Dayananda Sagar College of Engineering

Department of Electronics and Communication Engineering

Shavige Malleshwara Hills, Kumaraswamy Layout, Bangalore – 560 111.

(An Autonomous Institute affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment and Accreditation Council (NAAC) with 'A' grade

AAT

Program: B.E. Branch: ECE

Course: Advanced Digital Design Semester/Section :5th Sem

Course Code: 22EC552 Date: 23-12-2024

A Report on

" Vending Machine Controller Using FSMs"

Submitted by

Nagalakshmi S	**1DS20EC138**
Nagaraj Hunashal	**1DS20EC139**
Swetha	**1DS20EC181**
Ullas R J	**1DS20EC240**

Faculty In-charge

Dr. Madhura R

Contents

Abstract:

The vending machine is a ubiquitous tool that provides quick and easy access to a variety of products, simplifying everyday transactions. Its seamless operation relies on a well-designed control system that governs its functionality. This project explores the design and implementation of a Vending Machine Controller (VMC) using Finite State Machines (FSMs) to ensure precise, efficient, and error-free operation.

Finite State Machines (FSMs) are computational models widely used in embedded systems for designing systems with predictable behaviors. In this project, FSMs are employed to model the operation of the vending machine, where each state represents a specific step in the interaction process, such as accepting user input, verifying payment, dispensing the product, or handling errors.

The VMC begins in an idle state, waiting for a user to interact with the machine. It transitions through states based on inputs such as coin insertion, product selection, and availability checks. The FSM design ensures that all operations are handled systematically, allowing the controller to process inputs, update states, and produce corresponding outputs efficiently.

Key features of the system include:

Modularity: Each state and transition is explicitly defined, making it easier to modify or extend the system.
Error Handling: The FSM is designed to account for invalid inputs, insufficient payment, or product unavailability, ensuring reliability.
User Interaction: The system is intuitive, providing feedback for user actions through simple state-driven outputs like LEDs, sounds, or messages.
This approach not only simplifies the design but also enhances the reliability and scalability of the vending machine. The project demonstrates how FSMs can be leveraged to model real-world systems effectively, paving the way for advanced applications in automation and control.

By utilizing FSMs, the Vending Machine Controller achieves streamlined operation, ensuring both user satisfaction and operational efficiency, making it an ideal case study for modern embedded systems design.

Introduction:

Vending machines are widely used automated systems that provide goods like snacks, beverages, or other products upon receiving payment. The operation of a vending machine requires a controller to manage processes such as accepting coins, validating payments, selecting items, and dispensing goods. These processes are sequential and event-driven, making Finite State Machines (FSMs) an ideal design approach for their implementation.

Finite State Machines and their relevance:

Finite State Machines are mathematical models used to design systems that can exist in a finite number of states and transition between them based on specific inputs. FSMs are particularly suitable for vending machines due to their ability to:

- Model the sequential nature of vending machine operations.
- Simplify the design process by clearly defining states and transitions.
- Enhance modularity, making the system easier to debug, maintain, and extend.
- For a vending machine, each state corresponds to a specific part of the workflow, such as waiting for coins, allowing item selection, or dispensing products. Transitions between these states occur based on user interactions, such as inserting coins or selecting items, as well as internal conditions, like verifying sufficient payment or inventory.

Literature Survey:

- Design and Implementation of FSM for Automated Vending Machine by B. Kumar et al. (2015)
- Finite State Machine Design for Embedded Systems by J. Huang et al. (2017)
- Improved User Interface Design for Vending Machines Using FSM by K. R. Patel et al. (2018)
- Fault Tolerant Design of a Vending Machine Using FSMs by M. S. Sharma et al. (2020)
- Low Power Design of Embedded Controllers for Vending Machines Using FSM by J. Liu and L. Zhang (2021)
- Comparative Study of FSM and Conventional Control Methods in Vending Machines by T. Choi et al. (2019)
- Implementation of Vending Machine Controller in Verilog Using FSMs by M. R. Patel et al. (2022)

Problem Definition:

Automated vending machines are commonly used to dispense goods like snacks, beverages, and other products upon receiving appropriate payment. The operation of these machines requires a reliable and efficient controller to manage tasks such as validating payments, allowing item selection, dispensing products, and returning change. Traditional implementations often face challenges in terms of complexity, scalability, and adaptability to new features or scenarios.

