Lab 3: Finite State Machine: Vending Machine

Ethan Ngo

TA: Ananya Ravikumar (Section 3)

May 23, 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 vending machine. 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 vending machine with the following characteristics:

1. The vending machine has 10 different snacks for sale. Each snack has a two-digit code (10-14, 20-24).
2. Each snack is stored in a separate slot. There can be up to 10 units of the snack stored in 1 slot. A counter for every slot keeps track of the number of units of the snack remaining in that slot.
3. A buyer can purchase only 1 item at a time.
4. The machine only accepts payment by card.

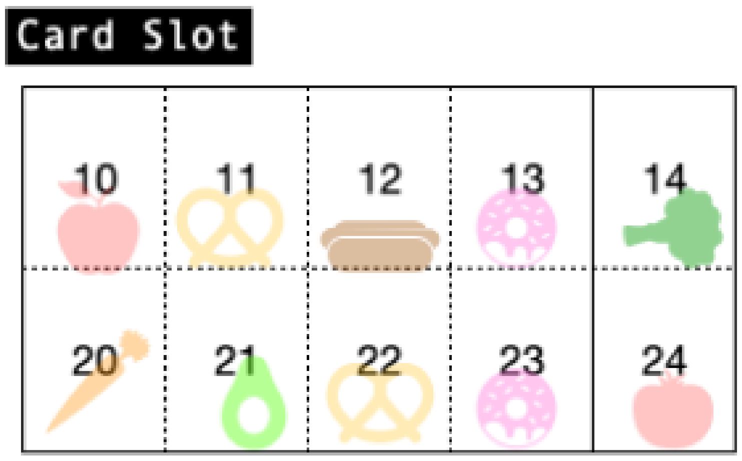


Figure 1: Vending Machine Display

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 and outputs are required to be.

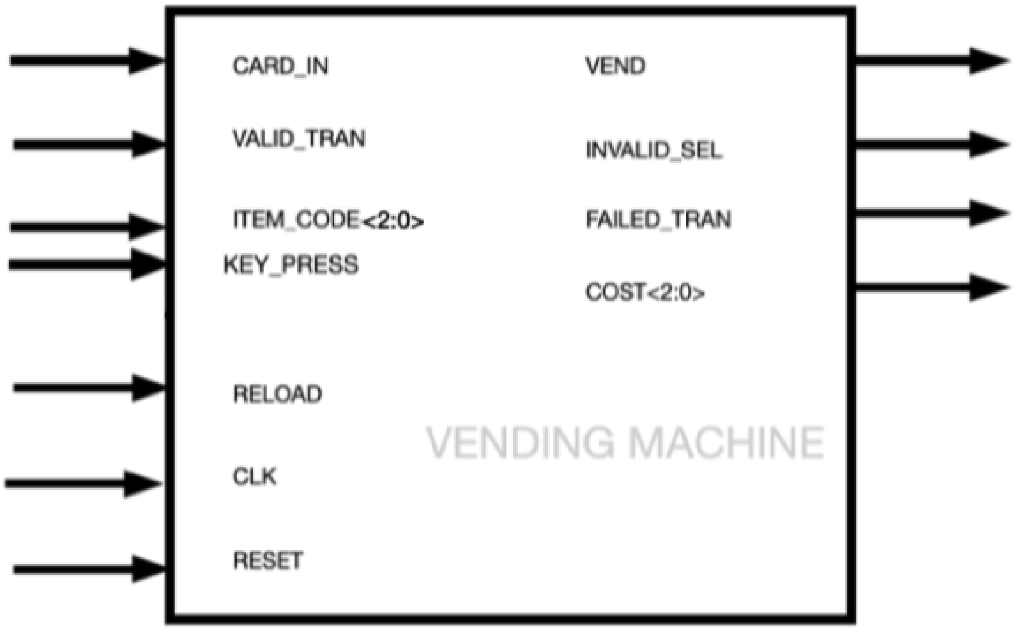


Figure 2: Vending Machine Schematic Overview

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 stRst signal is 1, we set all values 0. When it is 0, we are put in the stIdle state, which waits for inputs such as CARD_IN or stReload. Note that stRst takes precedence over all signals.
- When stReload signal is 1, the count for each slot item is set back to their maximum, which is 10. Afterwards, we are set the state to stIdle.
- While in the stIdle state, if the CARD_IN signal goes to 1, we begin the process of getting inputs for "ordering" in the stCardInsertWait state.

- In the stCardInsertWait state, we can do two things. First is that, after 5 cycles, if no input is given, we go back to the stIdle state. However, if the KEY_PRESS signal is 1, we take the ITEM_CODE input and store that. In this same state, we wait for another KEY_PRESS signal for another ITEM_CODE input to complete our 2-digit selection.
- If the input is invalid, we transition to the stInvalidIn, which then goes back to our stIdle state.
- If the input is valid, we go to the stValidIn state. In this state, we set the COST to the proper value and wait for a VALID_TRAN input to be 1. If it is 1, then we transition to the VEND state, as the payment has been accepted by the vending machine. If it is 0 for 5 cycles, we go to the stFailedTran state.
- In the stFailedTran state, we set the FAILED_TRAN output to 1 and go back to the stIdle state.
- In the stVend state, the COST of the requested item is output and set the VEND signal to 1, before setting the state back to stIdle after a cycle.

To implement this in Verilog, I created a vending_machine 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 4-bit registers to store the count for each item slot in the vending machine.

In Verilog, I utilize always blocks to first update the current_state, turn on the 5-cycle timer, or increment the 5-cycle timer. Next, the next_state is updated, depending on the current state and inputs. Lastly, I determine what the output should be based on the previous always blocks and output the necessary values.

