Project 2: Parking Meter

Haley Kim
405111152

Introduction and Requirement

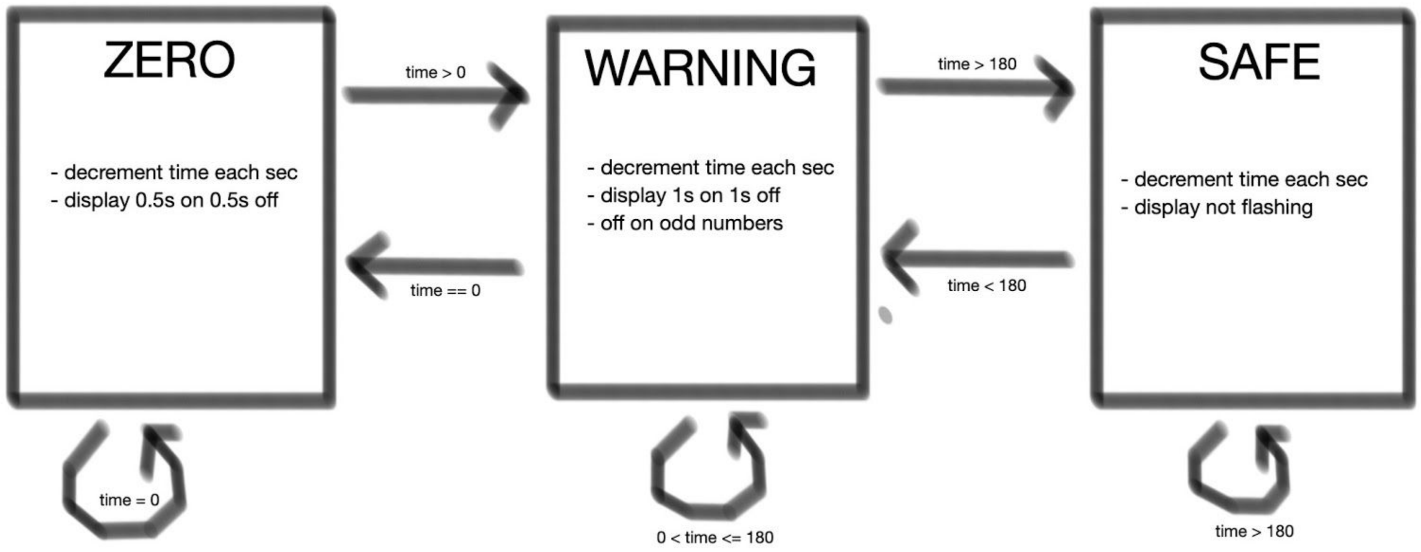
The parking meter for this project is a Moore machine whose output only results on the current state of the machine, regardless of the input fed in. The input helps determine the current state, and the current state determines the output. The parking meter needs to be able to handle each input and adjust its four seven-segment displays accordingly in order to represent the correct time left.

This project has eight inputs it must recognize and handle:

1. add1: adds 60 seconds
2. add2: adds 120 seconds
3. add3: adds 180 seconds
4. add4: adds 300 seconds
5. rst1: resets time to 16 seconds
6. rst2: resets time to 150 seconds
7. clk: 100Hz frequency clock
8. rst: resets to initial state

In the initial state, the parking meter must display 0000 for a period of 1 second with a 50% duty cycle. When any of the add buttons are pressed, the parking meter must add the corresponding time and begin counting down. When less than 180 seconds are remaining, the display must flash with a period of 2 seconds with a 50% duty cycle. When the time as expired, the display must exhibit the same behavior as the initial state. The maximum value to be displayed is 9999. Multiple buttons will not be pressed at the same time.

val1 through val4 represent the BCD values corresponding to each segment, while a1 through a4 represent the anodes driving the segments (low for each for one clock cycle). val1 is the least significant value, and a1 is the least significant digit.



Design Description

The main overarching module is parking_meter, with inputs:

    add1, add2, add3, add4,
    rst, rst1, rst2,
    clk,

and outputs:

    [6:0] led_seg,
    a1, a2, a3, a4,
    [3:0] val1, val2, val3, val4.

Main module parking_meter calls submodules:

    BCDencoder, and sevenseg.

parking_meter's logic:

- determines time left ([13:0] bin_time) based on inputs,
  - add1 adds 60 to time left, etc
  - decrments time left every 100 clk posedge using a div-by-100 pulse
    - (to decrement every second, not every 0.01 second)
- determines state based on time left,
  - based on time left, flash is set to 0, 1, or 2
    - if time left is 0, flash is 1

```
                if time left is smaller or equal to 180, flash is 2
                else flash is 0
    determines outputs (flashing) based on state
        if flash is 1, every 50 clk, off switches from 0 to 1 and 1 to 0
            (effectively off is 0 for 0.5s and 1 for 0.5s)
        if flash is 2, off is always 0 (we will use modulo for when flash is 2)
        if flash is 0, off is always 0
```

