Eugene Ngo: 1965514

3/13/2023

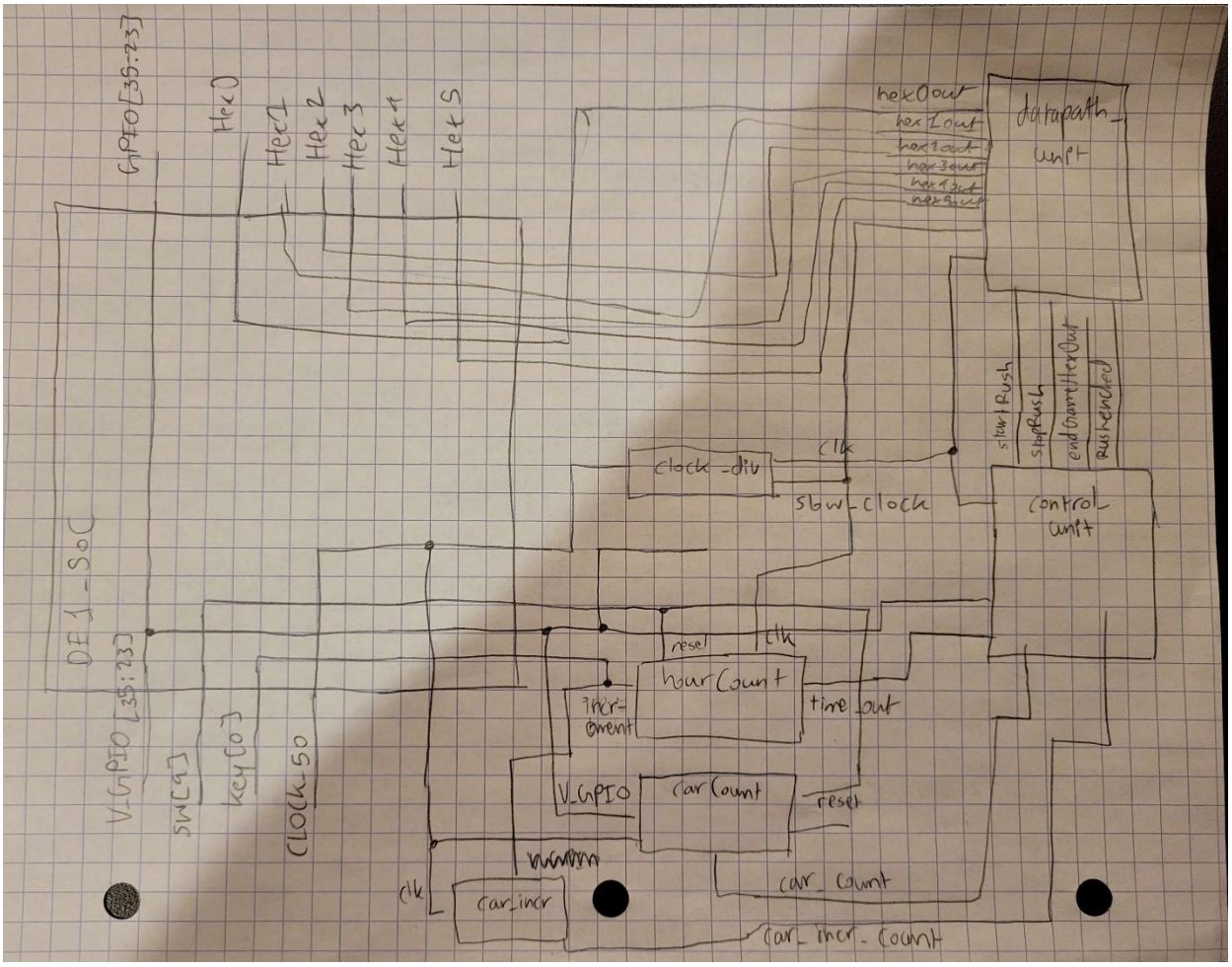
EE 371 – Lab 6 Report

Procedure

This lab was a continuation or development of lab 1 and comprises of two tasks. The first task was to take the same implementation from lab 1 and demonstrate its functionality onto the new breadboard remote lab surface. This entailed changing from “GPIO_0” references to “V_GPIO” references in my top-level module. Then when demonstrating the task on the new breadboard remote lab surface, I had to quickly wire the LEDs and switches to show the full functionality of the task.