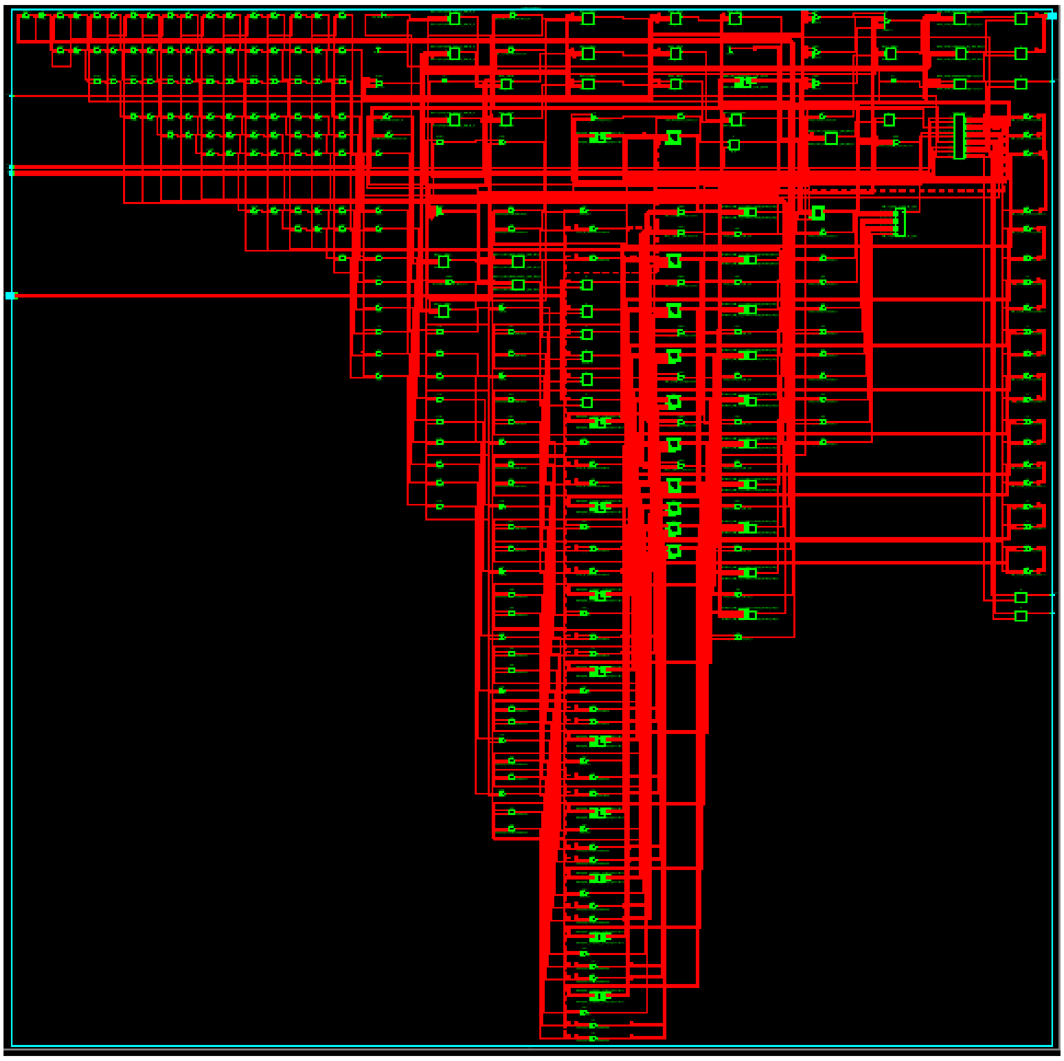


Figure 3: vending_machine Schematic

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 I use, which are updated by counters and manipulated with muxes to get the proper vaules I desire.

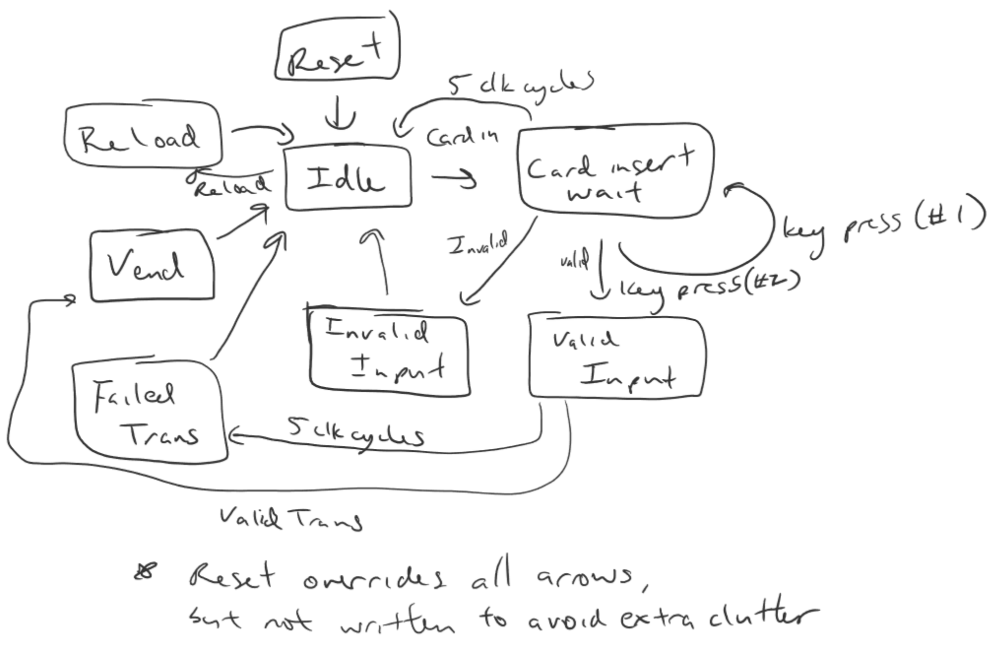
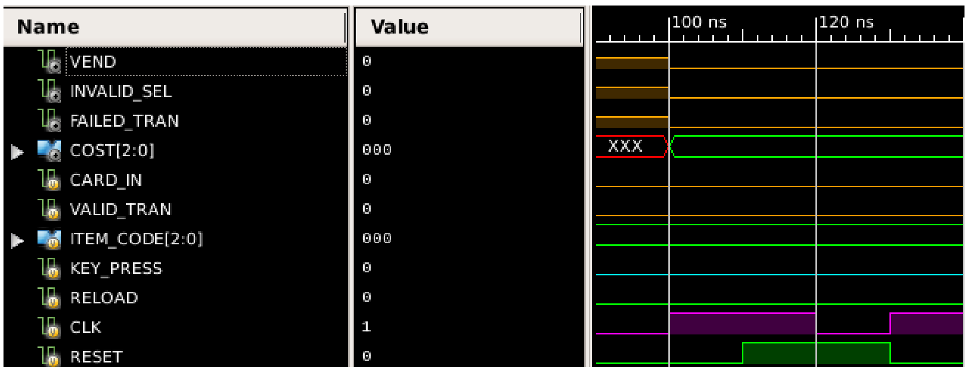


