, bit accurate
- Maintained all along the design flow
- Validated by simulations



Bus-cycle, bit accurate, reference model

- Golden reference all along the project
- Updated and re-validated when needed
- Fast in simulation
- Close from actual hardware:
  - Hierarchy compatibility
  - Data flow compatibility
- Far from actual hardware:
  - Better coverage
- Validated by functional simulations
  - Against a reference algorithmic model (C)
  - And performance / interface specifications

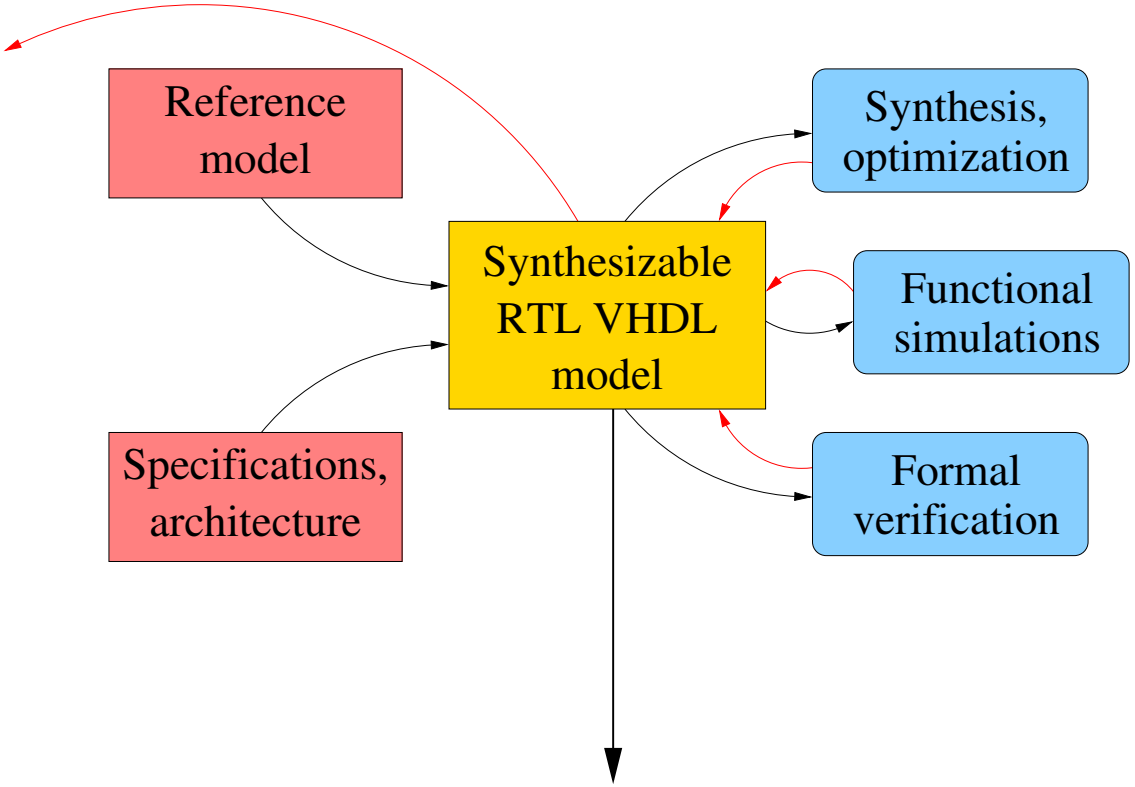
Example of high level model

```
1  package acc_pkg is
2    subtype ui8 is natural range 0 to 255;
3    type ui8_t is array(1 to 8) of ui8;
4    subtype ui11 is natural range 0 to 8 * 255;
5    function f(t: ui8_t) return ui11;
6  end package acc_pkg;
7
8  package body acc_pkg is
9    function f(t: ui8_t) return ui11 is
10     variable r: ui11 := 0;
11   begin
12     for i in 1 to 8 loop
13       r := r + t(i);
14     end loop;
15     return r;
16   end function f;
17 end package body acc_pkg;
18
19 use work.acc_pkg.all;
20
21 entity acc is
22   port(clk, start: in  bit;
23        di:         in  ui8;
24        dso:        out bit;
25        do:         out ui11);
26 end entity acc;
```

```
1  --pragma translate_off
2  architecture hlm of acc is
3  begin
4
5    process
6      variable t: ui8_t := (others => 0);
7      variable n: ui8   := 0;
8    begin
9      wait until clk = '1' and clk'event and
             start = '1';
10     t(1) := di;
11     for i in 2 to 8 loop
12       wait until clk = '1' and clk'event;
13       t(i) := di;
14     end loop;
15     wait until clk = '1' and clk'event;
16     dso <= '1'; do  <= f(t);
17     wait until clk = '1' and clk'event;
18     dso <= '0';
19   end process;
```

Refinement for synthesis

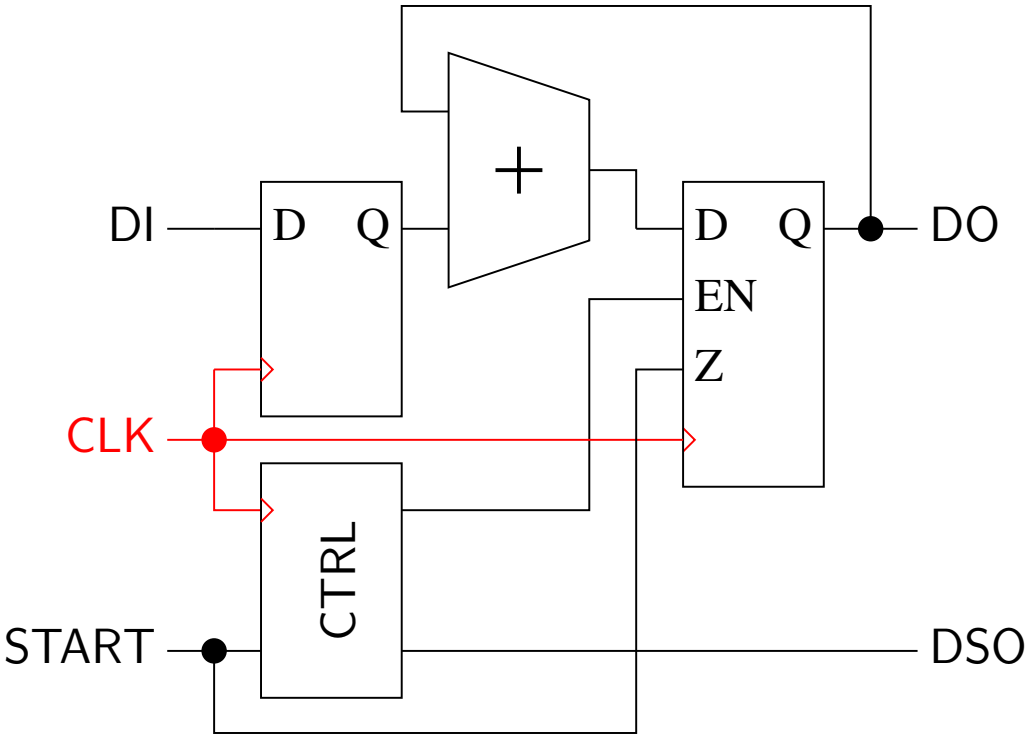
- Synthesizable RTL (Register Transfer Level) model
- Validated by simulations, formal verification and synthesis



