Lab 4: Finite State Machine: Parking Meter

Ethan Ngo

TA: Ananya Ravikumar (Section 3)

June 6, 2021

Contents

1 Introduction

The focus of this lab is to learn about finite state machines and directly implementing these ideas and behaviors through the design of a parking meter. Finite State Machines (FSMs) are very useful in a wide variety of real world systems. An FSM has some finite number of states, set to one specific state at a given time. FSMs can be designed in two different ways: as a Moore or Mealy machine. A Moore machine is a state machine whose outputs only depend on the current state. A Mealy machine is a state machine whose outputs depend on both the current state and the input states. Students are tasked with design a parking meter with the following characteristics:

1. Inputs add their respective times, mimicking the insertion of coins into a parking meter, which displays the time remaining with a 7-segment LED display.
2. The remaining time on the parking meter will dictate the speed at which the parking meter flashes: when zero, 0000 should be flashed at 1 Hz with a 50% duty cycle; when below 180, the time remaining should flash at 0.5 Hz with a 50% duty cycle; when above 180, there will be no flashing.
3. The maximum time that can be displayed is 9999, and the time remaining must not exceed that number.

Below is a diagram of how the 7-segment display works:

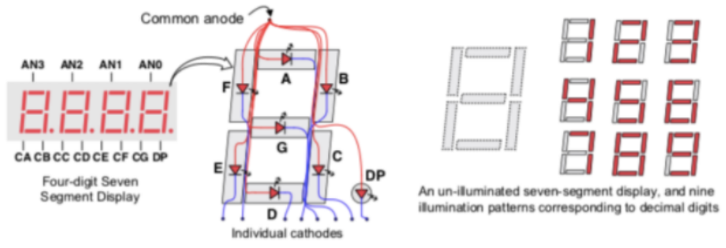
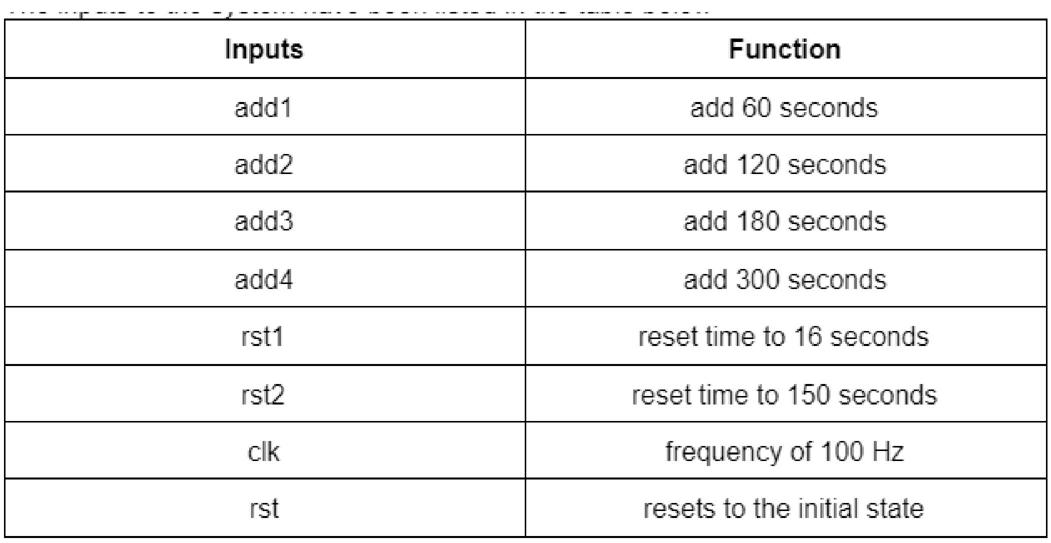

Common anode
AN3 AN2 AN1 AN0
CA CB CC CD CE CF CG DP
Four-digit Seven Segment Display
A
B
C
D
E
F
G
DP
Individual cathodes
An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits

Figure 1: LED Display

2 Module Description

For the design of the FSM, I utilized the design specifications to help me plan out my work, with the knowledge of what my inputs are required to be.



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

Figure 2: Parking Meter Inputs Overview

Using this ”schematic”, I created an FSM Moore machine, where my outputs are dependent on the current state. Descriptions of the states are as explained below:

- When the rst signal is high, we current state of the parking meter with be the INITIAL state, signifying that 0 time is remaining. From this, we wait for inputs before transitioning to a different state. This, along with rst1 and rst2, overrides any other signals.
- In addition to that, there exists to other reset signals, rst1 and rst2. If these signals are high, they will set the state to RST16 or RST150, which will set the remaining time to 16 or 150 seconds, respectively, which decreases by 1 every second. This, along with rst, overrides any other signals.
- From these above states, if the add1/add2/add3/add4 signal goes high, we will transition to the corresponding ADD60/ADD120/ADD180/ADD300 state. These

state will each add the necessary time to the parking meter. However, this does not change the time being decremented every second, irregardless of the addition.

- From the RST16, RST150, or any of the ADD signals, without any other inputs, we will transition the the CHECK state, which will check the time remaining and make sure we display the time in the correct manner, (e.g. with or without flashing).
- In the CHECK state, everything is standard, time still decrements, and if we get another signal, we transition to the next necessary signal. In this state, we update the display to make sure it displays at the proper frequency based on the time remaining. If the time is less than 180 seconds, we transition to the SLOW_FLASH state.
- In the SLOW_FLASH state, we display at 0.5 Hz on a 50% duty cycle.
- Note: In all state, time is decremented every second, and if a new input occurs, we transition to the proper next state.

To implement this in Verilog, I created a parking_meter module that mimics the schematic from the specifications. Using parameters to create variables to represent all of the necessary states in a specific 4-bit binary encoding, a 4-bit current_state and next_state, and 14-bit remaining_time register to time remaining on the parking meter. In addition, there were a variety of internal counters and registers used to implement all the necessary requirements for the parking meter.

In Verilog, I utilize both sequential and combinational always blocks to update the current_state, next_state, remaining_time, and led_seg. Aside from those, there smaller always blocks that dealt with smaller details, such as whether the display should be flashing.

