Project 4: Vending Machine

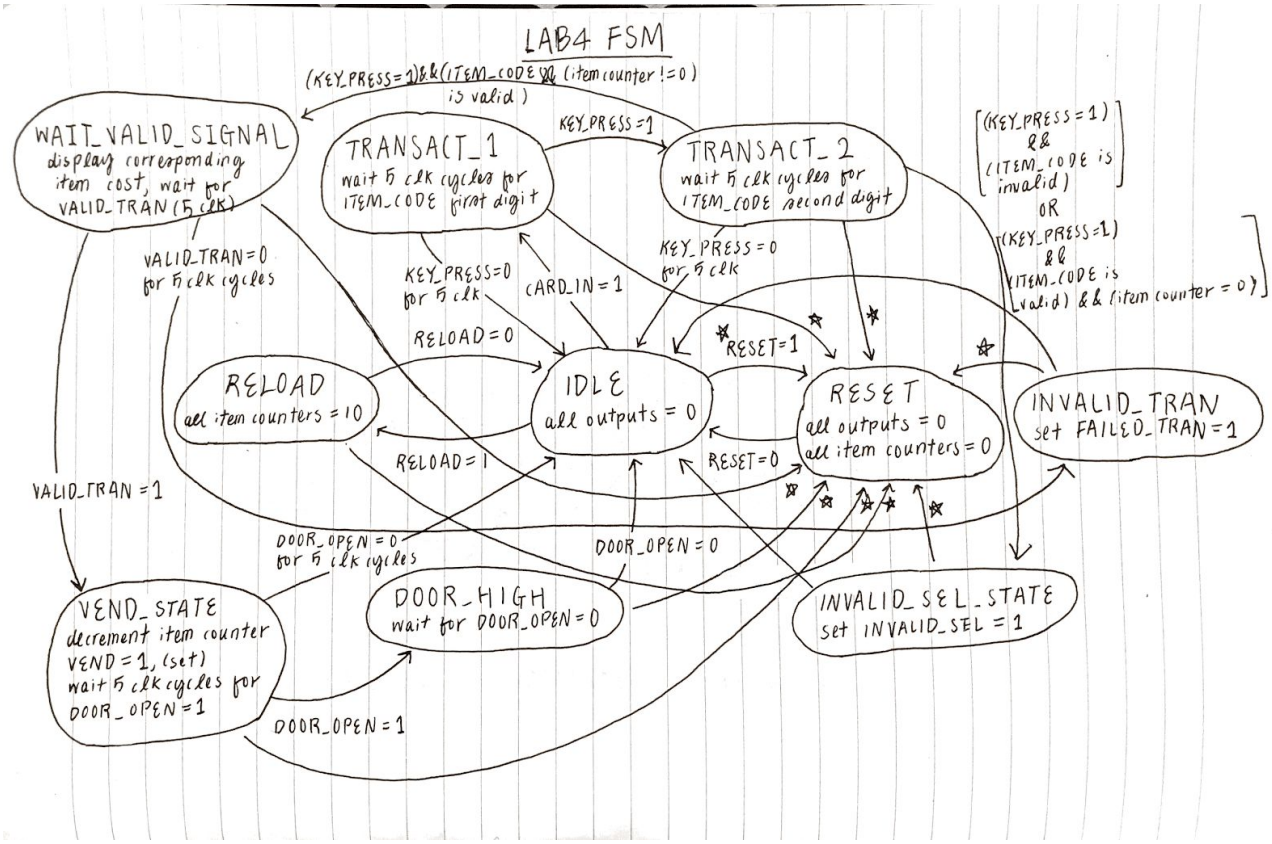
Haley Kim
405111152

Introduction and Requirement

The vending machine for this project is a Moore machine whose output only relies on the current state of the machine, regardless of the input fed in. The inputs help determine the current state, and the current state determines the output. The vending machine needs to be able to handle each input and adjust its four outputs accordingly in order to represent the state that the machine is in.

This project has eight inputs it must recognize and handle:

1. CARD_IN: remains high as long as card is inserted
2. VALID_TRAN: signal displaying validity of card
3. ITEM_CODE: 4 bit signal for one digit of the item code
4. KEY_PRESS: ITEM_CODE valid for reading when signal is high
5. DOOR_OPEN: high to indicate an open door
6. RELOAD: to reload machine when high
7. CLK: 100 MHz clock with 10ns period
8. RESET: to reset machine when high



High-Level Explanation of FSM Diagram:

The vending machine must be able to recognize that a card was inserted, then wait for the user's input for the two digits of the item code. If the item code is not received, the machine stops the transaction and returns to its idle state. If the received item code is valid and that item is in stock, the machine must wait for the VALID_TRAN signal to come from the bank. If the signal is not received, the machine must indicate that the transaction failed. If the signal is received, the machine must vend and decrement the item counter. If the door does not open, the machine returns to the idle state. If the door opens, the machine must wait for the door to close again before returning to the idle state.

Design Description

The main overarching module is vending_machine, with inputs:

CARD_IN,
VALID_TRAN,
[3:0] ITEM_CODE,
KEY_PRESS,
DOOR_OPEN,
RELOAD,
CLK,
RESET,

and outputs:

VEND,
INVALID_SEL,
FAILED_TRAN,
[2:0] COST

Main module vending_machine does not call any submodules.

vending_machine's logic:

always block #1:
if RESET is high, sets current_state to RESET_STATE
otherwise, sets current_state as next_state
always block #2:
for the 4 states that could timeout, increment counter
(TRANSACT_STATE_1, TRANSACT_STATE_2,
WAIT_VALID_SIGNAL_STATE, VEND_STATE)
for other states, set counter to 0

if current_state is not next_state, set counter to 0 (for a fresh counter start on state changes)

always block #3:

implementing the fsm diagram state transitions in verilog code

for the 4 states with possible timeouts, next_state determined by both counter AND inputs

for the other states, next_state determined by inputs

if current_state is IDLE_STATE:

if CARD_IN is high, go to TRANSACT_STATE_1

if RELOAD is high, go to RELOAD_STATE

otherwise, remain in IDLE_STATE

if current_state is RESET_STATE:

if RESET is low, go to IDLE_STATE

otherwise, remain in RESET_STATE

if current_state is RELOAD_STATE:

if RELOAD is low, go to IDLE_STATE

otherwise, remain in RELOAD_STATE

if current_state is TRANSACT_STATE_1:

if counter is at 5, timeout, and go to IDLE_STATE

if counter is less than 5,

if KEY_PRESS is high,

store ITEM_CODE in MSD for later

go to TRANSACT_STATE_2

otherwise, remain in TRANSACT_STATE_1

if current_state is TRANSACT_STATE_2:

if counter is at 5, timeout, and go to IDLE_STATE

if counter is less than 5,

if KEY_PRESS is high,

store ITEM_CODE in LSD

if the item code represented by MSD and LSD are invalid,

go to INVALID_SEL_STATE

if the item code represented by MSD and LSD are valid but the item counter for that item is 0,

go to INVALID_SEL_STATE

otherwise, go to WAIT_VALID_SIGNAL_STATE

otherwise, remain in TRANSACT_STATE_2

if current_state is INVALID_SEL_STATE:

