Vincent Kwok & Hunter Hatzenbeler
EE/CSE 371
April 4, 2023
Lab 1 Report

Parking Lot Design Procedure:

In this lab, we were tasked with creating a parking lot occupancy counter. The parking lot has a single path that is used as an entrance and exit. In order to track cars moving in and out, 2 sensors are placed at the entrance which are spaced at an approximate distance of a length of a car. Depending on the order and timing that the sensors are blocked, the occupancy counter can make changes to increment, decrement, or not change at all. Two switches are used to simulate the sensors being blocked while another switch is used as the reset signal.

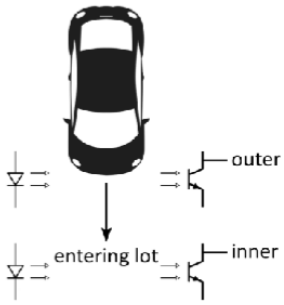


Figure 1: *The parking lot photosensor setup.*

The image in Figure 1 shows the position of the sensors. Depending on the order of how the sensors are activated, a vehicle can be recognised to be either entering or exiting. The spacing between the sensors is large enough that a pedestrian can only activate 1 sensor but large enough that a car passing through will activate both sensors at some point. This allows the system to differentiate between pedestrians and vehicles .

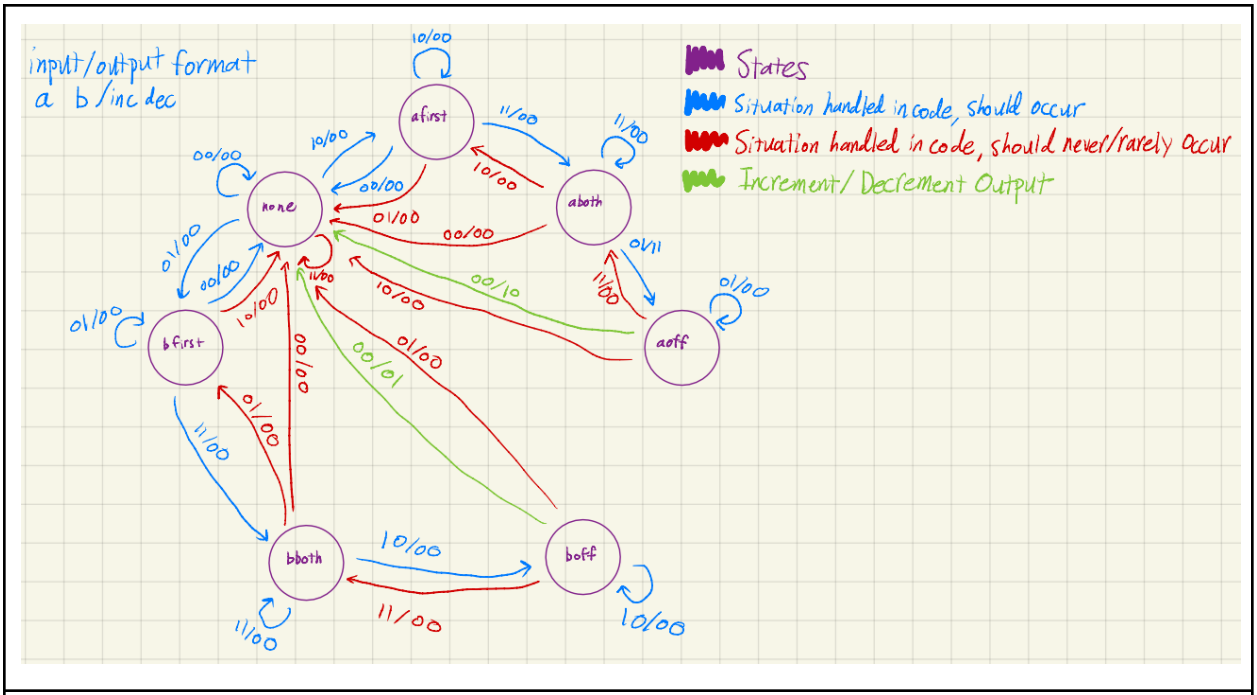


Figure 2: The Mealy FSM for the main module. The input/output format goes a|b/inc|dec, where a represents the outer sensor and b represents the inner sensor. All inputs and outputs are on active high. The FSM is color coded for likely and unlikely occurrences as well as increment and decrement outputs.