The second task was to built upon Lab 1 and the first task from this lab and implement a 3D Parking Lot simulation. The simulation was meant to represent an 8 hour work day and the rush hour that occurs within it. After the work day ends, or 8 hours pass, then the FPGA will display how many cars entered the parking lot at each hour of the day and will show when a rush hour sequence begins—the 3-car parking lot is full—and when it ends—the lot is emptied again. If no rush hour occurred, then a ‘-’ will instead be displayed on both output HEXs. This was achieved by using a dual-port ram that I generated from the IP catalog.

Block Diagram



Task 1

Reading the updated GPIO guide, I changed the task 1 to use the new V_GPIO infrastructure, changing GPIO 5 and 7 to 23 and 24 and then changing GPIO 26 and 27 to 32 and 35. Lastly, GPIO 30 was used for reset instead of GPIO 9. Then, using the newly mapped GPIO, I just updated the logic for the incrementation the same way, if the first sensor was triggered and then the second, and then the first is released after the second, creating an entrance output. Reversing that for exiting, the same parts from lab 1 to increment and decrement the signals were used.

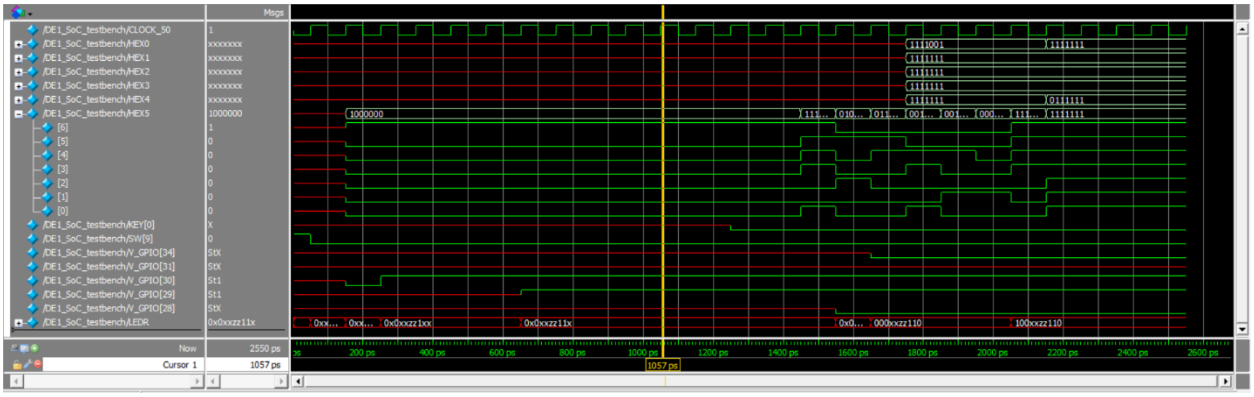
Task 2

Task 2 required using an ASMD with a datapath and control module to create a parking lot simulation with rush hour logic. Initially, the first things that were developed was mapping the

