

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**
- 2. How we think**
- 3. FSM Design Code**
- 4. Testbench Code**
- 5. Waveform Snippets**
- 6. Contribution Sheet**

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.

3. FSM Design Code

[Link for the files project on drive](#)

```
1  module a5leha_3la_allah_new(
2      input N,D,Q,
3      input rst,clk,
4      output reg dis,oN,oD,o2D
5  );
6
7  /*
8      Inputs: N (Nickel), D (Dime), Q (Quarter)
9      Inputs: rst (Reset), clk (Clock)
10     dis (Dispense), oN (Return Nickel),
11     oD (Return Dime), o2D (Return Two Dimes)
12 */
13 /*
14 */
15 reg [5:0] current,next;
16
17 parameter s0=0,    // Initial state: 0 cents
18           s5=5,    // State: 5 cents
19           s10=10,   // State: 10 cents
20           s15=15,s20=20,s25=25,s30=30,s35=35,s40=40,s45=45;
21
22 //Sequential block: Current state transitions
23
24 always@(posedge clk ,posedge rst)begin
25     if(rst)begin
26         current<=s0;    // Asynchronous reset: Go to initial state
27     end
28     else begin
29         current<=next; // Update current state on positive edge of clock
30     end
31 end
32
33 //combinational block: next state logic
34 // Coin inputs: N (Nickel), D (Dime), Q (Quarter)
35 //3'b100 when nickel inserted , 3'b010 dime inserted , 3'b001 quarter inserted ,default = no coin inserted
36
37 always@(*)begin
38
39     case(current)
40         // Handle coin inputs in state s0
41         s0:begin
42             case ({N,D,Q})
43                 3'b100:next=s5;           // Nickel inserted
44                 3'b010:next=s10;          // Dime inserted
45                 3'b001:next=s25;          // Quarter inserted
46                 default:next=s0;          // No coin inserted
47             endcase
48         end
49
50         s5:begin
51             case ({N,D,Q})
52                 3'b100:next=s10;
53                 3'b010:next=s15;
54                 3'b001:next=s30;
55                 default: next=s5; // No coin inserted
56             endcase
57         end
58
59         s10:begin
60             case ({N,D,Q})
61                 3'b100:next=s15;
62                 3'b010:next=s20;
63                 3'b001:next=s35;
64                 default: next=s10; // No coin inserted
65             endcase
66         end
67     end
68 end
```

```

68     s15:begin
69         case ({N,D,Q})
70             3'b100:next=s20;
71             3'b010:next=s25;
72             3'b001:next=s40;
73             default: next=s15; // No coin inserted
74         endcase
75     end
76
77     s20:begin
78         case ({N,D,Q})
79             3'b100:next=s25;
80             3'b010:next=s30;
81             3'b001:next=s45;
82             default: next=s20; // No coin inserted
83         endcase
84     end
85
86     s25:begin
87         next=s0; // Reset to initial state
88     end
89     s30:begin
90         next=s0; // Reset to initial state
91     end
92     s35:begin
93         next=s0; // Reset to initial state
94     end
95     s40:begin
96         next=s0; // Reset to initial state
97     end
98     s45:begin
99         next=s0; // Reset to initial state
100    end
101    default: next =current;// Default case to avoid latches
102 endcase
103 end
104
105
106
107 // Combinational block: Output logic
108 // Outputs depend on the current state
109
110 always@(*)begin
111     // Default output values
112     dis=0; oN=0; oD=0; o2D=0;
113
114     case(current)
115
116         s25:begin
117             dis=1; // Dispense soda
118         end
119
120         s30:begin
121             dis=1;oN=1; // Dispense soda + Return 1 Nickel
122         end
123
124         s35:begin
125             dis=1; oD=1; // Dispense soda + Return 1 Dime
126         end
127
128         s40:begin
129             dis=1;oN=1;oD=1; // Dispense soda + Return 1 Nickel + 1 Dime
130         end
131
132         s45:begin
133             dis=1;o2D=1; // Dispense soda + Return 2 Dimes
134         end
135
136     endcase
137 end
138 endmodule
139

