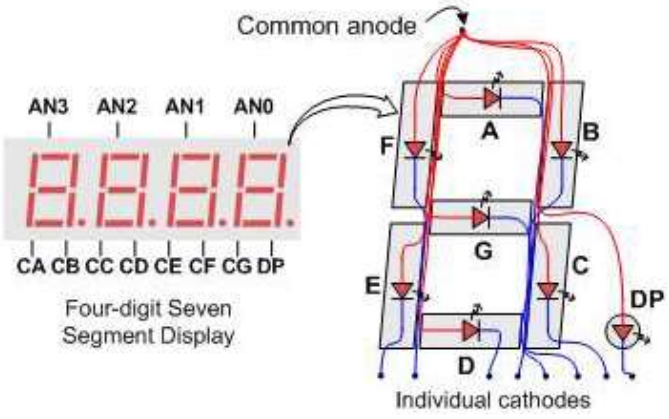
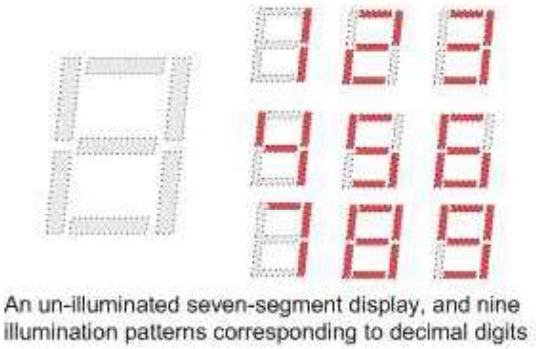
Introduction

The aim of this lab is to learn how to design finite state machine that performs a specified behavior and to use the Xilinx ISE software to build and test a finite state machine. Finite State Machines (FSMs) are a powerful tool that can be used to model many real- world systems and are particularly useful for the behavioral modeling of sequential circuits. An FSM has a finite number of 'states' and can be in any one of these states at a given time. There are two kinds of FSM machines: Moore Machine and Mealy Machine. Moore machine is a FSM in which its output only depends on which state it is in, but a Mealy machine its output depends on both the current state and input.

Requirement:

For this lab, we are required to design an FSM to model a parking meter which simulates coins being added and displays the appropriate time remaining. The time remaining will be displayed on a four-digit common anode seven-segment LED display on a Nexys3 board. Each of the four digits is composed of seven segments arranged in a pattern.





Our FSM Machine should have the following specification:

- The input buttons represent different coin denominations and the seven-segment LED display will display the time remaining before the meter expires in seconds:
  inputs:
  - add1: add 60 seconds
  - add2: add 120 seconds
  - add3: add 180 seconds
  - add4: add 300seconds
  - rsrt1 : reset time to 16 seconds
  - rst2 : reset time to 150 seconds
  - clk : frequency 100HZ
  - rst : reset to initial state

  outputs:
  - led_seg(7-bit vector): displays the actual value fed to the 4 segments corresponding to the digits being displayed.

    - a1,a2,a3,a4 : represents the anodes of the seven-segment display. (1 bit each)
    - (val1, val2, val3, val4): display the actual digit in BCD (binary coded decimal) corresponding to each of the segments.
- In the initial state, the seven-segment displays should be flashing 0000 with period 1 sec and duty cycle 50% (on for 0.5 sec and off for 0.5 sec).
- When any the inputs (add0, add1, add2, or add3) is high, the display adds to the corresponding time and starts counting down.
- When less than 180 seconds are remaining, the display should flash with a period of 2 seconds and 50% duty cycle.
- The max value of time will be 9999 and any attempt to increment beyond 9999, should result in the counter latching to 9999 and counting down from there.

Design Description:

For the design of this FSM machine, I followed the guidance of how to create an FSM in Verilog provided in the project specifications. My FSM is built on using about 9 states that aim to correctly model the behavior of the parking meter described in the project specifications. This FSM is a Moore machine which output depends only on the current state regardless of the input.

Descriptions of states and transitions of parking meter FSM:

Initially, the machine would begin at the initial state (state 4'b0000) in which the output will be flashing 0000 at 1HZ with 50% duty cycle. If the machine receives any coins, the corresponding time is added accordingly.

