CS M152A Project 5 Report

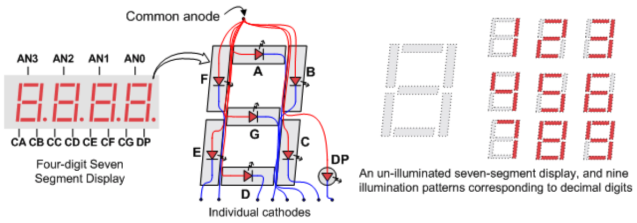
Melody Chen

June 8, 2020

1 Introduction and Requirement

The focus of this lab is for students to learn about how to design finite state machine that matches the specified behavior and to use the Xilinx ISE software to design and test state machines. Finite State Machines (FSMs) are used in many real world systems. An FSM has a finite number of states and can be in one state at a given time. There are two types of FSM machines: Moore Machine and Mealy Machine. A Moore machine's output only depends on which state it is in, but a Mealy machine's output depends on both the current state and input.

For this assignment, we're tasked with designing a FSM that models behavior of 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 that is on the Nexys3 board. The four digits is each composed of seven segments arranged in pattern shown in diagram below with LED light embedded in each segment. For example, to display the digit 0, we will pass in `Cathode[6:0] = 7'b0000001` as a vector and if we want 0 to be our least significant digit, anode 0 should be set to low, while anode 1-3 set to high.



7-segment Display Diagram from Nexys 3 Reference Manual

Our parking meter should have the following characteristics:

1. Input buttons rerpesent different types of coins and 7-segment LED display will display time remaining before meter expires in seconds.
2. When time remaining is zero, 0000 should be flashed at 1Hz with 50% duty cycle. When time remaining is below 180, it should be flashed at 0.5Hz with 50% duty cycle. When time remaining is above 180, there is no need to flash time remaining.
3. Our time remaining for parking meter cannot exceed 9999. If it does, display 9999.

More specifics about behavior of parking meter will be discussed in the following sections.

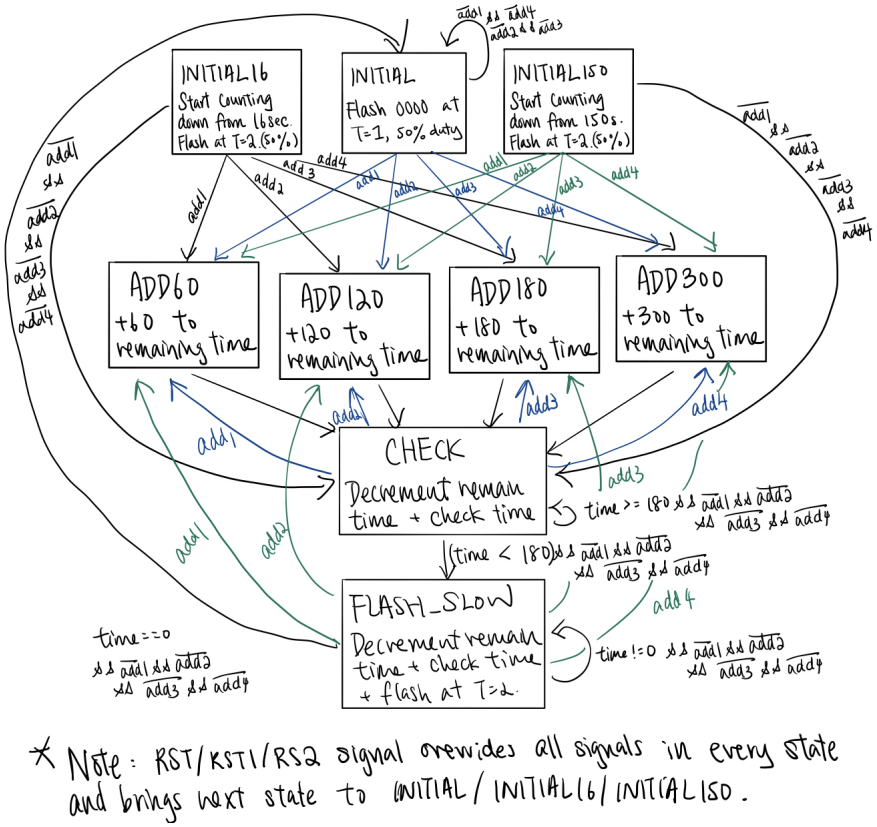
2 Design Description

For the design and implementation of the parking meter FSM, I followed guidance provided in the project specifications and consulted Project 4 Manuscript. I first consider the inputs provided and what my FSM will need to output. We want to output remaining time on parking meter via a seven-segment vector `led_seg` and 4 one bit anode signals `a1, a2, a3, a4` given the following inputs shown in diagram below:

Inputs	Function
add1	add 60 seconds
add2	add 120 seconds
add3	add 180 seconds
add4	add 300 seconds
rst1	reset time to 16 seconds
rst2	reset time to 150 seconds
clk	frequency of 100 Hz
rst	resets to the initial state

Summary of Required Inputs for FSM taken from Project Manuscript

I then designed my FSM with around 8 states that correctly captures the behavior described in the manuscript. My FSM is a Moore machine, meaning that my output depends only on which state I am in, regardless of the inputs. Because it is a Moore machine, I have many states, but that is to ensure for the correct behavior for every possible combination of inputs into my FSM.



Moore FSM Design of my Parking Meter

Descriptions of states and transitions of above FSM:

- Once `rst` signal is 1, we will begin at the INITIAL state, where we flash the current output `0000` at 1Hz with 50% duty cycle. This state represents a parking meter that is out of remaining time. If we receive any coins to add time, we will transition to a different state.
- Two other reset signals are present in our FSM. If either `rst1` or `rst2` is high, we begin at INITIAL16/INITIAL150 state where instead of having no remaining time, we will start with either 16 seconds or 150 seconds remaining on our parking meter. Remaining time will be flashed at 0.5Hz as remaining time is non-zero and is below 180 seconds. Beginning in this state, time remaining should decrease by 1 every second.

- From the three INITIAL states, if either `add1, add2, add3, add4` signal is 1, we transition to the corresponding ADD60/ADD120/ADD180/ADD300 state. The ADD states are where depending on which state, we add the specified number of seconds to our parking meter. Once this state ends, the time remaining on our meter should be updated. Note that during this state, time remaining is continued to being decremented by 1 per sec to simulate a real parking meter. Time remaining is also continued to be displayed in this state based on how it was displayed in its previous state, and added time will only show up after end of this state.

- From the INITIAL16/INITIAL150 states, if no add signals are asserted, we transition directly to the CHECK state. CHECK state checks whether time remaining is 0 or less than 180 in order for time remaining to be displayed in the appropriate manner. As before, time remaining is continued being decremented by 1 per sec.

- From the four ADD states, once the specified time is added, we directly transition to the CHECK state. This is to make sure now that our time remaining increased, we are displaying the remaining time in the appropriate manner matching the new value.

- In the CHECK state, we continue to decrement remaining time and display current time. Manner time is being displayed in this state is based on how it was being displayed in its previous state. Manner displayed will be updated after CHECK state. We use this state to check whether our new time remaining is greater or less than 180. If it is greater, we stay in this state as to continue checking if after the next second our time will be less than 180. If it is less than 180, we transition to FLASH_SLOW state. If we receive add signal in this state, we directly transition to the corresponding ADD state.

- In FLASH_SLOW state, time is continue being decremented and it is displayed at 0.5Hz with 50% duty cycle as time is below 180 seconds. If we receive add signal in this state, we directly transition to the corresponding ADD state to make sure time is increased. Otherwise, if time left is zero, we transition back to INITIAL for 0000 to be flashed at 1Hz. If time left is greater than zero and no add signal received, we stay in this state to continue decrementing and displaying time.

- Note 1: If `rst = 1` or `rst1 = 1` or `rst2 = 1` at any point in our state diagram, it overrides other signal and arrows, and we directly return to the corresponding INITIAL/INITIAL16/INITIAL150 state.

- Note 2: Time remaining is **always** being decremented by 1 every second if it is not zero(i.e. we're not in the INITIAL state). Time remaining is **always** being displayed, but there could be a 1-2 cycle delay as it takes time for our FSM to process input signal and update the outputs.

