CONTENTS

1. Introduction.....03

2. State Diagram..... 05

3. States transitions..... 06

4. About Language used..... 08

5. Verilog Design Code..... 10

6. Test Bench Code..... 13

7. Results.....14

8. References.....16

INTRODUCTION:

A vending machine is an automated machine that provides items such as snacks, beverages, cigarettes and lottery tickets to consumers after cash, a credit card, or other form of payment is inserted into the machine or otherwise made.

The first modern vending machines were developed in England in the early 1880s and dispensed postcards. Vending machines exist in many countries and, in more recent times, specialized vending machines that provide less common products compared to traditional vending machine items have been created.

The earliest known reference to a vending machine is in the work of Hero of Alexandria, an engineer and mathematician in first-century Roman Egypt. His machine accepted a coin and then dispensed holy water. When the coin was deposited, it fell upon a pan attached to a lever. The lever opened a valve which let some water flow out. The pan continued to tilt with the weight of the coin until it fell off, at which point a counterweight snapped the lever up and turned off the valve.

The earliest known reference to a vending machine is in the work of Hero of Alexandria, an engineer and mathematician in first-century Roman Egypt. His machine accepted a coin and then dispensed holy water. When the coin was deposited, it fell upon a pan attached to a lever. The lever opened a valve which let some water flow out. The pan continued to tilt with the weight of the coin until it fell off, at which point a counterweight snapped the lever up and turned off the valve.

Coin-operated machines that dispensed tobacco were being operated as early as 1615 in the taverns of England. The machines were portable and made of brass.

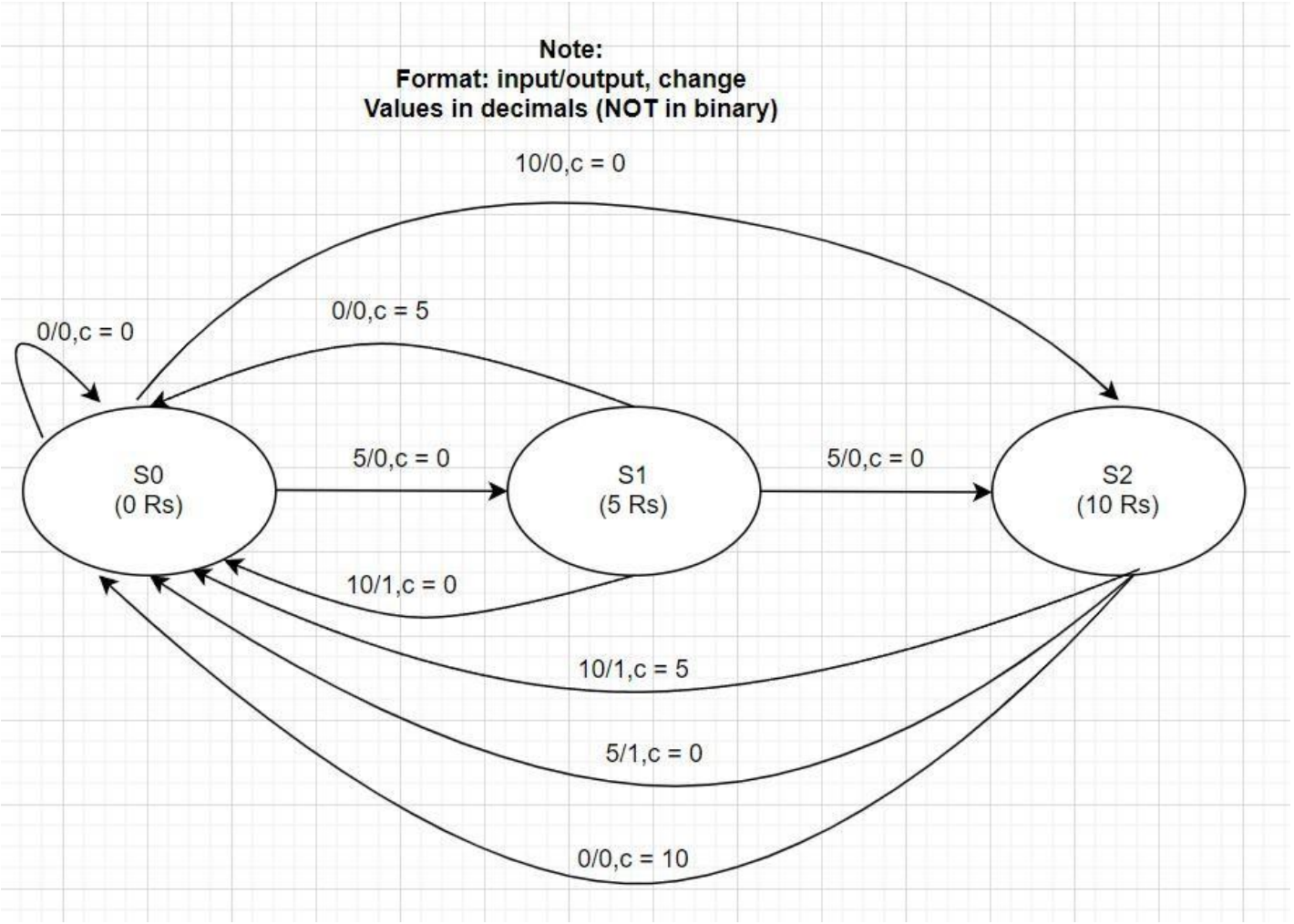
An English bookseller, Richard Carlile, devised a newspaper dispensing machine for the dissemination of banned works in 1822. Simon Denham was awarded British Patent no. 706 for his stamp dispensing machine in 1867, the first fully automatic vending machine.

