Soda Machine Dispenser

Course Name: Digital Systems Design (ECE 115)

Instructor Names:

1. Dr. Hany M. Zamil
2. Eng. Amr Al-Iraqi

Team Name: 5leha 3la Allah

“5leha 3la Allah” Members:

Name	ID
Ahmed Mahmoud Zaytoun	23-101328
Seif Eldin Haytham	23-101282
Abdulrahaman Gomaa	23-101279

Content:

1. Introduction

Purpose:

is to design a Finite State Machine (FSM) that controls a soda vending machine.

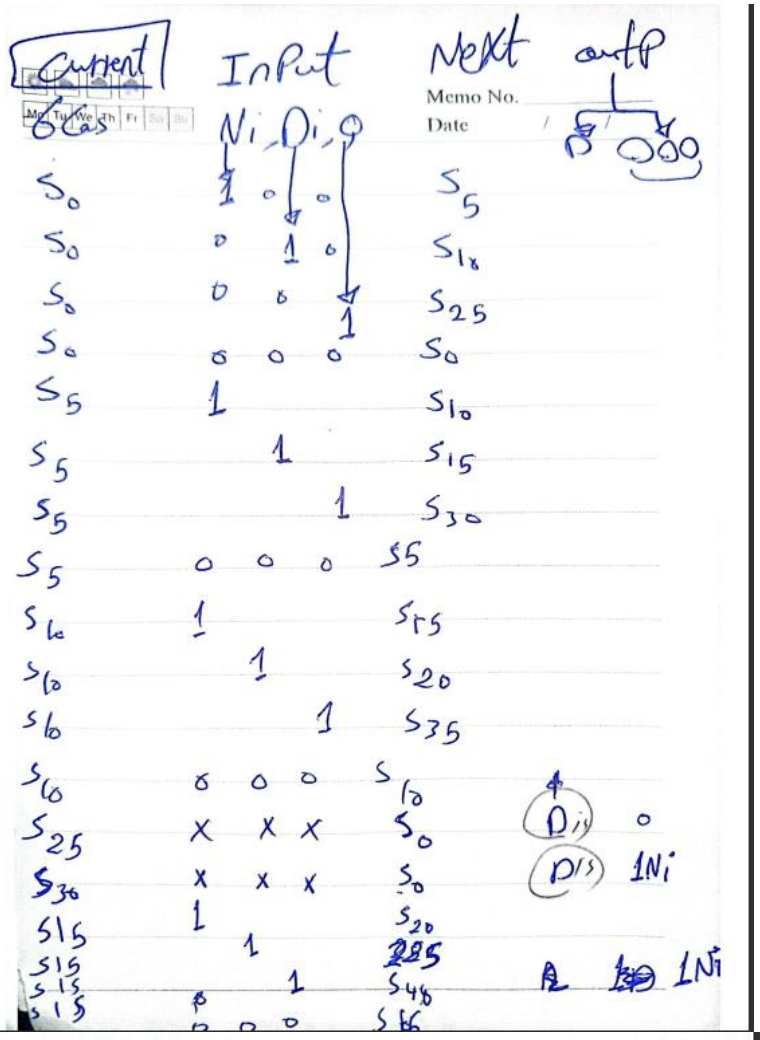
Functionality:

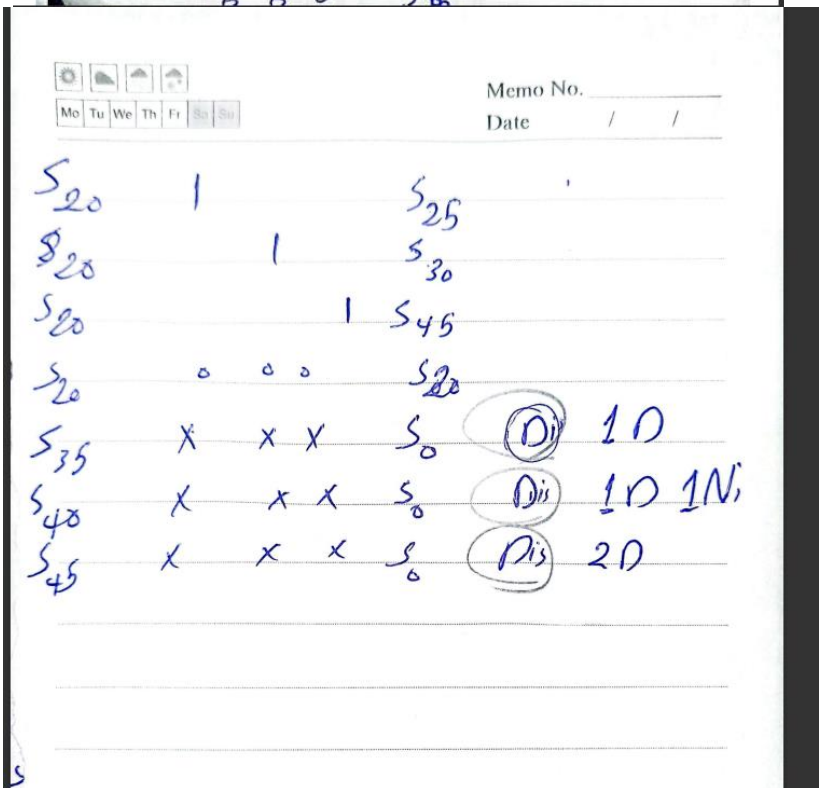
The machine accepts three types of coins: nickels (5 cents), dimes (10 cents), and quarters (25 cents). Once the total value of inserted coins reaches 25 cents, the machine dispenses a soda and returns any necessary change.