To implement the above FSM in Verilog, I created the `parking_meter` top module that takes in input listed in the table above and outputs: 4 anode signals, vector `led_seg`, and time remaining in 4 BCD digits. Within my module, I initialized parameters representing all the states I have with a unique 4 bit binary number, two 4-bit registers `current_state` and `next_state`, a `time_left` register to keep track of number of seconds remaining in decimal, and various internal signals and counters for keeping track of time. I have 4 main always block each with a specific task:

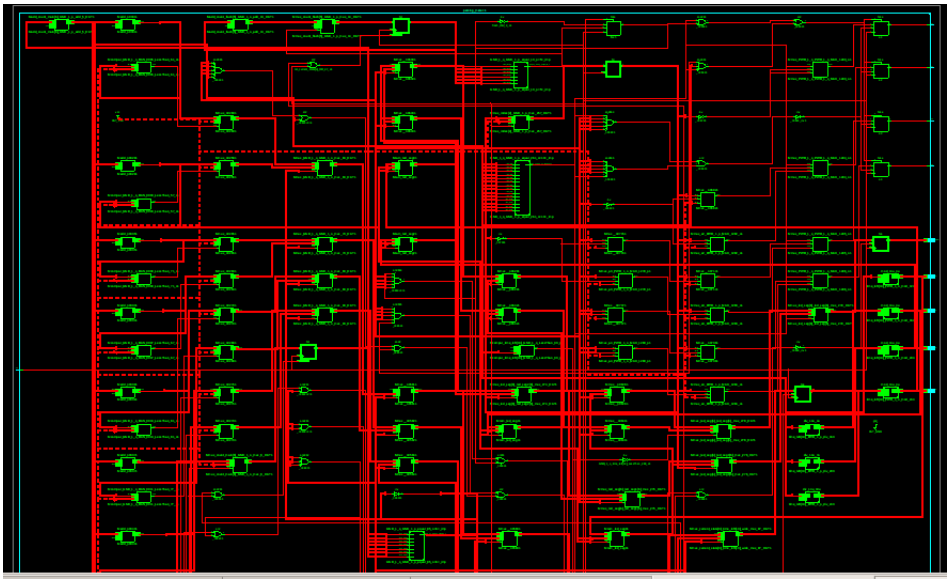
1. Sequential always block to update `current_state` to `next_state` at every `posedge` of the clock or update `current_state` to INITIAL/16/150 state when either `rst, rst1, rst2 = 1`.

2. Combinational always block to decide what `next_state` should be based on the current state and current inputs.

3. Sequential always block to determine how much `time_left` based on the current state and internal counters that exploit the input clock to keep track of time.

4. Sequential always block to determine the 4 1-bit anode signals and vector `led_seg`, which both control what is shown on our 7-segment LED display, based on our current state, internal counters, and our `time_left` register.

Aside from the blocks mentioned above, I also have smaller combinational logic blocks to determine value of `led_seg` based on time remaining. I also have smaller sequential logic blocks that control my internal counters that produces signal to inform other blocks when to decrement `time_left` register(i.e. when 1 second has passed) and signal for the appropriate flashing of our display. The logic and design of these counters are heavily based on previous projects such as 4-bit counter and clock divider.



RTL Schematic of Parking Meter FSM

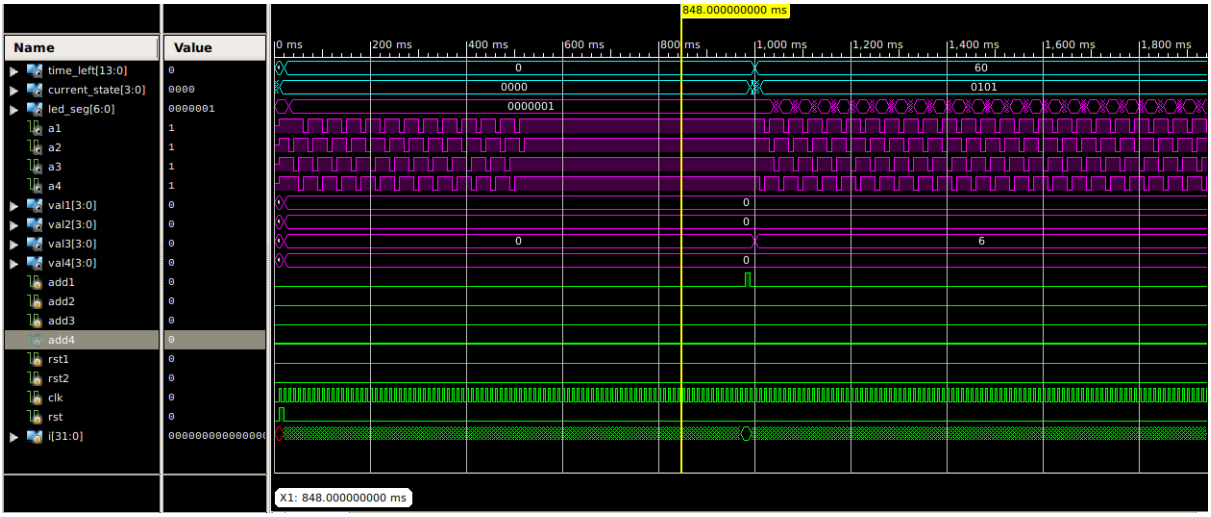
From first look, the RTL schematic above looks a bit complicated and filled with registers, muxes, and counters. This makes sense as our FSM is fairly complicated having to keep track of internal counters, over 4 always blocks with layers of conditions in addition to combinational logic for determining next state and to produce signals for 7-segment display. My Verilog code results in this schematic being generated(layers of mux, gates for combinational logic, and many registers) as the logic for determining what the remaining time is based on input signals on top of how to display the remaining time based on current time needs to be implemented through series of mux logic, registers/D-Flipflops are essential for our internal counters, and gates for our combinational logic.

3 Simulation Documentation

To test whether the FSM I implemented works as the manuscript described, I created my testbench that tests not only all the possible types of inputs, but all of the possible special cases, such as adding signal received at the exact time time is decremented. I provided a 100Hz clock input clock in my testbench as specified by the manuscript. To simulate behavior of someone interacting with the parking meter, in my testbench, I at each positive edge of the input clock, I change a particular input to observe how my FSM will deal with the change in input or I don't change any input to observe my FSM counting down. Below are all the different cases I tested and their respective waveform.
Note that in all the simulation waveforms shown below in addition to the outputs mention in the previous sections, I show an additional two outputs `time_left`, decimal representation of how much time remains, and `current_state`, binary representation of our current state, to illustrate how my FSM operate and for ease of readability. In addition, signals in waveforms are color coded: light blue indicates the two outputs I added, pink indicates output specified by manuscript, and green indicates inputs provided to FSM.

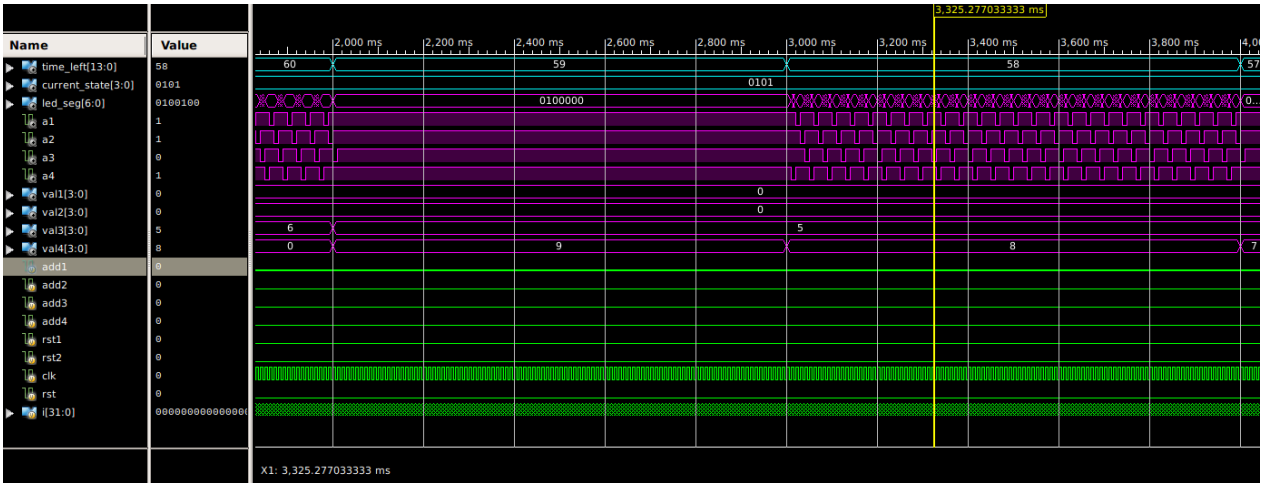
1. **Transition from No Time Remaining to 60 sec with `add1` signal**
   This case represents a parking meter that is not being used and has no time remaining, and then someone parks a car and added 60 seconds to the meter through the `add1` signal. How I tested this case was to first set `rst` to high for one clock cycle to put parking meter at INITIAL(0000) state.