To know when the parking lot counter must increment or decrement, a Mealy FSM (Figure 2) was created. There are many situations (red) that rarely occur, only when there are cars very tightly following each other while entering/exiting. Other unlikely situations occur when there are multiple pedestrians blocking sensors at timings that do not correspond to a typical vehicle's behavior or the behavior of a single pedestrian crossing through the sensor. Situations that cause the counter's value to change (green) involve a sensor being activated, then both, then the sensor that was not initially triggered to be activated, then none.

Overall System:

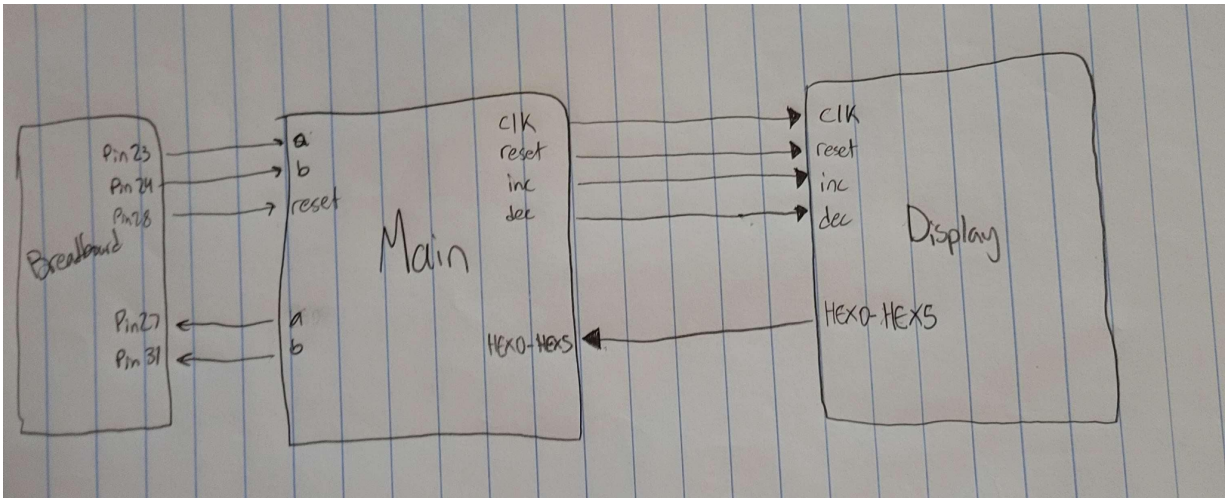


