EE/CSE 371
October 5, 2025
Lab 1 Report

Design Procedure

This lab tracks the number of cars in a parking lot with a single gate. Two photo sensors, seen in Figure 1, represented by switches, are used to detect vehicles entering or exiting the parking lot, with a maximum capacity of 16. When a car blocks a sensor, the output of that sensor becomes high and an LED lights up. Both sensors must be blocked simultaneously to register as a car - preventing incorrect tracking of pedestrians and other objects. By tracking the sequence of sensor activations, the system can determine whether a car is entering or leaving and display that information via a seven segment display. The design also handles invalid sequences, such as cars setting off sensors before reversing, to prevent miscounting and glitches.

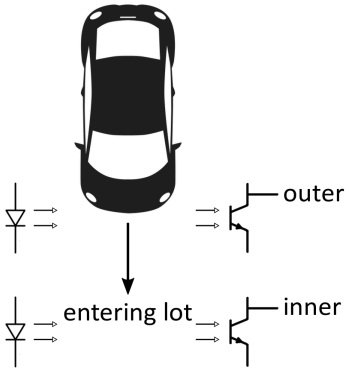


Figure 1: Sensor setup

Overall System

The parkingLotOccupancy_TopLevel module is the top-level integration for the parking lot occupancy system connecting directly to the DE1_SOC in addition to the other modules which control their own individual logic behavior. The top level has inputs for the clock signal, a reset switch, and two sensor switches called OuterSensor and InnerSensor. Its outputs are six seven-segment displays and two LEDs that indicate the state of sensors. Internally, it has two logic signals, incr and decr, and a five-bit count to track the number of cars 0 to16. Additionally, it monitors sensor inputs and generates incr and decr pulses when cars enter or exit. The Counter module updates the count based on these pulses, ensuring the count stays within range. The Display module converts the count into the signals and drives the seven-segment display. The LEDs ledOne and ledTwo are directly driven by the OuterSensor and InnerSensor inputs in this module, providing feedback when the sensors are being blocked. This module integrates all submodules to provide a fully functional parking lot occupancy display system. It is connected to the DE1_SoC, which is connected to a remote FPGA and breadboard.

