## **Lab 06**

Design a sequence detector implementing a Mealy state machine using three always blocks. The Mealy state

```
// malak Majeedullah khan
23
        module Sequencer_detector_FSM(clk, reset, In, Out);
25
        input clk, reset, In;
26
        output reg Out;
27
28
        // state encoding
    o reg [2:0] s0 = 3'b000;
29
30
     o reg [2:0] sl = 3'b001;
31
     0
       reg [2:0] s2 = 3'b010;
     o reg [2:0] s3 = 3'b011;
32
33
     reg [2:0] s4 = 3'bl00;
     o reg [2:0] s5 = 3'b101;
34
35
36
        // internal registers
37
        reg [2:0] state_reg, state_next;
38
39
         // current state logic

    always@(posedge clk, posedge reset)

40 €
41 🖯 🔾
       if (reset)
42
     0
            state_reg <= 1'b0;
43
        else
44 🖒 🔾
            state_reg <= state_next;
45
        // next_state logic
46
```

```
45
46
        // next state logic
    O always@(*)
47 €
48 🖯 🔾 case(state_reg)
49
     3'b000 : if (In) state_next = s1; else state_next = s0;
        3'b001 : if (In) state_next = s2; else state_next = s0;
50
51
        3'b010 : if (~In) state_next = s3; else state_next = s2;
        3'b011 : if (In) state_next = s4; else state_next = s0;
52
        3'bl00 : if (In) state_next = s5; else state_next = s0;
54
        3'bl01 : if (In) state_next = s1; else state_next = s0;
55
    default : state_next = state_reg;
56 A
        endcase
57
58
        // output logic
59 E
    O always@(*)
60 🗄
    Case (state_reg)
    3'b000 : Out = 1'b0;
61
62
     3'b001 : Out = 1'b0;
63
    0
        3'b010 : Out = 1'b0;
64
        3'b011 : Out = 1'b0;
        3'b100 : Out = 1'b0;
65
    0
66
    0
       3'b101 : Out = 1'b1;
    O default : Out = 1'b0;
67
68 🖨
        endcase
69
```

## Test bench

```
module Sequencer_detector_TB();
   reg clk, reset, In;
   wire Out;
   // Instatiate
   Sequencer_detector_FSM ins0(clk, reset, In, Out);
   // clock gen
   initial
O clk = 0;
   always
#10 clk = ~ clk;
   initial
   begin
       // reset
0
      reset = 0;
0
       @(negedge clk) reset = 1;
0
       @(negedge clk) reset = 0;
       // sequence application
```

```
w sequencer_detector_rsn.v · x a onuned / x
F:/8th semster/DSD/lab 01/lab_06/lab_06.srcs/sources_1/new/Sequencer_detector_FSM.v
         0
                 reset = 0;
93
         0
                 @(negedge clk) reset = 1;
CIL
    94
         0
                 @(negedge clk) reset = 0;
00
    95
                 // sequence application
    96
97
         0
                @(negedge clk) In = 1;
0
    98
                 @(negedge clk) In = 1;
                @(negedge clk) In = 0;
    99
         0
×
         0
   100
                @(negedge clk) In = 1;
//
                @(negedge clk) In = 1;
   101
         0
102
   103
                 // sequence application
æ
         0
                @(negedge clk) In = 1;
   104
<del>_</del>
   105
         0
                 @(negedge clk) In = 1;
TO THE
   106
         0
                 @(negedge clk) In = 0;
                 @(negedge clk) In = 0;
   107
          0
   108
                 @(negedge clk) In = 1;
   109
   110
                  // stop simulation
         0
   111
                 @(negedge clk) $stop;
   112
   113 🖨
             end
   114
   115
   116 🚊
             endmodule
```

