Design and Simulation of Vending Machine

Prof. Rutu Parekh
DAIICT, Gandhinagar, Gujarat

Dhriti Goenka
202201041@daiict.ac.in

Divyesh Ramani
202201241@daiict.ac.in

***Abstract*—This paper outlines the development of a digital vending machine using Verilog HDL. The system is capable of dispensing four different products, each assigned a cost of 15, 20, 25, or 30 rupees. It only accepts coins valued at 5 and 10 rupees. Users make a selection and then insert coins one at a time, and if excess money is inserted, the machine returns the balance. The entire process is managed by a Finite State Machine (FSM), which handles item selection, coin recognition, change return, and product delivery. The design was simulated and tested using Verilog-based tools, and its behaviour was verified using FSM state diagrams, RTL representations, and synthesis through QFLOW and EDA Playground. The project successfully demonstrates a practical application of digital design using FSM concepts.**

I. Introduction

Vending machines have become a common part of our everyday life. We see them almost everywhere—in schools, colleges, offices, railway stations, airports, shopping malls, and hospitals. They are used to quickly and easily buy snacks, drinks, tickets, or even medicines without the need for a shopkeeper. These machines help save time, reduce the need for staff, and work 24/7, making them very useful in today's fast-moving world.

The main idea behind a vending machine is simple: a user inserts money, selects an item, and the machine delivers the item. If the user gives more money than needed, the machine returns the remaining balance. While the task seems simple on the outside, the internal working of a vending machine involves careful planning and logic. The machine must be able to identify the amount inserted, compare it with the item cost, process the user's selection, and give back the correct change if needed.

In this project, we have designed and simulated a simple digital vending machine using Verilog Hardware Description Language (HDL). The system accepts coins of 5 and 10 rupees and offers four items priced at 15, 20, 25, and 30 rupees. The user first selects the item he has to buy. Then he keeps inserting coins until there is enough balance. The machine also checks if the balance is sufficient every time a coin is inserted. If yes, it dispenses the item and returns any extra coins as a change.

The logic of the vending machine is built using a Finite State Machine (FSM). An FSM is a model used in digital design to control the flow of operations step by step. In our case, the FSM handles actions such as detecting coins, adding the total, checking item cost, delivering the item, and calculating the change. Using FSM helps to make the system organised and easy to understand. It also makes it easier to test and debug the system.

To develop and test our design, we used two main tools—EDA Playground and Qflow. EDA Playground allowed us to write and simulate Verilog code online, which helped us test our logic, observe waveforms, and debug errors in a fast and interactive environment. Once the simulation was successful, we used Qflow, an open-source digital synthesis flow, to convert our Verilog code into gate-level representations. Qflow helped us generate RTL schematics, and FSM state diagrams, and understand how our design would behave in actual hardware. These tools were crucial in verifying the accuracy and completeness of our project and gave us a deeper insight into the real-world process of ASIC development and digital system verification.

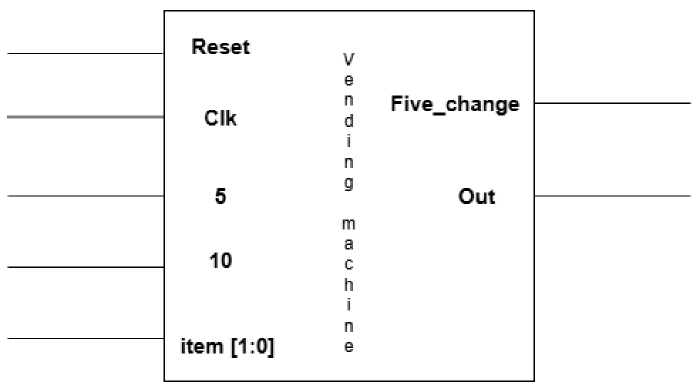

Fig. 1. Block diagram of vending machine

II. FSM of Vending Machine

The control logic of the vending machine is implemented using a modular FSM architecture. This design consists of two independent but interacting FSMs:

A. Item Selection FSM

The Item Selection FSM is responsible for allowing the user to choose a product. It starts from a central Base State and transitions to one of the four-item states—Item-1, Item-2, Item-3, or Item-4—based on a 2-bit binary input. This design simplifies selection handling and cleanly separates product choice from transaction logic.