The VHDL synthesizable model

- Logic synthesis is 100% functional
- The logic synthesizer supports only VHDL
- The logic synthesizer does not support all VHDL
- The coding style may impact the results
- Synthesis options are sometimes VHDL (attributes)
- ...and sometimes not:
  - Pragma comments
  - Command-line options
- Packages for synthesis exist
- Simulation speed is critical
- Joint verification function / synthesis
- Functional verification can be formal (equivalence or model checking)

Example of VHDL synthesizable model

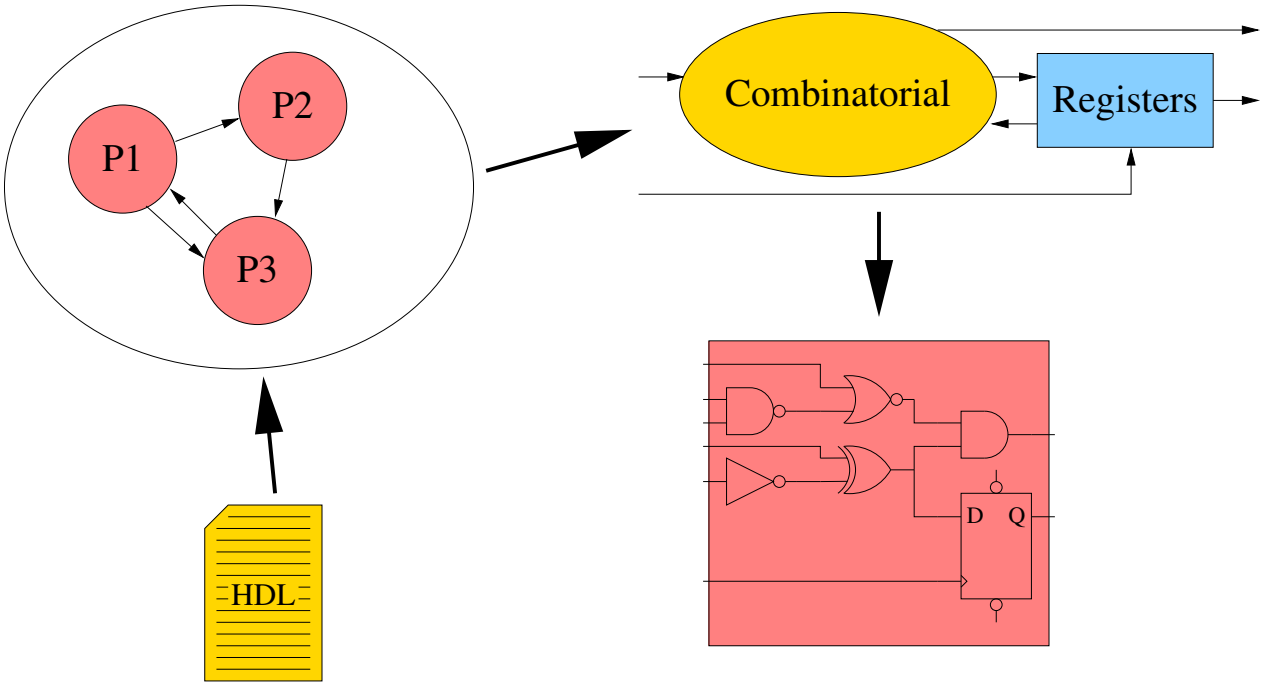


Example of VHDL synthesizable model

```
--pragma translate_on

use work.acc_pkg.all;

architecture rtl of acc is
  signal di_reg:   ui8;
  signal do_local: ui11;
  signal sum:      natural range 0 to 9 * 255;
  signal cnt:      natural range 0 to 8;
  signal en:       bit;
begin

  ctrl_p: process(clk)
  begin
    if clk = '1' and clk'event then
      dso <= '0';
      if cnt /= 0 then
        if cnt = 8 then
          dso <= '1';
          cnt <= 0;
        else
          cnt <= cnt + 1;
        end if;
      elsif start = '1' then
        cnt <= 1;
      end if;
```

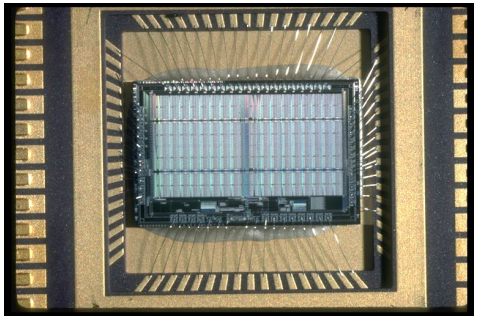
```
  end process ctrl_p;

  do  <= do_local;
  sum <= do_local + di_reg;
  en  <= '1' when cnt /= 0 else '0';

  di_p: process(clk)
  begin
    if clk = '1' and clk'event then
      di_reg <= di;
    end if;
  end process di_p;

  do_p: process(clk)
  begin
    if clk = '1' and clk'event then
      if start = '1' then
        do_local <= 0;
      elsif en = '1' then
        do_local <= sum;
      end if;
    end if;
  end process do_p;
```

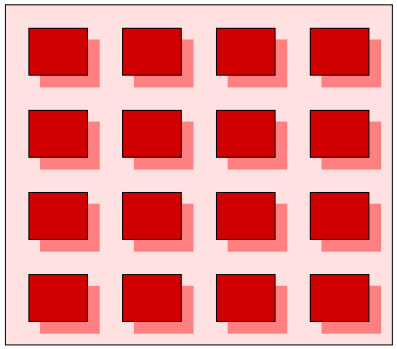
Logic synthesis

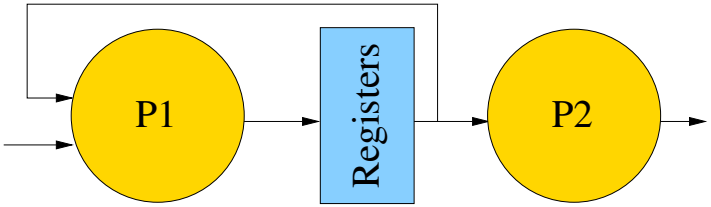


Synthesizable VHDL

- 100% hardware
  - No pointers
  - No files
  - Realistic types only (no reals)
  - No physical time (`after` clause)
- Resources identified, allocated, scheduled
- Reasonable complexity (granularity)