Internal communication in vending machines is typically based on the MDB standard, supported by National Automatic Merchandising Association (NAMA) and European Vending & Coffee Service Association (EVA).

After payment has been tendered, a product may become available by:
the machine releasing it, so that it falls in an open compartment at the bottom, or into a cup, either released first, or put in by the customer, or the unlocking of a door, drawer, or turning of a knob.Some products need to be prepared to become available. For example, tickets are printed or magnetized on the spot, and coffee is freshly concocted. One of the most common forms of vending machine, the snack machine, often uses a metal coil which when ordered rotates to release the product. The main example of a vending machine giving access to all merchandise after paying for one item is a newspaper vending machine (also called vending box) found mainly in the U.S. and Canada. It contains a pile of identical newspapers. After a sale the door automatically returns to a locked position. A customer could open the box and take all of the newspapers or, for the benefit of other customers, leave all of the newspapers outside of the box, slowly return the door to an unlatched position, or block the door from fully closing, each of which are frequently discouraged, sometimes by a security clamp. The success of such machines is predicated on the assumption that the customer will be honest (hence the nickname "honor box"), and need only one copy.

State Diagram:

The product water bottle costs 15 Rs.



STATES TRANSITIONS:

The product costs 15 Rs.

S0 : 0 Rs in Vending Machine –

- **If nothing added: Stay on State 0, OUTPUT = 0, CHANGE = 0.**
- **If 5 Rs added: Move to State 1, OUTPUT = 0, CHANGE = 0.**
- **If 10 Rs added: Move to State 2, OUTPUT = 0, CHANGE = 0.**

S1 : 5 Rs in Vending Machine –

- **If nothing added:** Here, this means the vending machine waited some time but no money was added signifying an incomplete transaction, thus the vending machine should return back the money added as CHANGE (5 Rs). No bottle given.

  Move to State 0, OUTPUT = 0, CHANGE = Rs 5 (01).
- **If 5 Rs added: Move to State 1, OUTPUT = 0, CHANGE = 0.**
- **If 10 Rs added:** Adding 10 Rs means the vending machine now has 15 Rs total thus, a bottle will be returned with no CHANGE.

  Move to State 0, OUTPUT = 1, CHANGE = Rs 0.

S2 : 10 Rs in Vending Machine –

- **If nothing added:** Again, incomplete transaction thus vending machine returns the money added as CHANGE (10 Rs). No bottle given.

  Move to State 0, OUTPUT = 0, CHANGE = Rs 10 (10).
- **If 5 Rs added:** Signifies a complete transaction, a bottle is returned

with no CHANGE.

Move to State 0, OUTPUT = 1, CHANGE = Rs 0.