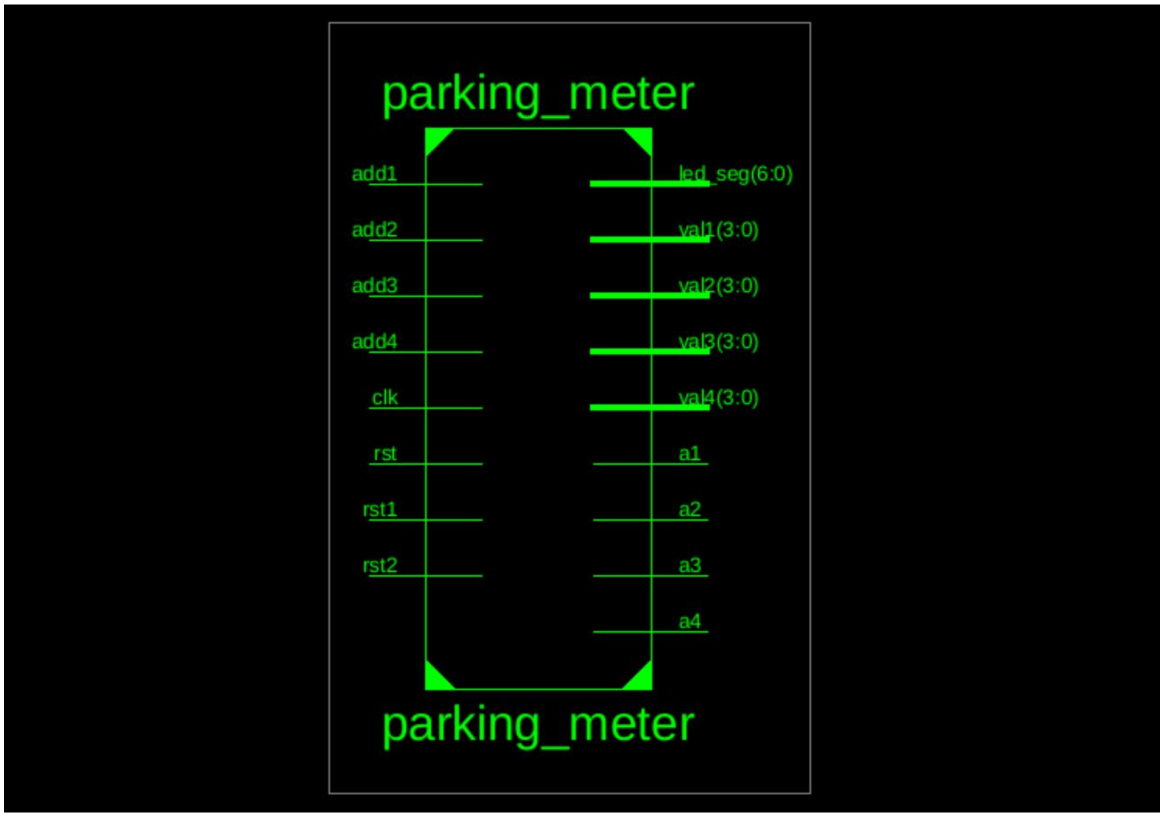
BCDencoder's logic:

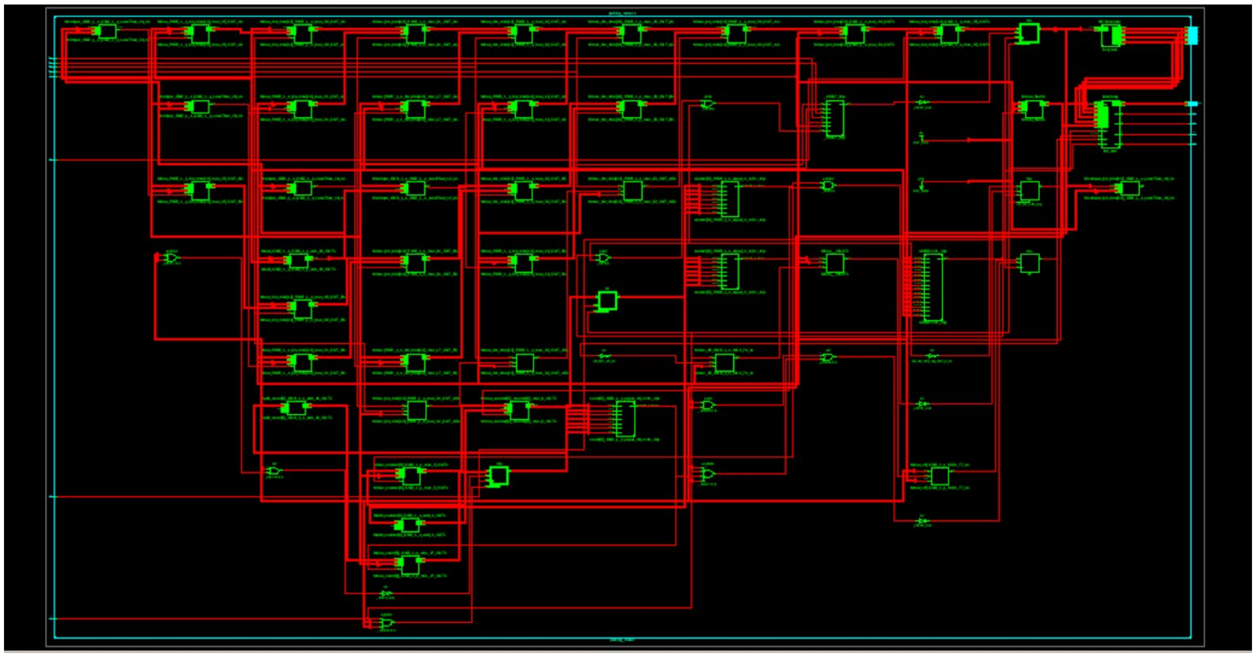
```
    takes in [13:0] "bin_time" as input
        turns 14 bit binary number into 4 binary-coded decimal numbers using modulo
        assigns to each val1, val2, val3, val4 a 4-bit binary coded decimal number
    outputs val1 through val4
```

