1. Introduction

In the Project 4, we will implement a prototype of a vending machine. The vending machine will have a total of 20 slots, with each slot having a total capacity of 10. The core functionality of the machine is the ability for a user to input a card and select an item within the 20 slots. Our machine subsequently processes these input signals and ensures that the selection is valid. Afterwards, it outputs the cost and waits for the transaction to be verified. Finally, when the transaction is complete, the vending machine vends the selected item to the user and decrements its internal counter of the respective item. For a high-level view of the machine, the input and output have been included below.

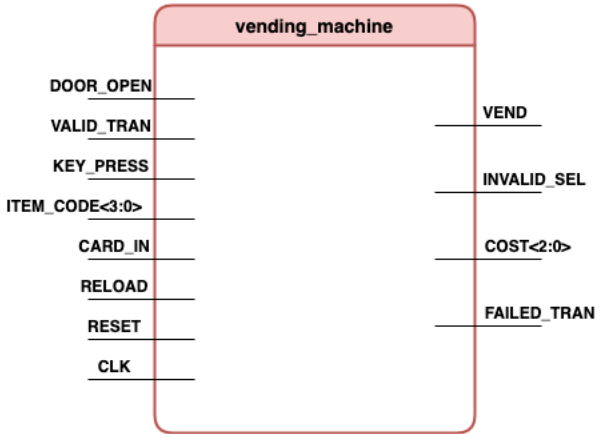


Figure 1: *High-level Module of Vending Machine Input and Output*

INPUT	SIZE / BEHAVIOR
CLK	1 bit. System clock (T= 10 ns)
RESET	1 bit. Set all item counters and outputs to 0 and go to the idle state.
RELOAD	1 bit. Reload the machine. Set all item counters to 10.
CARD_IN	1 bit. Stays high as long as the card remains inserted.
ITEM_CODE<3:0>	4-bit signal to input item code. The 2 digit item code is entered one digit at a time. **
KEY_PRESS	1 bit. ITEM_CODE is valid for reading when this signal is high.
VALID_TRAN	1 bit. HIGH = Transaction using the card is valid. (This can go high any time after item selection is determined to be valid. A card does not need to be inserted when this occurs.)
DOOR_OPEN	1 bit. HIGH = The vending machine door is open. (This can occur any time after the 'VEND' goes high.)

Table 1: *Output Description*

OUTPUT	SIZE / BEHAVIOR
VEND	1 bit. Set to HIGH once the transaction is deemed to be valid. Set to LOW once DOOR_OPEN goes high and then low or if the door does not open in 5 clock cycles.
INVALID_SEL	1 bit. Set to HIGH if: • Only 1 digit of ITEM_CODE is entered and there is no 2nd digit after 5 clock cycles or no digit is entered for clock cycles • The 2 digit ITEM_CODE is invalid. (i.e., 23) • The counter for one of the items is 0.
COST<2:0>	3 bits. Set to 000 by default. Set to the cost of an item once item code is entered, and remains at this value until a new transaction begins. (i.e., $5 = 101)
FAILED_TRAN	1 bit. Set to 1 if VALID_TRAN signal does not go high within 5 clock cycles of determining the ITEM_CODE.

Table 2: *Input Description*

2. Design

Our vending machine has to behave robustly and properly handle all possible input combinations. To deal with this level of complexity, we need to utilize a finite state machine (FSM) since it provides us a systematic mechanism for handling input and output at different stages. The core design of the vending machine utilizes a Moore FSM with two auxiliary logical blocks (for simplicity's sake we can visualize the blocks as "submodules" with inputs and output). The diagram below shows a detailed view of the design.

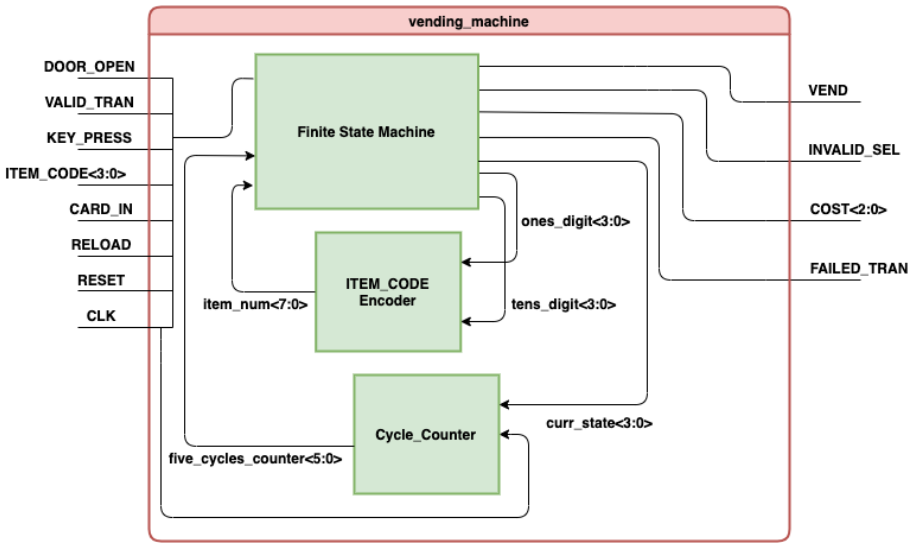


Figure 2: *Detailed-View of Vending Machine Design* – The core functionality of the vending machine is handled by a FSM with two auxiliary logical blocks are used to help it determine the next state.