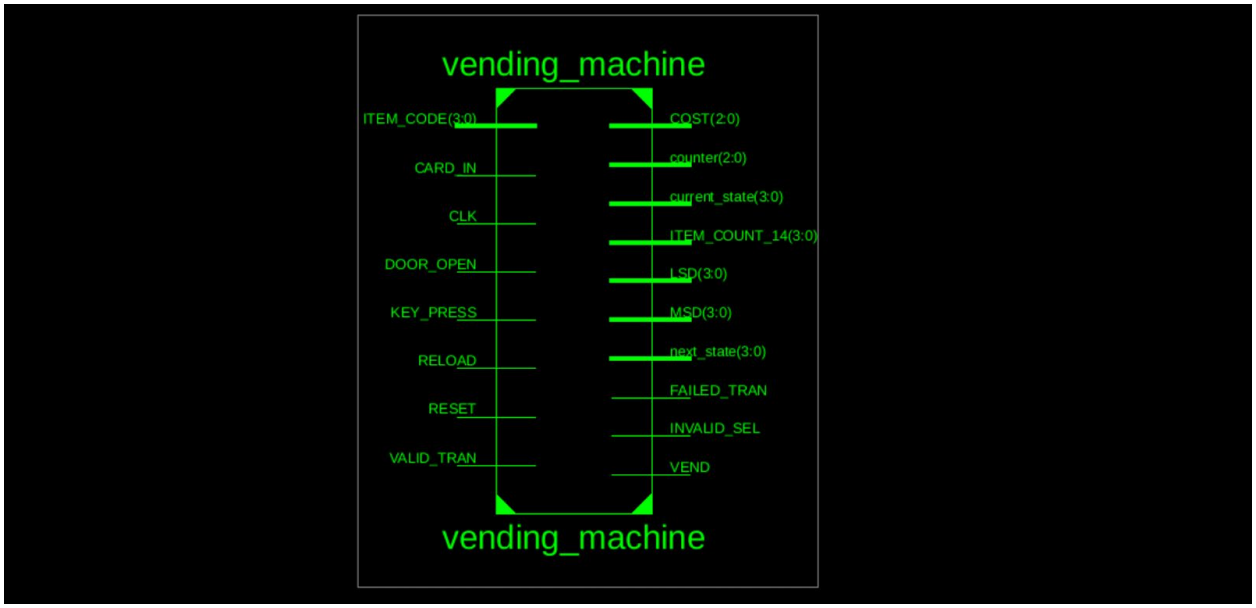
go to IDLE_STATE

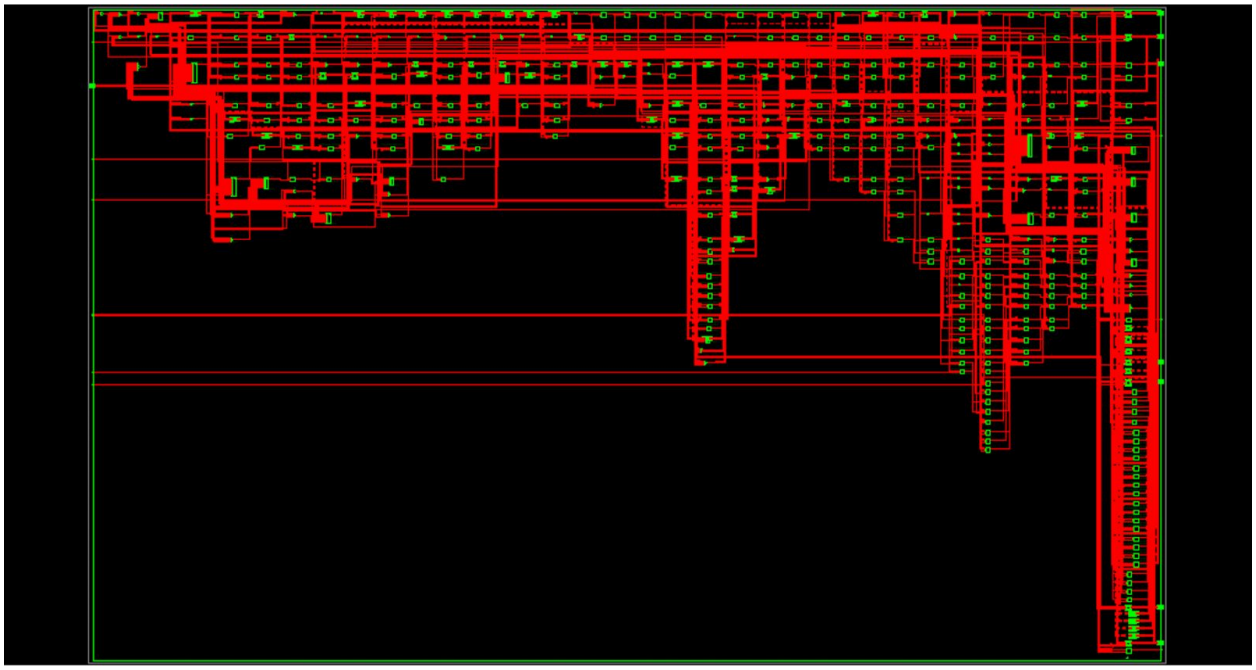
if current_state is WAIT_VALID_SIGNAL_STATE:

if counter is at 5, timeout, and go to INVALID_TRAN_STATE

```
                if counter is less than 5,
                        if VALID_TRAN is high,
                                go to VEND_STATE
                        otherwise, remain in WAIT_VALID_SIGNAL_STATE
        if current_state is VEND_STATE:
                if counter is at 5, timeout, and go to IDLE_STATE
                if counter is less than 5,
                        if DOOR_OPEN is high,
                                go to DOOR_HIGH_STATE
                        otherwise, remain in VEND_STATE
        if current_state is INVALID_TRAN_STATE:
                go to IDLE_STATE
        if current_state is DOOR_HIGH_STATE:
                if DOOR_OPEN is low, go to IDLE_STATE
                otherwise, remain in DOOR_HIGH_STATE
always block #4:
        **set outputs based on current_state**
        if current_state is IDLE_STATE:
                set all 4 outputs to 0
        if current_state is RESET_STATE:
                set all 4 outputs to 0
                set all 20 item counters to 0
        if current_state is RELOAD_STATE:
                set all 20 item counters to 10
        if current_state is TRANSACT_STATE_1:
                do nothing
        if current_state is TRANSACT_STATE_2:
                do nothing
        if current_state is INVALID_SEL_STATE:
                set INVALID_SEL to high
        if current_state is WAIT_VALID_SIGNAL_STATE:
                based on item code, set appropriate COST
        if current_state is VEND_STATE:
                set VEND to high
                decrement appropriate item counter based on item code
        if current_state is INVALID_TRAN_STATE:
                set FAILED_TRAN to high
        if current_state is DOOR_HIGH_STATE:
                do nothing
```

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 each element is an individual mux or gate. It is quite impossible to take in much information from the full schematic because everything is so broken down.

vending_machine Project Status			
Project File:	Proj4.xise	**Parser Errors:**	No Errors
Module Name:	vending_machine	**Implementation State:**	Placed and Routed
Target Device:	xc6slx4-3tqg144	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	139 Warnings (115 new)
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	94	4,800	1%	
Number used as Flip Flops	10			
Number used as Latches	84			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	116	2,400	4%	
Number used as logic	115	2,400	4%	
Number using O6 output only	71			
Number using O5 output only	0			
Number using O5 and O6	44			
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	1			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	40	600	6%	
Number of MUXCYs used	0	1,200	0%	
Number of LUT Flip Flop pairs used	135			
Number with an unused Flip Flop	42	135	31%	
Number with an unused LUT	19	135	14%	
Number of fully used LUT-FF pairs	74	135	54%	
Number of unique control sets	23			
Number of slice register sites lost to control set restrictions	90	4,800	1%	
Number of bonded IOBs	40	102	39%	
IOB Flip Flops	4			