Figure 4: vending_machine Schematic Hand-Drawn and Simplified

3 Test-bench Simulation

To test my modules, I tested my main module vending_machine 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.

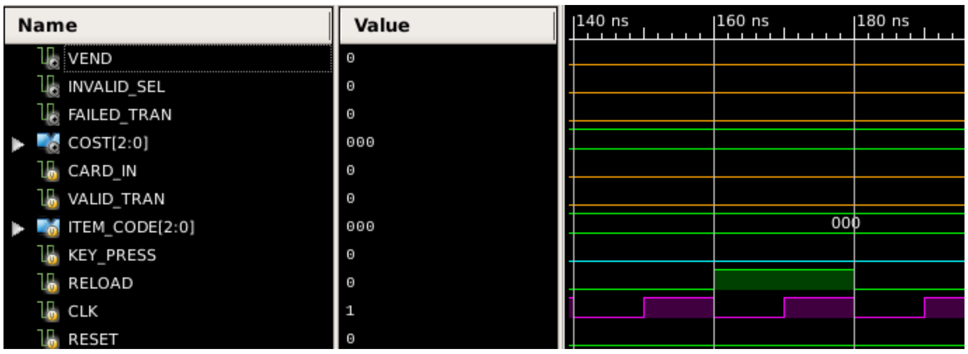
1. Reset Test Waveform: This test simply tested to make sure RESET put all outputs to 0.



Name	Value
VEND	0
INVALID_SEL	0
FAILED_TRAN	0
COST[2:0]	000
CARD_IN	0
VALID_TRAN	0
ITEM_CODE[2:0]	000
KEY_PRESS	0
RELOAD	0
CLK	1
RESET	0

Figure 5: vending_machine Reset Test Waveform

2. Reload Test Waveform: This test simply tested to make sure we were able to reload, although this test doesn't affect output, so the waveform is quite trivial.



Name	Value
VEND	0
INVALID_SEL	0
FAILED_TRAN	0
COST[2:0]	000
CARD_IN	0
VALID_TRAN	0
ITEM_CODE[2:0]	000
KEY_PRESS	0
RELOAD	0
CLK	1
RESET	0

Figure 6: vending_machine Reload Test Waveform

3. Normal Test 1 Waveform: This tested to ensure that the normal case worked properly. This one dealt with a item that had a cost of 2, which is properly done, as

shown in the image.

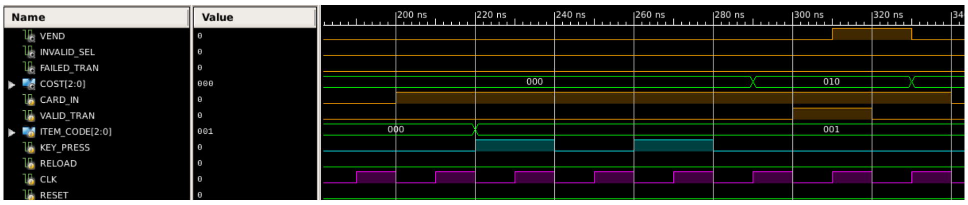


Figure 7: vending_machine Normal Test 1 Waveform

4. Normal Test 2 Waveform: This tested to ensure that the normal case worked properly. This one dealt with a item that had a cost of 5, which is properly done, as shown in the image.

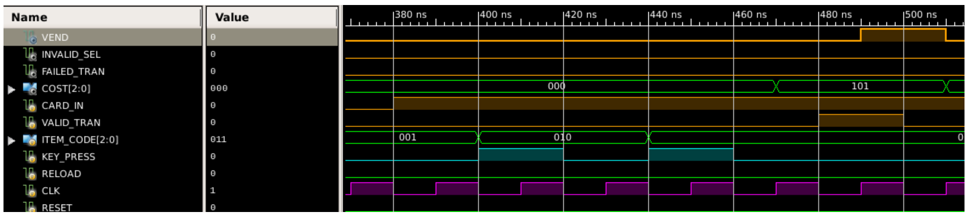


Figure 8: vending_machine Normal Test 2 Waveform

5. Double Transaction Test Waveform: This tests a transaction stream where CARD_IN stays high through an entire transaction, which would lead to the allowance of another transaction, which is properly done, as shown in the image with 2 high VEND signals.