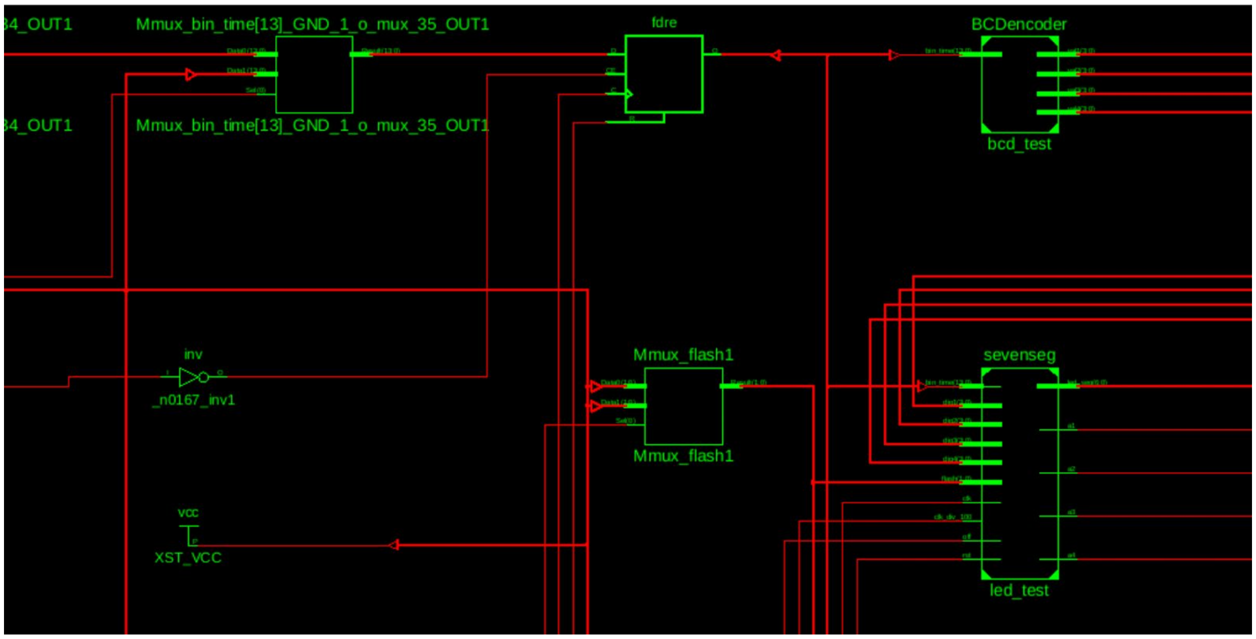
sevenseg's logic:

```
    takes in "bin_time", "clk", "rst", "off", "flash", "clk_div_100", "dig1", "dig2", "dig3",
    "dig4" as inputs
        function "ledsegdisplay" takes in a digit and outputs the corresponding
            seven-segment display binary
        if off is 1, automatically display nothing (led_seg = 111_1111)
        if flash is 2, then display nothing on odd numbers (modulo operation)
        otherwise at each posedge clk, oscillate between a1, a2, a3, and a4 to switch
            between digits, displays corresponding seven-segment binary for that digit
```

Simulation Documentation

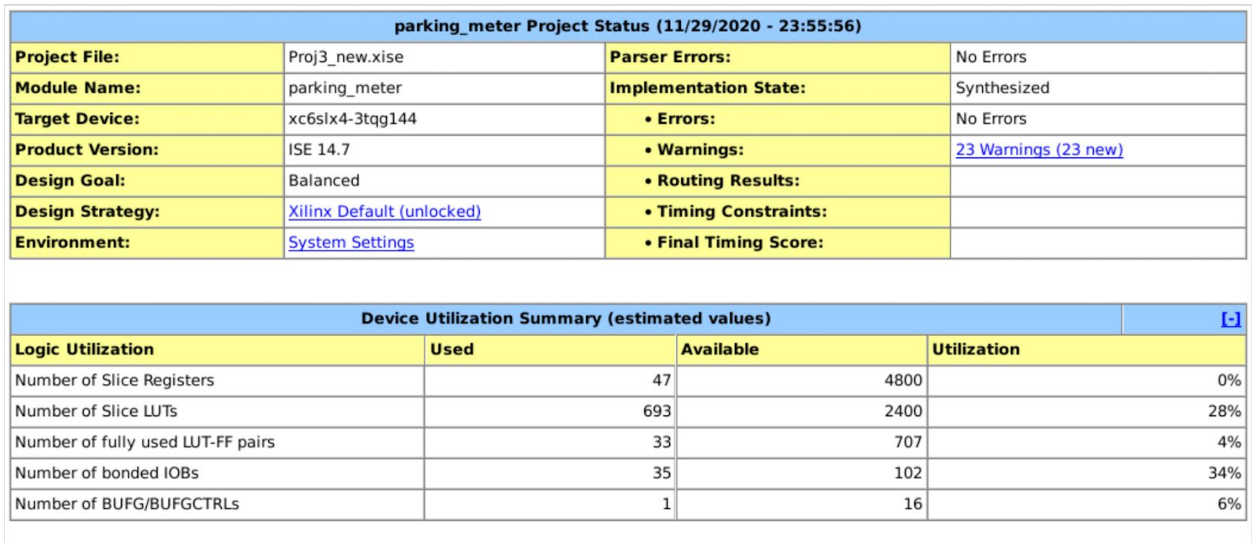






RTL Schematic:

From the RTL Schematic, it is obvious that the design is much more complex than either lab 1 or lab 2. It is also evident that my design is not very modular, since the schematic is huge and complex while only two of its blocks are for the BCDencoder and sevenseg submodules.