Block Diagram

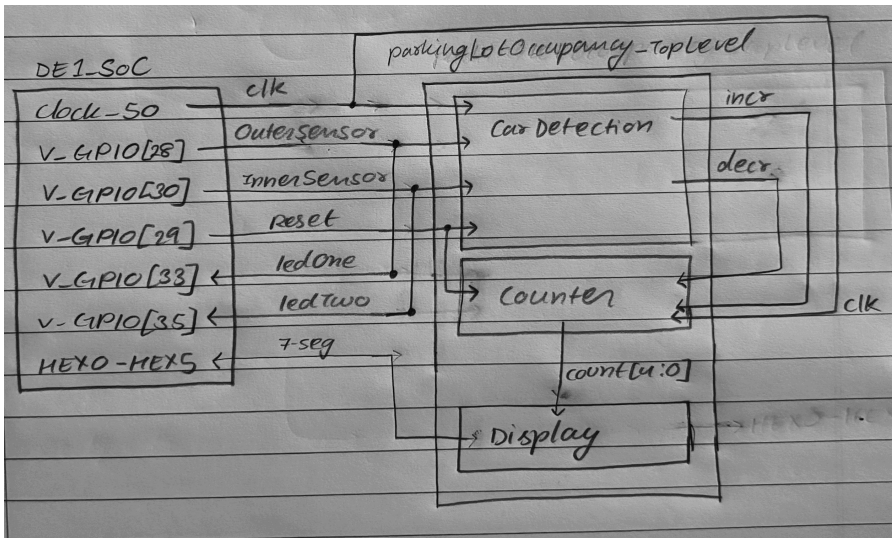


Figure 2: Block diagram for Parking Lot Occupancy Tracker

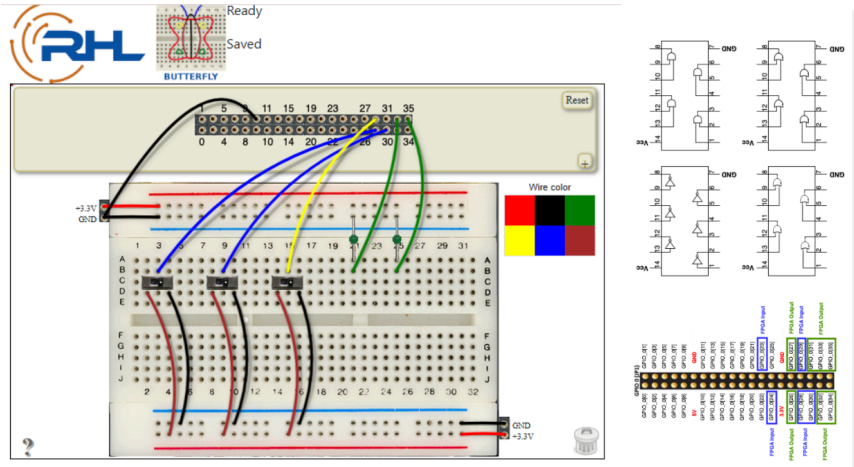


Figure 3. Wiring for the breadboard using GPIO board in LabLands (utilizing pins 28, 29, 30, 33, 35)

Task 1 - Car Detection

FSM

A Mealy FSM, Figure 4, serves as the main logic for car detection in the CarDetection module. The input and output format is INNER/OUTER PLUS/MINUS representing the inner and outer sensors as well as incrementation and decrementation once a corresponding entry or exit sequence is recognized.

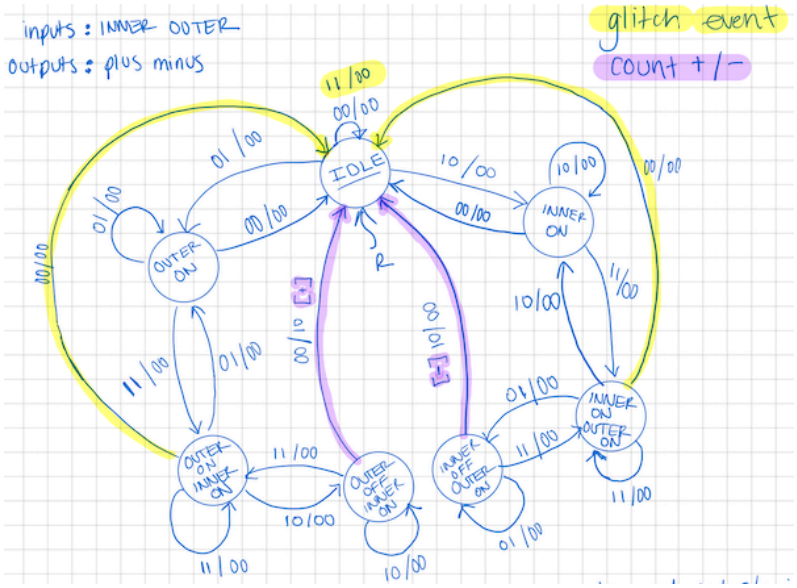


Figure 4: Mealy FSM for Car Detection Logic illustrating the seven states alongside transitions.

The FSM relies on the following behavior:

- Input - INNER and OUTER - 1 bit each representing inner and outer sensor
- Output - PLUS and MINUS - 1 bit each for incr and decr

The FSM starts off in an ‘IDLE’ state, at which point one of two sensors can be set high with an input of either ‘10’, ‘01’ or moving to either the ‘InnerOn’ or ‘OuterOn’ state. If no input is detected ‘00’, or if two inputs are detected simultaneously ‘11’ (a rare glitch event color coded in yellow), the FSM remains in the ‘IDLE’ state. In order for incrementation or decrementation to occur (labeled in purple) a user must activate one of two paths: InnerOn, InnerOnOuterOn, InnerOffOuterOn, IDLE or OuterOn, OuterOnInnerOn, OuterOffInnerOn, IDLE. Successful navigation of both paths increments the car counter once the last sensor moves from high to low outputting either ‘01’ or ‘10’ depending on incrementation or decrementation. All other paths output a ‘00’ since neither incrementation or decrementation should occur. All other paths including traversing backwards through states are accounted for and the logic ensures that cars that trigger the sensor system and then reverse, are not taken into account when calculating the count. The logic takes into account glitch events (labeled in yellow) and this FSM structure ensures that the output pulses are synchronized occurring only once per valid movement, preventing false counts from partial crossings.