IOB Latches	18			
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	4	200	2%	
Number used as ILOGIC2s	4			
Number used as ISERDES2s	0			
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	200	0%	
Number of OLOGIC2/OSERDES2s	18	200	9%	
Number used as OLOGIC2s	18			
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	3.27			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Fri Dec 11 23:20:12 2020	0	114 Warnings (90 new)	2 Infos (2 new)
Translation Report	Current	Fri Dec 11 23:20:21 2020	0	0	0
Map Report	Current	Fri Dec 11 23:20:34 2020	0	25 Warnings (25 new)	6 Infos (6 new)
Place and Route Report	Current	Fri Dec 11 23:20:41 2020	0	0	3 Infos (3 new)
Power Report					
Post-PAR Static Timing Report	Current	Fri Dec 11 23:20:45 2020	0	0	4 Infos (4 new)
Bitgen Report					

Secondary Reports			[-]
Report Name	**Status**	**Generated**	
ISIM Simulator Log	Out of Date	Fri Dec 11 23:19:55 2020	

Date Generated: 12/12/2020 - 04:53:54

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.

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

Macro Statistics
# RAMs                                               : 1
 16x7-bit single-port Read Only RAM                  : 1
# Adders/Subtractors                                 : 19
 3-bit adder                                         : 1
 4-bit subtractor                                    : 18
# Registers                                          : 2
 3-bit register                                      : 1
 4-bit register                                      : 1
# Latches                                            : 90
 1-bit latch                                         : 90
# Comparators                                        : 3
 4-bit comparator equal                              : 1
 4-bit comparator greater                            : 2
# Multiplexers                                       : 249
 1-bit 2-to-1 multiplexer                            : 232
 4-bit 2-to-1 multiplexer                            : 17

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

=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


Total REAL time to Xst completion: 5.00 secs
Total CPU time to Xst completion: 4.76 secs

-->


Total memory usage is 485720 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :  114 (   0 filtered)
Number of infos    :    2 (   0 filtered)
```

```
=========================================================================
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  |
-----------------------------------------------------------------------------------+-------------------------+-------+
current_state[3]_GND_6_o_Mux_143_o(Mmux_current_state[3]_GND_6_o_Mux_143_o11:O)  | NONE(*)(LSD_3)          | 8     |
current_state[3]_GND_10_o_Mux_151_o(Mmux_current_state[3]_GND_10_o_Mux_151_o11:O)| NONE(*)(MSD_3)          | 8     |
Mram__n11494(Mram__n114941:O)                                                      | NONE(*)(FAILED_TRAN)    | 1     |
Mram__n11492(Mram__n114921:O)                                                      | NONE(*)(VEND)           | 1     |
Mram__n1149(Mram__n114911:O)                                                       | NONE(*)(INVALID_SEL)    | 1     |
current_state[3]_GND_28_o_Mux_541_o(Mmux_current_state[3]_GND_28_o_Mux_541_o11:O)| NONE(*)(ITEM_COUNT_03_2)| 4     |
current_state[3]_GND_16_o_Mux_517_o(Mmux_current_state[3]_GND_16_o_Mux_517_o11:O)| NONE(*)(ITEM_COUNT_00_1)| 4     |
current_state[3]_GND_44_o_Mux_573_o(Mmux_current_state[3]_GND_44_o_Mux_573_o11:O)| NONE(*)(ITEM_COUNT_07_2)| 4     |
current_state[3]_GND_60_o_Mux_605_o(Mmux_current_state[3]_GND_60_o_Mux_605_o11:O)| NONE(*)(ITEM_COUNT_11_3)| 4     |
current_state[3]_GND_80_o_Mux_645_o(Mmux_current_state[3]_GND_80_o_Mux_645_o12:O)| NONE(*)(ITEM_COUNT_16_2)| 4     |
current_state[3]_GND_72_o_Mux_629_o(Mmux_current_state[3]_GND_72_o_Mux_629_o1:O) | NONE(*)(ITEM_COUNT_14_2)| 4     |
current_state[3]_GND_68_o_Mux_621_o(Mmux_current_state[3]_GND_68_o_Mux_621_o11:O)| NONE(*)(ITEM_COUNT_13_3)| 4     |
current_state[3]_GND_64_o_Mux_613_o(Mmux_current_state[3]_GND_64_o_Mux_613_o12:O)| NONE(*)(ITEM_COUNT_12_2)| 4     |
current_state[3]_GND_24_o_Mux_533_o(Mmux_current_state[3]_GND_24_o_Mux_533_o11:O)| NONE(*)(ITEM_COUNT_02_2)| 4     |
current_state[3]_GND_92_o_Mux_669_o(Mmux_current_state[3]_GND_92_o_Mux_669_o1:O) | NONE(*)(ITEM_COUNT_19_1)| 4     |
current_state[3]_GND_76_o_Mux_637_o(Mmux_current_state[3]_GND_76_o_Mux_637_o1:O) | NONE(*)(ITEM_COUNT_15_3)| 4     |
```

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

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

   Minimum period: 4.836ns (Maximum Frequency: 206.774MHz)
   Minimum input arrival time before clock: 7.747ns
   Maximum output required time after clock: 3.813ns
   Maximum combinational path delay: No path found

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

HDL Synthesis Report:

This report tells us which parts were utilized for the synthesis of this project. A large number of multiplexers are used, a total of 249, which is significantly lower than the number of multiplexers used for project 3. Unlike project 3 which used a 4x4 read only RAM, project 4 used a 16x7 read only RAM. The fact that all logic were in always blocks with no submodules and a overall simpler structure may have actually saved up on parts. Unlike previous projects, the Synthesis Report for this project also included a very long Timing Report, which went into great detail about the CLK implemented as well as timing constraints.

```
Design Summary
--------------
Number of errors:      0
Number of warnings:   25
Slice Logic Utilization:
  Number of Slice Registers:                    94 out of   4,800    1%
    Number used as Flip Flops:                  10
    Number used as Latches:                     84
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                0
  Number of Slice LUTs:                        116 out of   2,400    4%
    Number used as logic:                      115 out of   2,400    4%
      Number using O6 output only:              71
      Number using O5 output only:               0
      Number using O5 and O6:                   44
      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:      1
      Number with same-slice carry load:         0
      Number with other load:                    0

Slice Logic Distribution:
  Number of occupied Slices:                    40 out of     600    6%
  Number of MUXCYs used:                         0 out of   1,200    0%
  Number of LUT Flip Flop pairs used:          135
    Number with an unused Flip Flop:            42 out of     135   31%
    Number with an unused LUT:                  19 out of     135   14%
    Number of fully used LUT-FF pairs:          74 out of     135   54%
    Number of unique control sets:              23
    Number of slice register sites lost
      to control set restrictions:              90 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.