```

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.

```
1  module vprojnew();
2  // Declare registers and wires
3  reg rst,clk;
4  reg N,D,Q;
5  wire dis,oN,oD,o2D;
6  // Instantiate the FSM module
7  a5leha_3la_allah_new dut (
8      .rst(rst), .clk(clk),
9      .N(N), .Q(Q), .D(D),
10     .dis(dis),
11     .oN(oN), .oD(oD), .o2D(o2D)
12 );
13
14 // clock signal
15 initial begin
16     clk=0;
17     forever begin
18         #10
19         clk=~clk;
20     end
21 end
22
```

```

23 // Testbench to apply input cases
24 initial begin
25     // Initialize signals
26     rst=1; N=0; D=0; Q=0;
27     @(negedge clk); // Wait for a negative clock edge
28     rst=0; // Release reset signal
29
30
31 // Test Case 1: Insert 5 Nickels
32
33 repeat(5)begin
34     assign {N,D,Q} =3'b100; //nickel inserted
35     @(negedge clk);
36 end
37
38 assign {N,D,Q} =3'b000; // Clear inputs
39 @(negedge clk);
40
41 // Test Case 2: Insert 1 Nickel and 2 Dimes
42
43 assign {N,D,Q} =3'b100; //nickel inserted
44 @(negedge clk);
45 repeat(2)begin
46     assign {N,D,Q} =3'b010; //dime inserted
47     @(negedge clk);
48 end
49
50 assign {N,D,Q} =3'b000; // Clear inputs
51 @(negedge clk);
52
53
54 // Test Case 3: Insert 1 Nickel and 1 Quarter
55 assign {N,D,Q} =3'b100;//nickel inserted
56 @(negedge clk);
57 assign {N,D,Q} =3'b001;//quarter inserted
58 @(negedge clk);
59
60 assign {N,D,Q} =3'b000; // Clear inputs
61 @(negedge clk);
62
63
64 // Test Case 4: Insert 1 Nickel, 1 Dime, and 1 Quarter
65
66 assign {N,D,Q} =3'b100;//nickel inserted
67 @(negedge clk);
68 assign {N,D,Q} =3'b010;//dime inserted
69 @(negedge clk);
70 assign {N,D,Q} =3'b001;//quarter inserted
71 @(negedge clk);
72
73 assign {N,D,Q} =3'b000; // Clear inputs
74 @(negedge clk);
75
76
77 // Test Case 5: Insert 1 Nickel, 1 Dime, 1 Nickel, and 1 Quarter
78
79 assign {N,D,Q} =3'b100;//nickel inserted
80 @(negedge clk);
81 assign {N,D,Q} =3'b010;//dime inserted
82 @(negedge clk);
83 assign {N,D,Q} =3'b100;//nickel inserted
84 @(negedge clk);
85 assign {N,D,Q} =3'b001;//quarter inserted
86 @(negedge clk);
87
88 assign {N,D,Q} =3'b000; // Clear inputs
89 @(negedge clk);
90

```



```

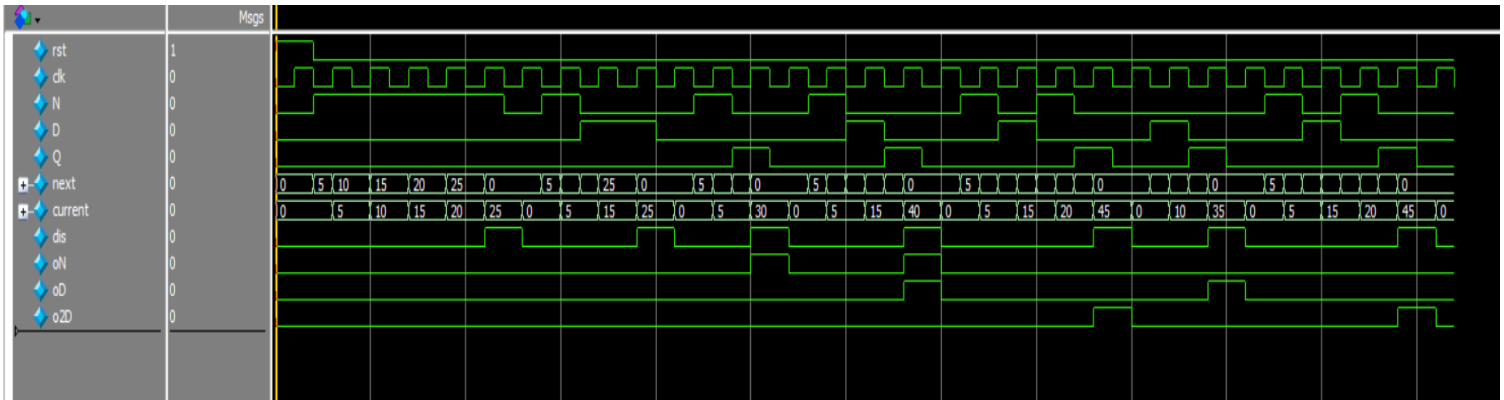
91 // Test Case 6: Insert 1 Dime and 1 Quarter
92
93 assign {N,D,Q} =3'b010; //dime inserted
94 @(negedge clk);
95 assign {N,D,Q} =3'b001; //quarter inserted
96 @(negedge clk);
97
98
99 assign {N,D,Q} =3'b000; // Clear inputs
100 @(negedge clk);
101
102 // Repeat Test Case 5 for additional validation
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