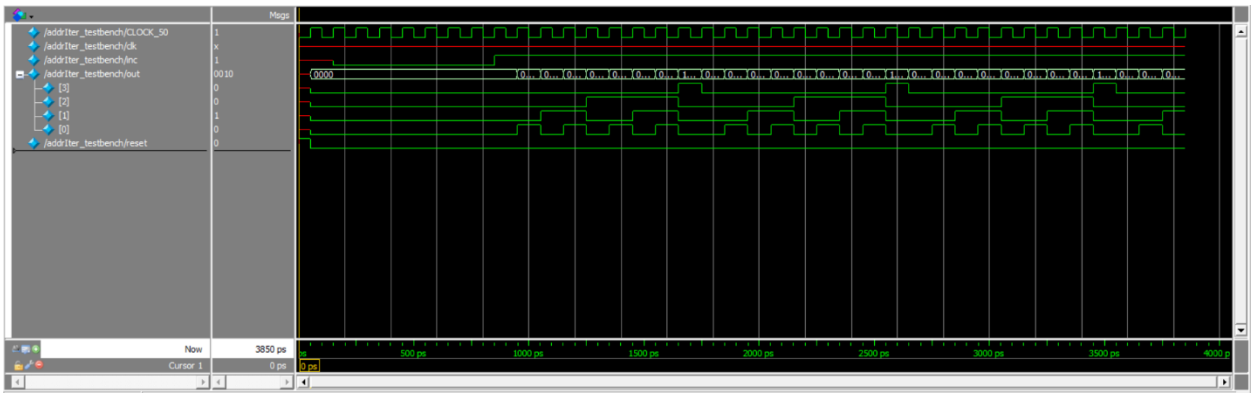
V_GPIO properly to ensure the behavior of the cars on entrance and exit was as expected. Once the behavior was proper, then I mapped out the logic for incrementing the number of hours, the number of cars entering and the number of cars present, using the V_GPIO and FSM logic. Then, using pipelining the number logics to a new unit which would become the control path. Within the control path, I created control signals as outputs that would be set to true based on the GPIO and number conditions. If the hour was 8, the game was in EndGameHex condition. Then, taking in the logic from control path, I created the datapath which was designed to update the Hex values based on the control signals. If the game was in endgame, hex0 to 5 took on values based on the endgame requirements. If the game was in the endgame, RAM was read from, otherwise it was written to. Pipelining all these units into each other, results in the full integration of the parking lot simulation with rush hour logic.

