# **Internship Project 1 - Vending Machine using Verilog HDL**

## **Design Block:-**

```
`timescale 1ns / 1ps
```

```
module
```

Vending\_Machine(clk,start,reset,cancel,products,product\_price,state,amount\_given,online\_payment,return\_change,dispense\_product);

```
//Input Declaration
input clk, start, reset, cancel, online_payment;
input [2:0]products;
input [6:0]amount given;
//Output Declaration for Outside World
output wire [3:0]state;
output wire dispense_product;
output wire [7:0]product_price;
output wire [7:0]return_change;
//Products Available
parameter Pen = 3'b000;
parameter Note = 3'b001;
parameter Book = 3'b010;
parameter Lays = 3'b011;
parameter Kurkure = 3'b100;
parameter Coke = 3'b101;
parameter Sprite = 3'b110;
//Product Price
parameter Pen_Price = 8'd15;
parameter Note_Price = 8'd65;
parameter Book_Price = 8'd80;
parameter Lays_Price = 8'd10;
parameter Kurkure_Price = 8'd20;
parameter Coke_Price = 8'd40;
parameter Sprite_Price = 8'd99;
```

```
parameter Idle_State
                            = 4'b0000;
parameter Product_Select_State
                                   = 4'b0001;
parameter Pen_Select_State
                                = 4'b0010;
parameter Note_Select_State
                                 = 4'b0011;
parameter Book_Select_State
                                 = 4'b0100;
parameter Lays Select State
                                 = 4'b0101;
parameter Kurkure_Select_State
                                  = 4'b0110;
parameter Coke_Select_State
                                 = 4'b0111;
parameter Sprite Select State
                                 = 4'b1000;
parameter Dispense_and_Return_State = 4'b1001;
//variables for Internal functions
reg [3:0] next_state, present_state;
reg [7:0] next_product_price, present_product_price;
reg [7:0] next_return_change, present_return_change;
      dispense and retun state;
reg
always @(posedge clk or posedge reset)
                                               //Asynchronous Clock Declaration
 begin
   if(reset) begin
                        <= Idle_State;
     present_state
     present_product_price <= 0;</pre>
     present_return_change <= 0;</pre>
    end
    else begin
     present_state
                        <= next_state;
     present_product_price <= next_product_price;</pre>
     present_return_change <= next_return_change;</pre>
   end
```

//States

end

```
always @(*)
 begin
   next_state
                = present_state;
   next_product_price = present_product_price;
   next_return_change = present_return_change;
   case (present_state)
     Idle_State: begin
       if(start)
         next_state = Product_Select_State;
       else if(cancel)
         next_state = Idle_State;
       else
         next_state = Idle_State;
     end
     Product_Select_State: begin
       case (products)
         Pen: begin
                         = Pen_Select_State;
           next_state
           next_product_price = Pen_Price;
         end
         Note: begin
           next_state
                         = Note_Select_State;
           next_product_price = Note_Price;
         end
         Book: begin
           next_state
                         = Book_Select_State;
           next_product_price = Book_Price;
         end
         Lays: begin
```

```
next_product_price = Lays_Price;
           end
           Kurkure: begin
             next_state
                           = Kurkure_Select_State;
             next_product_price = Kurkure_Price;
           end
           Coke: begin
             next_state
                           = Coke_Select_State;
            next_product_price = Coke_Price;
           end
           Sprite: begin
                           = Sprite_Select_State;
             next_state
             next_product_price = Sprite_Price;
           end
           default: begin
             next state
                           = Idle State;
             next_product_price = 0;
            next_return_change = 0;
           end
         endcase
       end
Pen_Select_State, Note_Select_State, Book_Select_State, Lays_Select_State, Kurkure_Select_State, Coke_Sel
ect_State,Sprite_Select_State:begin
         if(cancel) begin
           next_state = Idle_State;
           next_return_change = amount_given;
         end
         else if(amount_given >= present_product_price)
           next_state = Dispense_and_Return_State;
```