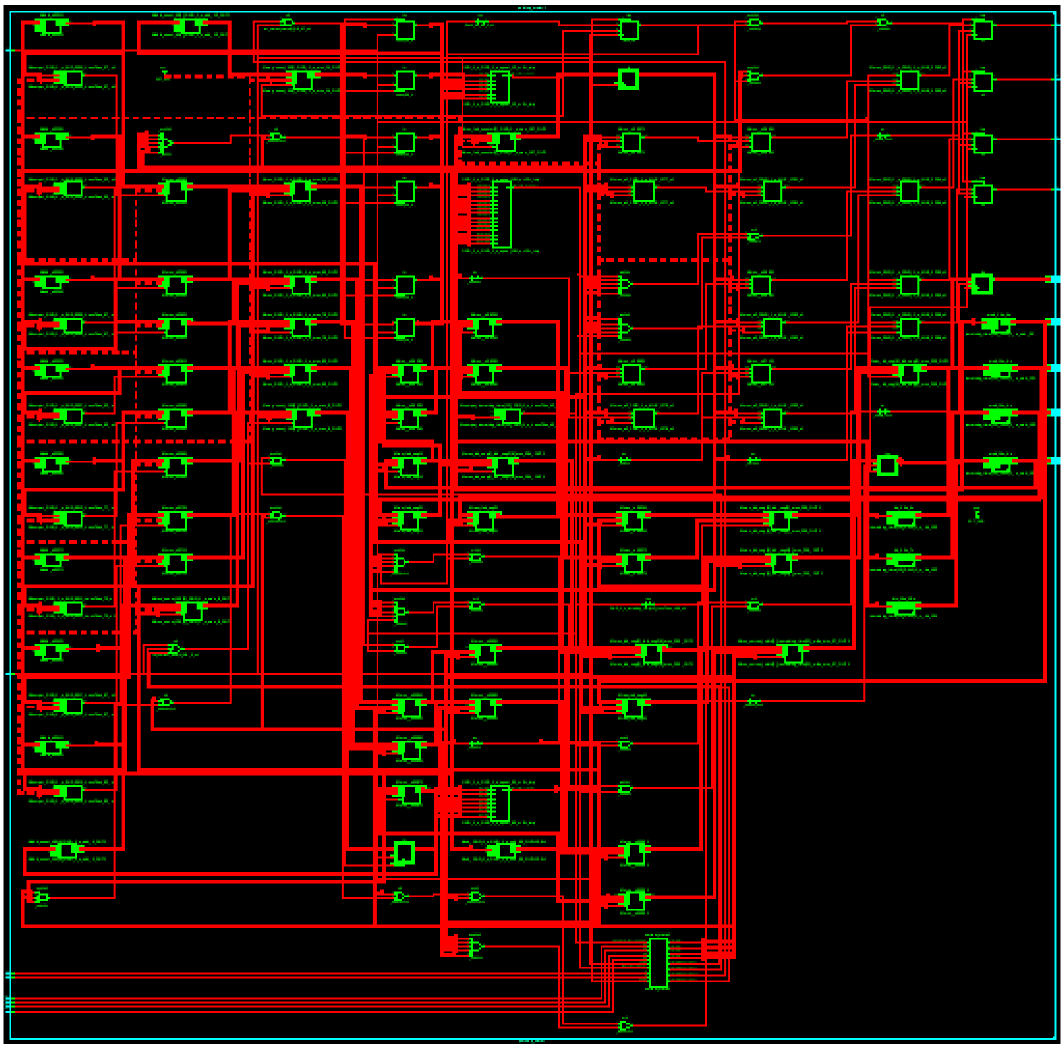
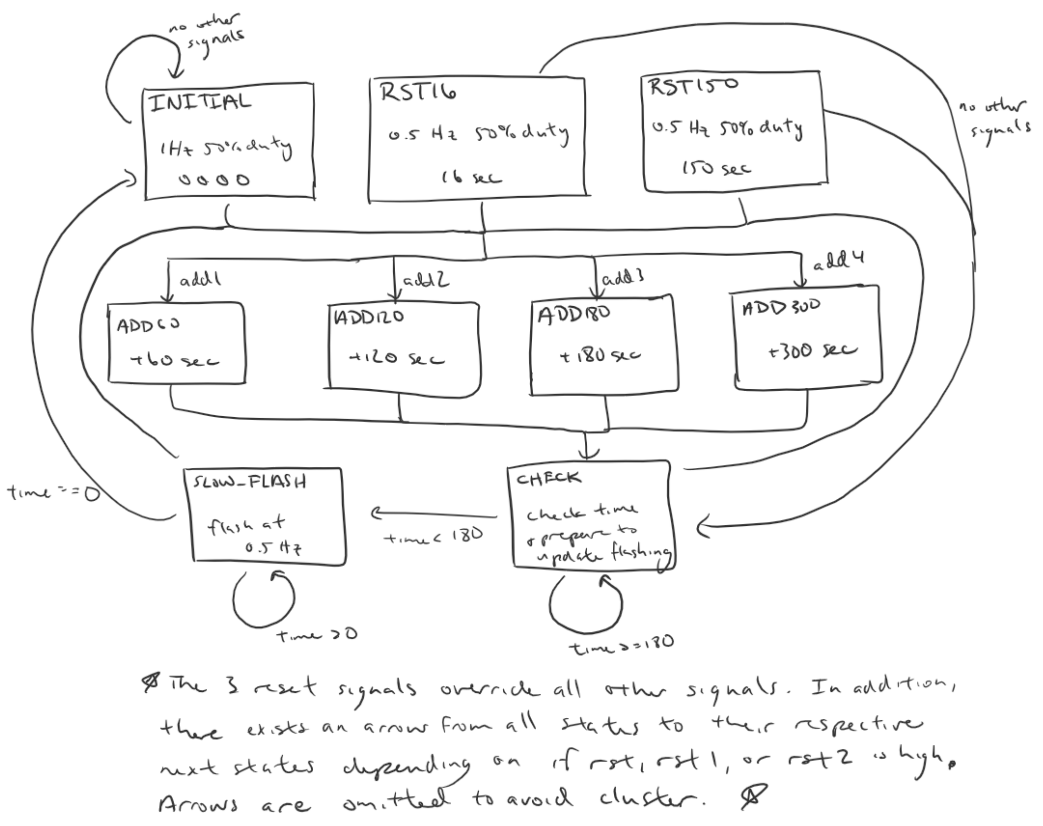


Figure 3: parking_meter Schematic

Above is the schematic of my vending machine FSM. It looks quite confusing and cluttered, which is the case because of the amount of registers and clock signals I use, which are updated by counters and manipulated with muxes to get the proper outputs. That, included with a variety of logic gates and flipflops, resulted in the very intricate schematic. Below is the overview of what the schematic achieves.



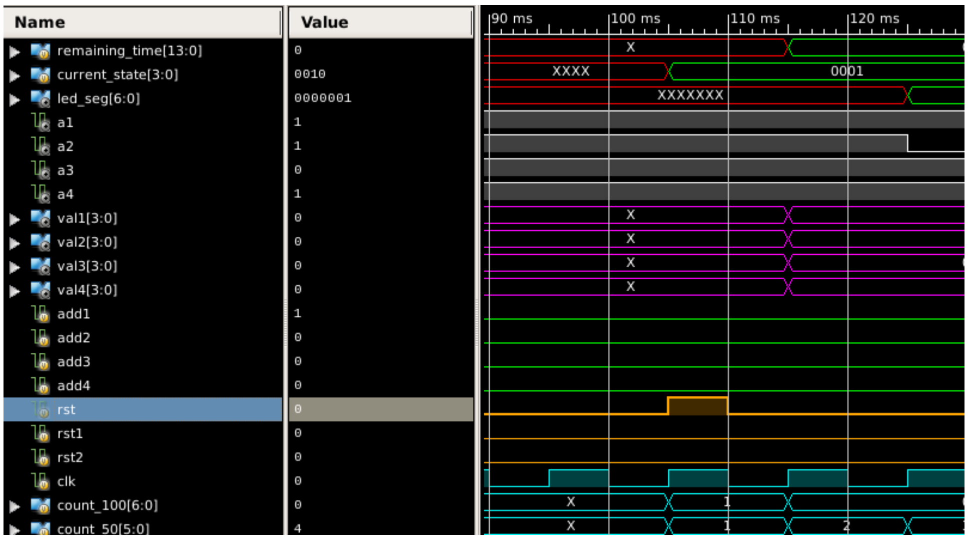
✗ The 3 reset signals override all other signals. In addition, there exists an arrow from all states to their respective next states depending on if rst, rst1, or rst2 is high. Arrows are omitted to avoid cluster. ✗

Figure 4: parking_meter Schematic Hand-Drawn and Simplified

3 Test-bench Simulation

To test my modules, I tested my main module parking_meter and compared the resulting waveforms to the output that I would expect if I had done it in my head. Below are all the test cases waveforms and a short description of what is being tested.

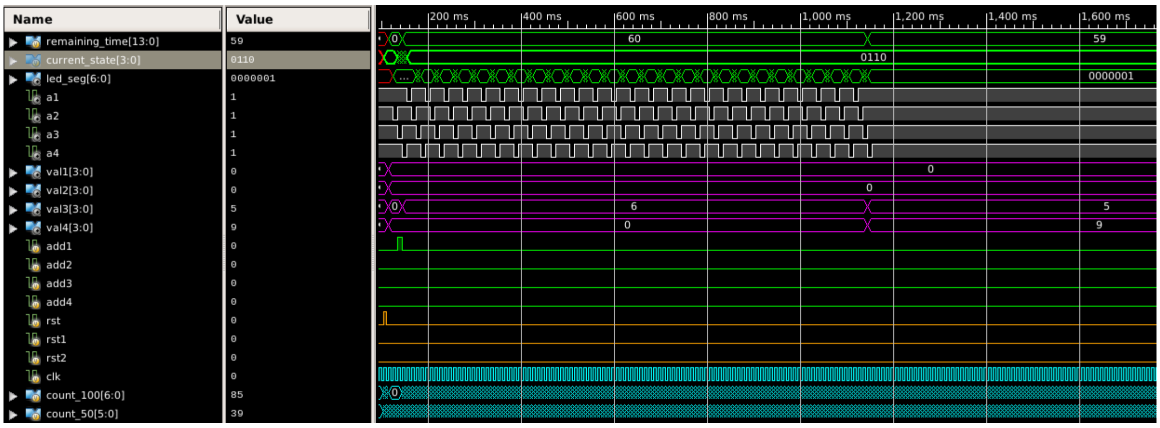
1. Reset Test Waveform: This test simply tested to make sure rst reset all outputs and put us back into the INITIAL state.