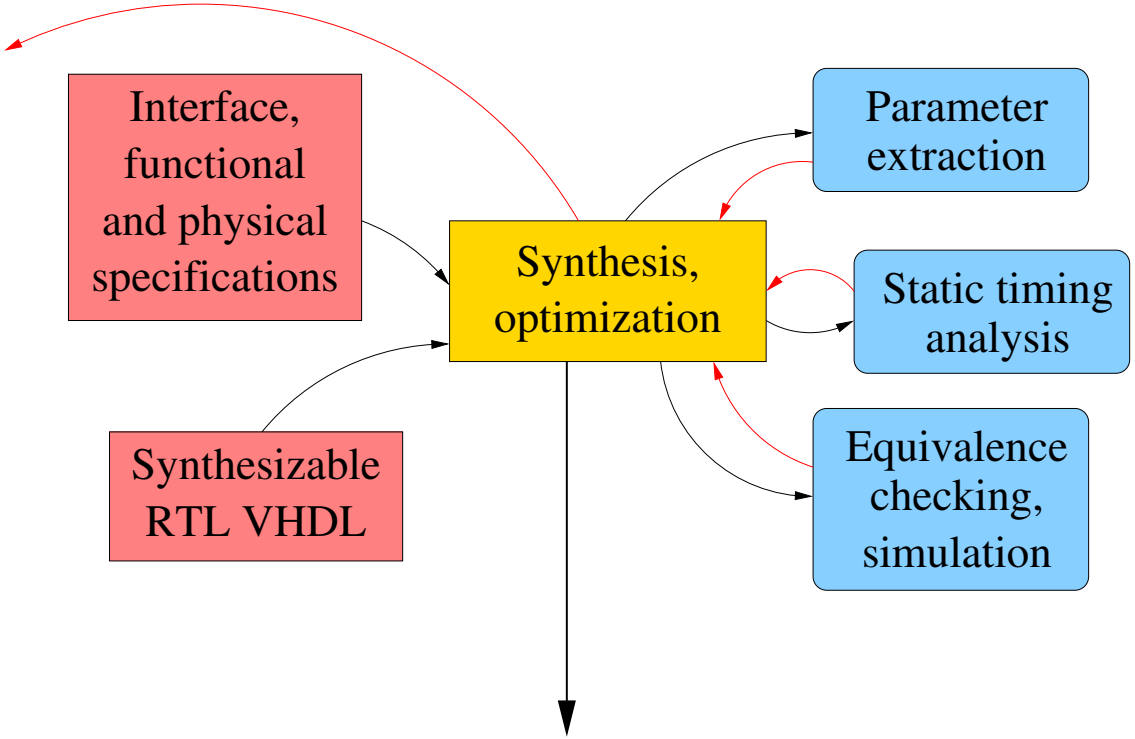
Logic synthesis: options

- They may be dedicated comments, interpreted by the synthesizer
- VHDL attributes
- Commands of the synthesizer
- Various types:
  - `synthesis on/off`
  - `translate on/off`
  - `set` and `reset` management
  - Architecture of arithmetic operators
  - Enumerated types encoding
  - Multiplexed or hard wired logic (`case`)
  - Coding and optimization of state machines
  - Semantics of resolution functions, ...

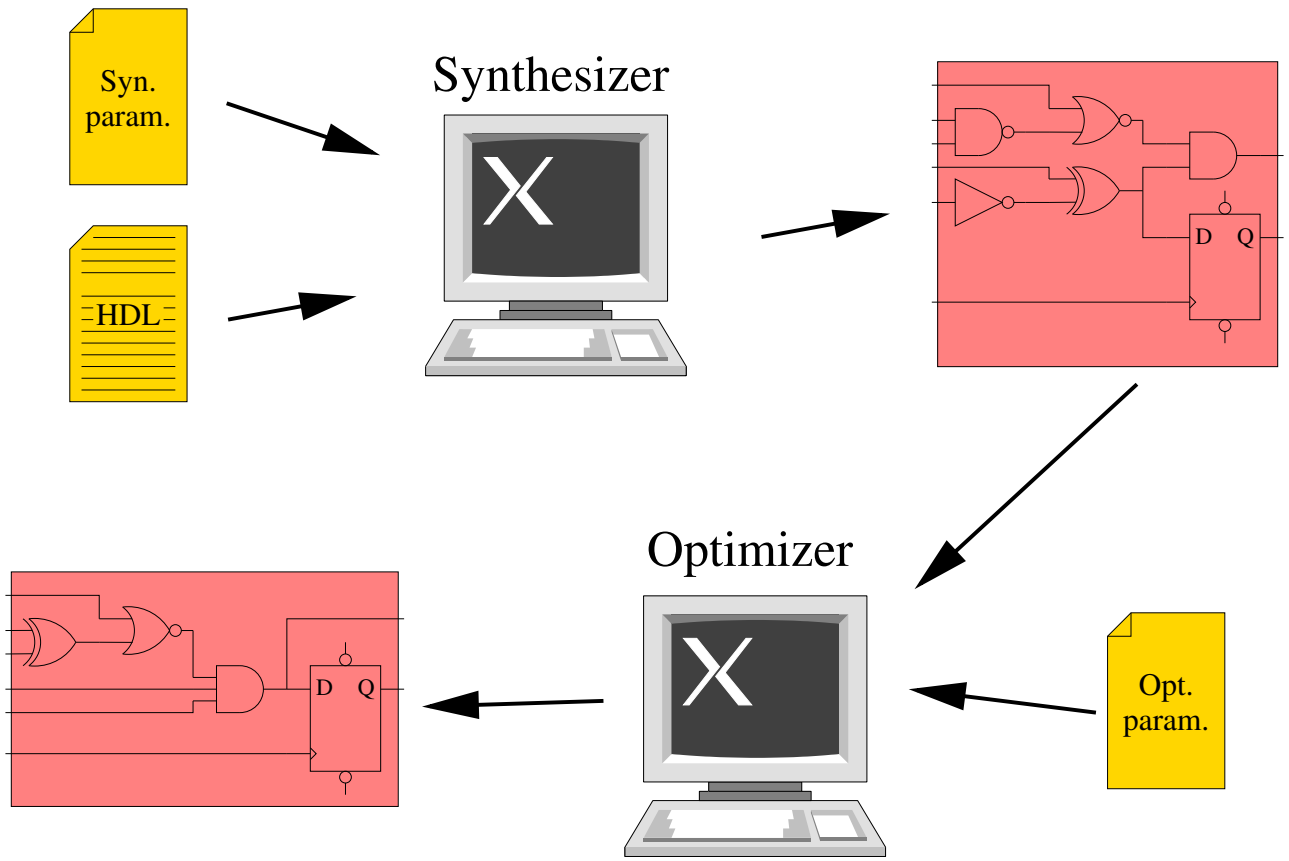
Logic synthesis: validation

- Carefully read the messages
  - Syntax errors
  - Warnings (latches, sensitivity lists,...)
- Count the registers
- Estimate and check complexity (adders)
- Beware the CPU run time

Netlist optimization



Synthesis, optimization



Synthesis, optimization, step by step

- Source code analysis
  - Memory elements inference
  - Combinatorial parts extraction
- Intermediate representation
- Logic minimization (Karnaugh, Quine-McCluskey, Espresso, ...)
- Logic optimization
  - Factorizations
  - Substitutions
  - Simplifications, ...
- Technology-independant gate mapping
- Technology mapping

Synthesis, optimization

```
S <= (A and B and (not C)) or
     (F1 and F2);
F1 <= A or (not C);
```

Flattening

```
S <= (A and B and (not C)) or
     ((A or (not C)) and (A and C));
```

