```
1   //SW[0] reset when 0
2   //SW[1] input signal
3
4   //KEY[0] clock signal
5
6   //LEDR[3:0] displays current state
7   //LEDR[9] displays output
8   `timescale 1ns / 1ns // `timescale time_unit/time_precision
9
10  module sequenceDetector(SW, KEY, LEDR);
11      input [9:0] SW;
12      input [3:0] KEY;
13      output [9:0] LEDR;
14
15      wire w, clock, resetn, out_light;
16
17      reg [3:0] y_Q, Y_D; // y_Q represents current state, Y_D represents next state
18
19      localparam A = 4'b0000, B = 4'b0001, C = 4'b0010, D = 4'b0011, E = 4'b0100, F =
        4'b0101, G = 4'b0110;
20
21      assign w = SW[1];
22      assign clock = ~KEY[0];
23      assign resetn = SW[0];
24
25      //State table
26      //The state table should only contain the logic for state transitions
27      //Do not mix in any output logic. The output logic should be handled separately.
28      //This will make it easier to read, modify and debug the code.
29      always@(*)
30      begin: state_table
31          case (y_Q)
32              A: begin
33                     if (!w) Y_D = A;
34                     else Y_D = B;
35                 end
36              B: begin
37                     if(!w) Y_D = A;
38                     else Y_D = C;
39                 end
40              C: begin
41                     if(!w) Y_D = E;
42                     else Y_D = D;
43                 end
44              D: begin
45                     if(!w) Y_D = E;
46                     else Y_D = F;
47                 end
48              E: begin
49                     if(!w) Y_D = A;
50                     else Y_D = G;
51                 end
52              F: begin
53                     if(!w) Y_D = E;
54                     else Y_D = F;
55                 end
56              G: begin
57                     if(!w) Y_D = A;
58                     else Y_D = C;
59                 end
60              default: Y_D = A;
61          endcase
62      end // state_table
63
64      // State Registers
65      always @(posedge clock)
66      begin: state_FFs
67          if(resetn == 1'b0)
68              y_Q <=  A; // Should set reset state to state A
```

```
69          else
70              y_Q <= Y_D;
71      end // state_FFS
72
73      // Output logic
74      // Set out_light to 1 to turn on LED when in relevant states
75      assign out_light = ((y_Q == F) | (y_Q == G));
76
77      assign LEDR[9] = out_light;
78      assign LEDR[3:0] = y_Q;
79  endmodule
80
81
```