Input Mapping:

- 00 - Select Item 1
- 01 - Select Item 2
- 10 - Select Item 3
- 11 - Select Item 4

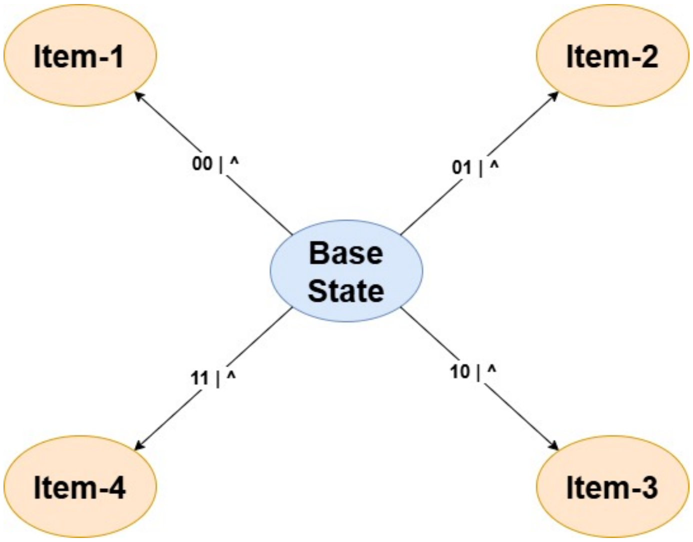

Fig. 2. FSM for item selection

Each transition is labelled in the form input—output. The output is ˆ always because this FSM does not give any output.

B. Payment and Dispensing FSM

The Payment FSM handles coin input, balance calculation, product dispensing, and change returns. It supports coins of 5 and 10 rupees, with item prices fixed at 15, 20, 25, and 30 rupees. FSM States:

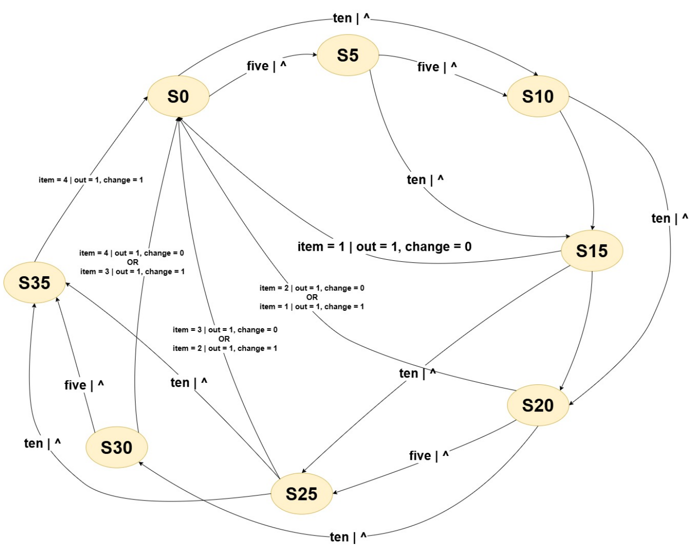

Fig. 3. FSM for Payment Processing and Item Dispensing

- S0(000): Initial state – Balance = 0
- S5(001): Balance = 5
- S10(010): Balance = 10
- S15(011): Balance = 15. Item 1 is dispensed if selected.
- S20(100): Balance = 20. Item 2 is dispensed if selected. Change is returned if item 1 was selected.
- S25(101): Balance = 25. Item 3 is dispensed if selected. Change is returned if item 2 was selected.
- S30(110): Balance = 30. Item 4 is dispensed if selected. Change is returned if item 3 was selected.
- S35(111): Balance = 35. Change is returned if item 4 was selected.

