Parking Lot Passcode Entrance

Lab Objectives

This lab provides a refresher of the Finite State Machine (FSM) and General Purpose Input/Output (GPIO) by designing a parking lot passcode entrance and occupancy counter.

Assigned Task

Parking Lot Passcode Entrance

Consider a parking lot with a single entry and exit gate. This parking lot is private and thus requires a passcode for the cars to enter. The passcodes use three buttons and are three characters long. The buttons, B1, B2 and B3, should be mapped to KEY[1], KEY[2], and KEY[3] respectively.

One car can enter the parking lot, and one car can leave the parking lot at a time. The user will control when a car arrives at the entrance gate through a button on the 3D simulation. When a car arrives, the rest of the functionality of the car is autonomous: the car will be parked at the parking spot of its choice. Similarly, the user will control when a car leaves the parking lot. The entering or exiting car will wait at the entrance/exit gate until the gate opens, which is controllable by asserting the appropriate GPIO pin.

Read the “3D Parking Lot Guide” for a guide on the UI of the 3D parking lot and the proper GPIO mappings.

There are two buildings that use this parking lot, and as such, there are two passcodes that can be used to enter the parking lot. The passcodes for the buildings are as follows:

Building 1: B2 □ B1 □ B3

Building 2: B1 □ B1 □ B2

Design a parking lot passcode system as follows:

1. Design an FSM with three input button signals, *B3*, *B2*, and *B1*. The FSM will have two output signals, *P1* and *P2*. The *P1* signal asserts true for one clock cycle if the building 1 passcode is entered, while the *P2* signal asserts true for one clock cycle if the building 2 passcode is entered. Derive the SystemVerilog code for the FSM and simulate it in a simulation tool.
2. If either of the passcodes is entered while a car is present at the entrance gate and there is room in the parking lot, the entrance gate should open. When a car is present at the exit gate, the exit gate should open automatically. (*Hint*: You can open the entrance/exit gate by setting V_GPIO[31]/V_GPIO[33] to 1)

   Note: Because of how the simulator interfaces with the DE1-SoC, many clock cycles are needed for the simulator to open either of its gates. For this reason, the provided module *signal_extender.sv* is included in the starter code. This will turn a single-cycle pulse into a pulse guaranteed to be long enough to trigger the gate. (*Hint*: This is only needed with the entrance gate)
3. Design a counter with two control signals, *inc* and *dec*, which increment and decrement the counter when asserted. The maximum capacity of this parking lot is 3 spots. Derive the SystemVerilog code for the FSM and simulate it in a simulation tool.
4. Combine the counter and the FSM, and then model the parking lot, using the 3D simulator and the onboard keys for entering and exiting, 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 display HEX0.
   b. If the counter reaches 3, display the word “FULL” on HEX5-HEX2, and the number ‘3’ on HEX0.
   c. As cars exit the lot, the counter decrements, and the corresponding number should be displayed on HEX0.
   d. When the lot is empty. Display the word “CLEAR” on HEX5-HEX1 and display the number ‘0’ on HEX0.
   e. Use 3 onboard keys, KEY[1], KEY[2], and KEY[3] to represent the keypad buttons B1, B2, and B3.
   f. Use SW[9] as the reset signal.

5. Change the color of the LED Parking (V_GPIO[26]/ V_GPIO[27]/ V_GPIO[32]) to red when their corresponding parking spots are occupied and green when they are available. For example, when parking spot 1 is occupied, LED Parking 1 should be red.
6. Change the color of the LED Full (V_GPIO[34]) to red when the parking lot is full and green when there are vacancies.
7. Simulate the system in a simulation tool.
8. Upload the program onto the LabsLand FPGA and record a 2-3 mins video to demonstrate your work. Note, we do not need to see your compilation part.
9. Create a block diagram for your system and include it in your lab report.
10. In the report, include any relevant diagrams that you used in designing this system, including all state diagrams and block diagrams, 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. Comment your code.
- Submit your lab report as a PDF file and all of your SystemVerilog files.