3. Implementation

3.1. Finite State Machine

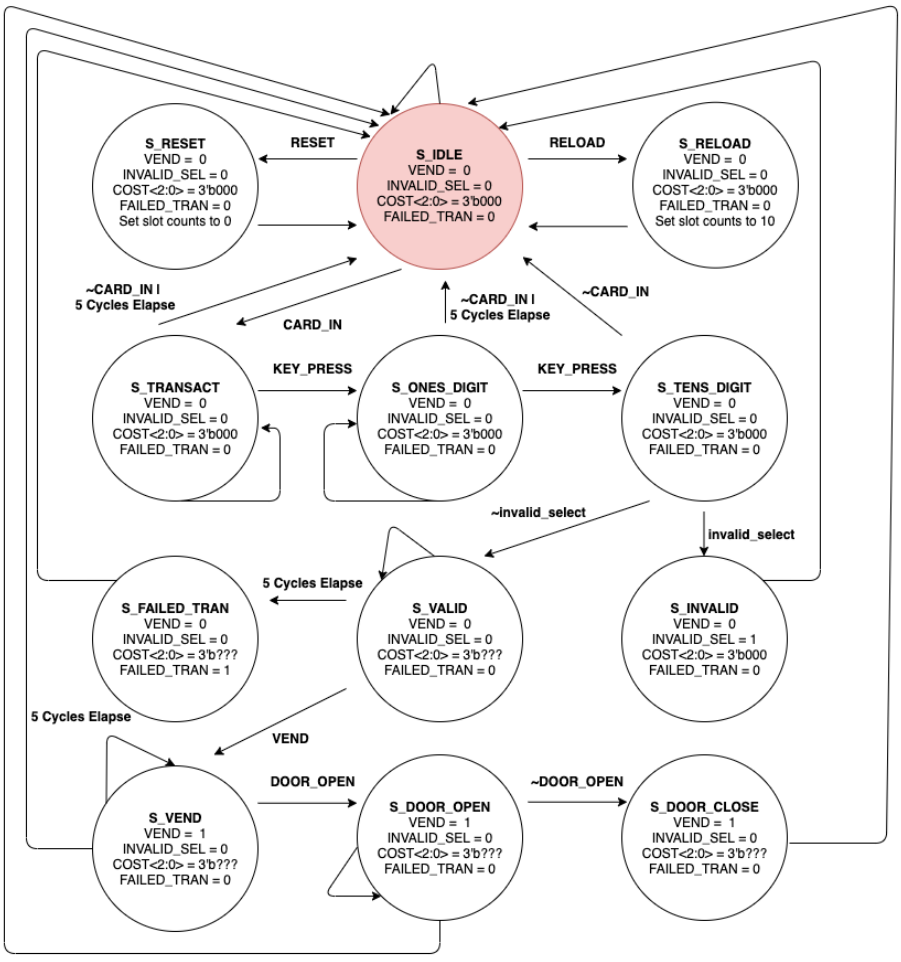


Figure 3: *Moore Finite State Machine with 12 States*– The core functionality of our vending machine is handled by a 12 state Moore FSM. S_IDLE is the initial state of the FSM. Note that whenever RESET is high, the next state is S_RESET regardless of current state.

State Descriptions:

- **S_IDLE:** The initial state of the FSM. In S_IDLE, all outputs of the system (VEND, INVALID_SEL, COST<2:0>, FAILED_TRAN) are set to 0 while the FSM waits for user input.
- **S_RESET:** The reset state of the FSM. The S_RESET state effectively resets the vending machine back to factory condition. All the internal counts of items in the 20 available slots are set to zero. The outputs are also all set to 0. The reset can occur at any state in the FSM.
- **S_RELOAD:** Reloads the vending machine so that all the internal counts of 20 slots are set to 10. The state can only be reached during the S_IDLE. Intuitively, this means that we cannot restock the machine while a transaction is occurring, or we are resetting the machine.
- **S_TRANSACT:** The first step of the user transaction process. In S_TRANSACT, we wait for the first KEY_PRESS and the first ITEM_CODE<2:0>. If KEY_PRESS does not go high within 5 cycles or the card is abruptly taken out (i.e. KEY_PRESS goes low), we go back to S_IDLE. Otherwise we go to S_ONES_DIGIT.
- **S_ONES_DIGIT:** During this state, we wait for the tens digit when KEY_PRESS goes high. Similar to S_TRANSACT, if KEY_PRESS does not go high in 5 cycles or the card is taken out, we go back to S_IDLE. Otherwise, we go to S_TENS_DIGIT to process the two ITEM_CODE inputted.
- **S_TENS_DIGIT:** This state makes use of the auxiliary logical block that converts ones_digit and the tens_digit to binary. With the binary representation, we can determine if the inputted code is within the bounds of the slot. Additionally, we need to make sure that the slot of the inputted item number is not 0. If these two conditions hold, the next state is S_VALID. Otherwise, the next state is S_INVALID.
- **S_INVALID:** Set the INVALID_SET output to 1 and then return to S_IDLE.
- **S_VALID:** Determine the cost of the item number inputted and set COST<2:0> to that value. To determine the next state, wait for the VALID_TRAN bit for 5 cycles. If VALID_TRAN does not go high within 5 cycles, the next state is S_FAILED_TRAN. Otherwise, the next state is S_VEND.
- **S_FAILED_TRAN:** Set the FAILED_TRAN output bit 1 and return to S_IDLE.
- **S_VEND:** Set the VEND bit to 1 and decrement the internal count of the item number selected. For the next state, wait for the DOOR_OPEN signal to go high for 5 cycles. If this does not occur, return to S_IDLE. Otherwise, go to S_DOOR_OPEN.
- **S_DOOR_OPEN:** Wait 5 cycles for DOOR_OPEN to go low and go to S_DOOR_CLOSE. Otherwise, go to S_IDLE.
- **S_DOOR_CLOSE:** Intermediary state that takes us to S_IDLE.

