# CS M152A - Lab 4, Designing a Vending Machine

Name: Sum Li

UID: 505146702

Due Date: December 13, 2020

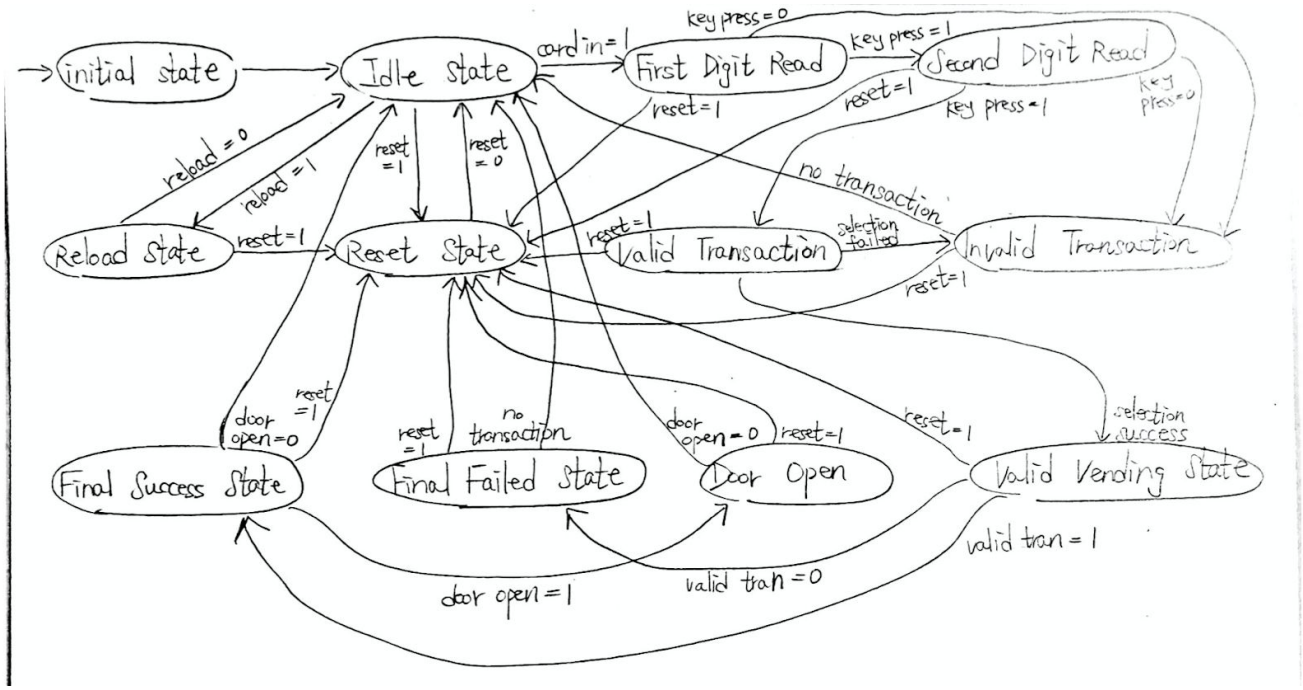
TA: Mohit Garg

## 1) Introduction and Requirement

For this lab, the goal is to use the Xilinx program to design and test a Finite State Machines (FSMs) which simulates as a vending machine when a buyer inserts a card and input two digit code to select corresponding items from the vending machine to purchase. The vending machine takes in 8 inputs and gives out 4 outputs. The inputs signals include: CARD_IN, VALID_TRAN, 4 bits ITEM_CODE, KEY_PRESS, DOOR_OPEN, RELOAD, CLK, RESET. The outputs include VEND, INVALID_SEL, FAILED_TRAN, 3 bits COST. There are a total of 20 different snacks available for selection in the vending machine. The available card slot starts from 00 to 19 which is expressed as BCD in the implementation as 00000 to 10011. Each of the card slots can stock up to 10 units of a specific snack. Each card slot has an individual counter to keep track of the remaining number of snacks units in the corresponding slot. The overview of the entire transaction of the vending machine is: (1) Buyer first insert the card into the machine, (2) Buyer then press the first and second item digit code each within 5 clock cycles, (3) Vending machine checks for valid digit code and clock cycles, (4) Vending machine decrement the correct item by one and wait for the door open signal, (5) Vending machine open the door within the correct clock cycles, (6) Buyer pick up the item and door close.

## 2) Design Requirement

**Finite State Machine Diagram**



**Output Table for Each State**

| State | COST (3 bits) | VEND | INVALID_SEL | FAILED_TRAN |
|---|---|---|---|---|
| Idle State | 000 | 0 | 0 | 0 |
| Reset State | 000 | 0 | 0 | 0 |
| Reload State | 000 | 0 | 0 | 0 |
| First Digit Read State | 000 | 0 | 0 | 0 |
| Second Digit Read State | 000 | 0 | 0 | 0 |
| Valid Transaction State | 000 | 0 | 0 | 0 |
| Invalid Transaction State | 000 | 0 | 1 | 0 |
| Valid Vending State | Cost of selected item | 0 | 0 | 0 |
| Final Success Transaction State | Cost of selected item | 1 | 0 | 0 |
| Door Open State | Cost of selected item | 0 | 0 | 0 |
| Final Failed Transaction State | Cost of selected item | 0 | 0 | 1 |

**Special Requirements of the Implementation**

There are several special characteristics and requirements of the vending machine. First, the buyer can only purchase 1 item at a time. Second, the machine only accepts payment by card. Third, the machine needs to refill all the snacks during the reset process. Third, the machine needs to reject invalid transactions such as selected items are below 1 units, digit item code is not in the valid range, second digit code is not entered within 5 clock cycles and corresponding signals are not set to high within 5 clock cycles. These situations are seen as failed transactions.