The problem lies in designing a vending machine controller that can handle these tasks systematically while maintaining ease of operation and high reliability. The controller must:

Accept and validate various denominations of coins.
Allow the user to select an item from multiple available options with different prices.
Ensure accurate calculation of payment and return change if the inserted amount exceeds the item cost.
Handle errors such as invalid coins, insufficient payment, or item unavailability.
Respond to reset or cancel operations to restore the machine to its initial state.
The primary challenge is to design a system that models this multi-step process efficiently using a digital logic framework. Finite State Machines (FSMs) are an effective solution to address this challenge due to their capability to represent sequential logic in a structured manner.

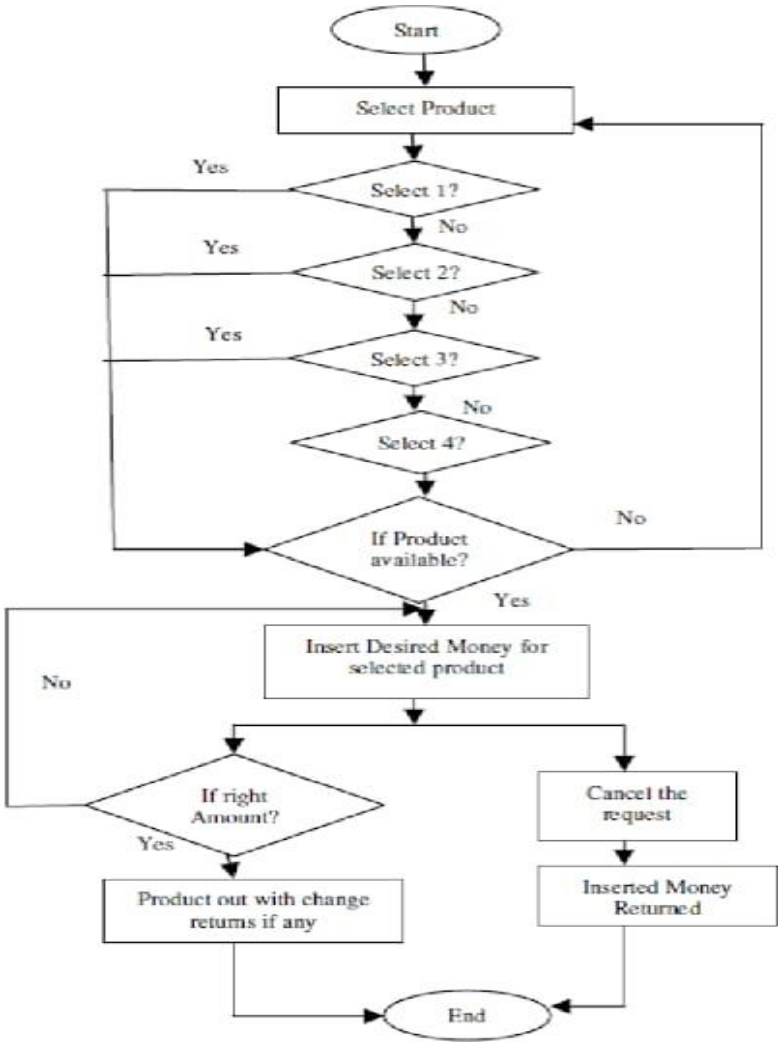
The task is to implement an FSM-based vending machine controller using Verilog HDL, which will:

Define the states for different operational phases of the vending machine.
Handle transitions between states based on user actions and system conditions.
Generate outputs for dispensing items, returning change, or indicating errors.
The solution must be scalable to accommodate additional features, such as multiple payment methods or dynamic pricing, while ensuring optimal performance and reliability. This controller will serve as the core logic for a vending machine, enabling seamless user interaction and accurate transaction management.