Note: Arrows that do not contain a label, indicate the default behavior of the FSM when none of the other input paths are available.

3.2. ITEM_CODE Encoder

The ITEM_CODE encoder is a combinatorial block that takes in the ones_digit<3:0> and the tens_digit<3:0> extracted from the finite state machine and encodes it into a binary value. The binary is subsequently feed back into the state machine to determine its validity and cost.

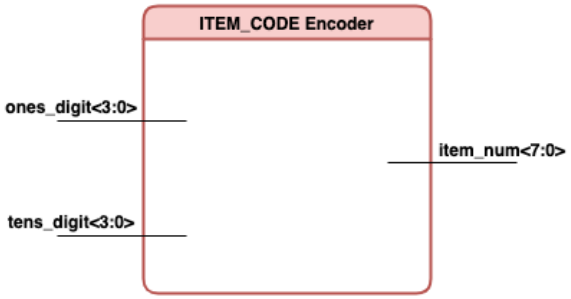


Figure 4: *ITEM_CODE Encoder I/O*

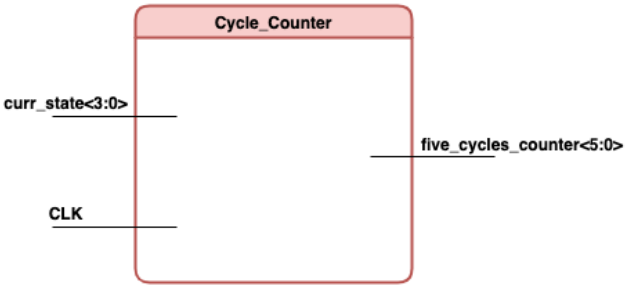


Figure 5: *Cycle_Counter*

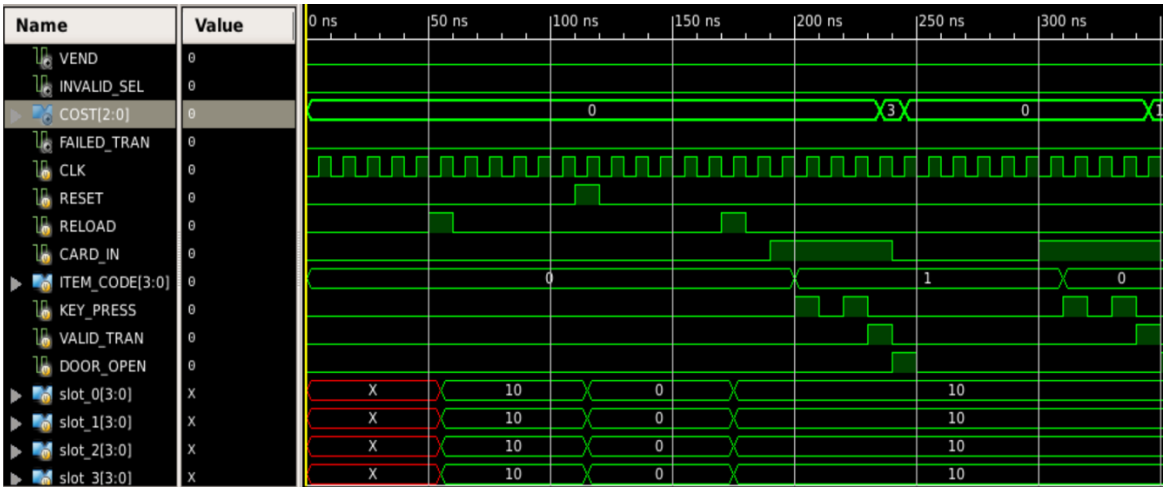
3.3. Cycle_Counter

The Cycle_Counter is a sequential logical block that takes in the curr_state and the CLK as input. If the curr_state is not S_TRANSACT, S_ONES_DIGIT, S_TENS_DIGIT, S_VALID, S_VEND, or S_DOOR_OPEN, the five_cycles_counter is incremented by 1 on every positive edge. Otherwise, the five_cycles_counter is equal to 0. The FSM uses the five_cycles_counter to determine if 5 clock cycles have passed before the necessary inputs to progress.

