# PROJECT REPORT

## VENDING MACHINE WITH CHANGE SYSTEM

Prepared by: Aiman Tariq
Roll number: 122EE0171
Department of Electrical Engineering
National Institute of Technology , Rourkela

01

# INTRODUCTION

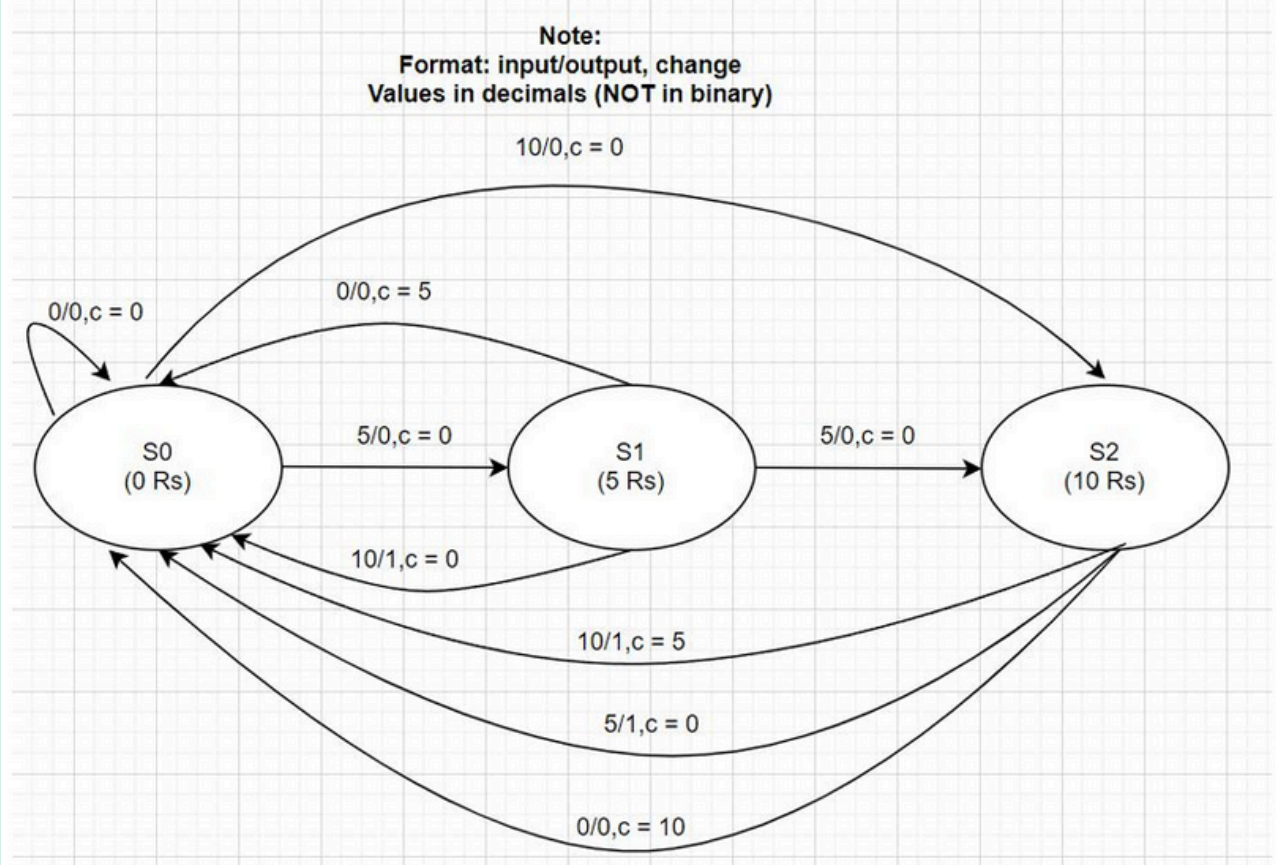
Vending machines are automated devices that dispense items such as snacks, beverages, tickets, and other consumer goods in exchange for payment. They are widely used in public spaces for their convenience, requiring no human operator and functioning around the clock. The idea of vending machines dates back to ancient times, with one of the earliest known models designed by Hero of Alexandria to dispense holy water. Modern vending machines have evolved significantly, incorporating electronics, sensors, and programmable logic to handle various currencies, manage inventory, and provide user feedback.

This project focuses on designing a simple digital vending machine system that demonstrates the working principles of such machines using hardware description language. The system is capable of accepting two denominations—₽5 and ₽10—to purchase a product costing ₽15. The machine is also programmed to handle situations like overpayment (returning appropriate change) and incomplete transactions (refunding inserted money). The logic is implemented using Verilog HDL and modeled as a finite state machine (FSM), then simulated and verified using EDA Playground to ensure correct operation under different input scenarios.

