Report Vending Machine(Self Project)

Name:Akul Kumar Singh

Roll:22M1133

Vending Machine

A vending machine is a mechanized or electronic device designed to autonomously dispense products or services to customers upon the insertion of the appropriate payment or credits. It operates through a combination of user interaction, control logic, and mechanical components. In this project, we explore the theoretical foundation of a digital vending machine implemented using Verilog, highlighting key concepts integral to its design and functionality.

Components of a Vending Machine

User Interface: The user interface comprises buttons or a keypad for product selection and an optional display for conveying information to the user.

Coin/Credit Acceptor: This component verifies and processes inserted coins or credits, calculating the total payment made by the user.

Product Dispenser: The product dispenser releases the selected item to the user upon successful payment.

Control Logic: The control logic orchestrates the interaction between the user, payment processing, product availability, and stock management.

Vending Process

The vending process involves several sequential steps:

User Interaction: The user selects a desired product by pressing the corresponding button on the interface.

Payment Processing: The coin or credit acceptor mechanism validates and accumulates the inserted money, determining if the payment is sufficient for the selected product.

Product Selection Confirmation: The control logic verifies product availability and confirms the selection if payment is adequate.

Dispensing: Upon confirmation, the product dispenser is activated to release the chosen item to the user.

Finite State Machine (FSM) Model

A Finite State Machine (FSM) is employed to model the behavior of the vending machine. Different states correspond to distinct phases of the vending process, including idle, product selection, purchase confirmation, and restocking.

Combinational Logic

Combinational logic circuits are utilized to process user inputs, calculate the total payment, determine product availability, and manage stock levels. These circuits perform calculations and generate control signals to facilitate the vending process.

Coin Processing and Payment Calculation

The coin processing mechanism detects and validates inserted coins or credits. The control logic tracks the cumulative payment by integrating the values of the accepted coins.

Product Selection and Dispensing Logic

User input for product selection triggers the control logic to evaluate the availability of the chosen item. If the product is available and payment is sufficient, the logic activates the dispenser to release the product.

Stock Management and Restocking

The vending machine maintains stock levels for each product. Upon user request or manual intervention, stock levels can be updated to ensure product availability.

Simulation and Verification

Simulation tools like Verilog are employed to model and verify the vending machine's behavior. Simulation aids in identifying design flaws, ensuring proper functionality, and refining the control logic.

Applications and Impact

Vending machines find applications in various sectors, enhancing convenience and automating transactions in retail, transportation hubs, and hospitality environments. Automation through vending machines streamlines customer interactions and contributes to efficient service delivery.

Code for vending machine

```
module vendingmachine(clk,coin1,coin2,select,buy,load,money,products,outofstock);
input clk; input coin1; //25 cents
input coin2; //1 dollar (100 cents)
input [3:0] select; input buy; input [3:0] load;
output reg [11:0] money=0; output reg [3:0] products=0;
output reg [3:0] outofstock=0;
reg coin1_prev=0,coin2_prev=0; reg buy_prev=0;
reg [3:0] stock1=4'b1010; reg [3:0] stock2=4'b1010;
reg [3:0] stock3=4'b1010; reg [3:0] stock4=4'b1010;


always @ (posedge clk) begin
coin1_prev <= coin1; coin2_prev <= coin2; buy_prev <= buy;
if (coin1_prev == 1'b0 && coin1 == 1'b1) money <= money + 12'd25;
else if (coin2_prev == 1'b0 && coin2 == 1'b1) money <= money + 12'd100;
else if (buy_prev == 1'b0 && buy == 1'b1) begin
case (select)
4'b0001: if (money >= 12'd25 && stock1 > 0) begin
products[0] <= 1'b1; stock1 <= stock1 - 1'b1; money <= money - 12'd25;
end
4'b0010: if (money >= 12'd75 && stock2 > 0) begin
products[1] <= 1'b1; stock2 <= stock2 - 1'b1; money <= money - 12'd75;
end
4'b0100: if (money >= 12'd150 && stock3 > 0) begin products[2] <= 1'b1; stock3 <= stock3 -
1'b1; money <= money -
12'd150; end
4'b1000: if (money >= 12'd200 && stock4 > 0) begin products[3] <= 1'b1; stock4 <= stock4 -
1'b1; money <= money -
12'd200; end
endcase
end
else if (buy_prev == 1'b1 && buy == 1'b0 || buy_prev == 1'b1 && buy == 1'b1 || buy_prev
== 1'b0 && buy == 1'b0 ) begin products[0] <= 1'b0; products[1] <= 1'b0; products[2] <=
1'b0;
products[3] <= 1'b0; end
else begin
if (stock1 == 4'b0) outofstock[0] <= 1'b1; else outofstock[0] <= 1'b0;
if (stock2 == 4'b0) outofstock[1] <= 1'b1; else outofstock[1] <= 1'b0;
if (stock3 == 4'b0) outofstock[2] <= 1'b1; else outofstock[2] <= 1'b0;
if (stock4 == 4'b0) outofstock[3] <= 1'b1; else outofstock[3] <= 1'b0;


case (load)
4'b0001: stock1 <= 4'b1111;
4'b0010: stock2 <= 4'b1111;
4'b0100: stock3 <= 4'b1111;
```

```
4'b1000: stock4 <= 4'b1111;
endcase
end
end
endmodule
```

TESTBENCH

```
module testbench;
reg coin1,coin2,buy,clk;
reg [3:0] select;
reg [3:0]load;
wire [11:0] money;
wire [3:0] products;
wire [3:0] outofstock;

initial
clk=1'b0;
always
#5 clk=~clk;

vendingmachine dut(clk,coin1,coin2,select,buy,load,money,products,outofstock);

initial
begin
coin1=0;
coin2=0;select=4'd0;buy=0;load=4'd0;
end
initial
begin

$monitor("coin1=%b,coin2=%b,select=%h,buy=%b,load=%h,money=%h,products=%h,outofs
tck=%h", coin1,coin2,select,buy,load,money,products,outofstock );
#5 coin1=1'b1;
#3 coin1=1'b0;
#5 coin2=1'b1;
#3 coin2=1'b0;
#3 select=4'd4;
#5 coin1=1'b1;
#3 coin1 =1'b0;
#7 coin2=1'b1;

#3 coin2=1'b0;
#5 buy=1'b1;
#6 select = 4'd2;
#6 buy = 1'b1;
```

```
end

initial
#10000 $finish;
endmodule
```

Simulation Result: