Procedure

In this task, we progressed to a more integrated system involving real-world simulation and complex interactions on an FPGA platform. This task entailed two main tasks, each with distinct objectives and implementations. Task 1 focused on adapting and enhancing a parking lot system to manage physical inputs and outputs through an FPGA interfaced with a breadboard. Task 2 extended the system's functionality into a 3D simulated environment, where the dynamics of a parking lot were visualized and controlled through simulation software. Both tasks required a meticulous approach to digital design, emphasizing state management, interface configuration, and interactive feedback mechanisms.

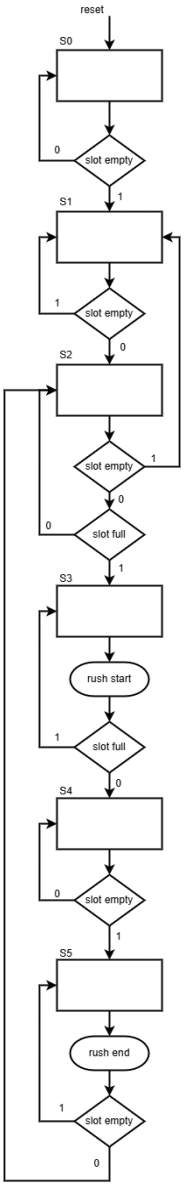

Figure 1. ASMD Chart

This ASMD chart outlines the state machine diagram for controlling parking logic, detailing transitions based on car arrivals and departures. The transitions between states demonstrate the system's ability to manage different scenarios, making real-time decisions based on the current inputs and previous states, crucial for accurate and responsive parking lot management.

Task #1

Task 1 involved configuring the FPGA to interface directly with a breadboard setup, simulating the basic operations of a parking lot system. This setup required reconfiguring GPIO pins to accommodate inputs from virtual sensors and outputs to indicators like LEDs and gates. The primary challenge was ensuring that the FPGA could handle real-world signals such as car detection and gate control effectively.

The FPGA code was adjusted to manage these inputs and outputs accurately, with special attention given to debouncing input signals to avoid erroneous gate triggers. The system was designed to provide immediate feedback on car presence through LED indicators, which represented the occupancy status of each parking spot.

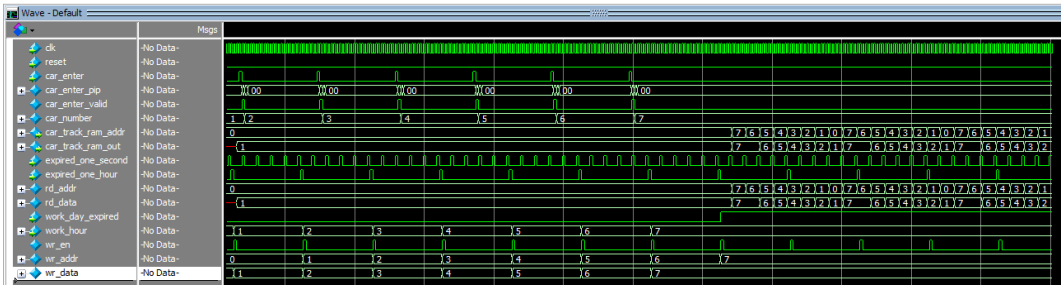

Figure 2. Car_track.sv

From the diagram, it is observed that when no cars are entering the parking lot, the vehicle counter is incremented by 1; each time the current work hour updates, i.e., when the hour counter completes an hour, the number of cars that have entered the parking lot within that current hour is written into RAM. When the work hour ends, data is read from RAM every 1 second from address 7 to 0."

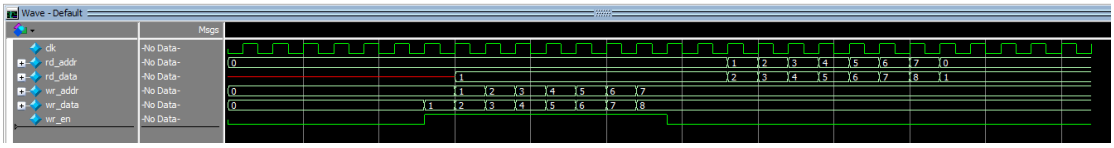

Figure 3. Dual_ram8x16.sv

A dual-port RAM, data is written into the corresponding RAM location when wr_en is high; reading is always enabled, updating the address outputs the corresponding data.