02

# STATE DIAGRAM :

*The product water bottle costs 15 Rs*



03

# STATESTRANSITIONS :

**The product costs 15 Rs.**

**S0 : 0 Rs in Vending Machine**

☒ If nothing added: Stay on State 0, OUTPUT = 0, CHANGE = 0.
☒ If5Rsadded:MovetoState 1, OUTPUT = 0,CHANGE =0.
☒ If10Rsadded: Moveto State 2, OUTPUT =0, CHANGE= 0.

**S1 : 5 Rs inVending Machine**

☒ If nothing added: Move to State 0, OUTPUT = 0, CHANGE=Rs5(01).
☒ If5Rsadded:MovetoState 1, OUTPUT = 0,CHANGE =0.
☒ If10Rsadded: Move to State 0, OUTPUT = 1, CHANGE= Rs0.

**S2 : 10 Rs in Vending Machine**

☒ If nothing added: Move to State 0, OUTPUT = 0, CHANGE=Rs10(10).
☒ If5Rsadded: Move to State 0, OUTPUT = 1, CHANGE=Rs0.
☒ If10Rsadded: Here the customer over payed, thus a bottle should be returned but with CHANGE(5 Rs). Move to State 0, OUTPUT = 1, CHANGE= Rs5(01)

04

# ABOUT LANGUAGE USED:

Verilog is a Hardware Description Language (HDL) standardized as IEEE 1364, used for modeling and designing electronic systems, particularly digital circuits. It enables designers to describe hardware at various levels of abstraction, especially at the register-transfer level (RTL). Verilog supports the design and verification of both digital and mixed-signal systems.

Originally introduced in 1984, Verilog greatly improved productivity by allowing concise and structured representations of circuit behavior, replacing earlier schematic-based designs. In 2009, Verilog was merged into the SystemVerilog standard under IEEE 1800-2009, with the most current version being IEEE 1800-2017.

The language resembles the C programming language in syntax and structure, making it accessible to engineers familiar with software development. It includes common programming constructs like if-else, for, while, and case, along with both blocking (=) and non-blocking (<=) assignments to control data flow and timing behavior in simulations.

Verilog modules define system hierarchy through ports (input, output, and bidirectional), and can contain internal logic using wire, reg, and procedural blocks. The execution of these blocks follows a dataflow model, where multiple blocks operate concurrently, similar to how real hardware behaves.

A significant portion of Verilog is synthesizable, meaning the code can be transformed into a physical implementation using synthesis tools. These tools convert the Verilog code into a netlist, which represents logic gates and flip-flops that can be used to generate layout files for FPGA or ASIC manufacturing.

Verilog is widely used in academia and industry for building digital systems, control logic, processors, and embedded applications due to its versatility, modularity, and efficiency.

05

## VERILOG DESIGN CODE:

```
module vending_machine_18105070(
    input clk,
    input rst,
    input [1:0] in, // 01 = 5 rs, 10 = 10 rs
    output reg out,
    output reg [1:0] change
);

parameter s0 = 2'b00;
parameter s1 = 2'b01;
parameter s2 = 2'b10;

reg [1:0] c_state, n_state;

always @ (posedge clk)
begin
    if (rst == 1) begin
        c_state = s0;
        n_state = s0;
        change = 2'b00;
        out = 0;
    end else
        c_state = n_state;

    case (c_state)
        s0: begin
            if (in == 2'b00) begin
                n_state = s0;
                out = 0;
                change = 2'b00;
            end else if (in == 2'b01) begin
                n_state = s1;
                out = 0;
                change = 2'b00;
            end else if (in == 2'b10) begin
                n_state = s2;
                out = 0;
                change = 2'b00;
            end
        end
```

# 06

```
        s1: begin
            if (in == 2'b00) begin
                n_state = s0;
                out = 0;
                change = 2'b01;
            end else if (in == 2'b01) begin
                n_state = s2;
                out = 0;
                change = 2'b00;
            end else if (in == 2'b10) begin
                n_state = s0;
                out = 1;
                change = 2'b00;
            end
        end
        s2: begin
            if (in == 2'b00) begin
                n_state = s0;
                out = 0;
                change = 2'b10;
            end else if (in == 2'b01) begin
                n_state = s0;
                out = 1;
                change = 2'b00;
            end else if (in == 2'b10) begin
                n_state = s0;
                out = 1;
                change = 2'b01;
            end
        end
    endcase
end
endmodule
```