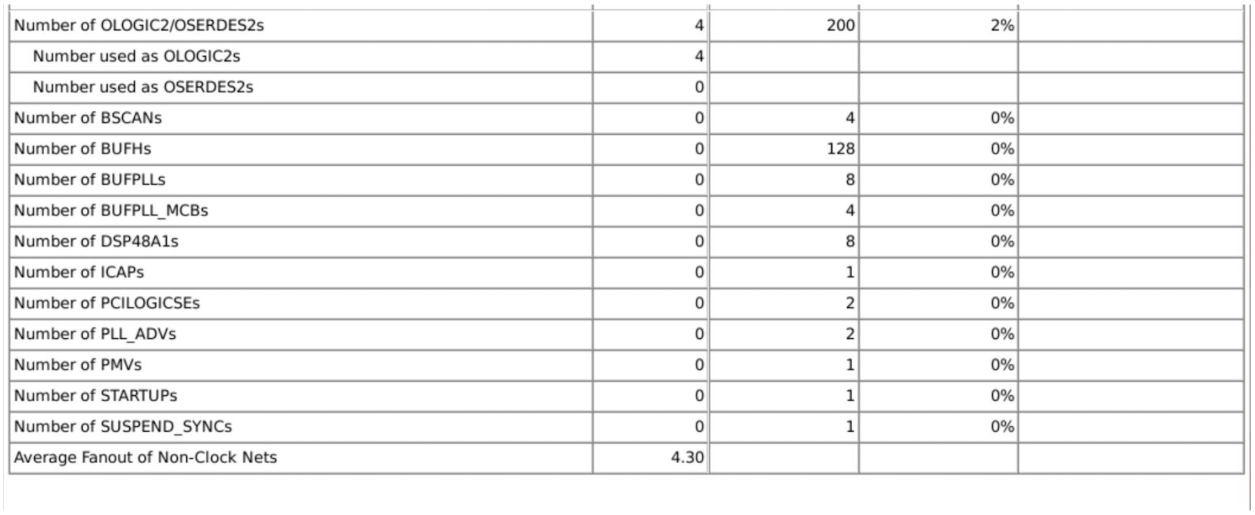
parking_meter Project Status (11/29/2020 - 23:55:56)			
Project File:	Proj3_new.xise	**Parser Errors:**	No Errors
Module Name:	parking_meter	**Implementation State:**	Synthesized
Target Device:	xc6slx4-3tqg144	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	23 Warnings (23 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values) [-]

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	47	4800	0%
Number of Slice LUTs	693	2400	28%
Number of fully used LUT-FF pairs	33	707	4%
Number of bonded IOBs	35	102	34%
Number of BUFG/BUFGCTRLs	1	16	6%

Device Utilization Summary [-]

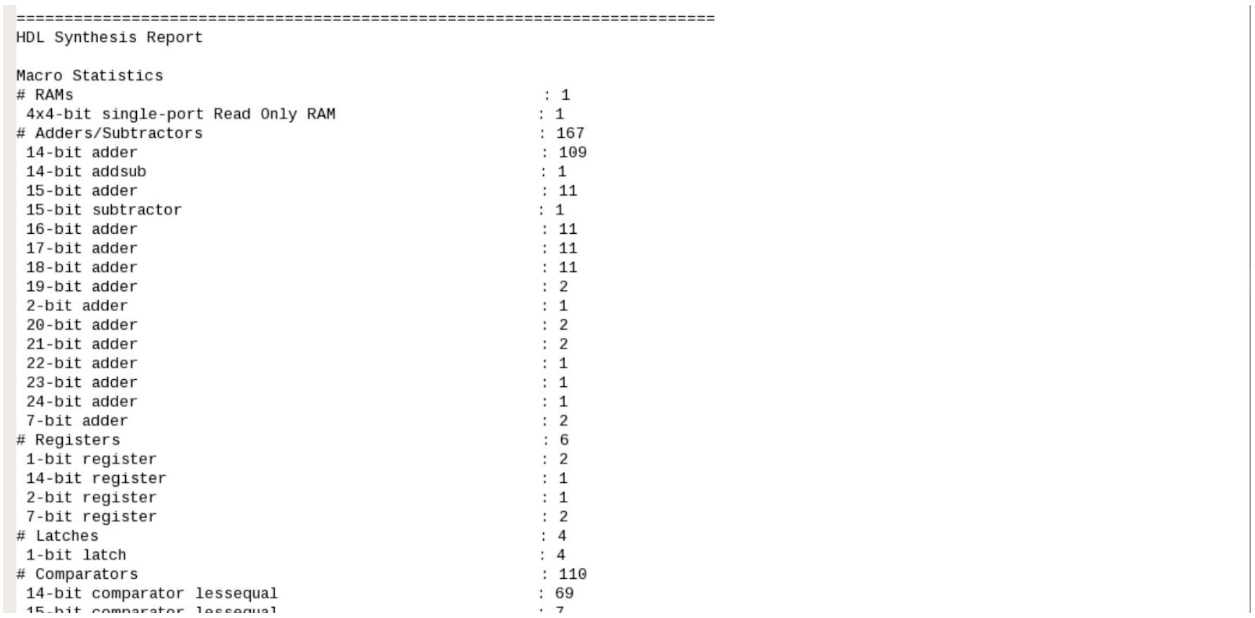
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	47	4,800	1%	
Number used as Flip Flops	47			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	639	2,400	26%	
Number used as logic	638	2,400	26%	
Number using O6 output only	482			
Number using O5 output only	0			
Number using O5 and O6	156			
Number used as ROM	0			
Number used as Memory	0	1,200	0%	
Number used exclusively as route-thrus	1			
Number with same-slice register load	0			
Number with same-slice carry load	1			
Number with other load	0			
Number of occupied Slices	236	600	39%	
Number of MUXCYs used	240	1,200	20%	
Number of LUT Flip Flop pairs used	644			
Number with an unused Flip Flop	601	644	93%	
Number with an unused LUT	5	644	1%	
Number of fully used LUT-FF pairs	38	644	5%	
Number of unique control sets	5			
Number of slice register sites lost to control set restrictions	17	4,800	1%	
Number of bonded IOBs	35	102	34%	
IOB Latches	4			
Number of RAMB16BWERs	0	12	0%	
Number of RAMB8BWERs	0	24	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	200	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	200	0%	



