EE 271: Final Project

Parking Lot Occupancy Counter

Consider a parking lot with a single entry and exit gate. Two pairs of photosensors are used to monitor the activity of cars, as shown in Figure 1. When an object is between the photo transmitter and the photoreceiver, the light is blocked, and the corresponding output is asserted to 1. By monitoring the events of two sensors, we can determine whether a car is entering or exiting, or a pedestrian is passing through. For example, the following sequence indicates that a car enters the lot:

- Initially, both sensors are unblocked (i.e., the a and b signals are 00).
- Sensor a is blocked (i.e., the a and b signals are 10).
- Both sensors are blocked (i.e., the a and b signals are 11).
- Sensor a is unblocked (i.e., the a and b signals are 01).
- Both sensors become unblocked (i.e., the a and b signals are 00).

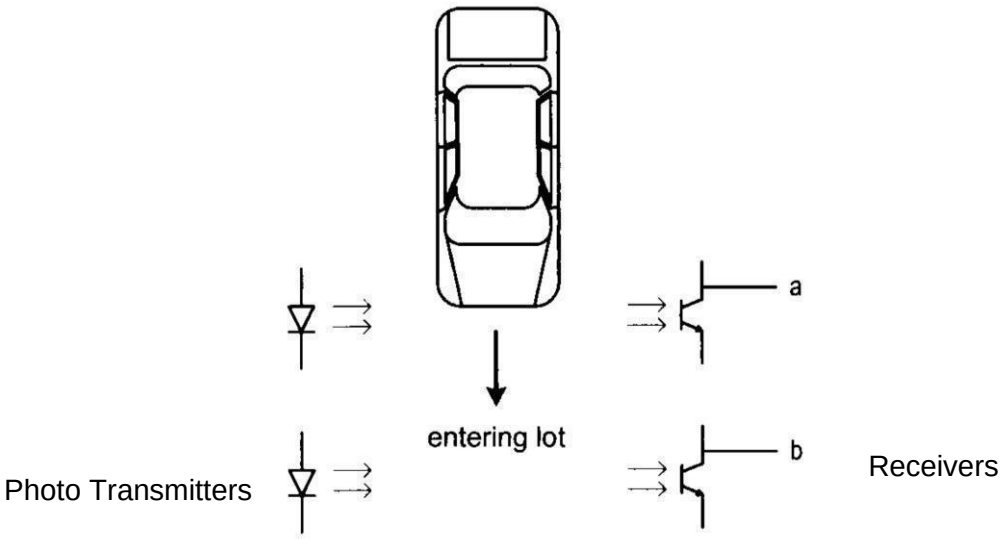

Figure 1. Parking lot sensors

Design a parking lot occupancy counter as follows:

1. Design an FSM with two input signals, a and b, and two output signals, *enter* and *exit*. The *enter* and *exit* signals assert true for one clock cycle when a car enters or exits the lot, respectively. Derive the SystemVerilog code for the FSM and simulate it in

ModelSim. Do not assume that cars will not change direction while entering or exiting the parking lot.

2. Design a counter with two control signals, *inc* and *dec*, which increment and decrement the counter when asserted. Assume that the maximum capacity of the parking lot is 25 spots. however, you may test (and demo) your system with a maximum of 5 spots. Your system must work with both of these sizes. Derive the SystemVerilog code for the FSM and simulate it in ModelSim.
3. Combine the counter and the FSM and then model the parking lot, using two switches on the breadboard to mimic the two sensor outputs and the seven-segment displays to display the car count. Your system should have the following:
    a. Display the car counts as they enter the parking lot on the seven-segment displays HEX0 and HEX1.
    b. If the counter reaches 25 (or 5 for simulation and demo purposes), display the word “FULL” on HEX5-HEX2
    c. As cars exit the lot, the counter decrements and the corresponding number should be displayed on HEX0 and HEX1.
    d. When the lot is empty. Display the word “CLEAR” on HEX5-HEX1 and display the number ‘0’ on HEX0.
    e. Use 2 LEDs on the breadboard to represent the a and b signals. When a is 1, turn on the LED on the left, and when a is 0, turn off the LED on the left. Stimulatingly, when b is 1, turn on the LED on the right, and when b is 0, turn off the LED on the right.
    f. Use the third switch on the breadboard as the reset signal.
4. Please read the “GPIO guide” for the instructions regarding how to wire and use switches and LEDs. The general rule is labeled as below.
    a. Wire 2 LEDs to output ports (GPIO0_0[26] and GPIO0_0[27])
    b. Wire 3 switches to any input ports (GPIO0_0[5] – GPIO0_0[22]) (Please pick any three you would like to use, and update your SystemVerilog code accordingly)