Then, after almost one second, I set the `add1` signal to high for one clock cycle to simulate someone putting coins in to the meter and then I continued flipping clock signal to observe its behavior. We can observe from the waveform below that while there is no time remaining in our meter, outputs `val1-4` remain zero which is the correct behavior. The `led_seg` vector has value `0000001` which is correct vector for outputting 0 on a 7-segment display. We also observe outputs `a1-4` rotating which of the four signal is low in order to display 0 on all four 7-segment displays. Right after `add1` signal is high for 1 cycle, we immediately observe `time_left` increased from 0 to 60 seconds as expected. After 1 sec, we see that value of `led_seg` changes and alternates between vector for digit 0 and 6 as expected.



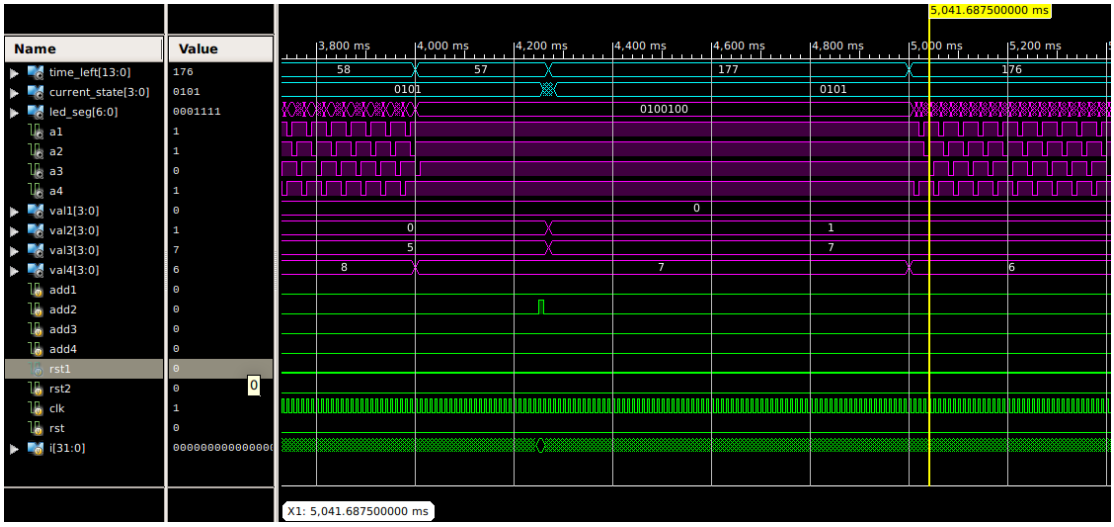
Simulation Waveform for Case 1

2. **Correct Flashing of Time Remaining under 180 sec**
This case represents a time meter that has less than 180 seconds remaining where time is decreasing appropriately and flashed at period of 2 seconds. To test this case, I simply continue to flip the clock signal for a 100Hz clock for a specific number of cycles. From the diagram below, in addition to `time_left` being decremented exactly after every 1 second and the correct `val-4` values, we also observe only when time remaining is an odd number, all `a1-3` remain high meaning nothing is shown on the displays. This is correct behavior as to have a period of 2 seconds with 50% duty cycle, we only want there to be time remaining shown for 1 second. The specs specified only the even times to be shown.



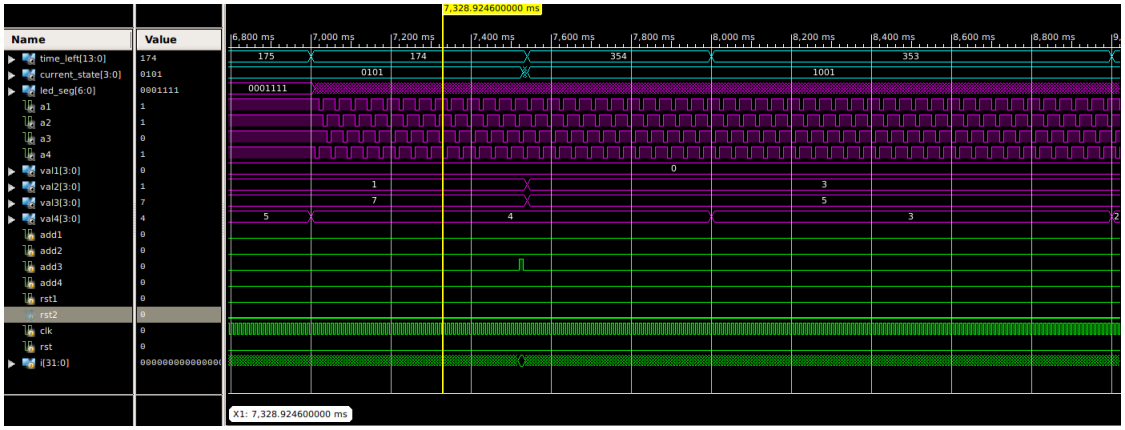
Simulation Waveform for Case 2

3. **Adding time with `add2` signal while time remaining is under 180 sec**
   This case represents someone adding 120 seconds to a parking meter that has current remaining time less than 180 seconds. To simulate this behavior, I set the `add2` signal to high for one clock cycle while time remaining is displaying 57 seconds. From the diagram below we can see that at 4000ms we go from 58 sec to 57 sec remaining. At around 4200ms, our FSM detects the `add2` signal and time remaining becomes 177 sec. Since `add2` signal is detected around 200ms after our FSM has been showing 57sec, we expect 177 sec to be displayed not for a full 1 second, but only for the remaining 800ms. This is because we don't want our parking meter to "gain extra time" just because coin is inserted when 56.8 seconds remain instead of 57 seconds. This matches the behavior of our FSM as at exactly 5000ms, 1000ms after time remaining decremented to 57 seconds, 177 sec is decremented to 176 seconds. No extra time is gained.



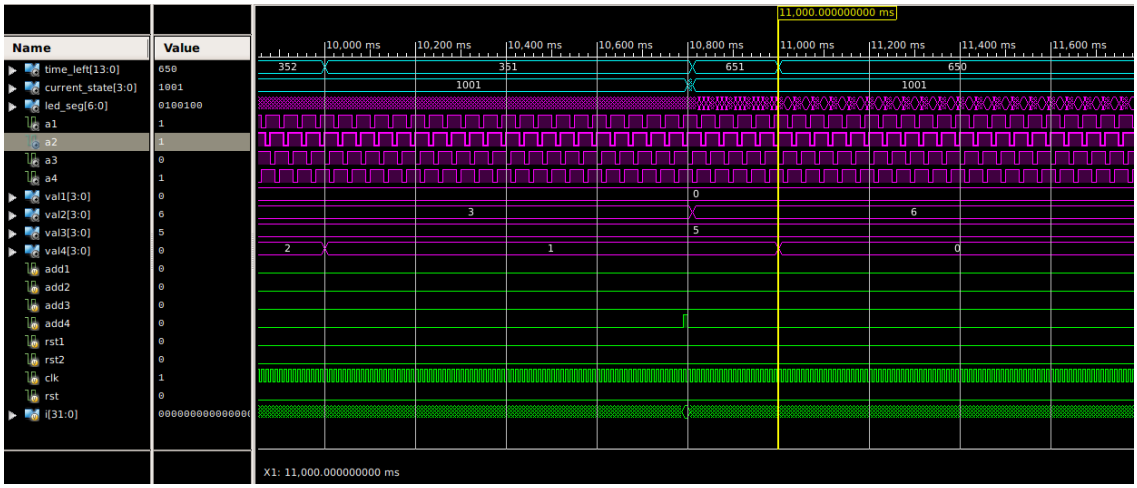
Simulation Waveform for Case 3

4. **Adding time with `add3` signal with resulting time exceeding 180 sec**
   This case represents when parking meter has time remaining below 180 seconds and someone put in coins to add 180 seconds of time. To simulate this behavior, I set the `add3` signal to be high for one clock cycle while time remaining is displaying 174 seconds. As mentioned in the last case, since signal is detected midway through display of 174 sec, we only want 354 sec to be shown for the remaining half of the second(and decrement right after) instead of for a full second. We can observe this correct behavior in our waveform below where 180 seconds is correctly added and at exact 8000ms, 354 is decremented to 353. We can also observe from our `a1-4` signals that our 7-segment display is no longer being flashed at 0.5Hz, but continued to be displayed without being flashed. This matches behavior in the manuscript.