**Logic of the Implementation**

1. **Reset State**
    a. First, check if my current RESET signal = 0, if yes, then the next state is idle state. If not, no change of state need to be made
    b. I need to reset my rst_signal and proceed_signal for my five clock cycles counter
    c. I need to reset my first digit item code register to 4 bits of zero
    d. I need to reset my second digit item code register to 4 bits of zero
2. **Idle State**
    a. First, check if my RELOAD signal is high, if yes, then the next state is the reload state. Else if check if my RESET signal is high, if yes, then the next state is the reset state. Else if my CARD_IN signal is high, if yes, then the next state is the first digit item code reading state. If not, no change of state needs to be made.
    b. I need to reset my rst signal for my five clock cycles counter
    c. I need to reset my first digit item code register to 4 bits of zero
    d. I need to reset my second digit item code register to 4 bits of zero
3. **Reload State**
    a. First, check if my RELOAD signal is = 0, if yes, then the next state is the idle state. Else if RESET signal is high, if yes, then the next state is the reset state. If not, no change of state needs to be made.
    b. All the snacks in the machine are each reload with 10 units
4. **First Digit Item Code Reading**
    a. First, I need to run and check my five clock cycles counter for the purpose of monitor the time it takes for the KEY_PRESS signal to be high
    b. Check if my RESET signal is high, if yes, then the next state is the reset state. Else if check if my KEY_PRESS signal is high, if yes, then the next state is the second digit item code reading state. Else if my KEY_PRESS signal is low and the pass five clock cycles, then the next state is invalid transaction state. If not, no change of state needs to be made.
5. **Second Digit Item Code Reading**
    a. First, I need to run and check my five clock cycles counter for the purpose of monitor the time it takes for the KEY_PRESS signal to be high

b. Check if my RESET signal is high, if yes, then the next state is the reset state. Else if check if my KEY_PRESS signal is high, if yes, then the next state is valid transaction state. Else if check if my KEY_PRESS signal is low, if yes, then the next state is invalid transaction state. If not, no change of state needs to be made.

6. **Valid Transaction State**
    a. First, check if my RESET signal is high, if yes, then the next state is reset state.
    b. If the RESET signal is low, then get the slot number (calculated by: first digit code * 10 + second digit code) and check if the units of items are below 1 or not. If units of item for the selected snack is below 1, then the next state is invalid transaction state. Also, check if the slot number is within the valid range which is between 0 and 19, if not, then the next state is invalid transaction state.
    c. If all of the above is good, then the next state is a valid vending state.
7. **Invalid Transaction State**
    a. First, set the INVALID_SEL to 1 to indicate a invalid transaction.
    b. Check if my RESET signal is high, if yes, then the next state is reset state. If not, then the next state is the idle state.
8. **Valid Vending State**
    a. First, COST is set to the corresponding cost of the selected snack from the vending machine
    b. Then, I need to run and check my five clock cycles counter for the purpose of monitor the time it takes for the VALID_TRAN signal to be high
    c. Check if my VALID_TRAN signal is high, if yes, then the next state is the final success transaction state. If VALID_TRAN signal is low, then the next state is the final failed transaction state. If not, no change of state needs to be made.
9. **Final Success Transaction State**
    a. First, the VEND signal is set to high and then the units of item for the selected snack is decremented by 1.
    b. Then, I need to run and check my five clock cycles counter for the purpose of monitor the time it takes for the DOOR_OPEN signal to be high
    c. Check if my RESET signal is high, if yes, then the next state is reset state. Then, check if my DOOR_OPEN signal is high, if yes, then the next state is the door

open state. If my DOOR_OPEN signal is low after 5 clock cycles, then the next state is idle state. If not, no change of state needs to be made.

**10. Door Open State (Item Pick up)**

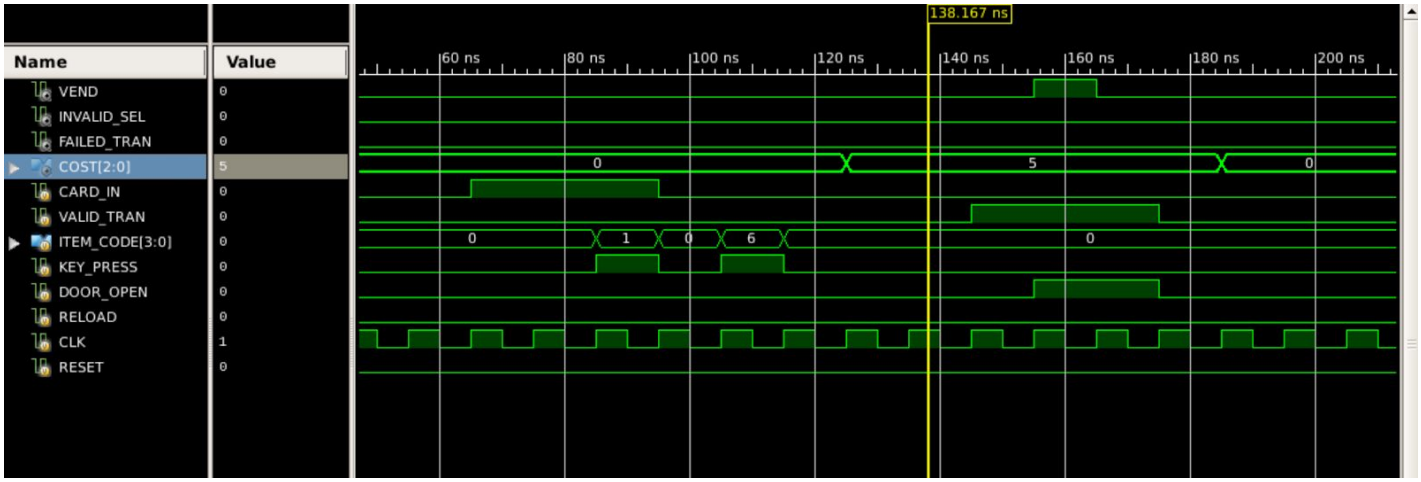
a. First, the VEND signal is set to low.
b. Check if my RESET signal is high, if yes, then the next state is reset state. Then, check if my DOOR_OPEN signal is high, if yes, then the next state is the idle state. If not, no change of state needs to be made.