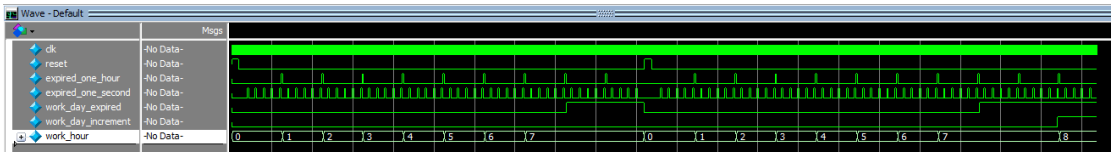

Figure 4. Hour_count.sv

The timer ticks every 20ns, counting 50,000,000 ticks for 1 second; 3600s corresponds to 1 hour. To speed up the simulation, every 5 cycles count as 1 second, and every 5 seconds count as 1 hour (25 cycles).

Task #2

Task 2 expanded upon the basic parking lot management system by incorporating a 3D simulation interface. This task utilized the FPGA to process inputs from the simulation software, translating them into visible changes within the simulated parking lot environment. It involved complex logic handled by a finite state machine (FSM) that determined the availability of parking spaces and controlled the entry and exit gates based on simulated car movements.

The development of the FSM was critical in managing the flow of cars into and out of the lot, ensuring that the simulation reflected accurate parking dynamics. Additionally, this task involved enhancing the user interface to display real-time information about the lot's status, including the number of free spots and notification of full capacity.

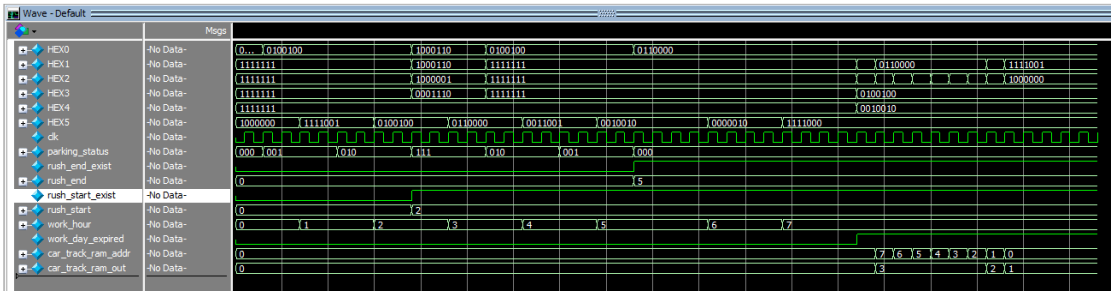

Figure 5. Display. sv

This script controls the HEX display outputs based on the parking lot status. It displays the number of available parking spots, and during full capacity, it shows 'FULL'. This

interactive feature enhances user experience by providing immediate visual feedback on parking lot status, aiding in decision-making during high-capacity periods.

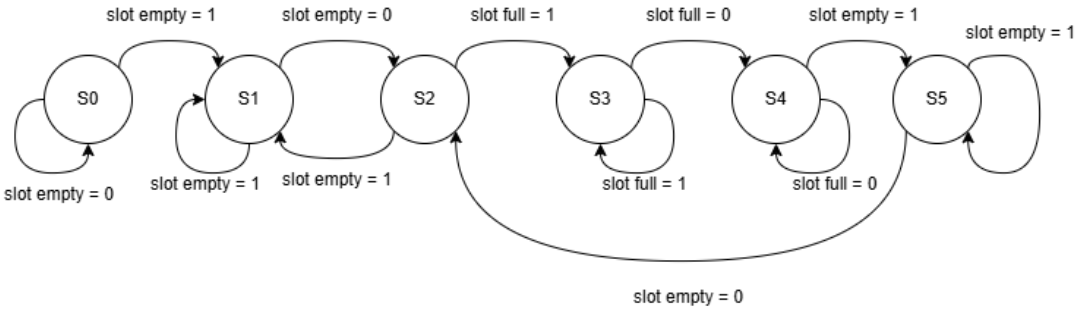

Figure 6. FSM Chart

This Finite State Machine diagram illustrates the comprehensive logic used to manage the parking lot's dynamics in the simulated environment. The FSM is crucial for:

- State Identification: Identifying various states such as 'Empty Lot', 'Partially Full', 'Full', and 'Rush Hour', each representing a specific condition of the parking lot.
- Transitions: Showcasing the transitions that occur based on the arrival and departure of cars. For instance, transitions from 'Empty Lot' to 'Partially Full' occur when the first car arrives, while transitions to 'Full' occur when the parking capacity reaches its limit.
- Actions: Detailing actions taken in each state, like opening or closing gates, updating display boards, and triggering alerts when the lot becomes full.

This FSM not only ensures smooth operational flow within the parking system but also enhances the responsiveness and accuracy of the simulation, directly impacting user experience by providing timely and accurate feedback on the parking lot status.

Results

Task #1

Task 1 effectively demonstrated real-time processing and control of a simulated parking environment using physical components. The results emphasized the system's ability to accurately process inputs and control outputs:

- Performance: The system responded instantaneously to car presences, with LEDs accurately reflecting the occupancy status of each spot. This responsiveness is crucial in real-world applications where delays can lead to bottlenecks.
- Reliability: The setup proved highly reliable, with the debouncing logic effectively preventing any false triggers, ensuring that the gate operations and LED indications were only activated by genuine car presences.

The block diagrams and signal flow charts provided deeper insights into the system's operations. For example, the car tracking diagram illustrated how vehicle detection was handled and integrated into the system, triggering corresponding actions such as gate openings and LED status updates, which were crucial for real-time feedback on parking lot status.

Task #2

Task 2 pushed the boundaries by incorporating a 3D simulation, enhancing the interaction and visualization of the parking management system:

- Dynamic Interaction: The ability to adjust parking settings in real-time within a simulation demonstrated the FPGA's robust handling of dynamic data and complex logic, providing immediate feedback on adjustments.
- Enhanced User Interface: Real-time updates on parking availability and system status enhanced user experience significantly, making the management of the parking space more intuitive and effective.

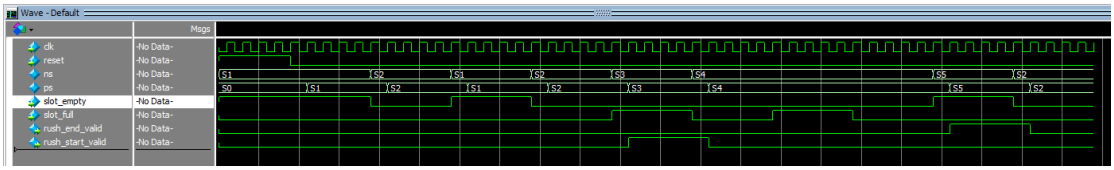

Figure 7. Control.sv

The waveform shows the logic state changes during rush hour detection. Once rush hour starts, the waveform shows the changes in state during rush hour; once rush hour ends, the waveform returns to its normal state.

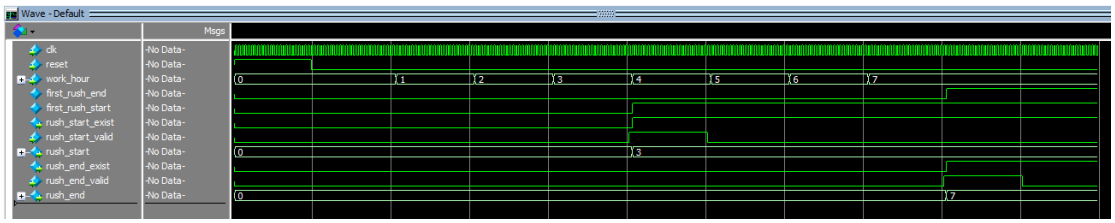

Figure 8. Datapath.sv

When one work hour ends, the system checks if it is a rush hour; if it is a rush hour, the waveform shows the changes during rush hour; once the work hour that is a rush hour ends, it marks the end of a rush hour start/end. If there is no rush hour, it does not mark; rush_start_exist/rush_end_exist mark if there is a start/end of rush hour.