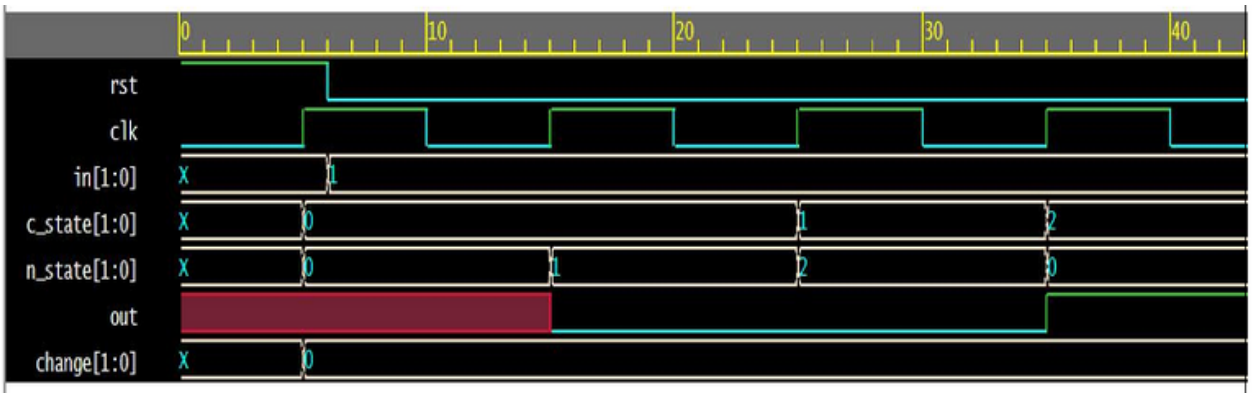
# 07

## TEST BENCH CODE:

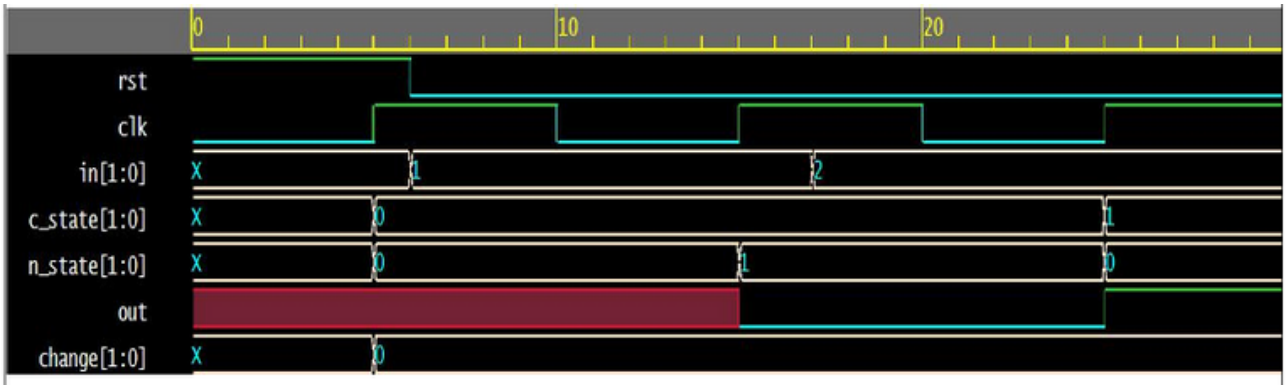
```
module vending_machine_tb;

reg clk;
reg [1:0] in;
reg rst;
wire out;
wire [1:0] change;

vending_machine_18105070 uut (
    .clk(clk),
    .rst(rst),
    .in(in),
    .out(out),        B
    .change(change)
);

initial begin
    $dumpfile("vending_machine.vcd");
  $dumpvars(1, vending_machine_tb);

    rst = 1; clk = 0;
    #6 rst = 0; in = 2;  // Insert 10 Rs
    #19 in = 2;          // Insert another 10 Rs (should get bottle + 5 Rs)
    #25 $finish;
end

always #5 clk = ~clk;  // Clock generator

endmodule
```

# 08 RESULTS:

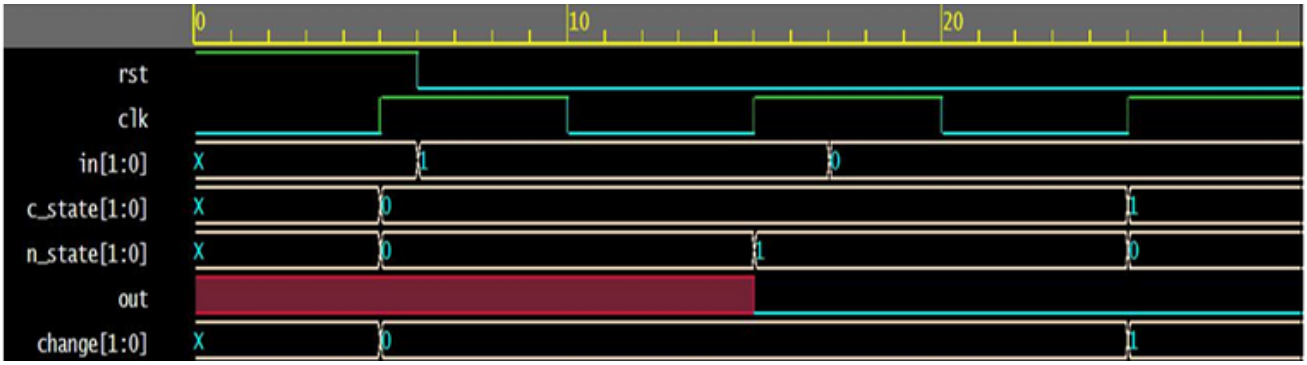
*1. Adding 5 Rs three times consecutively*