4. Testbench

The design does not call for outputting the internal counts of the slots or the curr_state and the next_state of the FSM. However, for a more comprehensive understanding of how our vending functions, there signals have been included.

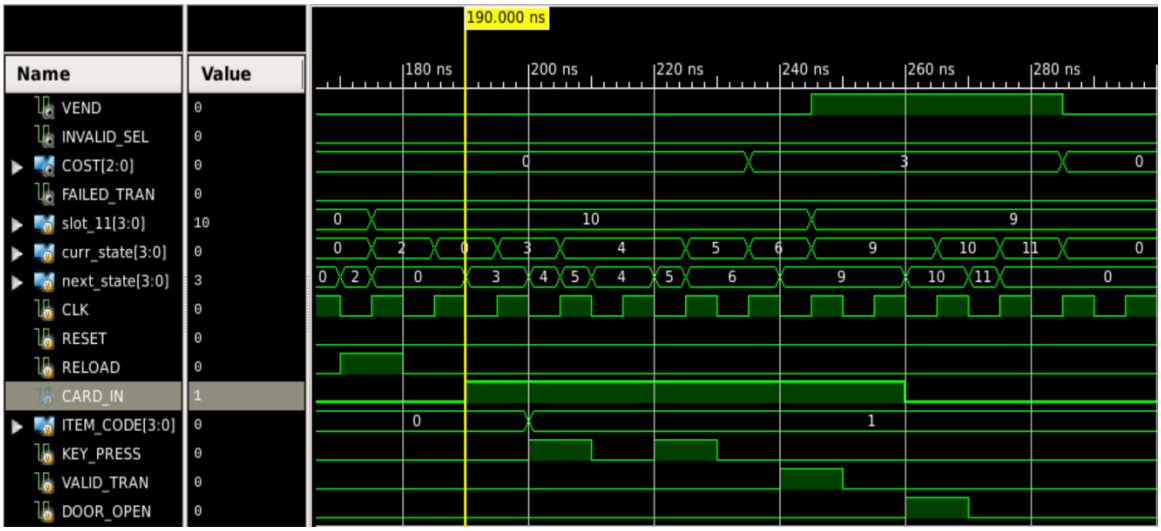
4.1. Reload and Reset



Waveform 1: *Reload and Reset Test with 100 MHz Clock* – RELOAD signal goes high at 50 ns, and RESET signal goes high at 110 ns.

- When RELOAD signal goes high at 50 ns, the internal counts of each of the slots are set to 10.
- When RESET signal goes high at 110 ns, the internal counts of the slots are set to 0
- VEND=0, INVALID_SET=0, COST=3'b000, FAILED_TRAN=0 in both cases

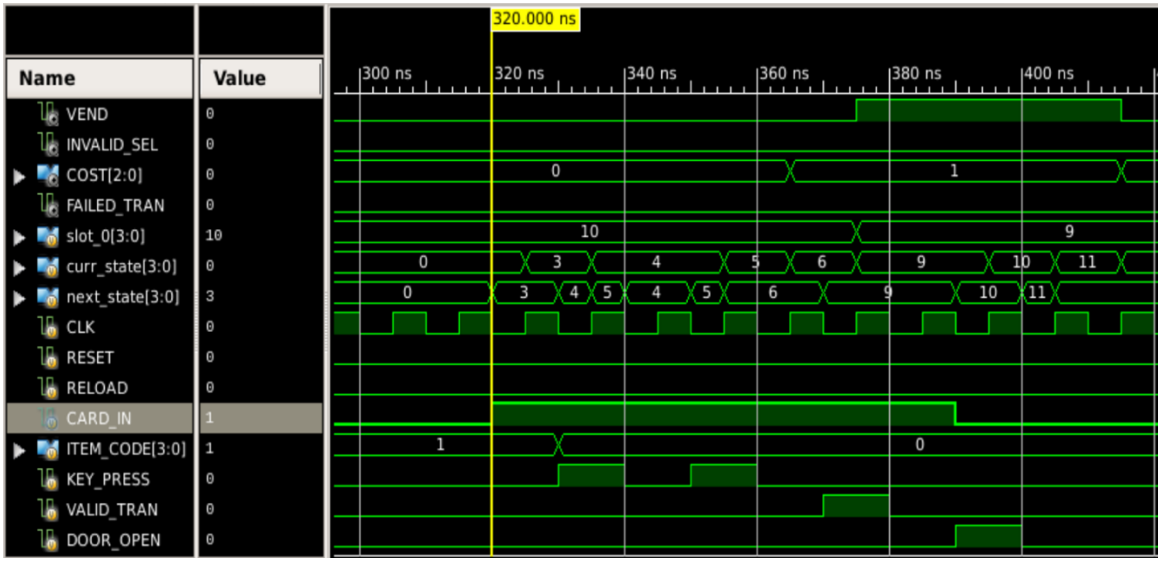
4.2. Vend Item in Slot 11



Waveform 2: *Vend Item in Slot 11 Test with 100 MHz Clock*– CARD_IN signal goes high at 190 ns, which signals the FSM to begin processing transaction.