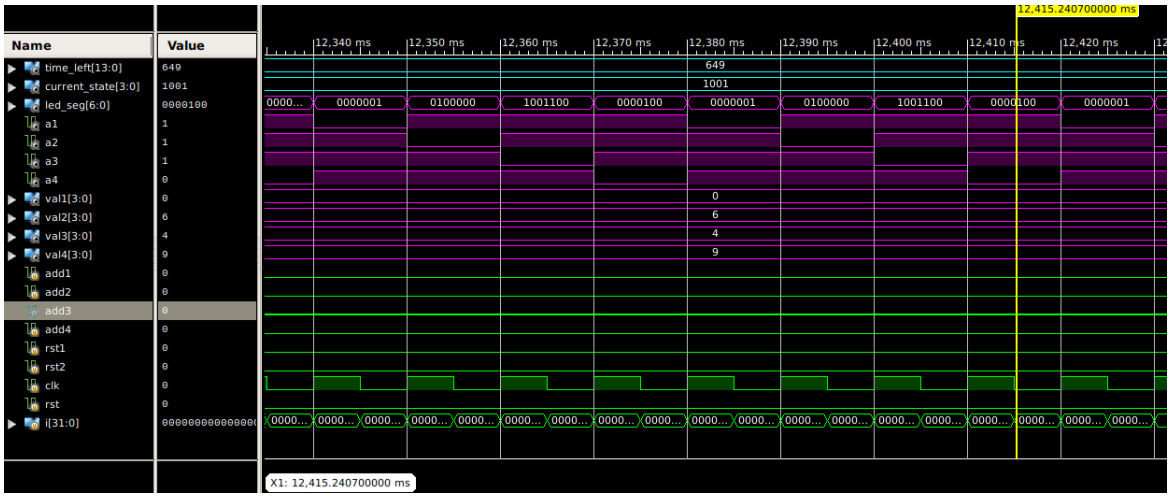
Simulation Waveform for Case 4

5. **Adding time with `add4` signal while time remaining exceed 180 sec**
   This case represents when parking meter has time remaining that is above 180 seconds and someone put in coins to add 300 seconds of time. To simulate this behavior, I set the `add4` signal to be high for one clock cycle while time remaining is displaying 351 seconds. As mentioned in the last case, since add signal is detected around 800ms after time remaining began displaying 351, we only want 651 sec to be shown for 200ms instead of for a full second. We can observe this correct behavior in our waveform below where 300 seconds is correctly added and at exact 11000ms, 651 is decremented to 650. We can also observe from our `a1-4` signals that our 7-segment display is not being flashed. This matches behavior in the manuscript.



Simulation Waveform for Case 5

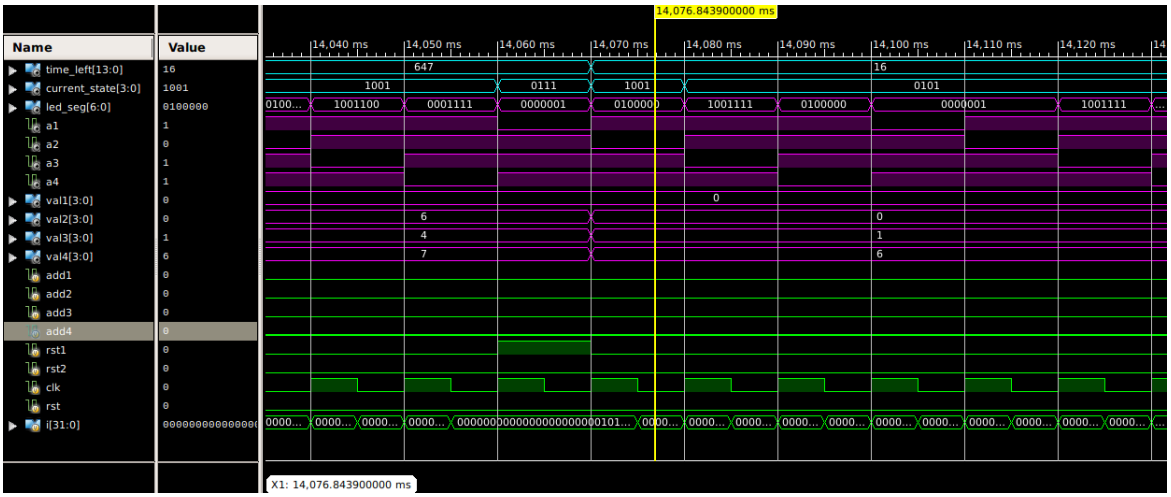
6. **Correct 7-Segment Signal for Multiplexed Display with Large Time Remaining**
   In this case, we look closer into how we're producing signals for the multiplexed display and whether they will allow our 4 7-segment displays to correctly show the remaining time on our parking meter. From the waveform diagram below, we observe that current time remaining is 649 seconds. At $t = 12,340s$, signal `a1` is low for 1 cycle while signals `a2-4` are high. This means that whatever is in the `led_seg` vector is what will be displayed for the left-most digit on our display. At $t = 12,340s$, `led_seg` vector has values corresponding to showing a 0 which is correct as left-most digit of 649 is 0. At $t = 12,3450s$ the next clock cycle, only anode signal that is low is `a2`, so we will display the number 6 on second left-digit of our display, which is correct. In the same manner, digit 4 will be displayed for the 3rd left-most digit and digit 9 will be displayed for the right most digit in the following cycle. Thus, our parking meter FSM correctly displays time remaining on our 7-segment displays.



Simulation Waveform for Case 6

7. **Closer look at `rst1` signal when time remaining is large**

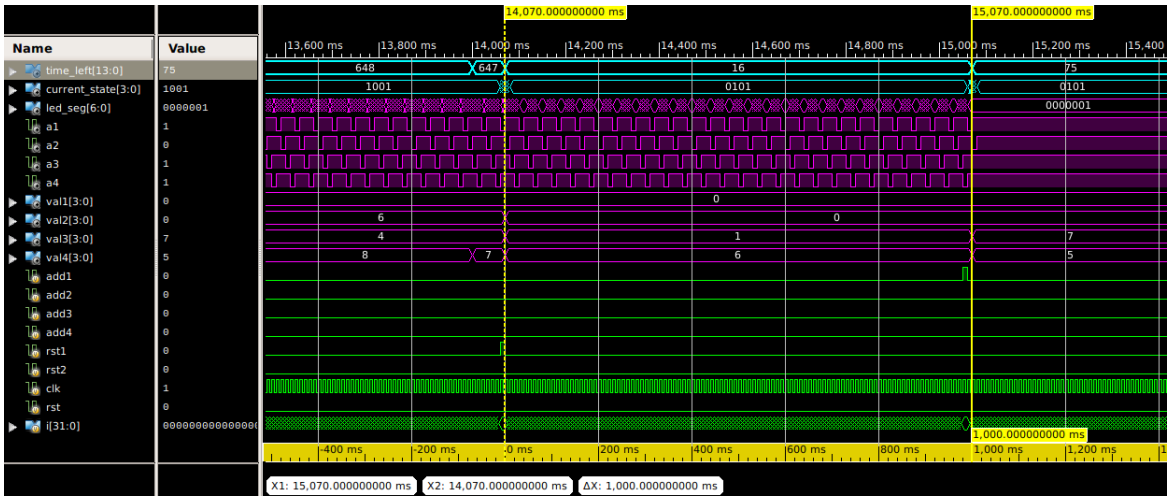
   This case represents situation where our parking meter is resetted to 16 seconds remaining. As you can observe from waveform below, `rst1` is high for one cycle, and in the following cycle our time remaining is updated to 16. However, looking at the `led_seg` value, in the cycle after reset signal is asserted, there is one cycle where we display 6 in our second left-most digit. But after that cycle, values corresponding to 16 sec is displayed. This is the correct behavior as it takes one state for our FSM to process the reset signal and compute through combination logic what should be passed to `led_seg`. This will not affect customers of our parking meter as this occurs only for one brief cycle. We can also observe how for the rest of the cycles with time remaining being 16, the outputs are all valid.



Simulation Waveform for Case 7

8. **Correct timing after reset signal while time remain decreases**

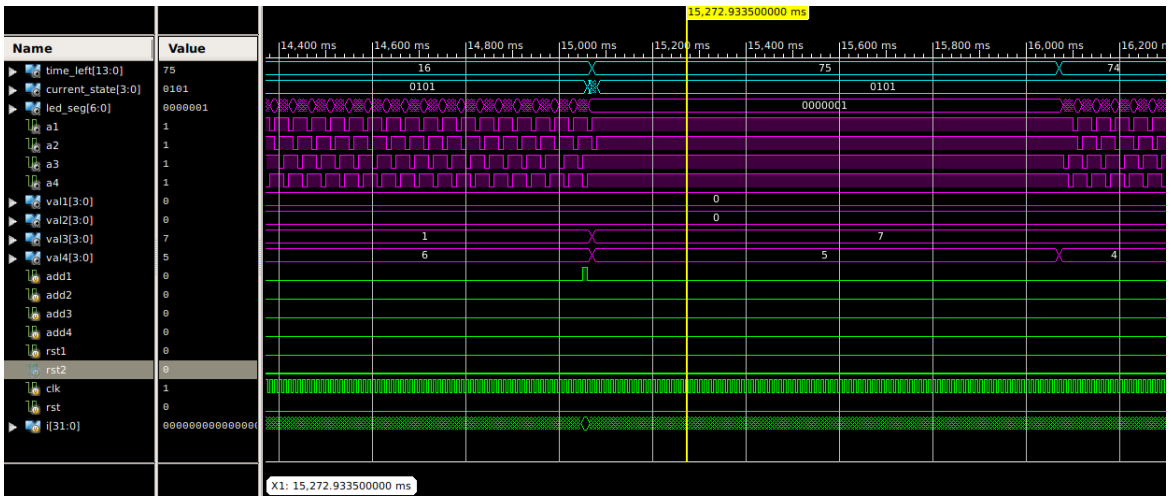
   This case examines whether after a `rst1/2` signal, our parking meter correctly shows the time remaining from either 16/150 seconds for 1 full second. From the waveform below, we can observe that `rst1` signal is detected at $t = 14070s$. So the value 16 should be displayed on our parking meter until $t = 15070s$. This correctly matches what is shown in diagram below.



Simulation Waveform for Case 8

9. **Add signal detected right as even time is decremented to odd time**

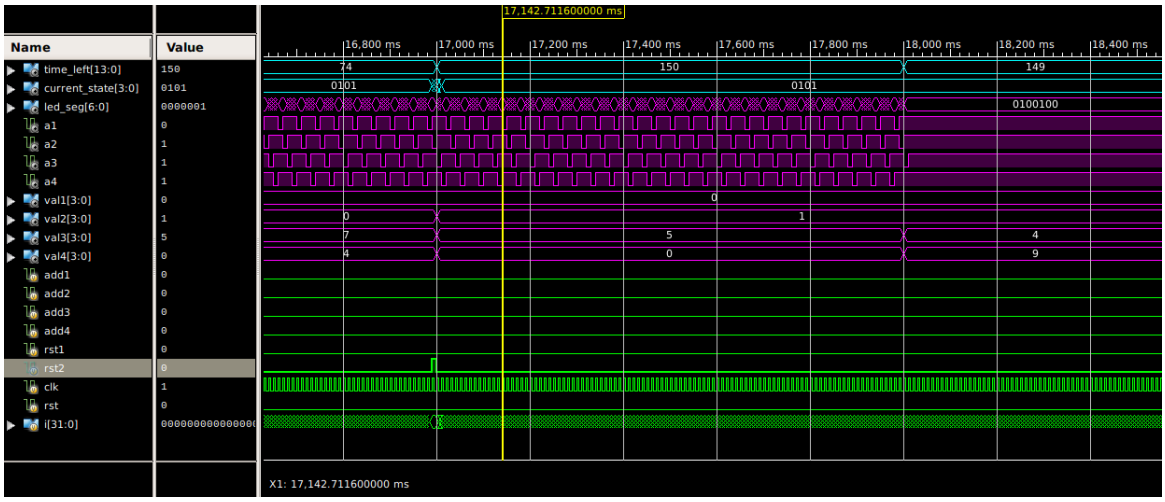
This case is an edge case where right when time remaining is supposed to go down by 1, we add time into our parking meter. To simulate this behavior, I set `add1` signal to be high for one cycle when 16 second is almost decremented to 15 seconds. Without this signal, in the next cycle we were supposed to show 15, but now we should show 75 seconds remaining. 76 seconds remaining is an incorrect output as 60 seconds is added when 16 seconds becomes 15 seconds. From waveform below, we can see that our FSM correctly displays 75 seconds remaining after the `add1` signal.



Simulation Waveform for Case 9

10. **Correct behavior of `rst2` signal**

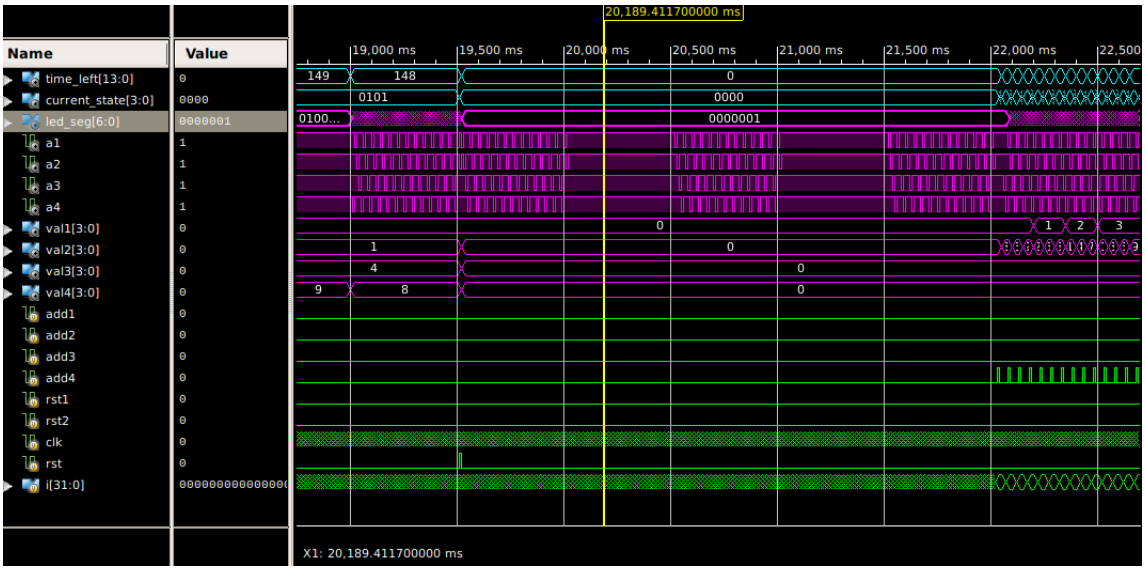
In this case, we model the behavior of our parking meter being reset to have 150 seconds remaining. I tested this by setting the `rst2` signal to high for one cycle. From the diagram below, we can observe that FSM is correctly reset to 150 seconds at $t = 17s$, at displays 150 until $t = 17s$ which is the expected behavior.