Transition Logic:
Input is either 5 or 10. On reaching or exceeding the selected item's price:
An output out = 1, change = x is triggered. FSM dispenses the item and returns any excess amount. FSM resets to S0 after a successful transaction.

Output Format:

- out = 1, change = 0: Exact payment, item dispensed
- out = 1, change = 1: Overpayment, item and change dispensed
- out = 0, change = 0: Waiting for more input

Example: If a user selects Item-2 (20 units) and inserts 10 + 10, FSM transitions: S0 $\rightarrow$ S10 $\rightarrow$ S20, triggers output 1,0, and resets.

C. Integrated System Behavior

The overall vending machine operates as follows:
User selects an item (handled by Item Selection FSM).
The user inserts coins (handled by Payment FSM).
On sufficient payment, the item is dispensed and change is returned (if applicable).
The system resets and waits for the next transaction.

This dual-FSM setup ensures modularity, simplifies debugging and enhances reusability. Additional features—such as adding items or new coin values—can be easily incorporated with minimal changes.

III. Verilog Implementation

A. Module Summary

This Verilog module simulates a vending machine based on a Finite State Machine (FSM).

B. Inputs and Outputs

- `clk` – Clock signal used to synchronize state updates.
- `reset` – Asynchronous signal to reset the system to its initial condition.
- `item` – A 2-bit input that specifies the selected product and its cost:
  - `2'b00` $\rightarrow$ 15
  - `2'b01` $\rightarrow$ 20
  - `2'b10` $\rightarrow$ 25
  - `2'b11` $\rightarrow$ 30
- `five` – Indicates the insertion of a 5 coin.
- `ten` – Indicates the insertion of a 10 coin.
- `out` – Output signal that goes high when the item is dispensed.
- `five_change` – Output signal that indicates the return of 5 as change.

C. Defined Parameters

These constants represent the different states of the FSM, each reflecting the total amount of money inserted so far:

- S0 to S35 correspond to 0 to 35 in increments of 5.

D. Registers and Internal Signals

- price – A 5-bit register storing the cost of the chosen item.
- c_state – Register holding the current state of the FSM.
- n_state – Register representing the next state based on current inputs.

E. Operational Details

1) **Determining Item Cost:**
   - A combinational logic block assigns a value to price based on the input item.
2) **State Progression:**
   - At each positive clock edge or if reset is triggered, the current state is updated to the next state.
3) **FSM Functionality:**
   - Another logic block computes the next state (n_state) and controls the out and five_change signals.
   - As coins are inserted, the FSM progresses through states reflecting the cumulative amount.
   - Once the total inserted value meets or exceeds the product cost:
     - out is asserted to release the item.
     - If the user inserted 5 more than needed, five_change is asserted to return the extra 5.
     - Afterward, the FSM returns to the initial state S0.

```
module vending_machine (
    clk, reset, item, five, ten, out, five_change
);
    input clk;
    input reset;
    input [1:0] item;      // 2-bit item selector
                          // (00 to 11)
    input five;
    input ten;        // coin inputs:  5  and  10
    output reg out;
    output reg five_change;

    parameter S0  = 3'b000;
    parameter S5  = 3'b001;
    parameter S10 = 3'b010;
    parameter S15 = 3'b011;
    parameter S20 = 3'b100;
    parameter S25 = 3'b101;
    parameter S30 = 3'b110;
    parameter S35 = 3'b111;

    reg [4:0] price;
    reg [2:0] c_state;
    reg [2:0] n_state;

    // Item price selection based on input
    always @(*) begin
        case (item)
            2'b00: price = 5'd15;
            2'b01: price = 5'd20;
            2'b10: price = 5'd25;
            2'b11: price = 5'd30;
            default: price = 5'd0;
        endcase
    end

    // State update on clock or reset
    always @(posedge clk or posedge reset) begin
        if (reset)
            c_state <= S0;
        else
            c_state <= n_state;
    end

    // FSM logic
    always @(*) begin
        out = 0;
        five_change = 0;
        n_state = c_state;

        case (c_state)
            S0: begin
                if (five)      n_state = S5;
                else if (ten) n_state = S10;
            end

            S5: begin
                if (five)      n_state = S10;
                else if (ten) n_state = S15;
            end

            S10: begin
                if (five)      n_state = S15;
                else if (ten) n_state = S20;
            end

            S15: begin
                if (price == 15) begin
                    out = 1;
                    five_change = 0;
                    n_state = S0;
                end else begin
                    if (five)      n_state = S20;
                    else if (ten) n_state = S25;
                end
            end

            S20: begin
                if (price == 20) begin
                    out = 1;
                    five_change = 0;
                    n_state = S0;
                end else if (price == 15) begin
                    out = 1;
                    five_change = 1;
                    n_state = S0;
                end else begin
                    if (five)      n_state = S25;
                    else if (ten) n_state = S30;
                end
            end

            S25: begin
                if (price == 25) begin
                    out = 1;
                    five_change = 0;
                    n_state = S0;
                end else if (price == 20) begin
                    out = 1;
                    five_change = 1;
                    n_state = S0;
                end else begin
                    if (five)      n_state = S30;
```

```
                    else if (ten) n_state = S35;
                end
            end

            S30: begin
                if (price == 30) begin
                    out = 1;
                    five_change = 0;
                    n_state = S0;
                end else if (price == 25) begin
                    out = 1;
                    five_change = 1;
                    n_state = S0;
                end else begin
                    if (five) n_state = S35;
                end
            end

            S35: begin
                if (price == 30) begin
                    out = 1;
                    five_change = 1;
                    n_state = S0;
                end
            end

            default: n_state = S0;
        endcase
    end
endmodule
```

