```
module runway (clk, reset, in, out);
   input  logic  clk, reset;
   input  logic  [1:0] in;
   output logic  [2:0] out;

   // Define light states
   enum { L, M, R, D} ps, ns;
   always_comb begin
      case (ps)

         L: if ((in == 2'b00) || (in == 2'b10)) begin
               ns = M;
               out = 3'b010;
               end
            else begin // (in == 2'b01)
               ns = R;
               out = 3'b001;
               end

         M: if (in == 2'b00) begin
               ns = D;
               out = 3'b101;
               end
            else if (in == 2'b01) begin
               ns = L;
               out = 3'b100;
               end
            else begin // (in == 2'b10)
               ns = R;
               out = 3'b001;
               end

         R: if ((in == 2'b00) || (in == 2'b01)) begin
               ns = M;
               out = 3'b010;
               end
            else begin // (in == 2'b10)
               ns = L;
               out = 3'b100;
               end

         D: begin
            ns = M;
            out = 3'b010;
            end

      endcase

   end

   always_ff @(posedge clk) begin
      if (reset)
         ps <= M;
      else
         ps <= ns;
   end

endmodule


module runway_testbench();
   logic  clk, reset;
   logic  [1:0] in;
   logic  [2:0] out;

   runway dut (clk, reset, in, out);

   // Set up the clock.
   parameter CLOCK_PERIOD=100;
   initial begin
      clk <= 0;
      forever #(CLOCK_PERIOD/2) clk <= ~clk;
   end
```

```
74
75      // Set up the inputs to the design.  Each line is a clock cycle.
76      initial begin
77                                  @(posedge clk);
78         reset <= 1;              @(posedge clk);
79         reset <= 0; in <= 2'b00; @(posedge clk);
80                                  @(posedge clk);
81                                  @(posedge clk);
82                                  @(posedge clk);
83                                  @(posedge clk);
84                                  @(posedge clk);
85                     in <= 2'b01;   @(posedge clk);
86                                  @(posedge clk);
87                                  @(posedge clk);
88                                  @(posedge clk);
89                                  @(posedge clk);
90                                  @(posedge clk);
91                     in <= 2'b10;   @(posedge clk);
92                                  @(posedge clk);
93                                  @(posedge clk);
94                                  @(posedge clk);
95                                  @(posedge clk);
96                                  @(posedge clk);
97                                  @(posedge clk);
98         $stop; // End the simulation.
99      end
100
101  endmodule
```