The FSM is designed to handle input efficiently, ensure correct outputs, and reset properly to start a new transaction.

This report documents the FSM design, testing scenarios, and simulation results.

2. How We think in project





3. FSM Design Code

Link for the files project on drive

```
module a5leha_3la_allah_new(
    input N,D,Q,
    input rst,clk,
    output reg dis,oN,oD,o2D
);

/*
    Inputs: N (Nickel), D (Dime), Q (Quarter)
    Inputs: rst (Reset), clk (Clock)
    dis (Dispense), oN (Return Nickel),
    oD (Return Dime), o2D (Return Two Dimes)

*/

reg [5:0] current,next;

parameter s0=0,   // Initial state: 0 cents
          s5=5,   // State: 5 cents
          s10=10, // State: 10 cents
          s15=15,s20=20,s25=25,s30=30,s35=35,s40=40,s45=45;

//Sequential block: Current state transitions

always@(posedge clk ,posedge rst)begin
    if(rst)begin
        current<=s0;   // Asynchronous reset: Go to initial state
    end
    else begin
        current<=next; // Update current state on positive edge of clock
    end
end
```

```
//combinational block: next state logic
// Coin inputs: N (Nickel), D (Dime), Q (Quarter)
//3'b100 when nickel inserted , 3'b010 dime inserted , 3'b001 quarter inserted  ,default = no coin inserted

always@(*)begin

    case(current)
        // Handle coin inputs in state s0
        s0:begin
            case ({N,D,Q})
                3'b100:next=s5;          // Nickel inserted
                3'b010:next=s10;       // Dime inserted
                3'b001:next=s25;     // Quarter inserted
                default:next=s0;   // No coin inserted
            endcase
        end

        s5:begin
            case ({N,D,Q})
                3'b100:next=s10;
                3'b010:next=s15;
                3'b001:next=s30;
                default: next=s5; // No coin inserted
            endcase
        end

        s10:begin
            case ({N,D,Q})
                3'b100:next=s15;
                3'b010:next=s20;
                3'b001:next=s35;
                default: next=s10; // No coin inserted
            endcase
        end

```

```
        s15:begin
            case ({N,D,Q})
                3'b100:next=s20;
                3'b010:next=s25;
                3'b001:next=s40;
                default: next=s15; // No coin inserted
            endcase
        end

        s20:begin
            case ({N,D,Q})
                3'b100:next=s25;
                3'b010:next=s30;
                3'b001:next=s45;
                default: next=s20; // No coin inserted
            endcase
        end

        s25:begin
            next=s0;     // Reset to initial state
        end
        s30:begin
            next=s0;     // Reset to initial state
        end
        s35:begin
            next=s0;     // Reset to initial state
        end
        s40:begin
            next=s0;     // Reset to initial state
        end
        s45:begin
            next=s0;     // Reset to initial state
        end
        default: next =current;// Default case to avoid latches
    endcase
end



// Combinational block: Output logic
// Outputs depend on the current state

always@(*)begin
    // Default output values
    dis=0; oN=0; oD=0; o2D=0;

    case(current)

        s25:begin
            dis=1;  // Dispense soda
        end

        s30:begin
            dis=1;oN=1;     // Dispense soda + Return 1 Nickel
        end

        s35:begin
            dis=1; oD=1;    // Dispense soda + Return 1 Dime
        end

        s40:begin
            dis=1;oN=1;oD=1;    // Dispense soda + Return 1 Nickel + 1 Dime
        end

        s45:begin
            dis=1;o2D=1;    // Dispense soda + Return 2 Dimes
        end

    endcase
end
endmodule
```

4. Testbench code

Link for the files project on drive

This section includes the testbench code used to verify the functionality of the FSM.