    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

IO Utilization:
  Number of bonded IOBs:                        40 out of     102   39%
    IOB Flip Flops:                              4
    IOB Latches:                                18

Average Fanout of Non-Clock Nets:               3.27

Peak Memory Usage:  762 MB
Total REAL time to MAP completion:  9 secs
Total CPU time to MAP completion:   8 secs
```

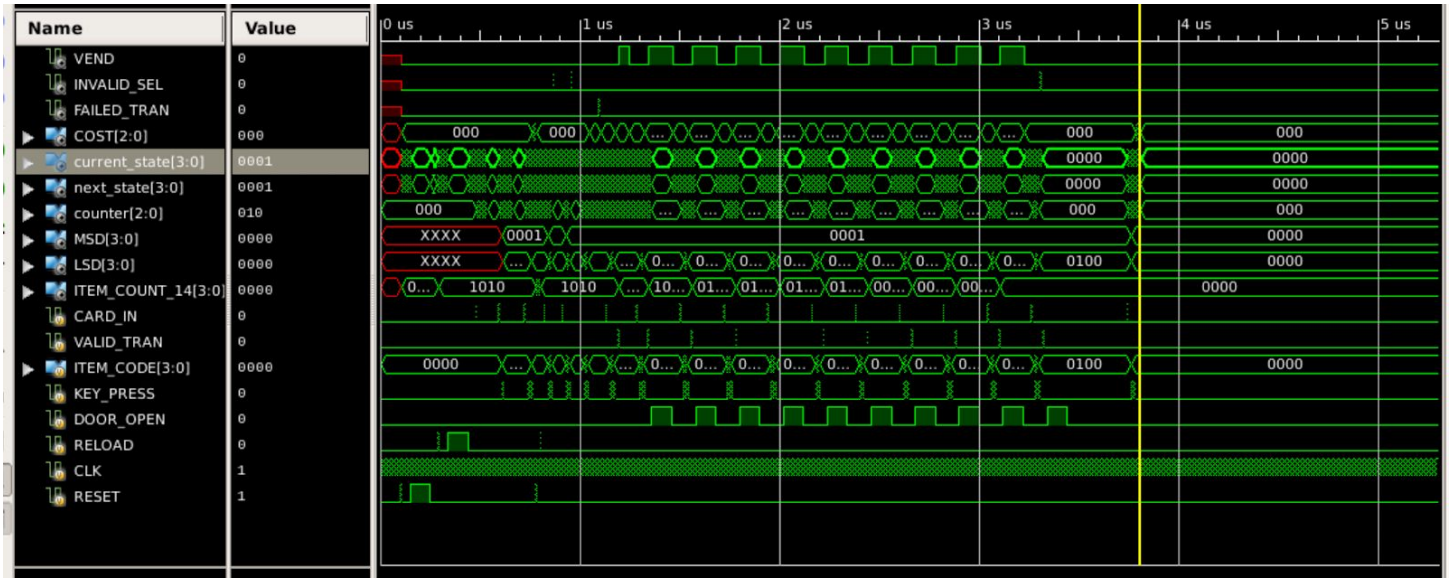
Map Report:

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 lesser by 1 second compared to that of project 3, which is not that great of a difference. The two projects have a similar level of complexity, so this similarity makes sense. Overall, this project did not consume as many parts as I anticipated.

```
Starting initial Timing Analysis.  REAL time: 2 secs
Finished initial Timing Analysis.  REAL time: 2 secs

Starting Router


Phase  1  : 796 unrouted;      REAL time: 2 secs

Phase  2  : 722 unrouted;      REAL time: 3 secs

Phase  3  : 347 unrouted;      REAL time: 3 secs

Phase  4  : 346 unrouted; (Par is working to improve performance)     REAL time: 3 secs

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

Phase  5  : 0 unrouted; (Par is working to improve performance)     REAL time: 3 secs

Phase  6  : 0 unrouted; (Par is working to improve performance)     REAL time: 3 secs

Phase  7  : 0 unrouted; (Par is working to improve performance)     REAL time: 3 secs

Phase  8  : 0 unrouted; (Par is working to improve performance)     REAL time: 3 secs

Phase  9  : 0 unrouted; (Par is working to improve performance)     REAL time: 3 secs