```

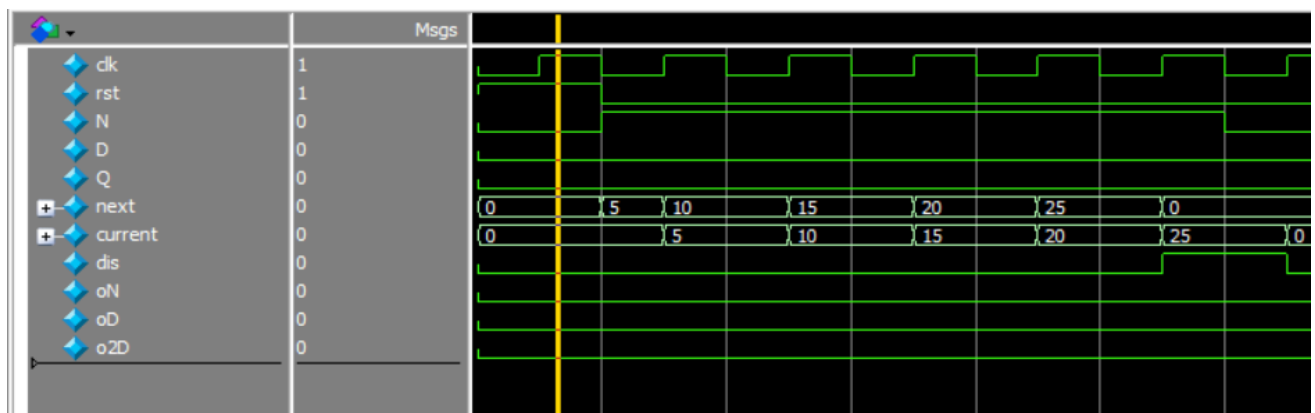
5. Waveform

This section presents the results of the simulation.

A bird view of the waveform

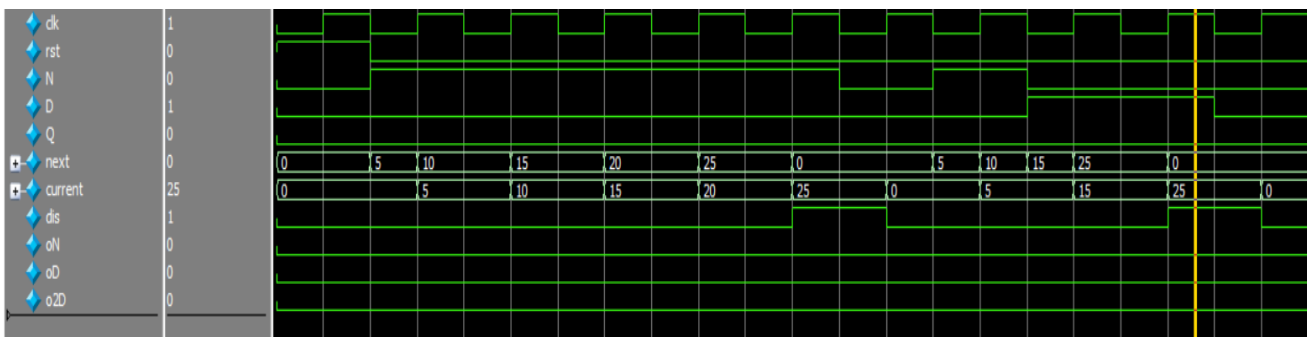


Reset is on (rst=1): current and next state to (s0: initial state)

