VENDING MACHINE

AIM: Implement a Vending Machine using Verilog

TOOLS USED: Xilinx Vivado 2014

THEORY: Vending machine is an automated system that provides items such as snacks and beverages to consumers when they deposit a certain amount into the machine. A vending machine is an example of a Finite state machine FSM as the number of states in a Vending machine are finite.

In this project the designed Vending Machine offers two products (Chocolate costs 2 Dollars and Drink costs 5 Dollars) and takes 3 coins as input (1 Dollar, 2 Dollar and 5 Dollar coins). Facility of return is available if an amount greater than price of product is deposited.

State diagram of the machine is shown below:

Encoding used:

S0: 000
S1: 001
S2: 010
S3: 011
S4: 100

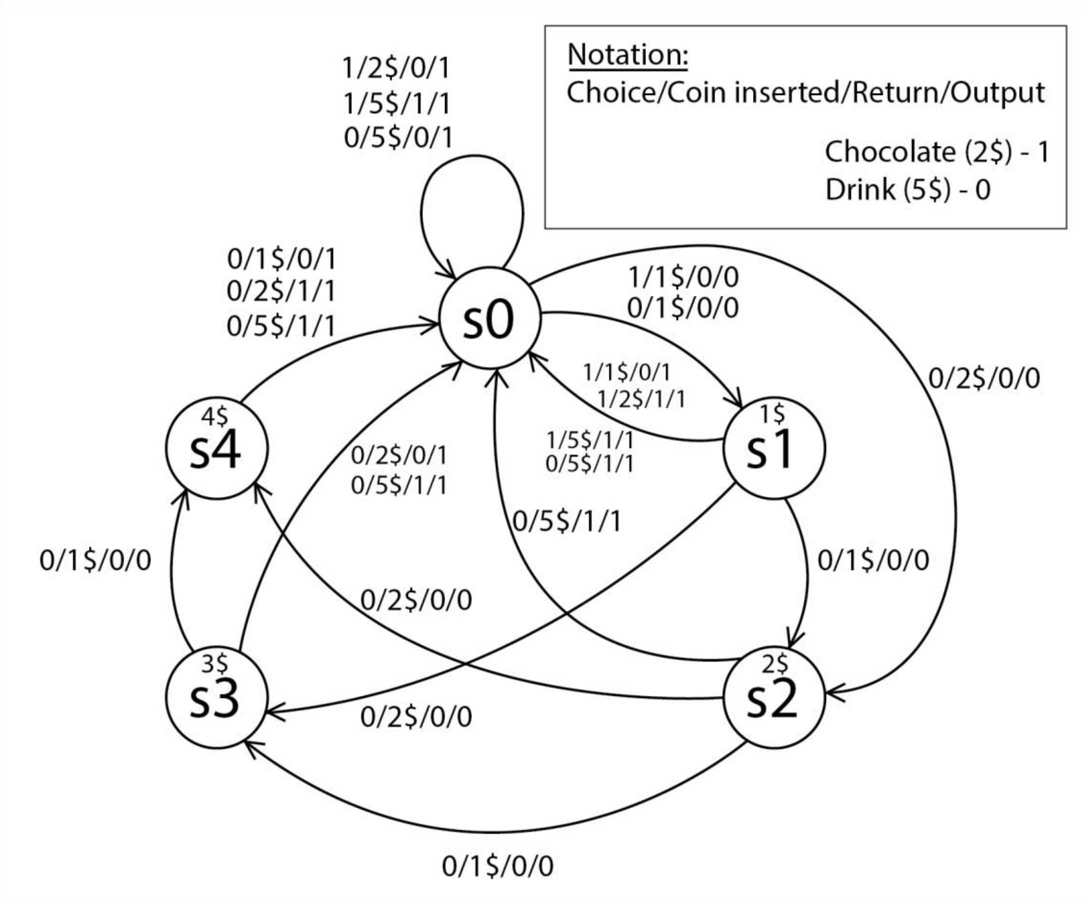

Figure 1. State Diagram of Vending Machine

Verilog Code:

```
module VendingMachine(out,ret,i,choice,rst,clk);

input [2:0]i;   // i[0] = 1 Dollar Coin , i[1] = 2 Dollar Coin , i[2] = 5 Dollar Coin
input rst;
input choice;            //choice = 1 for Chocolate and 0 for Drink
input clk;
output reg out;
output reg [2:0]ret;

parameter s0=3'b000, s1=3'b001, s2=3'b010, s3=3'b011, s4=3'b100;      //Encoding
reg [2:0]pr_state,nxt_state;
always@(posedge clk)            //Sequential Logic Block
begin
    if(rst)
      pr_state<=0;
    else
      pr_state<=nxt_state;
end

always@(i,pr_state)           // Next State Combinational Logic
begin
  if(choice)
   case(pr_state)
     s0: if(i[0])  nxt_state=s1;                    //Output = 0
        else if(i[1])  nxt_state=s0;                // Output = 1
        else if(i[2])  nxt_state=s0;        //Output = 1 But return change
        else nxt_state=s0;                          // Output = 0
     s1: if(i[0])  nxt_state=s0;                   // Output 1
        else if(i[1])    nxt_state=s0;              //Output = 1 and return
        else if(i[2])    nxt_state=s0;            //Output = 1 and return
        else  nxt_state=s1;                       //wait for coins
     default: $display("Insert Coins");
    endcase
  else
    case(pr_state)
      s0: if(i[0])    nxt_state=s1;                       //Output = 0
         else if(i[1])       nxt_state=s2;                //Output = 0
         else if(i[2])         nxt_state=s0;     // Output = 1 and return change
          else  nxt_state=s0;                             //wait for coins
      s1: if(i[0])      nxt_state=s2;
```

```
            else if(i[1])   nxt_state=s3;
            else if(i[2])   nxt_state=s0;         //Output = 1 and return money
            else  nxt_state=s1;                              // Wait for coin
         s2:  if(i[0])   nxt_state=s3;                       //output =0
             else if(i[1])  nxt_state=s4;                   //Output = 0;
             else if(i[2])  nxt_state=s0;                   // Output = 1 and return
             else nxt_state=s2;                             //wait for coins