The CarDetection module monitors two sensors OuterSensor and InnerSensor, and generates one cycle pulses for incr (increment) or decr (decrement) signals when triggered. It uses a 3-bit enumerated state variable with seven defined states that represent different sensor combinations as a car moves through an entry and exit path. Transitions between states are based on sensor values of 1 or 0, tracking whether a sensor is triggered first, in which order and when each sensor turns off. An always_ff block updates the present state on each rising clock edge. An always_comb block

determines the next state according to described sensor conditions in the FSB diagram. A separate always_comb block generates an incr pulse when the FSM moves from OuterOffInnerOn to Idle, car has entered, and a decr pulse when transitioning from InnerOffOuterOn to Idle, car has exited.

Task 2 - Counter

A 5-bit counter is implemented that monitors parking lot occupancy. It has four inputs: clk, reset, incr, and decr. The output is count, a 5-bit value ranges from 0 to 16. If reset is high, count set to 0, else the count increments by 1 if incr is high and the count is below 16, or decrements count by 1 if decr is high and count is above 0, ensuring the counter never goes below zero or above 16.

Task 3 - Seven Segment Display

The Display module converts a 5-bit count value into signals for a 7-segment display. The input is count. The outputs are HEX5 - HEX0. The module initializes all displays to blank (1111111 active-low). If count is 0 or below, the message CLEAR 0 displays. If count is 16 or higher, FULL is displayed across four displays. For counts between 1 and 15, the number is displayed as a decimal, if the count is 10 or higher, the tens digit HEX1 shows '1', the ones digit HEX0 logic shows the remainder modulo 10. This thorough design ensures that count is displayed accurately whether empty, full or partially filled.

Task 4 - Top Module

The parkingLotOccupancy_TopLevel module is the integration point connecting the system logic in their respective modules to the DE1 board and serves as the top level of logic control for the system. It receives four inputs from the DE1: clk, reset and two sensor inputs (OuterSensor and InnerSensor). Internal logic include signals incr and decr represent pulses indicating when a car enters or exits, as well as a 5-bit count register that stores the current car total. Additionally, ledOne and ledTwo are directly tied to the OuterSensor and InnerSensor signals to visually indicate sensor activation on external LEDs. By utilizing a top module, synchronization, sensor feedback, and display outputs are managed without the risk of interrupting or interfering with hardware connected components on the DE1 board.

Results and Testing Report

Top Level

Note the DE1 was solely used to connect GPIOs and hardware components, it is not the top level module for logic. The testbench for the top level was designed to go through paths of the FSM, and test the general behavior of the code including sensors, LEDs, HEX's, incr, decr, reset and count. Because extensive testing was done with each submodule, exhaustive testing was not performed in the top module, only enough to ensure accurate behavior and connections between modules. The testbench initialized variables to 1 before forcing a reset. Two cars entering were simulated,

followed by a vehicle entering partially, followed by multiple cars leaving, ensuring that the count accurately reflected sensor feedback and did not drop below 0 even when proper sensor sequences were activated. Count states go through the proper increment and decrement coinciding with sensor activations. HEX displays properly display the correct count and LED's accurately respond to sensor changes as well - they continue to light up even count < 0.

