Ishan Dane idane
Ismail Memon ishmemon

Lab 1 Report

Design Procedure

For Lab 1, we are asked to create a car parking lot system which keeps track of how many cars are parked inside the lot and displays this information on a HEX board. The system also outputs when the lot is FULL (16 cars inside occupying all spaces) or when it is CLEAR (0 cars inside with no occupation). The car counter uses two sensor inputs “Inner” and “Outer” with Outer represented by GPIO_0[6] and Inner by GPIO_0[7]. These sensors are used for the system to determine if a car has entered the parking lot or exited the parking lot.

Car Detection

The simple.sv module is used to detect whether a car has exited or entered the parking lot. This is done through 2 inputs: Inner, Outer whereby Inner and Outer inputs are provided by the light sensors where a blocked sensor returns a 1 and an unblocked sensor a 0. We assume that a pedestrian cannot block both lights like a car and therefore the system has no output if the state “BOTH” is not traveled through in the process.

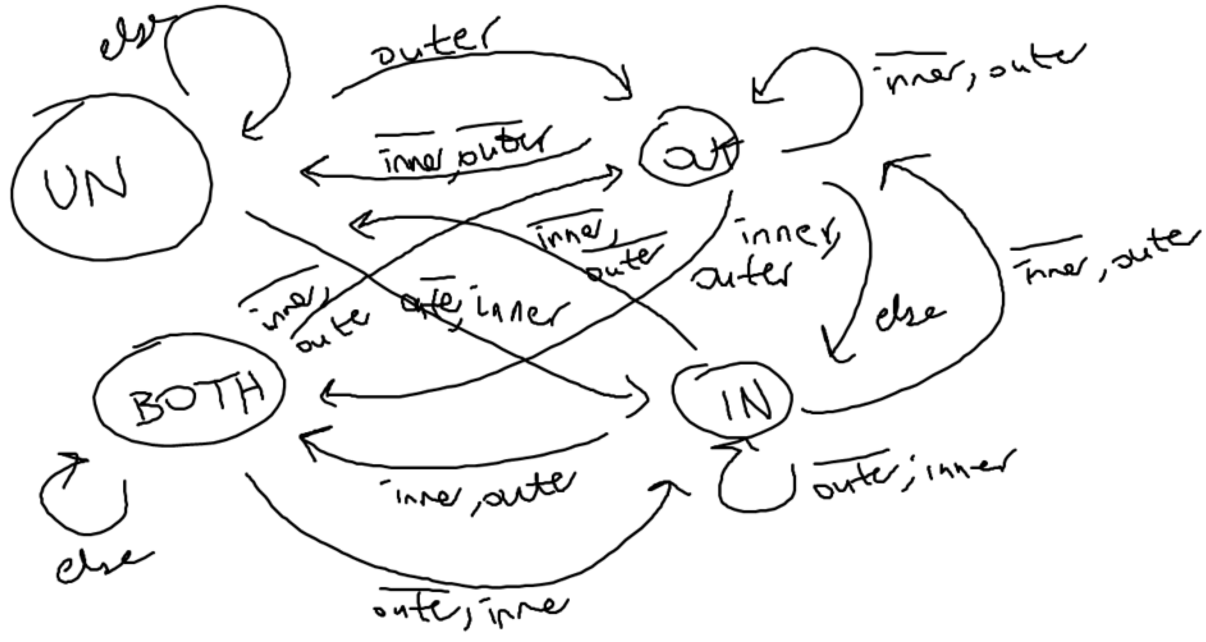

else
outer
UN
$\overline{\text{inner}}, \text{outer}$
out
$\overline{\text{inner}}$, outer
$\overline{\text{inner}}$, outer
inner, outer
$\overline{\text{inner}}$, outer
else
$\overline{\text{inner}}$, outer
$\overline{\text{outer}}$, inner
BOTH
IN
$\overline{\text{outer}}$, inner
$\overline{\text{inner}}$, outer
else
$\overline{\text{outer}}$, inner