= Lays\_Select\_State;

next\_state

```
else if(online_payment)
         next_state = Dispense_and_Return_State;
       else
         next_state = present_state;
     end
     Dispense_and_Return_State:begin
       if(amount_given >= product_price) begin
         next_state = Idle_State;
         next_return_change = amount_given - present_product_price;
       end
       else if(online_payment) begin
         next_state = Idle_State;
         next_return_change = 8'd0;
       end
       else begin
         next_state = Idle_State;
         next_return_change = present_return_change;
       end
     end
     default: begin
           next_state
                         = Idle_State;
           next_product_price = 0;
           next_return_change = 0;
         end
   endcase
 end
assign state
                = present_state;
assign dispense_product = (present_state == Dispense_and_Return_State) ? 1
                                                                                   : 0;
assign return_change = (present_state == Dispense_and_Return_State) ? next_return_change : 0;
assign product_price = (present_state == Dispense_and_Return_State) ? next_product_price : 0;
```

#### TestBench Block:-

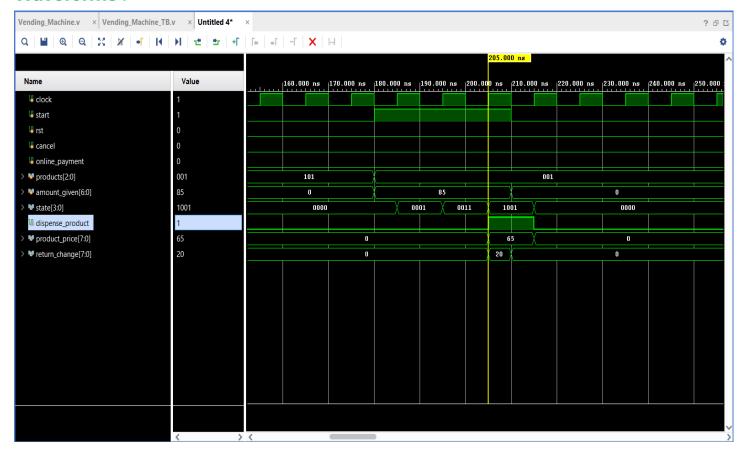
```
`timescale 1ns / 1ps
module Vending_Machine_TB;
 reg clock, start, rst, cancel, online_payment;
 reg [2:0]products;
 reg [6:0]amount_given;
 wire [3:0]state;
 wire dispense_product;
 wire [7:0]product_price;
 wire [7:0] return_change;
 Vending_Machine
V1(clock,start,rst,cancel,products,product_price,state,amount_given,online_payment,return_change,dispen
se_product);
 always #5 clock = ~clock;
 initial
   begin
     clock
               = 1'b0;
     rst
             = 1'b1;
     start
              = 1'b0;
     cancel
                = 1'b0;
     online_payment = 1'b0;
     amount_given = 1'b0;
     products
                 = 1'b0;
```

```
#50 rst
             = 1'b0;
#50;
         = 1'b1;
start
products = 3'b101;
online_payment = 1'b1;
#30 \text{ start} = 1'b0;
online_payment = 1'b0;
#50;
start
        = 1'b1;
products = 3'b001;
amount_given = 8'd85;
#30 \text{ start} = 1'b0;
amount_given = 8'd0;
#50;
start
        = 1'b1;
products = 3'b011;
amount_given = 8'd10;
#30 \text{ start} = 1'b0;
amount_given = 8'd0;
#50;
start
        = 1'b1;
products = 3'b110;
amount_given = 8'd110;
#30 \text{ start} = 1'b0;
amount_given = 8'd0;
#50;
start
        = 1'b1;
```