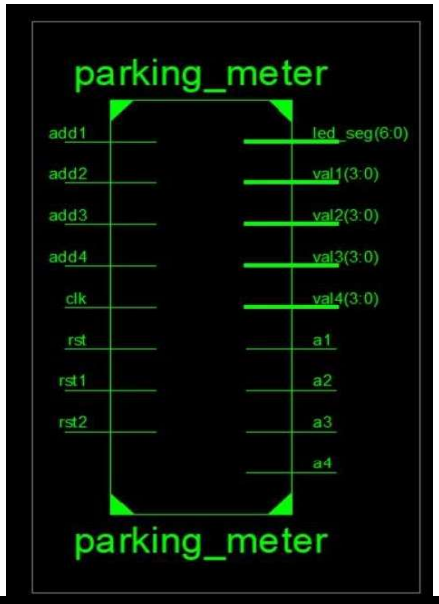
If either rst1 or rst2 are set to 1, The machine will be set to the two other states (states 4'b0001 and 4'b0010) in which in one sate the remaining will be set to 16 seconds and the other would be set to 150 seconds respectively. By counting the time down, the remaining time would be flashed at 0.5HZ since the time not equal to zero and it is less than 180 seconds.

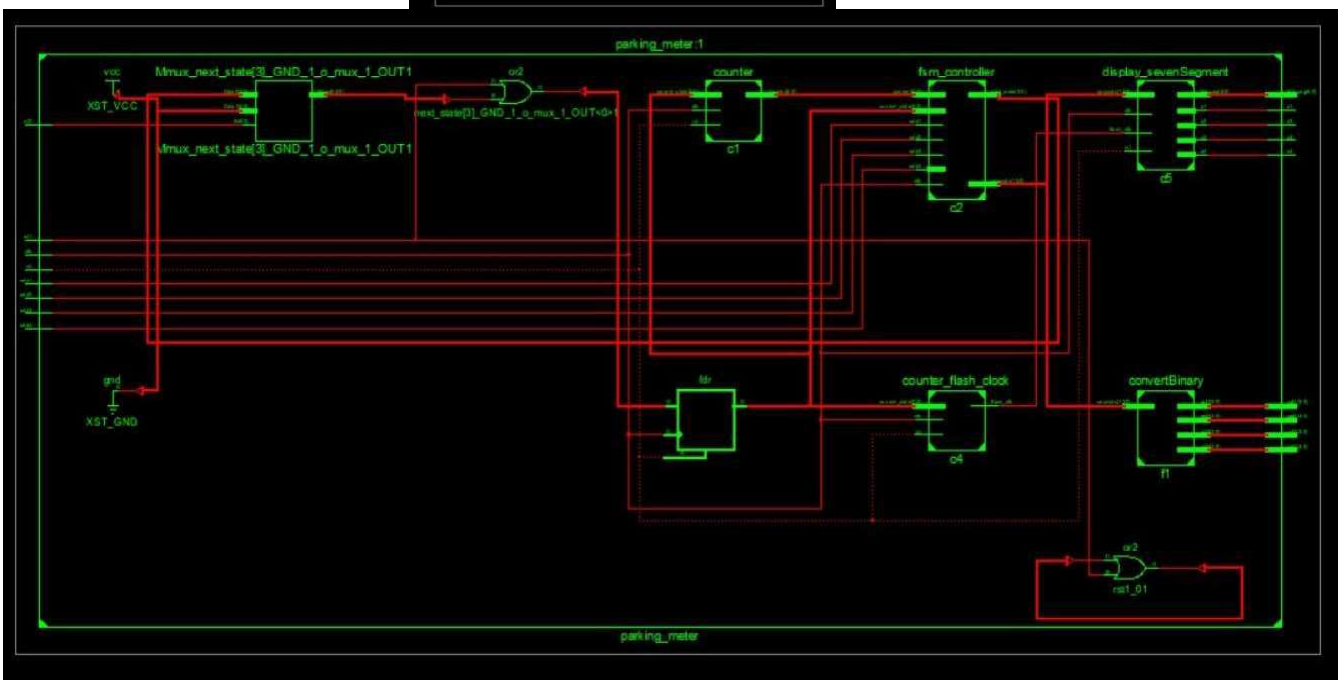
If one of the inputs add1, add2, add3, add4 is set to 1, The machine would move to the next state in which the corresponding time is added to the parking meter and continue decrementing the parking time by 1 (states 4'b0011, 4'b0100, 4'b0101, and 4'b0110). The display of the time would also be displayed in the same manner we displayed it earlier.

If no coins are added. The machine would be in a state (state 4'b1000) in which the time is decremented by one, and the machine would check if the remaining time is equal to zero or less than 180 to display the remaining time with the correct frequency.

When the time is below 180, the machine would transition to a state (state 4'b0111) in which the remaining time is decremented and flashed at 0.5HZ with 50% duty cycle. If any coin added, the state would move to the next state modeling the state of the corresponding add, and the display would be flashed accordingly. If the remaining time became 0 the machine would move to the initial state and the display would be flashed accordingly.

