Alejandro Hernandez-Farias and Carlos A. Morelos Escalera

EE 371

03/09/2024

Lab 6 Report

Procedure

The purpose of this lab was to explore the board's general purpose input/output (GPIO) ports, which are commonly used for the circuit board to interact with external circuits by either reading in signals (inputs) or by sending signals (output). In this case several inputs were used as sensors to detect the presence of a car in the parking spot, and the outputs were connected to LEDs on the board to indicate if a spot is either taken or available, as well as another one for the parking lot.

Task 1

As stated in the previous section, the goal of task one was to simply implement the lab 1 code using the breadboard lab interface and GPIO pin numbers according the the inputs and outputs specified by the spec sheet.

Task 2

For this task, the parking lot simulator capability in labs land was used to simulate a parking lot with 1 entrance gate, 1 exit gate and 3 spots for cars to come in and park and depending on which spot is taken, an LED light will either turn on or off. The gates will open whenever a car is sensed on either the entrance or the exit gate according to the FPGA inputs V_GPIO[23] and V_GPIO[24], which will send the signal to the FPGA outputs V_GPIO[31] and V_GPIO[33] to open the entrance or exit gate respectively. A similar procedure was followed to connect the presence sensor for the each parking spot from the input to the output which was connected to an LED to indicate the state of that spot.

A rush hour mechanic was also added, it consisted of an 8 hour work day in which at some point during the day, the parking lot was full (3 spots taken) and a later time it was empty (3 spots available). After the 8th hour, HEX3 and HEX4 marked the start and end of rush hour if any. Figure 1 shows the FSM for the controller of this mechanic.

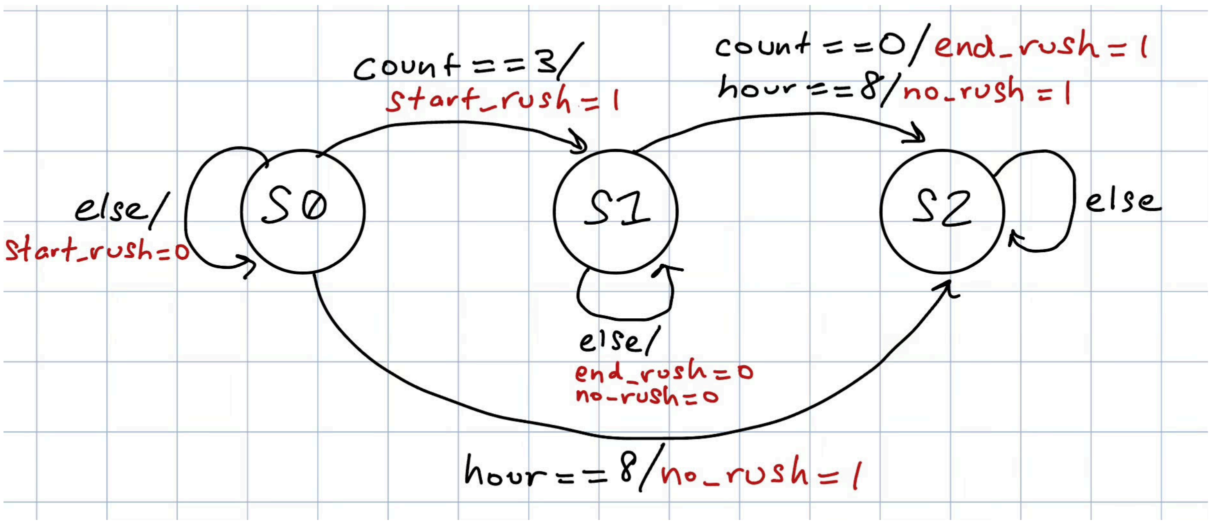

Figure 1. Controller FSM

An 8x16 RAM module was also implemented to count the total number of cars that have entered the parking lot at each hour, in each address, the total number of thats that came in the parking lot at that hour was stored, and in a similar fashion to the rush hour mechanic, at the end of the 8th hour the total number of cars in each address was displayed in FPGA board, HEX1 was used for the data while HEX2 was used for the address of the data being read. Figure 2 shows the FSM for the counter connected to the read address of the RAM module, starting at 0 and going up to 7 every clock cycle when the start signal is asserted, after that, the counter goes back down to 0 until the start signal is reasserted.