Objectives:

1. Develop an FSM-Based Design: Create a Finite State Machine (FSM) to manage the sequential operations of a vending machine, including payment validation, item selection, and dispensing.
2. Implement Payment Validation: Ensure the controller accurately accepts and validates coins of specific denominations and calculates the total amount.
3. Facilitate User Interaction: Enable smooth user operations, including inserting coins, selecting items, canceling transactions, and receiving real-time feedback.
4. Handle Error and Change Management: Detect and respond to invalid inputs or errors (e.g., insufficient payment, out-of-stock items) and accurately calculate and return change.
5. Use Verilog for Implementation: Model and simulate the vending machine controller using Verilog HDL for efficient hardware synthesis and real-world deployment.

Main Block Diagram:



Here’s a detailed explanation of each block in the flowchart:

1. **Start**:

   The process begins here. This block indicates the initialization of the vending machine’s workflow.

2. **Select Product**:

   The user is prompted to choose a product. This initiates the selection process, allowing the user to input their desired choice.

3. **Select 1, 2, 3, 4**:

   These decision blocks represent the available product options. The system checks if the user has selected a specific product (e.g., Product 1, Product 2, Product 3, or Product 4).

   - If "Yes" for any selection, the flow proceeds to check product availability.
   - If "No," it proceeds to the next option. If no valid product is selected, the flow returns to the "Select Product" block.

4. **If Product Available?**:

   This decision block verifies if the selected product is in stock.

   - If "Yes," the process continues to the payment stage.
   - If "No," the user is prompted to select another product, looping back to "Select Product."

5. **Insert Desired Money for Selected Product**:

   The user is prompted to insert the required amount of money for the selected product. This is a crucial step before proceeding to the validation of the payment.

6. **If Right Amount?**:

   This decision block checks whether the correct amount has been inserted:

   - If "Yes," the product is dispensed, and any necessary change is returned.
   - If "No," the user can either cancel the request or add more money.

7. **Cancel the Request**:

   If the user decides to cancel the transaction:

   - The inserted money is returned.
   - The process ends without dispensing the product.

8. **Product Out with Change Returns if Any**:

   If the correct amount is inserted, the selected product is dispensed, and any required change is returned to the user.

9. **End**:

   The process concludes here. All operations related to the transaction are finalized.

This flowchart represents a vending machine's logic, focusing on product selection, payment validation, and product dispensing. It ensures smooth and logical transactions with options to retry or cancel.

Verilog code

```
timescale 1ns / 1ps
module VendingMachine_tb;
   // Inputs to the VendingMachine module
   reg [3:0] item;
   reg five_in, ten_in, reset, clock;
   // Outputs from the VendingMachine module
   wire dispense, five_out;

   // Instantiate the VendingMachine module
   VendingMachine DUT (
      .item(item),
      .five_in(five_in),
      .ten_in(ten_in),
      .clock(clock),
      .reset(reset),
      .dispense(dispense),
      .five_out(five_out)
   );

   // Clock generation: 50MHz clock (20ns period)
   initial begin
      clock = 0;
      forever #10 clock = ~clock; // Toggle clock every 10ns
   end

   // Test sequence
   initial begin
      // Initialize inputs
```

```
item = 4'b0000;
five_in = 0;
ten_in = 0;
reset = 0;

// Apply reset
$display("Applying reset...");
reset = 1;
#20; // Wait for a few clock cycles
reset = 0;

// Test Item 1 (Price: 15)
$display("Testing Item 1 (Price: 15)");
item = 4'b0001;
five_in = 1; #20; five_in = 0; // Insert 5
ten_in = 1; #20; ten_in = 0; // Insert 10
#20; // Wait for dispense
$display("Dispense: %b, Five_out: %b", dispense, five_out);

// Test Item 2 (Price: 25)
$display("Testing Item 2 (Price: 25)");
item = 4'b0010;
five_in = 1; #20; five_in = 0; // Insert 5
five_in = 1; #20; five_in = 0; // Insert 5
ten_in = 1; #20; ten_in = 0; // Insert 10
#20; // Wait for dispense
$display("Dispense: %b, Five_out: %b", dispense, five_out);

// Test Item 3 (Price: 35)
$display("Testing Item 3 (Price: 35)");
item = 4'b0100;
five_in = 1; #20; five_in = 0; // Insert 5
```

```
        five_in = 1; #20; five_in = 0; // Insert 5
        ten_in = 1; #20; ten_in = 0; // Insert 10
        ten_in = 1; #20; ten_in = 0; // Insert 10
        #20; // Wait for dispense
        $display("Dispense: %b, Five_out: %b", dispense, five_out);

        // Test Item 4 (Price: 45)
        $display("Testing Item 4 (Price: 45)");
        item = 4'b1000;
        ten_in = 1; #20; ten_in = 0; // Insert 10
        ten_in = 1; #20; ten_in = 0; // Insert 10
        ten_in = 1; #20; ten_in = 0; // Insert 10
        five_in = 1; #20; five_in = 0; // Insert 5
        #20; // Wait for dispense
        $display("Dispense: %b, Five_out: %b", dispense, five_out);

        // Test Reset Functionality
        $display("Testing Reset Functionality");
        reset = 1; #20; reset = 0;
        $display("Reset Complete: Dispense: %b, Five_out: %b", dispense, five_out);

        // End simulation
        $stop;
    end
endmodule
```