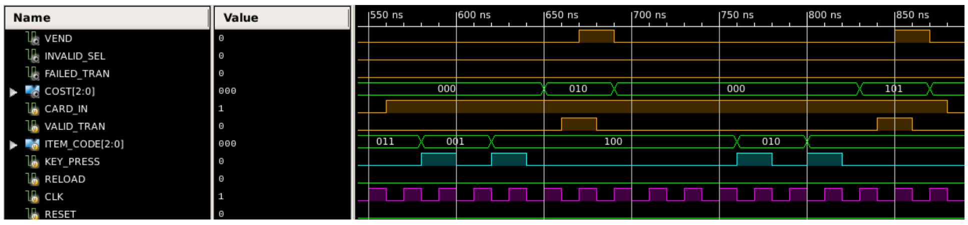


Figure 9: vending_machine Double Transaction Test Waveform

6. Card W/ No Input Test Waveform: This tests that when CARD_IN is high, that

after 5 cycles, it will make INVALID_SEL high for a cycle and then return to idle.

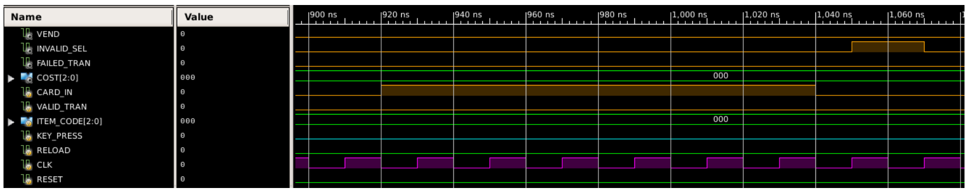


Figure 10: vending_machine Card W/ No Input Test Waveform

7. Card W/ 1 Input Test Waveform: This tests that when CARD_IN is high and one ITEM_CODE is inputted, that after 5 cycles, it will make INVALID_SEL high for a cycle and then return to idle.

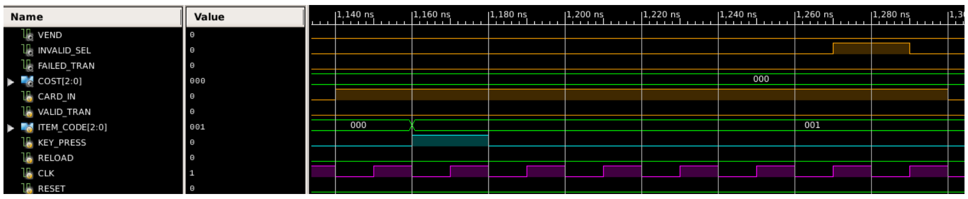


Figure 11: vending_machine Card W/ 1 Input Test Waveform

8. Invalid Code Test Waveform: This test checks to see that when an invalid code (i.e a code that isn't assigned to an item slot), we will get an INVALID_SEL high signal and then return to idle.

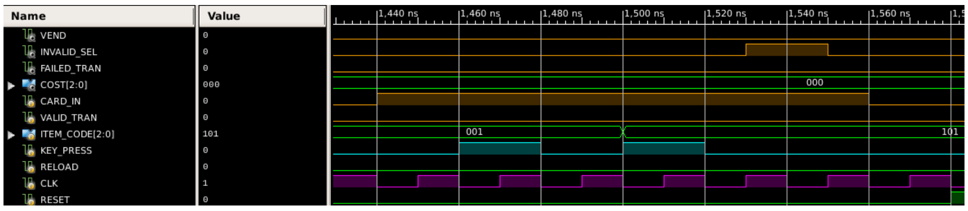


Figure 12: vending_machine Invalid Code Test Waveform

9. Failed Transaction Test Waveform: This test checks to see that when the vending machine is has all the valid inputs and is waiting more than 5 cycles for a

VALID_TRAN signal to go high, it will make the FAILED_TRAN signal go high and go back to idle.

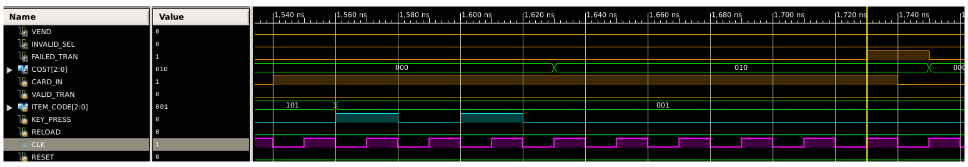


Figure 13: vending_machine Failed Transaction Test Waveform

10. Empty Machine Test Waveform: This test checks to see that when the vending machine is empty and we try to order something, that we will get an INVALID_SEL high signal because there are no more of that item to order.

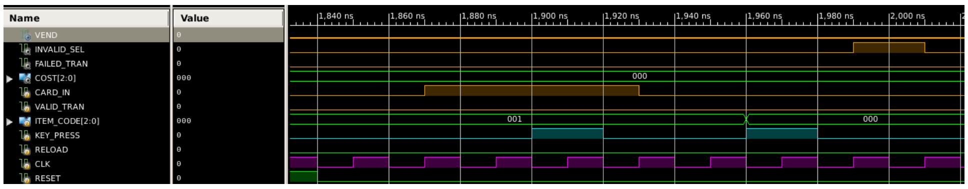
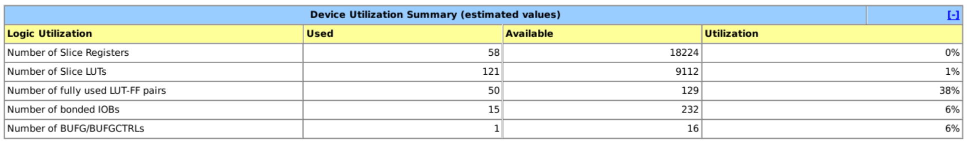


