

Intro to Digital Logic, Lab 3

Finite State Machines

Lab Objectives

In this lab, you will put what you learned about FSM into action. You will run the sequence detector explained in class and then you will design a new FSM and run it on the DE1-SoC board.

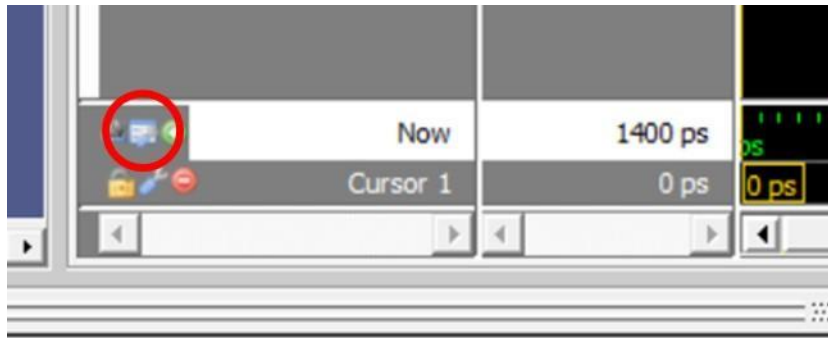
Task 1 – Sequence Detector

You are provided by the code of the sequence detector explained in class that detects the sequence 101. The whole project is provided to you on canvas. Download the zip file called “fsm”, extract it to your computer then open the project in Quartus. Please do the following:

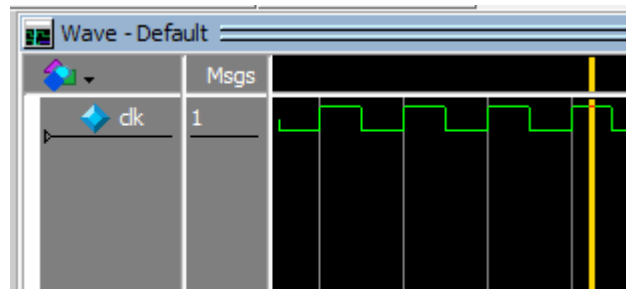
1. Run the simulation in ModelSim. However, before you do that, comment the “clock_divider” module in the DE1_SoC module. This clock divider is needed to run the code on the FPGA at a slower rate. In simulation, it is not needed and if you forget to comment it, the simulation will look confusing if it doesn’t cause any other problems.
2. Uncomment the “clock_divider” module and download your design to the board and test it. This design uses KEY0 for reset (press to reset the circuit) and SW0 as input, while LEDR5 shows the clock, and LEDR0 shows the output of the FSM.
3. Modify the code to create a sequence detector that detects the sequence ‘1101’. The code provided doesn’t have much comments. Please fix that and write comments according to the guidelines provided on canvas. Before modifying the code, you need to come up with a Mealy FSM diagram that detects the sequence 1101.
4. Run the simulation in ModelSim (don’t forget to comment the clock_divider for simulation). Take a screen shot of the waveforms and save it for your deliverables. Please take screen shot of the waveforms only not the whole modelsim windows as it will be hard to view the waveforms.
5. Run the design on the board (don’t forget to use the clock_divider) and test it.
6. The deliverables for this task are the commented code and state diagram for the ‘1101’ sequence detector, and a screen shot of its simulation.

Notes:

- There are two ways to assign variables in SystemVerilog, and it is important you understand which to use and when. When you are assigning a variable in an “always_comb” block, or after an “assign” statement, you should ALWAYS use “=” to assign that variable a value. If you are assigning a variable in an “always_ff @(posedge clk)” block, use “<=” to assign the variable a value. These rules will hold true for all your labs.
- **ModelSim Tip:** Since clocks are so important to FSMs, it can be useful to set the grey gridlines in the wave display to line up with the positive edge of the clocks. Look in the lower-left corner of the waveform window, and click the “Edit Grid & Timeline Properties...” icon:



If you set the “Grid Period” to 100ps, which is the clock period, and set “Grid Offset” to 50ps, the grid lines will now line up with the positive edge of the clock:



Task 2 – Wind Indicators

Your task is to build special wind indicators. Your circuit will be given two inputs (SW[0] and SW[1]), which indicate wind direction, and three lights to display the corresponding sequence of lights:

SW[1]	SW[0]	Meaning	Pattern (LEDR[2:0])
0	0	Calm	1 0 1 0 1 0
0	1	Right to Left	0 0 1 0 1 0 1 0 0
1	0	Left to Right	1 0 0 0 1 0 0 0 1

For each situation, the lights should cycle through the given pattern. Thus, if the wind is calm, the lights will cycle between the outside lights lit, and the center light lit, over and over. The

right to left and left to right crosswind indicators repeatedly cycle through three patterns each, which has the light “move” from right to left or left to right respectively.