Name	Value
remaining_time[13:0]	0
current_state[3:0]	0010
led_seg[6:0]	0000001
a1	1
a2	1
a3	0
a4	1
val1[3:0]	0
val2[3:0]	0
val3[3:0]	0
val4[3:0]	0
add1	1
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	0
count_100[6:0]	0
count_50[5:0]	4

Figure 5: parking_meter Reset Test Waveform

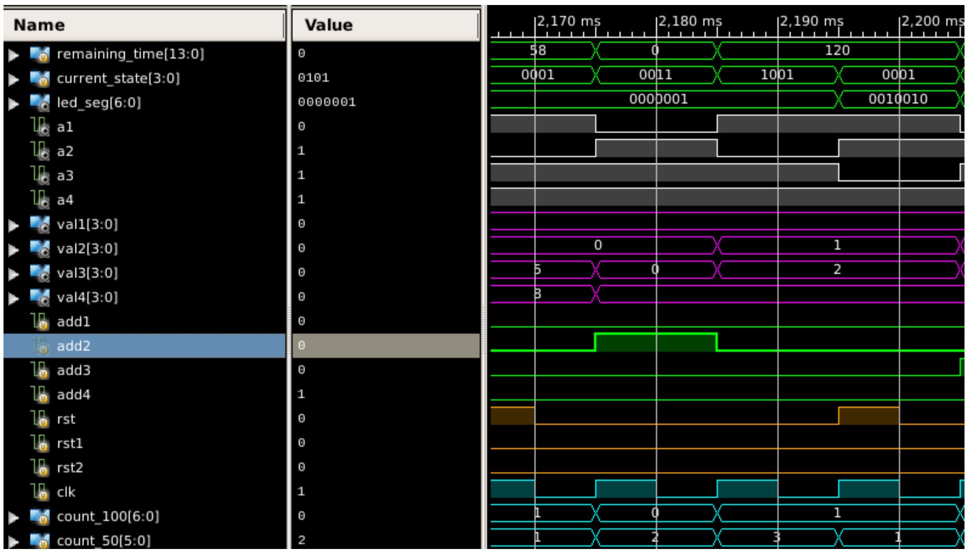
2. ADD60 and 0.5 Hz Test: This test checked to make sure that when add1 went high, that the time remaining on the parking meter would increase by 60 seconds. In addition, we let the time decrease by a few seconds to make sure that it properly decremented and to show that on odd decrements, the display would not show anything (i.e that a1-a4 were all 1, indicating the display showed nothing).



Name	Value
remaining_time[13:0]	59
current_state[3:0]	0110
led_seg[6:0]	0000001
a1	1
a2	1
a3	1
a4	1
val1[3:0]	0
val2[3:0]	0
val3[3:0]	5
val4[3:0]	9
add1	0
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	0
count_100[6:0]	85
count_50[5:0]	39

Figure 6: parking_meter ADD60 Test Waveform

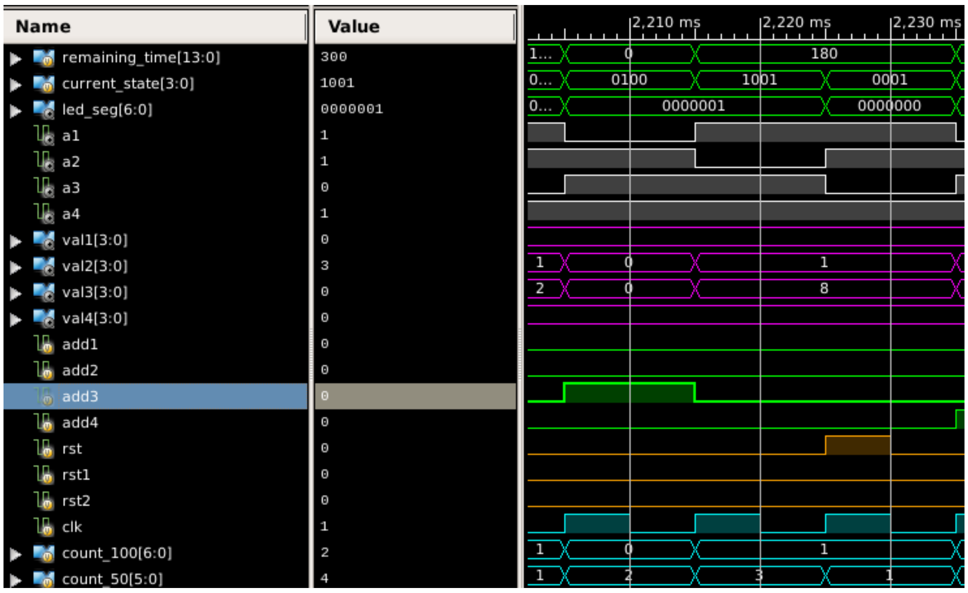
3. ADD120 Test: This test simply ensured that when add2 was 1, that it would increase the remaining time by 120 seconds.



Name	Value
remaining_time[13:0]	0
current_state[3:0]	0101
led_seg[6:0]	0000001
a1	0
a2	1
a3	1
a4	1
val1[3:0]	0
val2[3:0]	0
val3[3:0]	0
val4[3:0]	0
add1	0
add2	0
add3	0
add4	1
rst	0
rst1	0
rst2	0
clk	1
count_100[6:0]	0
count_50[5:0]	2

Figure 7: parking_meter ADD120 Test Waveform

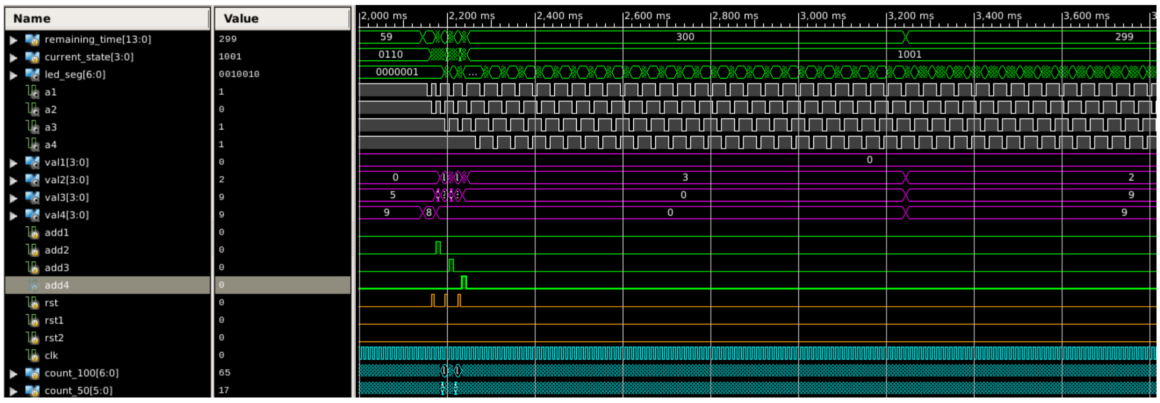
4. ADD180 Test: This test simply ensured that when add3 was 1, that it would increase the remaining time by 180 seconds.



Name	Value
remaining_time[13:0]	300
current_state[3:0]	1001
led_seg[6:0]	0000001
a1	1
a2	1
a3	0
a4	1
val1[3:0]	0
val2[3:0]	3
val3[3:0]	0
val4[3:0]	0
add1	0
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	1
count_100[6:0]	2
count_50[5:0]	4

Figure 8: parking_meter ADD180 Test Waveform

5. ADD300 and 1 Hz Test: This test checked to make sure that when add4 went high, that the time remaining on the parking meter would increase by 300 seconds. In addition, we let the time decrease by a few seconds to make sure that it properly decremented and to show that on odd decrements, the display would still display the time remaining (i.e that a1-a4 were still changing).