Figure 14: vending_machine Empty Machine Test Waveform

4 Reports

4.1 ISE Design Overview Summary Report

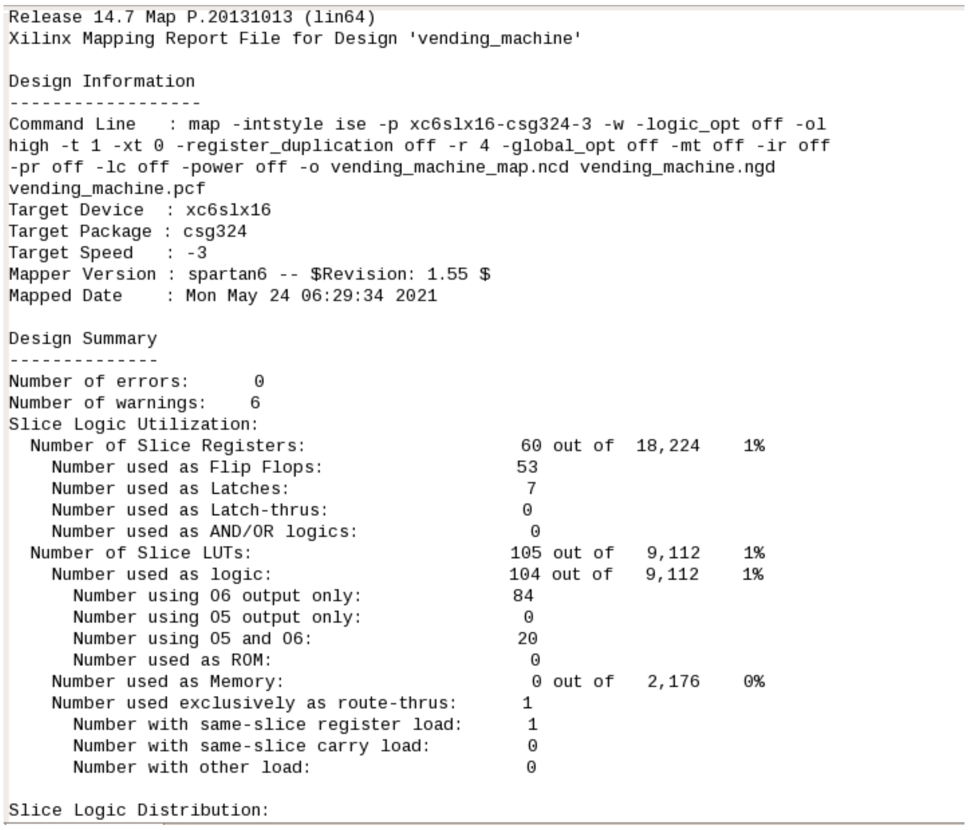


Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	58	18224	0%
Number of Slice LUTs	121	9112	1%
Number of fully used LUT-FF pairs	50	129	38%
Number of bonded IOBs	15	232	6%
Number of BUFG/BUFGCTRLs	1	16	6%

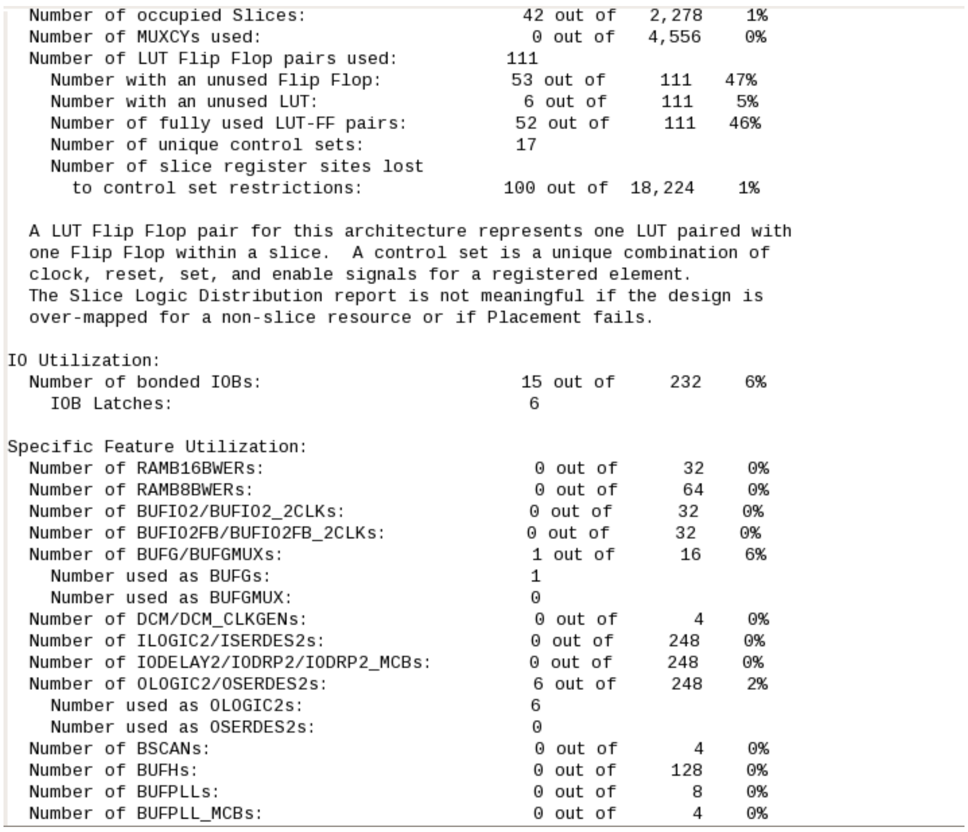
Figure 15: Design Summary

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 were utilized, which is as expected.