**11. Final Failed Transaction State**

a. First, the FAILED_TRAN signal is set to high.
b. Check if my RESET signal is high, if yes, then the next state is reset state. If not, then the next state is the idle state.

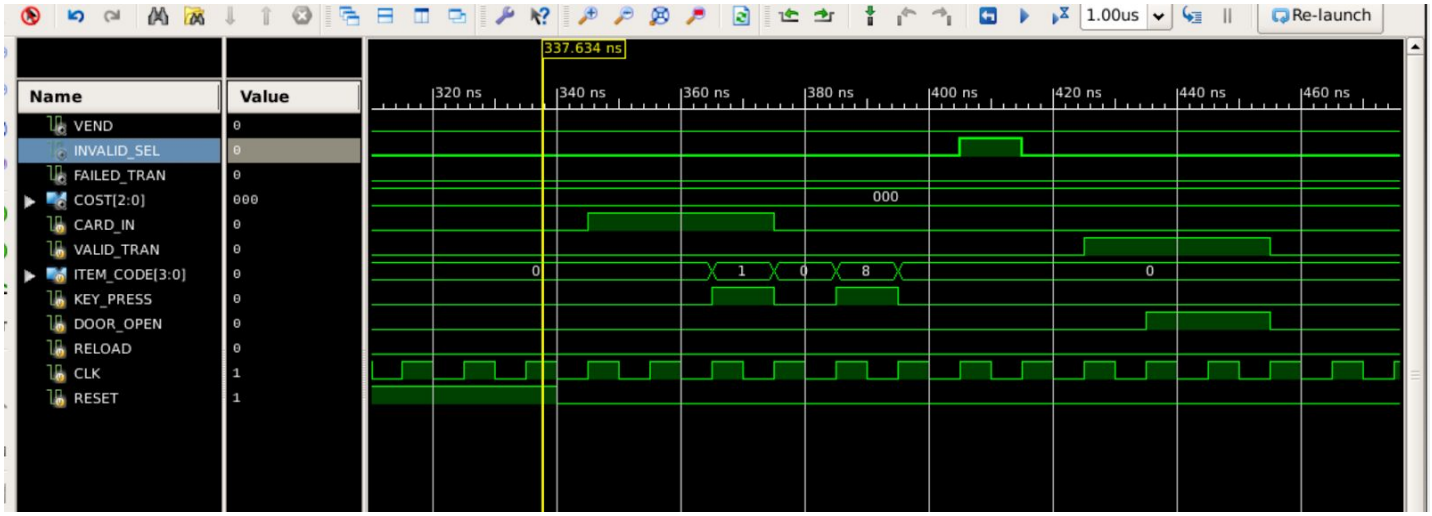
### 3) Simulation Documents

**Test Case 1: Successful Transaction when buyer selects item 16**



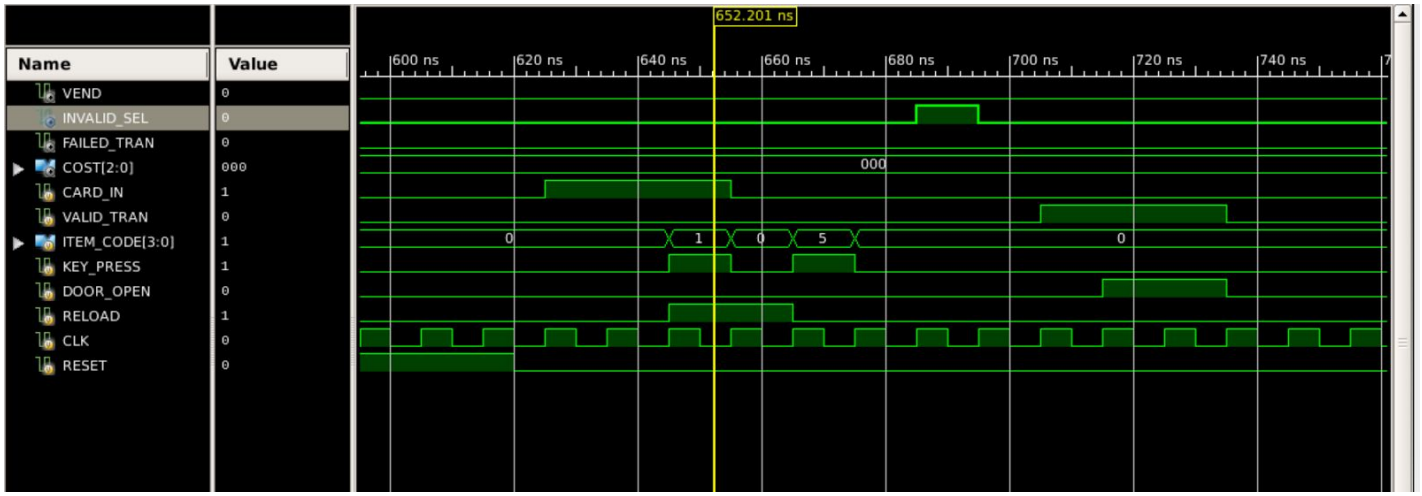

Test case 1 is a successful transaction when the buyer enters item code 16 to purchase the selected snacks. Before reading the first digit item code, the card is already inserted as the CARD_IN signal is set to high. Both of the first and second digit item code is read when the KEY_PRESS signal is high. User enters item code 16 which then translates into a cost of 5 that is shown on the waveform diagram. After 1 clock cycle of the reading of the second digit code, the corresponding cost is shown on the diagram. After the VALID_TRAN signal is being set to high, the DOOR_OPEN signal is also set to high within the 5 clock cycles. Therefore, this is an example of a valid transaction.