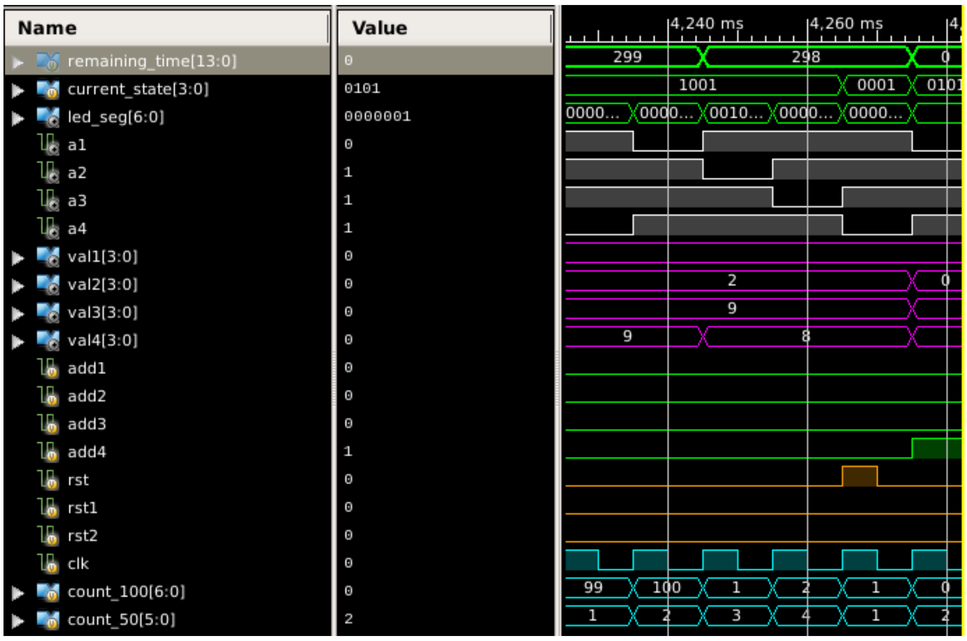


Name	Value
remaining_time[13:0]	299
current_state[3:0]	1001
led_seg[6:0]	0010010
a1	1
a2	0
a3	1
a4	1
val1[3:0]	0
val2[3:0]	2
val3[3:0]	9
val4[3:0]	9
add1	0
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	0
count_100[6:0]	65
count_50[5:0]	17

Figure 9: parking_meter ADD300 Test Waveform

6. Correct 7-Segment Display Test: This test checked that our 4 7-segment display showed the proper time remaining. We see that the current time is 298 seconds. We

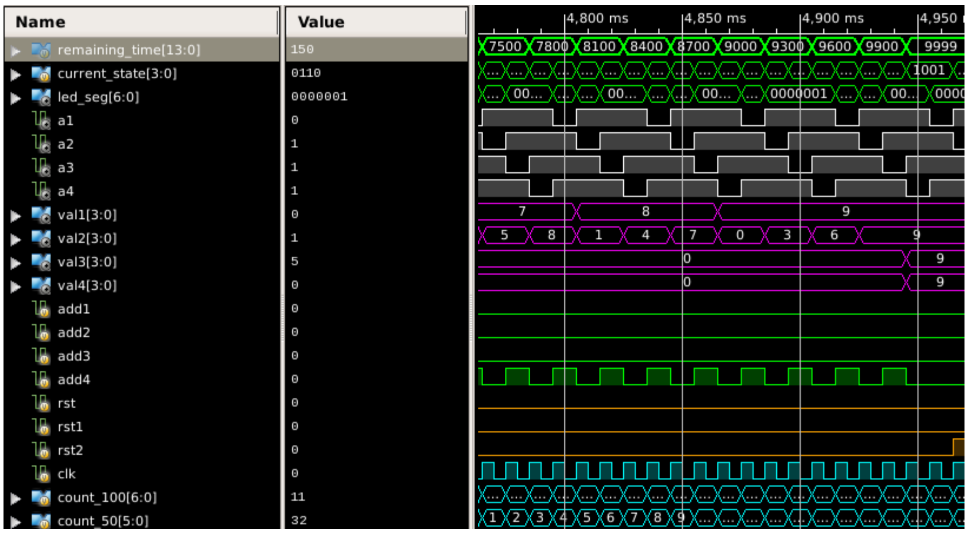
can see that at each a1-a4 going low, the led_seg value corresponds to the proper segments of the 7-segment display being turned on.



Name	Value
remaining_time[13:0]	0
current_state[3:0]	0101
led_seg[6:0]	0000001
a1	0
a2	1
a3	1
a4	1
val1[3:0]	0
val2[3:0]	0
val3[3:0]	0
val4[3:0]	0
add1	0
add2	0
add3	0
add4	1
rst	0
rst1	0
rst2	0
clk	0
count_100[6:0]	0
count_50[5:0]	2

Figure 10: parking_meter 7-Segment Display Test Waveform

7. Maximum Time Test: This test checks to make sure that the maximum time remaining is 9999 seconds.

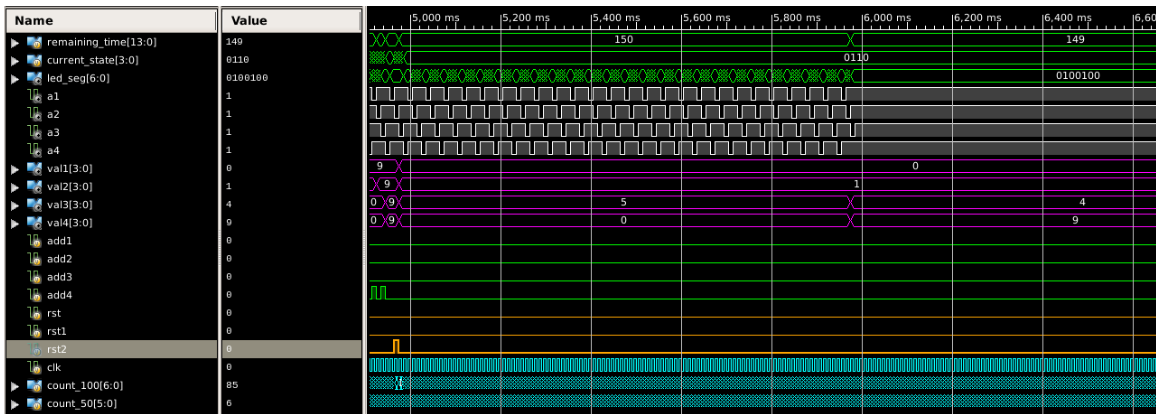


Name	Value
remaining_time[13:0]	150
current_state[3:0]	0110
led_seg[6:0]	0000001
a1	0
a2	1
a3	1
a4	1
val1[3:0]	0
val2[3:0]	1
val3[3:0]	5
val4[3:0]	0
add1	0
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	0
count_100[6:0]	11
count_50[5:0]	32

Figure 11: parking_meter Maximum Time Test Waveform

8. Rst2 Test: This test makes sure that when rst2 goes high, that the time remaining

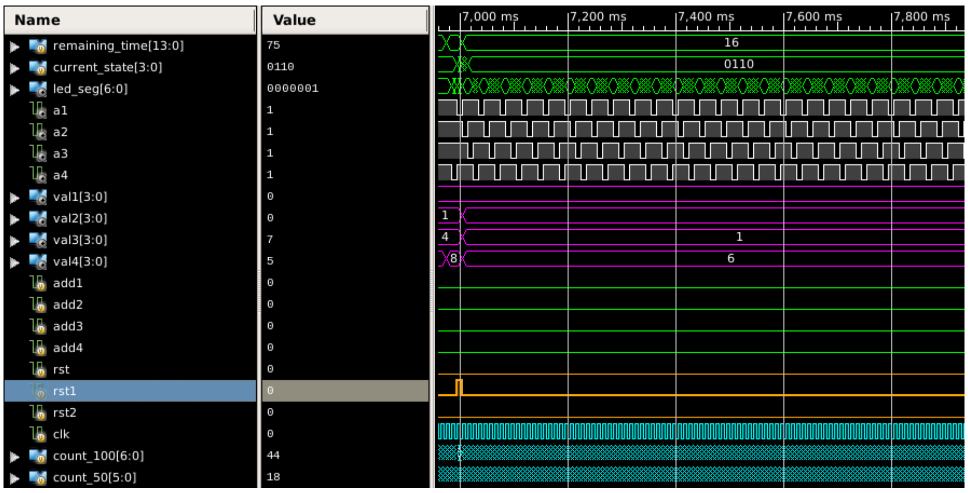
will be set to 150 seconds.



