CS223 Laboratory Assignment 5
FSM for Traffic Light Controller

Lab date and time:

Section 1,2,3,4: 01.05.2020 Friday 08:40-12:25

Location: EA-Z04 (Zoom Meeting)
Groups: Each student will do the lab individually. Group size = 1

Lab Assignment

In a community, people need traffic lights to slow down drivers from going too fast. To reduce danger at the intersections, time for switching red light to green light and green light to red light should be regulated carefully. Studies show that if both lights at an intersection remain red for a certain amount of time and afterwards if one light turns green, huge amount of traffic accidents can be prevented.

In this lab, you are going to design a traffic light controller (FSM) that is similar to the example in pages 124-129 of the text book, but with a small modification. The system will control the traffic at the intersection of two roads as shown in Figure 1. There are two traffic sensors SA and SB installed on Road A and Road B, respectively. Each sensor will indicate TRUE if traffic is present and FALSE if the road is empty. There are two traffic lights LA and LB to control the traffic on these roads. Every 2 seconds, the controller examines the traffic and decides what to do next. On each rising edge of the clock with a 2-second period, the lights may change based on the traffic sensors. If a sensor output is TRUE the lights do not change until it is set to FALSE (as long as traffic is present on a road, the lights do not change). While a light is green, if its sensor becomes FALSE that light will turn to yellow first and then turn red. Both lights on two roads will remain red for 2 seconds (one clock cycle). Then the red light on the other road will turn yellow for 2 seconds and after that it will turn green. In other words, traffic lights LA and LB will be GR, YR, RR, RY, RG (G:*Green*, Y:*Yellow*, R:*Red*) on consecutive clock cyles.

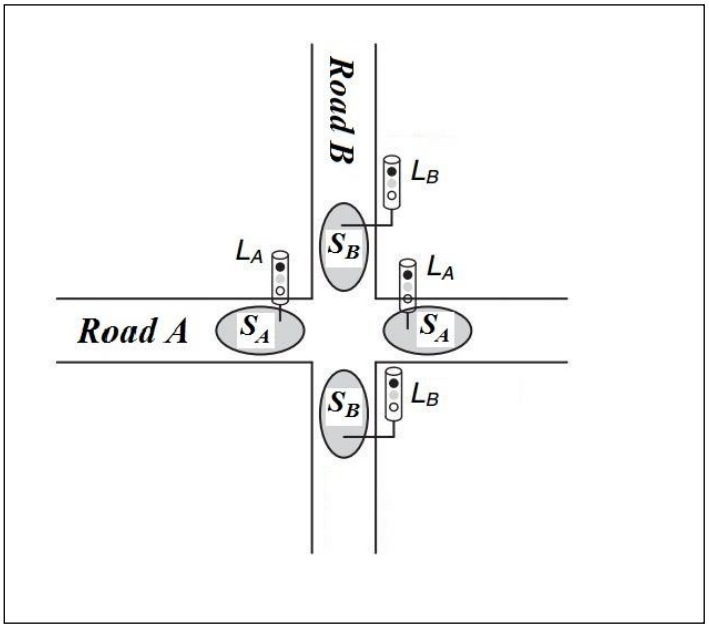

Figure 1: Traffic Light System

Design an FSM for the traffic light controller in the form of Moore machine as shown in Figure 2.

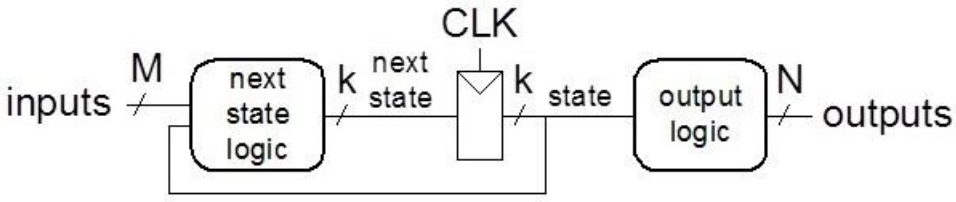

Figure 2: Moore machine

Prepare your circuit schematics, SystemVerilog models and testbenches to be included in a Lab Report that has a cover page and pages for the schematics and SystemVerilog codes. You can ask your questions about the lab assignment to TAs during Zoom Meeting. The **Lab Report** must be uploaded to Unilica after the lab. The content of the report will be as follows:

(a) A cover page including: course code, course name and section, the number and title of the lab, your name-surname, student ID, date.

(b) How many flip-flops do you need to implement this circuit?

(c) Sketch the state transition diagram, write the state transition table and output table with binary encodings, and write the Boolean equations for the next-state and output logic.

(d) Write a SystemVerilog module for the FSM you designed in part (c) and a testbench for it.

You can refer to the slides of Chapter 4 in Unilica while preparing your SystemVerilog modules and testbenches.

Additional pre-lab work:

You should read the following documents (available on Unilica) to be familiar with steps of design flow (Simulation, Synthesis, Implementation, Bitstream Generation, Downloading to FPGA board), using Xilinx **Vivado** tool. You can download, install and practice working with Xilinx Vivado on your own computer with free webpack license.

- Suggestions for Lab Success.
- Basys 3 Vivado Decoder Tutorial.
- Vivado Tutorial.
- Basys 3 FPGA Board Reference Manual.

Implementation on FPGA

In this part, you will implement your code on FPGA board. You don't need to connect your **Basys 3** board to the Beti board. Working with standalone Basys 3 and having it connected to your computer is enough for this lab. There are some user switches and LEDs available on Basys 3 which you can use them to test your design.

- *Create a new Xilinx Vivado Project. Use appropriate names for files and folders, keeping the project in a directory where you can find and upload it after the lab.*

(1) **Simulation**: Using the SystemVerilog module for the FSM and testbench code you wrote in preliminary part (d), simulate your circuit and verify that it works correctly. In the simulation, try all possible combinations of SA and SB inputs and observe the LA and LB traffic lights.

(2) **Program the FPGA**: Now, follow the Xilinx Vivado design flow to synthesize, implement, generate bitstream file, and download your FSM to Basys 3 FPGA board.

(3) **Test your design**: Using the slide switches and LEDs (on Basys 3) that you have assigned in constraint file (.xdc), test your FSM. When you are convinced that it works correctly, upload your files to Unilica.

- Remember that the clock frequency of Basys 3 oscillator is 100 MHz. Slow down this clock frequency to a 2-second period to be able to see the changes in the lights. You can use the clock divider code written for Lab4 and modify that code slightly to change the output frequency to 0.5 Hz.
- Use LEDs on Basys 3 board for outputs of LA and LB traffic lights.

  Red: * (one LED is ON)
  Yellow: ** (two LEDs are ON)
  Green: *** (three LEDs are ON)

- The SA and SB sensors will be two leftmost switches (SW15, SW14). The sensor output will be TRUE as long as the corresponding user switch is placed in the ON position.

Submit your code for MOSS similarity testing

Finally, when you are done, you need to upload the file *StudentID_SectionNumber.txt* created in the Implementation on FPGA part. Be sure that the file contains exactly and only the codes which are specifically detailed above. If you have multiple files, just copy and paste them in order, one after another inside text file. Check the specifications! Even if you didn't finish or didn't get the SystemVerilog part working, you must submit your code to the Unilica Assignment for similarity checking. Your codes will be compared against all the other codes in all sections of the class, by the MOSS program, to determine how similar it is (as an indication of plagiarism). So be sure that the code you submit is the code that you actually wrote yourself! All students must upload their codes to the 'Unilica Assignment' specific for this lab until the last submission date. **Check submission time and don't miss it after the lab**.