The inputs are provided at each negative clock edge.

Each scenario corresponds to a specific sequence of coin inputs to test various paths of the FSM, ensuring it behaves correctly for all cases.

```
module vprojnew();
// Declare registers and wires
reg rst,clk;
reg N,D,Q;
wire dis,oN,oD,o2D;
// Instantiate the FSM module
a5leha_3la_allah_new dut (
                          .rst(rst), .clk(clk),
                          .N(N), .Q(Q), .D(D),
                          .dis(dis),
                          .oN(oN), .oD(oD), .o2D(o2D)
                          );

 // clock signal
 initial begin
    clk=0;
    forever begin
        #10
        clk=~clk;
    end
end

```

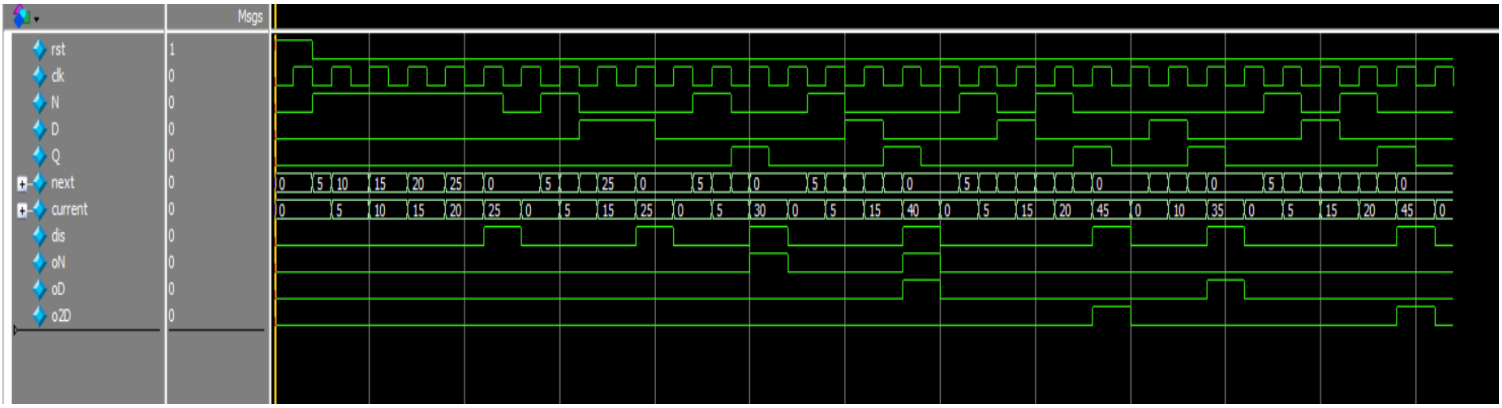
```
// Testbench to apply input cases
initial begin
    // Initialize signals
    rst=1; N=0; D=0; Q=0;
    @(negedge clk);     // Wait for a negative clock edge
    rst=0;      // Release reset signal


    // Test Case 1: Insert 5 Nickels

    repeat(5)begin
        assign {N,D,Q} =3'b100;     //nickel inserted
        @(negedge clk);
    end

    assign {N,D,Q} =3'b000;         // Clear inputs
    @(negedge clk);

    // Test Case 2: Insert 1 Nickel and 2 Dimes

    assign {N,D,Q} =3'b100;         //nickel inserted
    @(negedge clk);
    repeat(2)begin
        assign {N,D,Q} =3'b010;     //dime inserted
        @(negedge clk);
    end

    assign {N,D,Q} =3'b000;         // Clear inputs
    @(negedge clk);


     // Test Case 3: Insert 1 Nickel and 1 Quarter
     assign {N,D,Q} =3'b100;//nickel inserted
     @(negedge clk);
     assign {N,D,Q} =3'b001;//quarter inserted
     @(negedge clk);

     assign {N,D,Q} =3'b000;         // Clear inputs
     @(negedge clk);


     // Test Case 4: Insert 1 Nickel, 1 Dime, and 1 Quarter

     assign {N,D,Q} =3'b100;//nickel inserted
     @(negedge clk);
     assign {N,D,Q} =3'b010;//dime inserted
     @(negedge clk);
     assign {N,D,Q} =3'b001;//quarter inserted
     @(negedge clk);

     assign {N,D,Q} =3'b000;         // Clear inputs
     @(negedge clk);


     // Test Case 5: Insert 1 Nickel, 1 Dime, 1 Nickel, and 1 Quarter

     assign {N,D,Q} =3'b100;//nickel inserted
     @(negedge clk);
     assign {N,D,Q} =3'b010;//dime inserted
     @(negedge clk);
     assign {N,D,Q} =3'b100;//nickel inserted
     @(negedge clk);
     assign {N,D,Q} =3'b001;//quarter inserted
     @(negedge clk);

     assign {N,D,Q} =3'b000;         // Clear inputs
     @(negedge clk);