Phase 10  : 0 unrouted; (Par is working to improve performance)     REAL time: 3 secs
Total REAL time to Router completion: 3 secs
Total CPU time to Router completion: 3 secs
```

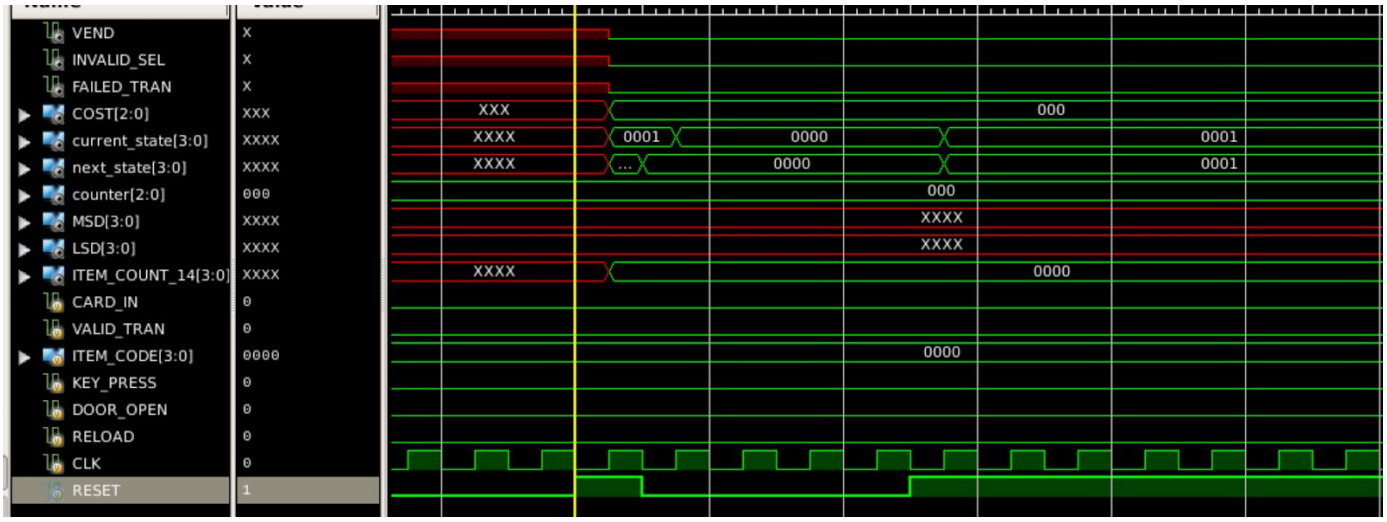
Place and Route Report:

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 11 seconds REAL time.



// zoomed out, entire view

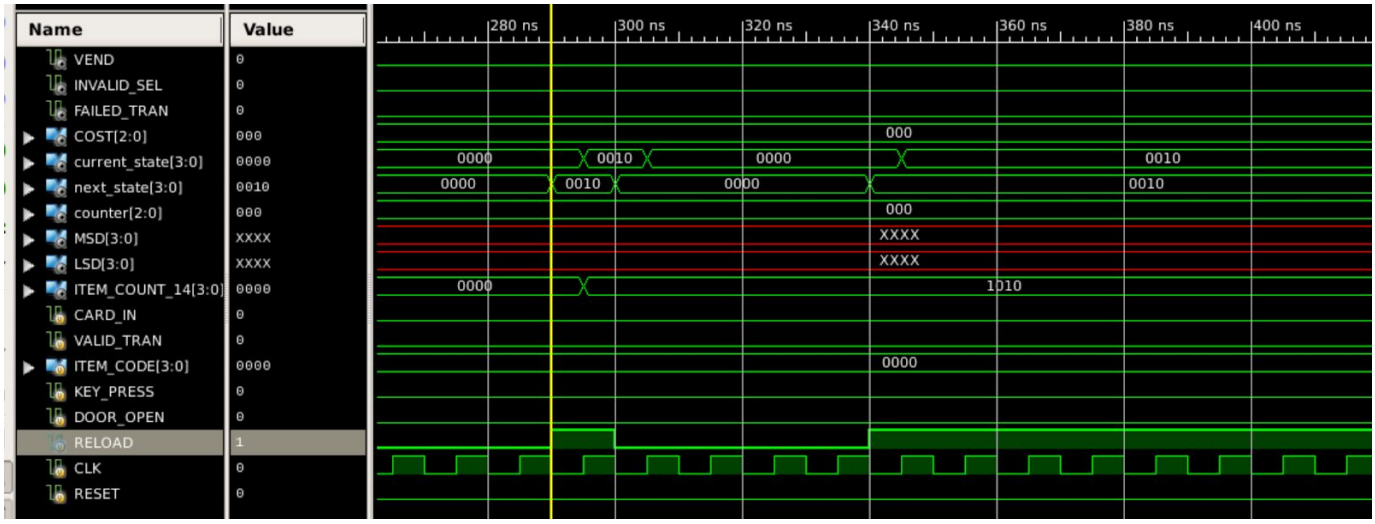
Test Case 1: RESET



- when RESET is high, the fsm appropriately enters the RESET_STATE (0001)
- returns back to IDLE_STATE(0000) when RESET is low
- all outputs are set to 0
- ITEM_COUNT_14 is set to 0 (thus, all item counters must be set to zero — 14 was the only one I displayed for debugging purposes)

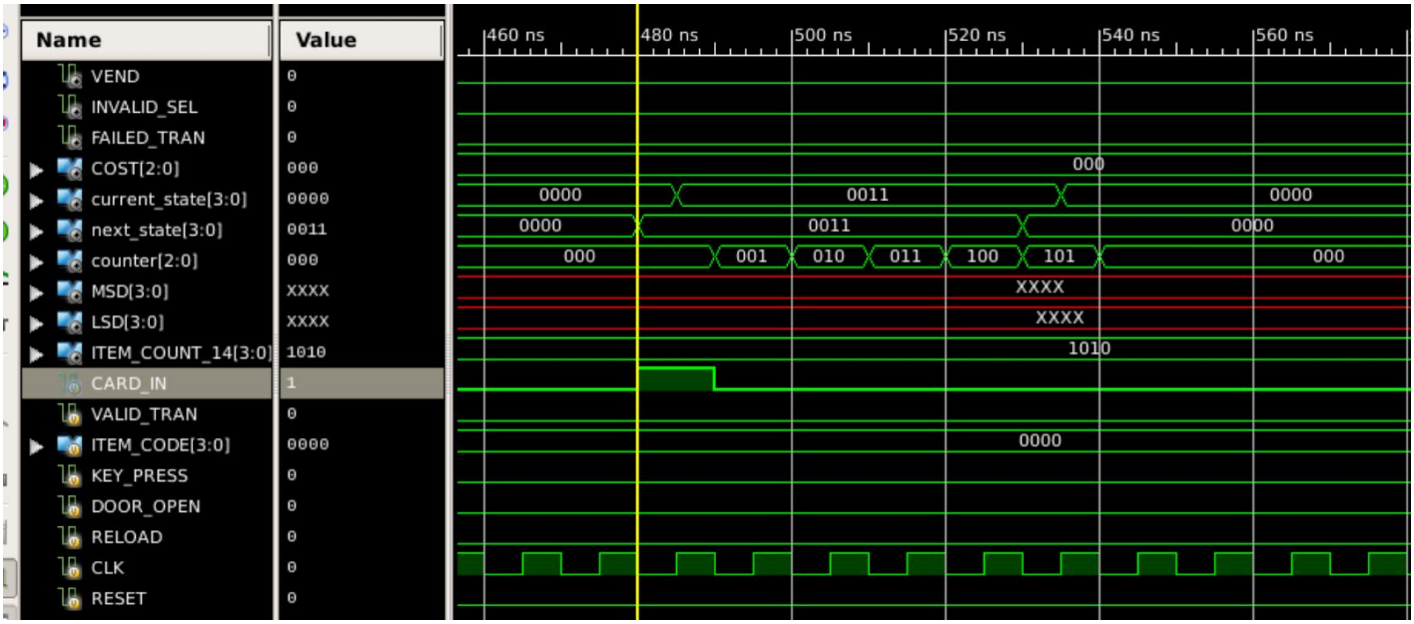
- the second time RESET goes high, it remains in the RESET_STATE for as long as RESET is high

Test Case 2: RELOAD



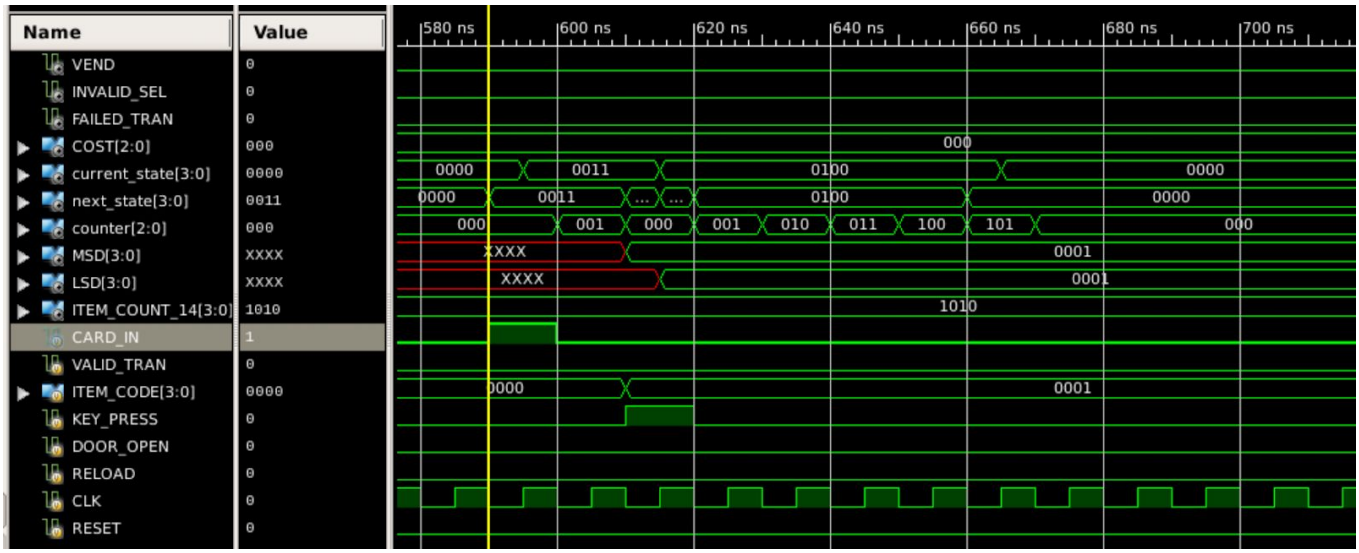
- when RELOAD is high, the fsm appropriate enters the RELOAD_STATE(0010)
- returns back to IDLE_STATE(0000) when RELOAD is low
- all item counters are set to 10 (only 14 is visible on waveform, but hard-coded so should be no difference)
- second time RELOAD goees high, it remains in the RELOAD_STATE for as long as RELOAD is high

Test Case 3: IDLE_STATE -> TRANSACT_STATE_1, TRANSACT_STATE_1 timeout



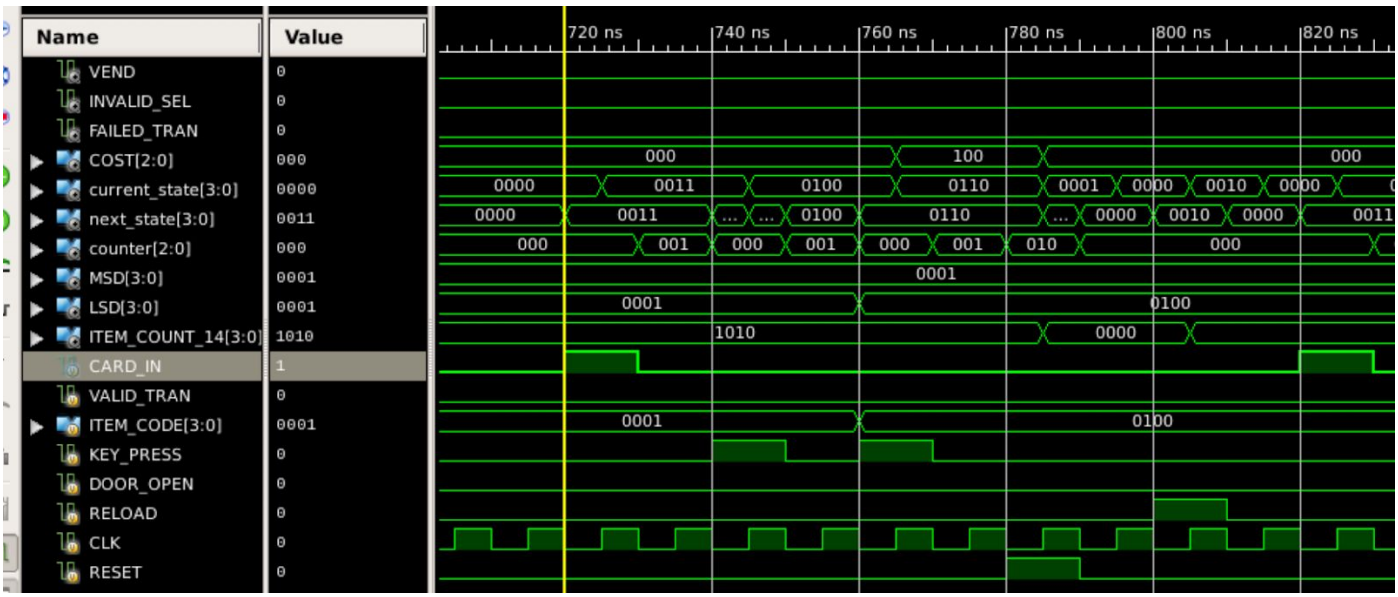
- when CARD_IN is high, the fsm appropriately enters the TRANSACT_STATE_1(0011)