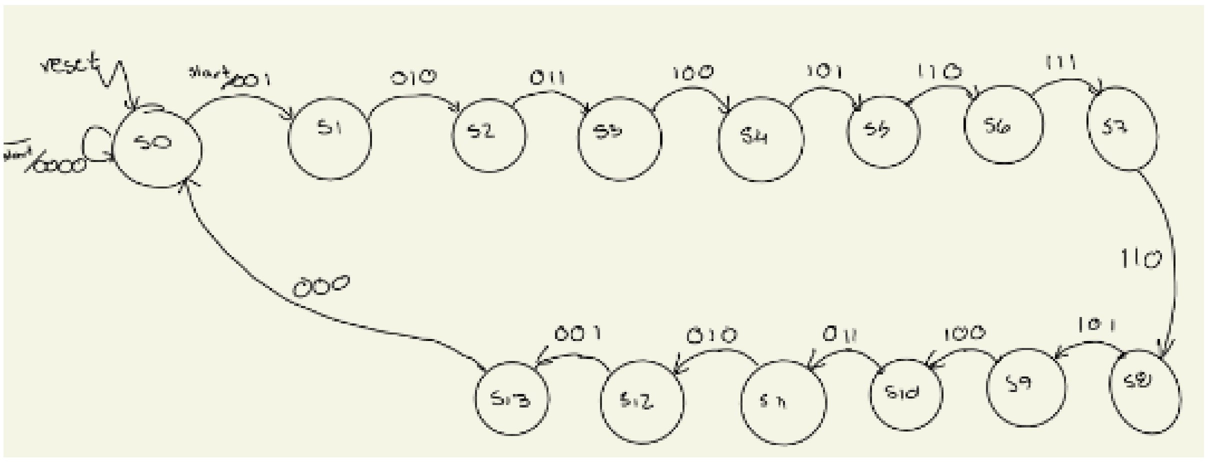

Figure 2. FSM for the counter

When there was no more spaces in the parking lot, displays HEX3 to HEX0 display the word ‘FULL’. The current hour of the work day was displayed in HEX5 advancing everytime KEY[0] was pressed.

Results

Task 2

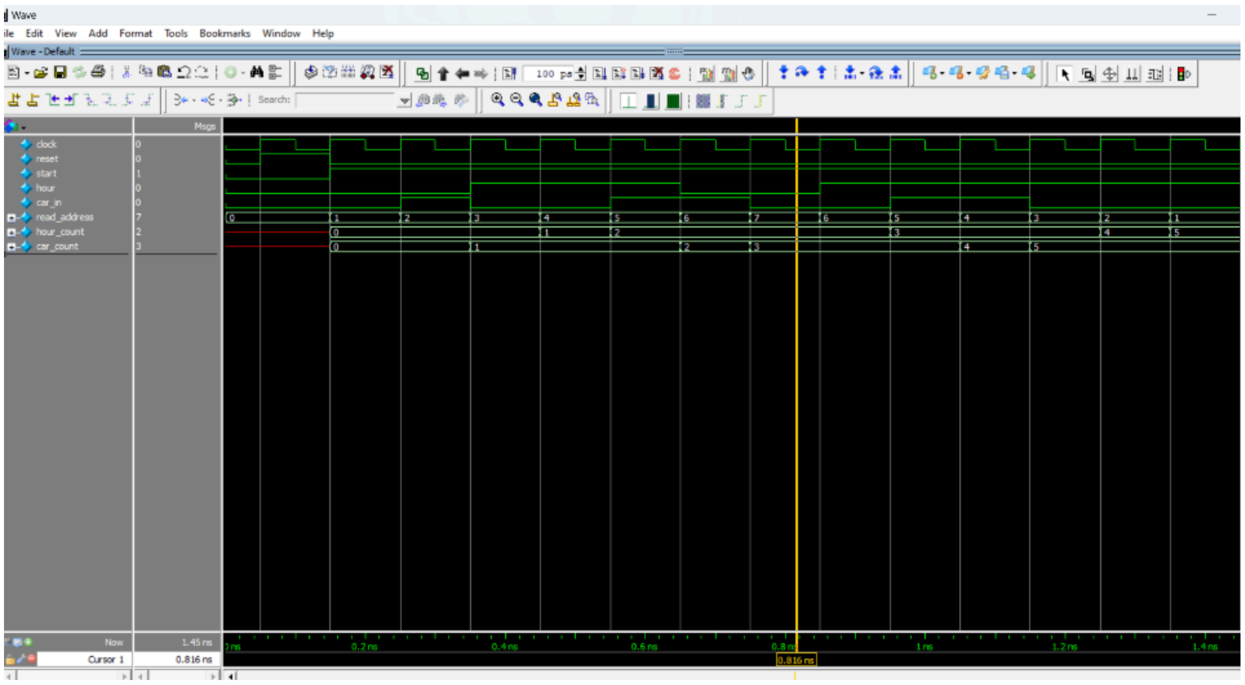

Figure 3. car tracking RAM simulation

The simulation above shows the behavior for the car tracking RAM, it has several control signals, when start is on, the counter connected to the read address port goes up from 0 to 7 every clock cycle, once it reaches 7, it goes back down to 0 decreasing its value by 1 every cycle. When the hour or car in signals are asserted, the car counter and hour counter increment by 1