- CARD_IN signal goes high at 190 ns, followed by two clicks to KEY_PRESS
- ITEM_CODE<3:0> during both KEY_PRESS is 1, which means that item number = 11
- When curr_state is S_VALID (6), the COST is set. Items in slot 11 cost $3.
- Once VALID_TRAN goes high, the curr_state goes to S_VEND (9) and the VEND output is set to 1. Internal count slot_11 is decremented from 10 to 9 in S_VEND.
- We also ensure that DOOR_OPEN goes high then low, before curr_state returns to S_IDLE (0)

4.3. Vend Item in Slot 0

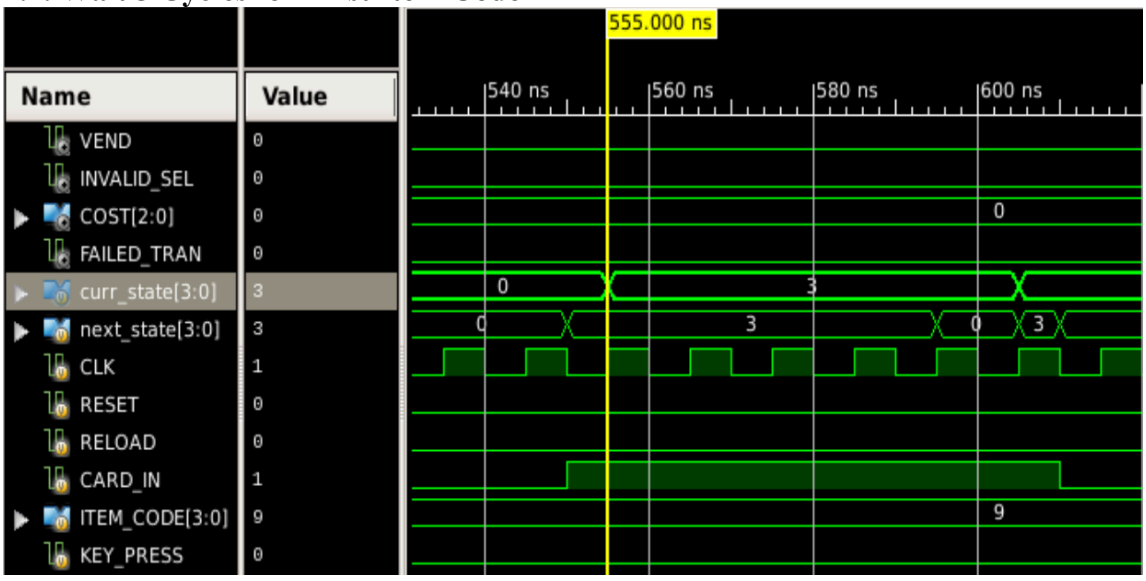


Waveform 3: *Vend Item in Slot 00 Test with 100 MHz Clock*– CARD_IN signal goes high at 320 ns.

- CARD_IN signal goes high at 320 ns, followed by two clicks to KEY_PRESS
- ITEM_CODE<3:0> during both KEY_PRESS is 0; item number = 00
- When curr_state is S_VALID (6), COST is set to 1

- Once VALID_TRAN goes high, the curr_state goes to S_VEND (9) and the VEND output is set to 1. Internal count slot_0 is decremented from 10 to 9 in S_VEND.
- DOOR_OPEN signal goes high then low before returning to S_IDLE

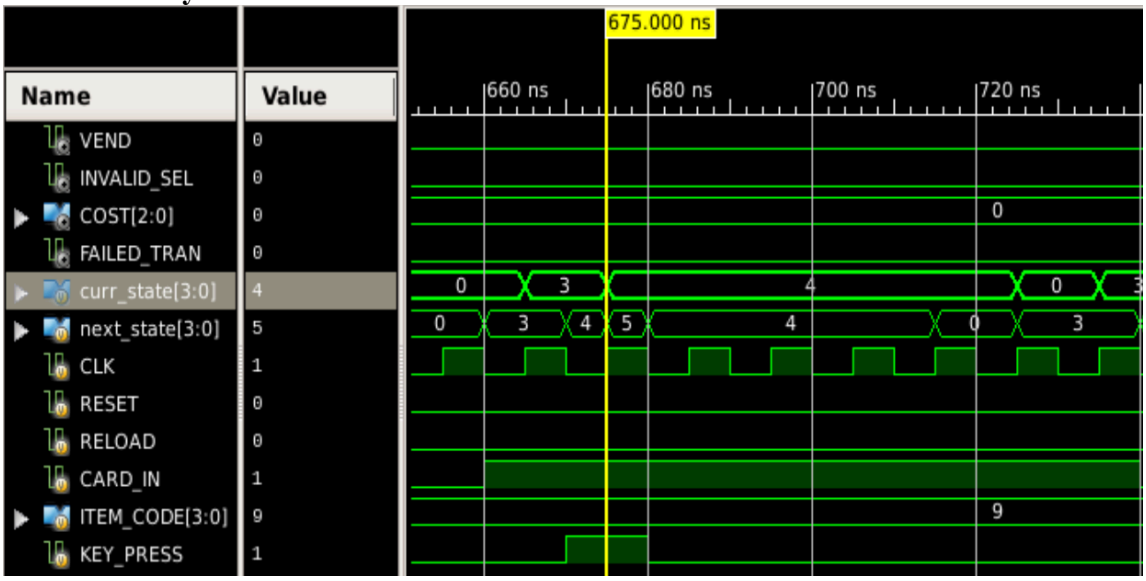
4.4. Wait 5 Cycles for First Item Code



Waveform 4: *Wait 5 Cycles for First Item Code with 100 MHz Clock*– CARD_IN signal goes high at 550 ns.