- the fsm stays in the TRANSACT_STATE_1 for 5 CLK cycles (50ns)
- since KEY_PRESS does not go high for 50ns, there is a timeout and the fsm returns to IDLE_STATE
- TRANSACT_STATE_1 does not alter any outputs

Test Case 4: TRANSACT_STATE_1 -> TRANSACT_STATE_2, valid first digit entry, TRANSACT_STATE_2 timeout



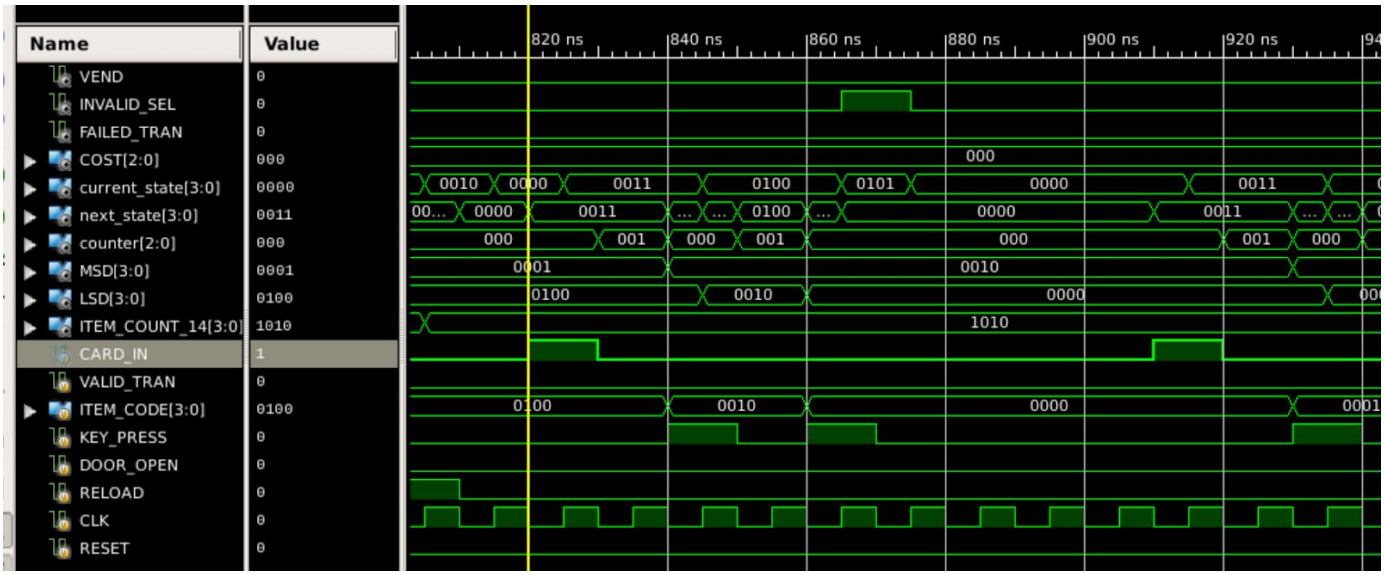
- when CARD_IN is high, the fsm appropriately enters the TRANSACT_STATE_1(0011)
- the fsm receives a KEY_PRESS signal and ITEM_CODE of 1 before it times out
- the fsm enters TRANSACT_STATE_2(0100)
- the fsm stores the first digit into MSD
- TRANSACT_STATE_2 does not alter any outputs
- the fsm times out after 50ns from TRANSACT_STATE_2 since there was no other KEY_PRESS and ITEM_CODE
- the fsm returns to IDLE_STATE

Test Case 5: TRANSACT_STATE_2 -> WAIT_VALID_SIGNAL_STATE, valid second digit



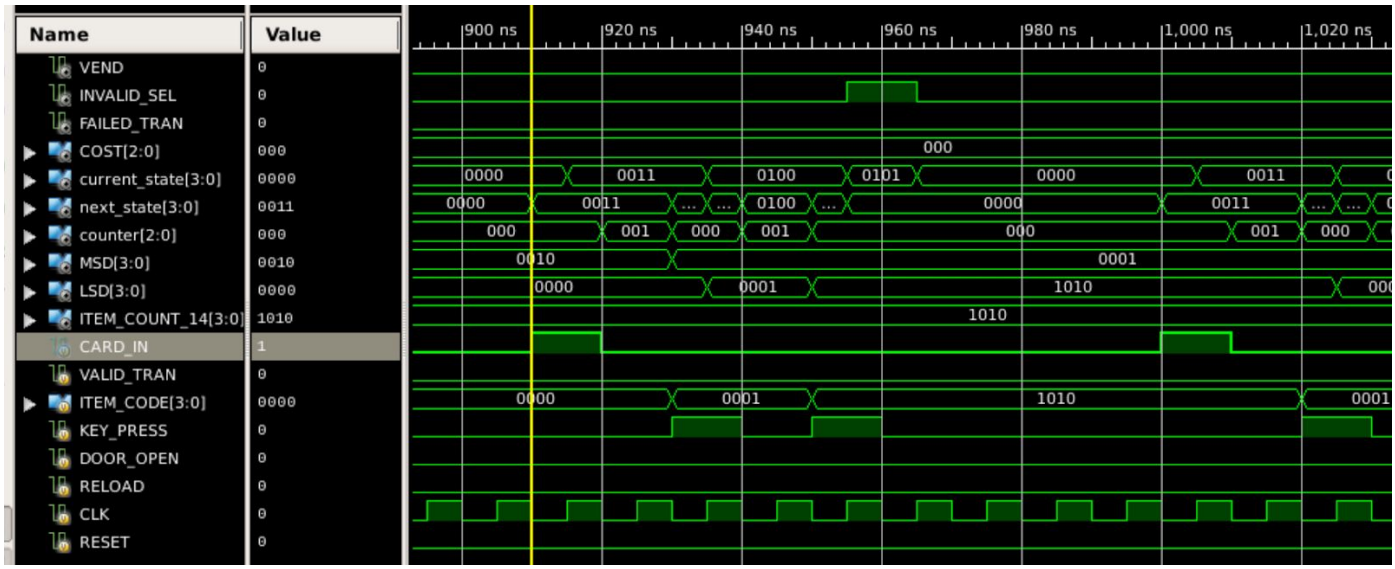
- from TRANSACT_STATE_2, fsm appropriately enters the WAIT_VALID_SIGNAL_STATE(0110) when KEY_PRESS goes high and the second digit is entered
- MSD is 1 and LSD is 4, and 14 is a valid item code
- WAIT_VALID_SIGNAL_STATE does not alter any outputs
- RESET and RELOAD in order to return to IDLE_STATE for the next test case

Test Case 6: TRANSACT_STATE_2 -> INVALID_SEL_STATE, invalid first digit entry



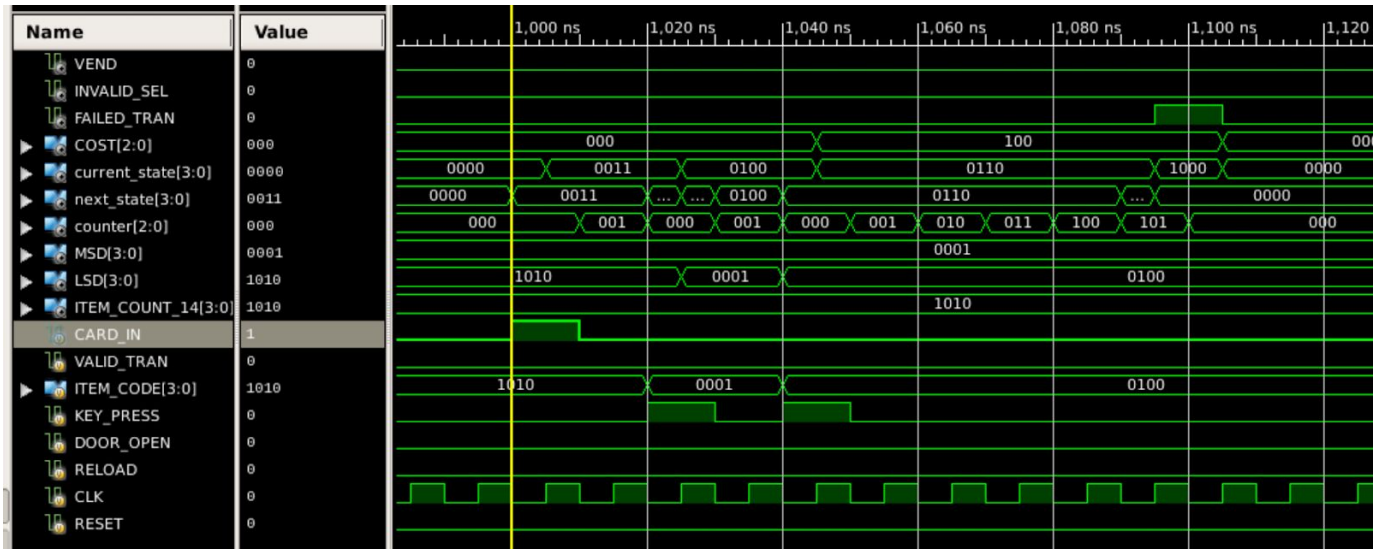
- from TRANSACT_STATE_1, fsm receives 2 as MSD (invalid)
- from TRANSACT_STATE_2, fsm receives 0 as LSD
- there is no item 20, thus the item code is invalid