Testbench

```
`timescale 1ns / 1ps
module VendingMachine_tb;
    // Inputs to the VendingMachine module
    reg [3:0] item;
    reg five_in, ten_in, reset, clock;
    // Outputs from the VendingMachine module
```

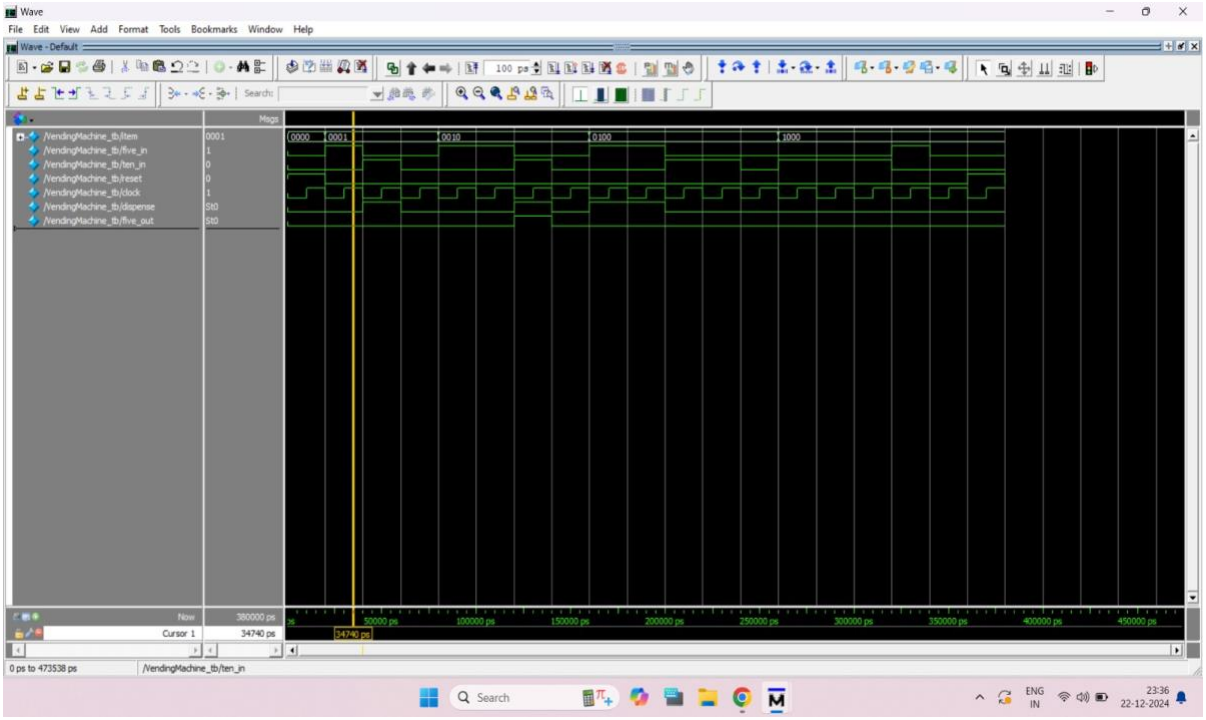
```
wire dispense, five_out;

// Instantiate the VendingMachine module
VendingMachine DUT (
    .item(item),
    .five_in(five_in),
    .ten_in(ten_in),
    .clock(clock),
    .reset(reset),
    .dispense(dispense),
    .five_out(five_out)
);

// Clock generation: 50MHz clock (20ns period)
initial begin
    clock = 0;
    forever #10 clock = ~clock; // Toggle clock every 10ns
end

// Test sequence
initial begin
    // Initialize inputs
    item = 4'b0000;
    five_in = 0;
    ten_in = 0;
    reset = 0;

    // Apply reset
    $display("Applying reset...");
    reset = 1;
    #20; // Wait for a few clock cycles
    reset = 0;
```