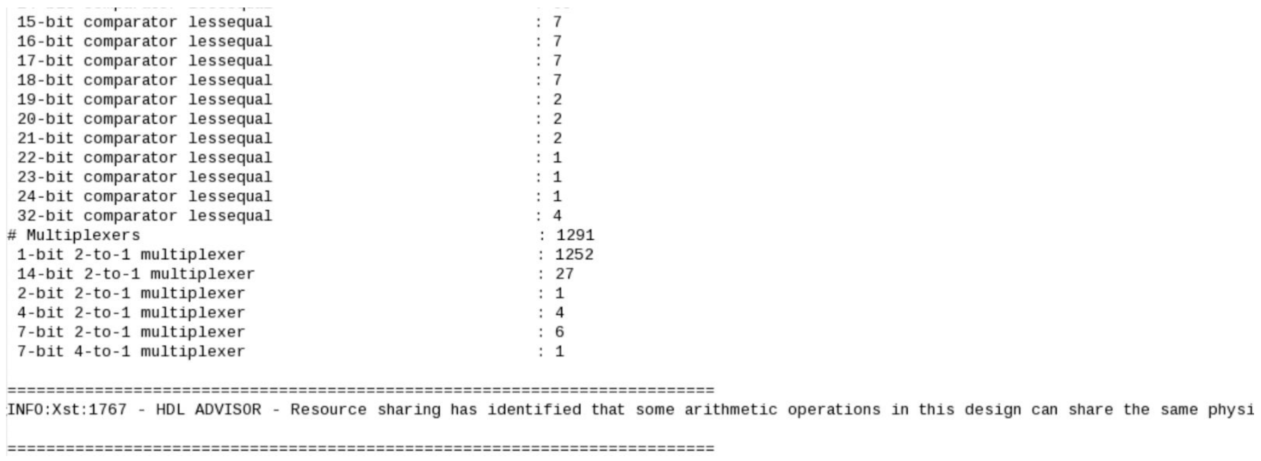
Number of OLOGIC2/OSERDES2s	4	200	2%	
Number used as OLOGIC2s	4			
Number used as OSERDES2s	0			
Number of BSCANs	0	4	0%	
Number of BUFHs	0	128	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	8	0%	
Number of ICAPs	0	1	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	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	4.30			

Design Summary Report:

The design was synthesized with no errors. Other than this crucial information, the design summary report gives us how many of each part was utilized for the synthesis of the design.



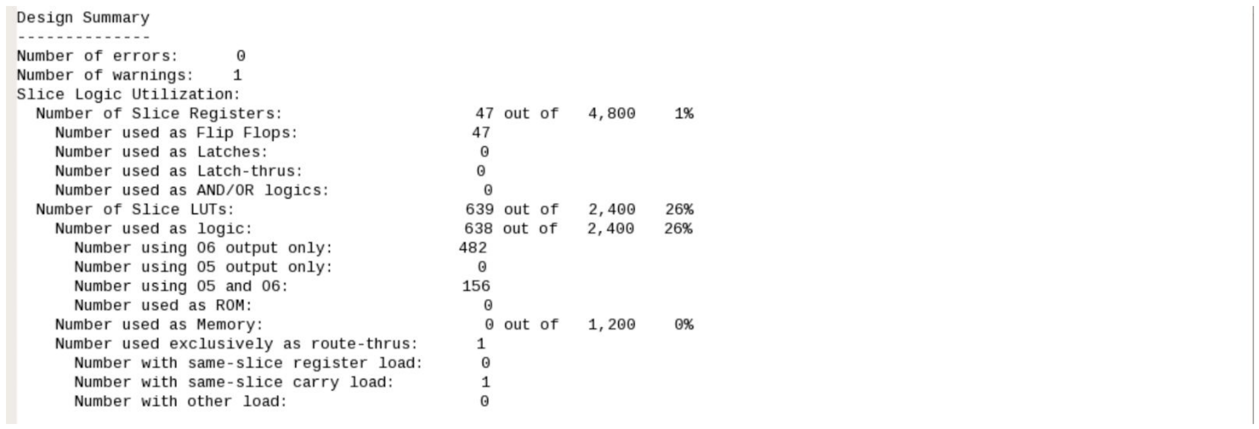
```
=========================================================================
HDL Synthesis Report

Macro Statistics
# RAMs                                                 : 1
 4x4-bit single-port Read Only RAM                     : 1
# Adders/Subtractors                                   : 167
 14-bit adder                                          : 109
 14-bit addsub                                         : 1
 15-bit adder                                          : 11
 15-bit subtractor                                     : 1
 16-bit adder                                          : 11
 17-bit adder                                          : 11
 18-bit adder                                          : 11
 19-bit adder                                          : 2
 2-bit adder                                           : 1
 20-bit adder                                          : 2
 21-bit adder                                          : 2
 22-bit adder                                          : 1
 23-bit adder                                          : 1
 24-bit adder                                          : 1
 7-bit adder                                           : 2
# Registers                                            : 6
 1-bit register                                        : 2
 14-bit register                                       : 1
 2-bit register                                        : 1
 7-bit register                                        : 2
# Latches                                              : 4
 1-bit latch                                           : 4
# Comparators                                          : 110
 14-bit comparator lessequal                           : 69
 15-bit comparator lessequal                           : 7
```