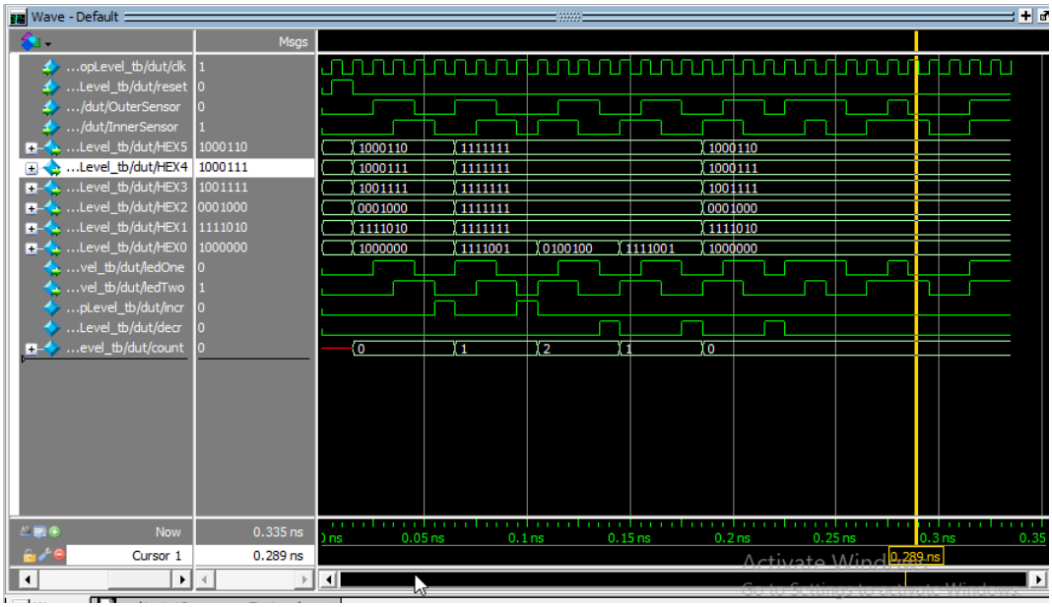


Figure 5: Top Level Waveforms

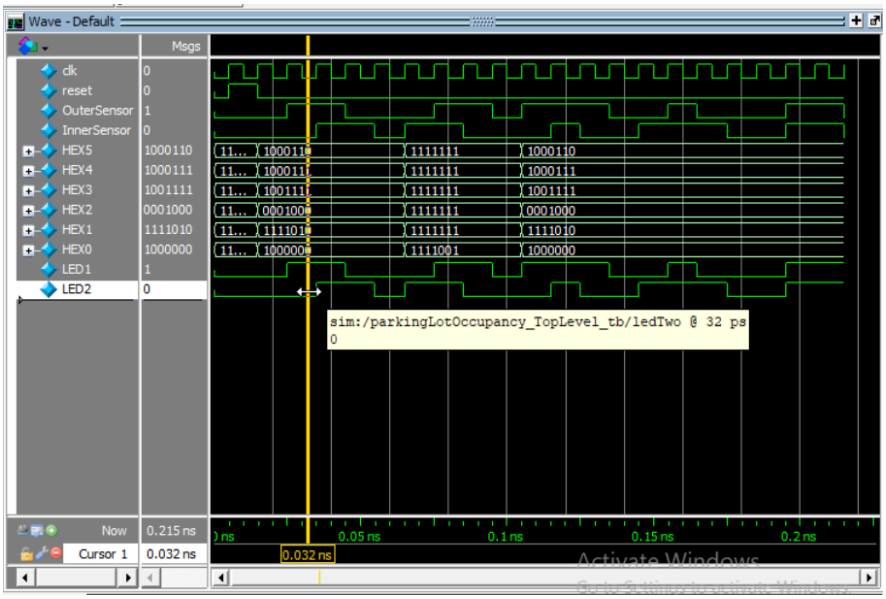
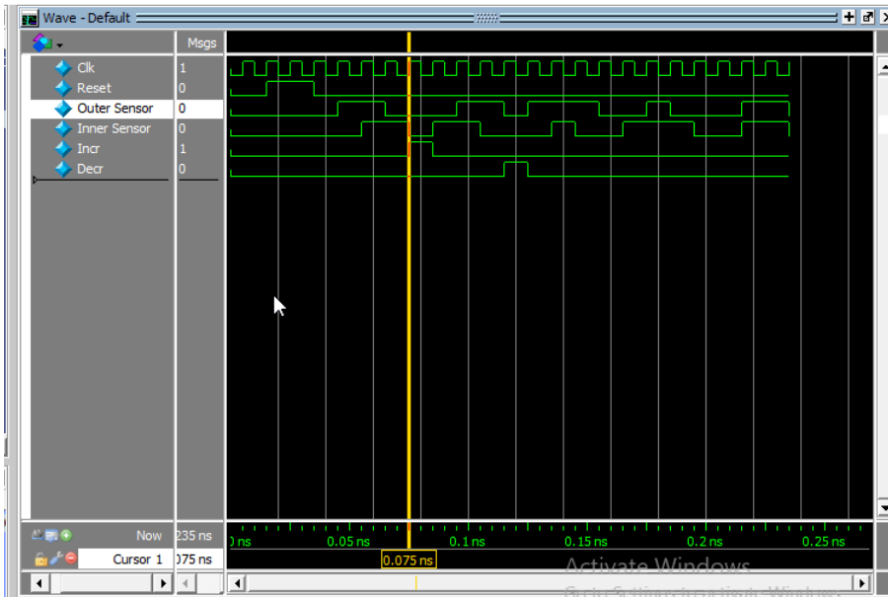