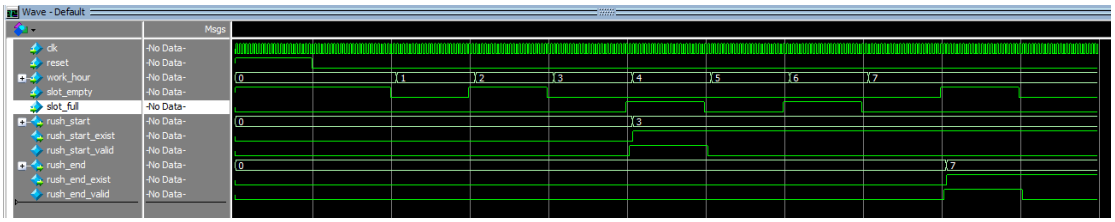

Figure 9. Rush_hour.sv

During rush hour, the control and datapath send a signal to indicate the start, and rush_start_valid becomes high to indicate the start of work hour; similarly, rush_end_valid becomes high to indicate the end of the work hour, marking the end of rush hour with rush_end_exist.

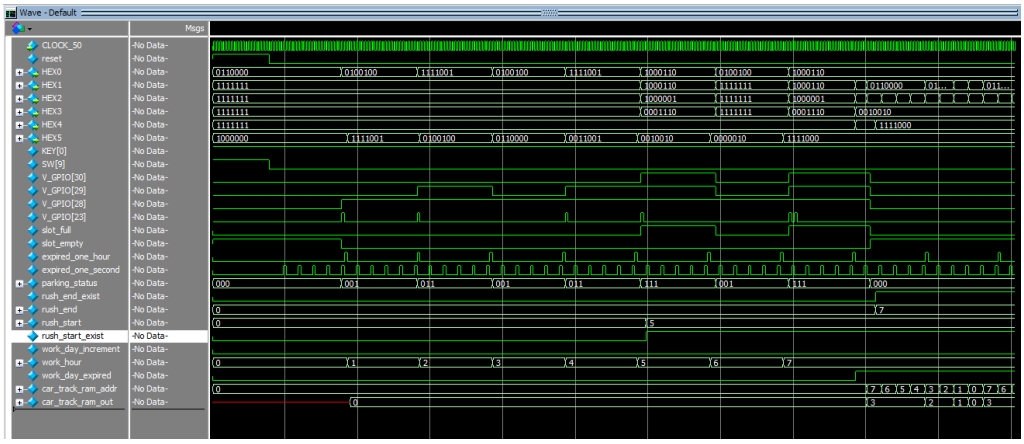

Figure 10. DEC_SoC.sv

When the lot is full, the output state changes, and the entrance display shows 'FULL'; the entrance gate does not open even if there are free parking spaces elsewhere. For example,

when the entrance gate is open, a car enters, and upon closing the gate, the system records a 70% full state in RAM (70_0).

Control.sv, Datapath.sv, Rush_hour.sv, DEC_SoC.sv, these components form the core of the parking system's control logic. Control.sv orchestrates the overall system behavior, coordinating inputs and outputs, while Datapath.sv handles data processing and storage operations. Rush_hour.sv tracks peak parking periods, and DEC_SoC.sv integrates all the components, ensuring coherent operation. Their seamless interaction is vital for maintaining system efficiency and reliability, directly impacting the user experience by providing stable and predictable system behavior.

Final Product

The successful completion of this task has demonstrated the effective use of FPGA technology to design and implement a sophisticated, interactive parking lot management system. This system integrates real-world physical interfacing with dynamic 3D simulation, managed by intricate control logic depicted in the FSM. The system's capability to handle real-time data and user interactions efficiently proves its potential applicability in smart city infrastructure, particularly in automated parking solutions.