```
/* Name: Eugene Ngo
          Date: 3/7/2023
 3
          Class: EE 371
          Lab 6: Parking Lot 3D Simulation
 5
6
      // DE1_SoC combines all the modules together, pipelining the control and // various incrememnt modules to Switches and VGPIO to then
      // send them to the datapath module to output to the HEX and RAM modules.
 9
      timescale 1 ps / 1 ps module DE1_SoC (CLOCK_50, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, SW, LEDR, V_GPIO);
10
11
          // define ports
12
          input logic CLOCK_50;
13
          output logic [6:0] HEXO, HEX1, HEX2, HEX3, HEX4, HEX5;
          input logic [3:0] KEY; input logic [9:0] SW;
14
15
16
          output logic [9:0] LEDR; inout logic [35:23] V_GPIO;
17
18
19
          logic clk;
20
21
          assign c1k = CLOCK_50;
          logic [1:0] counter_out;
logic [3:0] hour_out, incr_out, addr_out;
22
23
24
          logic [31:0] divided_clocks;
25
26
          // FPGA input
27
          assign V_{GPIO[26]} = V_{GPIO[28]};
                                                      // LED parking 1
                                                      // LED parking 2
28
          assign V_{GPIO[27]} = V_{GPIO[29]};
29
                                                     // LED parking 3
          assign V_{GPIO[32]} = V_{GPIO[30]};
30
31
          // Parking spot 1 occupied and Parking spot 2 occupied and Parking spot 3 occupied
32
          // = parking lot = full.
33
          assign V_{GPIO[34]} = V_{GPIO[28]} & V_{GPIO[29]} & V_{GPIO[30]};
                                                                                          // LED full
34
35
          // Parking lot isnt full and presence at entrance = open entrance gate assign V_GPIO[31] = \sim V_GPIO[34] \& V_GPIO[23]; // Open entrance
          // If anyone at exit sensor, open exit gate.
assign V_GPIO[33] = V_GPIO[24]; // Open exit
37
38
39
40
           // Helper logic, passed along to different modules.
41
          logic zeroOccupied;
42
      // logic [6:0] rushHourBegin, rushHourEnd;
43
          assign zeroOccupied = !(V_GPIO[28] | V_GPIO[29] | V_GPIO[30]);
44
45
          logic startRush;
46
          logic rushEnded;
47
          logic stopRush;
48
          logic endGameHexOut;
49
          // FPGA output, for debugging
assign LEDR[0] = V_GPIO[28]; // Presence parking
assign LEDR[1] = V_GPIO[29]; // Presence parking
assign LEDR[2] = V_GPIO[30]; // Presence parking
assign LEDR[3] = V_GPIO[23]; // Presence entrance
assign LEDR[4] = V_GPIO[24]; // Presence exit
assign LEDR[9] = endGameHexOut; // Presence exit
assign LEDR[8] = starRush;
assign LEDR[7] = starRush;
50
51
52
                                                 // Presence parking 1
// Presence parking 2
// Presence parking 3
// Presence entrance
53
54
55
56
57
58
          assign LEDR[7] = stopRush;
59
60
          // clockDivide generates slower clocks to process different inputs
61
          clock_divider clockDivide (.clock(clk), .reset(SW[9]), .divided_clocks(divided_clocks));
62
63
          // hourCount counter takes in the KEY[0] signal and increases the
64
           // current hour, progressing the day in the parking lot
65
          hourCount timeCounter (.inc(~KEY[0]), .clk(clk), .reset(SW[9]), .out(hour_out));
66
67
          // carCount counter takes in the parking spot signals from VGPIO 28-30 and outputs
68
          // the number of spots left
69
          carCount \#(3) counter (.park1(V_GPIO[28]), .park2(V_GPIO[29]), .park3(V_GPIO[30]), .full(
      V_GPIO[34]), .clk(clk), .reset(SW[9]), .out(counter_out));
71
          // The carIncrCounter, used for saving the values to RAM
72
          carIncrCounter incrCounter (.inc(V_{GPIO[31]}), .buttonReset(KEY[0]), .clk(clk), .reset(SW
```

```
[9]), .out(incr_out));
 73
          // The addressIterator, used for reading the values from RAM
addrIter addrIterator (.inc(endGameHexOut), .clk(clk), .reset(SW[9]), .out(addr_out));
 74
 75
 76
           // Pipelining the values from hourCount and carCount, we can generate control signals
       that are used to control the seven segment display.