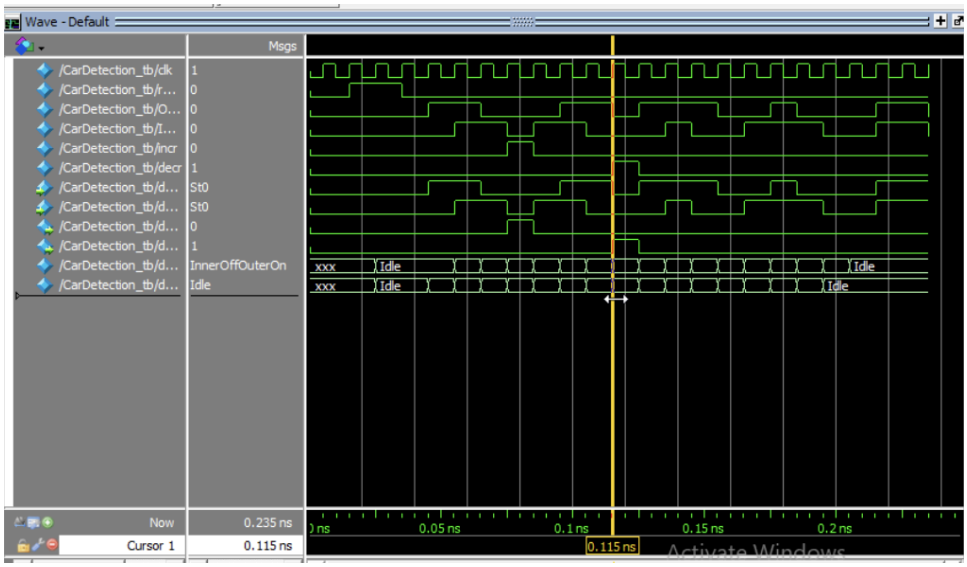
Figure 6: Top Level Waveforms

Car Detection

Extensive testing was performed in this testbench due to the bulk of the logic residing here. Sensor activation patterns first simulate a car entering the lot, with the outer sensor triggered first followed by appropriate sequence, the incr signal is high and count is incremented. Next, it simulates a car exiting, the inner sensor triggers first followed by appropriate sequence, decr signal is high and count decrements. This testbench shows both output signals and following sensor inputs predictably. Additional tests check more

complex behaviors, such as a car beginning to enter or exit but reversing before completing the action, which should not trigger any count changes and does not. Finally, glitch tests simulate random sensor activations to ensure the FSM interprets valid car movements and rejects invalid sequences.





Figures 7 & 8 : Car detection sensor testing

Counter

This test bench verifies the counter's ability to increment properly by repeatedly asserting the incr signal for eighteen clock cycles while keeping decr low, printing the count value at each step to confirm it increases sequentially. This both shows proper counting while also demonstrating that count cannot exceed 16 per design. The second part tests the counter's decrement behavior by asserting decr while disabling incr, again repeating for eighteen cycles and displaying resulting count values to verify proper counting down and demonstrating that count cannot drop below 0 per design. These

tests ensure that the counter responds timely and correctly to increment and decrement inputs, resets properly, and stays within counter thresholds.