Name	Value
remaining_time[13:0]	149
current_state[3:0]	0110
led_seg[6:0]	0100100
a1	1
a2	1
a3	1
a4	1
val1[3:0]	0
val2[3:0]	1
val3[3:0]	4
val4[3:0]	9
add1	0
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	0
count_100[6:0]	85
count_50[5:0]	6

Figure 12: parking_meter Reset2 Test Waveform

9. Rst1 Test: This test makes sure that when rst1 goes high, that the time remaining will be set to 16 seconds.

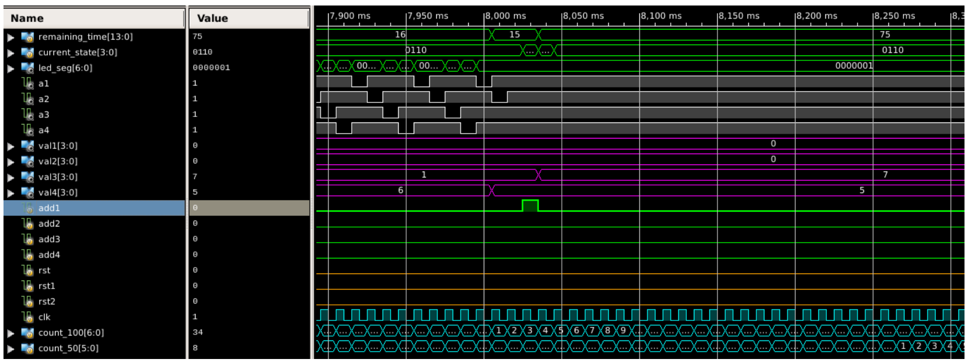


Name	Value
remaining_time[13:0]	75
current_state[3:0]	0110
led_seg[6:0]	0000001
a1	1
a2	1
a3	1
a4	1
val1[3:0]	0
val2[3:0]	0
val3[3:0]	7
val4[3:0]	5
add1	0
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	0
count_100[6:0]	44
count_50[5:0]	18

Figure 13: parking_meter Reset1 Test Waveform

10. Adding Time During Odd Decrement Test: This test checks that when the time remaining is less than 180 seconds (i.e that the display is flashing at 0.5 Hz), on an odd decrement where the display doesn't show anything, if we were to add time to the parking meter, that it would properly add the amount of time to the odd time amount, and not the even time. In this example, we add 60 to the 15 seconds, so

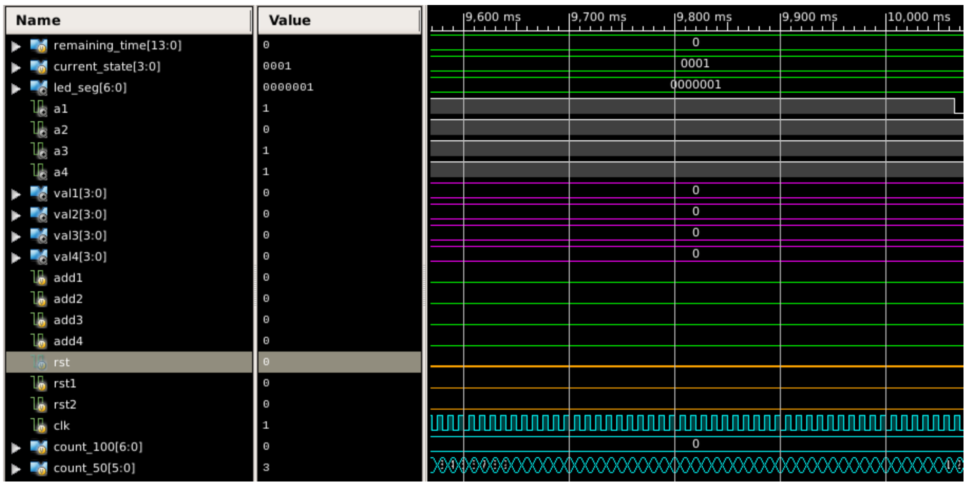
it should display 75 seconds, and not 76 seconds that would show if we added 60 to the last displayed value of 16 seconds.



Name	Value
remaining_time[13:0]	75
current_state[3:0]	0110
led_seg[6:0]	0000001
a1	1
a2	1
a3	1
a4	1
val1[3:0]	0
val2[3:0]	0
val3[3:0]	7
val4[3:0]	5
add1	0
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	1
count_100[6:0]	34
count_50[5:0]	8

Figure 14: parking_meter Odd Decrement Addition Test Waveform

11. Zero Timer Test: This test checks to make sure that when the timer is 0, it will still display 0 and also not go into negative numbers, for instance.

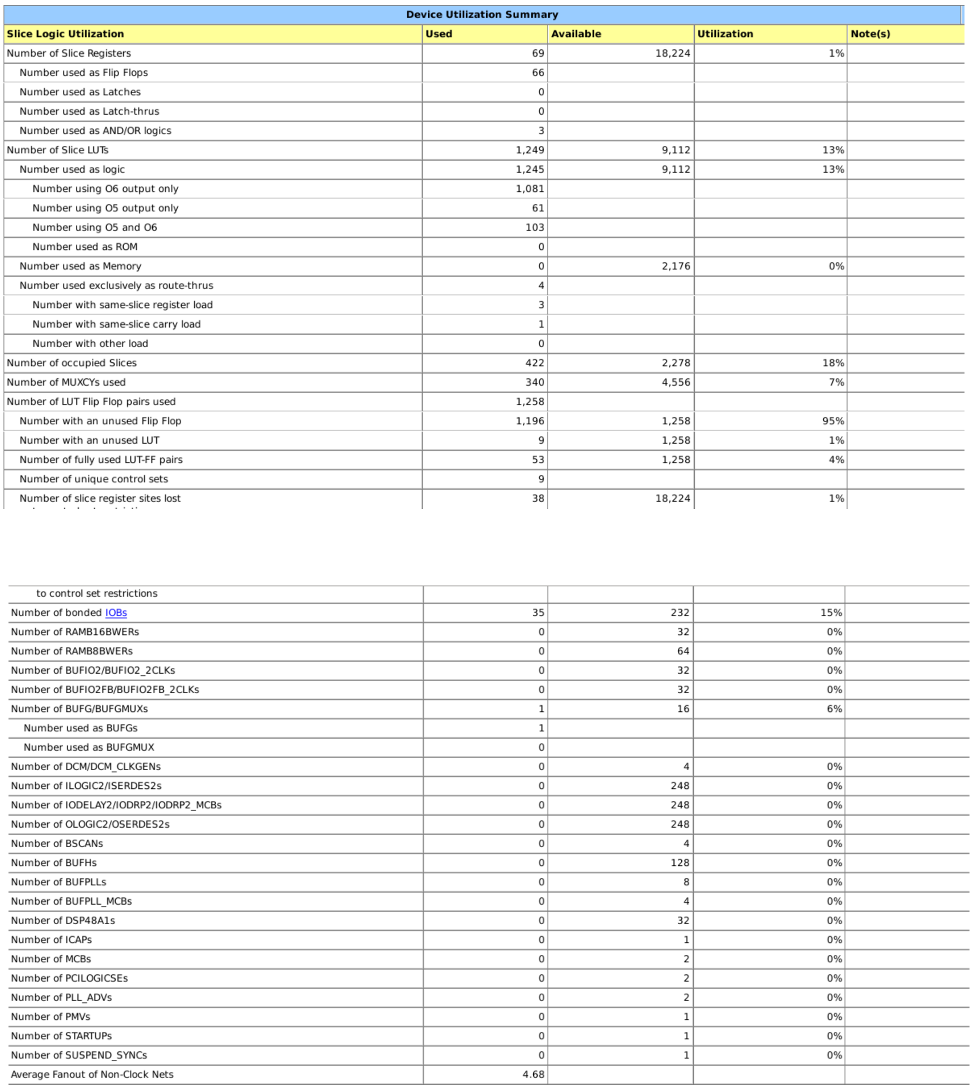


Name	Value
remaining_time[13:0]	0
current_state[3:0]	0001
led_seg[6:0]	0000001
a1	1
a2	0
a3	1
a4	1
val1[3:0]	0
val2[3:0]	0
val3[3:0]	0
val4[3:0]	0
add1	0
add2	0
add3	0
add4	0
rst	0
rst1	0
rst2	0
clk	1
count_100[6:0]	0
count_50[5:0]	3