```

```
91      // Test Case 6: Insert 1 Dime and 1 Quarter
92
93      assign {N,D,Q} =3'b010;//dime inserted
94      @(negedge clk);
95      assign {N,D,Q} =3'b001;//quarter inserted
96      @(negedge clk);
97
98
99      assign {N,D,Q} =3'b000;          // Clear inputs
100     @(negedge clk);
101
102     // Repeat Test Case 5 for additional validation
103
104     assign {N,D,Q} =3'b100;//nickel inserted
105     @(negedge clk);
106     assign {N,D,Q} =3'b010;//dime inserted
107     @(negedge clk);
108     assign {N,D,Q} =3'b100;//nickel inserted
109     @(negedge clk);
110     assign {N,D,Q} =3'b001;//quarter inserted
111     @(negedge clk);
112
113
114     assign {N,D,Q} =3'b000;          // Clear inputs
115     @(negedge clk);
116
117     // End simulation
118     $stop;
119 end
120 endmodule
121
```

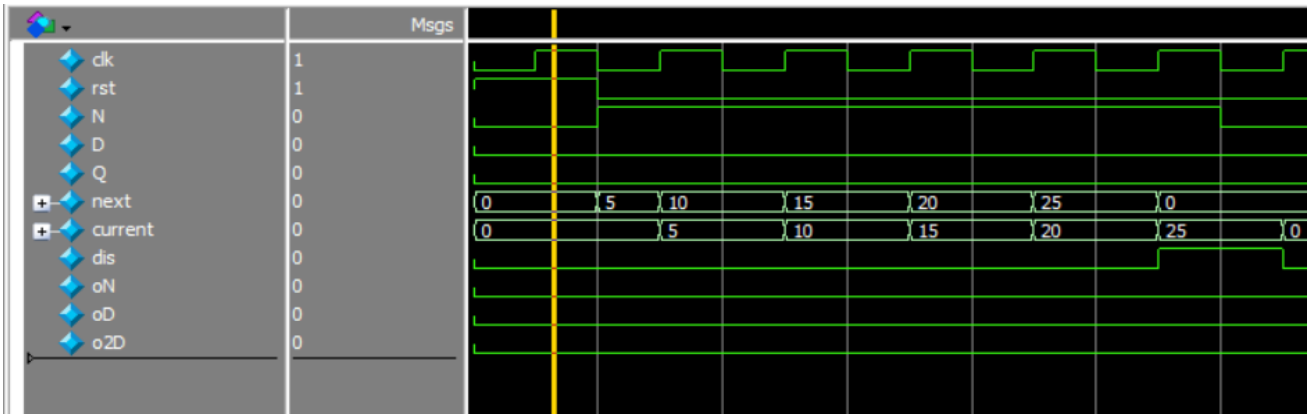
5. Waveform

This section presents the results of the simulation.

A bird view of the waveform

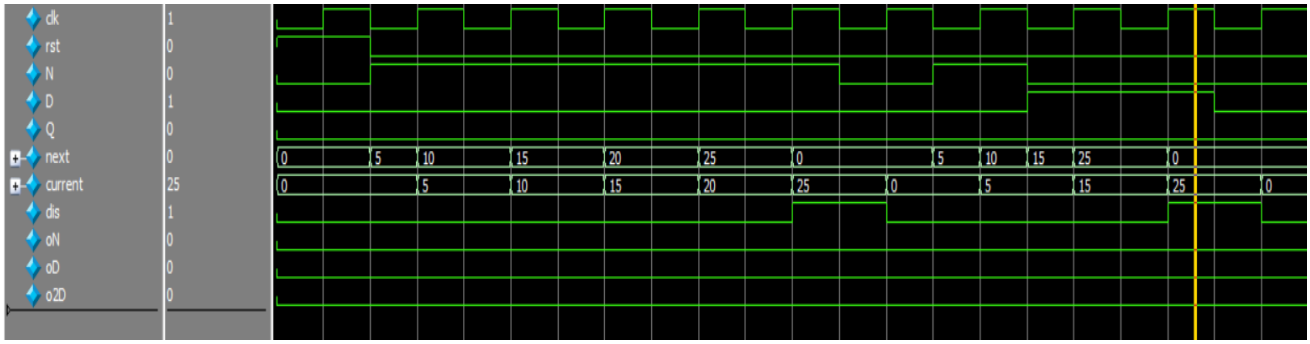


Reset is on (rst=1): current and next state to (s0: initial state)



