```
1   // Alan Li
2   // 01/15/2022
3   // EE 371
4   // Lab #1
5
6   // parkingLotSensors takes in 1-bit A, B and clk as inputs and return
    1-bit enter and exit as outputs.
7
8   module parkingLotSensors(clk, reset, A, B, enter, exit);
9      input logic A, B, clk, reset;
10     output logic enter, exit;
11
12     // States
13     enum {unblocked, beginIn, blockIn, almostIn, beginOut, blockOut,
    almostOut} ps, ns;
14
15     // The parking lot sensor can have six cases as list below. beginIn,
    blockin and almostIn are pairs with almostOut, blockOut and beginOut.
16     // Assume the entrance is north-south direction. If car is entering
    from North, it will trigger sensor A first.
17     // If car is exiting from South it will trigger sensor B first.
18     // For example, beginIn and almost out both indicate that only
    sensorA is blocked, they only differences is that beginIn means the car
    is going in the south direction and almostOut is car going north.
19     // This "case pair" is designed to tackle car changing direction
    issue.
20     always_comb
21        begin
22           enter = 0;
23           exit = 0;
24
25           case(ps)
26
27              unblocked:
28                 if (A & ~B)          ns = beginIn;      // car begin to
    enter
29                 else if (~A & B)     ns = beginOut;     // car begin to
    exit
30                 // else if (~A & ~B)    ns = unblocked; // impossible
31                 else                 ns = unblocked;    // nothing happens
32
33              beginIn:
34                 if (A & B)           ns = blockIn;      // car is
    halfway entering
35                 else if (~A & ~B)    ns = unblocked;    // car just
    enters then back up
36                 // else if (A & ~B)     ns = beginIn;    // impossible
37                 else                 ns = beginIn;      // car does not
    move
38
39              blockIn:
40                 if (~A & B)          ns = almostIn;     // car almost
    enters (trigger both sensors)
41                 else if (A & ~B)     ns = beginIn;      // car backs up
42                 // else if (A & B)      ns = blockIn;    // impossible
43                 else                 ns = blockIn;      // car does not
    move
44
```

```
45                almostIn:
46                   if (~A & ~B)
47                      begin
48                                              ns = unblocked;    // car enters
49                                              enter = 1;
50                                              exit = 0;
51                      end
52                   else if (A % B)            ns = blockIn;      // car backs up
53                   // else if (~A & B)           ns = almostIn;  // impossible
54                   else                       ns = almostIn;     // car does not
      move
55
56                beginOut:
57                   if (A & B)                 ns = blockOut;     // car is
      halfway exiting
58                   else if (~A & ~B)          ns = unblocked;    // car backs up
59                   // else if (~A & B)           ns = beginOut;  // impossible
60                   else                       ns = beginOut;     // car does not
      move
61
62                blockOut:
63                   if (A & ~B)                ns = almostOut;    // car almost
      exits (trigger both sensors)
64                   else if (~A & B)           ns = beginOut;     // car backs up
65                   // else if (A & B)            ns = blockOut;  // impossible
66                   else                       ns = blockOut;     // car does not
      move
67
68                almostOut:
69                   if (~A & ~B)                                  // car exits
70                      begin
71                         ns = unblocked;
72                         enter = 0;
73                         exit = 1;
74                      end
75                   else if (A & B)            ns = blockOut;     // car backs up
76                   // else if (A & ~B)           ns = almostOut; // impossible
77                   else                       ns = almostOut;    // car does not
      move
78
79                /*
80                default:
81                   begin
82                      enter = 0;
83                      exit = 0;
84                   end
85                */
86             endcase
87          end
88
89
90       // if reset, the parking lot sensor goes to unblocked case
91       always_ff @(posedge clk)
92          begin
93             if (reset)
94                ps <= unblocked;
95             else ps <= ns;
96          end
```

```
 97  endmodule
 98
 99  // parkingLotSensor_testbench tests all expected behavior that the
     parking lot occupancy counter system in the lab may encounter
100  module parkingLotSensor_testbench();
101     logic clk, reset, A, B, enter, exit;
102
103     parkingLotSensors dut (.clk(clk), .reset(reset), .A(A), .B(B), .enter
     (enter), .exit(exit));
104
105     parameter CLOCK_PERIOD = 100;
106
107     initial begin
108        clk <= 0;
109        forever #(CLOCK_PERIOD/2) clk <= ~clk;
110     end
111
112     // I have already simulate the case where car enters and exits
     without changing directions in DE1_SoC
113     // For this testbench I will simulate the case where the car changes
     direction multiple times
114     initial begin
115        reset <= 1;          @(posedge clk);
116        reset <= 0;          @(posedge clk);
117        {A, B} <= 2'b00;     @(posedge clk);
118        {A, B} <= 2'b10;     @(posedge clk);
119        {A, B} <= 2'b11;     @(posedge clk);
120        {A, B} <= 2'b01;     @(posedge clk);
121        {A, B} <= 2'b11;     @(posedge clk);
122        {A, B} <= 2'b10;     @(posedge clk);
123        {A, B} <= 2'b11;     @(posedge clk);
124        {A, B} <= 2'b01;     @(posedge clk);
125        {A, B} <= 2'b00;     @(posedge clk); // simulation for car enters
126
127        {A, B} <= 2'b00;     @(posedge clk);
128        {A, B} <= 2'b01;     @(posedge clk);
129        {A, B} <= 2'b11;     @(posedge clk);
130        {A, B} <= 2'b10;     @(posedge clk);
131        {A, B} <= 2'b11;     @(posedge clk);
132        {A, B} <= 2'b01;     @(posedge clk);
133        {A, B} <= 2'b11;     @(posedge clk);
134        {A, B} <= 2'b10;     @(posedge clk);
135        {A, B} <= 2'b00;     @(posedge clk); // simulation for car exits
136        $stop;
137     end
138  endmodule
139
140
141
142
```