Simplification

```
S <= (A and B) or
     (A and C);
```

Factorization

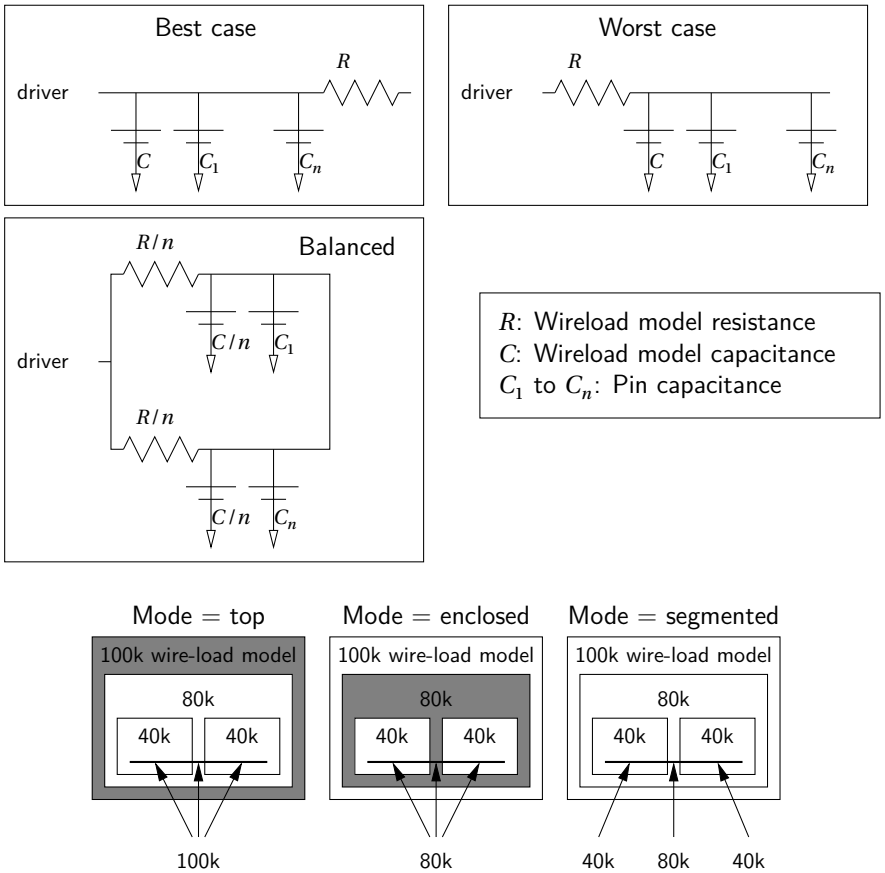
```
S <= A and F1;
F1 <= B or C;
```

Optimization

- Physical specifications taken into account
  - Silicon area
  - Speed
  - Environment
- Can be slow and CPU intensive
  - No useless optimizations
  - Fine grain designs
- Reproducibility
- Impact on and from other steps of the design flow
  - Testability
  - Floor plan
  - Place and route

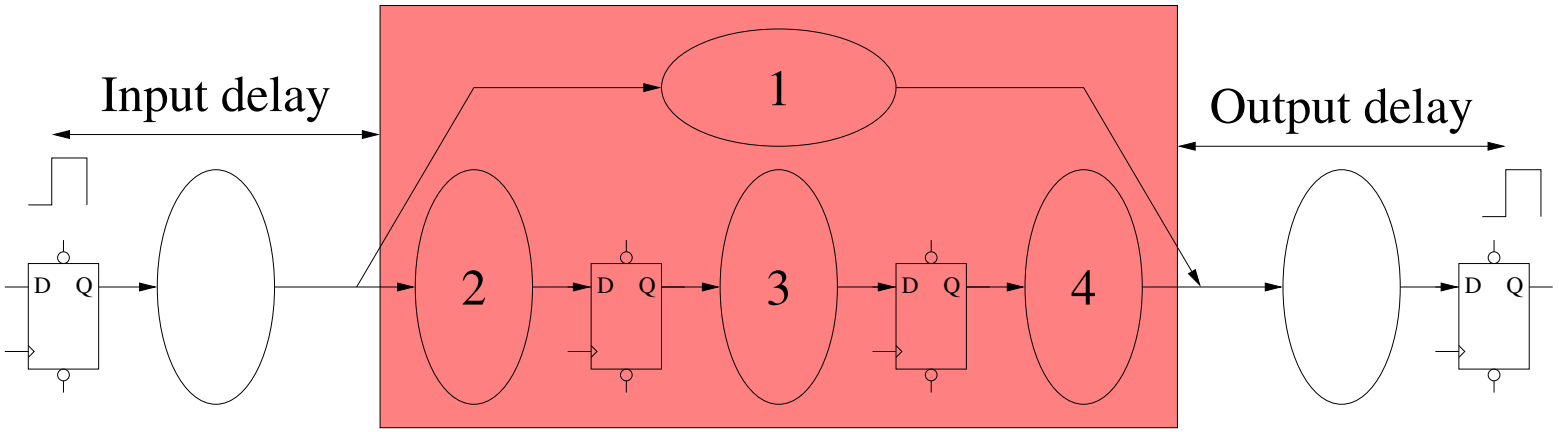
Propagation delay model

- Used by optimizer to predict delays
- Depends on operating conditions
  - Temperature
  - Voltage
  - Manufacturing process quality
- Depends on parameters
  - RC tree structure
  - Wire-load mode
  - Wire-load model (area)
  - Voltage



Timing constraints

- The designer specifies:
  - The clocks and their characteristics
  - The multi-cycle paths
  - The input delays
  - The output delays



Other parameters

- Driving gates of inputs
- Maximum allowed ramp delay
- Maximum allowed load capacitance
- Maximum number of gates driven by a signal (fanout)
- Hierarchy modifications (uniqueness, merge, flatten, etc.)
- Tuning of CPU run time and memory usage for each phase

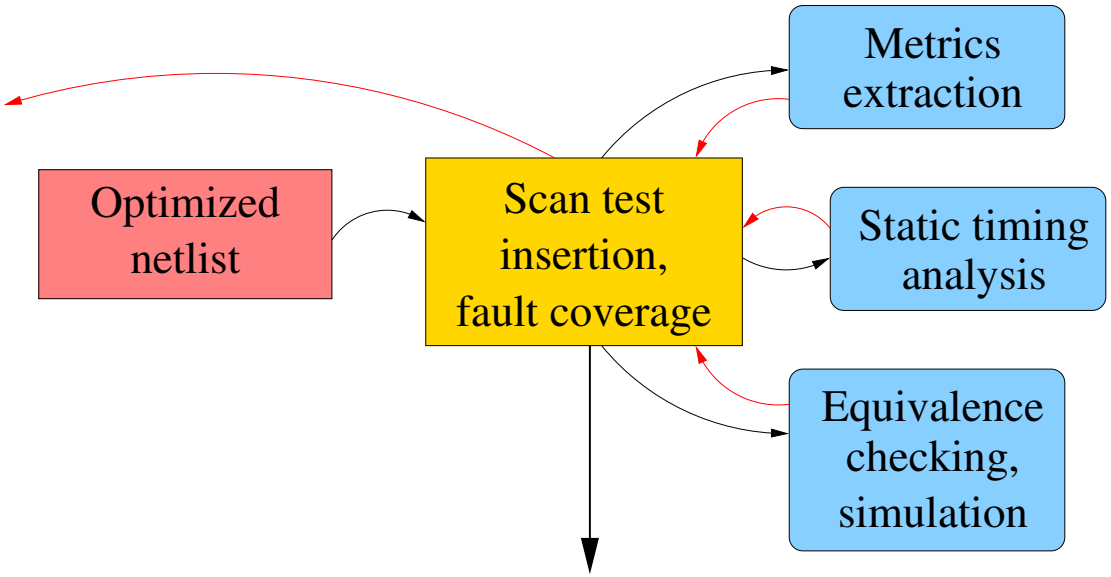
Optimization: validation

- Carefully read the messages
  - Syntax errors in input netlist
  - Number of registers
  - Estimate and check complexity (adders)
  - Timing constraints
- Beware the CPU run time
- Syntax checking
- Static timing analysis
- Equivalence checking
- Timed simulation
  - Initialization defects
  - Setup and hold violations
  - Hardware emulators can be usefull