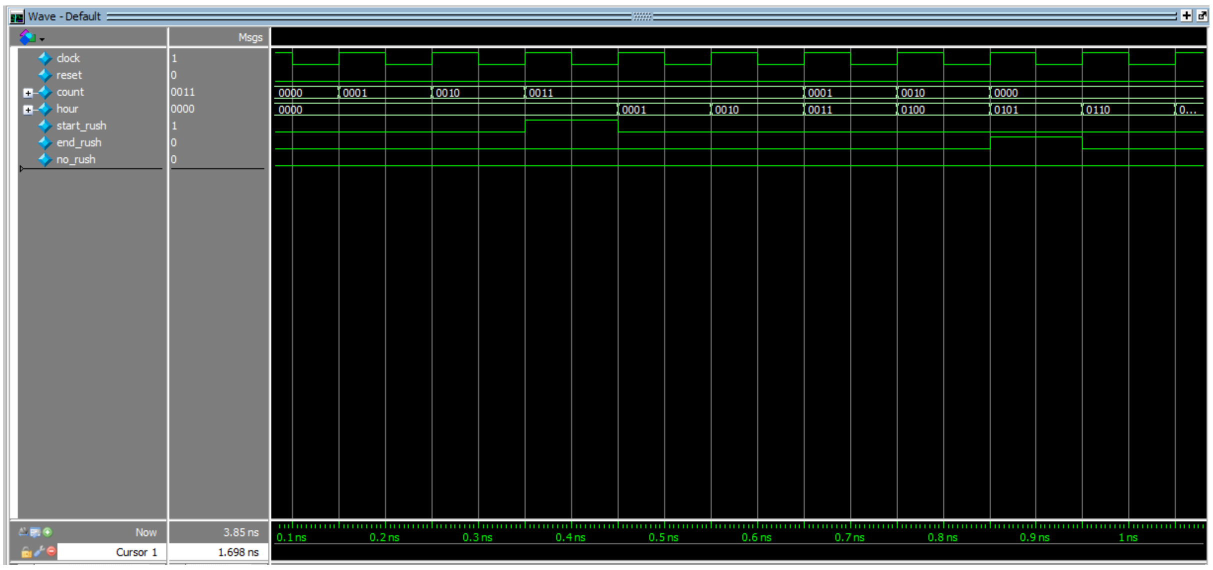

Figure 4. Controller simulation

The simulation above shows the behavior of the controller for the datapath, when the count variable gets to 3, the signal start rush is asserted for one clock cycle marking the start of rush hour, as the hours go by, the count signal decreases and goes back down to 0, in that same clock cycle, the signal end rush is asserted, marking the end of rush hour since the sequence 3 -> 2 -> 1 -> 0 was observed.