- **If 10 Rs added:** Here the customer over payed, thus a bottle should be returned but with CHANGE(5 Rs).

  Move to State 0, OUTPUT = 1, CHANGE = Rs 5 (01).

ABOUT LANGUAGE USED:

Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits, as well as in the design of genetic circuits. In 2009, the Verilog standard (IEEE 1364-2005) was merged into the SystemVerilog standard, creating IEEE Standard 1800-2009. Since then, Verilog is officially part of the SystemVerilog language. The current version is IEEE standard 1800-2017.

Hardware description languages such as Verilog are similar to software programming languages because they include ways of describing the propagation time and signal strengths (sensitivity). There are two types of assignment operators; a blocking assignment (=), and a non-blocking (<=) assignment. The non-blocking assignment allows designers to describe a state-machine update without needing to declare and use temporary storage variables. Since these concepts are part of Verilog's language semantics, designers could quickly write descriptions of large circuits in a relatively compact and concise form. At the time of Verilog's introduction (1984), Verilog represented a tremendous productivity improvement for circuit designers who were already using graphical schematic capture software and specially written software programs to document and simulate electronic circuits.

The designers of Verilog wanted a language with syntax similar to the C programming language, which was already widely used in engineering software development. Like C, Verilog is case-sensitive and has a basic preprocessor (though less sophisticated than that of ANSI C/C++). Its control flow keywords (if/else, for, while, case, etc.) are equivalent, and its operator precedence is compatible with C. Syntactic differences include: required bit-widths for variable declarations, demarcation of procedural blocks (Verilog uses begin/end instead of curly braces {}), and many other minor differences. Verilog requires that variables be given a definite size. In C these sizes are inferred from the 'type' of the variable (for instance an integer type may be 8 bits).

A Verilog design consists of a hierarchy of modules. Modules encapsulate design hierarchy, and communicate with other modules through a set of declared input, output, and bidirectional ports. Internally, a module can contain any combination of the following: net/variable declarations (wire, reg, integer, etc.), concurrent and sequential statement blocks, and instances of other modules (sub-hierarchies). Sequential statements are placed inside a begin/end block and executed in sequential order within the block. However, the blocks themselves are executed concurrently, making Verilog a dataflow language.

Verilog's concept of 'wire' consists of both signal values (4-state: "1, 0, floating, undefined") and signal strengths (strong, weak, etc.). This system allows abstract modeling of shared signal lines, where multiple sources drive a common net. When a wire has multiple drivers, the wire's (readable) value is resolved by a function of the source drivers and their strengths.

A subset of statements in the Verilog language are synthesizable. Verilog modules that conform to a synthesizable coding style, known as RTL (register-transfer level), can be physically realized by synthesis software. Synthesis software algorithmically transforms the (abstract) Verilog source into a netlist, a logically equivalent description consisting only of elementary logic primitives (AND, OR, NOT, flip-flops, etc.) that are available in a specific FPGA or VLSI technology. Further manipulations to the netlist ultimately lead to a circuit fabrication blueprint (such as a photo mask set for an ASIC or a bitstream file for an FPGA).

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 = 0;
          n_state = 0;
        change = 2'b00;
        end
      else
        c_state = n_state;