**Test Case 2: Failed Transaction when selected item is below 1 unit (zero number of items left in machine correspond to the code)**



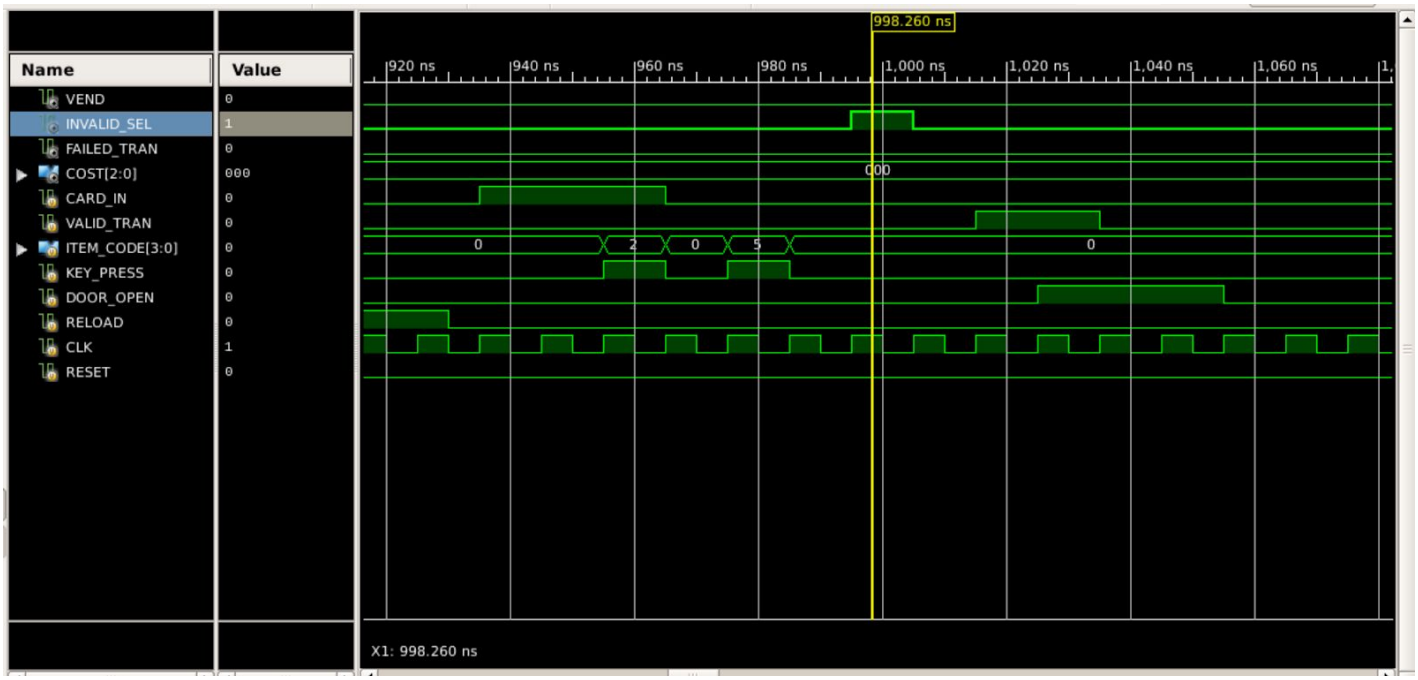

Test case 2 is a failed transaction when the buyer enters item code 18 to purchase the selected snacks. Before reading the first digit item code, the card is already inserted as the CARD_IN signal is set to high. Both of the first and second digit item code is read when the KEY_PRESS signal is high. Before reading the first and second digit item code, the RESET signal is being set to high to reset all the number of snacks to 0, but I do not set the RELOAD signal to high to reload each of snacks to 10. Therefore, when a buyer enters code 18 to purchase the selected snack, it is not available to purchase. So, the INVALID_SEL signal is being set to high as being shown in the waveform diagram.

**Test Case 3: Failed Transaction when Buyer tries to enter digit code during a reload state (reload is not happen in idle state)**