Figure 15: parking_meter Zero Timer Test Waveform

4 Reports

4.1 ISE Design Overview Summary Report

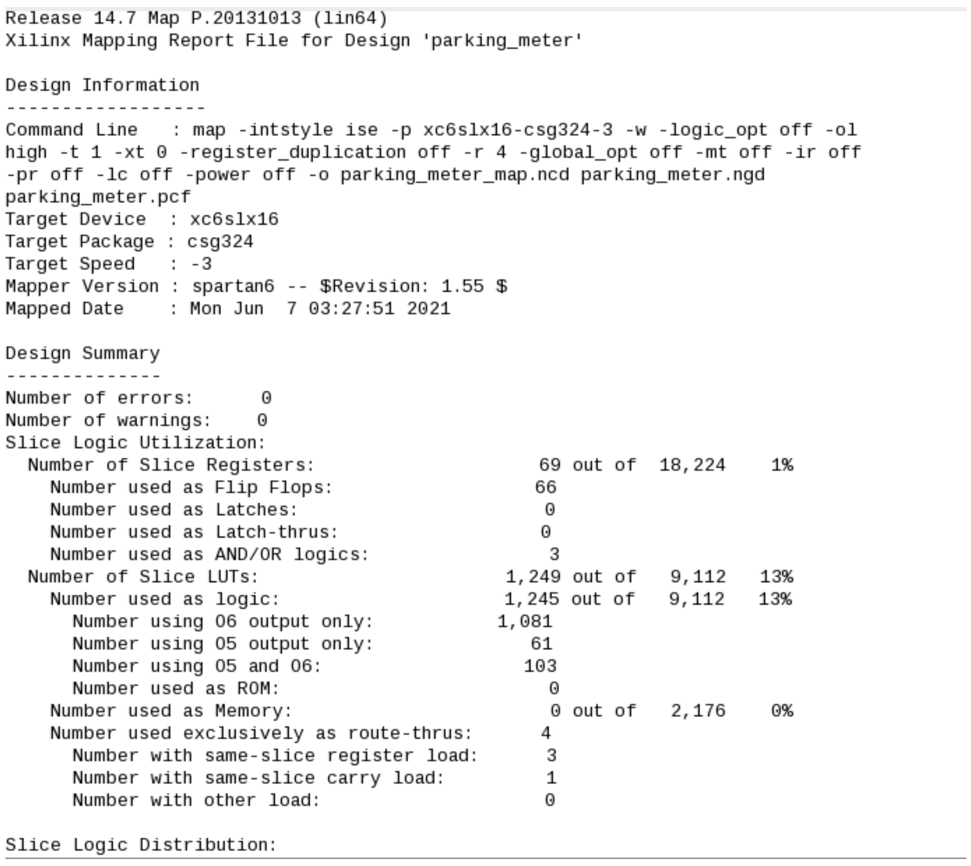


Device Utilization Summary				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	69	18,224	1%	
Number used as Flip Flops	66			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	3			
Number of Slice LUTs	1,249	9,112	13%	
Number used as logic	1,245	9,112	13%	
Number using O6 output only	1,081			
Number using O5 output only	61			
Number using O5 and O6	103			
Number used as ROM	0			
Number used as Memory	0	2,176	0%	
Number used exclusively as route-thrus	4			
Number with same-slice register load	3			
Number with same-slice carry load	1			
Number with other load	0			
Number of occupied Slices	422	2,278	18%	
Number of MUXCYs used	340	4,556	7%	
Number of LUT Flip Flop pairs used	1,258			
Number with an unused Flip Flop	1,196	1,258	95%	
Number with an unused LUT	9	1,258	1%	
Number of fully used LUT-FF pairs	53	1,258	4%	
Number of unique control sets	9			
Number of slice register sites lost to control set restrictions	38	18,224	1%	
Number of bonded IOBs	35	232	15%	
Number of RAMB16BWERs	0	32	0%	
Number of RAMB8BWERs	0	64	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	1	16	6%	
Number used as BUFGs	1			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	4	0%	
Number of ILOGIC2/ISERDES2s	0	248	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	248	0%	
Number of OLOGIC2/OSERDES2s	0	248	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	128	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	32	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	2	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	2	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	4.68			

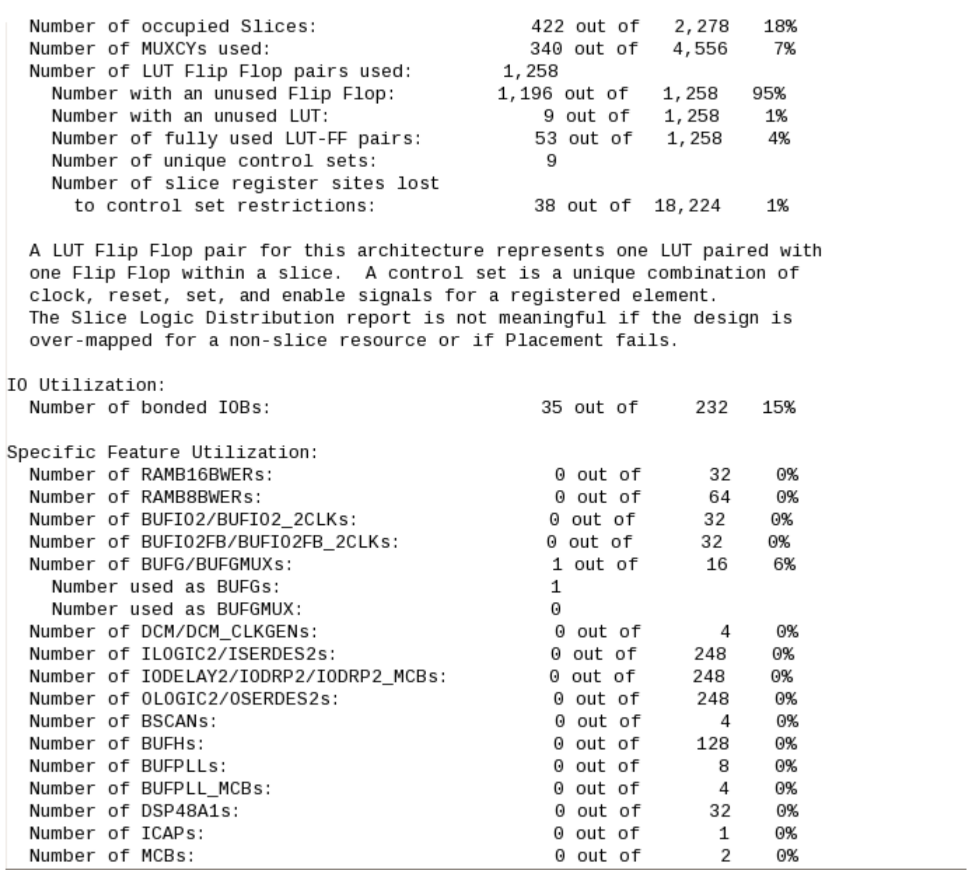
Figure 16: Design Summary

Above is the ISE Design Overview Summary Report. As indicated, there are no errors and simply a handful of warnings, which are simply warnings about potential truncation of register values. In addition, it is clear that a ton of registers, muxes, and flip-flops were utilized, which is as expected.