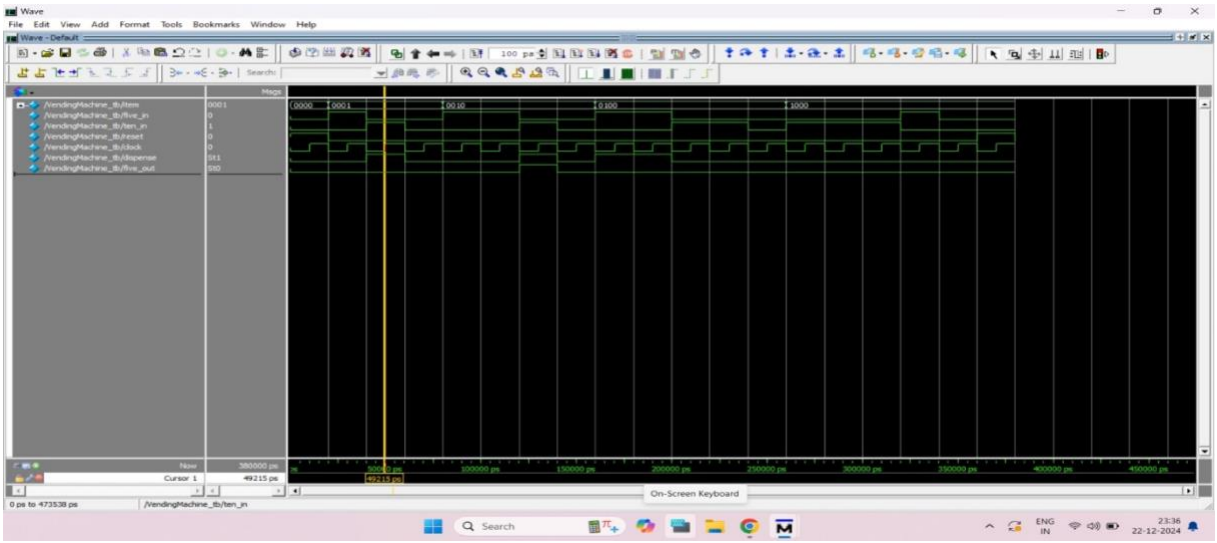
```
// Test Item 1 (Price: 15)
$display("Testing Item 1 (Price: 15)");
item = 4'b0001;
five_in = 1; #20; five_in = 0; // Insert 5
ten_in = 1; #20; ten_in = 0; // Insert 10
#20; // Wait for dispense
$display("Dispense: %b, Five_out: %b", dispense, five_out);

// Test Item 2 (Price: 25)
$display("Testing Item 2 (Price: 25)");
item = 4'b0010;
five_in = 1; #20; five_in = 0; // Insert 5
five_in = 1; #20; five_in = 0; // Insert 5
ten_in = 1; #20; ten_in = 0; // Insert 10
#20; // Wait for dispense
$display("Dispense: %b, Five_out: %b", dispense, five_out);

// Test Item 3 (Price: 35)
$display("Testing Item 3 (Price: 35)");
item = 4'b0100;
five_in = 1; #20; five_in = 0; // Insert 5
five_in = 1; #20; five_in = 0; // Insert 5
ten_in = 1; #20; ten_in = 0; // Insert 10
ten_in = 1; #20; ten_in = 0; // Insert 10
#20; // Wait for dispense
$display("Dispense: %b, Five_out: %b", dispense, five_out);

// Test Item 4 (Price: 45)
$display("Testing Item 4 (Price: 45)");
item = 4'b1000;
ten_in = 1; #20; ten_in = 0; // Insert 10
```

```
        ten_in = 1; #20; ten_in = 0; // Insert 10
        ten_in = 1; #20; ten_in = 0; // Insert 10
        five_in = 1; #20; five_in = 0; // Insert 5
        #20; // Wait for dispense
        $display("Dispense: %b, Five_out: %b", dispense, five_out);

        // Test Reset Functionality
        $display("Testing Reset Functionality");
        reset = 1; #20; reset = 0;
        $display("Reset Complete: Dispense: %b, Five_out: %b", dispense, five_out);

        // End simulation
        $stop;
    end
endmodule
```