Simulation Waveform for Case 10

11. **Correct flashing of no time remaining**

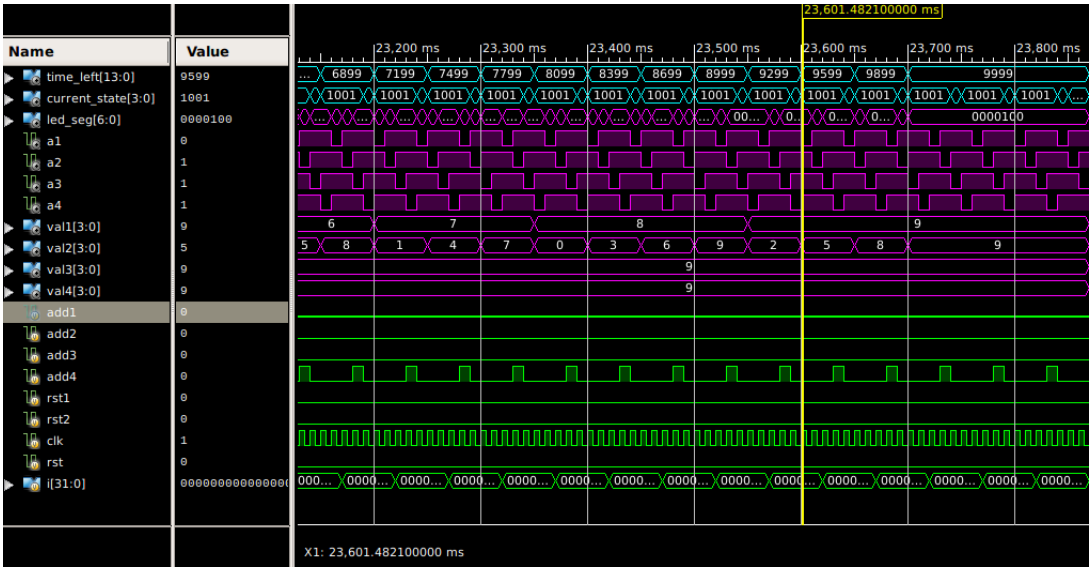
In this case, we test whether our FSM correctly flashes 0000 at 1Hz when there is no time remaining. I tested this by simply resetting the FSM and observing its behavior as clock signal is being flipped. From the diagram below, we can observe that from $t = 19.5s$ to $t = 20s$, we're showing 0000 on our display, but from around $t = 20s$ to $t = 20.5s$, we're not showing anything on our display. This pattern continues which indicates that we're correctly displaying 0000 at 1Hz with 50% duty cycle.



Simulation Waveform for Case 11

12. **Incrementing time remaining beyond 9999**

In this case, we test whether our FSM correctly behaves when we attempt to increment time beyond 9999 seconds. The expected behavior is to remain at 9999 seconds when more time is added to 9999 seconds. To test this case, I used the `add4` signal to continue to add 300 seconds to our time remaining until we reached 9999 seconds. From the diagram below, we can observe that once we attempt to add 300 seconds to 9899 seconds, we get output of 9999 seconds. This matches the behavior specified in the manuscript.



Simulation Waveform for Case 12

One bug I found during simulation for this project was that I accidentally flipped my anode signals. To have the `led_seg` display show on the left-most bit, `a1` should be set to low while rest of anodes set to high. But I had `a1` set to high while rest of anodes set to low. Because of the simulations, I was able to quickly find this major bug and fix my FSM accordingly.

4 Synthesis and Implementation Report

Sections of Synthesis and Implementation Report is attached at end of report. From the 'Design Summary' part of the synthesis report, we can conclude the different components that my implementation of the Parking Meter FSM uses. For example, we can see that we indeed do use a lot of registers from the synthesis report and we can also see the different clock signals needed for our module. From implementation report, we can see that there is no errors when trying to implement our module, but there are a few warnings due to truncation of numbers into registers. The warnings can be ignored as in my code I made sure that truncation will definitely not occur for those registers through conditions.

5 Conclusion

In this project, I designed and implemented the Parking Meter FSM according to the behavior in the project manuscript and how a real world Parking Meter will work. I designed the Parking Meter based on a Moore Machine, that has similar structure to the Vending Machine FSM implemented to in the last project. To correctly implement this, I had to first draw the FSM diagram to allow me to clearly see what output should be in each state and how to transition to a different state. One of the major difficulty I encountered in this assignment was understanding how to use the 7-segment display to display my remaining time correctly. Another major difficulty was to implement the various internal counters needed for different flashing frequencies. I was able to figure out what to do from help from my TA and from the work I did with counters and clock dividers in the previous projects.

6 Reports

6.1 Synthesis Report

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : parking_meter.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 1806
#      GND                         : 1
#      INV                         : 56
#      LUT1                        : 61
#      LUT2                        : 41
#      LUT3                        : 58
#      LUT4                        : 93
#      LUT5                        : 186
#      LUT6                        : 707
#      MUXCY                       : 284
#      MUXF7                       : 15
#      VCC                         : 1
#      XORCY                       : 303
# FlipFlops/Latches                : 63
#      FD                          : 4
#      FDE                         : 37
#      FDR                         : 17
#      FDSE                        : 5
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 52
#      IBUF                        : 7
#      OBUF                        : 45

Device utilization summary:
---------------------------

Selected Device : 6slx16csg324-3


Slice Logic Utilization:
 Number of Slice Registers:              63  out of  18224     0%
 Number of Slice LUTs:                 1202  out of   9112    13%
    Number used as Logic:              1202  out of   9112    13%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   1223
   Number with an unused Flip Flop:    1160  out of   1223    94%
   Number with an unused LUT:            21  out of   1223     1%
   Number of fully used LUT-FF pairs:    42  out of   1223     3%
   Number of unique control sets:         8

IO Utilization:
 Number of IOs:                          53
 Number of bonded IOBs:                  53  out of    232    22%
```

```
Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                1  out of     16     6%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------


=========================================================================
Timing Report

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
clk                                | BUFGP                  | 63    |
-----------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 20.233ns (Maximum Frequency: 49.424MHz)
   Minimum input arrival time before clock: 4.097ns
   Maximum output required time after clock: 21.435ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 20.233ns (frequency: 49.424MHz)
  Total number of paths / destination ports: 27095554062 / 121
-------------------------------------------------------------------------
Delay:               20.233ns (Levels of Logic = 23)
  Source:            time_left_12_1 (FF)
  Destination:       led_seg_6 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: time_left_12_1 to led_seg_6
                              Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
```