The 2 switches will never be true at the same time. The switches may be changed at any point during the current cycling of the lights, and the lights must switch over to the new pattern as soon as possible (however, it can enter any point in the other pattern’s behaviors).

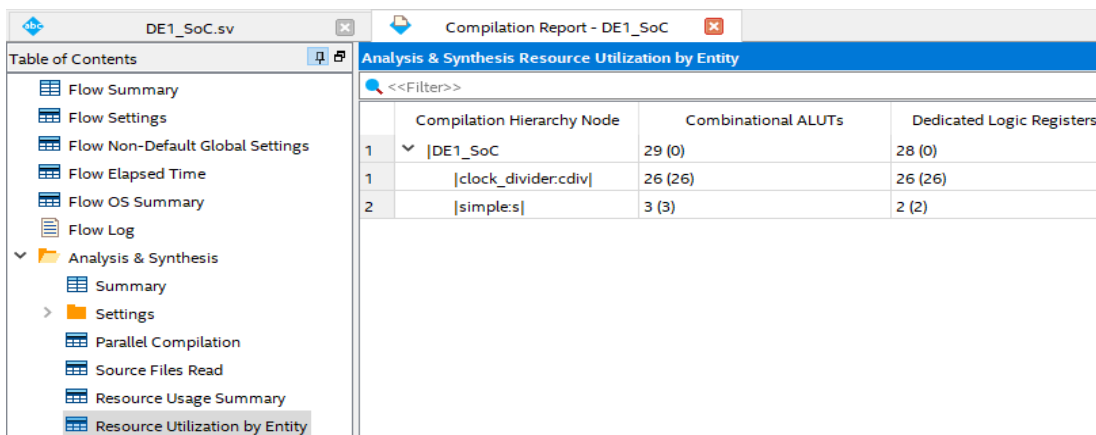
Please do the following:

1. Create a state diagram for the wind indicators.
2. Copy the project from task 1 into a new folder and change the modules names (and the corresponding files names) as necessary and modify the code to match your wind indicators state diagram.
3. Simulate your design in ModelSim and take a screen shot for your deliverables.
4. Download the design to the board and test it.

Remember: you won’t be able to perceive LEDs blinking 50,000,000 times per second, so you will need a slower clock. The clock divider from the first task will work well for the wind indicators lights. However, simulations won’t work well with the clock divider in use, because the first positive edge of the divided clock won’t happen until after ~34 million clock cycles! As such, you should simulate your design separately from the top-level module.

If you would like to simulate your top-level module, you need to temporarily remove the clock divider, and instead connect `CLOCK_50` as the clock input to your submodules.

5. Compute the size of your design in terms of #FPGA logic and DFF resources without the cost of the clock_divider. To do this, perform a compilation of your design, and look at the Compilation Report. In the left-hand column select Analysis & Synthesis > Resource Utilization by Entity. The “Combinational ALUTs” column lists the amount of FPGA logic elements being used, which are logic elements that can compute any Boolean combinational function of at most 6 inputs. The “Dedicated Logic Registers” is the number of DFFs used. For each entry, there is the listing of the amount of resources used by that specific module (the number in parentheses), and the amount of resources used by that specific module plus its submodules (the number outside the parentheses).



The screenshot shows the ModelSim interface with the 'Compilation Report - DE1_SoC' open. The left sidebar shows the 'Table of Contents' with 'Analysis & Synthesis' expanded, and 'Resource Utilization by Entity' selected. The main window displays a table titled 'Analysis & Synthesis Resource Utilization by Entity'.

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers
1	▼ DE1_SoC	29 (0)	28 (0)
1	clock_divider:cdiv	26 (26)	26 (26)
2	simple:s	3 (3)	2 (2)

To compute the size of your FSM, add the numbers outside the parentheses for the entire design under “Combinational ALUTs” and “Dedicated Logic Registers”. Subtract from that the same numbers from the “clock_divider” line. Include the size in the results section of your lab report.

Lab Demonstration and Submission Requirements

- Submit a video demonstrating task2. Your video is expected to be around 1-2 minutes. In the video demonstrate the functionality on the board..
- Submit a short lab report (about 3 pages and it’s ok if you go above 3 pages) that should include 3 main sections, detailed below.

Procedure

- For both tasks, describe how you approached the problem and include state diagrams.

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

Appendix

- Include screen shots of your code
- **On Padlet**, write about a problem you had in the lab and the fix to it, share a tip or trick you learned while working on the lab. You can also share an aha moment that you discovered while working on the lab. Avoid duplicating comments made by your classmates. NO videos for this padlet task, please use textual comments. The link to the padlet is [here](#)
- 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.