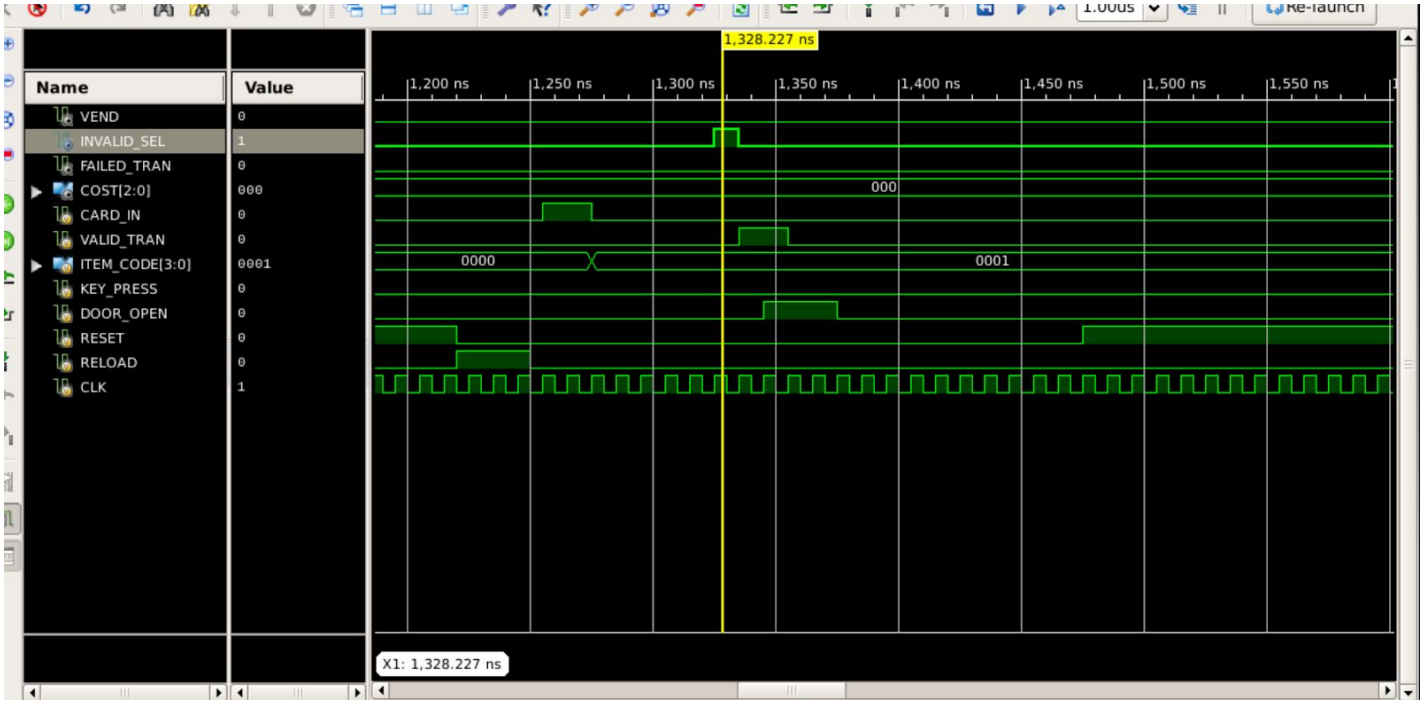

Test case 3 is a failed transaction when the buyer enters item code 15 to purchase the selected snacks. Before reading the first digit item code, the RESET signal is being set to high. Both of the first and second digit item code is read when the KEY_PRESS signal is high. However, before the reading of the first digit item code, the RELOAD signal is still not being set to high. The RELOAD signal is being set to high during the read of first digit item code instead. This means that item code 15 snack does not have enough items (units of item = 0) to sell when the user enters the first digit code. So, the INVALID_SEL signal is being set to high as being shown in the waveform diagram.

**Test Case 4: Failed Transaction when Buyer enters an out of range item code (not from 00 to 19)**



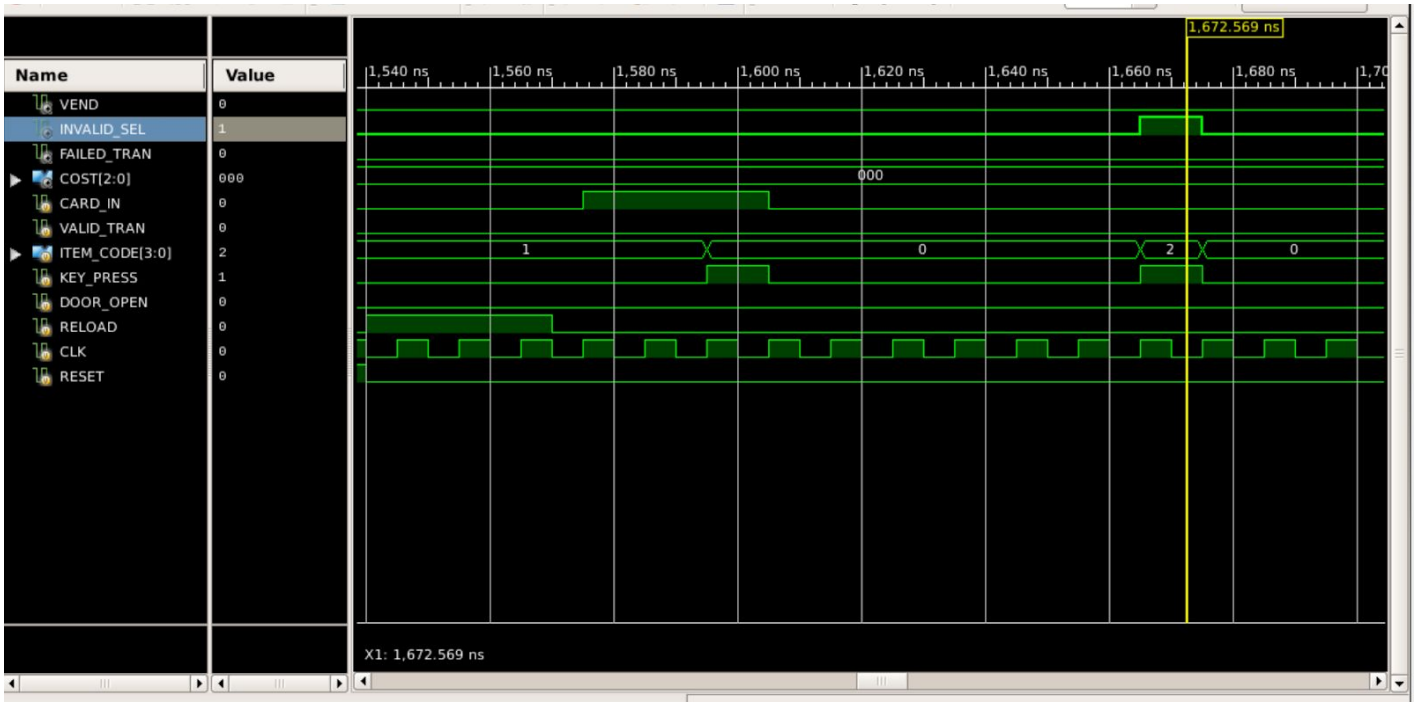

Test case 4 is a failed transaction when the buyer enters item code 25 to purchase the selected snacks. Both of the first and second digit item code is read when the KEY_PRESS signal is high. However, the overall item code 25 is not within the valid range which is from 00 to 19. So, the INVALID_SEL signal is being set to high right after the reading of the second digit item code as being shown in the waveform diagram.

