```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //  Class: CECS 361                                                             //
4   //  Project: Project1-Cecs361                                                   //
5   //                                                                              //
6   //  File name: <ledController.v>                                                //
7   //  Abstract: This module has an input clk because at the posedge               //
8   //            of clk, our moore FSM increment to next state                     //
9   //            regardless of the value. The output will be in                    //
10  //            relation to the analog pins {a7-a0} and the                       //
11  //            present state {seqSel}. Pixels are on from right                  //
12  //            to left of the display.                                           //
13  //  Created by       <Alina Suon> on <09-18-18>.                                //
14  //  Copyright © 2018 <Alina Suon>. All rights reserved.                         //
15  //
    //                                                                                  //
16  //  In submitting this file for class work at CSULB                             //
17  //  I am confirming that this is my work and the work                           //
18  //  of no one else. In submitting this code I acknowledge that                  //
19  //  plagiarism in student project work is subject to dismissal.                 //
20  //  from the class                                                              //
21  //////////////////////////////////////////////////////////////////////////////////
22  module ledController(clk, rst, a, seqSel);
23     input                clk, rst;
24     output  reg    [7:0] a;
25     output  reg    [2:0] seqSel;
26
27  //State register and next state
28     reg            [2:0] q;     //present state
29     reg            [2:0] d;     //next state
30
31  //Next state combinational logic
32     always @(q) begin
33        case(q)
34           3'b000:     d = 3'b001;
35           3'b001:     d = 3'b010;
36           3'b010:     d = 3'b011;
37           3'b011:     d = 3'b100;
38           3'b100:     d = 3'b101;
39           3'b101:     d = 3'b110;
40           3'b110:     d = 3'b111;
41           3'b111:     d = 3'b000;
42           default     d = 3'b000;
43        endcase
44     end
45
46  //State register sequential logic
47     always @ (posedge clk, posedge rst)
48        begin
49           if(rst == 1'b1)
50              q <= 3'b000;
51           else
52              q <= d;
53        end
54
55  //Output Combinational Logic
56     always @(q)
```

```
57          begin
58             case(q)
59                3'b000:     {a, seqSel} = 11'b01111111_000;
60                3'b001:     {a, seqSel} = 11'b10111111_001;
61                3'b010:     {a, seqSel} = 11'b11011111_010;
62                3'b011:     {a, seqSel} = 11'b11101111_011;
63                3'b100:     {a, seqSel} = 11'b11110111_100;
64                3'b101:     {a, seqSel} = 11'b11111011_101;
65                3'b110:     {a, seqSel} = 11'b11111101_110;
66                3'b111:     {a, seqSel} = 11'b11111110_111;
67                default:    {a, seqSel} = 11'b11111111_000;
68             endcase
69       end
70    endmodule
71
```