Figure 1: FSM for Car Detection Module

Car Counter HEX number display

The carcounter.sv module is used as the counter to count the number of cars entering or exiting the parking lot. Through the occupancy.sv module that combines the carcounter module and the cardetection module, input information from the Car Detection module (simple.sv) that relays when a car enters the parking lot and when a car exits, is inputted into the carcounter module which increases or decreases the counter accordingly.The carcounter module then outputs the resulting total counter value which is inputted

Ishan Dane idane
Ismail Memon ishmemon

into the occupancy.sv module for HEX display. From the specification provided, we can have up to 16 cars in the parking lot with 0 being the smallest number of cars in the lot, providing 17 different states. The module then outputs the number of cars to occupancy.sv which is a module that displays the number of cars in the parking lot on a HEX display. For the zero state, (start number of cars = 0), the HEX display shows "CLEAR" and for the 17th state (final number of cars = 16), the HEX display shows "FULL". With each car that enters, the Car Counter module increases by 1 and with each car that exits, it decreases by 1. Besides the min and max states mentioned above ("Clear and Full"), the number of cars in the parking lot are displayed on the HEX display through the occupancy.sv module that takes in carcounter.sv module's output total count of cars.

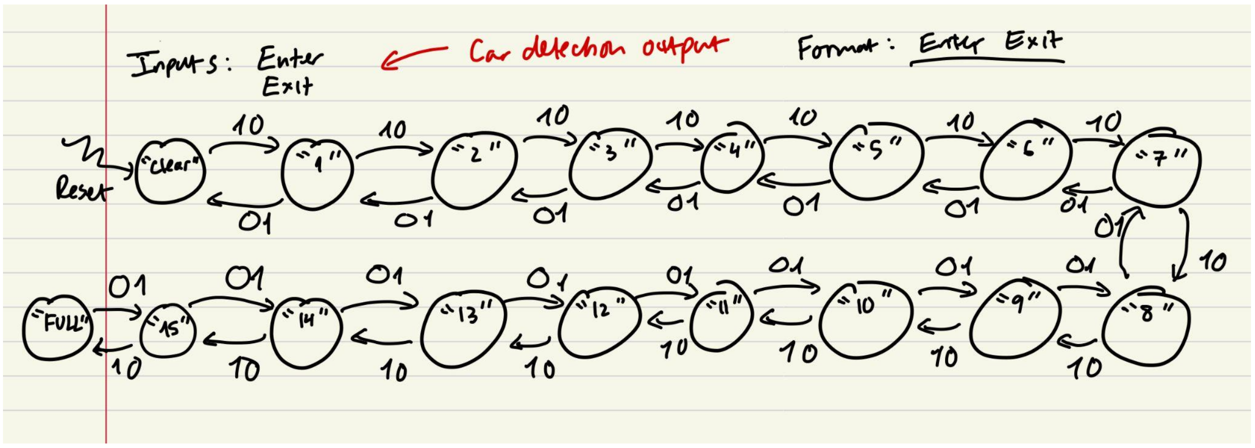
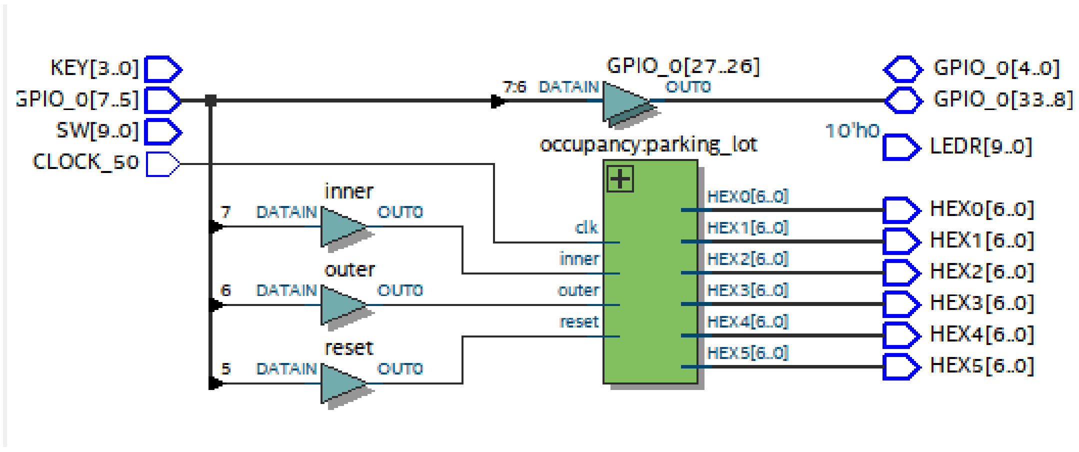