Static timing analysis

- The delay calculator is a major component of the optimizer
- It uses different models of the propagation delays
  - Prop ramp delays
  - Input slope
  - Wave tabular
- Its performance (CPU run time and accuracy) are critical
- It can be used in stand alone mode:
  - To evaluate an unoptimized design (hiererachical assembly)
  - To evaluate the impact of some modifications:
    - Operating conditions
    - Test insertion
    - Place and route
- STA and optimization parameters are almost the same

Test


Metrics extraction
Optimized netlist
Scan test insertion, fault coverage
Static timing analysis
Equivalence checking, simulation

Test

- It's a manufacturing test, not a functional verification
- The designer provides the test vectors
- The manufacturer usually tests
  - On the wafer
  - After packaging
- PCBs (boards) are also tested
- The manufacturing test has a great economics impact

? Exercise #1: what is the defect probability of a PCB:

- 10 ICs, 1.5 cm^2 each
- Manufactured with a defect probability of 10% per cm^2
- All tested with a 97% coverage

Test

- It's a manufacturing test, not a functional verification
- The designer provides the test vectors
- The manufacturer usually tests
  - On the wafer
  - After packaging
- PCBs (boards) are also tested
- The manufacturing test has a great economics impact

? Exercise #1: what is the defect probability of a PCB:
  - 10 ICs, 1.5 cm^2 each
  - Manufactured with a defect probability of 10% per cm^2
  - All tested with a 97% coverage

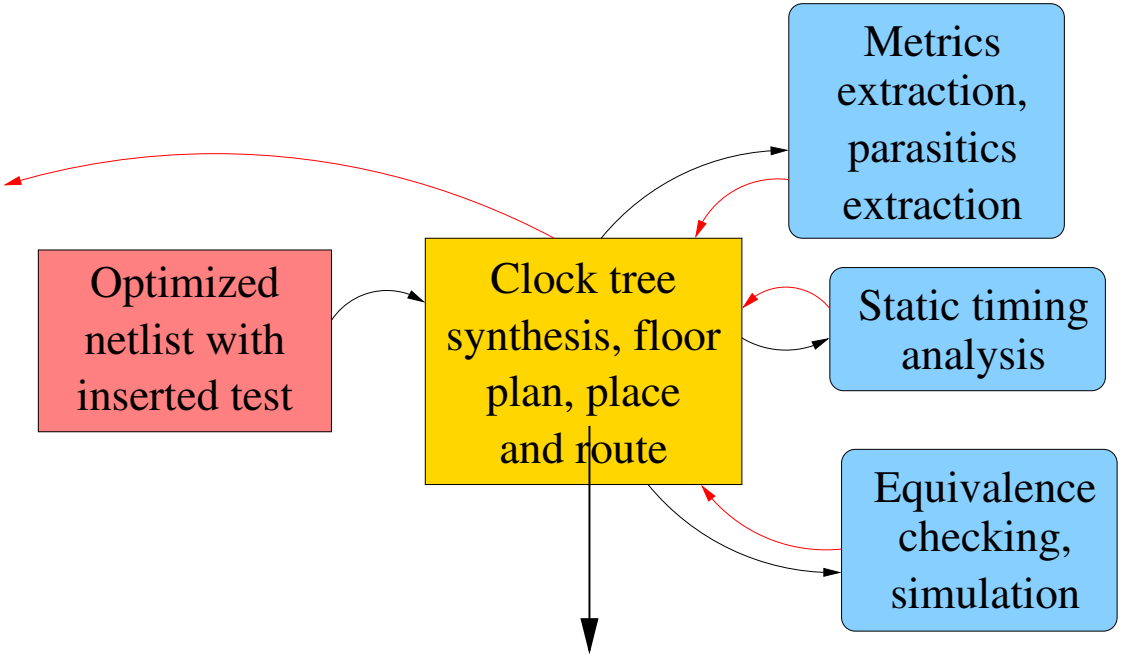
Test

- The fault model is usually simplified:
  - Stuck at 0 or 1 of a net
  - One single fault per chip
- Fault simulation:
  - The chip is simulated with the assumption of a fault
  - The fault is "covered" iff the outputs differ from the non-faulty chip
- 100% coverage means that the test vectors detect every possible fault
- The coverage depends on the controllability and observability of the chip

Test

- Different strategies
  - "Scan path"
  - Built-in self test ("BIST")
  - Autotest, ...
- The test usually impacts the timings
- The test usually increases the silicon area
- There are important dependencies between test and place and route
  - Connection of scan chains after placement
- PCB testing uses different approaches (boundary scan, JTAG standard) that must be taken into account at chip level

Physical design

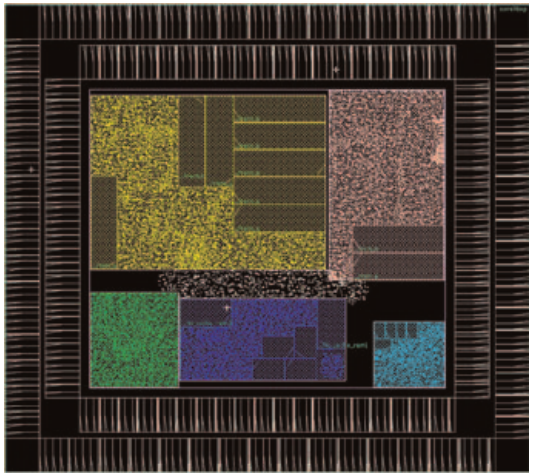


Floor plan

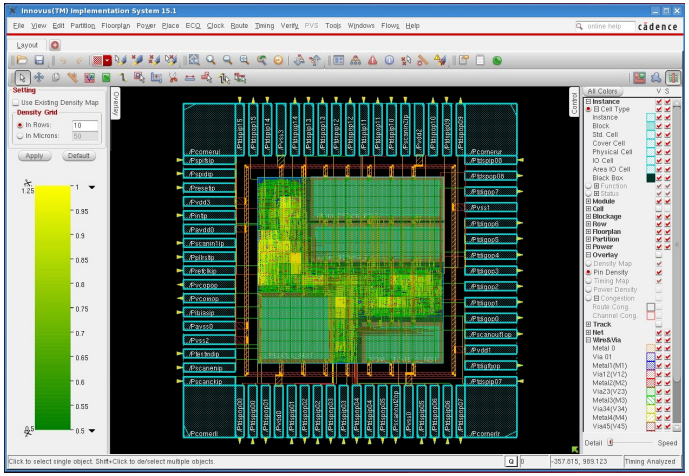
- Block-based organization of the chip
- Position of input / output pads
- Position and number of power pads
- Power supply routing strategy
- Clocks routing strategy
- Identify the critical interconnect delays
  - In deep submicron designs (since 500 nm) the propagation delays are wire-dominated

Floor planning tools

- Aspect ratio of standard cell areas
- Detect congestions
- ...and solve them
- Power routing strategy
- Clocks routing strategy



Source: Magma Design Automation



Source: Cadence Design Systems, Inc.