Figure 3: Top-level block diagram of the Parking Lot Counter System

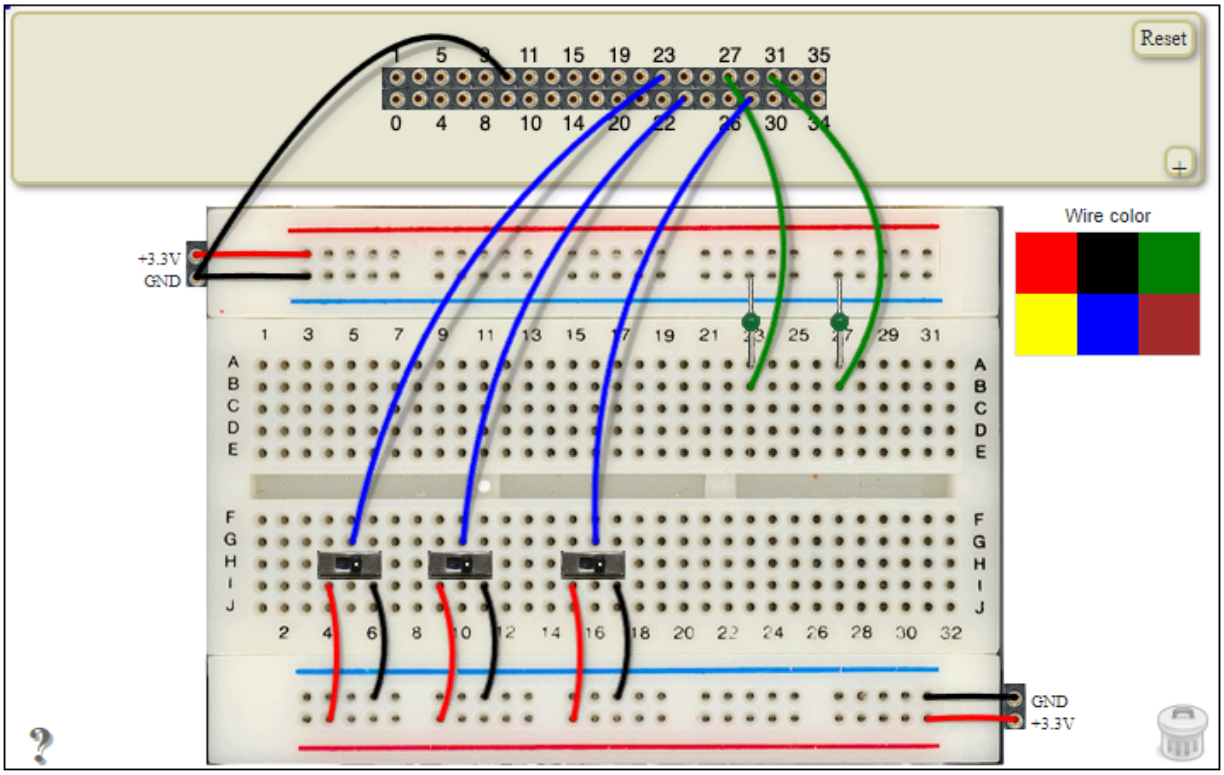


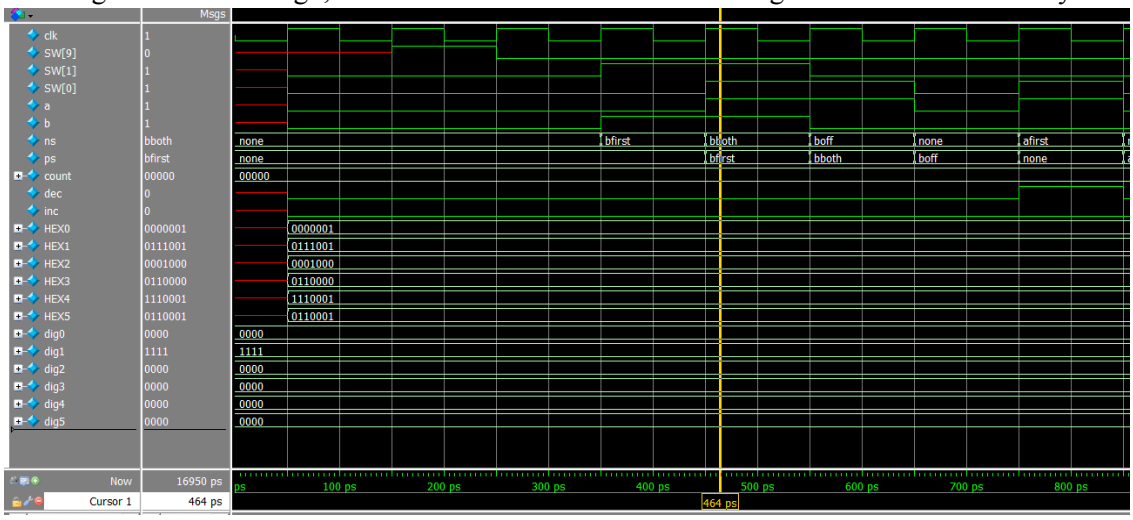
Figure 4: Offboard components using the GPIO board in LabLands (pin 24 connected to pin 27, pin 28 connected to pin 31)

External switches and LEDS from the default user interface of Lablands are connected to the FPGA using GPIO pins (Figure 4). The order of the switches from left to right are: reset, outer sensor, inner sensor. The left LED represents the activation of the outer sensor and the right LED represents the activation of the inner sensor.

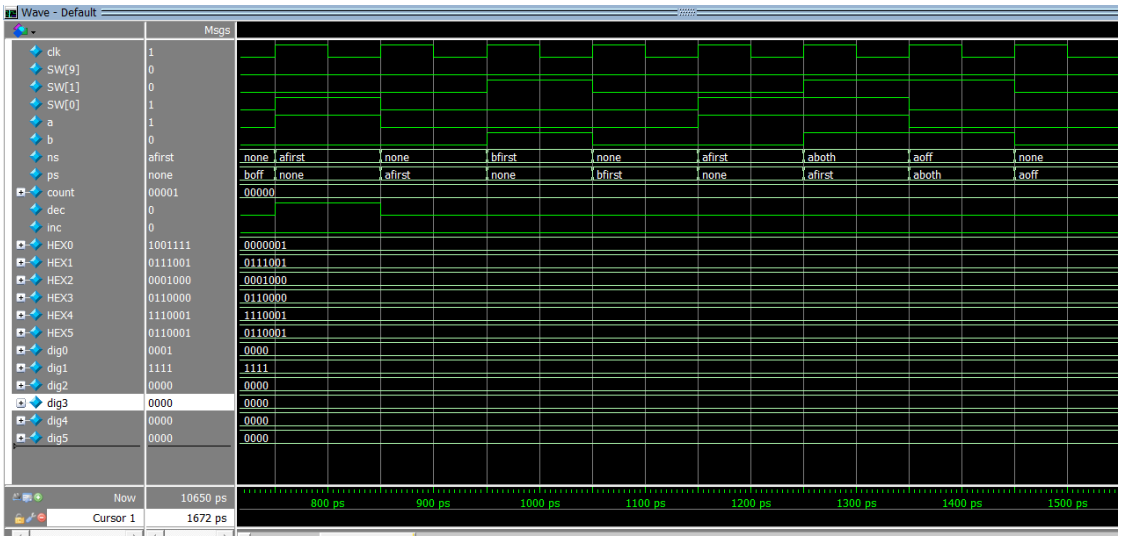
Results:

The testbench for main was designed to go through all paths of the FSM, in order to test the general behavior of the code, to see if it matches up with the expected behavior. To do this, we ran through 5 cases: counting below 0, a non-full signal, a standard increment, standard decrement, and counting above 16.

Case 1, Counting below 0: The first test in the simulation was to make sure that if count was at 0, and a decrement signal came in, and the count was already at 0, then the counter would not go below 0, and the hex display would not be updated. As you can see, the states go through the path for a decrement, a decrement signal comes through, but none of the other values are changed since count is already at 0.

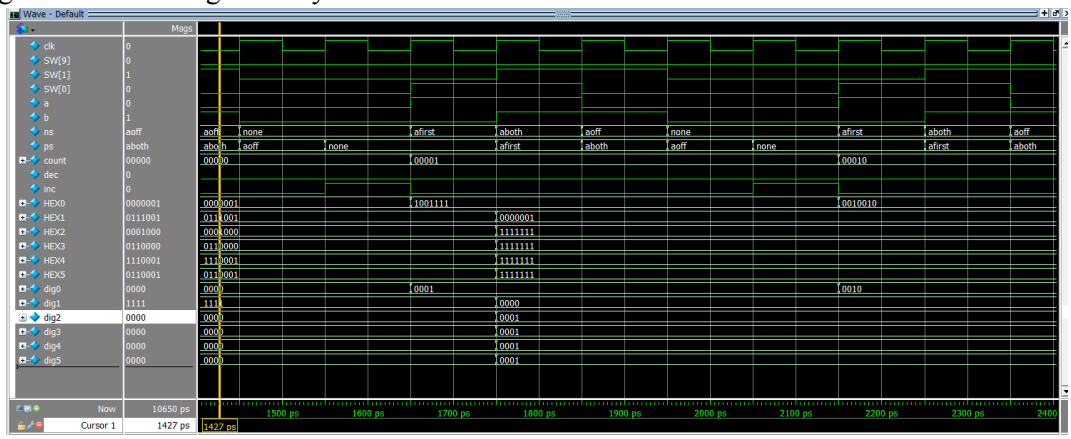


Case 2, Non-full signal: The next case that we wanted to cover was to simulate if either a pedestrian or motorcycle went through, or if a car went partially through and then backed out. To do this, we turned a on and then off, and then we turned b on and then off. The results of this are expected, where the counter neither increments nor decrements.

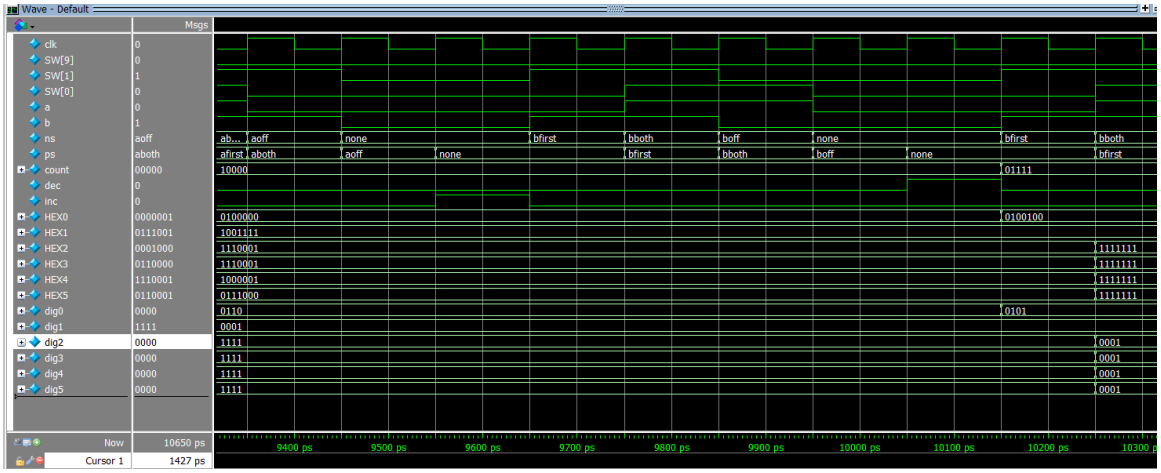


Case 3, Standard increment: The condition to create a standard increment is to first trigger the outer sensor, then both sensors, then the inner sensor. This condition using the variables in our code can be

