Digital Logic II

CST 133

Lab 7 - State Machine Design

Traffic Light controller problem statement

In this lab, you will design and implement a state machine for a traffic controller with the following specifications:

A little used farm road intersects with a multi-lane highway; a traffic light controls the traffic at the intersection. Detectors are placed along the little used farm road which causes the signal to change only if there is a vehicle wanting to cross the highway. The light controller is implemented to maximize the time the highway light remains green.



***“reset”;*** is a synchronous signal that forces all lights in all directions to be red for a 1 second time interval after the signal is ‘0’. After this reset, normal operation of the traffic system will begin. There is a ***sensor*** on the farm road to determine if a car is present. If the ***sensor*** indicates that there is a car on the farm road the highway light goes yellow for 2 seconds before turning red. The farm road then turns green for 3 seconds, yellow for 2 seconds before returning to red, allowing the car to enter the highway from the farm road. When the farm road light returns to red, the highway light will return to green.

***For timing, use 5 switch flips or button presses for each second.***

***Use two external RGB LEDs.*** One RGB LED will represent the main road’s stop light. The second RGB LED will represent the sensored road stop light (farm road).

State transition diagram

Develop a state transition diagram for this problem. In this state transition diagram, be sure to indicate the following:

1. All signals pertinent to the operation of the state machine, including counters, reset signal, farm road sensor, lights, etc.
2. All states needed to implement state machine drawn as circles.
3. All state transitions drawn as arcs, with an arrow pointing to the next state.
4. Transition arcs must have the transition inputs and outputs.(Note that for this state machine, it is easiest to have outputs of state colours in the state themselves.

State transition table

Form a state transition table similar to the one shown in lecture in the ***Introduction to Finite State Machines***slide package based on YOUR state diagram. You should use one-hot encoding to make your output and next state equations easier to deal with.

Note that in this table, you’ll have 6 outputs- 3 for the farm light and 3 for the highway light.

Next state equations

Please list the next state equations for all of your states.

Output equations

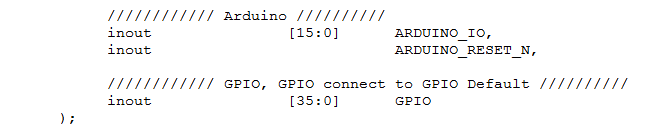
Please list the output equation for the following outputs: red\_farm, green\_farm, yellow\_farm, red\_highway, green\_highway, and yellow\_highway.

**Implement and simulate this circuit using Quartus software as follows:**

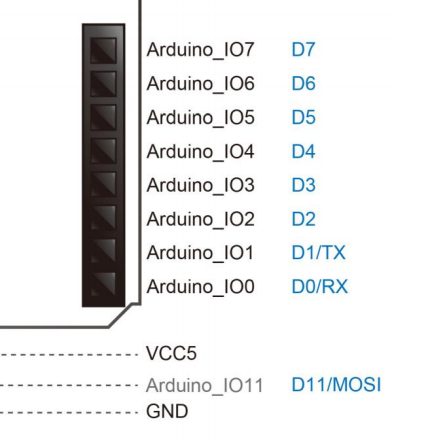
1. Make a copy of the generic top project for the de10-lite board. Use this as your project file.
2. Write a Verilog file that instantiates the state machine for the traffic controller. Use the IntelFPGA user encoded state machine template on the assignment webpage. The state machine should be behaviorally design for the traffic light controller. Don’t use the equations you generated in the last part of the lab.
3. Instantiate your state machine Verilog file in the top level file. Connect inputs to switches and outputs to lights.
4. Compile your code and use the RTL Viewer to examine the implemented circuit. Verify that the traffic controller is implemented correctly.
5. Use ModelSim to simulate you design and verify proper operation of the traffic controller.
   1. Show that your design works when
   2. There is no car on the sensor. (All inputs and outputs)
   3. There is a car on the sensor. (All inputs and outputs)
   4. There is a reset signal
6. Submit your RTL viewer screenshot and Simulation screenshots to the course website. Make sure to properly notate the simulation screenshot.
7. Configure your *design and download* it to the development board and then perform a physical test on your design. Use the signal list below to determine pins for connection to your input and output signals.

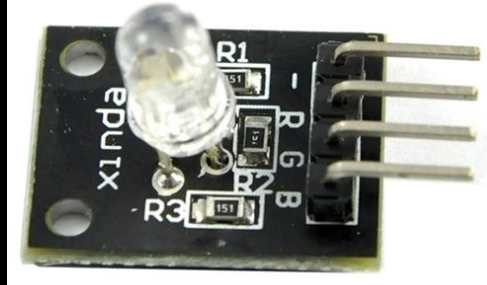
\*\*\*\*Disregard 8 and 9, ship to 10:

1. Note that for the DE10-Lite, you’ll need to use two external RGB LEDs as shown below.
   1. Configure Arduino\_IO7 down to Arduino\_IO0 as outputs (Change the inout to output).



* 1. Since this RGB LED is common cathode, set the – pin on each RGB to logic level 0.
  2. Each time you want to illuminate an LED, set logic level 1 to that output pin.
  3. Connect one RGB led to Arduino\_IO7 down to Arduino\_IO4. Connect the other RGB LED to Arduino\_IO3 down to Arduino\_IO0.





|  |  |  |  |
| --- | --- | --- | --- |
| **Switch 1** | **Signal** | **OUTPUT** | **Signal** |
| **SW0** | sensor | RGB1 Red | R Hwy |
|  |  | Create the colour yellow. | Y Hwy |
|  |  | RGB1  Green | G Hwy |
|  |  |  |  |
| **SW2** | clk | RGB2 Red | R Farm |
| **SW1** | rst | Create the colour yellow. | Y Farm |
|  |  | RGB2  Green | G Farm |

1. You will need to demonstrate the design to the instructor in person or via video before the due date.

**Demonstrate the design to the instructor.**

**Make sure you have your state machine diagram and table**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. \*\*\*Note: You still need to *MAKE* yellow (R + G), but port the Hwy Lights to LEDR [9:7] and the Frm\_rd lights to LEDR[2:0]