         s3:  if(i[0])   nxt_state=s4;                      //Output = 0;
             else if(i[1])   nxt_state=s0;                 // Output = 1 and no return
             else if(i[2])  nxt_state=s0;            //Output = 1 and Return money
             else  nxt_state=s3;                           //Wait fo coins
         s4:  if(i[0])        nxt_state=s0;                //Output = 1 No return
             else if(i[1])      nxt_state=s0;              //Output = 1 and return
             else if(i[2])      nxt_state=s0;              //Output = 1 and return
             else  nxt_state=s4;                          //Wait fo coins
          default: $display("Insert Coins");
        endcase
end

always@(*)                  //Output Combinational Logic: Mealy Machine
begin
   if(choice)
      case(pr_state)
         s0: if(i[0])   {out,ret}<=4'b0000;
            else if(i[1])   {out,ret}<=4'b1000;
            else if(i[2])    {out,ret}<=4'b1011;
            else {out,ret}<=4'b0000;
         s1: if(i[0])   {out,ret}<=4'b1000;
            else if(i[1])   {out,ret}<=4'b1001;
            else if(i[2])    {out,ret}<=4'b1100;
            else {out,ret}<=4'b0000;
         default: {out,ret}<=4'b0000;
       endcase
   else
      case(pr_state)
         s0: if(i[0])   {out,ret}<=4'b0000;
            else if(i[1])   {out,ret}<=4'b0000;
            else if(i[2])    {out,ret}<=4'b1000;
            else {out,ret}<=4'b0000;
         s1: if(i[0])   {out,ret}<=4'b0000;
            else if(i[1])   {out,ret}<=4'b0000;
```

```
                else if(i[2])    {out,ret}<=4'b1001;
                else {out,ret}<=4'b0000;
            s2: if(i[0])   {out,ret}<=4'b0000;
                else if(i[1])   {out,ret}<=4'b0000;
                else if(i[2])    {out,ret}<=4'b1010;
                else {out,ret}<=4'b0000;
            s3: if(i[0])   {out,ret}<=4'b0000;
                else if(i[1])   {out,ret}<=4'b1000;
                else if(i[2])    {out,ret}<=4'b1011;
                else {out,ret}<=4'b0000;
            s4: if(i[0])   {out,ret}<=4'b1000;
                else if(i[1])   {out,ret}<=4'b1001;
                else if(i[2])    {out,ret}<=4'b1100;
                else {out,ret}<=4'b0000;
            default:    {out,ret}<=4'b0000;
          endcase
    end
    endmodule
```

TestBench:

```
    module tb;
    reg [2:0]i;
    reg rst,choice,clk;
    wire out;
    wire [2:0]ret;

    VendingMachine VM(out,ret,i,choice,rst,clk);
    always #5 clk=~clk;

      initial
      begin
        clk=0;  rst=1'b1;
        #7 rst=0;
        #7 i=3'b100; choice=1'b1;
        #7 i=3'b001;
        #7 choice=1'b0;
        #7 i=3'b010;
        #7 i=3'b100;
        #7 i=3'b001;
        #7 choice=1'b1;
    end
endmodule
```

OUTPUT WAVEFORM:

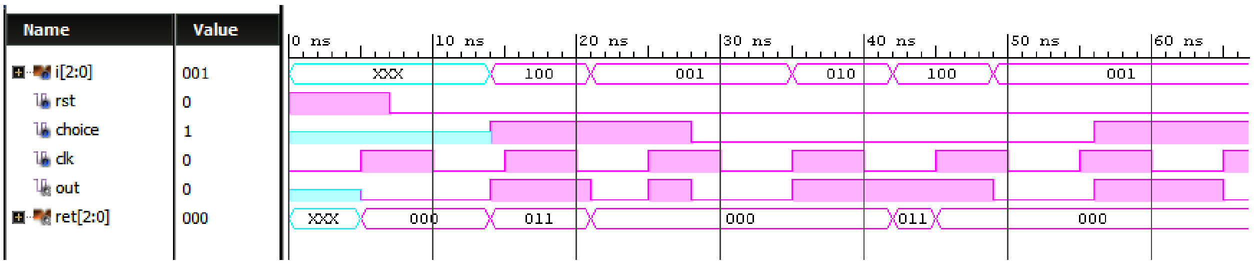

Figure 2: Output Waveform

RESULT: A vending machine offering 2 products (Drink and Chocolate) and accepting 3 coins as input was implemented using Verilog. Its output waveform was plotted and verified.