Real-Time Imaging and Control
Lab

MSCV/ESIREM

Antoine Lavault
antoine.lavault@u-bourgogne.fr

Lab 2 - VHDL

This goal of this lab are :

- Write basic VHDL descriptions
- Gain proficiency with Vivado and VHDL coding.
- Use a test bench as part of a VHDL development workflow.

You will use Windows for all of the lab sessions.
This lab is not meant to be completed in one session. Take your time and don't rush anything.

1 Basic Components

Problem 1

Create a AND3 with :

1. 3 inputs: a,b,c
2. 1 output: s
3. Concurrency assignment

Problem 2

Create a ADD1[1] with :

1. 3 inputs: a, b, cin
2. 2 output: s, cout
3. Concurrency assignment

[1]Add 2 signals of 1 bit

⌞ Problem 3 ⌝

Create a ADD4[2] with :

1. 3 inputs: a, b, cin
2. 2 outputs: s, cout
3. Concurrency assignment
4. use your previous ADD1 as a component

*Please consider **port map** instructions for instantiating the ADD1 component*

⌞ Problem 4 ⌝

Create a D flip-flop with :

1. 2 inputs: D, clk
2. 1 output: q

⌞ Problem 5 ⌝

Create a synchronous 8-bit counter with asynchronous reset :

1. 2 inputs: reset, clk
2. 1 output: s

⌞ Problem 6 ⌝

Create an 8-bit shift register:

NB: In digital circuits, a shift register is a cascade of D flip flops, sharing the same clock, in which the output of each flip-flop is connected to the 'data' input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the 'bit array' stored in it, 'shifting in' the data present at its input and 'shifting out' the last bit in the array, at each transition of the clock input.

[2] Add 2 signals of 4 bit

2 Intermediate VHDL

We propose in this lab some advanced examples of components to design in VHDL.

NB : all examples should be check using ***test-benchs*** *and simulatuon tools*

Problem 7

Create a multiplexer (mux) with :

1. 2 inputs : data(8),sel(3)
2. 1 output : s(1)
3. 1 CLK for synchronous behavior

Problem 8

Create a VHDL Design that allows the same behavior of the common-cathode BCD[3]-decoder (figure: 2), the 74LS48 IC[4].
A BCD-decoder is used to display the value on a 7-segment display (see figure: 1).

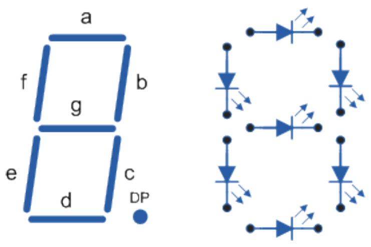

Figure 1: Example of 7 segments display

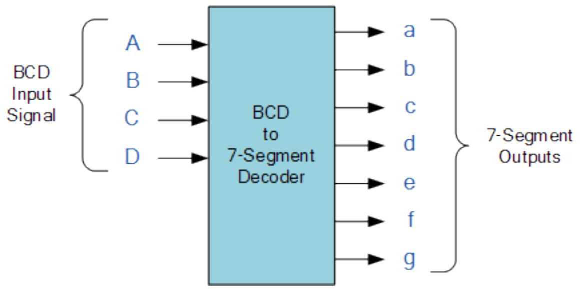

Figure 2: BCD-decoder

[3]Binary-Coded Decimal
[4]`http://pdf.datasheetcatalog.com/datasheet/motorola/74LS48.pdf`

NB: 7-segment LED (Light Emitting Diode) or LCD (Liquid Crystal Display) type displays provide a very convenient way of displaying information or digital data in the form of numbers, letters, or even alpha-numerical characters.
Typically, 7-segment displays consist of seven individual colored LEDs (called segments) within one display package. To produce the required numbers or HEX characters from 0 to 9 and A to F on the display, respectively, the correct combination of LED segments need to be illuminated and BCD to 7-segment Display Decoders.
A standard 7-segment LED display generally has 8 input connections, one for each LED segment and one as a common terminal or connection for all the internal display segments. Some single displays also have an additional input pin to display a decimal point in their lower right or left-hand corner.

Problem 9

Create the state-machine represented by the following pictures (figure: 3):

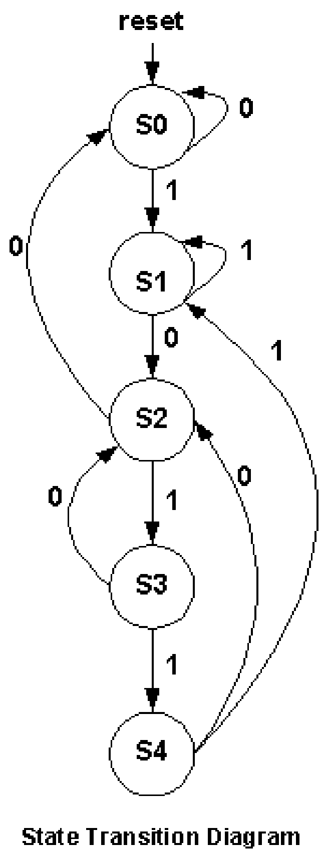

Figure 3: State machine

Hint: you can use custom sub-types to encode the states like

```
type state_type is (idle, state1, state2,...);
signal present_state, next_state: state_type;
```

3 Board implementation

*NB : all examples should be check using **test-benches** and simulation tools before to use the board!!!*

Problem 10

Create a design that allows the display of a value between 0 and F of one 7-segment Display, which will be selected using switches on the board.

Problem 11

Create a design that displays the value from a counter modulo F (i.e., a 4-bit counter) on a particular 7-segment display.
Use the clock of the card as an input for your counter.
A frequency around 5Hz will be sufficient.

Problem 12

Create a design that displays the value of a counter modulo 9999 on a particular 7-segment display.
Use the clock of the card as an input for your counter.
Hint: As written in the board manual, you will have to cycle the anodes if you want to display more than one number (multiplexing).

4 Putting everything together

For this portion of the lab, we want to build a system that will display multi-digit numbers on the seven-segment displays based on the following rules:

- If $btnc$ is **not** pressed, the hexadecimal value displayed on the lower four displays of the seven-segment display is based on the values of all 16 switches values (the toggle switches sw).
- If $btnc$ is pressed, the hexadecimal value displayed on the lower four digits of the seven-segment display is based on the current count value of an automatic counter incrementing at 10 Hz (this counter is expected to increment from 16'h0000 to 16'hFFFF)
- If $btnl$ is pressed (regardless of what $btnc$ is currently), the hexadecimal value displayed on the lower four displays of the seven-segment display is based on the number of button presses made to $btnu$ (using another counter that increments from 16'h0000 to 16'hFFFF).

Remark 1. *We must be careful when counting button pushes because switches bounce (literally), creating a high-frequency toggling signal. A human may not notice this, but a 100 MHz digital system will. The required solution is to "debounce" the signal, so we'll add a module that handles this for us. Switch bounce typically disappears after approximately 10 ms, so one approach to building a debouncer is to make a module that only "passes" the button's value after it has been constant for 10 ms (you can think of this as a form of low-pass filtering).*