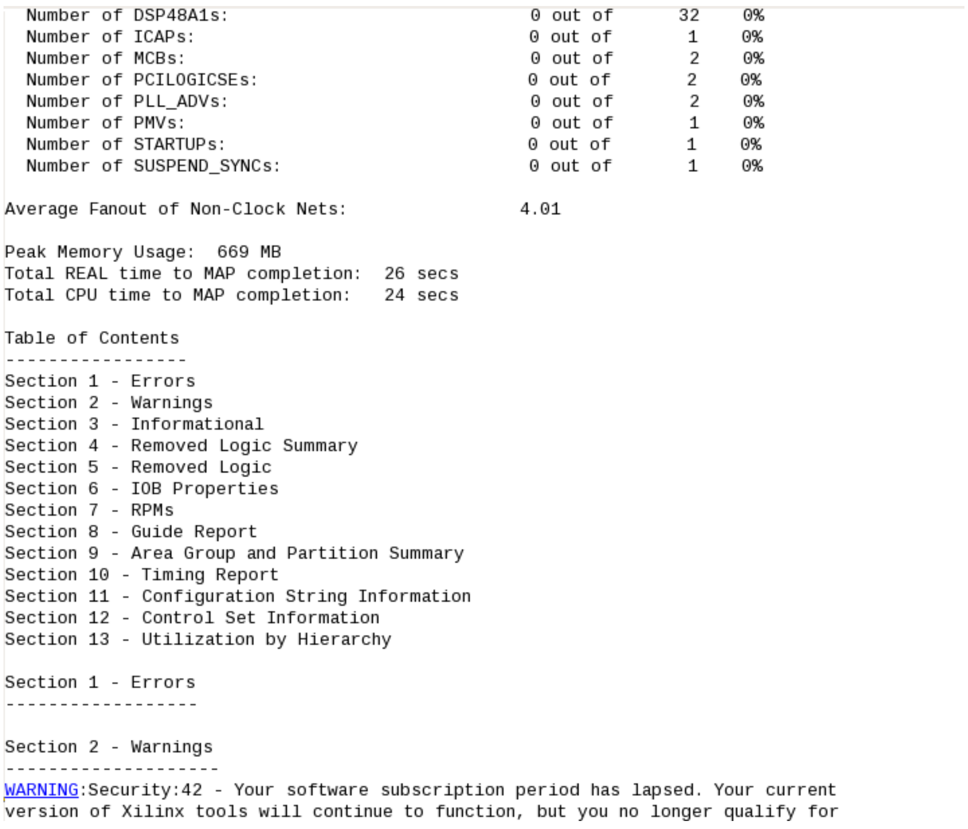
4.2 Map Report



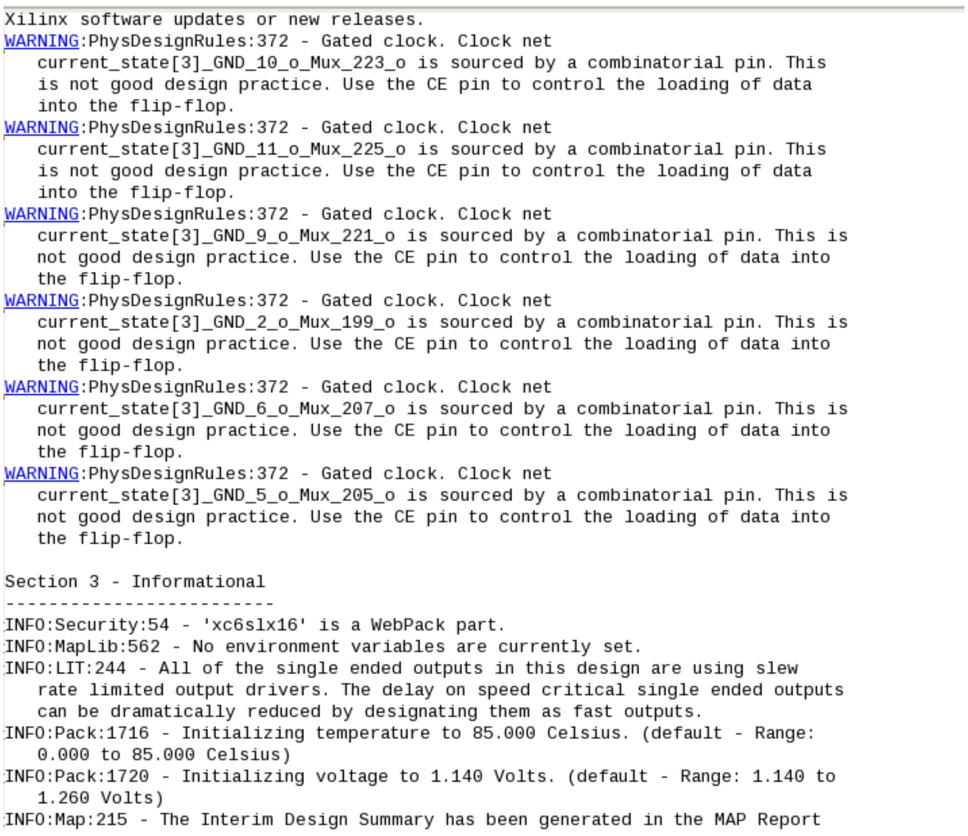
```
Release 14.7 Map P.20131013 (lin64)
Xilinx Mapping Report File for Design 'vending_machine'

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 vending_machine_map.ncd vending_machine.ngd
vending_machine.pcf
Target Device  : xc6slx16
Target Package : csg324
Target Speed   : -3
Mapper Version : spartan6 -- $Revision: 1.55 $
Mapped Date    : Mon May 24 06:29:34 2021

Design Summary
--------------
Number of errors:      0
Number of warnings:    6
Slice Logic Utilization:
  Number of Slice Registers:                    60 out of  18,224    1%
    Number used as Flip Flops:                  53
    Number used as Latches:                      7
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                0
  Number of Slice LUTs:                        105 out of   9,112    1%
    Number used as logic:                      104 out of   9,112    1%
      Number using O6 output only:              84
      Number using O5 output only:               0
      Number using O5 and O6:                   20
      Number used as ROM:                        0
    Number used as Memory:                       0 out of   2,176    0%
    Number used exclusively as route-thrus:      1
      Number with same-slice register load:      1
      Number with same-slice carry load:         0
      Number with other load:                    0

Slice Logic Distribution:
```



