EE 271 – Introduction to Digital Circuits and Systems

Lab 2: Digital Design using FPGAs

Lab Objectives

1. Follow the steps of designing a combinational logic system.
2. Use ModelSim to verify the design.
3. Execute the design on the remote FPGA lab.

Task 1: Design Problem – Multi-level logic.

In order to speed up the processing of returns at the XYZ store, the customer service department wants an electronic detector device. Its goal is to both determine those items that have been discounted so that the proper rebate can be calculated, as well as help find shoplifters returning their ill-gotten gains.

There are six products being sold. The UPC code for each is shown, as well as whether it was ever on sale (i.e. sold at a discounted price), and whether it is expensive, and thus is marked when sold.

Item Name	UPC Code	Discounted?	Expensive?
Shoes	0 0 0	No	Yes
Jewelry	0 0 1	No	No
Bike	0 1 1	Yes	No
Business Suit	1 0 0	No	Yes
Winter Coat	1 0 1	Yes	Yes
Socks	1 1 0	Yes	No

Note that since there are only 6 items, not all UPC codes are used. The behavior of your circuit for these situations is unimportant (i.e. Don't Care).

You will be given the UPC code of the item under test (signals "U", "P", and "C" for simplicity), and a detector will check for a secret mark ("M"). Your circuit should have one "discounted" light that lights up whenever a discounted item's UPC code is applied, as well as a "stolen" light that lights up whenever a theft is detected.

XYZ store has a special method for finding shoplifters. Whenever they sell an expensive item, they put a secret mark onto it. Thus, expensive items that are purchased are specially marked,

while stolen expensive items will not be so marked (inexpensive items are never marked since it is too expensive to mark everything sold). When there is a return, we want to make sure someone didn't steal the item, then return the stolen item for cash.

With the rules given, there are four cases for the stolen light logic to consider:

- An expensive item with the mark is not stolen.
- An expensive item without the mark is stolen.
- A non-expensive item without the mark is not stolen.
- A non-expensive item with the mark will never occur, so is a Don't Care.

Design this circuit and minimize the output equation(s) and write a SystemVerilog code for the design. Implement the design and simulate it in Quartus and ModelSim. **Name the folder of your Quartus project lab2a**. Finally, download the design to the remote FPGA and test its functionality. Please use switch Sw9, Sw8, Sw7 for U, P, C, respectively, and Sw0 for the secret Mark.

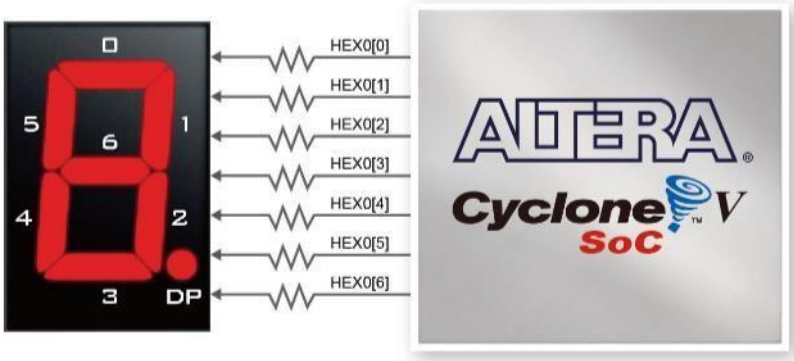
Task 2: Seven-segment display

The code for the seven-segment display is given below. **Create a copy of your lab2a project and call it lab2b**, then enter the code below into a new System Verilog file. Then, create a new module that instantiates the seg7 code taking input from SW9 – SW6 and displaying on Hex0 the corresponding digit. Simulate in ModelSim.

```
module seg7 (bcd, leds);
input  logic  [3:0] bcd;
 output logic  [6:0] leds;

 always_comb begin   case
(bcd)
   //          Light: 6543210
   4'b0000: leds = 7'b0111111; // 0
   4'b0001: leds = 7'b0000110; // 1
   4'b0010: leds = 7'b1011011; // 2
   4'b0011: leds = 7'b1001111; // 3
   4'b0100: leds = 7'b1100110; // 4
   4'b0101: leds = 7'b1101101; // 5
   4'b0110: leds = 7'b1111101; // 6
   4'b0111: leds = 7'b0000111; // 7
   4'b1000: leds = 7'b1111111; // 8
   4'b1001: leds = 7'b1101111; // 9
   default: leds = 7'bX;   endcase
 end
endmodule
```

As mentioned in lab1, the 7-segment display on the board is ACTIVE LOW. That means putting a FALSE on the wire makes it light up, while a TRUE means that light is off. You will have to adjust your design accordingly. (The figure below gives you a correct mapping of the HEX digits to its corresponding positions on the display.)



Finally, create a new module that hooks task1 with task2 together, so the system simultaneously computes the HEX0 display and the Sale and Stolen lights. Since we only have six items, we only need 3 switches for UPC. Change your design to allow SW8-SW6 to represent the UPC. You still need SW0 for the secret mark to compute the 'stolen' and 'sale' lights but it doesn't have anything to do with the HEX0 display. Test and debug with ModelSim, then load onto the remote FPGA board. (If you only feed SW8-6 into your seg7 module, the observable HEX patterns should only range from 0 to 7).

(Extra Credit) Task 3: Design Problem – UPC code to display

In task1 we built a system that took in a UPC code and output whether the item was on sale and whether it was stolen. Your challenge is to add a display on Hex5 – Hex0 that describes each product when it is entered. You can go with the original products listed in task1 or modify the table with names for different products that would make a good text display on the HEX displays. For example, you could put in "Dress Shoes" as an item and have the hex display show "ShOE". Note that some letters are not representable in HEX. For example, i or t, so you can choose to avoid those. Determine your items and Hex displays. You can use any or all of the lights on the hex display you choose, upper and lower case, pictograms, …etc. Be creative!

Once you have figured out what you'll display, create a high-level design for the circuit. It will be similar to the seg7 module, with 3 inputs (U, P, and C), but up to 6 7-bit outputs (for each of the 7seg displays). Create a module that combines task1 with this task such that the system simultaneously computes the HEX display, and the Sale and Stolen lights. Test and debug with ModelSim, then load onto the remote FPGA board.

ModelSim Tip: When you put your entire design together and simulate in ModelSim, you'll probably need some help organizing all your signals. A couple of tips:

1.) ModelSim can have signals from multiple modules displayed at the same time. Simply select in the left pane the module whose signals you want to display, then drag the signals you want from the middle pane to the waveform window.
2.) To order the signals, you can click and drag names in the waveform window.
3.) To organize signals for each module, highlight all of the signal names related to one module in the waveform window, right-click on one of the signal names, and select

"Group". Give it a memorable name, then hit okay. You can now move the signals as a group, and hide/expose them easily.

Lab Demonstration and Submission Requirements

- Submit a video demonstrating the tasks. Your video is expected to be around 1-2 minutes. In the video demonstrate the functionality on the board using the switches and LEDs and the 7-segment display.
- Submit a short lab report (about 3 pages and it's ok if you go above 3 pages) that should include:
  Results
  - Include screenshots of ModelSim results for all tasks
  - Describe what you tested in the simulation, and what the results in the screenshot show
  - Give a brief overview of the finished project, compared to what was asked
- Submit the SystemVerilog files (files with extension .sv). Make sure to follow the commenting guide provided. **A significant amount of grade will depend on the commenting style.**
- Submit your report, video and programs to Canvas.