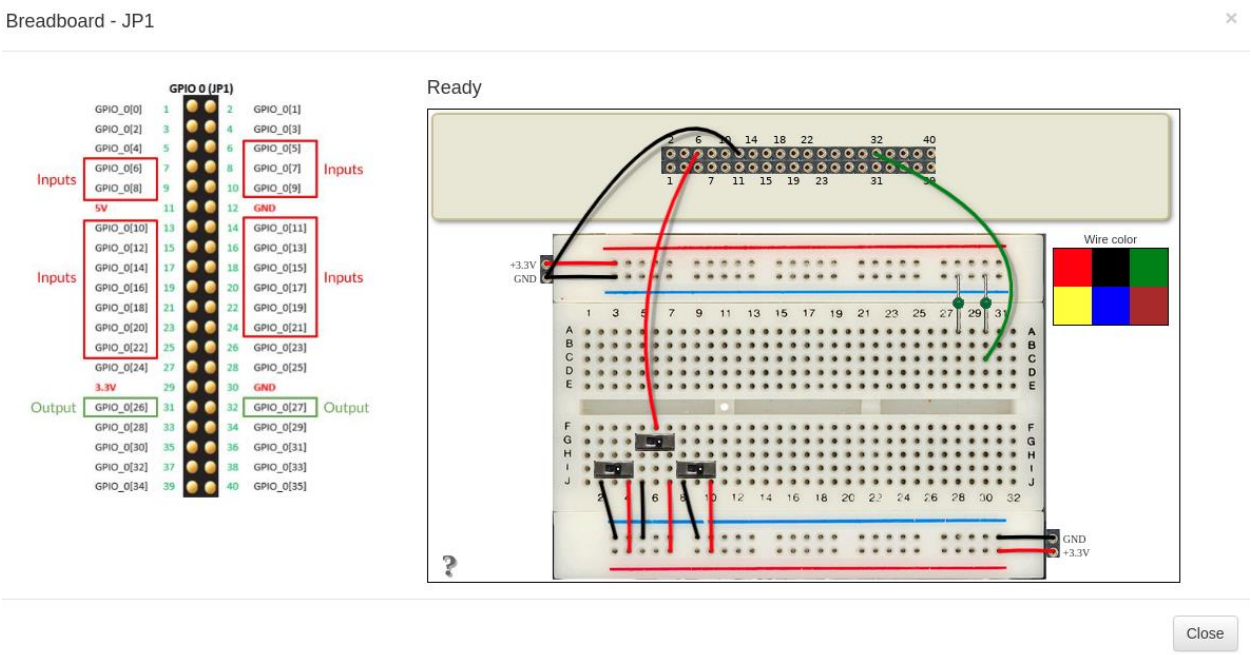

Figure 2. LabsLand GPIO headers

5. Simulate the system in ModelSim.
6. Upload the program onto LabsLand FPGA and record a 2-3 mins video to demonstrate your work. Note, we do not need to see your compilation part.
7. Create a block diagram for your system and include it in your lab report.
8. In the report, include the state diagram(s) that you used in designing this system

Lab Demonstration and Submission Requirements

- Demonstrate the project on the DE1_SoC board in a video that shows the functionality of your project.
- Submit a short lab report that should include 3 main sections, detailed below.

  Procedure
  - Describe how you approached the problem and include state diagrams for all FSMs.
  - Include a top-level block diagram for your entire design, showing the major modules and how they are interconnected.

  Results
  - Include screenshots of ModelSim simulation for **all** modules. You must have a testbench for every module in the project.
  - Turn in a screen shot of the “Resource Utilization by Entity” page. Write the computed size for your design.
  - 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

- 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.
- Export the code of any .sv file you have to a pdf version. Submit the Pdf version and the original .sv files to canvas.