IV. Simulation Waveform

The figure below shows how the vending machine FSM works during simulation. It displays the changes in states based on the item selected (item) and how the machine responds through the out and five_change signals when five and ten inputs are given. The waveform clearly indicates that the FSM correctly dispenses items when valid coin amounts are entered and successfully returns to the initial state after each transaction.

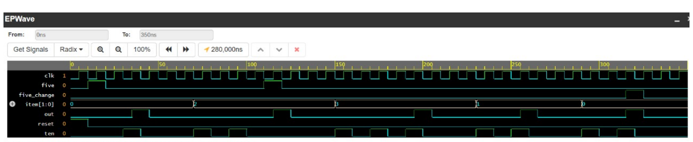

Fig. 4. Resultant Waveform

V. Qflow Results

The Vending Machine circuit was successfully synthesized using the Qflow toolchain. The process involved key steps such as logic synthesis, placement of components, routing of connections, and layout creation. The generated diagrams show how the internal gates are arranged and connected. These outputs confirm that all stages of the digital design flow were completed correctly and the FSM design was implemented as intended.

VI. Future Improvements

While the current vending machine design meets the basic functional requirements, several enhancements can be made to improve its usability, flexibility, and real-world applicability:

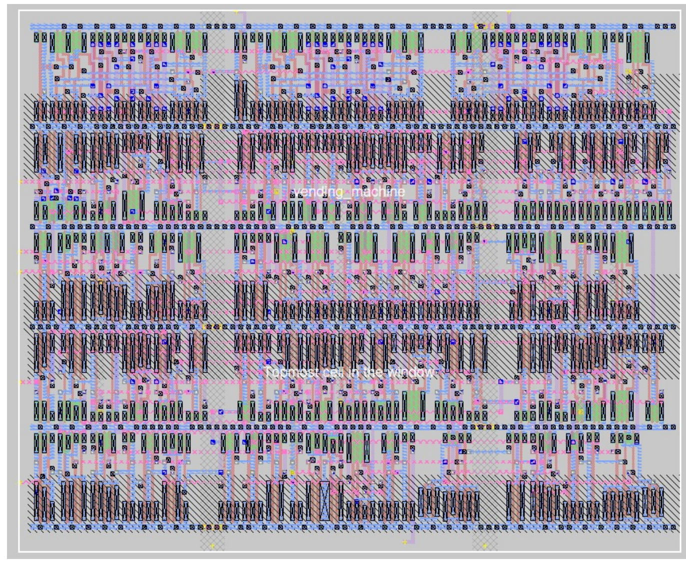

Fig. 5. Synthesized Netlist View

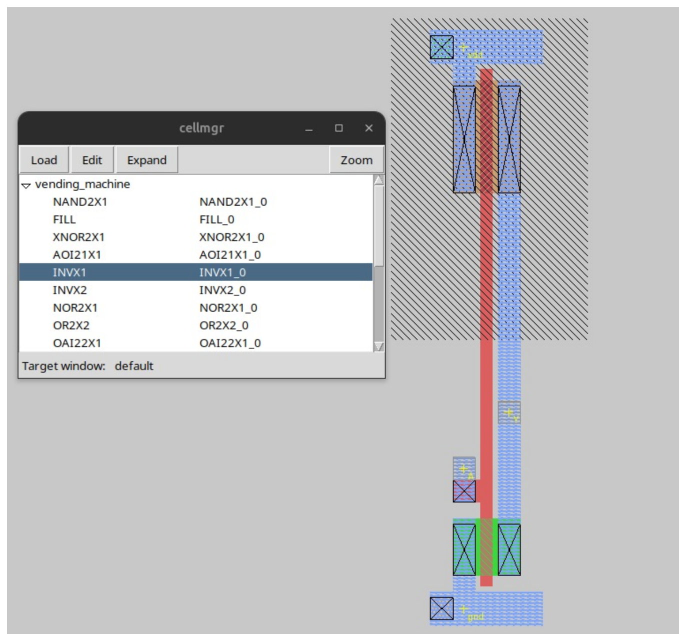

Fig. 6. View of Inverter Gate

1) **Support for Additional Coin Denominations**
   - The present system accepts only 5 and 10-unit coins. It can be extended to support more denominations such as 1, 2, or even currency notes using sensors or digital input scaling.
2) **Dynamic Item Pricing**
   - Currently, item prices are hardcoded. A future version could include programmable or real-time adjustable pricing, allowing vendors to change item costs based on demand or inventory.