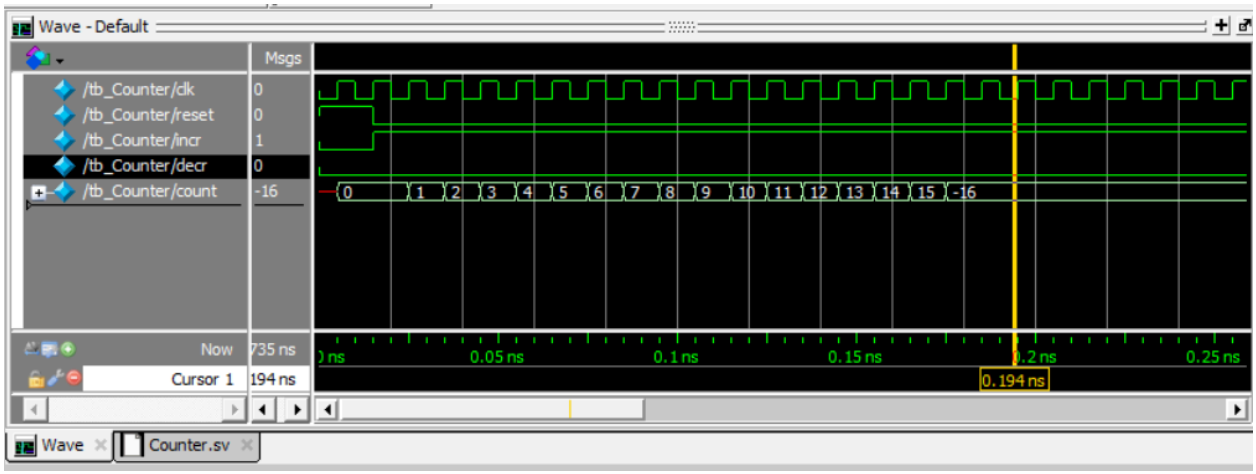


Figure 9: Shows counter incrementing 0 to 16, maxing out at 16 even when incr is high

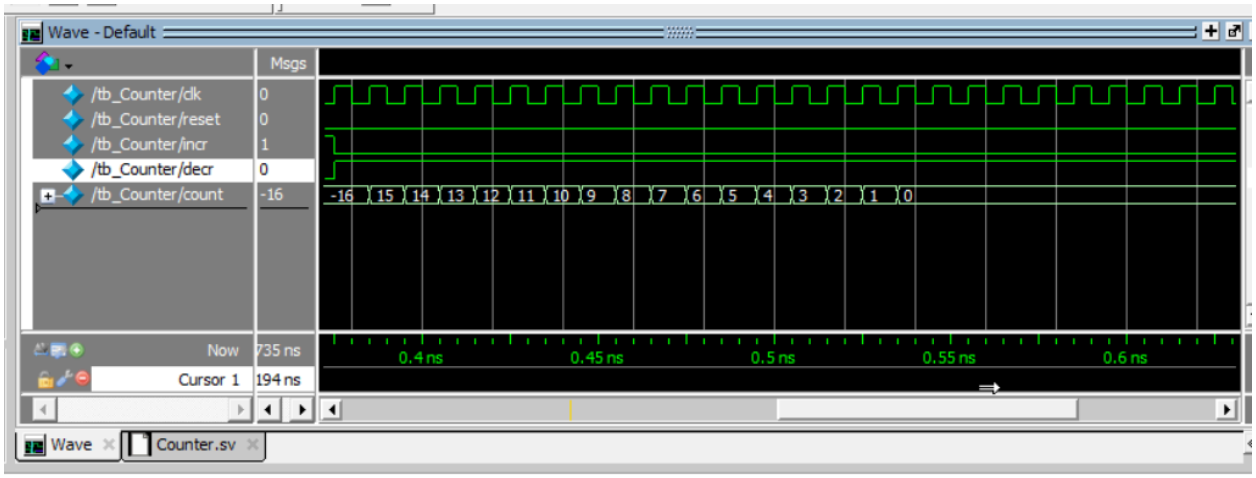


Figure 10: Shows counter decrementing 16-0, stopping at 0 even when decr is high

Display

This testbench increases the count value from 0 to 16, pausing briefly between each step to allow time for the display outputs to update. It then reverses the process, counting back down from 16 to 0. This test ensures the display correctly represents every possible count value in both ascending and descending order, including CLEAR0 and FULL.