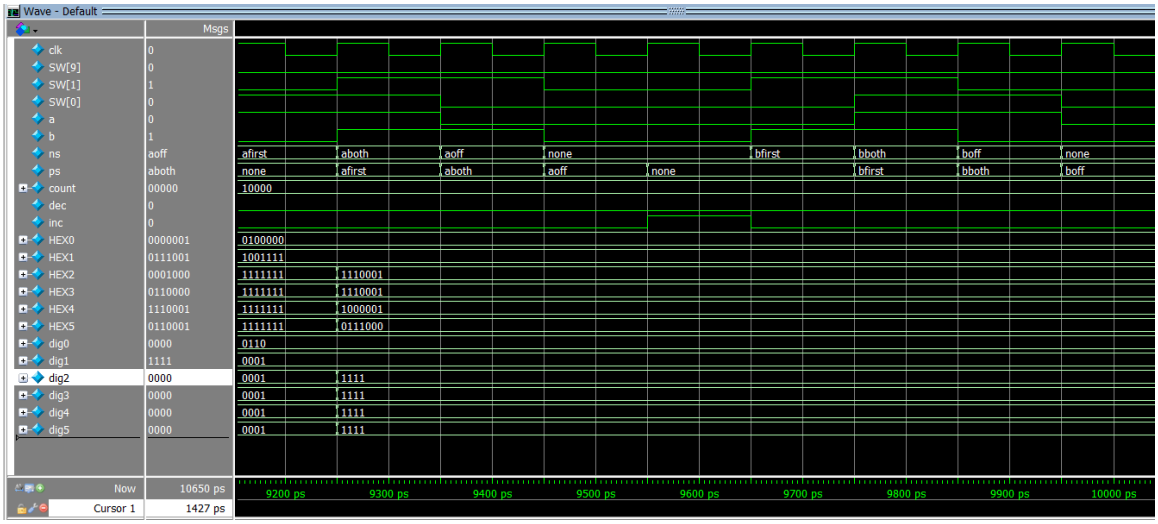
shown as a=1 -> a=1&b=1 -> b=1. Starting at around 1650ps, the outer sensor (a) is triggered, the correct sequence follows, ending at 2050ps where the inc variable registers an increment and causes HEX signals to change in the following clock cycle.



Case 4, Standard Decrement: The condition to create a standard decrement is to first trigger the inner sensor, then both sensors, then the outer sensor. This condition using the variables in our code can be shown as b=1 -> a=1&b=1 -> a=1. Starting at around 9650ps, the inner sensor (a) is triggered, the correct sequence follows, ending at 10050ps where the dec variable registers an decrement and causes HEX signals to change in the following clock cycle.



Case 5, Counting over 16: The last case that we wanted to cover was to check if once the parking garage was full, if it would keep counting. From the waveform, you can see that the count is already at 16, and the hex displays full. When an increment signal comes through after this, the count and hex displays do not change.



Flow Summary:

Flow Summary

<<Filter>>

Flow Status	Successful - Wed Apr 05 12:36:52 2023
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	main
Top-level Entity Name	main
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	44 / 56,480 (< 1 %)
Total registers	54
Total pins	79 / 268 (29 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

Figure_:The ModelSim Flow Summary of the main module compilation

Experience Report:

We found this lab to be straightforward though that experience was expected as this lab was supposed to be a review of SystemVerilog and FSM design. At times the lab specification was a bit confusing but was

still written well enough to be understood. The intended FSM design was fairly easy to design, though we think it may be possible to design an FSM with fewer states that is also a bit more prone to error. We initially coded the lab using the on-board switches and LEDS but changed this implementation at the end of designing our lab to use off-board switches and LEDS through the provided GPIO board in Lablands. There were several instances where we had to review the functionality of a binary number system as well as the details on how the HEX displays worked in code implementation.

Overall the design/ programming process of this lab went quite smoothly, the most challenging part being figuring out how to use the GPIO board as it has something that both of us do not have experience in, requiring us to read the example documentation.

This lab took approximately 6 hours, broken down as follows:

- Reading - 40 minutes
- Planning - 20 minutes
- Design- 60 minutes
- Coding- 150 minutes

- Testing- 40 minutes
- Debugging- 40 minutes