**Test Case 5: Failed Transaction when Buyer enters digit item code when the KEY_PRESS is not high**



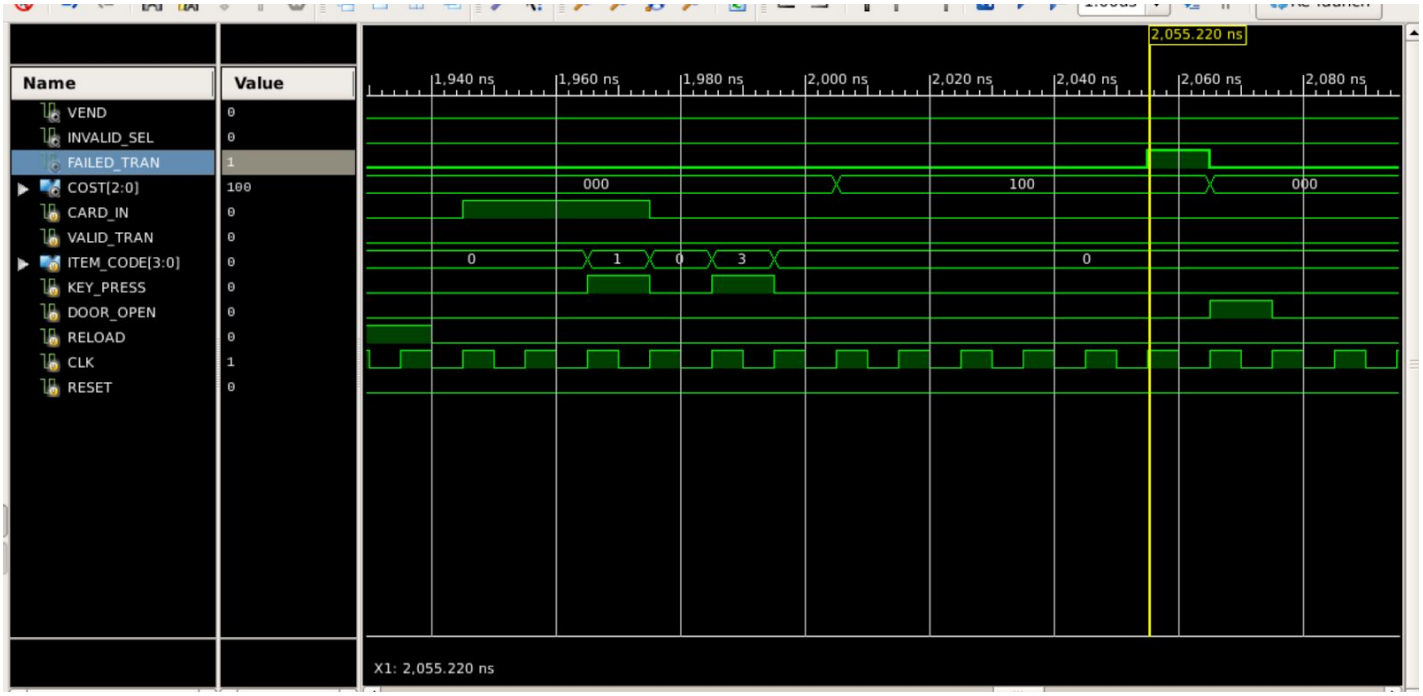

Test case 5 is a failed transaction when the buyer enters item code 11 to purchase the selected snacks. I do not set the KEY_PRESS signal to be high in the testbench. Both of the first and second digit item code is read when the KEY_PRESS signal is low. Therefore, both of the item codes are not a valid read. So, the INVALID_SEL signal is being set to high right after the reading of the first digit item code as being shown in the waveform diagram.