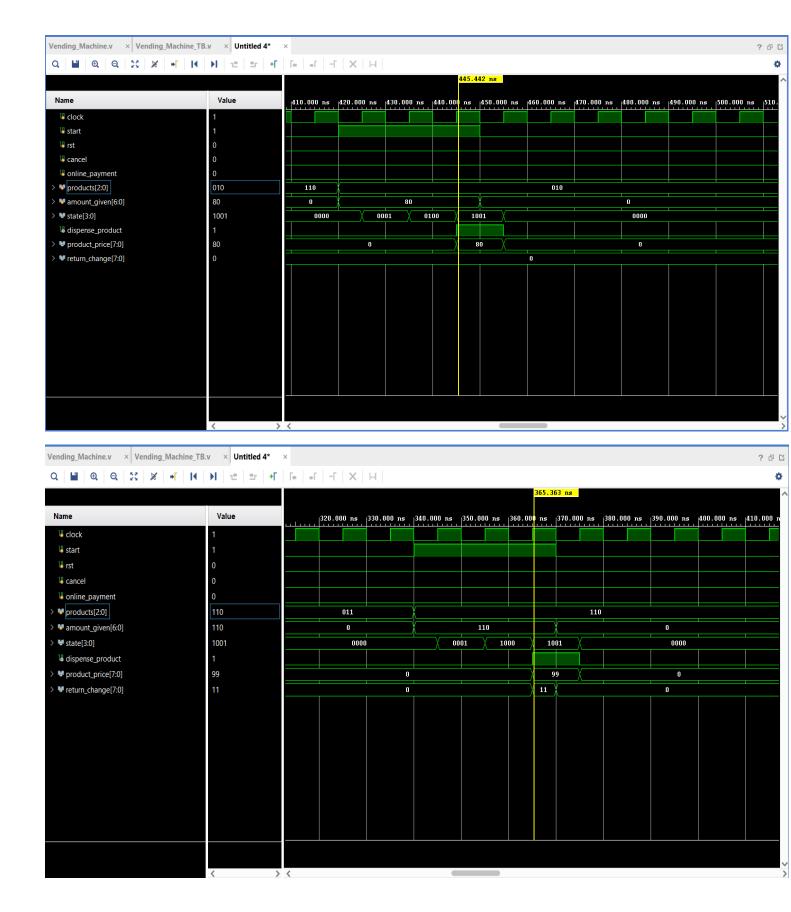
```
products = 3'b010;
amount_given = 8'd80;
#30 start = 1'b0;
amount_given = 8'd0;
#500 $finish;
end
```

