```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //                                                                              //
4   //  Class: CECS 360                                                             //
5   //  Project name: Project1_CECS360                                              //
6   //  File name: led_controller.v                                                   //
7   //                                                                              //
8   //  Created by Umar Khan 09/19/2017       .                                     //
9   //  Abstract:                                                                   //
10  //        This shift module uses clk as an input. At every posedge clk, the     //
11  //        Moore state machine will increment states regardless of the clk       //
12  //        value (which will always be 1). Every state will output the           //
13  //        corresponding analog pins {a7,a6,a5,a4,a3,a2,a1,a0} as well as         //
14  //        the present state {seg_sel}. For example the present state 000        //
15  //        will turn on the rightmost segment display, as well as selecting      //
16  //        the correct value to display. This moore state machine will           //
17  //        activate one pixel at a time from the rightmost display to the        //
18  //        leftmost display and then return to the rightmost display. This       //
19  //        will cycle indefinately.                                              //
20  //                                                                              //
21  //////////////////////////////////////////////////////////////////////////////////
22
23
24
25  module led_controller(clk, reset, a, seq_sel);
26     input                clk, reset;
27     output  reg    [7:0] a;
28     output  reg    [2:0] seq_sel;
29
30
31  ////////////////////////////////////////////////////////////////////////////////
32  //                               state register and
33  //                              next_state variables
34  ////////////////////////////////////////////////////////////////////////////////
35     reg            [2:0] Q;     //present state
36     reg            [2:0] D;     //next state
37
38  ////////////////////////////////////////////////////////////////////////////////
39  //                          Next State Combinational Logic
40  //   (next state values can change anytime but will only be "clocked" below)
41  ////////////////////////////////////////////////////////////////////////////////
42     always @(Q) begin
43        case(Q)
44           3'b000:     D = 3'b001;
45           3'b001:     D = 3'b010;
46           3'b010:     D = 3'b011;
47           3'b011:     D = 3'b100;
48           3'b100:     D = 3'b101;
49           3'b101:     D = 3'b110;
50           3'b110:     D = 3'b111;
51           3'b111:     D = 3'b000;
52           default     D = 3'b000;
53        endcase
54     end
55
56  ////////////////////////////////////////////////////////////////////////////////
57  //                     State Register Logic (Sequential Logic)
```

```
58  ////////////////////////////////////////////////////////////////////////////////
59     always @ (posedge clk, posedge reset) begin
60        if(reset == 1'b1)
61           Q <= 3'b000;
62        else
63           Q <= D;
64     end
65
66  ////////////////////////////////////////////////////////////////////////////////
67  //                            Output Combinational Logic
68  //             (outputs will only change when present state changes)
69  ////////////////////////////////////////////////////////////////////////////////
70     always @(Q) begin
71        case(Q)
72           3'b000:     {a, seq_sel} = 11'b01111111_000;
73           3'b001:     {a, seq_sel} = 11'b10111111_001;
74           3'b010:     {a, seq_sel} = 11'b11011111_010;
75           3'b011:     {a, seq_sel} = 11'b11101111_011;
76           3'b100:     {a, seq_sel} = 11'b11110111_100;
77           3'b101:     {a, seq_sel} = 11'b11111011_101;
78           3'b110:     {a, seq_sel} = 11'b11111101_110;
79           3'b111:     {a, seq_sel} = 11'b11111110_111;
80           default:    {a, seq_sel} = 11'b11111111_000;
81        endcase
82     end
83
84
85  endmodule
86
```