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

   logic [1:0] ps, in1, in2;

   always begin
      case (ps)
         2'b0:    out = 3'b101;
         2'b1:    out = 3'b100;
         2'b10:   out = 3'b001;
         2'b11:   out = 3'b010;
         default: out = 3'bX;
      endcase
   end

   always_ff @(posedge clk) begin
      in1 <= in;
   end

   always_ff @(posedge clk) begin
      in2 <= in1;
   end

   //DFF
   always_ff @(posedge clk) begin
      if (reset)
         ps <= 2'b11;
      else
         case (ps)
            2'b0:    ps <= 2'b11;
            2'b1:    if (in2 == 2'b1)  ps <= 2'b10;
                     else              ps <= 2'b11;
            2'b10:   if (in2 == 2'b10) ps <= 2'b1;
                     else              ps <= 2'b11;
            2'b11:   ps <= in2;
            default: ps <= 2'bX;
         endcase
   end

endmodule

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

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

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

   // Set up the inputs to the design. Each line is a clock cycle.
   initial begin
                              @(posedge clk);
      reset <= 1;             @(posedge clk);
      reset <=0; in <= 2'b0;  @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
      in <= 2'b10;            @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
      in <= 2'b01;            @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
                              @(posedge clk);
      in <= 2'b0;             @(posedge clk);
      in <= 2'b1;             @(posedge clk);
      in <= 2'b10;            @(posedge clk);
      $stop;   // End the simulation
   end
endmodule


```