Design and Implementation of a Coin-Operated Vending Machine Using Verilog

Samuel Singaram

CONTENTS

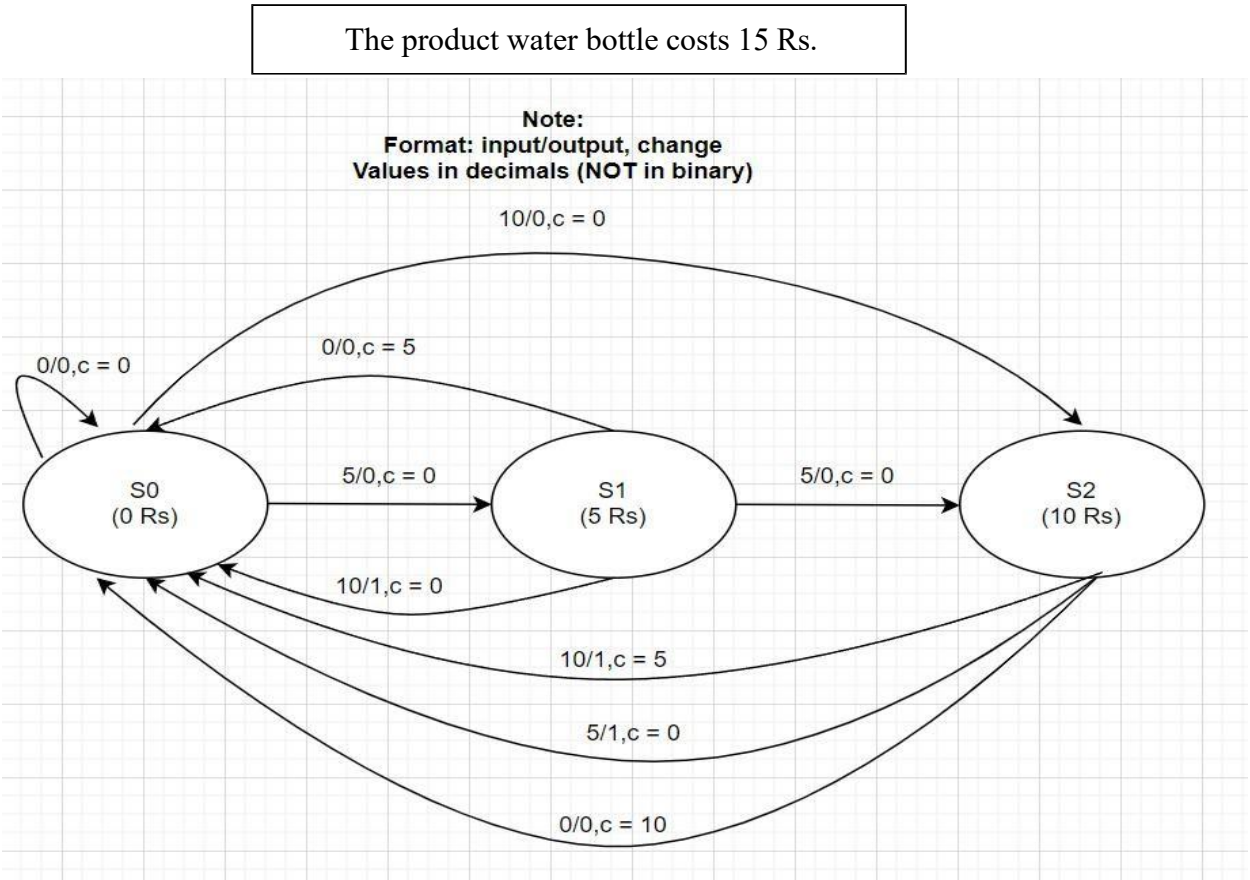
1. Introduction

The vending machine is a finite state machine (FSM) designed to dispense a product upon receiving the appropriate payment. This project simulates a vending machine with three states, accepting ₹5 and ₹10 coins, while calculating the total payment and returning change if needed. The machine is implemented using Verilog and tested using a Verilog test bench.

2. State Diagram

The vending machine operates in three states:

- **S0 (Initial State):** No coins inserted.
- **S1:** ₹5 has been inserted.
- **S2:** ₹10 has been inserted.