- The curr_state is set to S_TRAN (3) at 555 ns
- In S_TRAN, we have to wait for the first ITEM_CODE (i.e. for KEY_PRESS to go high)
- Since KEY_PRESS does not go high from 555 ns to 605 ns (5 clock cycles), curr_state goes back to S_IDLE (0)

4.5. Wait 5 Cycles for Second Item Code

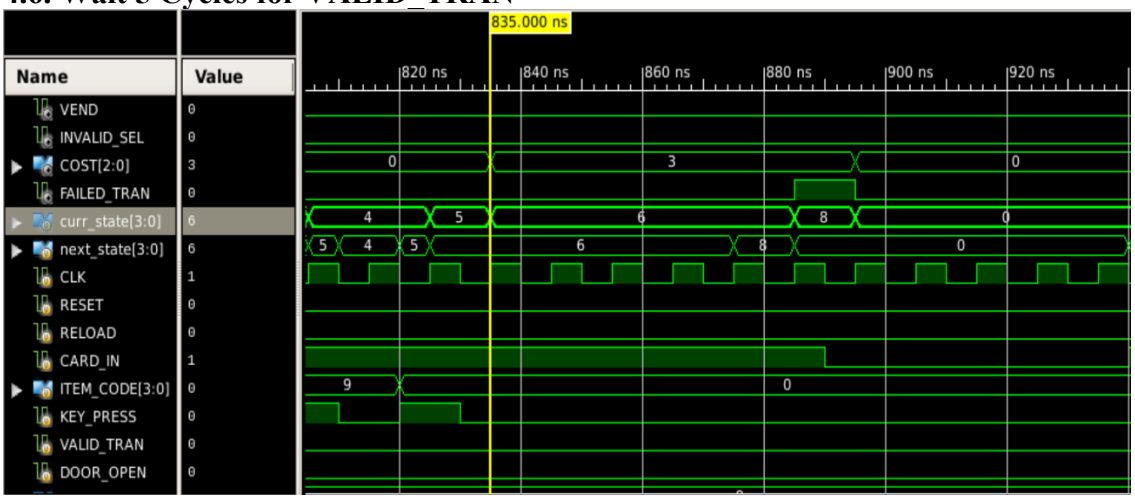


Waveform 5: *Wait 5 Cycles for Second Item Code with 100 MHz Clock*– CARD_IN signal goes high at 660 ns.

- Since KEY_PRESS goes high at 670 ns, we can get the first digit and curr_state is set to S_ONES_DIGIT (4)

- In S_ONES_DIGIT, wait for the second item code (i.e. for KEY_PRESS to go high)
- Since KEY_PRESS does not go high from 675 ns to 725 ns (5 clock cycles), curr_state goes back to S_IDLE (0)

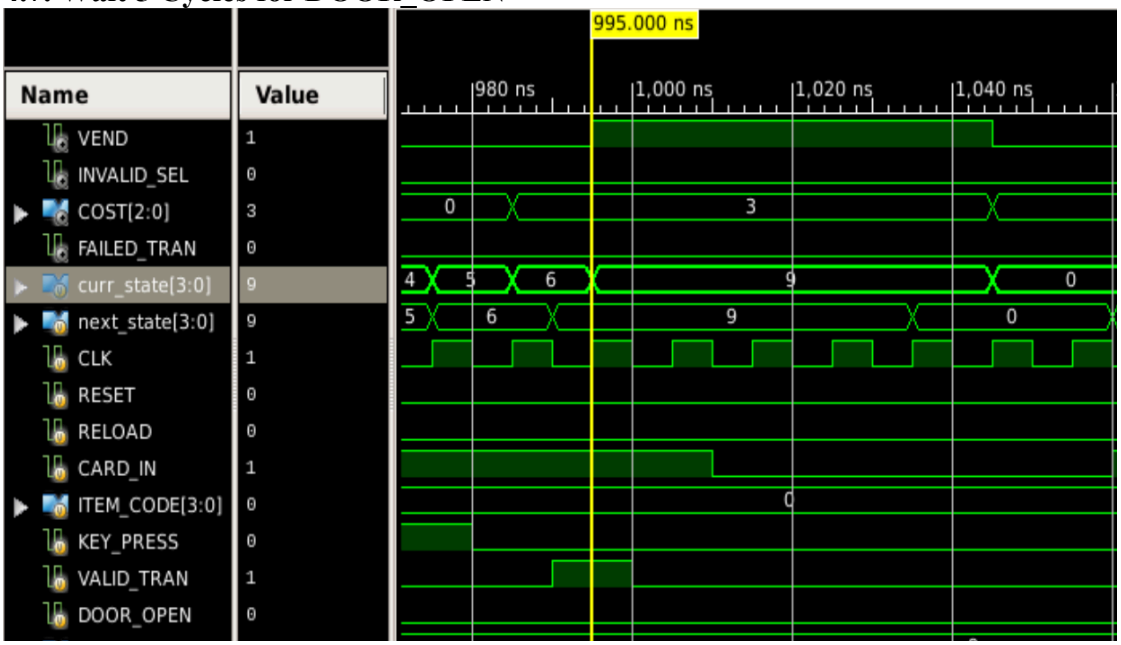
4.6. Wait 5 Cycles for VALID_TRAN



Waveform 6: *Wait 5 Cycles for S_VALID_TRAN with 100 MHz Clock*– S_VALID state begins at 835 ns and ends at 885 ns (five 10 period cycles).

