Digital Systems exam (2 hours)

Renaud Pacalet - 2025-06-19

You can use any document but communicating devices are strictly forbidden. Please number the different pages of your paper and indicate on each page your first and last names. You can write your answers in French or in English, as you wish. Precede your answers with the question's number. If some information or hypotheses are missing to answer a question, add them. If you consider a question as absurd and thus decide to not answer, explain why. If you do not have time to answer a question but know how to, briefly explain your ideas. Note: copying verbatim the slides of the lectures or any other provided material is not considered as a valid answer. Advice: quickly go through the document and answer the easy parts first.

The first question is worth 4 points. The other questions are worth 2 points each. The problem is worth 10 points

1. Questions

1.1. Synthesizable VHDL models

List the different constraints a combinatorial VHDL process must fulfill for proper synthesis.

1.2. VHDL signals and variables

What are the differences between signals and variables in VHDL?

1.3. VHDL synthesis

In your opinion, how many bits of register will be inferred when synthesizing the VHDL code of Listing 1? Draw a schematic of the synthesis result.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity e is
5    port(clk: in  std_ulogic;
6         x:   in  std_ulogic_vector(8 downto 5);
7         s:   out std_ulogic_vector(7 downto 0));
8  end entity e;
9
10 architecture a of e is
11 begin
12   p: process(clk)
13     variable v: std_ulogic_vector(7 downto 0);
14   begin
15     if rising_edge(clk) then
16       s <= v;
17       v := v(3 downto 0) & x;
18     end if;
19   end process p;
20 end architecture a;
```

Listing 1: Synthesizable model

1.4. Design of a Moore Finite State Machine

We want to design a Moore Finite State Machine (FSM) to control a motor. The inputs and outputs of the FSM are listed in Table 1.

Name	Direction	Type	Description
`clk`	input	`std_ulogic`	clock
`srstn`	input	`std_ulogic`	synchronous active low reset
`ms`	input	`natural range 0 to 10`	current motor speed
`rs`	input	`natural range 0 to 10`	requested motor speed
`up`	output	`std_ulogic`	increase speed command
`down`	output	`std_ulogic`	decrease speed command
`done`	output	`std_ulogic`	feedback to operator

Table 1: Inputs/outputs of FSM

FSM is synchronized on rising edges of `clk`. Motor speeds `ms` (current) and `rs` (requested) are represented as a numeric value between 0 (motor stopped) and 10 (full speed). Apart from being constrained by this range there are no other hypotheses about them; they can change any time, and take any value. For instance, from one clock edge to the next `ms` can change from 2 to 8 while `rs` changes from 10 to 1. The `up` and `down` commands are used by FSM to drive the motor; when set to `'0'` they have no effect on the motor, when set to `'1'` the action listed in Table 1 is applied.

For safety reasons `up` and `down` shall never be set to `'1'` at the same time, `up` shall be set to `'0'` when the current motor speed is 10, and `down` shall be set to `'0'` when the current motor speed is 0.

The `done` output is a feedback to the human operator; it is set to `'1'` when the current motor speed and the requested motor speed are equal, else to `'0'`.

When the reset is active (`srstn = '0'`) the FSM shall go to its reset state and the three outputs shall become `'0'`.

1.4.1. State diagram

Draw a state diagram of a Moore state machine implementing these specifications. Give meaningful names to your states. Clearly identify the reset state, the conditions for the transitions, and the FSM output values.

1.4.2. VHDL coding

Code in synthesizable VHDL 2008 the entity `fsm` and architecture `rtl` of the FSM. Add comments to your code to explain non-obvious parts, if any.

2. Problem: design and VHDL coding of a door lock