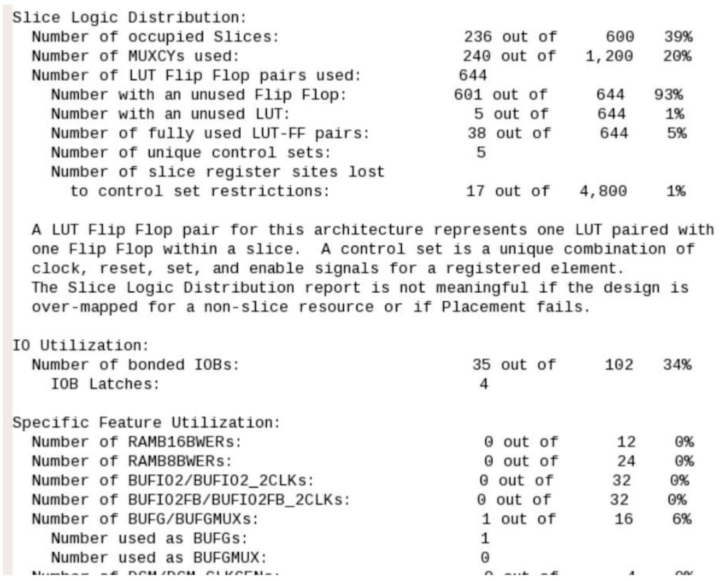
```
 15-bit comparator lessequal                    : 7
 16-bit comparator lessequal                    : 7
 17-bit comparator lessequal                    : 7
 18-bit comparator lessequal                    : 7
 19-bit comparator lessequal                    : 2
 20-bit comparator lessequal                    : 2
 21-bit comparator lessequal                    : 2
 22-bit comparator lessequal                    : 1
 23-bit comparator lessequal                    : 1
 24-bit comparator lessequal                    : 1
 32-bit comparator lessequal                    : 4
# Multiplexers                                  : 1291
 1-bit 2-to-1 multiplexer                       : 1252
 14-bit 2-to-1 multiplexer                      : 27
 2-bit 2-to-1 multiplexer                       : 1
 4-bit 2-to-1 multiplexer                       : 4
 7-bit 2-to-1 multiplexer                       : 6
 7-bit 4-to-1 multiplexer                       : 1

=========================================================================
INFO:Xst:1767 - HDL ADVISOR - Resource sharing has identified that some arithmetic operations in this design can share the same physi

=========================================================================
```

HDL Synthesis Report:

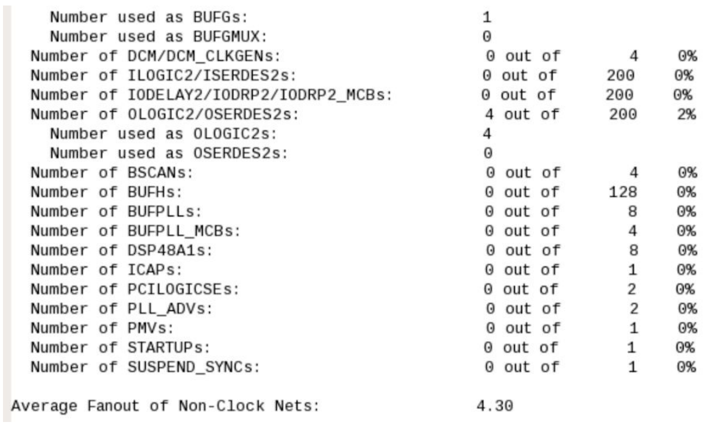
This report tells us which parts were utilized for the synthesis of this project. A large number of adders and subtractors are used, a total of 169, which is significantly larger than the number of adders and subtractors used for project 2. Unlike project 1 or 2, a 4x4 read only RAM is used. A ridiculous number of multiplexers are used, at a total of 1291.



```
Design Summary
--------------
Number of errors:      0
Number of warnings:    1
Slice Logic Utilization:
  Number of Slice Registers:                    47 out of   4,800    1%
    Number used as Flip Flops:                  47
    Number used as Latches:                      0
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                0
  Number of Slice LUTs:                        639 out of   2,400   26%
    Number used as logic:                      638 out of   2,400   26%
      Number using O6 output only:             482
      Number using O5 output only:               0
      Number using O5 and O6:                  156
      Number used as ROM:                        0
    Number used as Memory:                       0 out of   1,200    0%
    Number used exclusively as route-thrus:      1
      Number with same-slice register load:      0
      Number with same-slice carry load:         1
      Number with other load:                    0
```

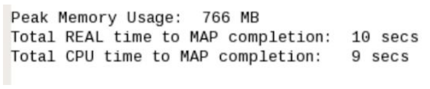


```
Slice Logic Distribution:
  Number of occupied Slices:                   236 out of     600   39%
  Number of MUXCYs used:                       240 out of   1,200   20%
  Number of LUT Flip Flop pairs used:          644
    Number with an unused Flip Flop:           601 out of     644   93%
    Number with an unused LUT:                   5 out of     644    1%
    Number of fully used LUT-FF pairs:          38 out of     644    5%
    Number of unique control sets:               5
    Number of slice register sites lost
      to control set restrictions:              17 out of   4,800    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     102   34%
    IOB Latches:                                 4

Specific Feature Utilization:
  Number of RAMB16BWERs:                         0 out of      12    0%
  Number of RAMB8BWERs:                          0 out of      24    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 used as BUFGs:                        1
    Number used as BUFGMUX:                      0
```