```
    FDE:C->Q              5   0.447   1.059  time_left_12_1 (time_left_12_1)
    LUT5:I0->O           17   0.203   1.256  time_left[13]_PWR_1_o_div_163_OUT<5>111 (time_left[13]_PWR_1_o_div_16
    LUT5:I2->O            9   0.205   0.829  time_left[13]_PWR_1_o_div_163_OUT<6>11 (time_left[13]_PWR_1_o_div_163
    MUXCY:DI->O           1   0.145   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_17_OUT_Madd_cy<6
    MUXCY:CI->O           1   0.019   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_17_OUT_Madd_cy<7
    XORCY:CI->O           8   0.180   0.803  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_17_OUT_Madd_xor<
    LUT5:I4->O            4   0.205   0.683  time_left[13]_PWR_1_o_mod_164/Mmux_a[8]_a[13]_MUX_288_o11 (time_left[
    MUXCY:DI->O           1   0.145   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_19_OUT_Madd_cy<8
    MUXCY:CI->O           1   0.019   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_19_OUT_Madd_cy<9
    XORCY:CI->O           9   0.180   0.830  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_19_OUT_Madd_xor<
    LUT6:I5->O           18   0.205   1.050  time_left[13]_PWR_1_o_mod_164/Mmux_a[10]_a[13]_MUX_300_o11 (time_left
    LUT6:I5->O            2   0.205   0.617  time_left[13]_PWR_1_o_mod_164/BUS_0011_INV_179_o2_SW4_SW0_SW0 (N816)
    LUT6:I5->O           16   0.205   1.005  time_left[13]_PWR_1_o_mod_164/Mmux_a[5]_a[13]_MUX_319_o11 (time_left[
    LUT6:I5->O            1   0.205   0.684  time_left[13]_PWR_1_o_mod_164/BUS_0012_INV_194_o2_SW0_SW4 (N917)
    LUT6:I4->O            6   0.203   1.109  time_left[13]_PWR_1_o_mod_164/Mmux_a[4]_a[13]_MUX_334_o11 (time_left[
    LUT6:I0->O           18   0.203   1.050  time_left[13]_PWR_1_o_mod_164/BUS_0013_INV_209_o2_SW0 (N62)
    LUT6:I5->O            3   0.205   0.898  time_left[13]_PWR_1_o_mod_164/Mmux_a[8]_a[13]_MUX_344_o11 (time_left[
    LUT6:I2->O            3   0.203   0.651  time_left[13]_PWR_1_o_mod_164/BUS_0014_INV_224_o21 (time_left[13]_PWR
    LUT6:I5->O            2   0.205   0.721  time_left[13]_PWR_1_o_mod_164/Mmux_a[0]_a[13]_MUX_366_o131 (time_left
    LUT6:I4->O            3   0.203   0.651  time_left[13]_PWR_1_o_mod_164/BUS_0015_INV_239_o23 (time_left[13]_PWR
    LUT6:I5->O           11   0.205   1.111  time_left[13]_PWR_1_o_mod_164/Mmux_o21 (val3_1_OBUF)
    LUT6:I3->O            1   0.205   0.000  Mmux_led_seg[6]_led_seg[6]_mux_279_OUT74_SW0_SW0_SW0_SW1_G (N1488)
    MUXF7:I1->O           1   0.140   0.580  Mmux_led_seg[6]_led_seg[6]_mux_279_OUT74_SW0_SW0_SW0_SW1 (N1124)
    LUT6:I5->O            1   0.205   0.000  Mmux_led_seg[6]_led_seg[6]_mux_279_OUT75 (led_seg[6]_led_seg[6]_mux_2
    FDE:D                     0.102          led_seg_6
    ----------------------------------------
    Total                    20.233ns (4.647ns logic, 15.586ns route)
                                      (23.0% logic, 77.0% route)

==========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'
  Total number of paths / destination ports: 49 / 33
--------------------------------------------------------------------------
Offset:              4.097ns (Levels of Logic = 4)
  Source:            add3 (PAD)
  Destination:       current_state_FSM_FFd4 (FF)
  Destination Clock: clk rising

  Data Path: add3 to current_state_FSM_FFd4
                              Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             3   1.222   0.755  add3_IBUF (add3_IBUF)
     LUT4:I2->O            1   0.203   0.580  current_state_FSM_FFd4-In1 (current_state_FSM_FFd4-In1)
     LUT5:I4->O            1   0.205   0.827  current_state_FSM_FFd4-In2 (current_state_FSM_FFd4-In2)
     LUT6:I2->O            1   0.203   0.000  current_state_FSM_FFd4-In3 (current_state_FSM_FFd4-In)
     FDR:D                     0.102          current_state_FSM_FFd4
    ----------------------------------------
    Total                     4.097ns (1.935ns logic, 2.162ns route)
                                      (47.2% logic, 52.8% route)

==========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 3038580636 / 45
--------------------------------------------------------------------------
Offset:              21.435ns (Levels of Logic = 23)
  Source:            time_left_12_1 (FF)
  Destination:       val3<3> (PAD)
  Source Clock:      clk rising
```

```
  Data Path: time_left_12_1 to val3<3>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDE:C->Q              5   0.447   1.059  time_left_12_1 (time_left_12_1)
     LUT5:I0->O           17   0.203   1.256  time_left[13]_PWR_1_o_div_163_OUT<5>111 (time_left[13]_PWR_1_o_div_16
     LUT5:I2->O            9   0.205   0.829  time_left[13]_PWR_1_o_div_163_OUT<6>11 (time_left[13]_PWR_1_o_div_163
     MUXCY:DI->O           1   0.145   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_17_OUT_Madd_cy<6
     MUXCY:CI->O           1   0.019   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_17_OUT_Madd_cy<7
     XORCY:CI->O           8   0.180   0.803  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_17_OUT_Madd_xor<
     LUT5:I4->O            4   0.205   0.683  time_left[13]_PWR_1_o_mod_164/Mmux_a[8]_a[13]_MUX_288_o11 (time_left[
     MUXCY:DI->O           1   0.145   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_19_OUT_Madd_cy<8
     MUXCY:CI->O           1   0.019   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_19_OUT_Madd_cy<9
     XORCY:CI->O           9   0.180   0.830  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_19_OUT_Madd_xor<
     LUT6:I5->O           18   0.205   1.050  time_left[13]_PWR_1_o_mod_164/Mmux_a[10]_a[13]_MUX_300_o11 (time_left
     LUT6:I5->O            2   0.205   0.617  time_left[13]_PWR_1_o_mod_164/BUS_0011_INV_179_o2_SW4_SW0_SW0 (N816)
     LUT6:I5->O           16   0.205   1.005  time_left[13]_PWR_1_o_mod_164/Mmux_a[5]_a[13]_MUX_319_o11 (time_left[
     LUT6:I5->O            1   0.205   0.684  time_left[13]_PWR_1_o_mod_164/BUS_0012_INV_194_o2_SW0_SW4 (N917)
     LUT6:I4->O            6   0.203   1.109  time_left[13]_PWR_1_o_mod_164/Mmux_a[4]_a[13]_MUX_334_o11 (time_left[
     LUT6:I0->O           18   0.203   1.050  time_left[13]_PWR_1_o_mod_164/BUS_0013_INV_209_o2_SW0 (N62)
     LUT6:I5->O            3   0.205   0.650  time_left[13]_PWR_1_o_mod_164/Mmux_a[0]_a[13]_MUX_352_o121 (time_left
     MUXCY:DI->O           1   0.145   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_27_OUT_Madd_cy<2
     MUXCY:CI->O           1   0.019   0.000  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_27_OUT_Madd_cy<3
     XORCY:CI->O           2   0.180   0.961  time_left[13]_PWR_1_o_mod_164/Madd_a[13]_GND_2_o_add_27_OUT_Madd_xor<
     LUT5:I0->O            1   0.203   0.580  time_left[13]_PWR_1_o_mod_164/BUS_0015_INV_239_o26_SW0_SW0 (N376)
     LUT6:I5->O            1   0.205   0.684  time_left[13]_PWR_1_o_mod_164/Mmux_o41_SW0 (N930)
     LUT6:I4->O           11   0.203   0.882  time_left[13]_PWR_1_o_mod_164/Mmux_o41 (val3_3_OBUF)
     OBUF:I->O                 2.571          val3_3_OBUF (val3<3>)
    ----------------------------------------
    Total                     21.435ns (6.705ns logic, 14.730ns route)
                                       (31.3% logic, 68.7% route)

==========================================================================

Cross Clock Domains Report:
--------------------------

Clock to Setup on destination clock clk
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |   20.233|         |         |         |
---------------+---------+---------+---------+---------+

==========================================================================


Total REAL time to Xst completion: 27.00 secs
Total CPU time to Xst completion: 27.30 secs

-->


Total memory usage is 403632 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    9 (   0 filtered)
```

```
Number of infos    :    4 (   0 filtered)
```

6.2 Implementation Report

```
Release 14.7 Map P.20131013 (lin64)
Xilinx Mapping Report File for Design 'parking_meter'

Design Information
------------------
Command Line   : map -intstyle ise -p xc6slx16-csg324-3 -w -logic_opt off -ol
high -t 1 -xt 0 -register_duplication off -r 4 -global_opt off -mt off -ir off
-pr off -lc off -power off -o parking_meter_map.ncd parking_meter.ngd
parking_meter.pcf
Target Device  : xc6slx16
Target Package : csg324
Target Speed   : -3
Mapper Version : spartan6 -- $Revision: 1.55 $
Mapped Date    : Sun Jun  7 22:53:36 2020

Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                    66 out of  18,224    1%
    Number used as Flip Flops:                  63
    Number used as Latches:                      0
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                3
  Number of Slice LUTs:                     1,185 out of   9,112   13%
    Number used as logic:                   1,183 out of   9,112   12%
      Number using O6 output only:          1,014
      Number using O5 output only:             60
      Number using O5 and O6:                 109
      Number used as ROM:                       0
    Number used as Memory:                      0 out of   2,176    0%
    Number used exclusively as route-thrus:     2
      Number with same-slice register load:     1
      Number with same-slice carry load:        1
      Number with other load:                   0

Slice Logic Distribution:
  Number of occupied Slices:                  390 out of   2,278   17%
  Number of MUXCYs used:                      340 out of   4,556    7%
  Number of LUT Flip Flop pairs used:       1,199
    Number with an unused Flip Flop:        1,139 out of   1,199   94%
    Number with an unused LUT:                 14 out of   1,199    1%
    Number of fully used LUT-FF pairs:         46 out of   1,199    3%
    Number of unique control sets:              8
    Number of slice register sites lost
      to control set restrictions:             25 out of  18,224    1%

  A LUT Flip Flop pair for this architecture represents one LUT paired with
  one Flip Flop within a slice.  A control set is a unique combination of
  clock, reset, set, and enable signals for a registered element.
  The Slice Logic Distribution report is not meaningful if the design is
  over-mapped for a non-slice resource or if Placement fails.