Figure 2: FSM state diagram for car counter module

Overall System Block Diagram:



Ishan Dane idane
Ismail Memon ishmemon

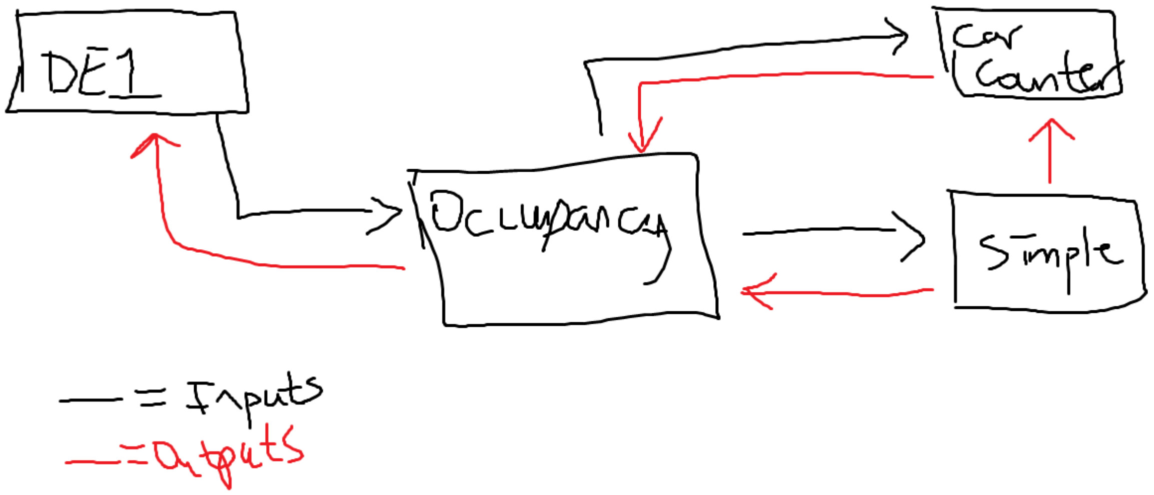


Figure 3: Full system block diagram

The DE1 module calls the occupancy module, which tracks the number of cars and displays that value on the HEX displays. The occupancy module calls the simple module which is a FSM of whether a car enters or exits, and uses that enter/exit data to call the car counter module, which uses a 5 bit counter to keep track of the number of cars in the parking lot. With this information, the occupancy module displays different numbers/letters on the HEX displays depending on the value of the counter from the car counter module.