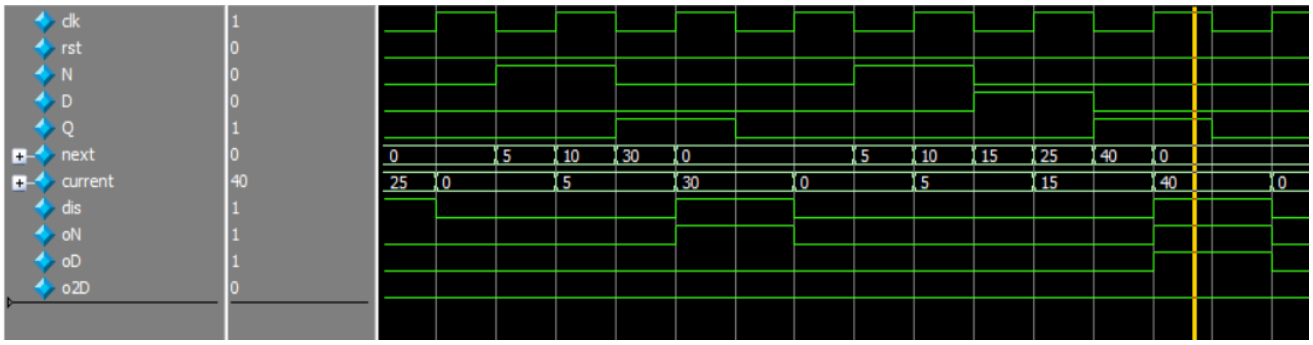
Reset is off(rst=0) , let's cover all cases

1st & 2nd cases: Dis =1, no change



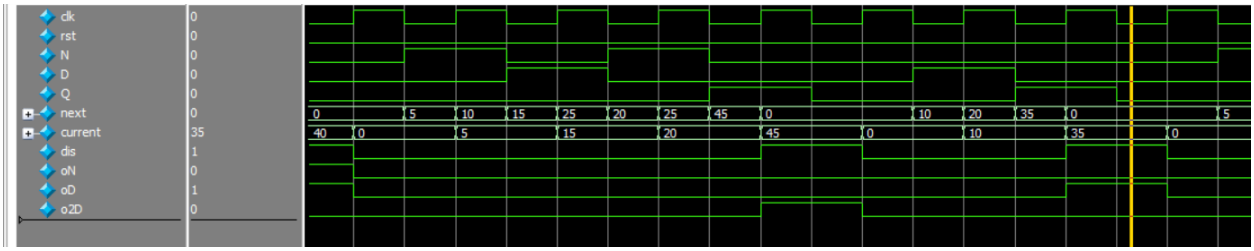
3rd & 4th cases: Dis =1, for (3rd change: one Nickel,

4th change: one Nickel, one Dime)

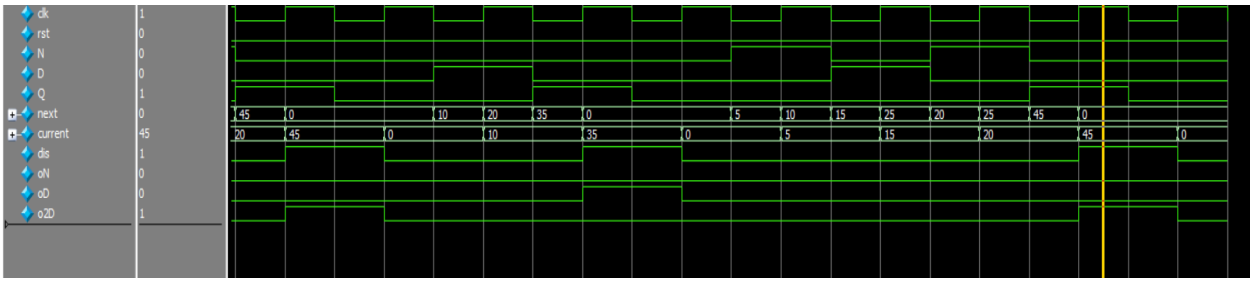


5th & 6th cases: Dis =1, for (5th change: two dimes,

6th change: one Dime)



At the end, we repeated the 5th case for more validation



6. Contribution sheet

Name	Work
Seif	
Ahmed	
Abdulrahman	FSM Design code Report