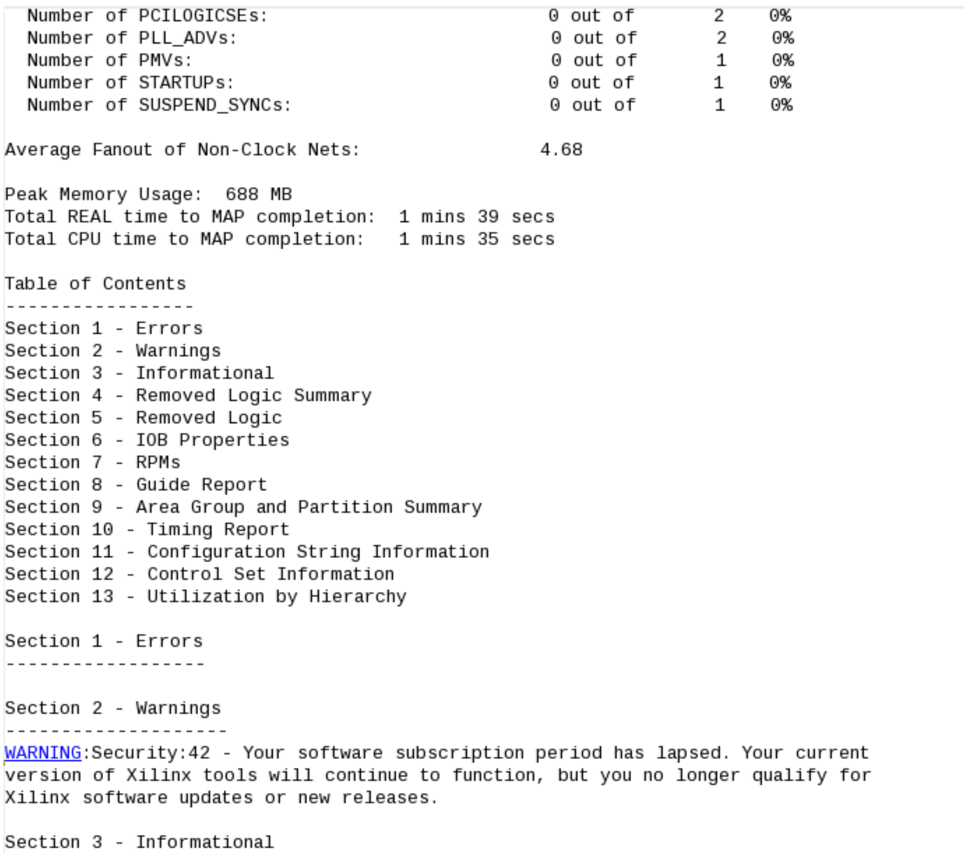
4.2 Map Report



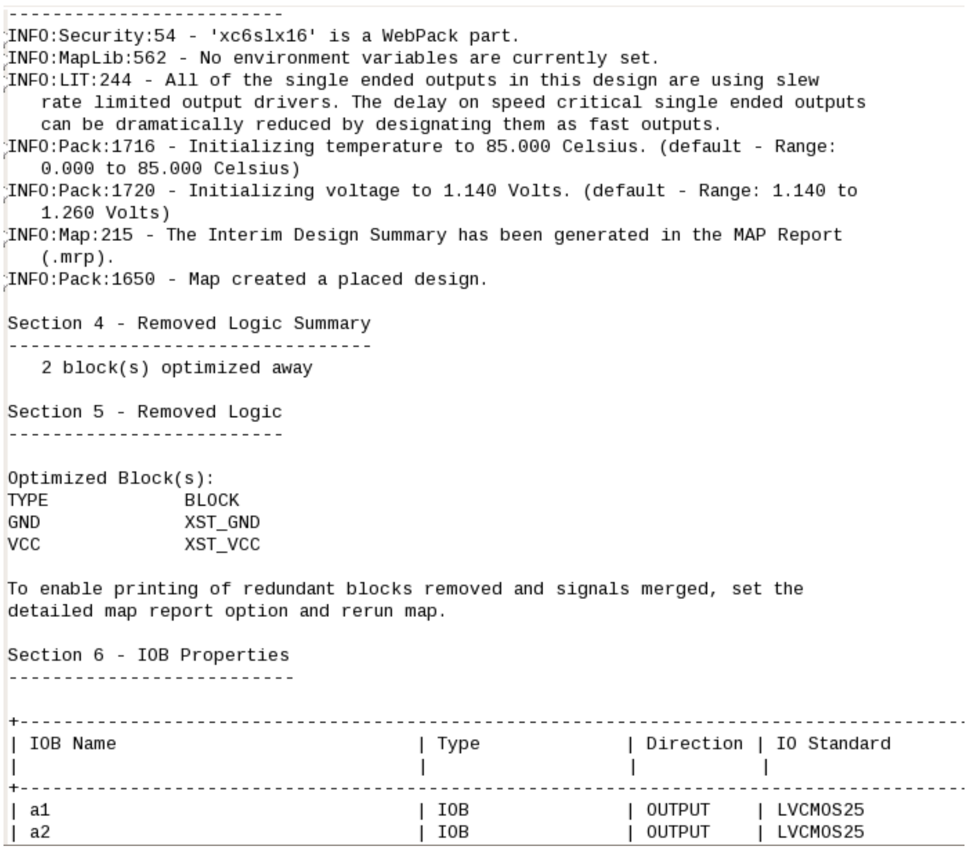
```
Release 14.7 Map P.20131013 (lin64)
Xilinx Mapping Report File for Design 'parking_meter'

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    : Mon Jun  7 03:27:51 2021

Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                    69 out of  18,224    1%
    Number used as Flip Flops:                  66
    Number used as Latches:                      0
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                3
  Number of Slice LUTs:                      1,249 out of   9,112   13%
    Number used as logic:                    1,245 out of   9,112   13%
      Number using O6 output only:           1,081
      Number using O5 output only:              61
      Number using O5 and O6:                  103
      Number used as ROM:                        0
    Number used as Memory:                       0 out of   2,176    0%
    Number used exclusively as route-thrus:      4
      Number with same-slice register load:      3
      Number with same-slice carry load:         1
      Number with other load:                    0

Slice Logic Distribution:
```



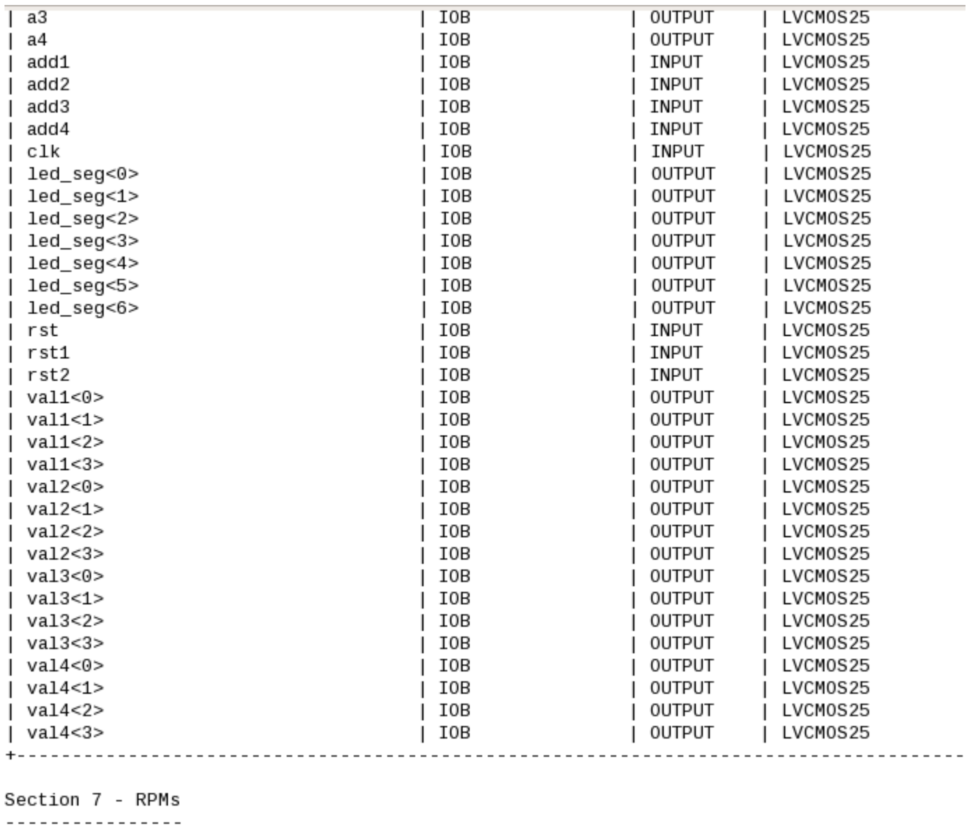
```
  Number of occupied Slices:                   422 out of   2,278   18%
  Number of MUXCYs used:                       340 out of   4,556    7%
  Number of LUT Flip Flop pairs used:        1,258
    Number with an unused Flip Flop:         1,196 out of   1,258   95%
    Number with an unused LUT:                   9 out of   1,258    1%
    Number of fully used LUT-FF pairs:          53 out of   1,258    4%
    Number of unique control sets:               9
    Number of slice register sites lost
      to control set restrictions:              38 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:                        35 out of     232   15%

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%
```