Results

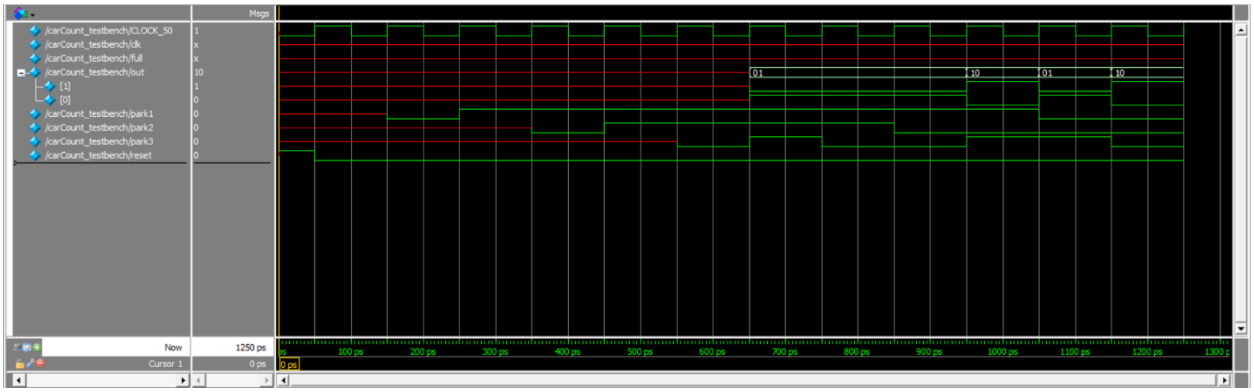
Task 2 Simulations



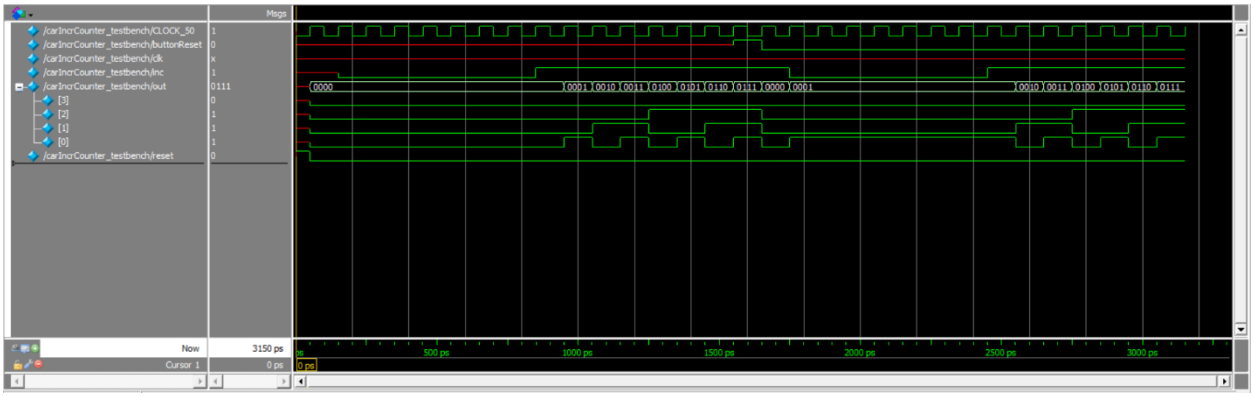
This is the simulation for the DE1_SoC, main module for lab 6, the parking lot simulation. This shows the changing data at Hex0-5, based on the changing hours and parking lot conditions. HEX5 outputs 1111111 at the end due to being switched off after hours are 8 or above.



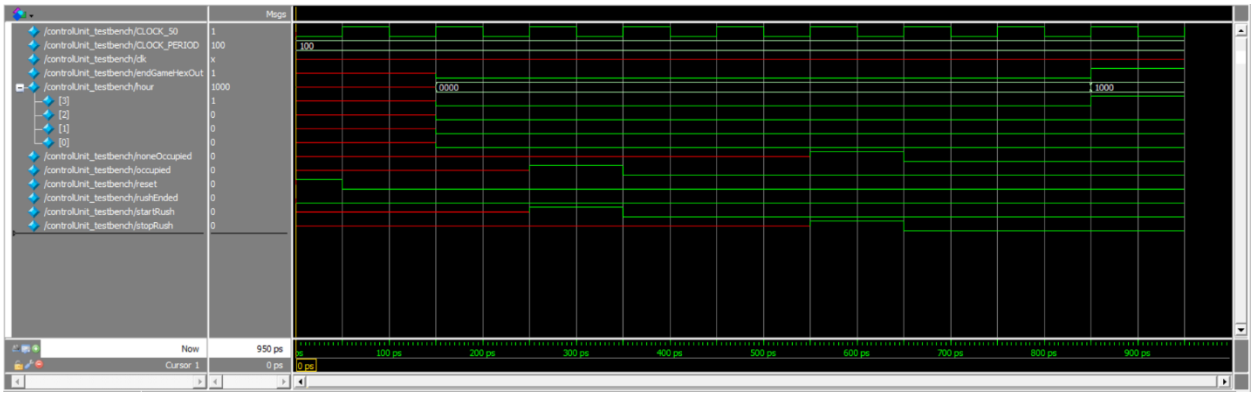
This unit is the address iterator. This is used by the datapath module to read and write from the RAM in different conditions. As you can see with the out variable, it iterates from 0 to 7 and then cycles back to 0 again.