- The FSM successfully processes the two ITEM_CODE (corresponding to item number 9), setting the current state to S_VALID (6) and the COST to 3
- In S_VALID, we wait for the VALID_TRAN to come before moving on to S_VEND(9)
- However, VALID_TRAN does not go high from 835 ns to 885 ns (5 cycles), so curr_state is set to S_FAILED_TRAN and FAILED_TRAN is set to 1

4.7. Wait 5 Cycles for DOOR_OPEN

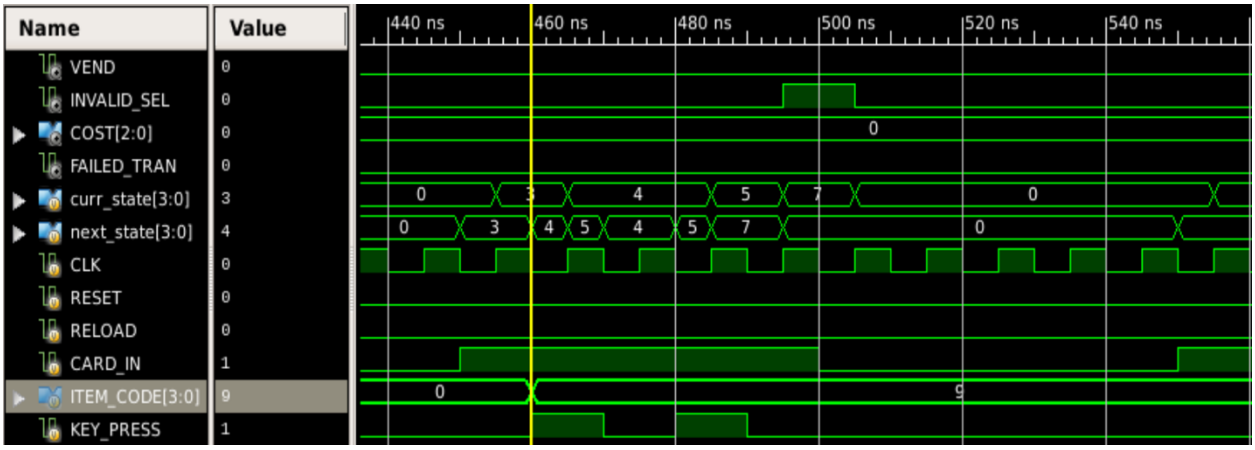


Waveform 7: *Wait 5 Cycles for S_VALID_TRAN with 100 MHz Clock*– S_VEND state begins at 995 ns and ends at 1045 ns (five 10 period cycles).

- The FSM is at curr_state = S_VEND (9) at 995 ns

- Since DOOR_OPEN signal does not go high from 995 ns to 1045 ns (5 clock cycles), we go back to S_IDLE

