LAB 09

OBJECTIVE

Develop, implement and Simulate Moore Machine using behavioral modeling

Lab Task 1

Implement and simulate Vending Moore Machine

Verilog Module Code:

```
module VendingMooreMachine(open,Clk,Reset,N,D);
    output open;
    input Clk;
    input Reset;
    input N;
    input D;

    reg open;
    reg [1:0]state;
    reg [1:0]next_state;

    parameter  [1:0]zero = 2'b00;
    parameter [1:0]five =2'b01;
    parameter [1:0]ten =2'b10;
    parameter [1:0]fifteen =2'b11;

    always@(N or D or state) // Next State determination
    begin
    case(state)

            zero:begin
               if(N) begin
                        next_state=five;
                     end
               else if(D) begin
                        next_state=ten;
                     end
               else begin
                        next_state=zero;
                     end
             end

           five:begin
               if(N) begin
                        next_state=ten;
                     end
               else if(D) begin
                        next_state=fifteen;
                     end
               else begin
                        next_state=five;
                     end
             end

```

```
65              ten:begin
66                 if(N) begin
67                          next_state=fifteen;
68                        end
69                 else if(D) begin
70                          next_state=fifteen;
71                        end
72                 else begin
73                          next_state=ten;
74                        end
75                  end
76
77              fifteen:begin
78
79                 if(N) begin
80                          next_state=five;
81                        end
82                 else if(D) begin
83                          next_state=ten;
84                        end
85                 else begin
86                          next_state=0;
87                        end
88                  end
89
90    endcase
91    end
94    always @(posedge Clk)  // State Registers
95    begin
96
97       if(Reset )
98       begin
99          state <= zero;
100      end
101
102      else begin
103         state<=next_state;
104      end
105   end
106
107 always@(state) // Output determination
108    begin
109    case(state)
110
111             zero:begin
112                   open<=0;
113                end
114
115             five:begin
116                   open<=0;
117                end
118
119             ten:begin
120                   open<=0;
121                end
```

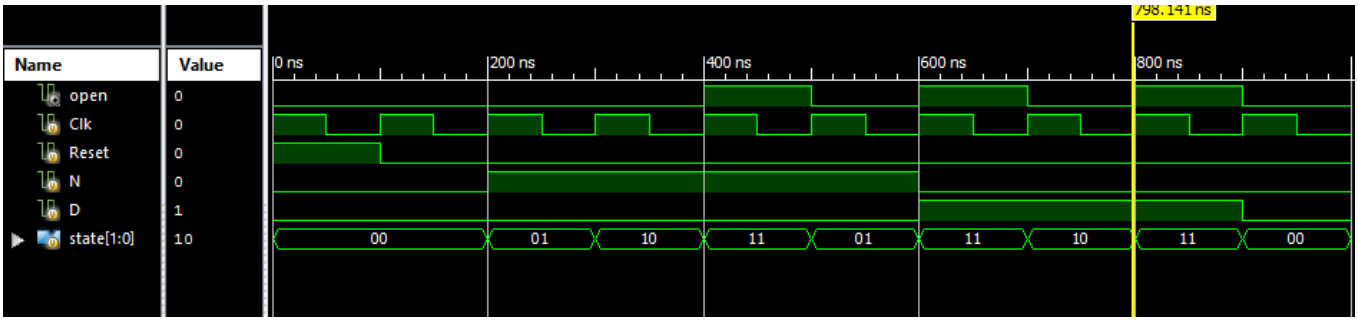
```
123             fifteen:begin
124                   open<=1;
125                 end
126
127     endcase
128     end
129 endmodule
```

Verilog Test Fixture Code:

```
24  module VMMoTF;
25
26     // Inputs
27     reg Clk;
28     reg Reset;
29     reg N;
30     reg D;
31
32     // Outputs
33     wire open;
34
35     // Instantiate the Unit Under Test (UUT)
36     VendingMooreMachine uut (
37        .open(open),
38        .Clk(Clk),
39        .Reset(Reset),
40        .N(N),
41        .D(D)
42     );
43
44     parameter PERIOD=100;
45
46     always
47     begin
48
49        Clk=1;
50        #(PERIOD/2);
```

```
52          Clk=0;
53          #(PERIOD/2);
54      end
55
56
57      initial begin
58          // Initialize Inputs
59
60          Reset = 1;
61          N = 0;
62          D = 0;
63
64          // Wait 100 ns for global reset to finish
65
66
67          // Add stimulus here
68
69      // Case: 3 Nikles are inserted
70      #100; N=0;D=0;Reset=0;
71
72      #100; N=1;D=0;
73      #100; N=1;D=0;
74      #100; N=1;D=0;
75
76  //#100;  N=0;D=0;Reset=0;
77
78  // Case: 1 Nikel 1 Dime are inserted
79      //#100;  N=0;D=0;
80      #100; N=1;D=0;
81      #100; N=0;D=1;
82
83      //#100;  N=0;D=0;Reset=0;
84      // Case: 2 Dimes are inserted
85      //#100;  N=0;D=0;
86
87      #100; N=0;D=1;
88      #100; N=0;D=1;
89          #100; N=0;D=0;
90
91
92      end
93
94
95  endmodule
```

WAVEFORM:



Lab Task 2

To develop a sequence detector
Sequence: 1101

Verilog Module Code:

```
21  module SequenceMooreFSM(Clk,Reset,squence_in,detector_out);
22      input Clk;
23      input Reset;
24      input squence_in;
25      output detector_out;
26
27
28      reg detector_out;
29      reg [2:0]state;
30      reg [2:0]next_state;
31
32
33      parameter [2:0]S0  = 3'b000;
34      parameter [2:0]S1  = 3'b001;
35      parameter [2:0]S2  = 3'b010;
36      parameter [2:0]S3  = 3'b011;
37      parameter [2:0]S4  = 3'b100;
38
39
40      always@(squence_in or state)  // Next State determination
41      begin
42      case(state)
43
```

```
44              S0:begin
45                  if(squence_in) begin
46                           next_state=S1;
47                         end
48
49                  else begin
50                           next_state=S0;
51                         end
52                   end
53
54               S1:begin
55                  if(squence_in) begin
56                           next_state=S2;
57                         end
58
59                  else begin
60                           next_state=S0;
61                         end
62                   end
63
64               S2:begin
65                  if(squence_in) begin
66                           next_state=S2;
67                         end
68
69                  else begin
70                           next_state=S3;
71                         end
72                    end
73
74               S3:begin
75
76                   if(squence_in) begin
77                            next_state=S4;
78                          end
79
80                   else begin
81                            next_state=S0;
82                          end
83                    end
84
85               S4:begin
86
87                   if(squence_in) begin
88                            next_state=S1;
89                          end
90
91                   else begin
92                            next_state=S0;
93                          end
94                    end
95
96      endcase
97      end
```

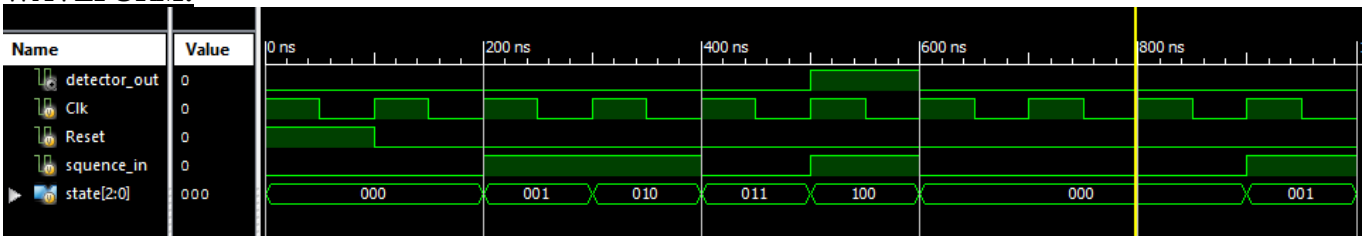
```
100     always @(posedge Clk)   // State Registers
101     begin
102
103         if(Reset)
104         begin
105             state <= S0;
106         end
107
108         else begin
109             state<=next_state;
110         end
111     end
112
113   always@(state)   // output determination
114      begin
115      case(state)
116
117                 S0:begin
118                       detector_out<=0;
119                    end
120
121                 S1:begin
122                       detector_out<=0;
123                    end
124
125                 S2:begin
126                       detector_out<=0;
127                    end
128
129                 S3:begin
130                       detector_out<=0;
131                    end
132
133                 S4:begin
134                       detector_out<=1;
135                    end
136
137     endcase
138     end
139
140
141   endmodule
```

Verilog Test Fixture Code:

```
24    module SMFSMTF;
25
26        // Inputs
27        reg Clk;
28        reg Reset;
29        reg squence_in;
30
31        // Outputs
32        wire detector_out;
33
34        // Instantiate the Unit Under Test (UUT)
35        SequenceMoorseFSM uut (
36            .Clk(Clk),
37            .Reset(Reset),
38            .squence_in(squence_in),
39            .detector_out(detector_out)
40        );
41
42
43    parameter PERIOD=100;
44        always
45        begin
46            Clk=1;
47            #(PERIOD/2);
48
49
```

```
50          Clk=0;
51          #(PERIOD/2);
52      end
53      initial begin
54          // Initialize Inputs
55
56          Reset = 1;
57          squence_in = 0;
58          // Wait 100 ns for global reset to finish
59          // Add stimulus here
60          #100; Reset=0;
61          #100; squence_in = 1;
62          #100; squence_in = 1;
63          #100; squence_in = 0;
64          #100; squence_in = 1;
65          #100; squence_in = 0;
66          #100; squence_in = 0;
67          #100; squence_in = 0;
68          #100; squence_in = 1;
69          #100; squence_in = 0;
70          #100; squence_in = 1;
71          #100; squence_in = 0;
72          #100; squence_in = 0;
73          #100; squence_in = 1;
74      end
75  endmodule
```

WAVEFORM:



CONCLUSION:

In this lab, we learned how to develop Vending Machine and also develop sequence detector using Moore machine. Using the ISE Design Suite, We also produced a wave output in which we can see the characteristics of the output after performing these task.

We also learned about the Moore machine concept in this lab and carried out activities involving this state machine. We do two tasks. The first is a vending machine that was designed with a sequence detector and was able to work as planned by accepting the utilising the proper input sequence and releasing the appropriate product. And the second one, a sequence detector, proved how well Moore's machine worked at detecting particular sequences by precisely identifying the input sequence and producing the right output signal. In the end, this idea proved the adaptability and dependability of Moore's machine in the creation of useful systems.