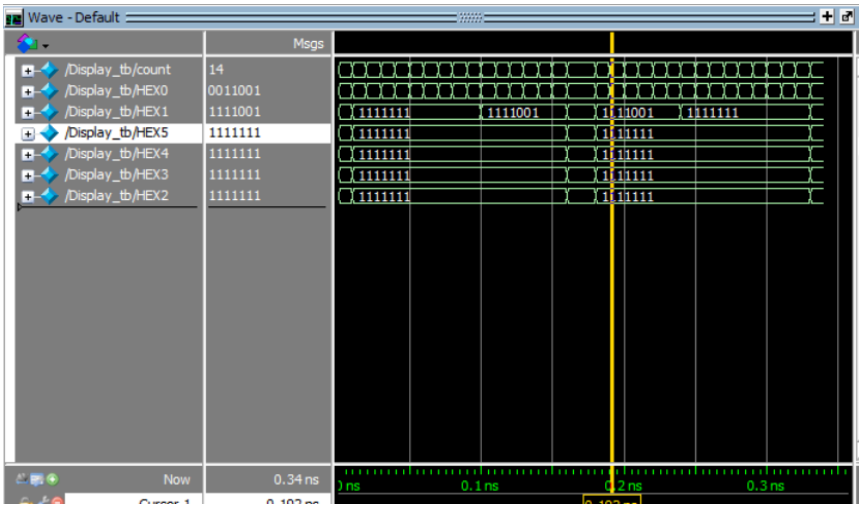


Figure 11: Display testing

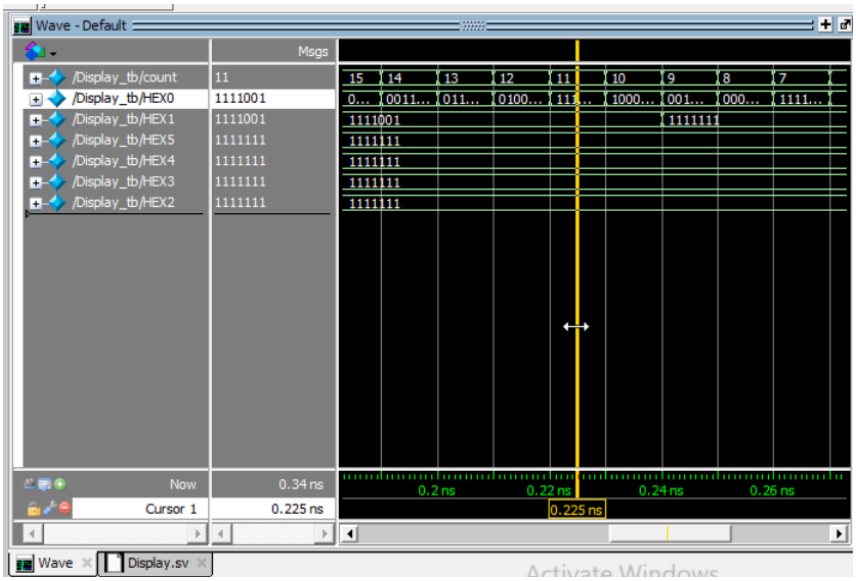
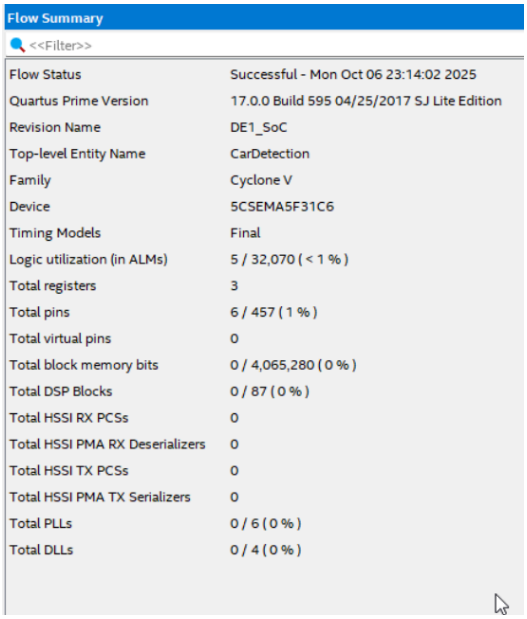