The RTL schematic:





The parking meter as a top module contains 6 modules:

- Counter module: which generates a signal to tell other modules when the machine should decrement the seconds (the remaining time).

- fsm_controller module: this module is responsible in updating the current_state and next_state based on the current state, the input signals.

- time_calculator: is a module instantiated by fsm_controller module which is responsible in calculating the remaining time based on the counter module signal, and the current_state.

- convertBinary module: which is responsible in converting the binary number seconds to 4 separate digits which outputs val1, val2, val3, val4.

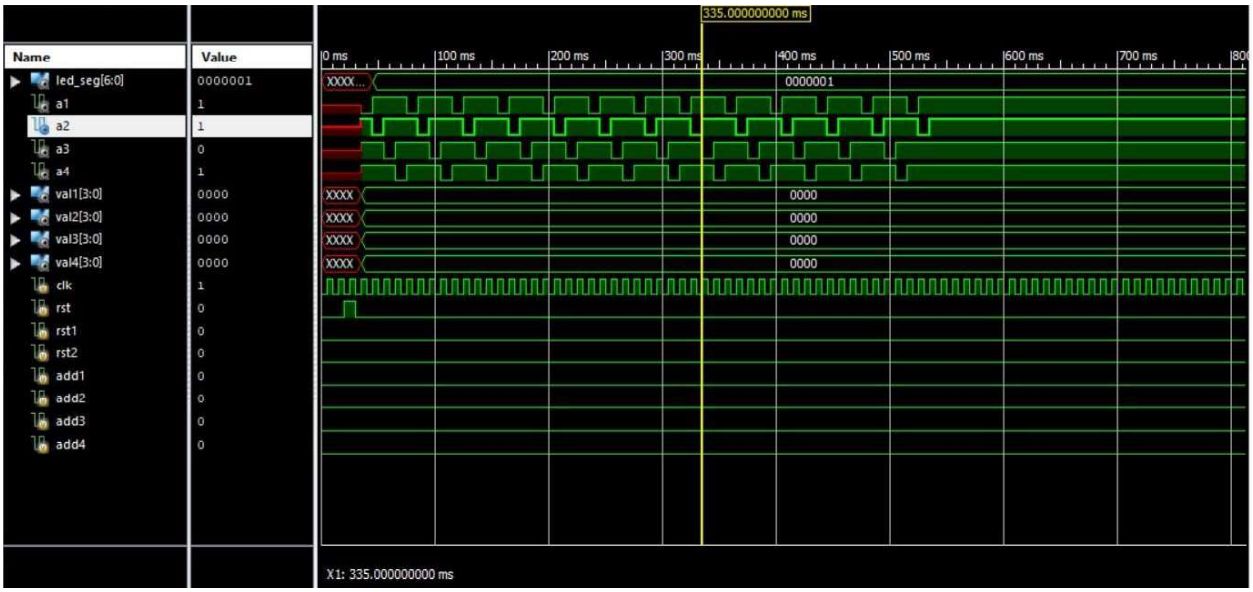
- display_sevenSegment: which would output the 4 anode signals (1-bit for each) and led_seg (7-bit vector) based on the machine's current_state, the current remaining time and the flashing counter module signal (counter_flash_clock).

Simulation:

To test if our parking meter works appropriately, I designed the testbench to test all the cases by providing a 100HZ clock input. Here are the test cases:

- Case of adding 60 seconds after the initial state

I simulated this case by, first setting the rst input to 1 to force the initial state and the remaining time to be in 0. Then, I let clock run for a while to check that the time remaining stays in 0. Then I set add1 to 1 to add 60 seconds to the remaining time. So, we can see from the waveform that as the remaining time is 0 after the rst is set to 1. Also, we see that led_seg vector is set to 0000001 and we can see that the four signals of anodes are rotating to be in low to display 0 in all four seven-segments displays.



After putting add1 to high, we see that 60 seconds time are immediately added (right in the next cycle) as expected, and one cycle after the led_seg and the anodes changes the and alternates to display the digits 0 and 6 as shown in this waveform.

- Test of the time remaining decrement and the flashing if the remaining time is less than 180:

In this case, I just let the clock keep running for little bit. As we can see that the remaining time is decremented properly every 1 sec. We also see that the anodes are all set to 1 whenever the value of the time remaining time is odd which doesn't display anything, while the nodes are changing and alternating the zero value when the value of the remaining time is even which displays the value of the remaining time. This models the effect of having: 58(even number), blank, 56(even number)….