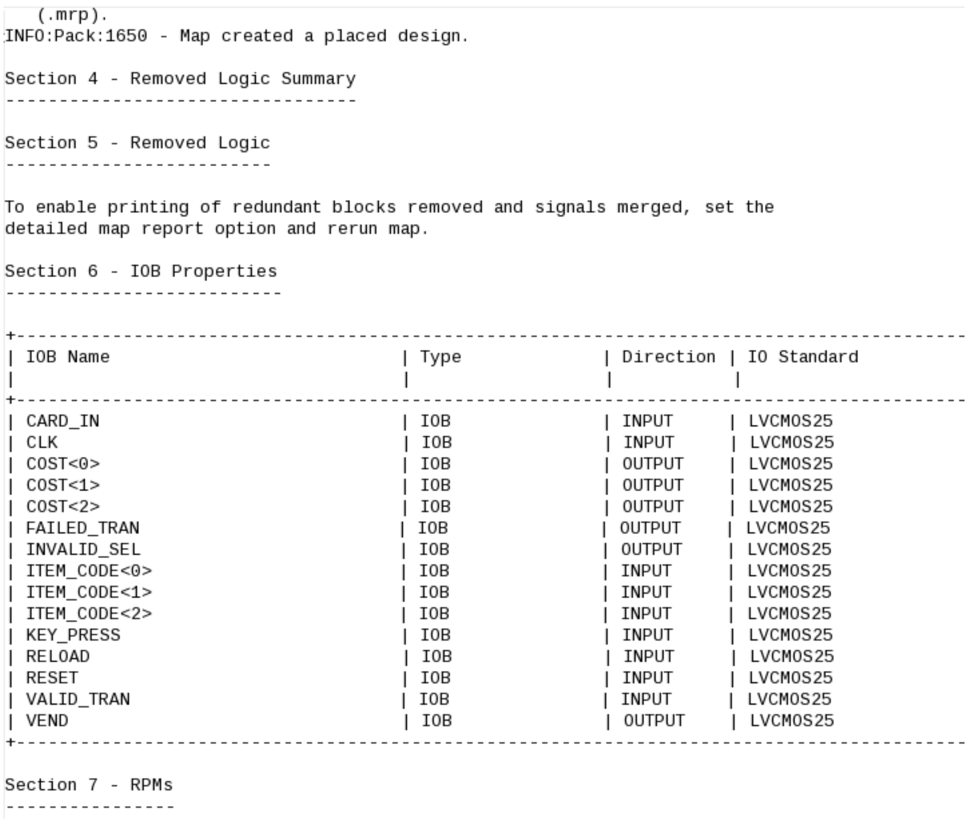
```
  Number of occupied Slices:                    42 out of   2,278    1%
  Number of MUXCYs used:                         0 out of   4,556    0%
  Number of LUT Flip Flop pairs used:          111
    Number with an unused Flip Flop:            53 out of     111   47%
    Number with an unused LUT:                   6 out of     111    5%
    Number of fully used LUT-FF pairs:          52 out of     111   46%
    Number of unique control sets:              17
    Number of slice register sites lost
      to control set restrictions:             100 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:                        15 out of     232    6%
    IOB Latches:                                 6

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:                   6 out of     248    2%
    Number used as OLOGIC2s:                     6
    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%
```