      case(c_state)
        s0: //state 0 : 0 rs
        if(in == 0)
          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
s1:  //state 1 : 5 rs
if(in == 0)
  begin
    n_state = s0;
    out = 0;
    change = 2'b01; //change returned 5 rs
  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
s2:  //state 2 : 10 rs
if(in == 0)
  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; //change returned 5 rs and 1 bottle
  end
```

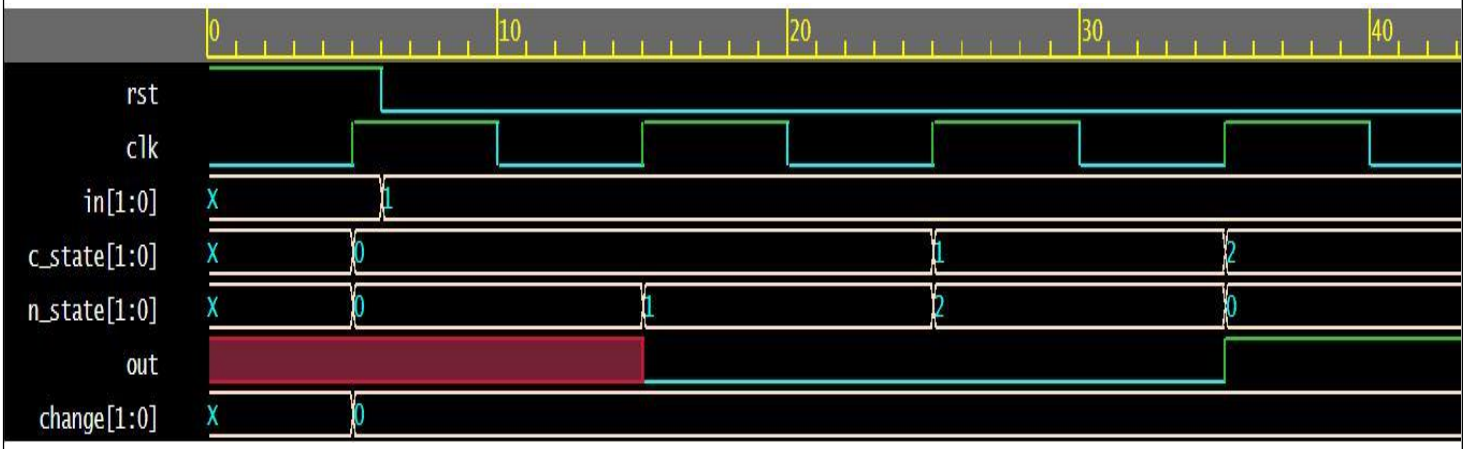
```
      endcase
    end
endmodule
```

TEST BENCH CODE:

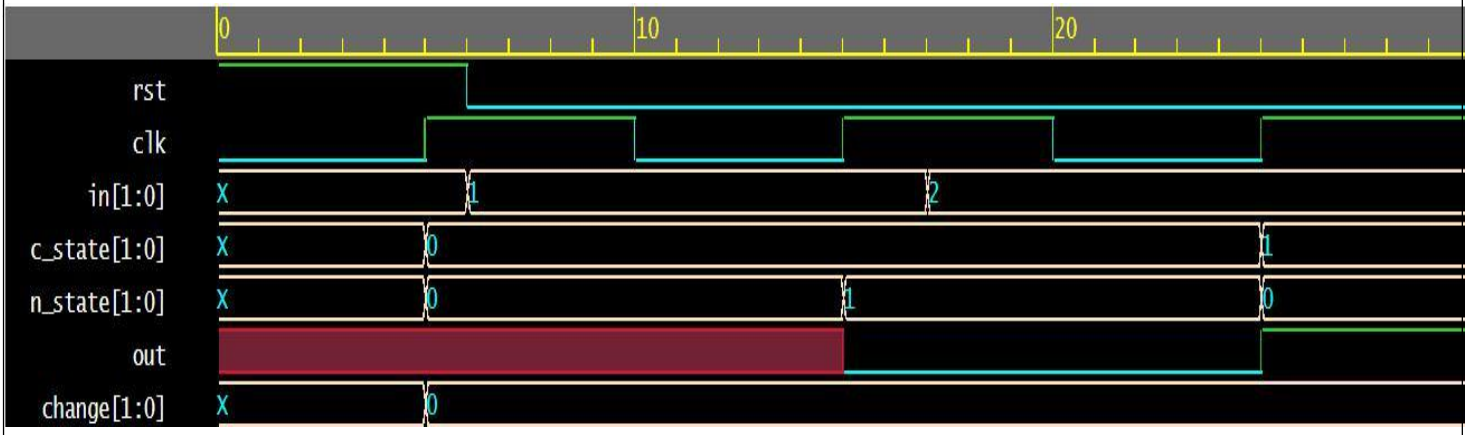
```
module vending_machine_tb;

  //inputs
  reg clk;
  reg[1:0] in;
  reg rst;

  //output
  wire out;
  wire[1:0] change;

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

  initial begin

    //initialise inputs
    $dumpfile("vending_machine_18105070.vcd");
    $dumpvars(0,vending_machine_tb);
    rst = 1;
    clk = 0;