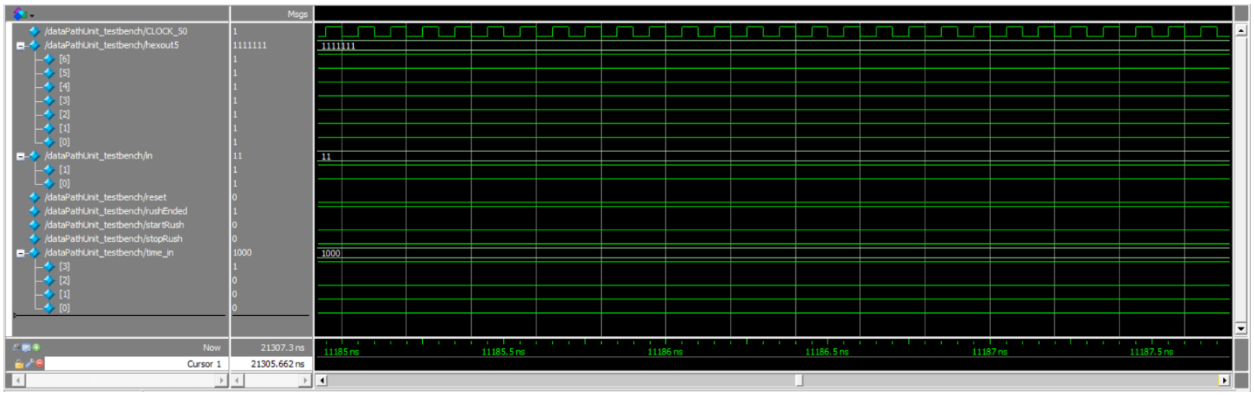
This unit is the car count. This is a module that keeps track of the number of spaces left in the parking lot based on the number of spaces taken first. As you can see the output changes based on if parking 1 or 2 or 3 is taken.



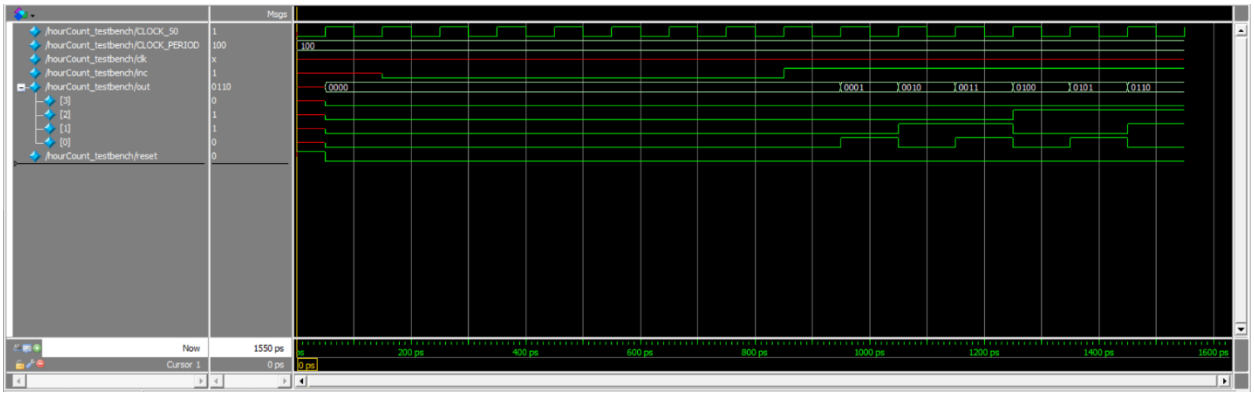
This unit is the car increment counter. This counts up every time there is a car that enters the parking lot and never decrements. It is reset every hour, to 0, so that the RAM can save the number of cars that entered for a specific hour.



The control unit is listed above. This updates various control signals such as startRush,stopRush, rushEnded, endGameHexOut, based on the inputs from VGPIO and number logic. As you can see the signals update based on various GPIO changes.



The datapath unit is listed above. This updates the Hex outputs based on the control signals. As you can see the hex output for HEX5 changes to be off based on the endGameHex signal while also enabling the RAM to be saved to.



This simulation I for the hour count. The hour is updated based on the input Key[0] or the inc signal. This shows that the output is updating every clock cycle based on the inc signal until t reaches the value 8 which it stays at, until reset.

Overview

Overall, this was an interesting lab using the new GPIO and breadboard system. The interactions in the 3D simulation were interesting to program. I learned more about utilizing ASMD logic in combination with a RAM module to then output to HEX0 to HEX5 with different values. It was interesting and I learned a lot about combining just about every aspect of what I learned in 371 to create this 3D parking lot simulator. The spec was long but necessarily so and provided all of the needed supporting documents. It was thorough in teaching us how to properly use the 3D parking simulator.

Overall, the systems work as expected.

Appendix

See following code