IO Utilization:
  Number of bonded IOBs:                       53 out of     232   22%
```

```
Specific Feature Utilization:
  Number of RAMB16BWERs:                        0 out of      32    0%
  Number of RAMB8BWERs:                         0 out of      64    0%
  Number of BUFIO2/BUFIO2_2CLKs:                0 out of      32    0%
  Number of BUFIO2FB/BUFIO2FB_2CLKs:            0 out of      32    0%
  Number of BUFG/BUFGMUXs:                      1 out of      16    6%
    Number used as BUFGs:                       1
    Number used as BUFGMUX:                     0
  Number of DCM/DCM_CLKGENs:                    0 out of       4    0%
  Number of ILOGIC2/ISERDES2s:                  0 out of     248    0%
  Number of IODELAY2/IODRP2/IODRP2_MCBs:        0 out of     248    0%
  Number of OLOGIC2/OSERDES2s:                  0 out of     248    0%
  Number of BSCANs:                             0 out of       4    0%
  Number of BUFHs:                              0 out of     128    0%
  Number of BUFPLLs:                            0 out of       8    0%
  Number of BUFPLL_MCBs:                        0 out of       4    0%
  Number of DSP48A1s:                           0 out of      32    0%
  Number of ICAPs:                              0 out of       1    0%
  Number of MCBs:                               0 out of       2    0%
  Number of PCILOGICSEs:                        0 out of       2    0%
  Number of PLL_ADVs:                           0 out of       2    0%
  Number of PMVs:                               0 out of       1    0%
  Number of STARTUPs:                           0 out of       1    0%
  Number of SUSPEND_SYNCs:                      0 out of       1    0%

Average Fanout of Non-Clock Nets:               4.64

Peak Memory Usage:  694 MB
Total REAL time to MAP completion:  28 secs
Total CPU time to MAP completion:   26 secs

Table of Contents
-----------------
Section 1 - Errors
Section 2 - Warnings
Section 3 - Informational
Section 4 - Removed Logic Summary
Section 5 - Removed Logic
Section 6 - IOB Properties
Section 7 - RPMs
Section 8 - Guide Report
Section 9 - Area Group and Partition Summary
Section 10 - Timing Report
Section 11 - Configuration String Information
Section 12 - Control Set Information
Section 13 - Utilization by Hierarchy

Section 1 - Errors
------------------

Section 2 - Warnings
--------------------

Section 3 - Informational
-------------------------
INFO:MapLib:562 - No environment variables are currently set.
INFO:LIT:244 - All of the single ended outputs in this design are using slew
   rate limited output drivers. The delay on speed critical single ended outputs
   can be dramatically reduced by designating them as fast outputs.
```

```
INFO:Pack:1716 - Initializing temperature to 85.000 Celsius. (default - Range:
   0.000 to 85.000 Celsius)
INFO:Pack:1720 - Initializing voltage to 1.140 Volts. (default - Range: 1.140 to
   1.260 Volts)
INFO:Map:215 - The Interim Design Summary has been generated in the MAP Report
   (.mrp).
INFO:Pack:1650 - Map created a placed design.

Section 4 - Removed Logic Summary
---------------------------------
   2 block(s) optimized away

Section 5 - Removed Logic
-------------------------

Optimized Block(s):
TYPE  BLOCK
GND  XST_GND
VCC  XST_VCC

To enable printing of redundant blocks removed and signals merged, set the
detailed map report option and rerun map.

Section 6 - IOB Properties
--------------------------
```

IOB Name	Type	Direction	IO Standard	Diff Term	Drive Strength	Sle Rat
a1	IOB	OUTPUT	LVCMOS25		12	SLO
a2	IOB	OUTPUT	LVCMOS25		12	SLO
a3	IOB	OUTPUT	LVCMOS25		12	SLO
a4	IOB	OUTPUT	LVCMOS25		12	SLO
add1	IOB	INPUT	LVCMOS25			
add2	IOB	INPUT	LVCMOS25			
add3	IOB	INPUT	LVCMOS25			
add4	IOB	INPUT	LVCMOS25			
clk	IOB	INPUT	LVCMOS25			
current_state<0>	IOB	OUTPUT	LVCMOS25		12	SLO
current_state<1>	IOB	OUTPUT	LVCMOS25		12	SLO
current_state<2>	IOB	OUTPUT	LVCMOS25		12	SLO
current_state<3>	IOB	OUTPUT	LVCMOS25		12	SLO
led_seg<0>	IOB	OUTPUT	LVCMOS25		12	SLO
led_seg<1>	IOB	OUTPUT	LVCMOS25		12	SLO
led_seg<2>	IOB	OUTPUT	LVCMOS25		12	SLO
led_seg<3>	IOB	OUTPUT	LVCMOS25		12	SLO
led_seg<4>	IOB	OUTPUT	LVCMOS25		12	SLO
led_seg<5>	IOB	OUTPUT	LVCMOS25		12	SLO
led_seg<6>	IOB	OUTPUT	LVCMOS25		12	SLO
rst	IOB	INPUT	LVCMOS25			
rst1	IOB	INPUT	LVCMOS25			
rst2	IOB	INPUT	LVCMOS25			
time_left<0>	IOB	OUTPUT	LVCMOS25		12	SLO
time_left<1>	IOB	OUTPUT	LVCMOS25		12	SLO
time_left<2>	IOB	OUTPUT	LVCMOS25		12	SLO
time_left<3>	IOB	OUTPUT	LVCMOS25		12	SLO
time_left<4>	IOB	OUTPUT	LVCMOS25		12	SLO
time_left<5>	IOB	OUTPUT	LVCMOS25		12	SLO

| time_left<6> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| time_left<7> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| time_left<8> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| time_left<9> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| time_left<10> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| time_left<11> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| time_left<12> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| time_left<13> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val1<0> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val1<1> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val1<2> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val1<3> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val2<0> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val2<1> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val2<2> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val2<3> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val3<0> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val3<1> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val3<2> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val3<3> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val4<0> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val4<1> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val4<2> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
| val4<3> | IOB | OUTPUT | LVCMOS25 | | 12 | SLO
+---

Section 7 - RPMs

Section 8 - Guide Report

Guide not run on this design.

Section 9 - Area Group and Partition Summary

Partition Implementation Status

  No Partitions were found in this design.

Area Group Information

  No area groups were found in this design.

Section 10 - Timing Report

A logic-level (pre-route) timing report can be generated by using Xilinx static
timing analysis tools, Timing Analyzer (GUI) or TRCE (command line), with the
mapped NCD and PCF files. Please note that this timing report will be generated
using estimated delay information. For accurate numbers, please generate a
timing report with the post Place and Route NCD file.

For more information about the Timing Analyzer, consult the Xilinx Timing

```
Analyzer Reference Manual; for more information about TRCE, consult the Xilinx
Command Line Tools User Guide "TRACE" chapter.

Section 11 - Configuration String Details
----------------------------------------
Use the "-detail" map option to print out Configuration Strings

Section 12 - Control Set Information
------------------------------------
Use the "-detail" map option to print out Control Set Information.

Section 13 - Utilization by Hierarchy
-------------------------------------
Use the "-detail" map option to print out the Utilization by Hierarchy section.
```