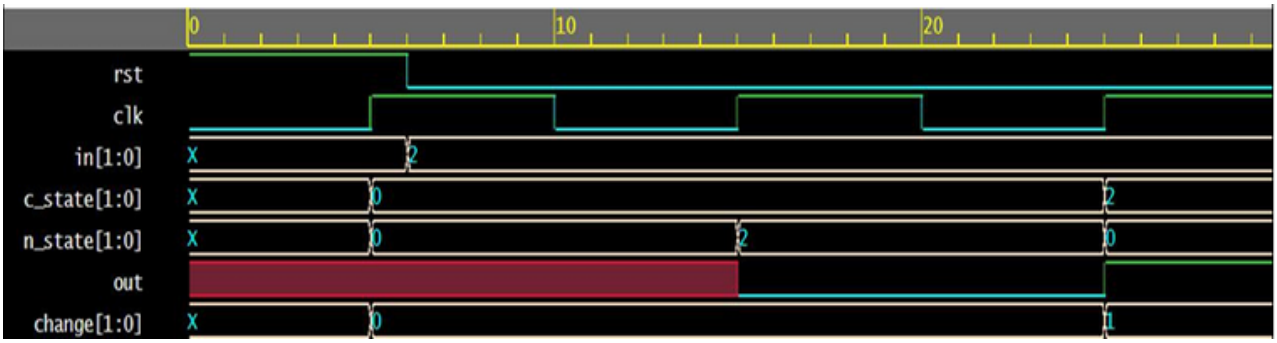

*2. Adding 5 Rs and then 10 Rs*



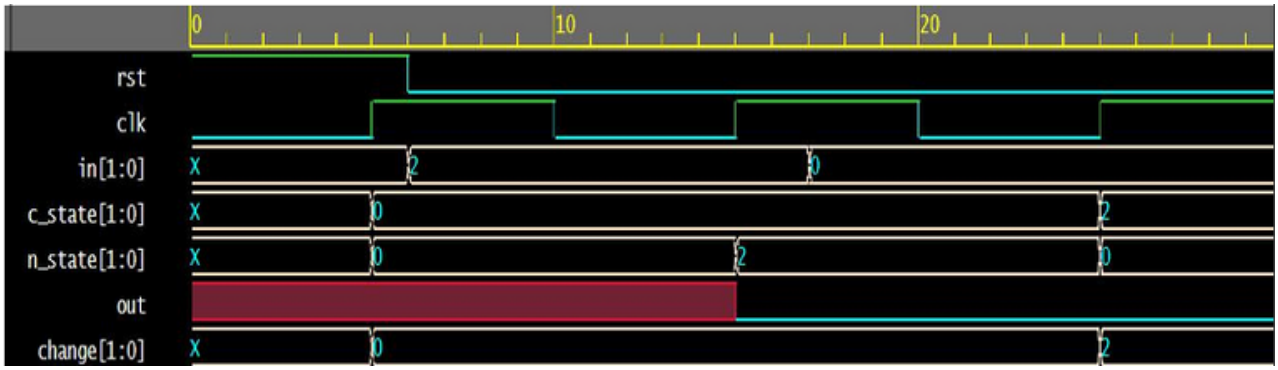

*3. Adding 10 Rs two times*



*4. Adding 5 Rs and then nothing*




*5. Adding 10 Rs and then nothing*



10

# CONCLUSION

This project successfully demonstrated the design and simulation of a digital vending machine using Verilog Hardware Description Language (HDL). By modeling the system as a finite state machine (FSM), we were able to replicate the essential behavior of a real-world vending machine, including coin handling, product dispensing, and change return.

The system was designed to accept ₱5 and ₱10 coins for a product priced at ₱15, and handled various input scenarios such as exact payment, overpayment (with change returned), and incomplete transactions (with refund). The use of Verilog allowed for clear representation of hardware logic, while simulation through EDA Playground provided visual verification of system behavior using testbenches and waveform analysis.

Through this project, key concepts in digital design such as state transition logic, synchronous clocking, and modular design were reinforced. It also provided hands-on experience with Verilog coding, testbench development, and the use of simulation tools in verifying hardware functionality.

In conclusion, the vending machine project serves as a practical example of how hardware description languages can be applied to build and test real-time embedded systems, bridging the gap between theoretical concepts and real-world digital logic implementation.