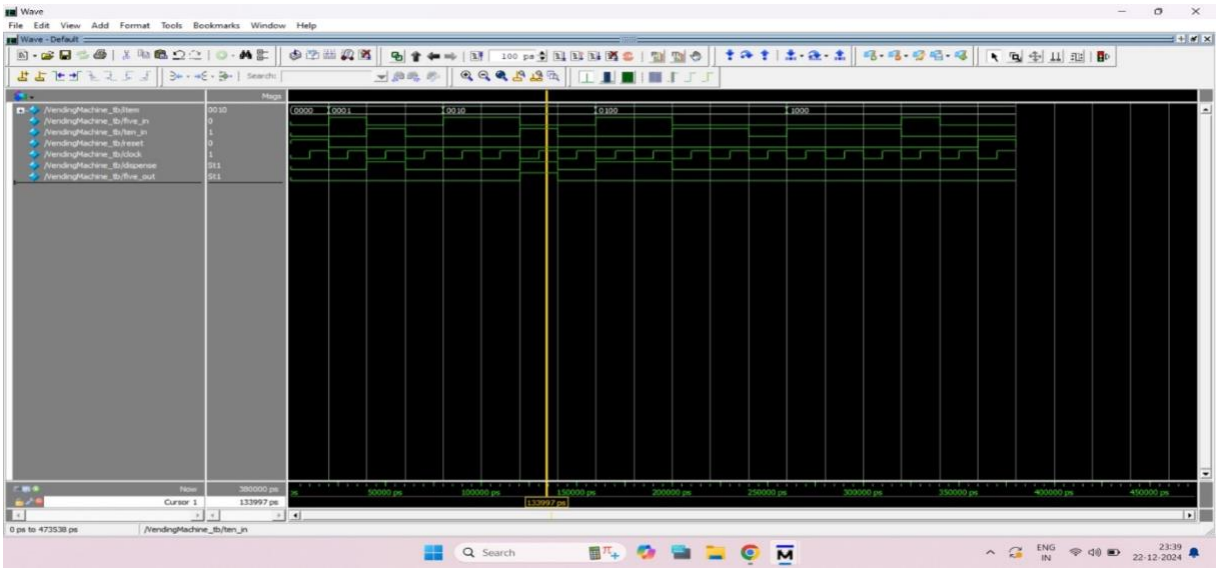
Results and Discussions

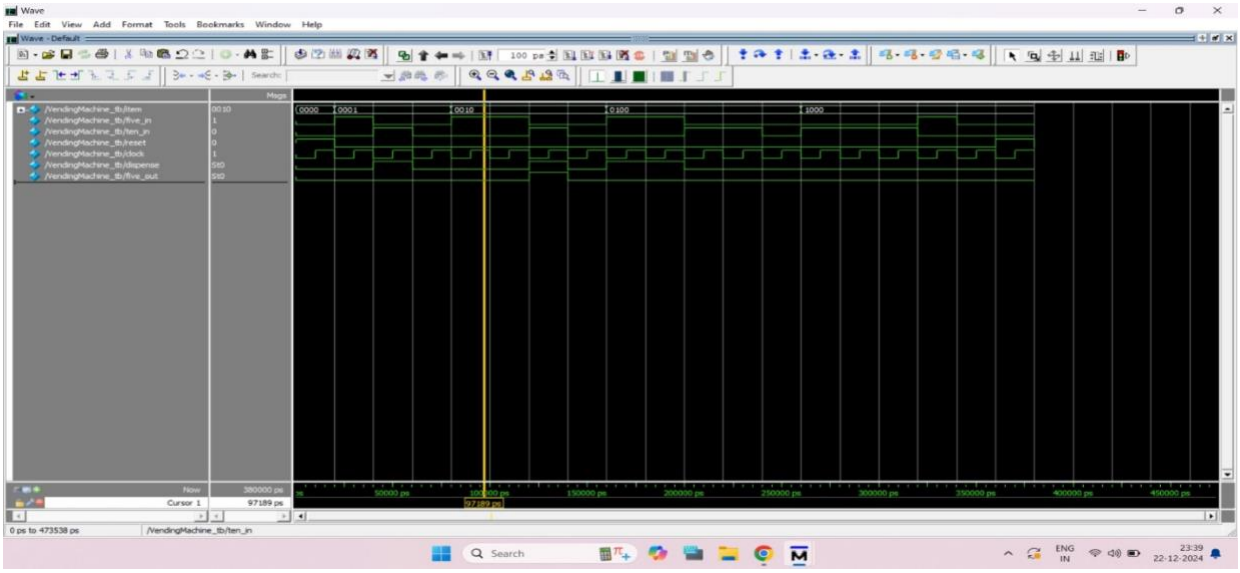
First item dispense:



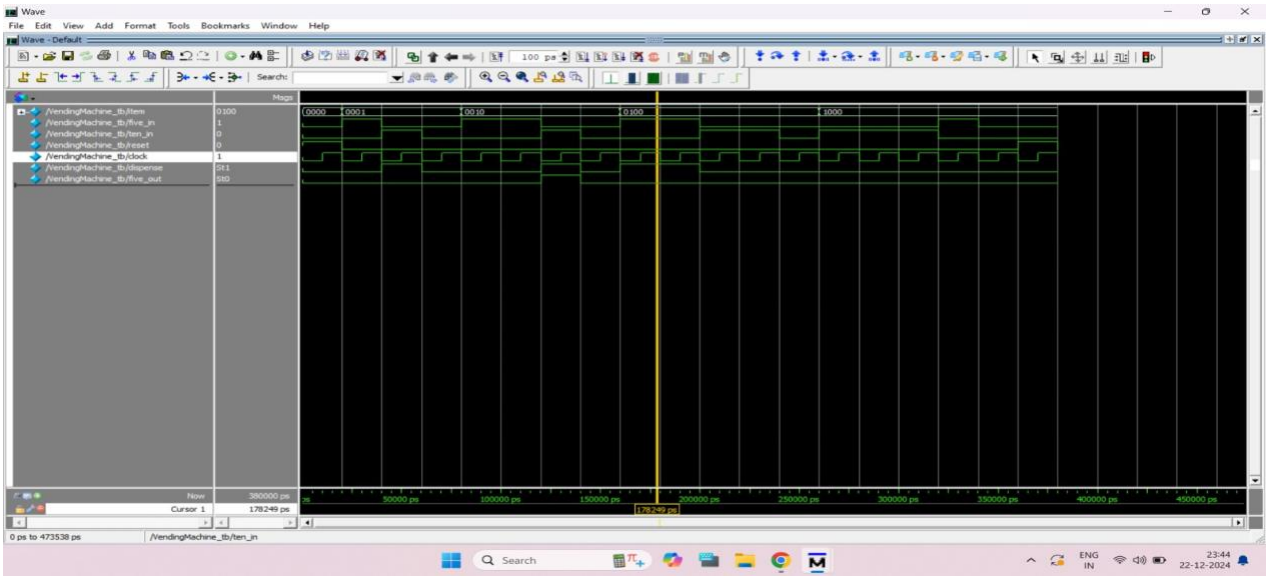


Second item dispenses:





Third item dispense:



The Verilog waveform output for the vending machine using Finite State Machines (FSMs) confirms the correct functionality of state transitions. It demonstrates seamless product selection, payment validation, and product dispensing. The simulation validates the FSM design, showcasing proper synchronization, error handling, and transitions between idle, selection, payment, and dispense states.

Conclusion

The Verilog simulation of the vending machine using FSMs concludes that the design operates efficiently and accurately. The waveform output confirms correct state transitions, ensuring reliable product selection, payment validation, and dispensing. This demonstrates the robustness of FSM-based designs in implementing sequential systems with clear and predictable behavior.