Place & route

- Slow, CPU intensive
- Timing characteristics changed
- Power supply dimensioning
- Impacts on the architecture
  - Partitioning
  - Bus segmentation
- Impacts on the synthesis / optimization
  - Timings
  - Silicon area
- Very complex tools
- Frequently done by specialized companies

Place & route

- Placement
  - Standard cells are placed in rows
  - Interaction with test (scan chains)
  - Attempts to minimize the congestion
  - Uses a lot of classical optimization algorithms
- Routing
  - Multi-level (cells, blocks)
  - Block routing is a bit more difficult (irregular block sizes and routing channels)
  - Critical nets can be manually routed or constrained
  - Provides a good estimation of parasitics

Last steps

- Parasitics extraction
- Back-annotated static timing analysis
- Back-annotated timed simulation
  - Power consumption estimation
  - Setup and hold violations
- Design Rules Checking (DRC)
- Electrical Rules Checking (ERC)
- Design for manufacturing
- Bonding diagram
- Tape-out
- Champagne

Outline

Conclusion

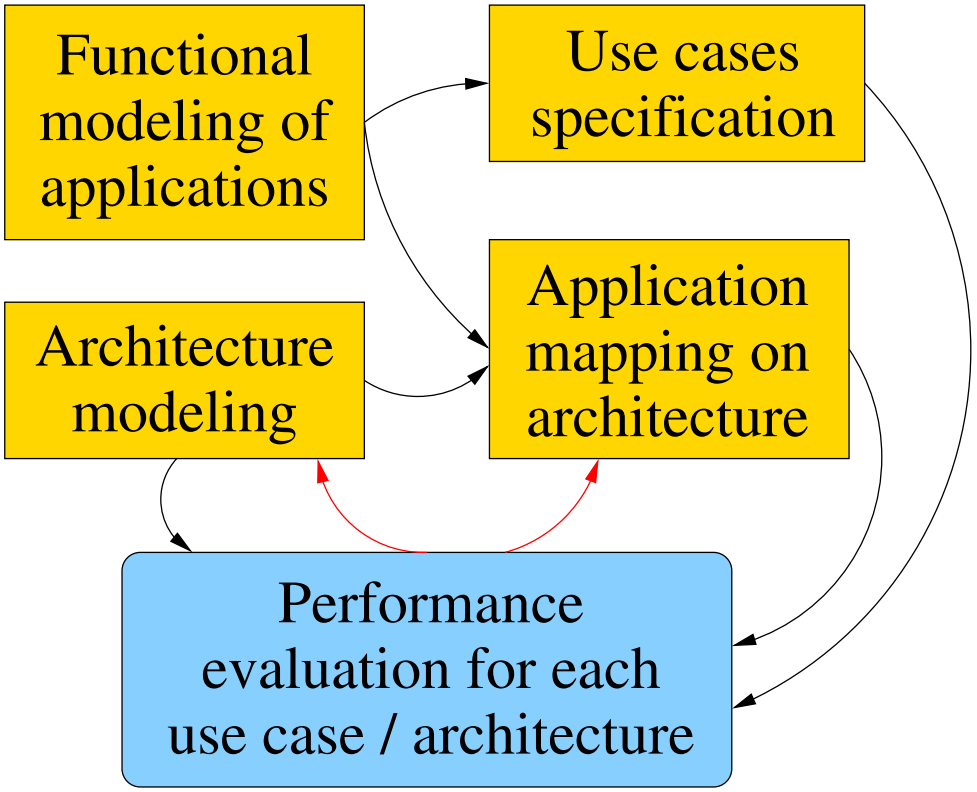
- Logic design is critical
- High added value
- Huge impact on product quality
- Verification is very expensive
- Hardware choices are frequently driven by non-hardware constraints:
  - Partitioning
  - Verification techniques
  - Performances of the tools
  - Manageable complexity
  - Reusability, ...

Outline

What is a SoC?

- "System" ⇒ a collection of interacting elements
- "On Chip" ⇒ on a single silicon
- But any integrated circuit is not a SoC:
  - A SoC implements a complete functionnality:
    - Video decoder + audio + system layers + transport + graphics + user interface
    - 4G handset: everything but RF
  - The elements of a SoC are "complex", reusable and heterogeneous:
    - Analog functions (AD/DA converters, filters, etc.)
    - Hardware digital functions (Viterbi decoder, FFT)
    - Software functions (on a microcontroller, a DSP, a RISC processor)
    - Others (memories, busses, etc.)

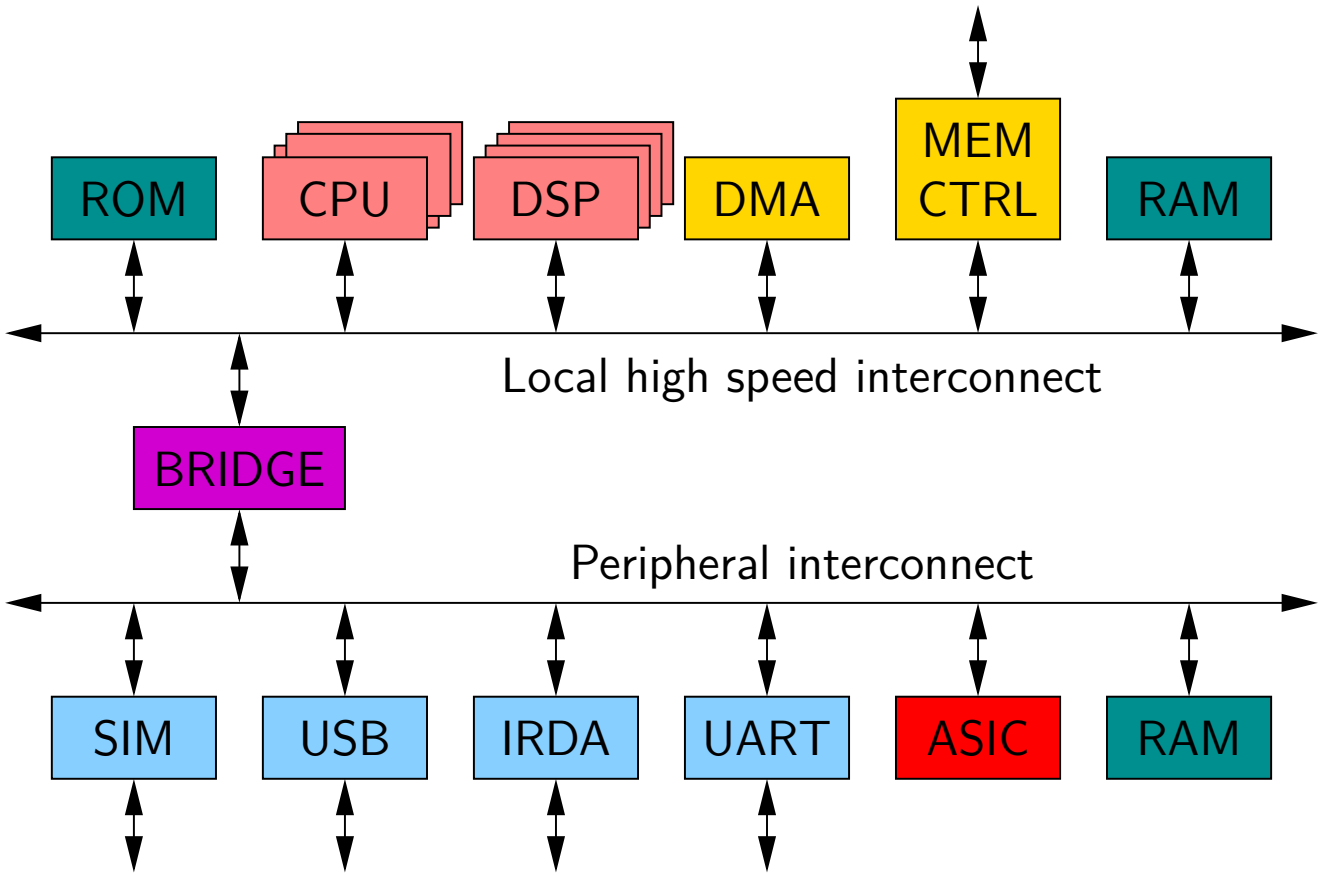
SoC: design space exploration



Example of application

- 4G mobile communications
- Basesand and application processor
- Telecommunications (multi-standards)
- Applications
  - Voice
  - Video
  - Graphics 2D/3D (clone animation, gaming)
  - Data (WEB, etc.)
  - Agenda, address book, PDA-like functions, ...

