```
1   /* Name: Eugene Ngo
2      Date: 1/13/2023
3      Class: EE 371
4      Lab 6
5      Taken from Lab 1 and adapted for lab 6 task 1 */
6
7   // DE1_SoC is the top-level module that defines the I/Os for the DE-1 SoC board.
8   // DE1_SoC takes three switches from the GPIO as inputs, and outputs to 2 LEDs on the
    breadboard through GPIO and 6 7-bit
9   // hex displays (HEX0-HEX5). It displays "full" or "clear" messages when the parking lot is
    either full or clear, and it
10  // displays the decimal value of the number of cars in the lot accordingly.
11
12  module DE1_SoC #(parameter MAX=25) (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, V_GPIO, CLOCK_50);
13      output logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
14      inout logic [35:23] V_GPIO;
15      input logic CLOCK_50;
16
17     // Assigning and clk to CLOCK_50
18     logic clk;
19     assign clk = CLOCK_50;
20
21     logic enter, exit, full, clear;
22     logic [4:0] counter_out;
23
24     // Outputting a and b to breadboard
25     assign V_GPIO[32] = V_GPIO[23];
26     assign V_GPIO[35] = V_GPIO[24];
27
28     // carSensor parkCheck takes two switches from the breadboard as the input of the two
    parking sensors,
29     // and outputs to enter and exit when an entering or exiting vehicle is detected.
30     carSensor parkCheck (.a(V_GPIO[24]), .b(V_GPIO[23]), .enter, .exit, .clk, .reset(V_GPIO[
    30]));
31
32     // carCount counter takes enter and exit from sensors, and outputs the car count to
    counter_out.
33     // it als outputs full and clear status to full and clear.
34     carCount #(MAX) counter (.inc(enter), .dec(exit), .out(counter_out), .full, .clear, .clk,
     .reset(V_GPIO[30]));
35
36     // seg7 display takes counter_out, full, and clear from ct, and outputs decimal values
    or status messages to hex displays.
37     seg7 display (.in(counter_out), .full, .clear, .hexout0(HEX0), .hexout1(HEX1), .hexout2(
    HEX2), .hexout3(HEX3), .hexout4(HEX4), .hexout5(HEX5));
38
39  endmodule // DE1_SoC
40
41  // DE1_SoC_testbench tests all expected, unexpected, and edgecase behaviors
42  module DE1_SoC_testbench();
43     logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
44     wire [35:23] V_GPIO;
45     logic CLOCK_50;
46
47     DE1_SoC #(5) dut (.HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .V_GPIO, .CLOCK_50);
48
49     // Setting up a simulated clock.
50     parameter CLOCK_PERIOD = 100;
51     initial begin
52        CLOCK_50 <= 0;
53        forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // Forever toggle the clock
54     end
55
56     // Assiging logic to wire
57     logic reset, a, b;
58     assign V_GPIO[30] = reset;
59     assign V_GPIO[24] = a;
60     assign V_GPIO[23] = b;
61
62     // Testing the module
63     initial begin
64        // reset
65        reset <= 1;                    repeat(3) @(posedge CLOCK_50);
```

```
66
67          //enters until full
68          reset <= 0;       a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
69                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
70                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
71                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
72                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 1st car enters
73                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
74                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
75                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
76                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 2nd car enters
77                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
78                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
79                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
80                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 3rd car enters
81                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
82                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
83                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
84                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 4th car enters
85                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
86                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
87                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
88                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 5th car enters, full
89
90      // exits until clear
91                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
92                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
93                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
94                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
95                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 1st car exits
96                            a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
97                            a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
98                            a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
99                            a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 2nd car exits
100                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
101                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
102                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
103                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 3rd car exits
104                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
105                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
106                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
107                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 4th car exits
108                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
109                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
110                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
111                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50); // 5th car exits, clear
112
113         // direction changes while entering
114                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
115                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
116                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
117                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
118                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
119                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
120                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
121                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
122                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
123         // direction changes while exiting
124                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
125                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
126                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
127                           a <= 0; b <= 1; repeat(2) @(posedge CLOCK_50);
128                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
129                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
130                           a <= 1; b <= 1; repeat(2) @(posedge CLOCK_50);
131                           a <= 1; b <= 0; repeat(2) @(posedge CLOCK_50);
132                           a <= 0; b <= 0; repeat(2) @(posedge CLOCK_50);
133
134         $stop;
135     end
136 endmodule // DE1_SoC_testbench
```