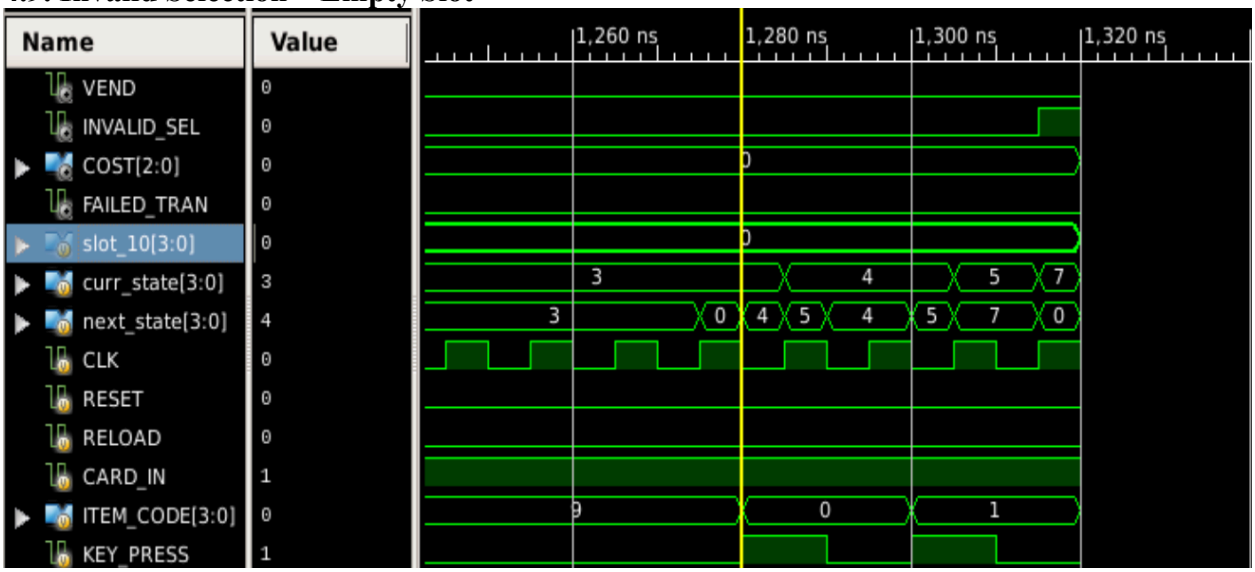
4.8. Invalid Selection – Item Number 99



Waveform 8: *Invalid Selection with Out of Bounds Item Number with 100 MHz Clock*

- The FSM processes the ITEM_CODE selected, which is 99
- 99 is out of bounds so the curr_state goes from S_TENS_DIGIT (5) to S_INVALID (7)
- INVALID_SEL is set to 1 during the S_INVALID to indicate that the input is incorrect

4.9. Invalid Selection – Empty Slot



Waveform 9: *Invalid Selection with Empty Slot with 100 MHz Clock*

- The FSM processes the ITEM_CODE selected, which is 10
- slot_10 is empty (0), so the selection is invalid
- The curr_state goes from S_TENS_DIGIT (5) to S_INVALID (7) at 1315 ns
- INVALID_SEL is set to 1

5. Design Summary

5.1. Synthesis Design Summary

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

Top Level Output File Name         : vending_machine.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                               : 292
#      GND                           : 1
#      INV                           : 1
#      LUT1                : 2
#      LUT2                          : 15
#      LUT3                          : 14
#      LUT4                          : 31
#      LUT5                          : 67
#      LUT6                          : 138
#      MUXCY                         : 10
#      MUXF7                         : 1
#      VCC                           : 1
#      XORCY               : 11
# FlipFlops/Latches        : 113
#      FD                  : 81
#      FDR                 : 10
#      LD                  : 22
# Clock Buffers                      : 1
#      BUFGP                         : 1
# IO Buffers                         : 16
#      IBUF                : 10
#      OBUF                          : 6

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

Selected Device : 6slx16csg324-3


Slice Logic Utilization:
 Number of Slice Registers:           107  out of  18224     0%
 Number of Slice LUTs:                268  out of   9112     2%
    Number used as Logic:             268  out of   9112     2%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    273
   Number with an unused Flip Flop:     166  out of    273    60%
   Number with an unused LUT:             5  out of    273     1%
   Number of fully used LUT-FF pairs:   102  out of    273    37%
   Number of unique control sets:        12

IO Utilization:
 Number of IOs:                  17
 Number of bonded IOBs:               17  out of    232     7%
    IOB Flip Flops/Latches:            6

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

5.2. Mapping Report Design Summary

```
Design Summary
--------------
Number of errors:     0
Number of warnings:   7
Slice Logic Utilization:
  Number of Slice Registers:              108 out of  18,224    1%
    Number used as Flip Flops:             91
    Number used as Latches:                16
    Number used as Latch-thrus:             0
    Number used as AND/OR logics:           1
  Number of Slice LUTs:                   228 out of   9,112    2%
    Number used as logic:                 228 out of   9,112    2%
      Number using O6 output only:        185
      Number using O5 output only:          2
      Number using O5 and O6:              41
      Number used as ROM:                   0
    Number used as Memory:                  0 out of   2,176    0%

Slice Logic Distribution:
  Number of occupied Slices:              80 out of   2,278    3%
  Number of MUXCYs used:                  12 out of   4,556    1%
  Number of LUT Flip Flop pairs used:     238
    Number with an unused Flip Flop:      132 out of    238   55%
    Number with an unused LUT:             10 out of    238    4%
    Number of fully used LUT-FF pairs:     96 out of    238   40%
    Number of unique control sets:          8
    Number of slice register sites lost
      to control set restrictions:        37 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:                  17 out of    232    7%
    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:               4.28

Peak Memory Usage:  763 MB
Total REAL time to MAP completion:  14 secs
Total CPU time to MAP completion:   12 secs
```

5.3. Design Summary Analysis

Both our synthesis report and our mapping report were able to be generated without any errors. There were, however, a total of 51 warnings. The synthesis reported have warnings about bit truncation and latches. Bit truncation should not be a problem since our code accounts for it. Latches were generated from the internal register used in our combinatorial logic block for next states to keep track of number of cycles that have elapsed. This may cause timing issues when we are operating at faster clocks speeds. However, for the specifications of this project, the latches should be okay.

The number of slice registers and slice LUTs utilized is also consistent with the contents of the source code. Combinatorial logic and sequential logic are both used in the finite state machine as part of the next state logic and output logic.

6. Conclusion

The purpose of this project was to further consolidate and our understanding of finite state machines and their utility in digital design. In this project, I learned how to create a comprehensive finite state machine to handle multiple user inputs in a systematic manner. The largest challenge with this project, was handling waiting cycles in between inputs. This was done by having another counter run outside of our state machine and resetting the counter at specific states.