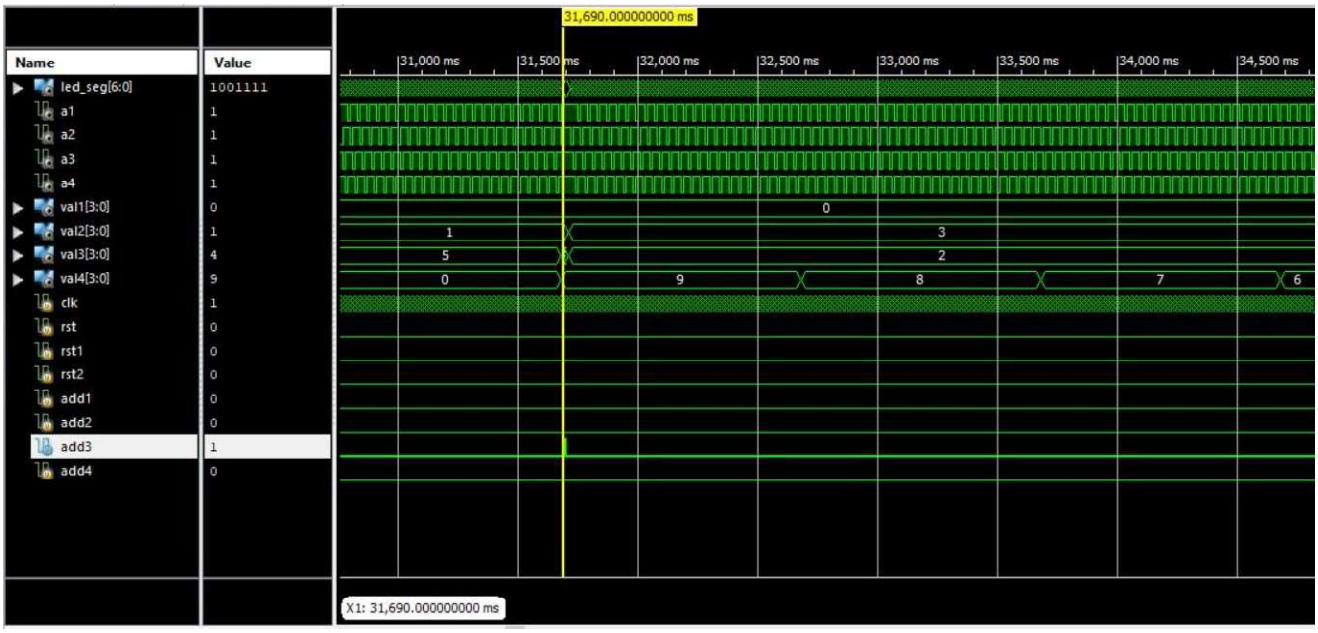
- Adding 120 secs by setting add2 to 1 (the remaining time is still under 180)

I this case is simulated by setting add2 to 1 which will add 120 secs to the remaining time in the 49th second which made the remaining time be 169 secs. Also, the flashing behavior remains the same as the remaining time is still less than 180.

- Adding 180 secs by setting add3 to 1:

This case is simulated by setting add3 to 1 which will add 180 secs to the remaining time. At the 149th seconds the add3 input is set to 1 which would increase the remaining time to 329 secs as expected. W

can also see that our anodes behavior has changed showing that our seven-segment display is not flashing anymore (blanking in odd numbers), and the numbers are being displayed without flashing.



- Adding 300 secs by setting add4 to 1:

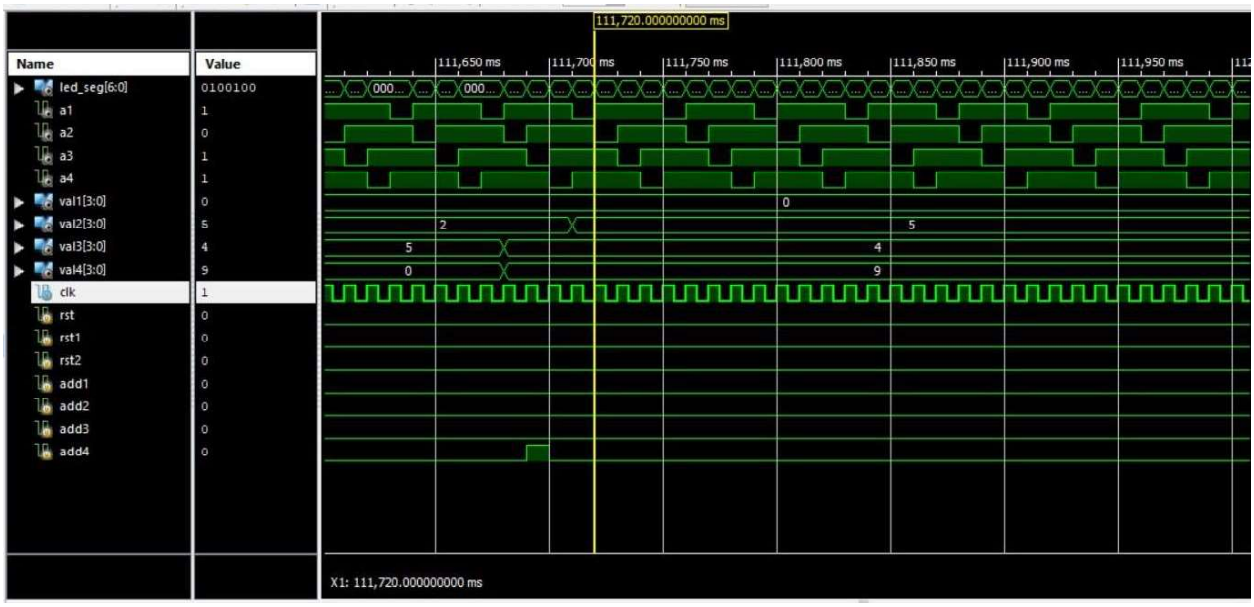
This case was simulated by setting add4 to 1 which will add 300secs to the remaining time. At the 249th second the add4 input is set to 1 resulting in adding 300 as expected to make the remaining time to be 549secs. Notice that the anodes are all alternating showing that the seven-segment display is not flashing anymore as the remaining time is more than 180.

Note that: the time remaining is being continuously decremented.

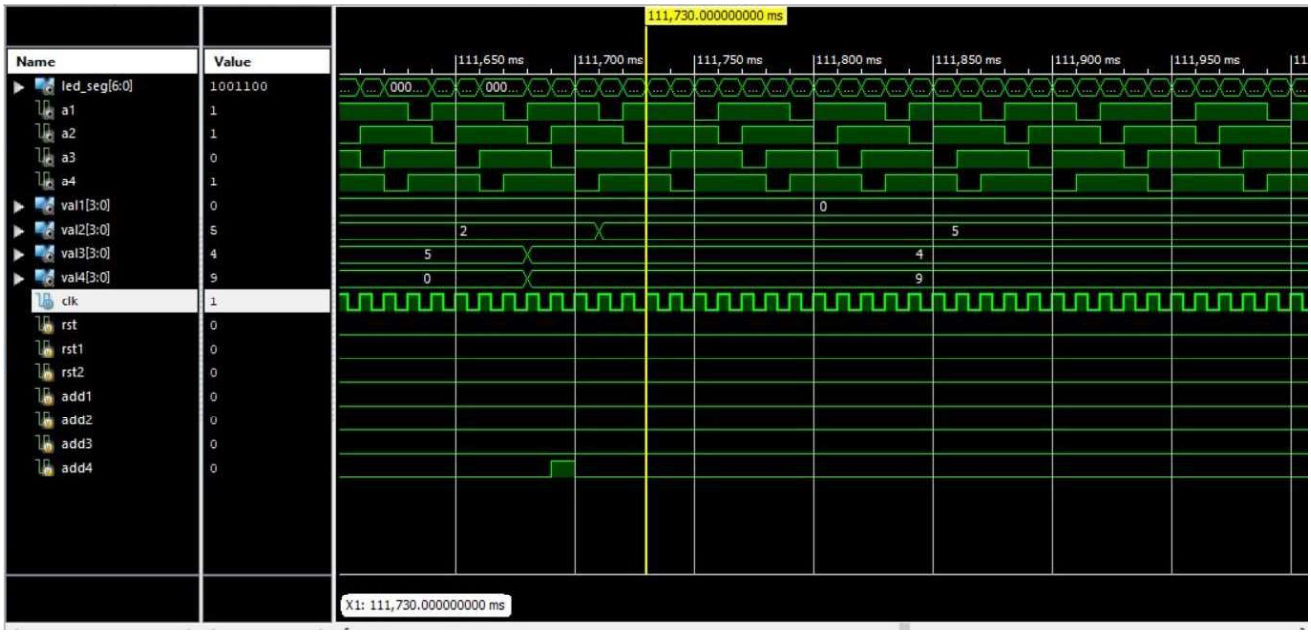
- Testing the seven-segment display:

After the 300 secs were added, we see that anode a1 is set to low while the other anodes are set to high for one cycle. This represents the display of the left most digit on the display. In this case this digit should be 0. If we check the led_seg we see that the value is set to 0000001 which represents effectively 0 in the seven-segment display.