Example of architecture



Example of a use case

- Situation #1
  - LTE
  - MPEG4 video streaming
- Situation #2
  - LTE
  - Visiophony
- Situation #3
  - LTE not available
  - Switch back to 2G (GSM)
  - Clone animation

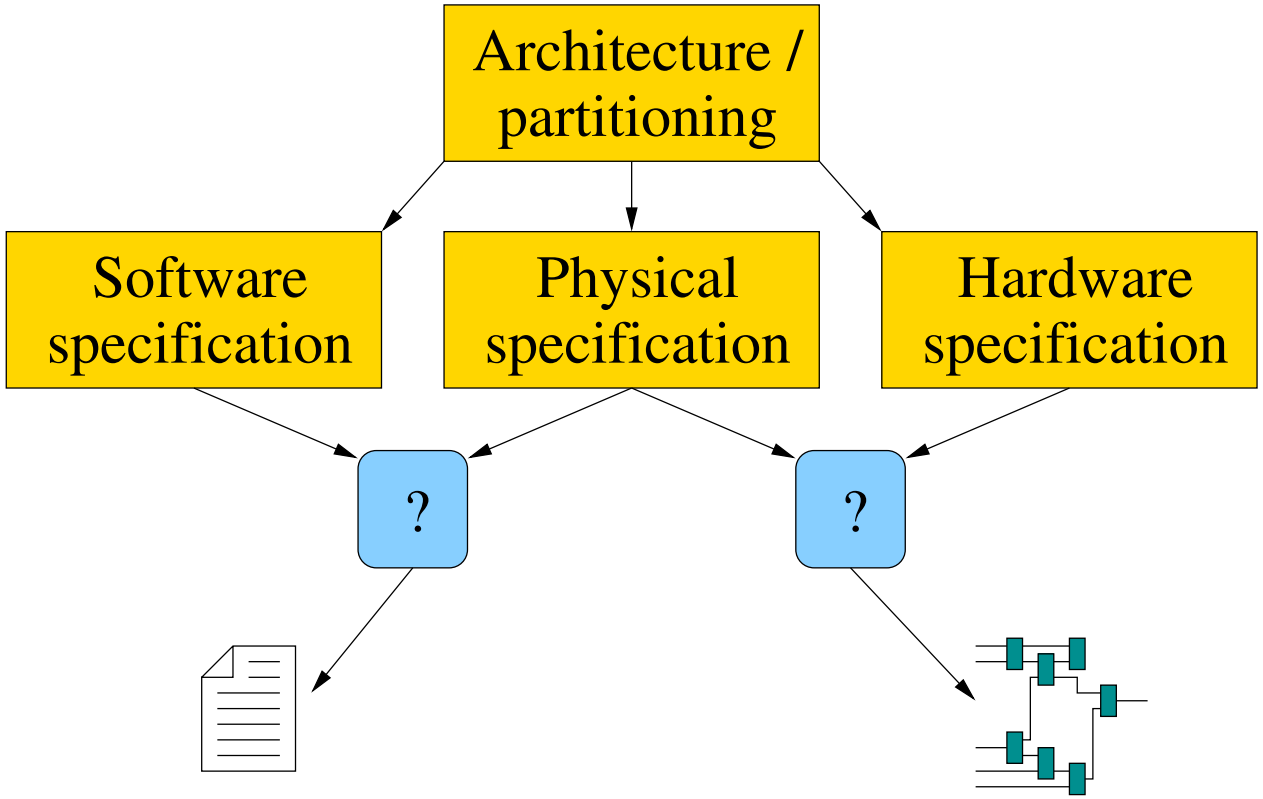
Design space exploration

- Hardware / software partitioning
- Selection of CPUs, processing power, level of parallelism
- Specification of software and hardware modules
- Selection of interconnect architectures (buses, crossbars, NoCs)
- Dimensioning of shared ressources (memories, buses), caches, ...
- Validation
  - Deadlock - livelock detection
  - Real time constraints, ...

Design space exploration

- Validation by simulation
- On cycle accurate models
  - 100% functional
  - Huge simulation time, reduced coverage
  - Long and expansive design tasks before first simulation
- On approximate models
  - Non functional
  - Reduced simulation time, increased coverage
  - Reduced design effort

SoC: refinement



SoC: design reuse

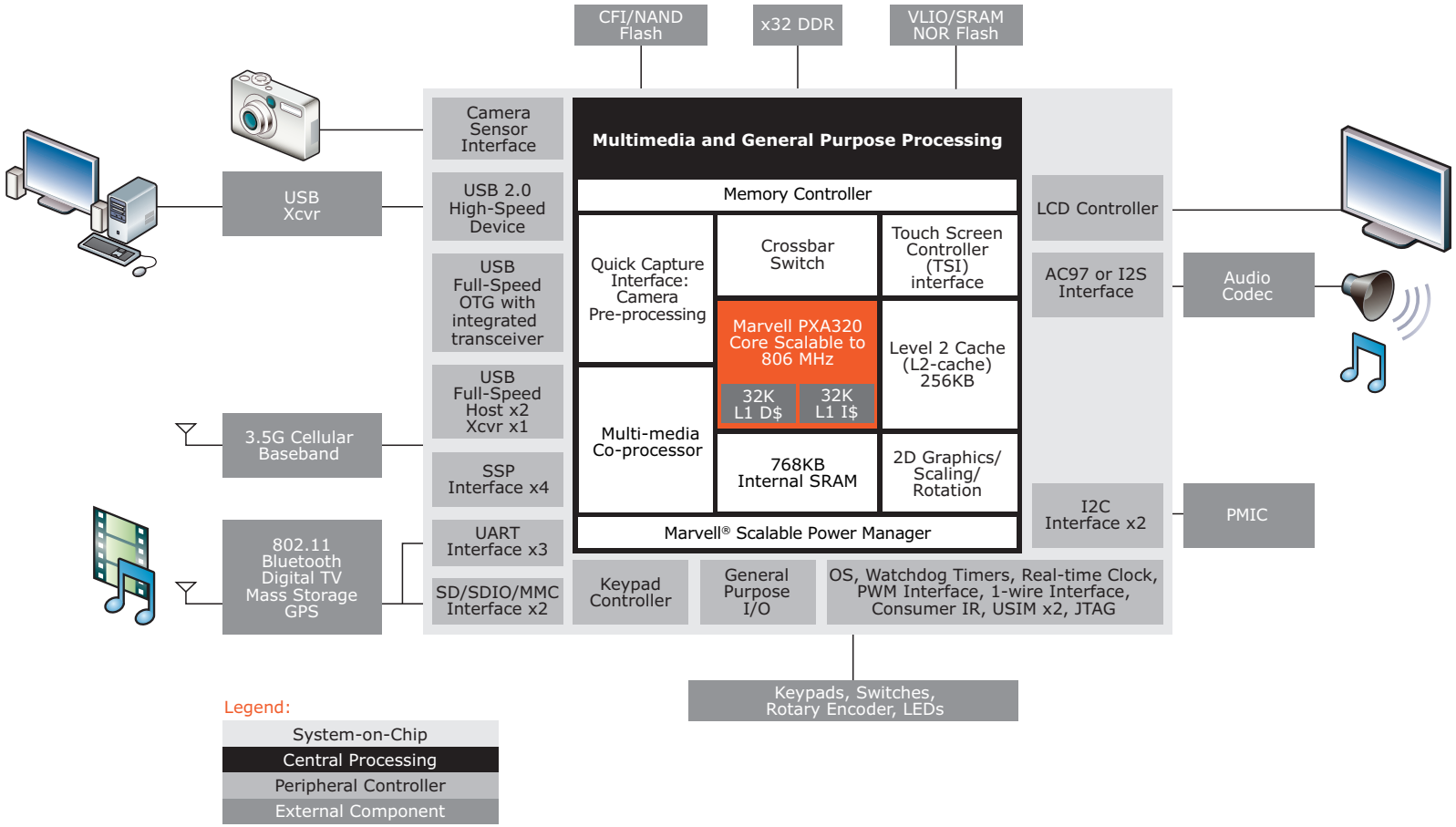
- Use of virtual components (IP blocks)
  - Changing environments
  - Highly generic
  - Various providers
  - Impact on verification strategy
- IP design
- IP maintenance