Results

Car Detection Simulation

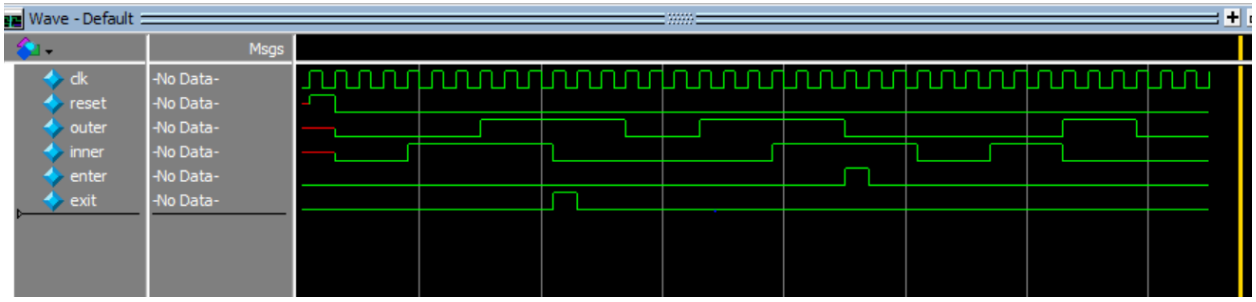


Figure 4: ModelSim Waveform for Car Detection module simple.sv

To test the Car detection module (simple.sv), a testbench was created which would move through the states accordingly and retrieve the correct output. For example, as shown in the Figure 4 simulation, Only Inner -> Both -> Only Outer resulted in an active high exit output whereas Only Outer -> Both -> Only Inner resulted in an active high enter output. This simulation therefore proves that the states work correctly. Furthermore, as mentioned by the specification, the simulation also tests the instance in which a pedestrian walks through instead

Ishan Dane idane
Ismail Memon ishmemon

of a car. This can be seen in the simulation as Only Inner -> Only Outer as we are assuming that the pedestrian is not big enough (like a car) to block both light sensors. This is therefore shown in the simulation as no output since the system ignores pedestrian cases.