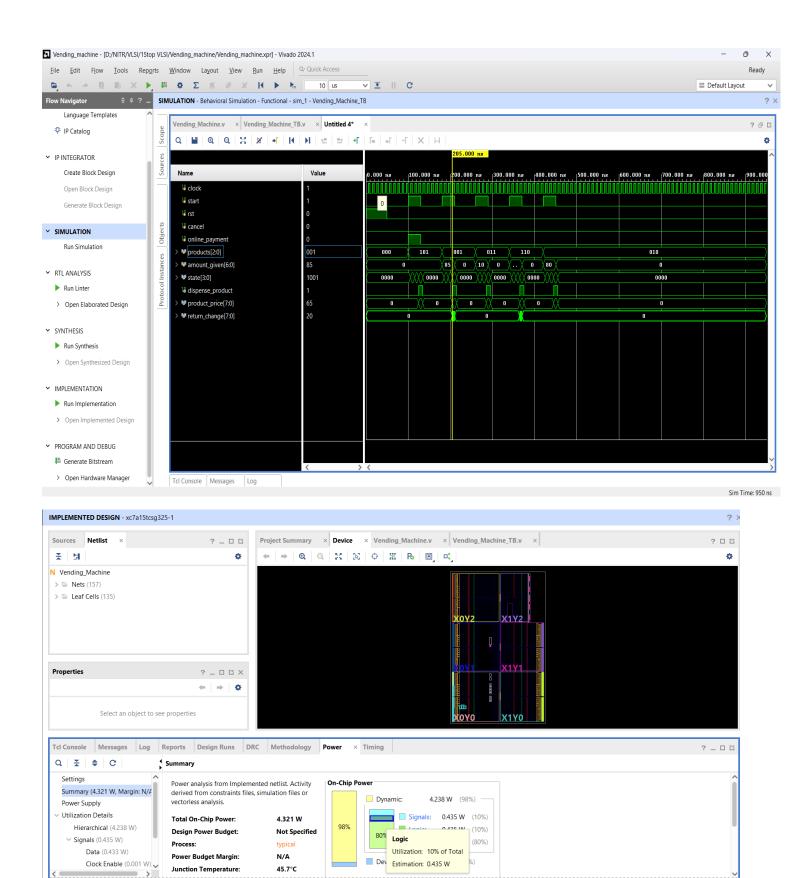
#### endmodule

### Waveforms:-

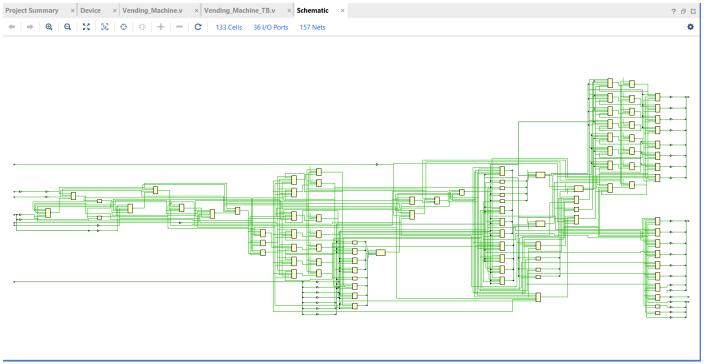


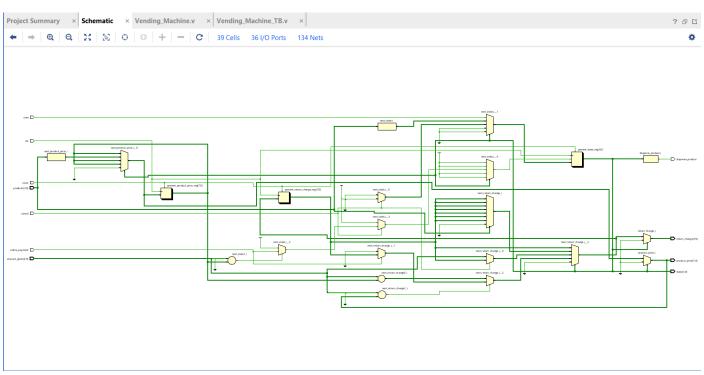


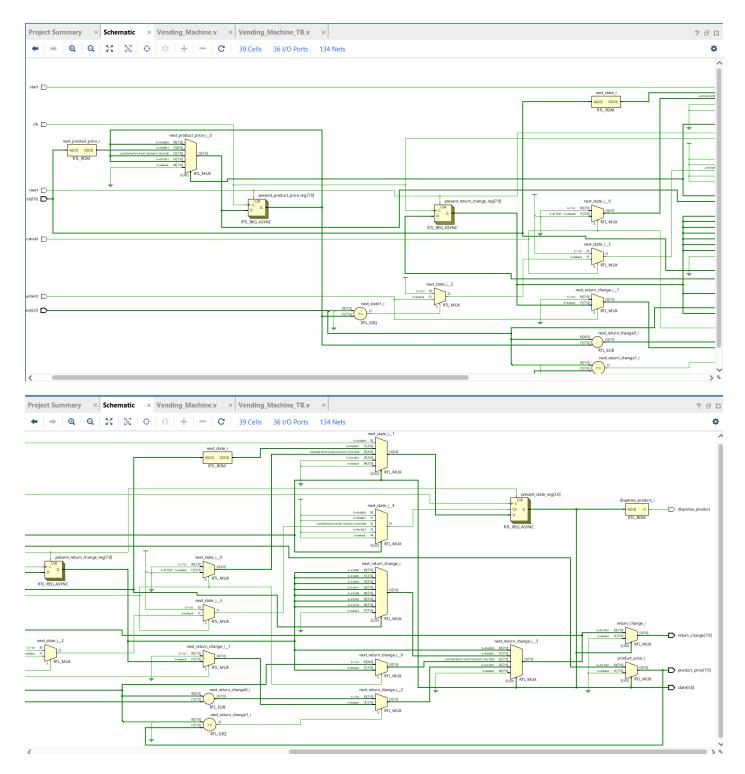




impl\_1 (saved)







https://github.com/SubhEE27/VLSI\_Projects