

```
1  /*
2    Ben Davis
3    3/5/24
4    EE 371
5    Lab 6, Task 2
6
7    This module increments the parking lots hour by one everytime
8    KEY[0] is pressed. It takes a reset signal from switch9,
9    a clock from CLOCK_50, and has a 3 bit output.
10 */
11
12 module hour_time (
13     input logic clk, //clock
14     input logic rst, //reset
15     input logic inc, //KEY[0]
16     output logic [3:0] hour //current hour
17 );
18     enum {s0, off1, s1, off2, s2, off3, s3, off4, s4, off5,
19         s5, off6, s6, off7, s7, s8} ps, ns;
20
21     //progresses states and also resets the state
22     always_ff @(posedge clk) begin
23         if(rst) begin
24             ps <= s0;
25         end else begin
26             ps <= ns;
27         end
28     end
29
30     //state logic
31     always_ff @(posedge clk) begin
32         case(ps)
33             //every s# has the hour match its #
34             //off states are intermediate states to
35             //allow the system to take a break
36             s0: begin
37                 hour <= 4'b0000;
38                 if(inc) ns <= s1;
39                 else ns <= s0;
40             end
41             off1: begin
42                 ns <= s1;
43                 hour <= hour;
44             end
45             s1: begin
46                 hour <= 4'b0001;
47                 if(inc) ns <= s2;
48                 else ns <= s1;
49             end
50             off2: begin
51                 ns <= s2;
52                 hour <= hour;
53             end
54             s2: begin
55                 hour <= 4'b0010;
56                 if(inc) ns <= s3;
57                 else ns <= s2;
58             end
59             off3: begin
60                 ns <= s3;
61                 hour <= hour;
62             end
63             s3: begin
64                 hour <= 4'b0011;
65                 if(inc) ns <= s4;
66                 else ns <= s3;
67             end
68             off4: begin
69                 ns <= s4;
70                 hour <= hour;
71             end
72             s4: begin
73                 hour <= 4'b0100;
```

```
74         if(inc) ns <= s5;
75         else ns <= s4;
76     end
77 off5: begin
78     ns <= s5;
79     hour <= hour;
80 end
81 s5: begin
82     hour <= 4'b0101;
83     if(inc) ns <= s6;
84     else ns <= s5;
85 end
86 off6: begin
87     ns <= s6;
88     hour <= hour;
89 end
90 s6: begin
91     hour <= 4'b0110;
92     if(inc) ns <= s7;
93     else ns <= s6;
94 end
95 off7: begin
96     ns <= s7;
97     hour <= hour;
98 end
99 s7: begin
100     hour <= 4'b0111;
101     if(inc) ns <= s8;
102     else ns <= s7;
103 end
104 s8: begin
105     hour <= 4'b1000;
106     ns <= s8;
107 end
108 endcase
109 end
110 endmodule
111 //testbench
112 module hour_time_tb ();
113
114     //logic
115     logic clk, rst, inc;
116     logic [3:0] hour;
117
118     hour_time dut (.*);
119
120     //clock setup
121     parameter cl_pd = 100;
122     initial begin
123         clk <= 0;
124         forever #(cl_pd /2) clk <= ~clk;
125     end //of clock setup
126
127     //tests a case where the hour is incremented 3 times
128     initial begin
129         rst <= 0; inc <= 0; @(posedge clk);
130         rst <= 0; inc <= 1; @(posedge clk);
131         rst <= 0; inc <= 1; @(posedge clk);
132         rst <= 0; inc <= 0; @(posedge clk);
133         rst <= 0; inc <= 0; @(posedge clk);
134         rst <= 0; inc <= 1; @(posedge clk);
135         rst <= 0; inc <= 0; @(posedge clk);
136         $stop;
137     end
138 endmodule
```