Lab #5: Sequential Logic Design using VHDL

Introduction

In **Lab #4** you designed a) a **Counter** and b) a **Finite State Machine (FSM)**, and you implemented these designs using schematic diagrams within Quartus. In **parts 5.1** and **5.2** of this lab, you will implement the same counter and a new FSM in **VHDL** and simulate the behavior of your code in ModelSim before programming your design on the board. In **part 5.3** you will design and implement another FSM, this time to create a '**Walking 1**' where the active '1' appears to move across the DE10 board's LED's as the FSM is clocked.

5.1 Counter Design (in VHDL)

In Lab#4, you designed and implemented your personalized counter. The aim of this part is to implement the same counter using VHDL.

1. Create a new folder called **Lab5P1**, and download and unzip **Lab5P1.zip**, which includes a template file **myCounter.vhd** and a simulation do file named **Lab5P1.do**, to this new folder.
2. Also create a new **Project** called **Lab5P1** in this folder, remembering to set the device to be **Cyclone V**, **5CSXFC6D6F31C6**, and setting the **Assignments > Device > Pin Options** > **Voltage** to **3.3-V LVTTL**.
3. Select **Project > Add files in Project** to add your downloaded **myCounter.vhd** file to your Project. This contains an (incomplete) template for the VHDL code for implementing a Counter. Set this file as the top-level entity.
4. **Open** the given VHDL file and edit it to implement your counter:
   - Read through the entire template and all the comments before you start.
   - Note from the entity input/output signals that:
     - Pushbutton **Key[0]** provides the falling-edge triggered clock pulse.
     - The three FF outputs in the state vector **Q** appear on **LEDR**[2]..[0].
     - **SW[0]** is an additional input used to **Reset** the state Q back to first state.
   - Compile and fix errors if any.
5. Now **Simulate** your design using **ModelSim** to see if your code behaves as you would expect:
   - Select **Tools > Run Simulation Tool -> RTL Simulation** to bring up **ModelSim**
   - Within ModelSim, **Simulate >Start Simulation** and then select **myCounter** from within the **work** library
   - If necessary, click **View > Wave** and **View > Objects** to open the Waveform and Objects windows
   - Drag or **R-click > Add Wave** the signals of interest from the Objects window into the Wave window.
   - Run the simulation using the inputs specified in your downloaded Lab5p1.do file, by typing **do ../../Lab5P1.do** in the Transcript/Command window.
   - Zoom the Wave window nicely, and check if the correct counting sequence is generated. If you need to change your VHDL code, close ModelSim and return to Quartus to make changes. Then compile again.
   - After you are sure the simulation waveform shows the correct behavior, ask the instructor/TA to check.

- Then use **the Snipping** tool to capture a **picture** of the wave window and save it to your Lab5P1 folder. Finally, select **Simulate > End Simulation** to exit the simulation. Close ModelSim.

6. In Quartus, open **Pin Planner**, and assign pins to the input and output signals. Compile again.
7. Program and **test** your circuit on the **DE10** board.
   - Use the **Reset** signal to place your counter to the first state, then clock your counter and **verify** the counter sequence on the LEDRs

Ask the instructor/TA to check off your work for 5.1.

5.2 Finite State Machine (FSM) design.

(Pre-Lab) Get a specification of pattern matching FSM from the instructor, and work out a state diagram and the expected output based on the given input bit stream on the paper.

Ask the instructor/TA to check off your work for the pre-lab assignment.

1. Create a new folder called **Lab5P2;** download **Lab5P2.zip** from the BB and unzip it in the folder. You will see two provided files: myFSM.vhd and Lab5P2.do.
2. Load Quartus, and create a Project called **Lab5P2**, and add the given file **myFSM.vhd** file to the project. Right click on myFSM.vhd and set it as top-level entity.
3. **Open** the VHDL file and **complete** it to implement your FSM. Read through the entire file first.
4. Now **Simulate** your design using **ModelSim** to see if your code behaves as you would expect.
   - Select **Tools > Run Simulation Tool -> RTL Simulation** to bring up **ModelSim**
   - Modify **Lab5P2.do** to force the input **SW(1)** to be the given test bits of your FSM. Each input bit lasts for one clock cycle, i.e., force it to be a new value every 20 ns.
   - Run the simulation using **Lab5P2.do**.
   - Zoom your results nicely, and check that the FSM responds to the simulated inputs as specified in the original State Transition Table. Save a picture of the wave window and exit the simulation.
5. Assign pins to the input and output signals. Compile, program and test your program on a DE10 board.
   - Force the FSM into the first state using the **Reset** signal (SW(0)). Then apply the given input sequence to the input SW(1) at each clock cycle. During the process, compare what you observe on LEDR(3) with the expected output.