SoC: constraints

- Complexity increases
- Non recurring engineering costs increase
- Design time decreases
- Performance of CAD tools always too limited

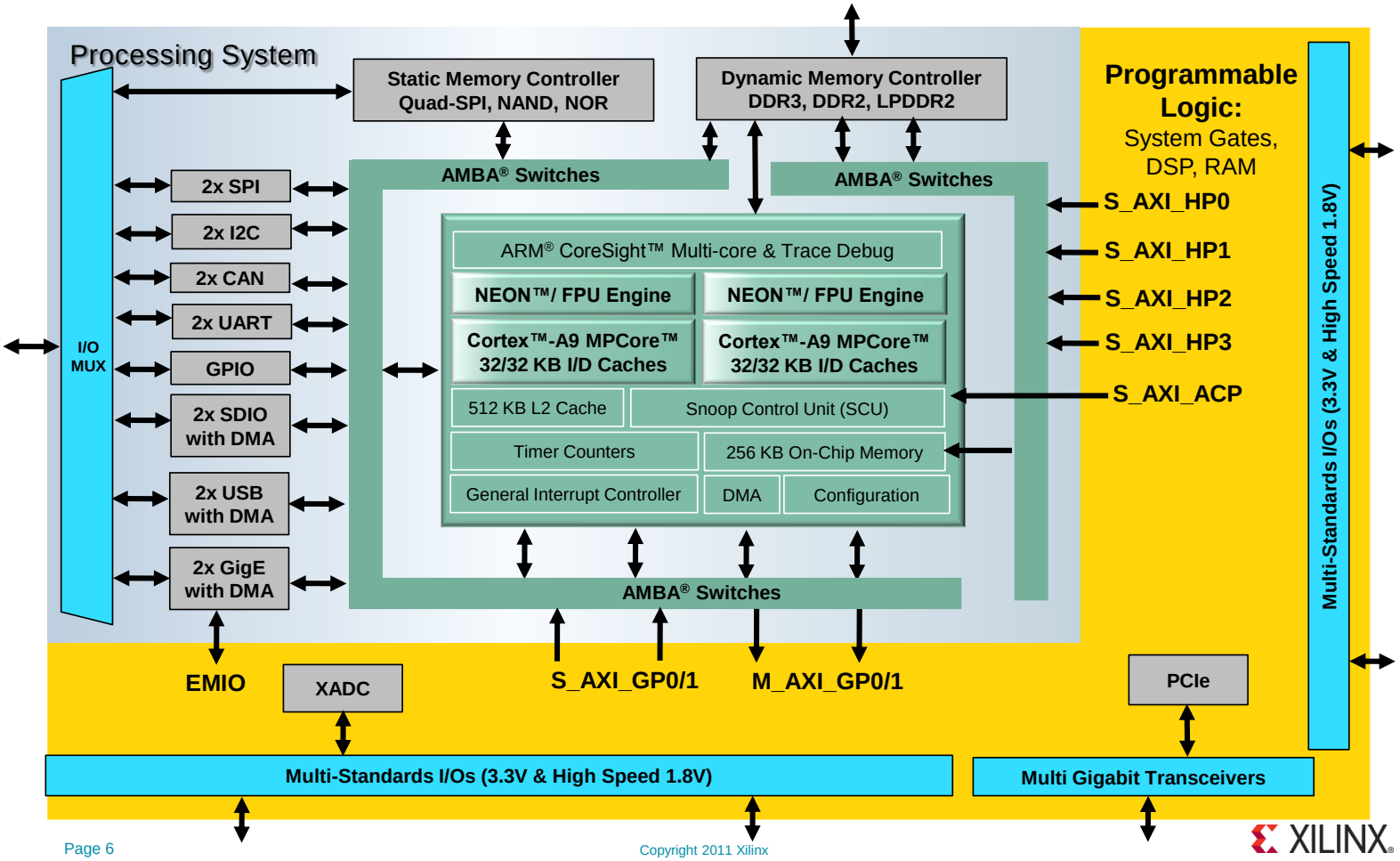
How to design as fast as possible very complex systems, without errors and with limited CAD tools?

Example: the PXA320 (Smartphone Application Processor) by Marvell



PXA320 Processor Block Diagram

Example: the Zynq family by Xilinx



Application-Specific ICs (ASICs) to SoC

- System level skills mandatory
- Embedded CPUs (hardware / software co-design)
- Embedded large memories (increased bandwidth)
- Design reuse, virtual components (IPs)
- Interconnect sandards (AMBA, NoCs, ...)
- Mixed signal
- Low power
- Testability
- Complexity

Summary

- DigitalSystems: the course. Questions?
- IC design flow. Questions?
- SoC. Questions?

Further Reading I

- R. Zurawski
  Embedded systems handbook
  CRC Press - 16/08/2005 - 1160 p.
- F. Nekoogar
  From ASICS to SoCs: a practical approach
  Prentice-Hall - 2003 - 188 p.
- D. Chinnery, K. W. Keutzer
  Closing the gap between ASIC & custom: tools & techniques for high-performance ASIC design
  Kluwer academic - 2002 - 407 p.
- D. Jansen
  Electronic design automation handbook
  Kluwer academic - 2003 - 675 p.

Further Reading II

M. Balch
Complete digital design: a comprehensive guide to digital electronics and computer system architecture
McGraw-Hill - 2003 - 460 p.

M. J. S. Smith
ASICs... the book
Addison-Wesley - 06/1997 - 1040 p.

Ashenden, Peter J
Designer's guide to VHDL
Morgan Kaufmann - 06/2008 - 936 p.

Zwolinski, Mark
Digital system design with VHDL
Prentice-Hall - 2004 - 384 p.

Further Reading III

Wolf, Wayne Hendrix
Modern VLSI design: systems on chip design
Prentice-Hall - 2002 - 618 p.

And a lot more in the library...