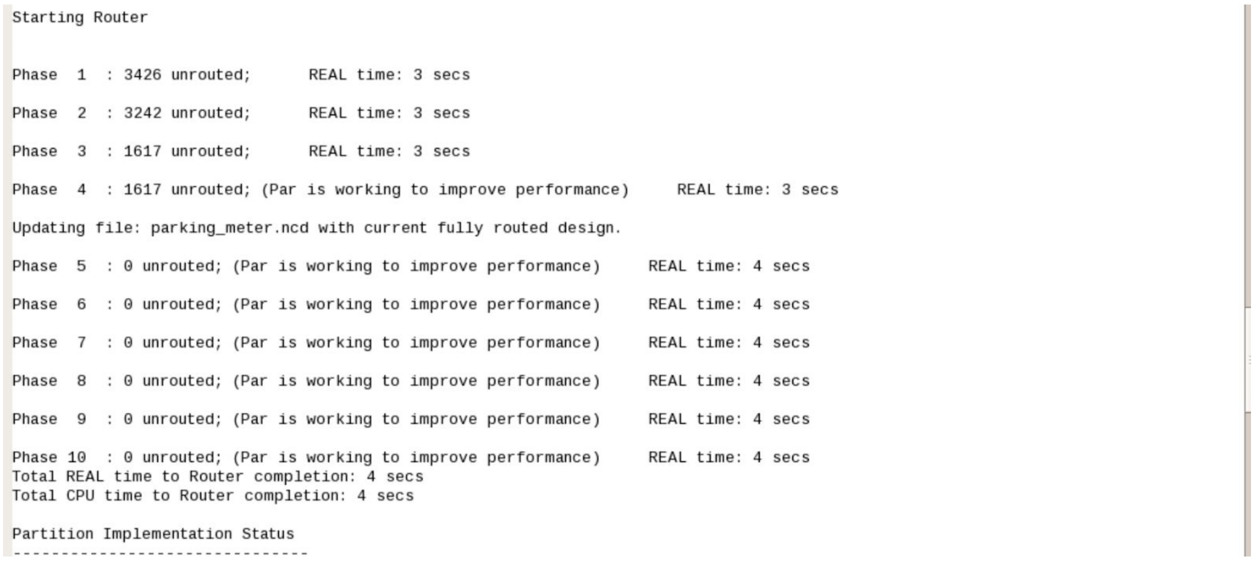
```
  Number of DCM/DCM_CLKGENs:                     0 out of       4    0%
  Number of ILOGIC2/ISERDES2s:                   0 out of     200    0%
  Number of IODELAY2/IODRP2/IODRP2_MCBs:         0 out of     200    0%
  Number of OLOGIC2/OSERDES2s:                   4 out of     200    2%
    Number used as OLOGIC2s:                     4
    Number used as OSERDES2s:                    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       8    0%
  Number of ICAPs:                               0 out of       1    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:              4.30
```



```
Peak Memory Usage:  766 MB
Total REAL time to MAP completion:  10 secs
Total CPU time to MAP completion:   9 secs
```

Map Report:

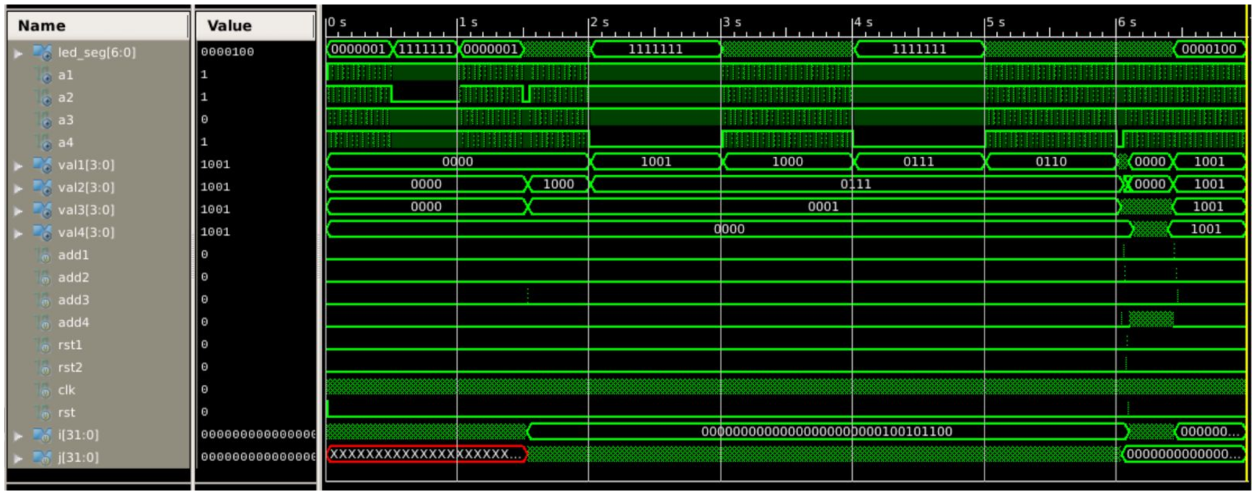
While other information is similar to that given by the HDL Synthesis Report and Design Summary Report, the Map Report uniquely identifies the peak memory usage and the total REAL time to MAP completion. The total REAL time to completion is greater by 1 second compared to that of project 2, which is not that great of a difference as I anticipated. A ridiculous number of slice LUTs and flip-flops were used in this design.