**Test Case 6: Failed Transaction when Buyer enter the second digit item code after 5 clock cycles**



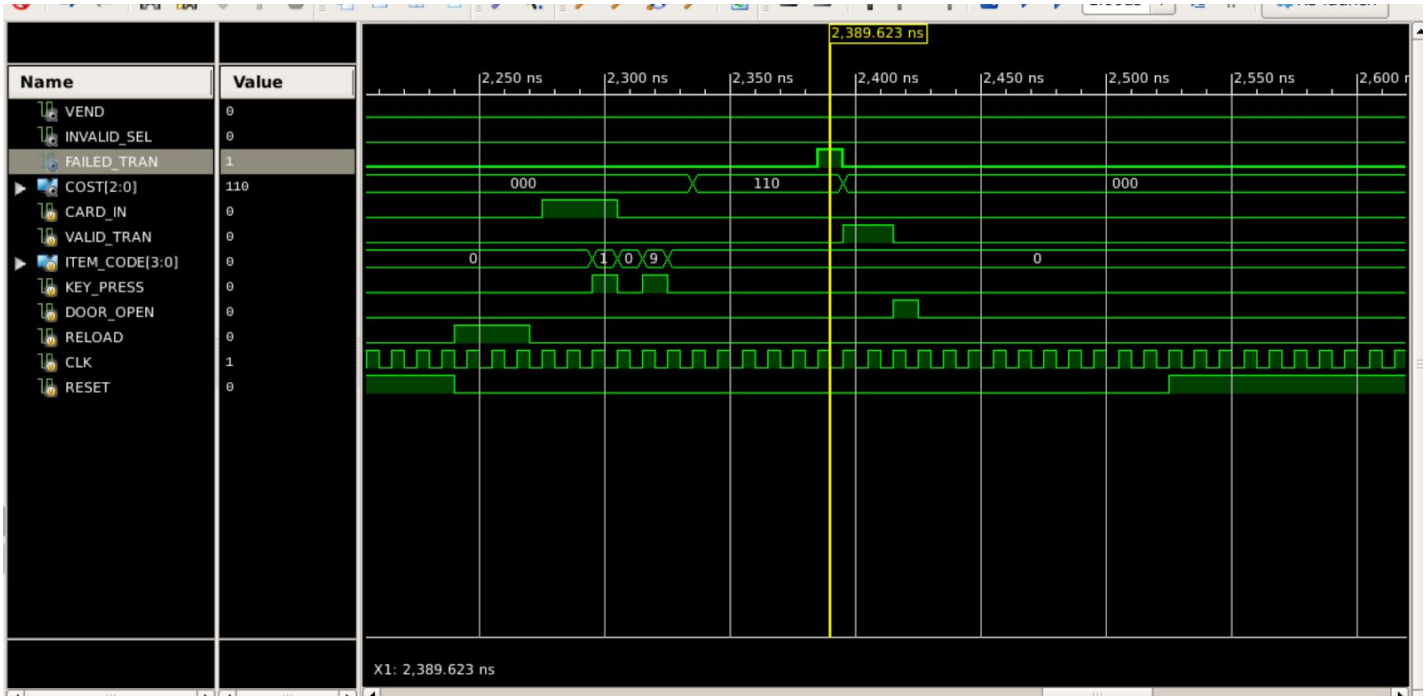

Test case 6 is a failed transaction when the buyer enters item code 12 to purchase the selected snacks. After reading the first digit item code, the second digit item is not read within 5 clock cycles. It is being read after 6 clock cycles. Therefore, the second read of the digit item code is not a valid read. So, the INVALID_SEL signal is being set to high right after the reading of the second digit item code as being shown in the waveform diagram.

**Test Case 7: Failed Transaction after valid code selection and correct number of sale items, but the VALID_TRAN signal does not become high within 5 clock cycles**



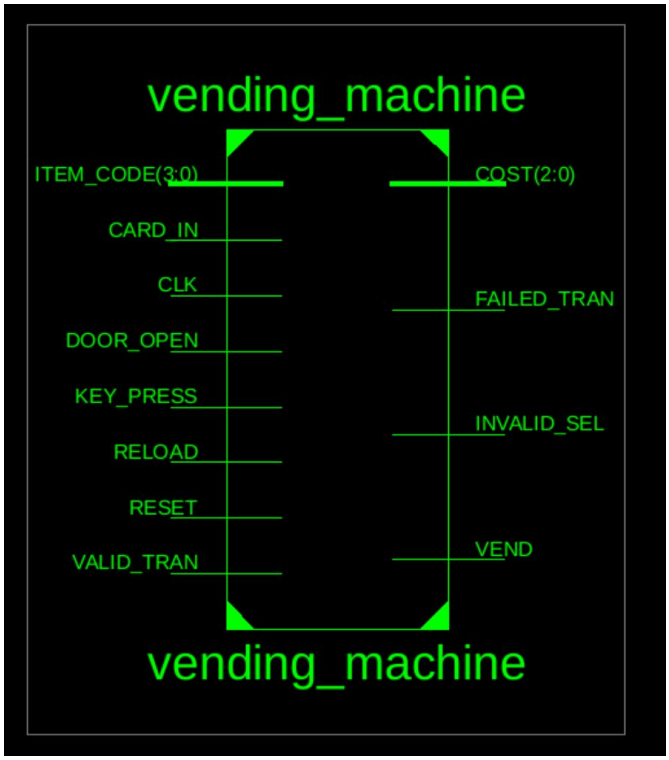

Test case 7 is a failed transaction when the buyer enters item code 13 to purchase the selected snacks. Before reading the first digit item code, the card is already inserted as the CARD_IN signal is set to high. Both of the first and second digit item code is read when the KEY_PRESS signal is high. User enters item code 13 which then translates into a cost of 4 (BCD: 100) that is shown on the waveform diagram. After 1 clock cycle of the reading of the second digit code, the corresponding cost is shown on the diagram. However, the VALID_TRAN signal does not go high within 5 clock cycles. Therefore, the FAILED_SEL signal is being set to high as being shown in the waveform diagram.

**Test Case 8: Failed Transaction after valid code selection and correct number of sale items, but the VALID_TRAN signal become high after 5 clock cycles**




Test case 7 is a failed transaction when the buyer enters item code 19 to purchase the selected snacks. Before reading the first digit item code, the card is already inserted as the CARD_IN signal is set to high. Both of the first and second digit item code is read when the KEY_PRESS signal is high. User enters item code 19 which then translates into a cost of 6 (BCD: 110) that is shown on the waveform diagram. After 1 clock cycle of the reading of the second digit code, the corresponding cost is shown on the diagram. However, the VALID_TRAN signal goes high after 5 clock cycles. Therefore, the FAILED_SEL signal is being set to high as being shown in the waveform diagram.

**High Level Schematics**




The high level schematic matches with the given inputs and output give on the instructions. There are a total of 8 inputs: 4 bits ITEM_CODE, 1 bit CARD_IN, 1 bit CLK, 1 bit DOOR_OPEN, 1 bit KEY_PRESS, 1 bit RELOAD, 1 bit RESET, 1 bit VALID_TRAN. There are a total of 4 inputs: 3 bits cost, 1 bit FAILED_TRAN, 1 bit INVALID_SEL, 1 bit VEND.