3) **Multiple Item Selection in One Transaction**
   - Enhancing the system to support multiple item

purchases in a single session would make it more user-friendly and closer to real-world machines.

4) **LCD Display and User Interface Integration**
   - A small screen can be added to display balance, item names, prices, and messages like “Insert more coins” or “Item dispensed,” improving the interaction between the user and the machine.
5) **Item Availability and Stock Monitoring**
   - Sensors can be used to check item availability. If an item is out of stock, the machine could prevent selection and notify the user.
6) **Password-Protected Maintenance Mode**
   - A secured mode can be introduced for service staff to restock items, check earnings, or reset the system without interfering with regular operation.
7) **Real-Time Logging and Monitoring**
   - Logs for all transactions and errors can be stored locally or sent to a server, which can be useful for maintenance, analytics, and theft prevention.

VII. References

- R. Sati, V. Mishra and G. Verma, ”Simulation of Vending Machine Design using Verilog HDL,” 2022 2nd Asian Conference on Innovation in Technology (ASIANCON), Ravet, India, 2022, pp. 1-5, doi: 10.1109/ASIANCON55314.2022.9909003. keywords: Technological innovation;Costs;Urban areas;Toy manufacturing industry;Electronics industry;Consumer products;Behavioral sciences;FSM;Verilog HDL;FPGA;RTL Netlist
- Design and implementation of vending machine using Verilog HDL MA Qureshi, A Aziz, HF Rasool, M Ibrahim… - 2011 2nd International …, 2011
- Design of a Vending Machine Using Verilog HDL and Implementation in Genus & Encounter by Mahamudul Hassan Fuad, Rahbaar Yeassin, K. M. Mehedi Hassan, Md Mahamudunnobi Sykot, Md Faysal Nayan