Each state transition depends on the input coin value (₹0, ₹5, or ₹10) and the cumulative amount.

3. State Transitions

The state transitions are as follows:

- From **S0**:
  - ₹5 → Move to **S1**.
  - ₹10 → Move to **S2**.
- From **S1**:
  - ₹5 → Move to **S2**.
  - ₹10 → Dispense product and return to **S0**.
- From **S2**:
  - ₹5 → Dispense product and return to **S0**.
  - ₹10 → Dispense product, return ₹5 as change, and move to **S0**.

The default behavior ensures the FSM resets to **S0** on invalid inputs or a reset signal.

4. About Language Used

The vending machine project is implemented using **Verilog HDL (Hardware Description Language)**, a popular language for designing and simulating digital circuits. Verilog provides a concise and flexible framework for defining hardware behavior, enabling the modeling of combinational and sequential logic with ease.

Key Features of Verilog:

Structural and Behavioral Modeling:

Verilog supports both low-level structural descriptions and high-level behavioral descriptions, making it suitable for a wide range of applications.

Event-Driven Simulation:

Verilog uses an event-driven simulation model, allowing for the precise analysis of circuit behavior over time.

Concurrency:

It allows concurrent execution of statements, reflecting real-world hardware behavior where multiple components operate simultaneously.

Parameterized Designs:

Verilog allows parameterization, making designs more reusable and scalable.

5. Verilog Design Code

```
module vending_machine(
input clk,
input reset,
input [1:0]in,  //input 2'b0=0rs, 2'b01=5rs 2'b10=10rs
output reg out, //dispenses product
output reg [1:0]change, //returns change or balance amount
output reg [1:0]c_state,n_state
 );
                parameter s0=2'b00;
                 parameter s1=2'b01;
                 parameter s2=2'b10;

                 always@(posedge clk)
                 begin
                 if(reset)
                   begin
                       c_state=0;
                       n_state=0;
                      change=2'b00;
                   end
                 else
```

```
 c_state=n_state;


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

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

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

6. Test Bench Code

```
module vending_machine_tb;

        // Inputs
        reg clk;
        reg reset;
        reg [1:0] in;

        // Outputs
```

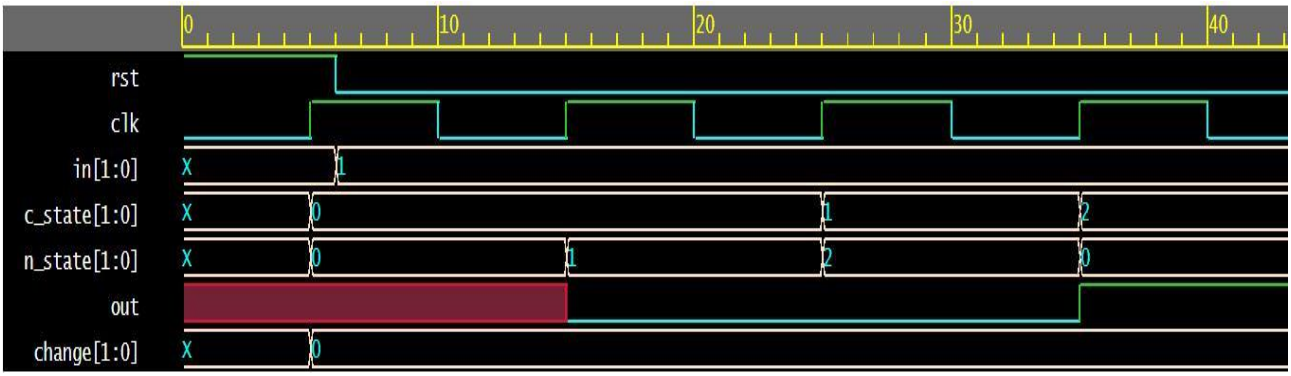
```
wire out;
wire [1:0] change;
wire [1:0] c_state;
wire [1:0] n_state;