# ISE Design Summary Report

| vending_machine Project Status (12/07/2020 - 06:53:26) | | | |
|---|---|---|---|
| **Project File:** | vending_machine.xise | **Parser Errors:** | No Errors |
| **Module Name:** | vending_machine | **Implementation State:** | Programming File Generated |
| **Target Device:** | xc6slx16-3csg324 | • **Errors:** | No Errors |
| **Product Version:** | ISE 14.7 | • **Warnings:** | 121 Warnings (16 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 | 102 | 18,224 | 1% | |
| Number used as Flip Flops | 8 | | | |
| Number used as Latches | 94 | | | |
| Number used as Latch-thrus | 0 | | | |
| Number used as AND/OR logics | 0 | | | |
| Number of Slice LUTs | 90 | 9,112 | 1% | |
| Number used as logic | 90 | 9,112 | 1% | |
| Number using O6 output only | 72 | | | |
| Number using O5 output only | 0 | | | |
| Number using O5 and O6 | 18 | | | |
| Number used as ROM | 0 | | | |
| Number used as Memory | 0 | 2,176 | 0% | |
| Number used as Memory | 0 | 2,176 | 0% | |
| Number of occupied Slices | 50 | 2,278 | 2% | |
| Number of MUXCYs used | 4 | 4,556 | 1% | |
| Number of LUT Flip Flop pairs used | 144 | | | |
| Number with an unused Flip Flop | 42 | 144 | 29% | |
| Number with an unused LUT | 54 | 144 | 37% | |
| Number of fully used LUT-FF pairs | 48 | 144 | 33% | |
| Number of unique control sets | 28 | | | |
| Number of slice register sites lost to control set restrictions | 122 | 18,224 | 1% | |
| Number of bonded IOBs | 17 | 232 | 7% | |
| IOB Latches | 6 | | | |
| Number of RAMB16BWERs | 0 | 32 | 0% | |
| Number of RAMB8BWERs | 0 | 64 | 0% | |
| Number of BUFIO2/BUFIO2_2CLKs | 0 | 32 | 0% | |
| Number of BUFIO2FB/BUFIO2FB_2CLKs | 0 | 32 | 0% | |
| Number of BUFG/BUFGMUXs | 2 | 16 | 12% | |
| Number used as BUFGs | 2 | | | |
| Number used as BUFGMUX | 0 | | | |
| Number of DCM/DCM_CLKGENs | 0 | 4 | 0% | |
| Number of ILOGIC2/ISERDES2s | 0 | 248 | 0% | |
| Number of IODELAY2/IODRP2/IODRP2_MCBs | 0 | 248 | 0% | |
| Number of OLOGIC2/OSERDES2s | 6 | 248 | 2% | |
| Number used as OLOGIC2s | 6 | | | |
| Number used as OSERDES2s | 0 | | | |
| Number of BSCANs | 0 | 4 | 0% | |

| | | | | |
|---|---|---|---|---|
| Number of ILOGIC2/ISERDES2s | 0 | 248 | 0% | |
| Number of IODELAY2/IODRP2/IODRP2_MCBs | 0 | 248 | 0% | |
| Number of OLOGIC2/OSERDES2s | 6 | 248 | 2% | |
| Number used as OLOGIC2s | 6 | | | |
| 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 | 32 | 0% | |
| Number of ICAPs | 0 | 1 | 0% | |
| Number of MCBs | 0 | 2 | 0% | |
| Number of PCILOGICSEs | 0 | 2 | 0% | |
| Number of PLL_ADVs | 0 | 2 | 0% | |
| Number of PMVs | 0 | 1 | 0% | |
| Number of STARTUPs | 0 | 1 | 0% | |
| Number of SUSPEND_SYNCs | 0 | 1 | 0% | |
| Average Fanout of Non-Clock Nets | 3.37 | | | |

| 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 | Mon Dec 7 06:52:33 2020 | 0 | 105 Warnings (0 new) | 2 Infos (0 new) |


| | | | | |
|---|---|---|---|---|
| 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 | 3.37 | | | |

| 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 | Mon Dec 7 06:52:33 2020 | 0 | 105 Warnings (0 new) | 2 Infos (0 new) |
| Translation Report | Current | Mon Dec 7 06:52:39 2020 | 0 | 0 | 0 |
| Map Report | Current | Mon Dec 7 06:52:49 2020 | 0 | 8 Warnings (8 new) | 6 Infos (6 new) |
| Place and Route Report | Current | Mon Dec 7 06:52:58 2020 | 0 | 0 | 3 Infos (3 new) |
| Power Report | | | | | |
| Post-PAR Static Timing Report | Current | Mon Dec 7 06:53:03 2020 | 0 | 0 | 4 Infos (4 new) |
| Bitgen Report | Current | Mon Dec 7 06:53:24 2020 | 0 | 8 Warnings (8 new) | 0 |

| Secondary Reports | | [-] |
|---|---|---|
| **Report Name** | **Status** | **Generated** |
| ISIM Simulator Log | Out of Date | Mon Dec 7 06:50:55 2020 |
| WebTalk Report | Current | Mon Dec 7 06:53:25 2020 |
| WebTalk Log File | Current | Mon Dec 7 06:53:26 2020 |

**Date Generated:** 12/07/2020 - 06:53:26

According to the summary and utilization report, I have used 102 registers for storing variables and outputs. I have also used if else statements which are shown by the number of comparators and flip flops.

## 4) Conclusion

After I have finished this lab, I understand how to use Verilog and Xilinx ISE programs to design and FSM model to simulate the vending machine. Through this project, I am able to have a better

understanding of FSM and to use that idea to design and simulate a real world system. One of the difficulties that I have is to figure out which state to go after certain signals are on. However, I am able to figure it out by first drawing the FSM diagram and then write a rough listing of the implementation logic before implementation. Second, it takes me awhile to figure out some interesting cases to test in the testbench file. However, I am able to figure it out with the help of detailed descriptions of the instructions and the QA sessions with instructor which further answers my doubt about the project.