

```
1  /*
2   Ben Davis
3   3/5/24
4   EE 371
5   Lab 6, Task 2
6
7   This module keeps record of when rush hour begins
8   and when it ends. If rush hour starts, the output
9   hours are defaulted to 15, an hour not in the work day.
10  It uses inputs from parking spot presences and also
11  the parking lot's clock.
12  */
13
14  module rush_hour (
15      input logic clk, //clock
16      input logic rst, //reset
17      input logic [3:0] hour, //current hour
18      input logic car1, //car presence 1
19      input logic car2, //car presence 2
20      input logic car3, //car presence 3
21      output logic [3:0] rush_start, //hour rush hour starts
22      output logic [3:0] rush_end //hour rush hour ends
23  );
24
25      enum {off, start, middle, finish, done} ps, ns;
26      logic full; //full lot
27      logic empty; //empty lot
28      logic day_end; //the day has ended
29
30      assign day_end = (hour == 4'b1000); //if the day is over
31
32      assign full = (car1 && car2 && car3); //if the lot is full
33      assign empty = ((!car1) && (!car2) && (!car3)); // if the lot is empty
34
35      // flip flop to progress the state to ns or to reset
36      always_ff @(posedge clk) begin
37          if(rst) begin
38              ps <= off;
39          end else begin
40              ps <= ns;
41          end
42      end
43
44      //state progressions and logic
45      always_comb begin
46          case(ps)
47              off: begin
48                  if(full) ns <= start;
49                  else if(day_end) ns <= done;
50                  else ns <= off;
51              end
52              start: begin
53                  ns <= middle;
54              end
55              middle: begin
56                  if(empty) ns <= finish;
57                  else if(day_end) ns <= done;
58                  else ns <= middle;
59              end
60              finish: begin
61                  ns <= done;
62              end
63              done: begin
64                  ns <= done;
65              end
66          endcase
67      end
68
69      // flip flop to update rush hour start if it begins
70      always_ff @(posedge clk) begin
71          if(rst) begin
72              rush_start <= 4'b1110; //initial value is 14
73          end else if((ps == off) && (full)) begin
```

```
74     rush_start <= hour;
75   end else if (day_end && (ps == off)) begin
76     rush_start <= 4'b1111; //if no rush hour, value is 15
77   end
78 end
79
80 // flip flop to track rush hour end if it ends
81 always_ff @(posedge clk) begin
82   if(rst) begin
83     rush_end <= 4'b1110;
84   end else if((ps == middle) && (empty)) begin
85     rush_end <= hour;
86   end else if (day_end && (ps == off)) begin
87     rush_end <= 4'b1111;
88   end
89 end
90 endmodule
91 // testbench
92 module rush_hour_tb ();
93   logic clk, rst, car1, car2, car3;
94   logic [3:0] hour, rush_start, rush_end;
95
96   rush_hour dut (.*));
97
98   //clock setup
99   parameter clock_period = 100;
100  initial begin
101    clk <= 0;
102    forever #(clock_period /2) clk <= ~clk;
103  end //of clock setup
104
105  //tests a case where rush hour starts hr 2 and ends hr 4
106  initial begin
107    rst <= 0; hour <= 4'b0000; car1 <= 0; car2 <= 0;
108    car3 <= 0; @(posedge clk);
109    rst <= 0; hour <= 4'b0000; car1 <= 1; car2 <= 0;
110    car3 <= 0; @(posedge clk);
111    rst <= 0; hour <= 4'b0001; car1 <= 1; car2 <= 0;
112    car3 <= 0; @(posedge clk);
113    rst <= 0; hour <= 4'b0010; car1 <= 1; car2 <= 1;
114    car3 <= 1; @(posedge clk);
115    rst <= 0; hour <= 4'b0011; car1 <= 1; car2 <= 0;
116    car3 <= 0; @(posedge clk);
117    rst <= 0; hour <= 4'b0100; car1 <= 0; car2 <= 0;
118    car3 <= 0; @(posedge clk);
119    rst <= 0; hour <= 4'b0101; car1 <= 0; car2 <= 0;
120    car3 <= 0; @(posedge clk);
121    $stop;
122  end
123 endmodule
124
```