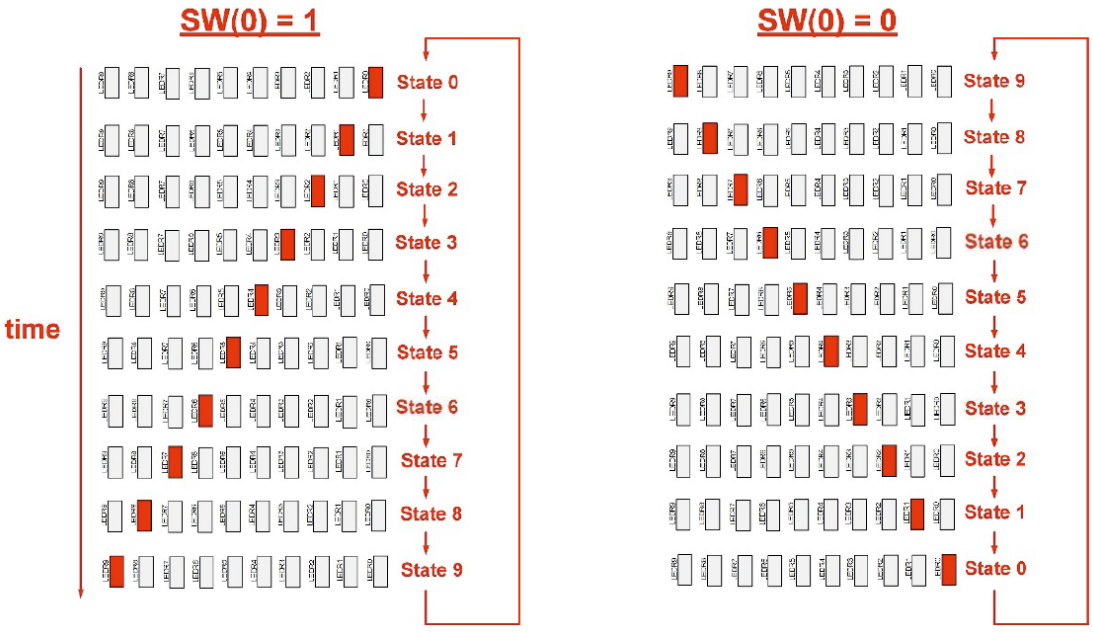
Ask the instructor/TA to check off your work for 5.2.

5.3 Walking LED Finite State Machine design.

A walking 1 pattern is a pattern that asserts one, and only one, signal at any given time within a group of signals. When this is displayed on LEDs, it appears that the asserted LED 'walks' across the display, as shown in the picture on next page. This type of pattern can be created using a finite state machine in which each state represents one of the output patterns.

- In this lab exercise, you will create a state machine that will produce a walking 1 pattern within a 10-bit output vector which will drive the ten red LEDRs on the DE10 board.
- When SW(0) is set to '1', the lighting LEDR walks towards left; Otherwise, it walks towards right. The direction can change when the lighting LED is in the middle. When the LED reaches the last LEDR, it goes back to the first LED for that direction, as shown in the picture.
- **SW[1]** is an input to **Reset** the state Q back to first state
- The push button **KEY[0]** will be used to clock your state machine from one state LED position to the next.
- **Suggestions:**
  - Create a new folder and a new Quartus project for this part.
  - Download and unzip Lab5P3.zip, which includes a starting VHDL file and a .do file.
  - Following a similar style of your myFSM.vhd architecture, develop the state machine behavior using CASE and IF statements.
  - In the CASE statement, your next state should depend on the direction SW(0), which functions similar to the input **x** in myFSM.vhd of Section 5.2.
  - Use the defined constant names for states.
- **Simulate** your code in ModelSim.
- Assign pins to the port signals, and compile. Program and test your FSM design on the DE10 board.

Ask the instructor/TA to check off your work for 5.3.



Submission for Lab 5:

After you have been checked off all the parts by the instructor/TA, zip all the Lab5 project folders into one zip file, name it Lab5_<your last name>.zip, and submit it in the Blackboard. Please submit only one zip file, and ensure everything is contained in the single zip file.