After one cycle, the anodes would alternate the 0 switching 0 to be in a2 and having all the other anodes to be high. This show represents that the next digit in the right of the most left digit should be displayed. This number is displaying the digit 5.



In the next cycle, similarly a3 would be set to 0 to display the digit 4.

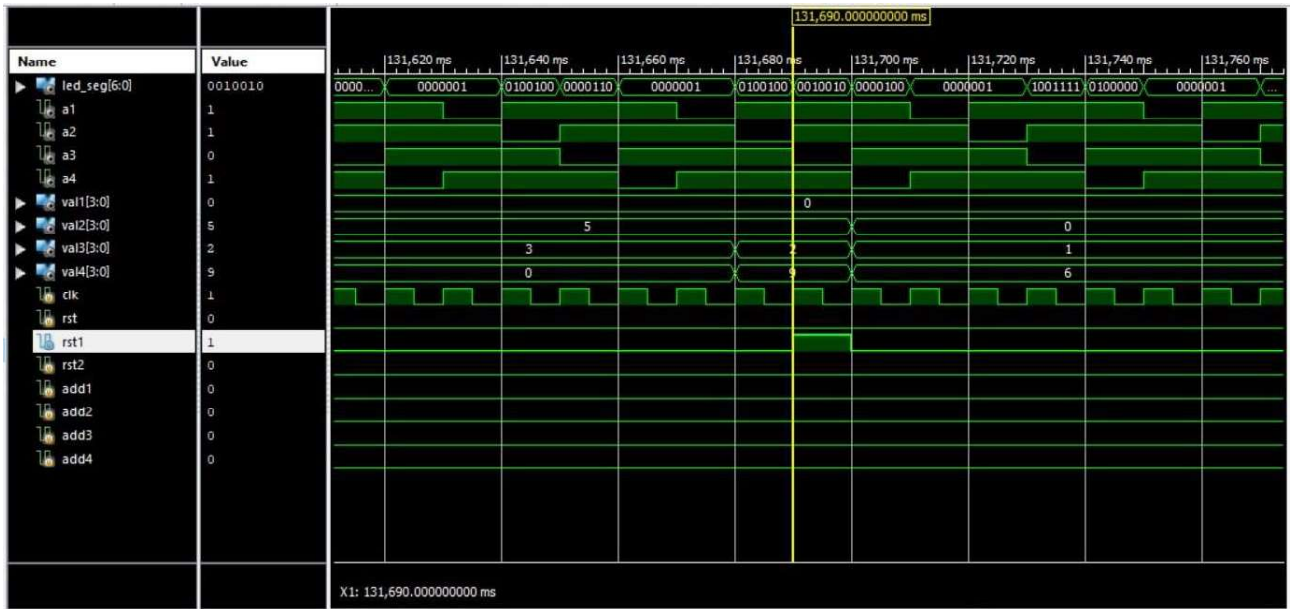


In the next cycle, a4 would be set to 0 to display the digit 9.

This shows that our seven-segment display displays our digits correctly.