DE1_SoC Simulation

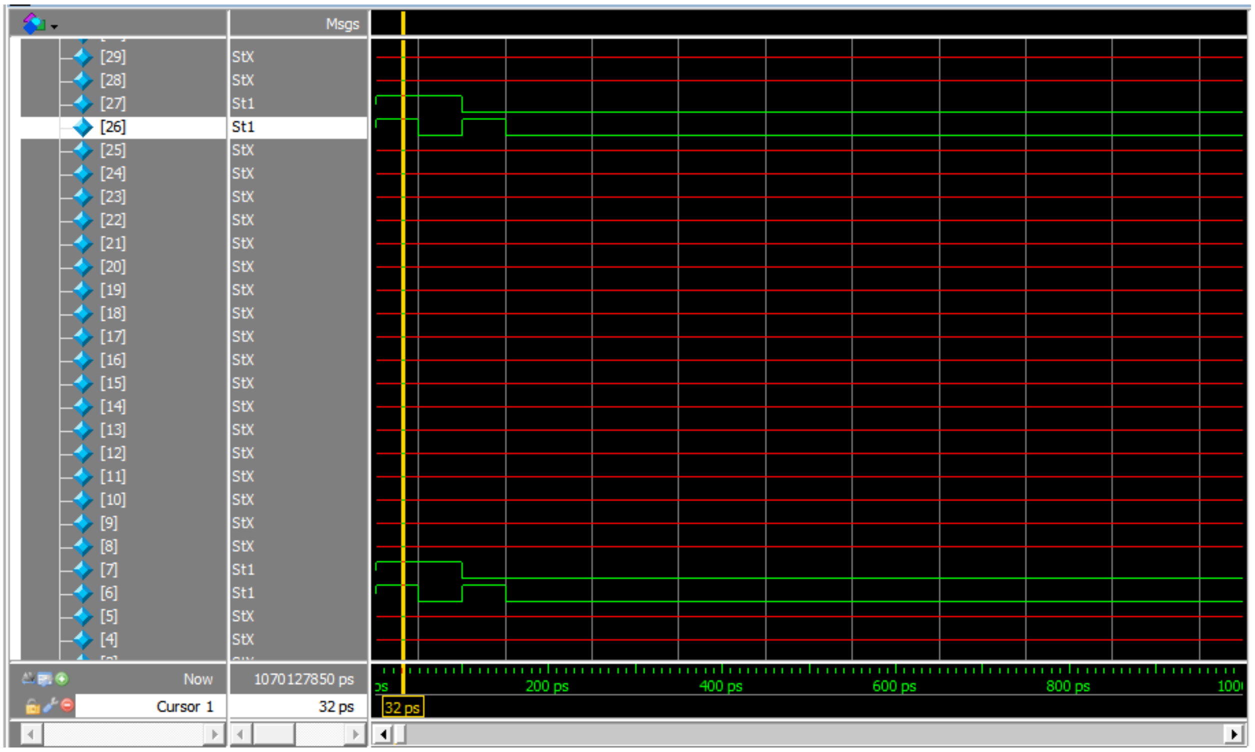


Figure 5: ModelSim Waveform for DE1_SoC module DE1_SoC.sv
To test the DE1_SoC module, the objective was to see whether the switches are connected to the LED's as they should be. We want GPIO_0[6] (representing outer) to drive GPIO_0[26] and GPIO_0[7] (representing inner) to drive GPIO_0[27]. As we can see from Figure 6, the output pattern of GPI0[26] is exactly the same as that of GPIO_0[6] and the output pattern of GPI0[27] is exactly the same as that of GPIO_0[7], meaning that the switches and LEDs are properly connected.

HEX Number Display Simulation

Ishan Dane idane
Ismail Memon ishmemon

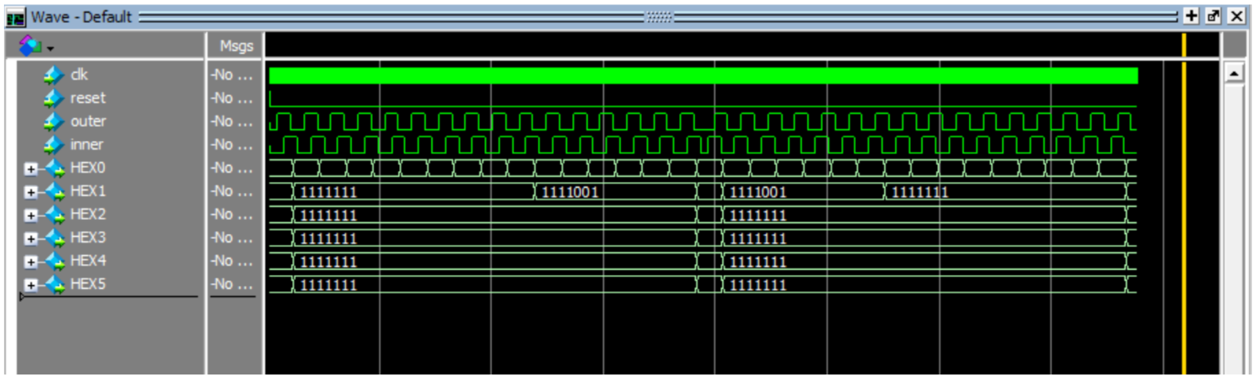


Figure 6: ModelSim Waveform for HEX display counter occupancy.sv

To test the HEX number display module (occupancy.sv) which combines the car counter and car detection modules, a testbench was created which tests the HEX number display for up to 16 cars entering the parking lot and 16 cars exiting the parking lot. This simulation shows the HEX number display for each instance that the car count increases from the minimum “CLEAR” CASE (count = 0) up to the “FULL” case (count = 16) then back down to the “CLEAR” case (count = 0). Besides the min and max cases, the numbers in between are displayed on the HEX display accordingly as shown on Figure 6.

Experience Report

This Lab took ___ 9 __ hours:

- Reading Specification/ Lab manual: 0.5 hours
- Drawing out FSM’s: 0.5 hours
- Code planning and Design: 1 hours
- Coding: 3 hours
- Testing/ Simulations: 2 hours
- Writing Report: 2 hours