The goal of this problem is to design and code in synthesizable VHDL 2008 `dlctrl`, a controller for a digital door lock. The keypad and `dlctrl` are represented on Figure 1.

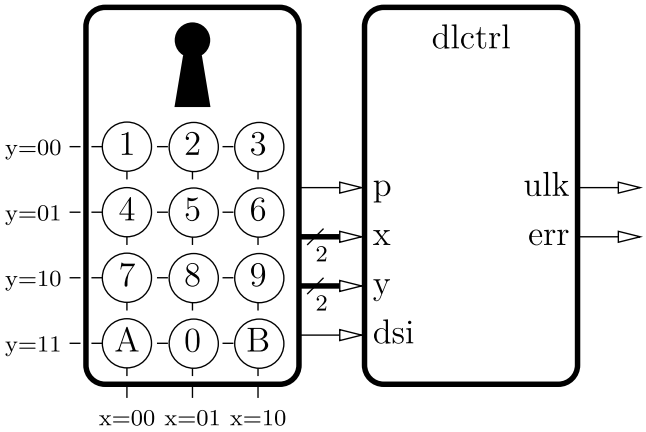

Figure 1: The keypad and `dlctrl`

To open the door a user must enter a 6-symbols code. After the 6$^{\text{th}}$ button was pressed, if the entered code is the same as the secret code stored in `dlctrl`, the door opens, else the door lock plays an error sound, restarts from scratch and waits until the user enters a new code.

The secret code is programmable. If the programming key is inserted into the key hole of the keypad while a button is pressed, the secret code changes: the symbols of the old secret code are shifted to the left, the new

symbol becomes the rightmost of the new secret code, and the leftmost symbol of the old secret code is lost. Example: if the current secret code is `1234AB` and the caretaker presses button `7` with the programming key inserted, the secret code becomes `234AB7`.

Table 2 describes the interface of `dlctrl`; the `x` and `y` coordinates of the buttons are as shown on Figure 1.

Name	Direction	Bit width	Description
`clk`	input	1	clock
`srstn`	input	1	active low synchronous reset
`p`	input	1	`'1'` if programming key inserted, else `'0'`
`x`	input	2	`x` coordinate of pressed button
`y`	input	2	`y` coordinate of pressed button
`dsi`	input	1	`'1'` when button pressed, else `'0'`
`ulk`	output	1	`'1'` to unlock door, else `'0'`
`err`	output	1	`'1'` to play error sound, else `'0'`

Table 2: Interface of `dlctrl`

The keypad and `dlctrl` are synchronized on rising edges of the same 1000 Hz (1 kHz) clock `clk` and have the same synchronous active low reset `srstn`, not represented on Table 2; there is no need for `dlctrl` to re-synchronize the inputs it receives from the keypad.

When a button is pressed on the keypad its `x` and `y` coordinates (2 bits each) are sent to `dlctrl` through the `x` and `y` inputs and the `dsi` input is set to `'1'` during one clock period. It is guaranteed that `x` never takes value `"11"`. The `p` input is set to `'1'` when the programming key is inserted in the key hole of the keypad, else it is set to`'0'`.

To open the door `dlctrl` sets its `ulk` output to `'1'` during one clock period. To play the error sound `dlctrl` sets its `err` output to `'1'` during one clock period. When `dlctrl` is reset the secret code becomes `AAAAAA`.

2.1. Design of block and state diagrams

Draw a block diagram of `dlctrl` using the symbols represented on Figure 2. Clearly identify the registers, give their size in bits, name them and explain what their role is.

If you use state machines represent them as “other synchronous elements” in your block diagram, with named inputs and outputs. Draw their state diagrams, explain whether they are Moore or Mealy state machines, and provide a description of their role.

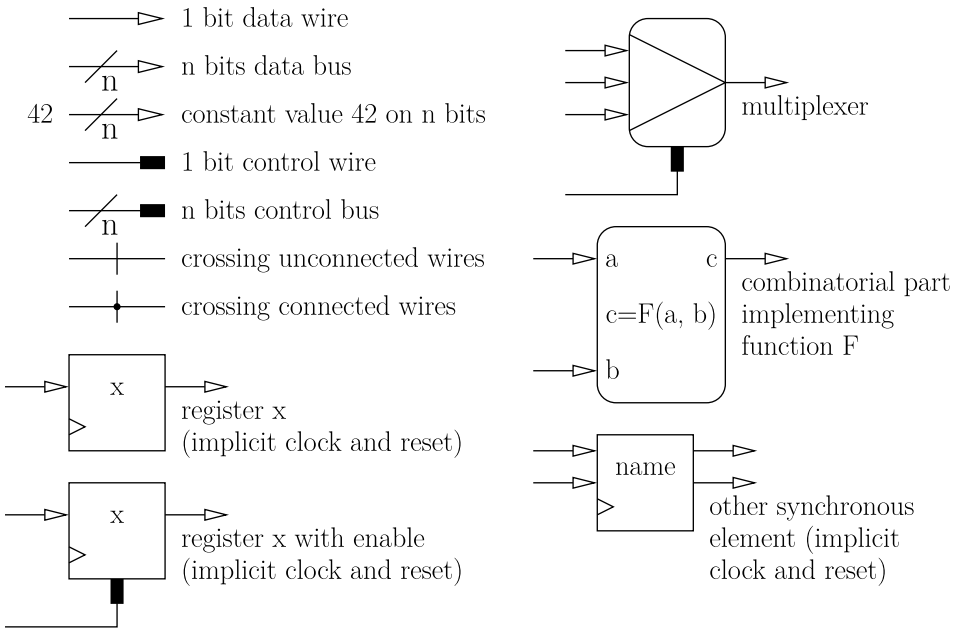

Figure 2: Symbols to design block diagrams

2.2. VHDL coding

Code `dlctrl` (entity and architecture) in synthesizable VHDL 2008. Do not use sub-designs and entity instantiations: code everything in the `dlctrl` architecture. Add comments to explain non-obvious parts of your code.

2.3. Timeout

A user may abandon before pressing 6 buttons. With the current design the next user will start entering their code while 1 to 5 keys have already been entered by the previous user. This will likely lead to an error or, maybe worse, to the door opening after less than 6 buttons have been pressed.

To avoid this we can add an internal timer to `dlctrl` such that the door lock automatically restarts from scratch some delay after the last button was pressed. Explain how you would use a timer like the one we designed during the labs to add this functionality. What would you modify? Do not code in VHDL, just explain what changes in your block and/or state diagrams.