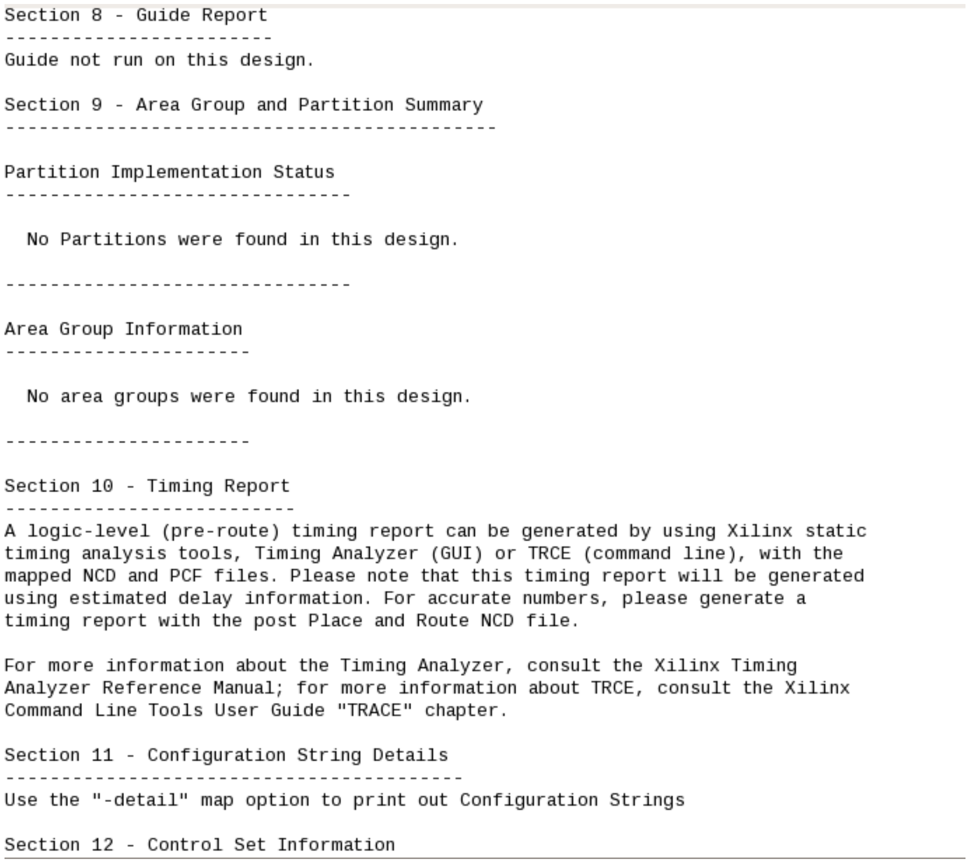
```
  Number of PCILOGICSEs:                        0 out of       2    0%
  Number of PLL_ADVs:                           0 out of       2    0%
  Number of PMVs:                               0 out of       1    0%
  Number of STARTUPs:                           0 out of       1    0%
  Number of SUSPEND_SYNCs:                      0 out of       1    0%

Average Fanout of Non-Clock Nets:              4.68

Peak Memory Usage:  688 MB
Total REAL time to MAP completion:  1 mins 39 secs
Total CPU time to MAP completion:   1 mins 35 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
--------------------
WARNING:Security:42 - Your software subscription period has lapsed. Your current
version of Xilinx tools will continue to function, but you no longer qualify for
Xilinx software updates or new releases.

Section 3 - Informational
```



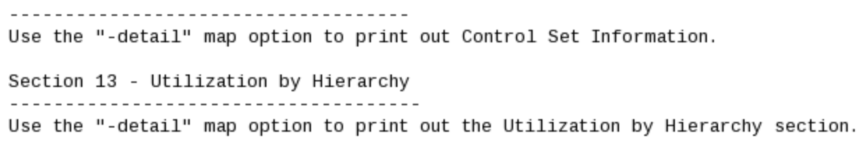
```
-------------------------
INFO:Security:54 - 'xc6slx16' is a WebPack part.
INFO:MapLib:562 - No environment variables are currently set.
INFO:LIT:244 - All of the single ended outputs in this design are using slew
   rate limited output drivers. The delay on speed critical single ended outputs
   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
|                                     |                   |           |
+-------------------------------------------------------------------------------------
| a1                                  | IOB               | OUTPUT    | LVCMOS25
| a2                                  | IOB               | OUTPUT    | LVCMOS25
```



```
| a3                                     | IOB                | OUTPUT     | LVCMOS25
| a4                                     | IOB                | OUTPUT     | LVCMOS25
| add1                                   | IOB                | INPUT      | LVCMOS25
| add2                                   | IOB                | INPUT      | LVCMOS25
| add3                                   | IOB                | INPUT      | LVCMOS25
| add4                                   | IOB                | INPUT      | LVCMOS25
| clk                                    | IOB                | INPUT      | LVCMOS25
| led_seg<0>                             | IOB                | OUTPUT     | LVCMOS25
| led_seg<1>                             | IOB                | OUTPUT     | LVCMOS25
| led_seg<2>                             | IOB                | OUTPUT     | LVCMOS25
| led_seg<3>                             | IOB                | OUTPUT     | LVCMOS25
| led_seg<4>                             | IOB                | OUTPUT     | LVCMOS25
| led_seg<5>                             | IOB                | OUTPUT     | LVCMOS25
| led_seg<6>                             | IOB                | OUTPUT     | LVCMOS25
| rst                                    | IOB                | INPUT      | LVCMOS25
| rst1                                   | IOB                | INPUT      | LVCMOS25
| rst2                                   | IOB                | INPUT      | LVCMOS25
| val1<0>                                | IOB                | OUTPUT     | LVCMOS25
| val1<1>                                | IOB                | OUTPUT     | LVCMOS25
| val1<2>                                | IOB                | OUTPUT     | LVCMOS25
| val1<3>                                | IOB                | OUTPUT     | LVCMOS25
| val2<0>                                | IOB                | OUTPUT     | LVCMOS25
| val2<1>                                | IOB                | OUTPUT     | LVCMOS25
| val2<2>                                | IOB                | OUTPUT     | LVCMOS25
| val2<3>                                | IOB                | OUTPUT     | LVCMOS25
| val3<0>                                | IOB                | OUTPUT     | LVCMOS25
| val3<1>                                | IOB                | OUTPUT     | LVCMOS25
| val3<2>                                | IOB                | OUTPUT     | LVCMOS25
| val3<3>                                | IOB                | OUTPUT     | LVCMOS25
| val4<0>                                | IOB                | OUTPUT     | LVCMOS25
| val4<1>                                | IOB                | OUTPUT     | LVCMOS25
| val4<2>                                | IOB                | OUTPUT     | LVCMOS25
| val4<3>                                | IOB                | OUTPUT     | LVCMOS25
+------------------------------------------------------------------------------------------

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.
```

Above is my Map Report. Although there is a lot of details, I can see that just like in the Synthesis Report, we are using a large amount of registers, muxes, and flip-flops, although not anywhere near the maximum number possible that I could use. In addition, there a handful of warnings which notify me of my bad practices that I wasn't even aware of myself, nor do I know much about how to fix them. Luckily however, it doesn't affect my code and the outputs of it, and simply is a practice I should not use in the future, such as warnings about potential truncations, but they don't affect my code.

5 Conclusion

In this lab, I designed a FSM parking meter, based on a Moore machine. I implemented a wide variety of topics discussed in lecture, including the use of parameters, switch-case statements, and a clock to create a simulated parking meter, which had similarities to the vending machine, as they were both Moore FSMs.

There were quite a few difficulties that I ran into. The more prominent ones included trying to figure out when the positive edge of the clock would occur as I was writing out my tests, so I had a few #(number) delays, in between setting certain inputs to high/low because I had to always make sure I was keeping track of when the positive edge of the clock was occurring. In addition, it took some time to try and figure out how to have two internal clocks in my code, that would display at different frequencies. In addition, it took some time and asking for help to figure out the waveform configuration, with being able to add certain variables/registers/etc to my waveform to make the simulation and debugging easier. Overall, I believe that this lab truly captured all the topics taught over the course of this class and it was fun seeing the outcome, as I was able to create a parking meter.