```
/*
   Ben Davis
   3/5/24
   EE 371
   Lab 6, Task 2

   This module increments the parking lots hour by one everytime
   KEY[0] is pressed. It takes a reset signal from switch9,
   a clock from CLOCK_50, and has a 3 bit output.
*/

module hour_time (
      input logic clk, //clock
      input logic rst, //reset
      input logic inc, //KEY[0]
      output logic [3:0] hour //current hour
);
   enum {s0, off1, s1, off2, s2, off3, s3, off4, s4, off5,
   s5, off6, s6, off7, s7, s8} ps, ns;

   //progresses states and also resets the state
   always_ff @(posedge clk) begin
      if(rst) begin
         ps <= s0;
      end else begin
         ps <= ns;
      end
   end

   //state logic
   always_ff @(posedge clk) begin
      case(ps)
      //every s# has the hour match its #
      //off states are intermediate states to
      //allow the system to take a break
         s0: begin
               hour <= 4'b0000;
               if(inc) ns <= s1;
               else ns <= s0;
            end
         off1: begin
               ns <= s1;
               hour <= hour;
            end
         s1: begin
               hour <= 4'b0001;
               if(inc) ns <= s2;
               else ns <= s1;
            end
         off2: begin
               ns <= s2;
               hour <= hour;
            end
         s2: begin
               hour <= 4'b0010;
               if(inc) ns <= s3;
               else ns <= s2;
            end
         off3: begin
               ns <= s3;
               hour <= hour;
            end
         s3: begin
               hour <= 4'b0011;
               if(inc) ns <= s4;
               else ns <= s3;
            end
         off4: begin
               ns <= s4;
               hour <= hour;
            end
         s4: begin
               hour <= 4'b0100;
```

```
                if(inc) ns <= s5;
                else ns <= s4;
            end
        off5: begin
                ns <= s5;
                hour <= hour;
            end
        s5: begin
                hour <= 4'b0101;
                if(inc) ns <= s6;
                else ns <= s5;
            end
        off6: begin
                ns <= s6;
                hour <= hour;
            end
        s6: begin
                hour <= 4'b0110;
                if(inc) ns <= s7;
                else ns <= s6;
            end
        off7: begin
                ns <= s7;
                hour <= hour;
            end
        s7: begin
                hour <= 4'b0111;
                if(inc) ns <= s8;
                else ns <= s7;
            end
        s8: begin
                hour <= 4'b1000;
                ns <= s8;
            end
        endcase
    end
endmodule
//testbench
module hour_time_tb ();

    //logic
    logic clk, rst, inc;
    logic [3:0] hour;

    hour_time dut (.*);

    //clock setup
    parameter cl_pd = 100;
    initial begin
        clk <= 0;
        forever #(cl_pd /2) clk <= ~clk;
    end //of clock setup

    //tests a case where the hour is incremented 3 times
    initial begin
        rst <= 0; inc <= 0; @(posedge clk);
        rst <= 0; inc <= 1; @(posedge clk);
        rst <= 0; inc <= 1; @(posedge clk);
        rst <= 0; inc <= 0; @(posedge clk);
        rst <= 0; inc <= 0; @(posedge clk);
        rst <= 0; inc <= 1; @(posedge clk);
        rst <= 0; inc <= 0; @(posedge clk);
        $stop;
    end
endmodule
```