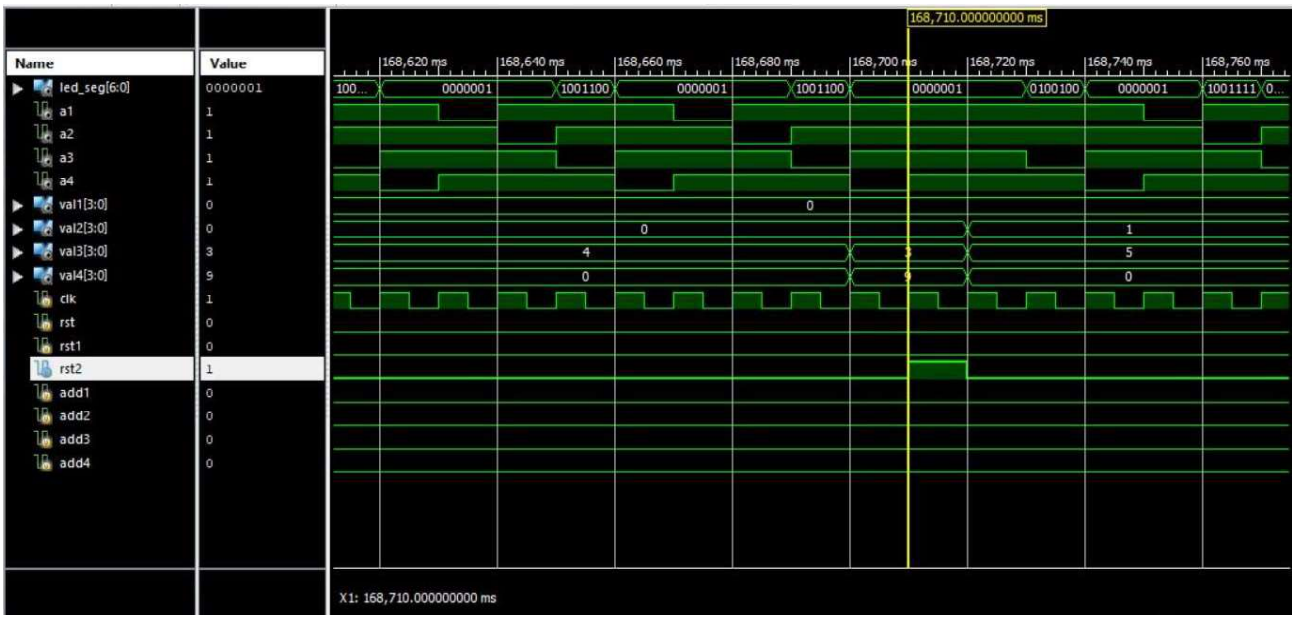
- Test of rst1 set to 1

This case is simulated by letting the remaining time keeps decrementing and then set rst1 to 0 which set the remaining time to 16 secs. As we see when the remaining time get to 529, we set rst1 to 1 which resulted to have the remaining time set to 16 secs.



- Test rst2 set to 1

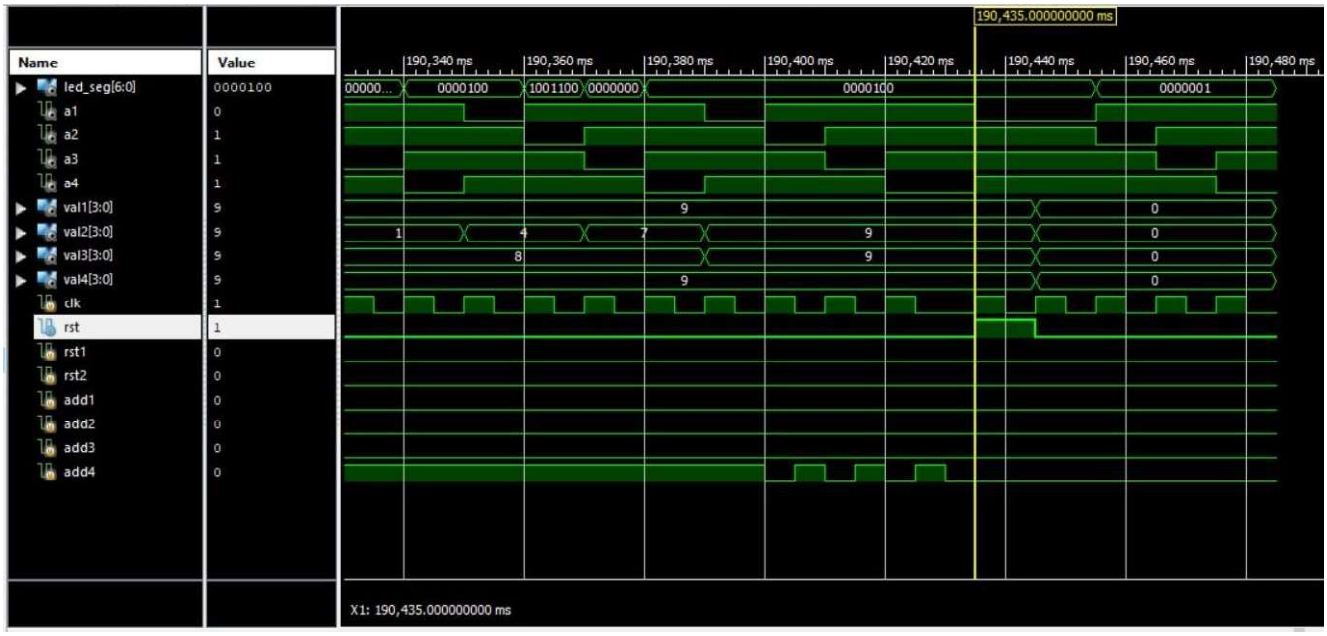
Similarly, this case is simulated by setting rst2 to 1 which will set the remaining time to 150secs. As the remaining time arrived to 39 secs, we set rst2 to 1 which resulted to set the



remaining time to 150sec.