Figure 12: Display testing

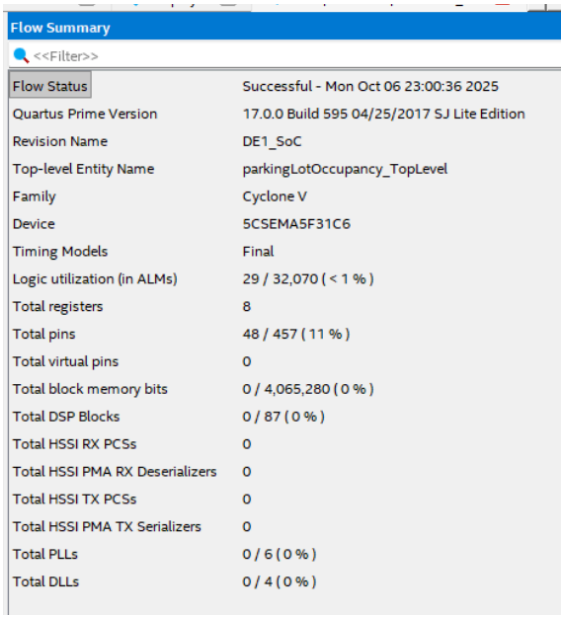
Flow Summary



Flow Summary

<<Filter>>

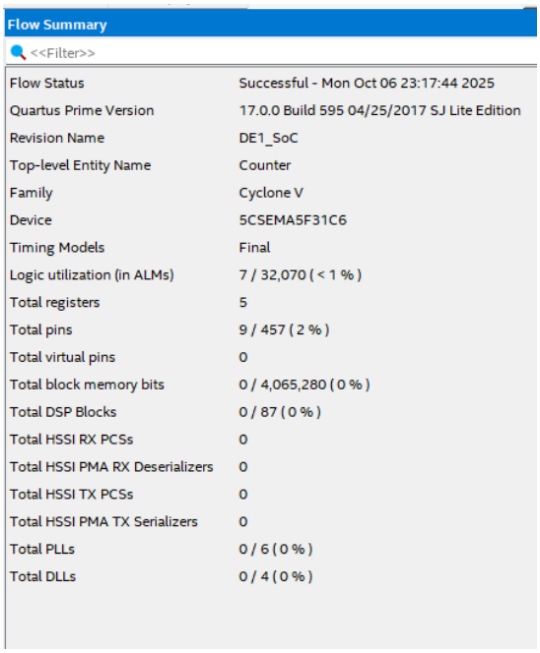
Flow Status	Successful - Mon Oct 06 23:14:02 2025
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	CarDetection
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	5 / 32,070 (< 1 %)
Total registers	3
Total pins	6 / 457 (1 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)



Flow Summary

<<Filter>>

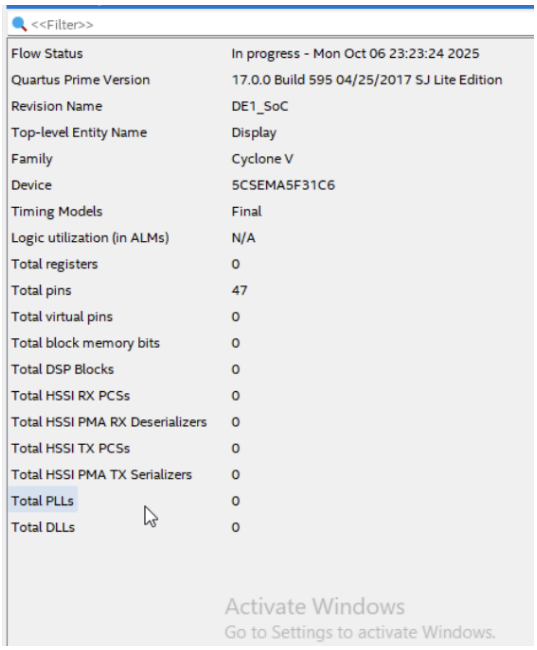
Flow Status	Successful - Mon Oct 06 23:00:36 2025
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	parkingLotOccupancy_TopLevel
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	29 / 32,070 (< 1 %)
Total registers	8
Total pins	48 / 457 (11 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)



Flow Summary

<<Filter>>

Flow Status	Successful - Mon Oct 06 23:17:44 2025
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	Counter
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	7 / 32,070 (< 1 %)
Total registers	5
Total pins	9 / 457 (2 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)



<<Filter>>

Flow Status	In progress - Mon Oct 06 23:23:24 2025
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	Display
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	0
Total pins	47
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Flow Summary

<<Filter>>

Flow Status	Successful - Mon Oct 06 23:27:51 2025
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	DE1_SoC
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	29 / 32,070 (< 1 %)
Total registers	8
Total pins	79 / 457 (17 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)