- fsm does not go to WAIT_VALID_SIGNAL_STATE and goes to INVALID_SEL_STATE(0101) instead
- INVALID_SEL is set to high
- fsm immediately returns to IDLE_STATE

Test Case 7: TRANSACT_STATE_2 -> INVALID_SEL_STATE, invalid second digit entry



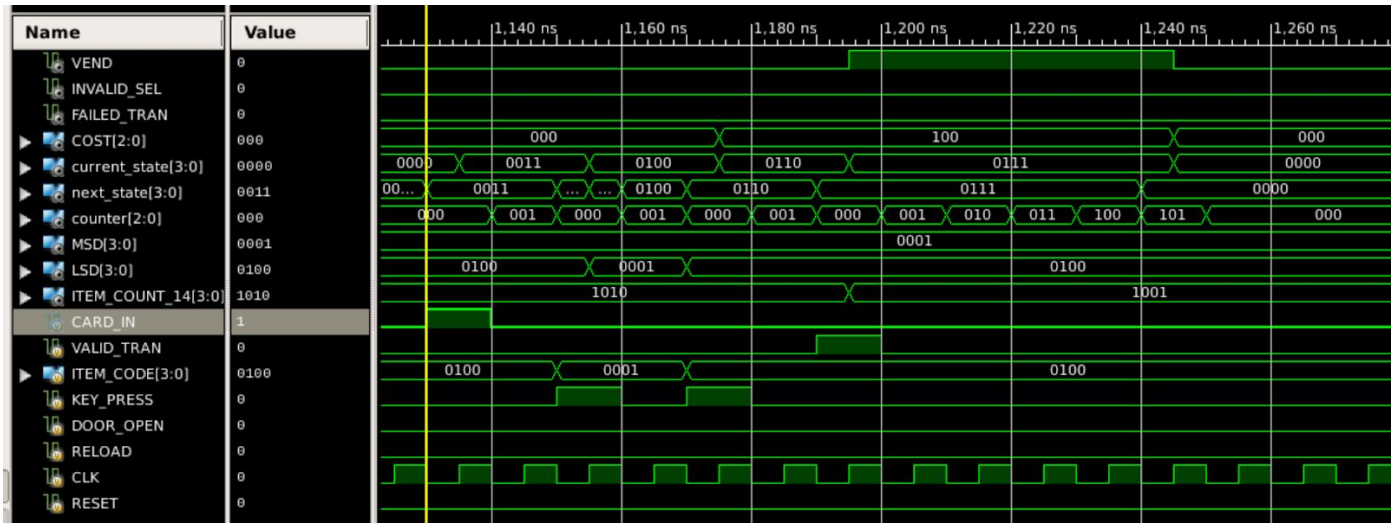
- from TRANSACT_STATE_1, fsm receives 1 as MSD
- from TRANSACT_STATE_2, fsm receives 10 as LSD (invalid)
- the second digit cannot be two digits long, thus the item code is invalid
- fsm does not go to WAIT_VALID_SIGNAL_STATE and goes to INVALID_SEL_STATE(0101) instead
- INVALID_SEL is set to high
- fsm immediately returns to IDLE_STATE

Test Case 8: WAIT_VALID_SIGNAL_STATE -> INVALID_TRAN_STATE,
WAIT_VALID_SIGNAL_STATE timeout



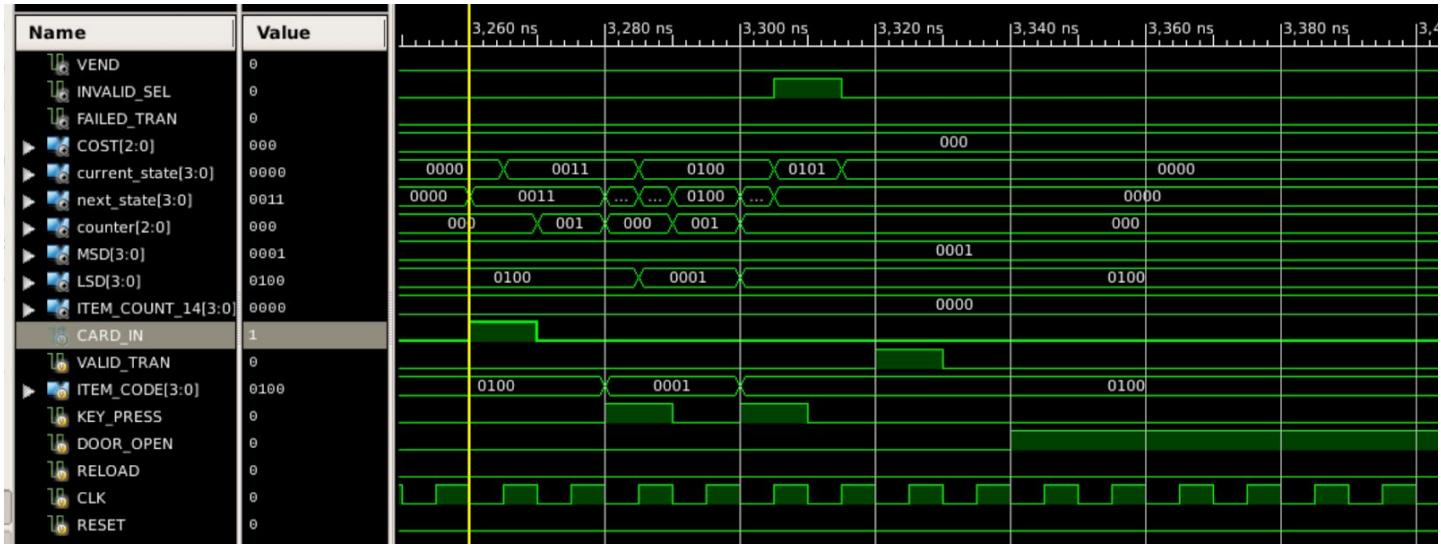
- with an appropriate item code of 14, fsm enters the WAIT_VALID_SIGNAL_STATE
- for 50ns, there is not VALID_TRAN input, and there is a timeout
- the fsm instead enters the INVALID_TRAN_STATE(1000) where it sets FAILED_TRAN to high
- the fsm immediately returns to IDLE_STATE

Test Case 9: first successful vending of item 14, VEND_STATE timeout



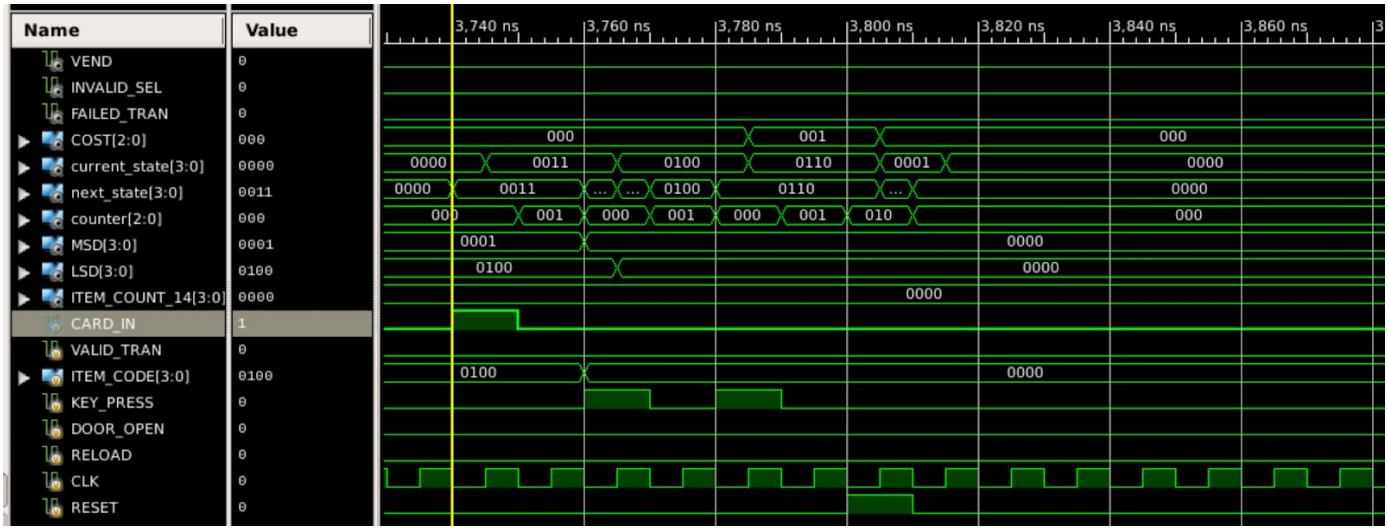
- appropriate item code of 14 is entered
- goes from TRANSACT_STATE_2 to WAIT_VALID_SIGNAL_STATE
- fsm recieves VALID_TRAN signal, enters VEND_STATE
- during VEND_STATE, fsm decrements item 14's counter from 10 to 9, and sets VEND to high
- fsm waits 50ns in VEND_STATE for the DOOR_OPEN signal, but no such signal comes
- fsm times out and returns to IDLE_STATE, but this was a successful vend

Test Case 10: 11th vending of item 14



- all 10 items of item 14 has been vended, and the counter is at 0
- the fsm tries to vend item 14 for the 11th time
- from TRANSACT_STATE_2, item code received is 14, and the fsm enters INVALID_SEL_STATE(0101)
- from INVALID_SEL_STATE, fsm returns to IDLE_STATE

Test Case 11: returning to RESET_STATE_ from a non-RELOAD_STATE or IDLE_STATE



- successfully enters RESET_STATE(0001) from WAIT_VALID_SIGNAL_STATE(0110)

Conclusion

The design of vending_machine includes no submodules, and deals with current_state, next_state, counter, and outputs in its own always block. The changes made to current_state and counter are offset in time by using a posedge and a negedge. The next_state is either a function of inputs and currenet_state or a function of inputs, counter, and current_state.

The most difficult part of this project was figuring out the timing, and how to really implement the counter in a way that the states would time out after 50ns. There were many more state transitions than project 3, but the transitions were much easier to visualize and implement compared to project 3. The given four always block setup made implementation possible.

I cannot think of a way to improve this project. The project was explained in more detail and guided compared to project 3, and was overall less stressful to implement.