DigitalSystems: exam

R. Pacalet

2022-06-23

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 5 questions are worth 2 points each. The problem is worth 10 points

1 Questions

1.1 State machines

What are the differences between a Mealy and a Moore state machine? What are their advantages and drawbacks?

1.2 Metastability

What is the metastability problem? How is it handled in digital hardware systems?

1.3 Verification of logic synthesis results

How do you verify the results of the logic synthesis of your design with, e.g., Xilinx Vivado? List all checks that you do and for each of them explain what you look for, where, why, what can be wrong, what are the main possible reasons for the issue and how to fix it.

1.4 VHDL variables

In a VHDL model where do you declare variables (in which declarative region) and why?

1.5 Synthesizable VHDL code

Assume the designer of the following VHDL code wanted to design a combinatorial synthesizable adder / subtracter. What do you think of her work? Identify the errors if any and, for each of them, explain why it is an error, what undesirable effect it has and finally, write down a new VHDL code with all the errors fixed.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity addsub is
    port(add_not_sub: in  std_ulogic;
         a, b:        in  signed(31 downto 0);
         s:           out signed(31 downto 0));
end entity addsub;

architecture arc of addsub is
begin
    process(a, b, s)
    begin
        if add_not_sub = '1' then
            s <= a + b;
        elsif add_not_sub = '0' then
            s <= a - b;
        end if;
    end process;
end architecture arc;
```

2 Problem: design of a rotation sensor

Figure 1 represents a rotating disk, two stationary copper brushes (A and B) and a digital system (**ds**).

A and B touch the surface of the disk at the same distance of its center. The sectors of the disk are conducting and isolated one from the other. The white sectors are connected to a 3.3 volts power supply while the striped sectors are connected to the 0 volt ground. The disk rotates clockwise or anticlockwise. During rotation A and B pass alternately over white and striped sectors. They are close enough to each other to fit in the same sector.

ds is synchronized on the rising edge of clock **clk**. **rstn** is its asynchronous active low reset, **a**, **b** are its 2 data inputs and **cw** is its data output. The entity of **ds** is shown in Listing 1.

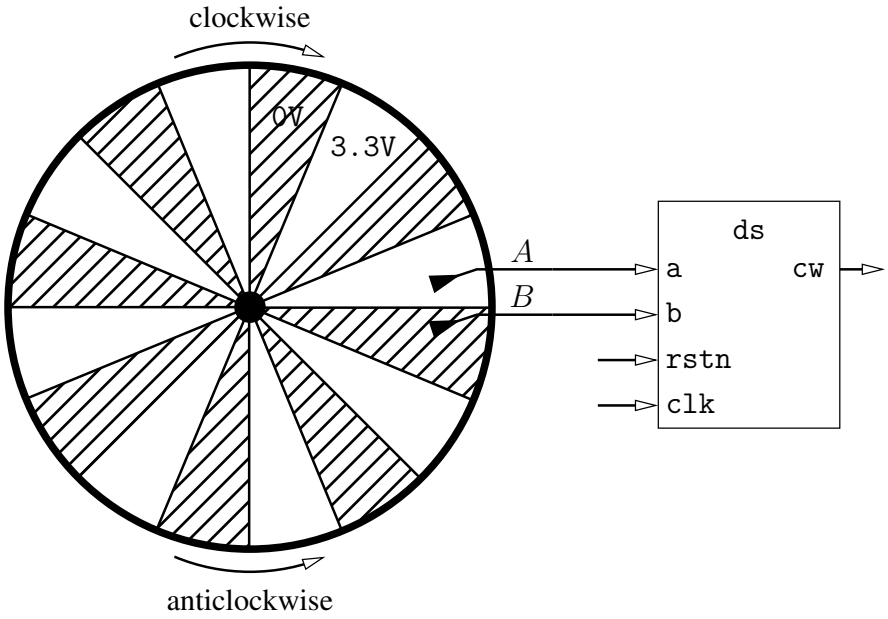

Figure 1: The rotating disk

Listing 1: Entity

```
library ieee;
use ieee.std_logic_1164.all;

entity ds is
    port(clk, rstn, a, b: in  std_ulogic;
         cw:              out std_ulogic);
end entity ds;
```

A and B are connected to the `a` and `b` inputs of `ds`. In the digital world of `ds` when A is over a white sector `a = '1'`, and when A is over a striped sector `a = '0'`. Same for input `b` and brush B. The situation depicted on Figure 1 is thus characterized by `a = '1'` and `b = '0'` but the 3 other combinations of `a` and `b` are also encountered during the rotation.

When the reset is active (`rstn = '0'`), the output of `ds` is low (`cw = '0'`). Else `cw` indicates the direction of rotation of the disk: `cw = '1'` for clockwise, `cw = '0'` for anticlockwise. If the rotation stops `cw` keeps its current value to indicate the last recorded direction of rotation. If the direction changes it can take a few clock periods before `cw` is updated. In normal operation the frequency of `clk` is high enough, compared to the speed of the disk, to ensure that `a` and `b` cannot both change during the same clock period. Starting from the situation depicted on Figure 1, if the disk rotates clockwise at about the maximum speed, the sequence of different values taken by `ab` is $10, 11, 01, 00, 10\ldots$ While, if the disk rotates anticlockwise at about half the maximum speed, the sequence is something like $10, 00, 00, 01, 01, 11, 11, 10\ldots$ If for any reason `a` and `b` both

change during the same clock period the behavior is unspecified and **`cw`** can take value `'0'` or `'1'`, as you wish.

Design a block diagram of **`ds`**, clearly identify the registers, give their size in bits, name them and explain what their role is. Clearly identify the combinatorial parts, name them, and provide a description of their behavior with pseudo code. Clearly draw the interconnections between the different parts. If you use state machines draw their state diagrams, explain whether they are Moore or Mealy state machines, and provide a description of their role.

Write the VHDL 2008 model of the architecture of **`ds`** and comment it: for each process explain if it is synchronous or combinatorial, and clearly identify the parts of your block diagram that it models. What is the minimum number of processes that would be needed to model **`ds`** in VHDL 2008?