// Instantiate the Unit Under Test (UUT)
vending_machine uut (
        .clk(clk),
        .reset(reset),
        .in(in),
        .out(out),
        .change(change),
        .c_state(c_state),
        .n_state(n_state)
);

initial begin
        clk = 0;
        reset = 1;
        // Test Case 1: Insert ₹5, ₹5, ₹5
        #6  reset=0;
           in = 2'b01;  // ₹5
        #11 in=2'b01;   // ₹5
        #16 in=2'b01;   // ₹5
```

```
/* Test Case 2: Insert ₹5, ₹10
                #6  reset=0;
                 in = 2'b01;  // ₹5
               #11 in=2'b10;   // ₹10  */


/* Test Case 3: Insert ₹10, ₹10
                #6  reset=0;
                 in = 2'b10;  // ₹10
               #11 in=2'b10;   // ₹10  */

/* Test Case 4: Insert ₹5, then nothing
                #6  reset=0;
                 in = 2'b01;  // ₹5 */

/* Test Case 5: Insert ₹10, then nothing
                #6  reset=0;
                 in = 2'b10;  // ₹10 */

   #11 $finish;
       end
            initial begin
      $dumpfile("vending_machine.vcd");
      $dumpvars(1,vending_machine_tb);
```

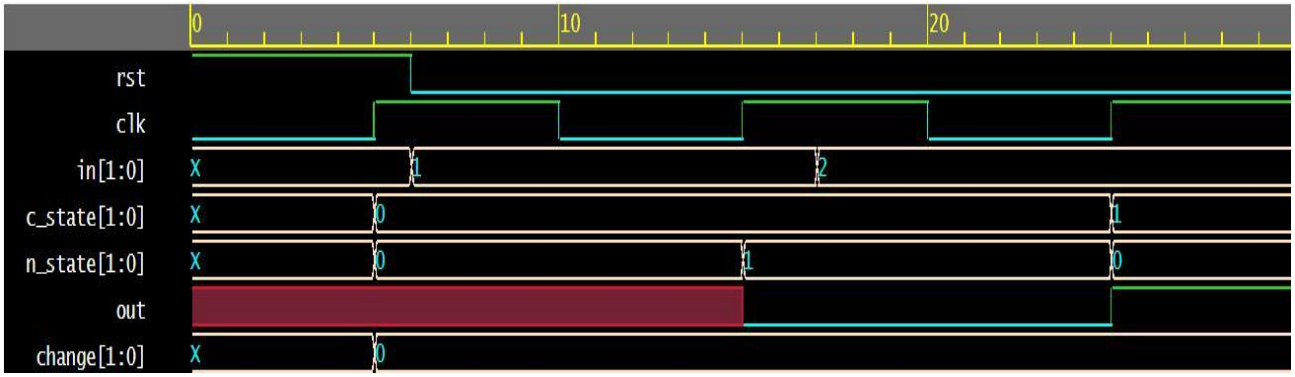
```
        forever #5 clk=~clk;
        end
Endmodule
```

7. Results

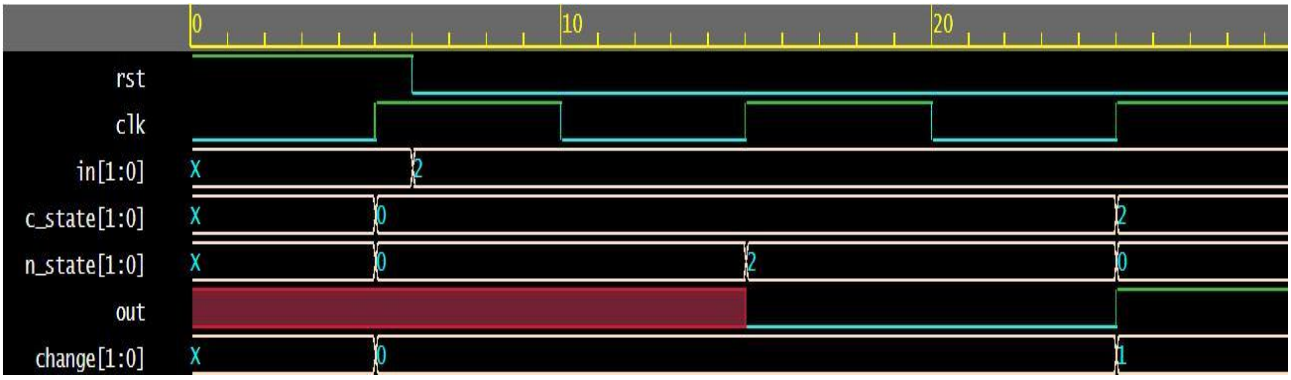
1.Adding 5 Rs three times consecutively



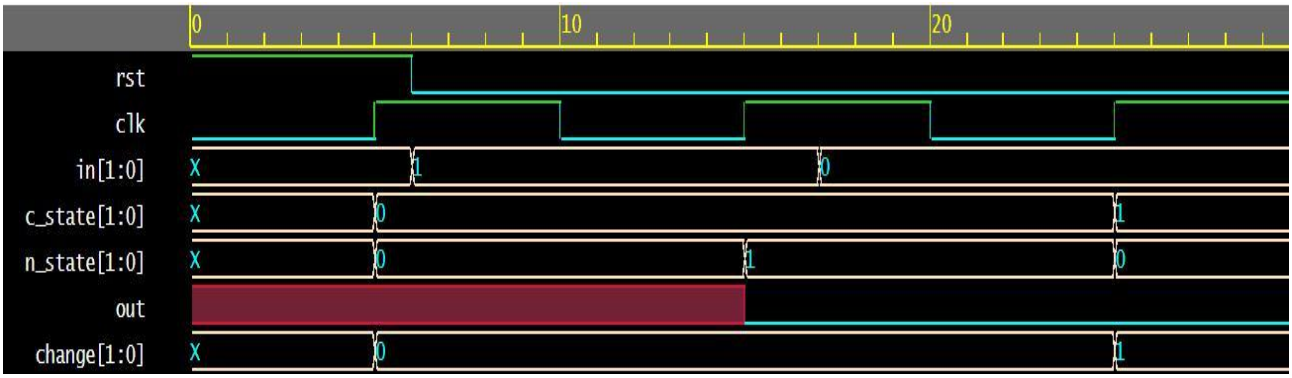
2.Adding 5 Rs and then 10 Rs



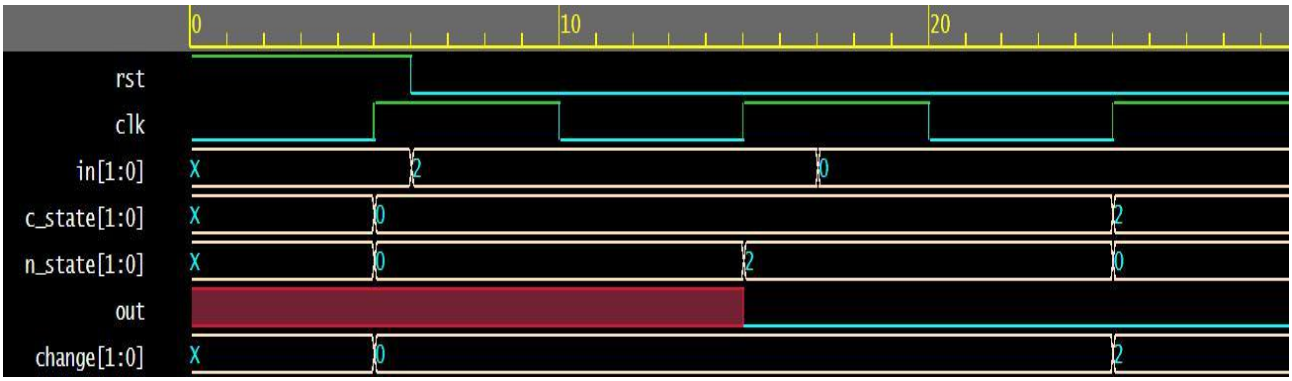
3.Adding 10 Rs two times



4.Adding 5 Rs and then nothing



5.Adding 10 Rs and then nothing



8. References

- Samir Palnitkar, *"Verilog HDL: A Guide to Digital Design and Synthesis."*
- IEEE Standard for Verilog Hardware Description Language.
- Online resources on FSM design and Verilog simulation(youtube etc..).