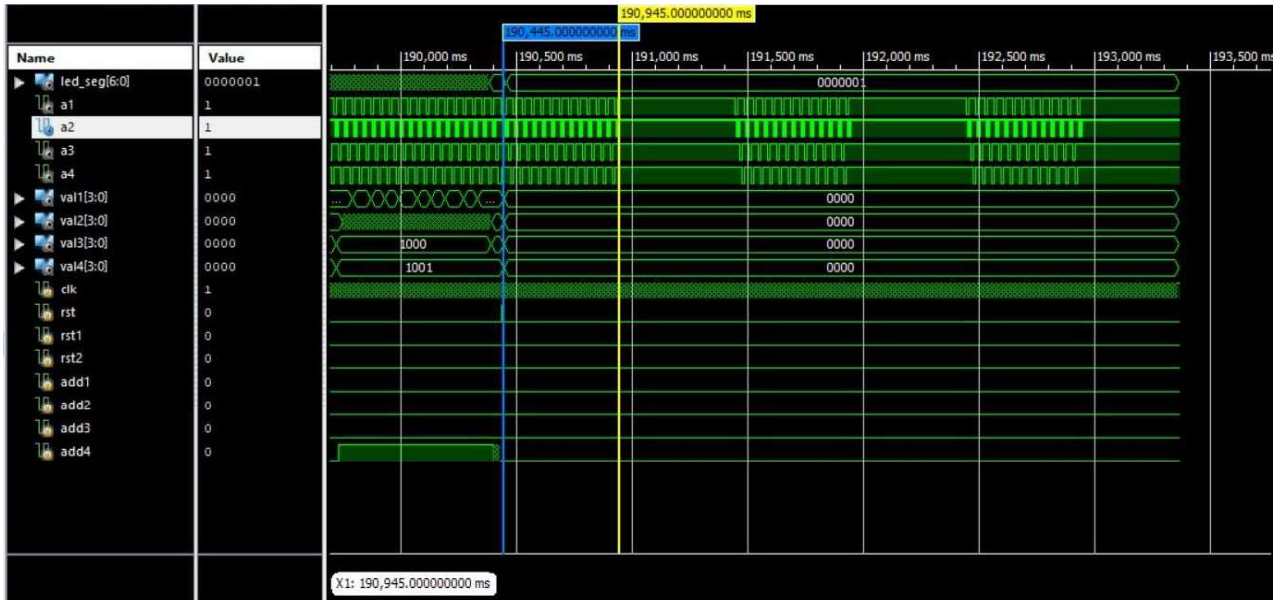
- Test the Maximum limit of the remaining time

In this case, I kept adding seconds to the remaining time until it reached 9999, after that I tried to add 300 three times, but the remaining time is still 9999 and didn't change, and I set rst to 1 to test the hard reset of the remaining time to 0 when it reaches the maximum.



- Testing the flashing behavior if the remaining time is set to 0.

I simulated this by making the clock keep running.



As we see. At 190445 ms when we set rst to 1 to 190945ms the anodes are alternating the 0 showing that the 0000 are displaying in the seven-segment display. This period is taking 500ms which is 0.5 second. From 190945ms to 19445ms all the anodes are set to 1 which show that the 0000 are not displaying. This period is 500ms which is 0.5s. This shows that our seven-segment display behave correctly by displaying for 0.5 sec and off for 0.5 sec which represents period 1 sec and duty cycle 50%.

Conclusion:

In this assignment, I built a parking meter FSM that models the behavior described in project description. This FSM is built in Moore machine. One of the most difficulties that I encountered in this assignment is how to deal with the seven-segment display to display the remaining time, and how test its various outputs including anodes and the led_seg(7-bit vector). This assignment was a good chance to explore FSM in Verilog language and work with seven-segment display.

Synthesize summary:

parking_meter Project Status (11/29/2020 - 19:18:18)			
Project File:	parking_meter.xise	**Parser Errors:**	No Errors
Module Name:	parking_meter	**Implementation State:**	Placed and Routed
Target Device:	xc6slx16-3csg324	• **Errors:**	
Product Version:	ISE 14.7	• **Warnings:**	
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	89	18,224	1%	
Number used as Flip Flops	68			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	21			
Number of Slice LUTs	5,661	9,112	62%	
Number used as logic	5,661	9,112	62%	
Number using O6 output only	3,123			
Number using O5 output only	106			