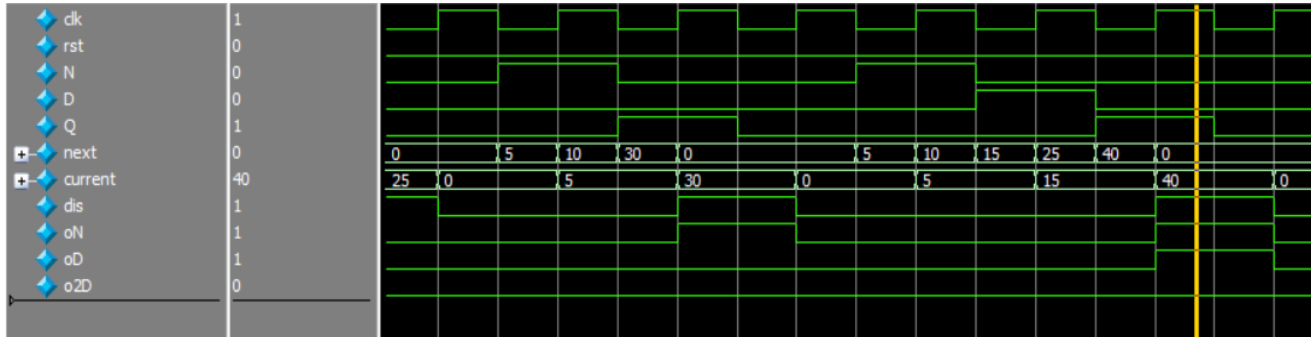


Reset is off (rst=0), let's cover all cases

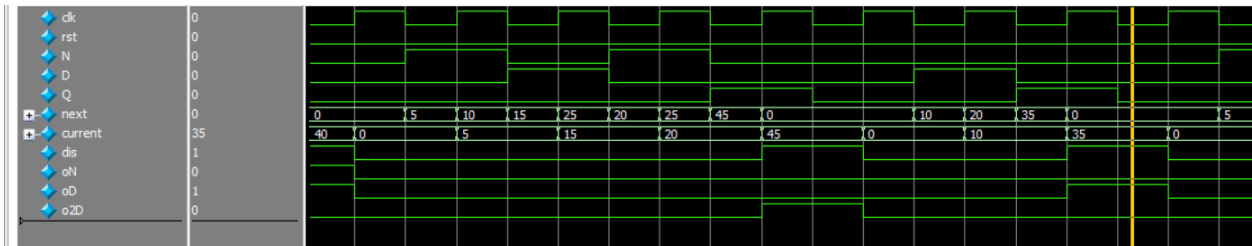
1st & 2nd cases: Dis =1, no change



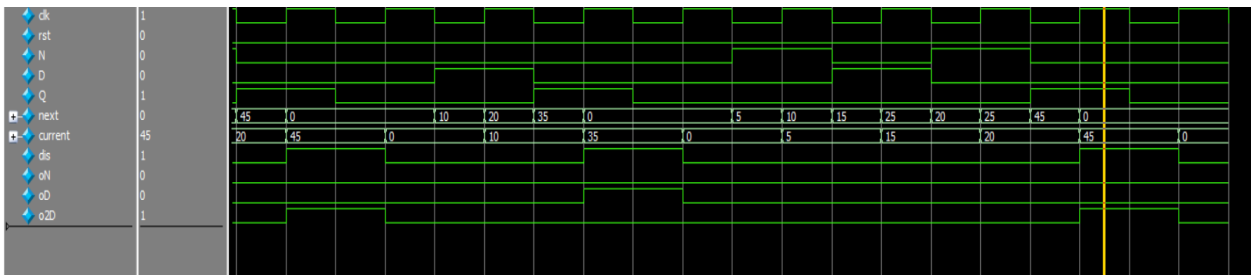
**3rd & 4th cases: Dis =1, for (3rd change: one Nickel,
4th change: one Nickel, one Dime)**



**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