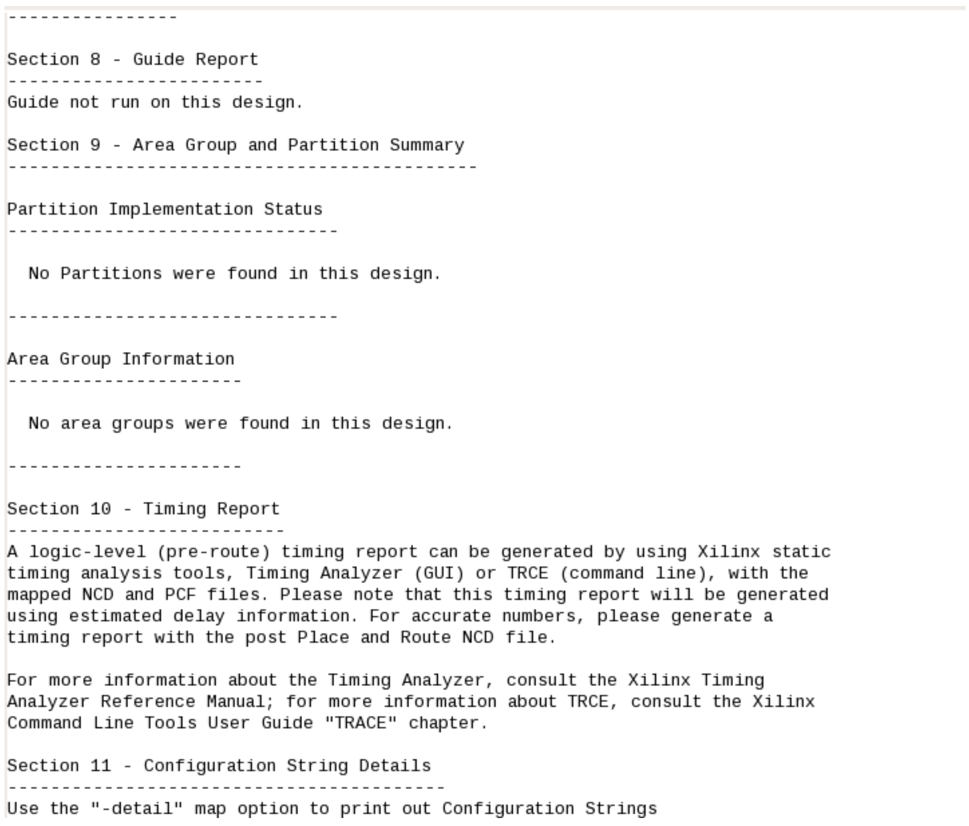
```
  Number of DSP48A1s:                            0 out of      32    0%
  Number of ICAPs:                               0 out of       1    0%
  Number of MCBs:                                0 out of       2    0%
  Number of PCILOGICSEs:                         0 out of       2    0%
  Number 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:                4.01

Peak Memory Usage:  669 MB
Total REAL time to MAP completion:  26 secs
Total CPU time to MAP completion:   24 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
```



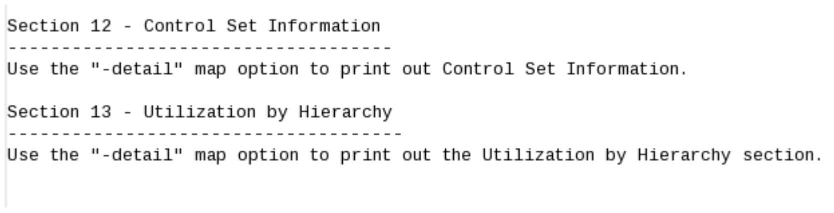
```
Xilinx software updates or new releases.
WARNING:PhysDesignRules:372 - Gated clock. Clock net
   current_state[3]_GND_10_o_Mux_223_o is sourced by a combinatorial pin. This
   is not good design practice. Use the CE pin to control the loading of data
   into the flip-flop.
WARNING:PhysDesignRules:372 - Gated clock. Clock net
   current_state[3]_GND_11_o_Mux_225_o is sourced by a combinatorial pin. This
   is not good design practice. Use the CE pin to control the loading of data
   into the flip-flop.
WARNING:PhysDesignRules:372 - Gated clock. Clock net
   current_state[3]_GND_9_o_Mux_221_o is sourced by a combinatorial pin. This is
   not good design practice. Use the CE pin to control the loading of data into
   the flip-flop.
WARNING:PhysDesignRules:372 - Gated clock. Clock net
   current_state[3]_GND_2_o_Mux_199_o is sourced by a combinatorial pin. This is
   not good design practice. Use the CE pin to control the loading of data into
   the flip-flop.
WARNING:PhysDesignRules:372 - Gated clock. Clock net
   current_state[3]_GND_6_o_Mux_207_o is sourced by a combinatorial pin. This is
   not good design practice. Use the CE pin to control the loading of data into
   the flip-flop.
WARNING:PhysDesignRules:372 - Gated clock. Clock net
   current_state[3]_GND_5_o_Mux_205_o is sourced by a combinatorial pin. This is
   not good design practice. Use the CE pin to control the loading of data into
   the flip-flop.

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

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

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
|                                    |                  |            |
+---------------------------------------------------------------------------------------------
| CARD_IN                            | IOB              | INPUT      | LVCMOS25
| CLK                                | IOB              | INPUT      | LVCMOS25
| COST<0>                            | IOB              | OUTPUT     | LVCMOS25
| COST<1>                            | IOB              | OUTPUT     | LVCMOS25
| COST<2>                            | IOB              | OUTPUT     | LVCMOS25
| FAILED_TRAN                        | IOB              | OUTPUT     | LVCMOS25
| INVALID_SEL                        | IOB              | OUTPUT     | LVCMOS25
| ITEM_CODE<0>                       | IOB              | INPUT      | LVCMOS25
| ITEM_CODE<1>                       | IOB              | INPUT      | LVCMOS25
| ITEM_CODE<2>                       | IOB              | INPUT      | LVCMOS25
| KEY_PRESS                          | IOB              | INPUT      | LVCMOS25
| RELOAD                             | IOB              | INPUT      | LVCMOS25
| RESET                              | IOB              | INPUT      | LVCMOS25
| VALID_TRAN                         | IOB              | INPUT      | LVCMOS25
| VEND                               | 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, although not anywhere near the maximum number of registers 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.

5 Conclusion

In this lab, I designed a FSM vending machine, based on a Moore machine. I implemented a wide variety of topics discussed in lecture, including the use of parameters and a switch-case statement.

There were quite a few difficulties that I ran into. The more prominent ones included trying to get the 5 clock cycles to line up properly. How I implemented my code, I actually had to take into consideration the time to process the changes in signals after, say, a KEY_PRESS signal went low. In addition, I wasn't sure how to implement the steady clock in this code. I feel that I could have, but instead, I just had a bunch of `#10` delays, which made timing too difficult.