 78
          controlUnit controls (.occupied(V_GPIO[34]), .noneOccupied(zeroOccupied), .hour(hour_out
          .reset(SW[9]), .clk(clk), .startRush(startRush), .stopRush(stopRush), .endGameHexOut(
       endGameHexOut), .rushEnded(rushEnded));
 79
 80
           // Pipelining the values from control signals to the datapath to generate the rush hour
       values and then push them to the seg7 display dataPathUnit path (.startRush(startRush), .stopRush(stopRush), .endGameHexOut(
 81
       endGameHexOut), .clk(clk), .in(counter_out), .time_in(hour_out), .incr_in(incr_out), .
       addr_in(addr_out),
 82
                             hexoutO(HEXO), .hexout1(HEX1), .hexout2(HEX2), .hexout3(HEX3), .hexout4(HEX4), .hexout5(HEX5
       ));
 83
 84
       endmodule // DE1_SoC
 85
       // DE1_SoC_testbench tests all expected, unexpected, and edgecase behaviors
 86
 87
       module DE1_SoC_testbench();
 88
           logic CLOCK_50;
          logic [6:0] HEXO, HEX1, HEX2, HEX3, HEX4, HEX5;
logic [3:0] KEY;
 89
 90
 91
           logic [9:0] SW;
 92
          logic [9:0] LEDR;
 93
          wire [35:23] V_GPIO;
 94
 95
          logic[4:0] sensors;
 96
 97
          assign \{V_GPIO[31], V_GPIO[34], V_GPIO[28], V_GPIO[29], V_GPIO[30]\} = sensors;
 98
 99
          DE1_SOC dut (CLOCK_50, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, SW, LEDR, V_GPIO);
100
101
          // Setting up the clock.
           parameter CLOCK_PERIOD = 100;
102
103
           initial begin
              CLOCK_5\bar{0} \leftarrow 0;
104
105
              forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // toggle the clock forever
106
          end // initial
107
108
          initial begin
                                              @(posedge CLOCK_50); // reset
@(posedge CLOCK_50); // inc past max limit
              SW[9] < = 1;
109
              SW[9] \leftarrow 0;
110
              sensors[0] <= 0;</pre>
111
                                               @(posedge CLOCK_50);
                                              @(posedge CLOCK_50)
@(posedge CLOCK_50)
@(posedge CLOCK_50)
@(posedge CLOCK_50)
112
              sensors[0]
                           <= 1;
                           <= 1;
113
              sensors[0]
114
              sensors[0]
              sensors[0]
115
                           <= 1;
                                              @(posedge CLOCK_50)
116
              sensors[1]
                          <= 1;
                                              @(posedge CLOCK_50)
              sensors[1]
                          <= 1;
117
                                              @(posedge CLOCK_50)
118
              sensors[1] \ll 1;
              sensors[1] <= 1;
sensors[1] <= 1;
119
                                               @(posedge CLOCK_50);
                                              @(posedge CLOCK_50);
120
121
              sensors[1] \ll 1;
                                               @(posedge CLOCK_50);
              KEY[0] <= 0;
KEY[0] <= 0;</pre>
122
                                               @(posedge CLOCK_50);
123
                                               @(posedge CLOCK_50);
124
              KEY[0] \leftarrow 0;
                                              @(posedge CLOCK_50)
              sensors[2] \leftarrow 0;
125
                                              @(posedge CLOCK_50)
                                              @(posedge CLOCK_50);
@(posedge CLOCK_50);
@(posedge CLOCK_50);
@(posedge CLOCK_50);
@(posedge CLOCK_50);
              sensors[3] \leftarrow 0;
126
127
              sensors[4]
                          <= 0;
              KEY[0] <= 0;
KEY[0] <= 0;</pre>
129
              KEY[0] \leftarrow 0
130
                                              @(posedge CLOCK_50);
@(posedge CLOCK_50);
131
              KEY[0] \leftarrow 0;
              KEY[0] \leftarrow 0;
132
              KEY[0] \leftarrow 0;
                                              @(posedge CLOCK_50);
133
134
              KEY[0] \leftarrow 0;
                                               @(posedge CLOCK_50);
135
              $stop;
136
          end // initial
```

137

Revision: DE1_SoC