    #6 rst = 0;
    in = 2;
    #19 in = 2;
    #25 $finish;

  end
  always #5 clk = ~clk;

endmodule
```

RESULTS:

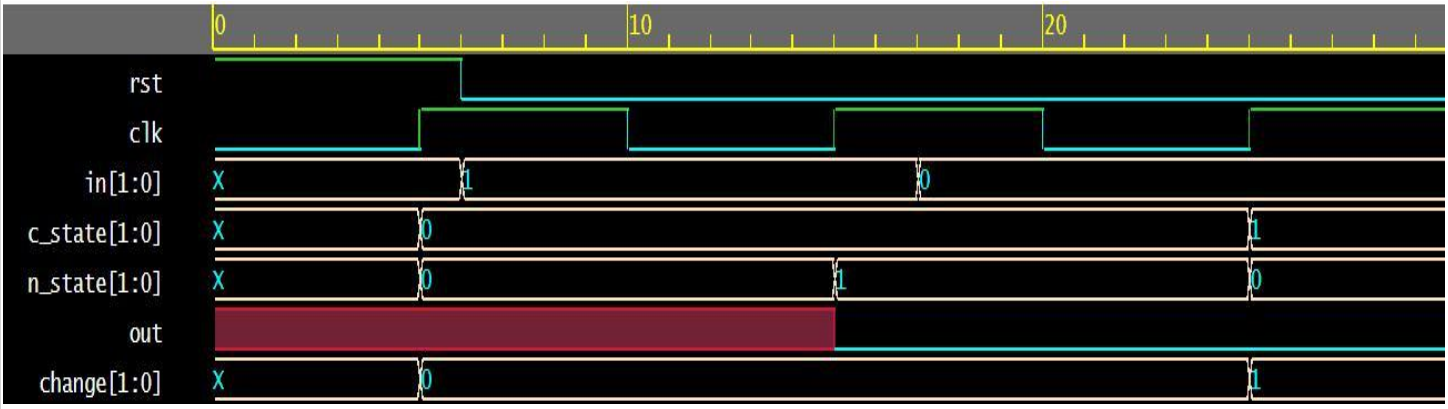
1. Adding 5 Rs three times consecutively



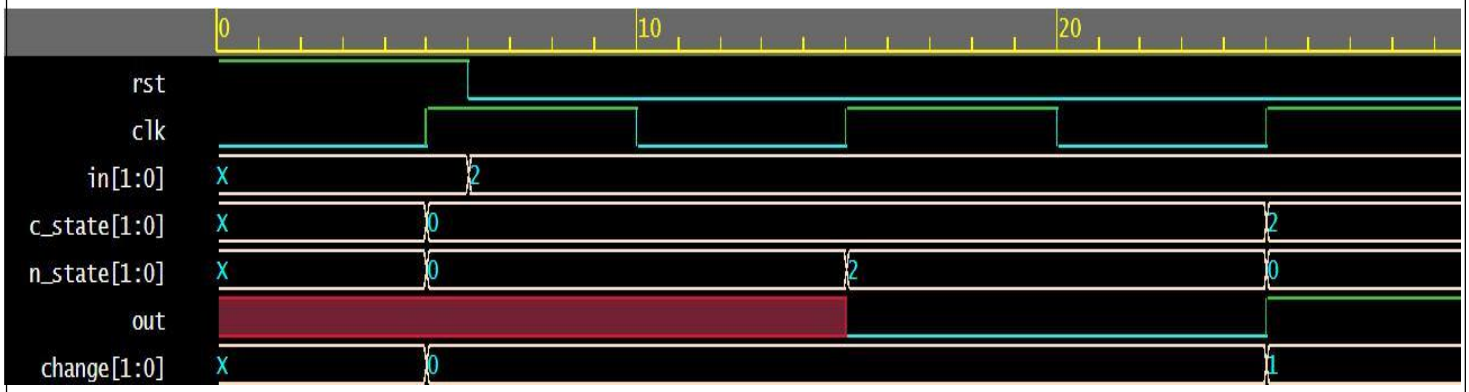
2. Adding 5 Rs and then 10 Rs



3. Adding 10 Rs two times



4. Adding 5 Rs and then nothing



5. Adding 10 Rs and then nothing