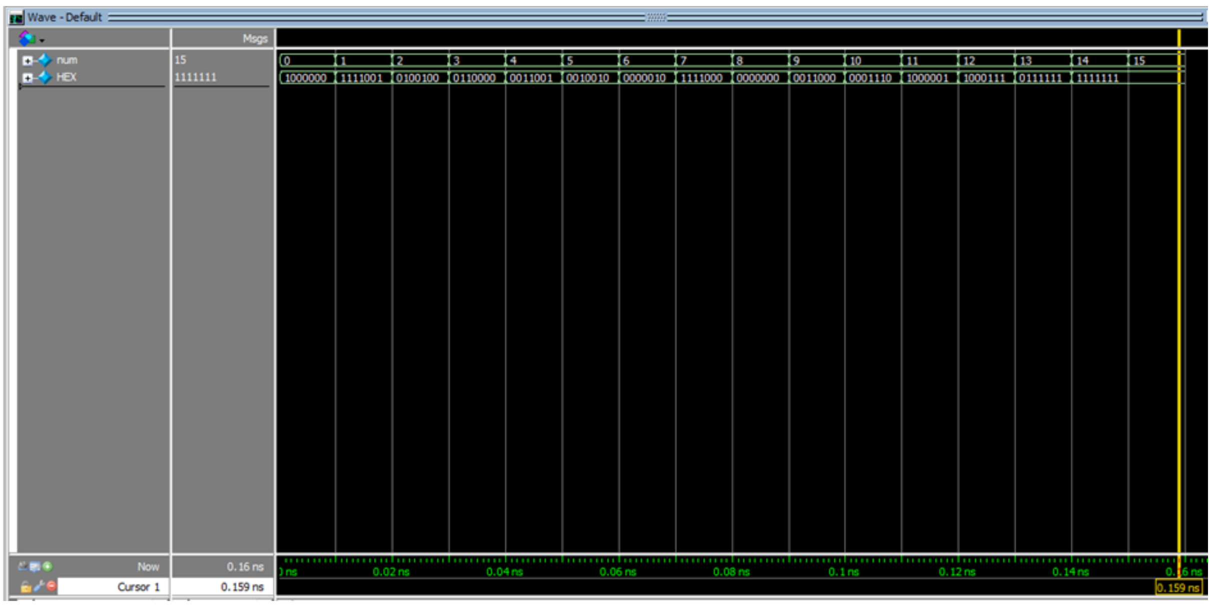

Figure 5. seg7 simulation

Figure 5 shows the generated waveform from the seg7 module which just translates the code from a binary number to its HEX seven segment counterpart to show in each display on the FPGA board

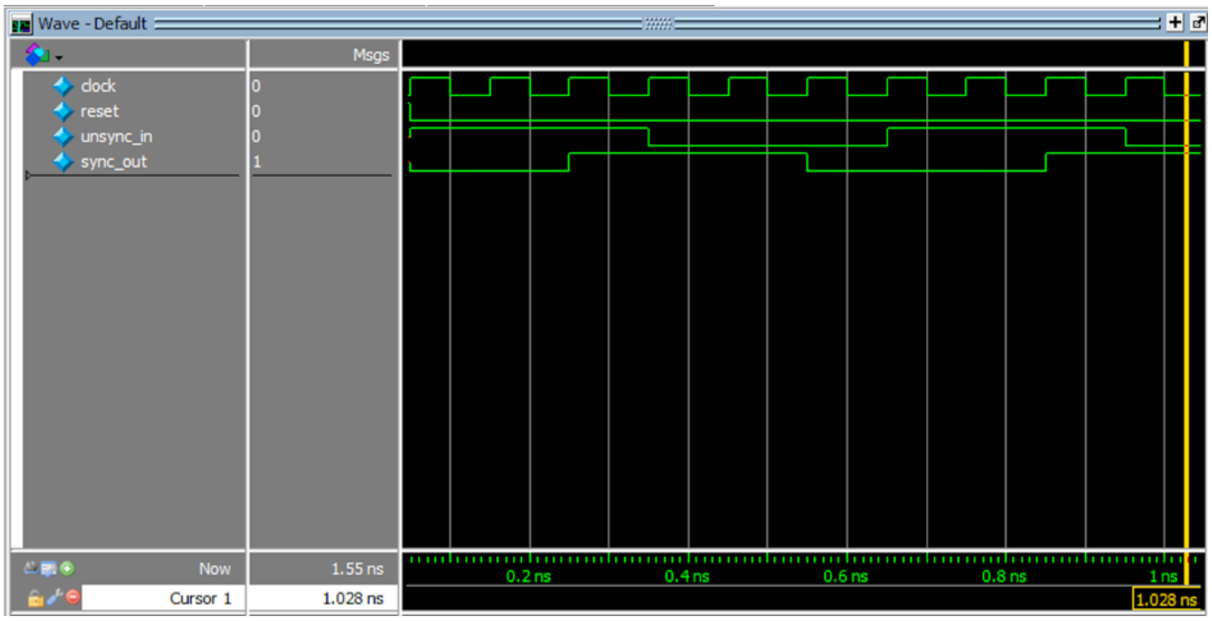

Figure 6. signal_sync module simulation

Figure 6 demonstrates the behavior of the signal synchronizer module, which takes in an unsynced signal from the board and synchronizes it with the clock to avoid metastability issues, from the simulation, an unsynced signal is asserted, the module outputs a synced signal (sync_out).

