EE/CSE 371:
Design of Digital Circuits and Systems

Lab1: Parking Lot Occupancy Counter

Lab Objectives

This lab provides a refresher of the finite state machines that you learned in EE 271 by designing a parking lot occupancy counter.

Laboratory Design Kits

We will use the Quartus Prime Software/ ModelSim and LabsLand. You do not need to use any physical lab kits.

Please see Lab 0 regarding how to install Quartus Prime Software/ModelSim and log in LabsLand.

Assigned Task

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).

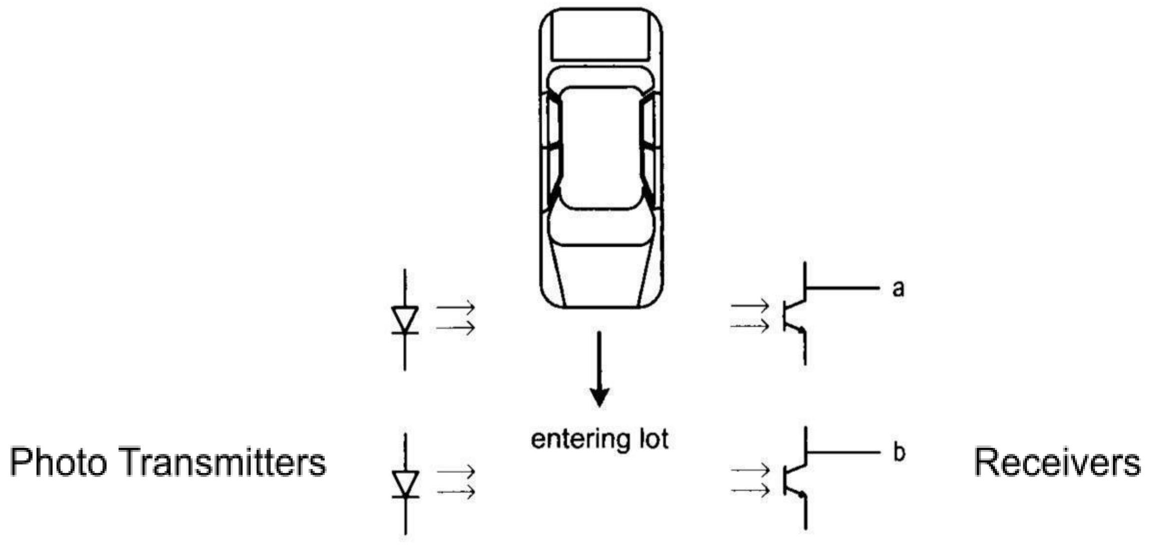

Figure 1. Parking lot sensors

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)

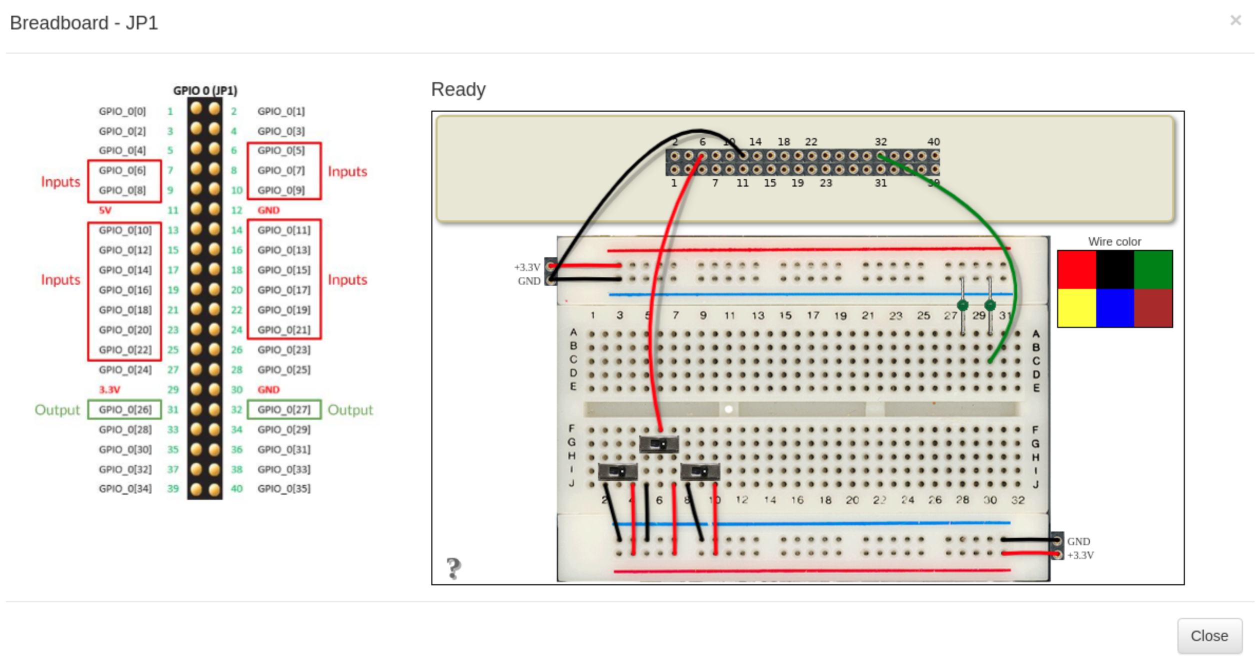
Breadboard - JP1
GPIO 0 (JP1)
GPIO_0[0] 1 2 GPIO_0[1]
GPIO_0[2] 3 4 GPIO_0[3]
GPIO_0[4] 5 6 GPIO_0[5]
GPIO_0[6] 7 8 GPIO_0[7]
GPIO_0[8] 9 10 GPIO_0[9]
5V 11 12 GND
GPIO_0[10] 13 14 GPIO_0[11]
GPIO_0[12] 15 16 GPIO_0[13]
GPIO_0[14] 17 18 GPIO_0[15]
GPIO_0[16] 19 20 GPIO_0[17]
GPIO_0[18] 21 22 GPIO_0[19]
GPIO_0[20] 23 24 GPIO_0[21]
GPIO_0[22] 25 26 GPIO_0[23]
GPIO_0[24] 27 28 GPIO_0[25]
3.3V 29 30 GND
GPIO_0[26] 31 32 GPIO_0[27]
GPIO_0[28] 33 34 GPIO_0[29]
GPIO_0[30] 35 36 GPIO_0[31]
GPIO_0[32] 37 38 GPIO_0[33]
GPIO_0[34] 39 40 GPIO_0[35]
Inputs
Inputs
Output
Output
Ready
+3.3V
GND
Wire color
GND
+3.3V
Close

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

- Record a video to demo the task assigned above. You will need to demonstrate the soundness of your design by executing the design on the FPGA. Moreover, you are expected to utilize testbench simulations to demonstrate that your modules work as expected.
- Write a Lab Report, as framed by the Lab Report Outline document on Canvas. Comment your code. Follow commenting guidelines as discussed in the Commenting Code document on Canvas. Submit your lab report as a pdf file and all of your SystemVerilog files on Canvas.
- On Padlet, write about a problem you had in the lab and the fix to it, and 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 can be found in the corresponding Canvas assignment.