```
1   `timescale 1ns / 1ps
2   /**********************************************************************
3    * File Name: led_ontroller.v
4    * Project: Counter using AISO
5    * Designer: Marc Cabote
6    * Email: marcdominic011@gmail.com
7    * Rev. Date: 20 September 2017
8    *
9    * Purpose: The LED Controller module is a sequential logic circuit designed to
10   *          cycle between eight "states" using a Moore FSM implementation. In
11   *          this project, each "state" represents one of the eight anodes on the
12   *          Nexys4 DDR 7-segment display. Only one of the eight anodes are on
13   *          at any given time, however the clock frequency from the LED Clock
14   *          module gives the "illusion" that all eight anodes are on at the same
15   *          time due to the rate in which the Moore FSM cycles between "states".
16   *          The output of the Moore FSM also determines what data is displayed
17   *          on each anode by sending a 3-bit "segment select" output to the
18   *          8-to-1 Multiplexer module.
19   *
20   * Notes:   -  This module has an asynchronous reset input.
21   *          -  This Moore FSM implemtation contains no inputs.
22   *          -  This Module is from Computer Logic Design II
23   *          -  Slight modifications for the clock and reset
24   **********************************************************************/
25  module led_controller(input            clk, rst,
26                        output reg       a7, a6, a5, a4, a3, a2, a1, a0,
27                        output reg [2:0] seg_sel
28                        );
29
30     reg [2:0] ps,
31               ns;
32
33     // Next State Combinational Logic
34     always @( ps ) begin
35        case ( ps )
36           3'b000 : ns = 3'b001; // S1
37           3'b001 : ns = 3'b010; // S2
38           3'b010 : ns = 3'b011; // S3
39           3'b011 : ns = 3'b100; // S4
40           3'b100 : ns = 3'b101; // S5
41           3'b101 : ns = 3'b110; // S6
42           3'b110 : ns = 3'b111; // S7
43           3'b111 : ns = 3'b000; // Start
44           default: ns = 3'b000; // Start
45        endcase
46     end
47
48     // State Register
49     always @( posedge clk, posedge rst ) begin
50        if (rst) begin
51           ps = 3'b000;
52        end
53        else begin
54           ps = ns;
55        end
56     end
57
```

```
58      // Output Combinational Logic
59      always @( ps ) begin
60         case ( ps )
61            3'b000 : { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b000_01111111;
62            3'b001 : { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b001_10111111;
63            3'b010 : { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b010_11011111;
64            3'b011 : { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b011_11101111;
65            3'b100 : { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b100_11110111;
66            3'b101 : { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b101_11111011;
67            3'b110 : { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b110_11111101;
68            3'b111 : { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b111_11111110;
69            default: { seg_sel, a7, a6, a5, a4, a3, a2, a1, a0 } = 11'b000_11111111;
70         endcase
71      end
72
73   endmodule
74
```