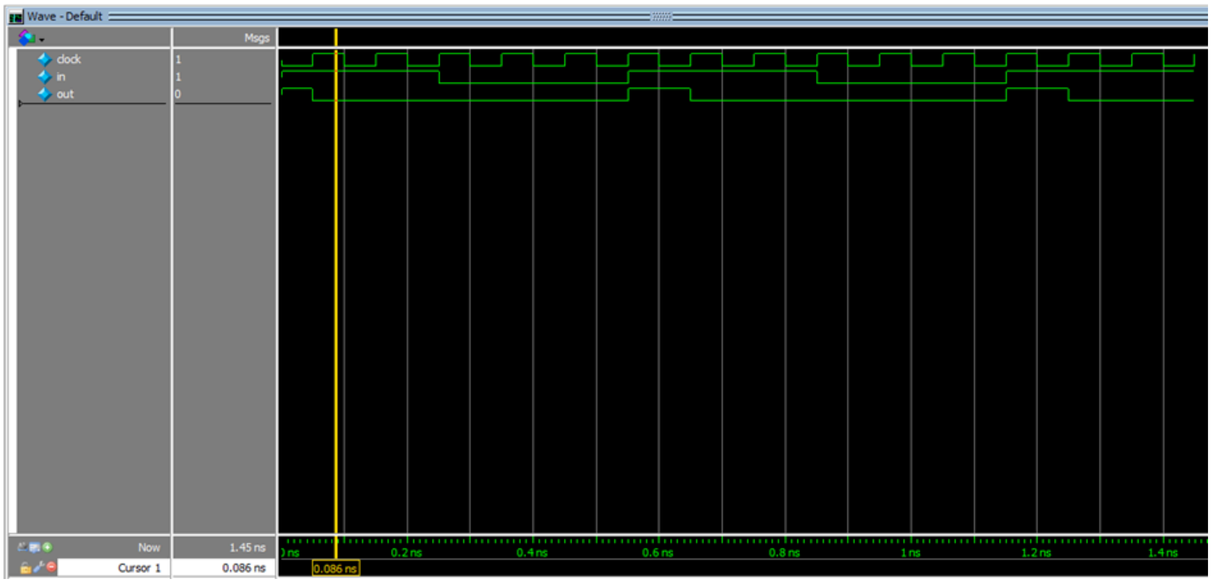

Figure 7. pulse_edge simulation

The pulse_edge module takes in a signal of variable length, like a button press, and translates it into a one pulse signal for the duration of the clock cycle, from the simulation, when the in signal is asserted for multiple clock cycles, the out signal is on for just the first clock cycle and turns off after that.

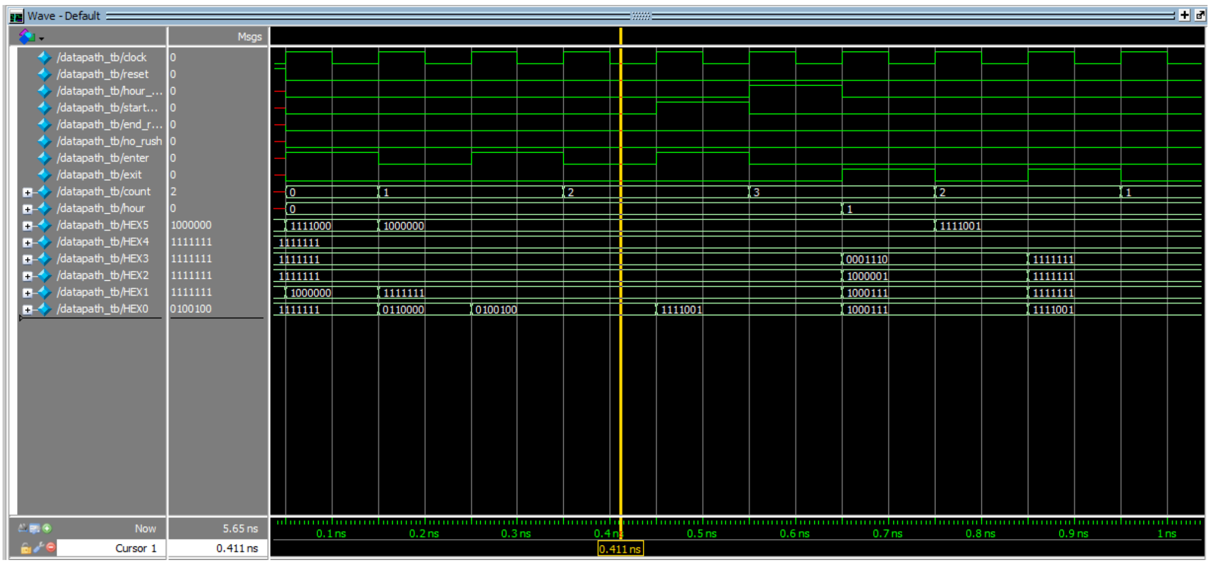

Figure 7. Datapath simulation

The above figure shows the simulation from the datapath, which is also connected to the RAM, the counters and the HEX displays that show the information on the FPGA board.