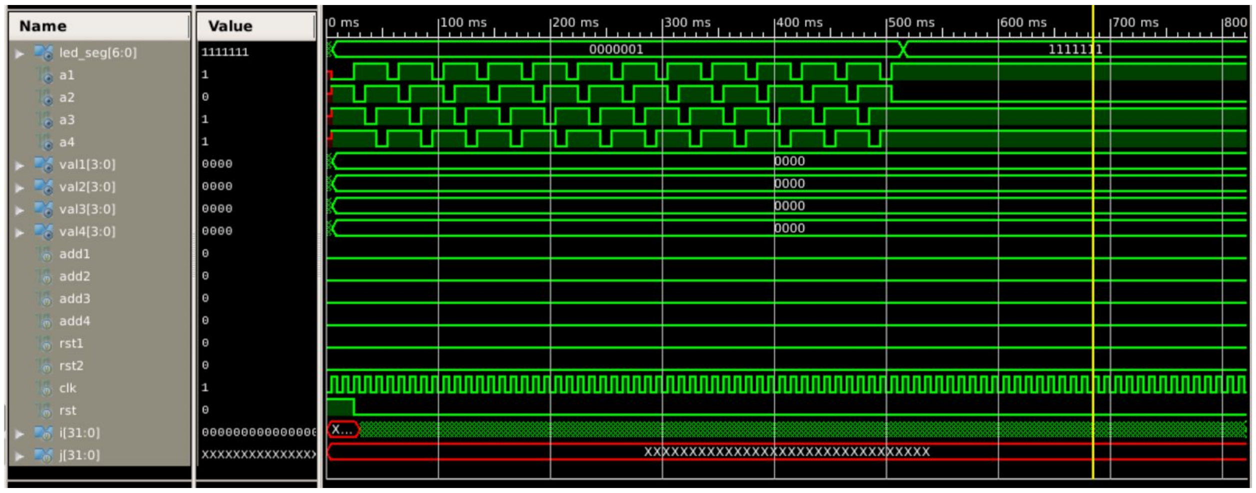
```
Starting Router


Phase  1  : 3426 unrouted;      REAL time: 3 secs

Phase  2  : 3242 unrouted;      REAL time: 3 secs

Phase  3  : 1617 unrouted;      REAL time: 3 secs

Phase  4  : 1617 unrouted; (Par is working to improve performance)     REAL time: 3 secs

Updating file: parking_meter.ncd with current fully routed design.

Phase  5  : 0 unrouted; (Par is working to improve performance)     REAL time: 4 secs

Phase  6  : 0 unrouted; (Par is working to improve performance)     REAL time: 4 secs

Phase  7  : 0 unrouted; (Par is working to improve performance)     REAL time: 4 secs

Phase  8  : 0 unrouted; (Par is working to improve performance)     REAL time: 4 secs

Phase  9  : 0 unrouted; (Par is working to improve performance)     REAL time: 4 secs

Phase 10  : 0 unrouted; (Par is working to improve performance)     REAL time: 4 secs
Total REAL time to Router completion: 4 secs
Total CPU time to Router completion: 4 secs

Partition Implementation Status
-------------------------------
```

Place and Route Report:

While the other information given by the Place and Route Report overlaps with the HDL Synthesis Report, the Design Summary Report, and the Map Report, the Place and Routee Report alone gives us information about the phases taken to route, and the REAL time it took. For this specific design, routing was completed in 4 phases, and 12 seconds REAL time.



// zoomed out



// zoomed in

Simulation Waveforms:

Above is a sample simulation waveform to show how the outputs look under certain circumstances, but does not show all of the test cases and results. I had to run the simulation for a total of 7 seconds to gather all the results, since this fsm runs with a very low frequency clock.

Test Cases and Results				
INPUT	TIME LEFT	FLASH STATUS	LED_SEG	PURPOSE
nothing	0000	0.5s on 0.5s off	000_0001	to observe flashing at 0000 T=1
add3	0180	N/A	000_0001 for a1 and a4, 000_0000 for a2, 100_1111 for a3	add3 functionality (+180)
nothing	0176	1s on 1s off	corresponding values	to observe flashing at below 0180 T=2
add4	0476	none	corresponding values	add4 functionality (+300)
add1	0536	none	corresponding values	add1 functionality (+60)
add2	0656	none	corresponding values	add2 func. (+120)
rst2	0150	1s on 1s off (but cannot be observed here, moved on to rst1 at next clkedge)	corresponding values	rst2 functionality (reset to 0150)
rst1	0016	same as above	corresponding values	rst1 functionality (reset to 0016)
rst	0000	0.5s on 0.5s off (cannot be observed here, moved on)	corresponding values	rst functionality (reset to 0000)
add4 34 times	9999	none	corresponding values	maximum 9999 cutoff when adding too much
add1	9999	none	corresponding values	maximum cutoff for each add input
add2	9999	none	corresponding values	maximum cutoff for each add input
add3	9999	none	corresponding values	maximum cutoff for each add input

Conclusion

The design of parking_meter includes 2 submodules for turning binary into binary coded decimal, and for outputting the correct led_seg and a1 through a4 values for each number. Both of them heavily depend on the time left, or bin_time in our design. The main module of parking_meter has the heavy task of figuring out how much time is left based on given inputs, and the task of providing the sevenseg submodule with the correct flash status based on the time left.

The most difficult part of this project was figuring out how much time was left, and how to determine the flash status based on the time left. State transitions were difficult since neither project 1 nor 2 had anything to do with states. I was able to figure out the states by completely redoing my design halfway through, from an fsm that depended on inputs for transitions to an fsm that depended on the time left for transitions. The three always block setup given in the spec did not end up becoming very helpful for this project.

While the BCDencoder and sevenseg modules were well explained during sections, the matter of states and the fsm was barely touched during sections. I think it would help students with the design to spend more time on the state transitions and how to implement them rather than on the led display or BCDencoding.