CS223 Laboratory Assignment 4

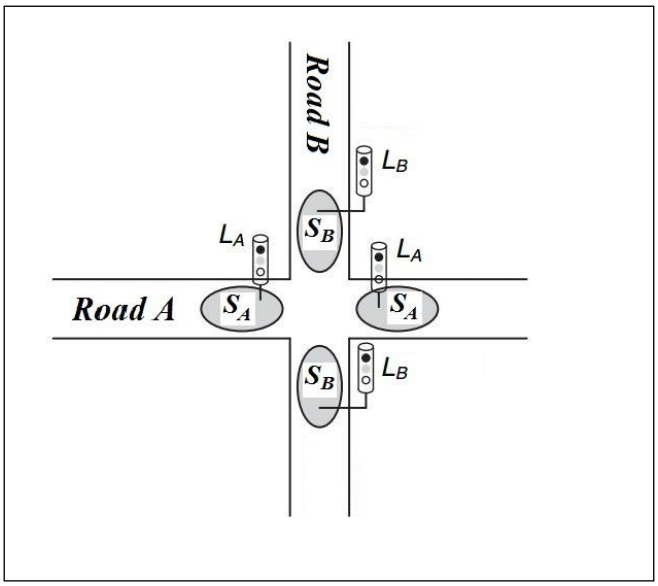
Traffic Light System

Section 1:	08.04.2021 Thursday	13:30-17:20
Section 2:	06.04.2021 Tuesday	08:30-12:20
Section 3:	07.04.2021 Wednesday	08:30-12:20
Section 4:	08.04.2021 Thursday	08:30-12:20

Location: EA Z04 (in the EA building, straight ahead past the elevators)
Groups: Each student will do the lab individually. Group size = 1

Traffic Light System

In a community, people need stop-signs and 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. Traffic light system is similar to the example in pages 124-129 in the text book, but with some improvements. As depicted in below figure, in the junction where road A and road B intersect, there are sensors SA and SB, installed in each road to sense the traffic. Each sensor will be TRUE if traffic is present and FALSE if the road is empty. There are two traffic lights LA and LB to control the traffic. The lights may change every 3 seconds, depending on the sensors. If a sensor is TRUE, the lights will not change until it is set to FALSE. If a light is green and sensor is FALSE it will turn to yellow and then red. Both lights will be red for 3 seconds, then red light will turn yellow for 3 seconds, and then turn green.



Preliminary Report (40 points)

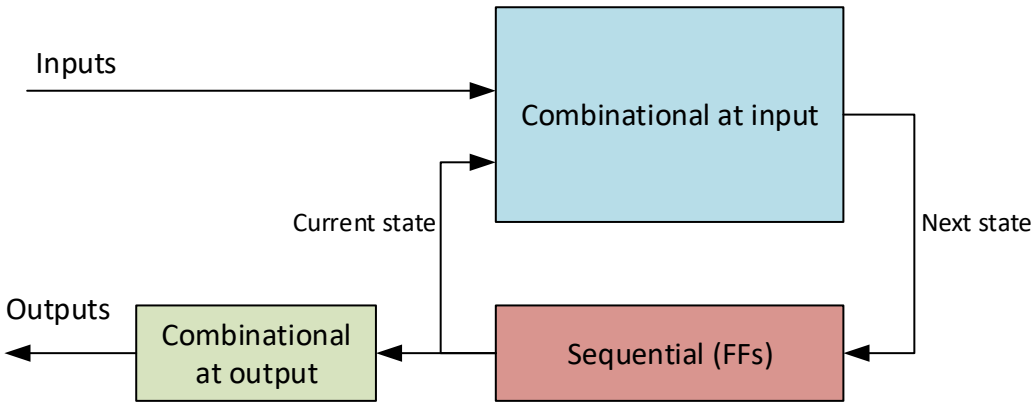
A number of tasks in the today's lab need preparation. The designs and Systemverilog modules should be prepared in advance, and assembled neatly into a Preliminary Design

Report. The preliminary report should include the following and must be uploaded to Moodle before the start of lab for each section.

a) Sketch your improved Moore Finite State Machine (FSM) transition diagram, state encodings, state transition table, output table, next state, output equations and your FSM's circuit schematic.
b) How many flip-flops do you need to implement this problem?
c) As depicted in below figure, you will have two combinational functions:

next_state = Func-Blue(inputs, current_state)
output = Func-Green(current_state)

You have learnt already that combinational functions can be implemented in various ways: Sum of Product (SOP), Product of Sum (POS), by multiplexers or by decoders. Implement the combinational logic at input (blue block which generates next_state) by using decoders. Implement combinational logic at output (green block which generates outputs) by using multiplexers. You are allowed to use only inverters other than the multiplexers and decoders plus OR gates, where needed. Implementing in a different way than what is specified is not acceptable.



A general Moore FSM

Simulation (20 Points)

Make a Xilinx Vivado project for your module, prepare a testbench for it and simulate it. In the simulation, try all possible variations through SA and SB sensors and observe the LA and LB traffic lights. When you are done, call your TA to evaluate your work.

Implementation on FPGA (40 Points)

In this part you are going to implement your design on FPGA and have a demo. You may need a new project for this part, because you need to modify the clock.

1) Slow down the clock rate to around 1/3 Hz (3 seconds) to be able to see the change in the lights (It is not necessary to make it precisely 3 seconds). Remember again that the

clock rate of BASYS3 is 100 MHz. Use the given clock divider code and, if needed, modify the code slightly to have the desired clock output frequency.

2) Use LEDs on BASYS3 board for LA and LB traffic lights.
   Red: * (one LED)
   Yellow: ** (two LEDs)
   Green: *** (three LEDs)
3) The SA and SB sensors will be the two left side switches. The sensor will be active as long as the switch is set to 1.

Now test your code and show the result to your TA.

```
module ClockDivider(
   input clk_in,
   output clk_out
   );

logic [25:0] count = {26{1'b0}};
logic clk_NoBuf;

always@ (posedge clk_in) begin
        count <= count + 1;
end

// you can modify 25 to have different clock rate
assign clk_NoBuf = count[25];

BUFG BUFG_inst (
  .I(clk_NoBuf),  // 1-bit input: Clock input
  .O(clk_out) // 1-bit output: Clock output
);

endmodule
```

Use below lines in your constraint file:

```
#------------------------------------------------------------
#---------------------- Clock signal ------------------------
#------------------------------------------------------------
set_property PACKAGE_PIN W5 [get_ports clk_in]
set_property IOSTANDARD LVCMOS33 [get_ports clk_in]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk_in]
```

Submit your code for MOSS similarity testing

Finally, when you are done and before leaving the lab, you need to upload the file *StudentID_SVerilog.txt* created in the Implementation with 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 Moodle 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 code that you actually wrote yourself! All students must upload their code to the 'Moodle>Assignment' specific for your section. Check submission time and don't miss it before leaving the lab. After taking a backup of your work, don't forget to delete it from computer. Because students of other sections will work with your system too.

Clean Up

1) Clean up your lab station, and return all the parts, wires, the Beti trainer board, etc. Leave your lab workstation for others the way you would like to find it.

2) CONGRATULATIONS! You are finished with this Lab and are one step closer to becoming a computer engineer.