```
1   module button_press (clk, reset, in, out);
2
3      // Takes in a "button press", where 1 will only be output once
4      // despite the actual input being on for many clock cycles
5
6      input  logic clk, reset, in;
7      output logic out;
8
9      enum { NONE, WAITOFF } ps, ns;
10
11     // NS logic + output
12     always_comb begin
13        case (ps)
14
15           NONE:    begin
16                       if (in)  begin
17                          ns = WAITOFF;
18                          out = 1'b1;
19                       end
20                       else     begin
21                          ns = NONE;
22                          out = 1'b0;
23                       end
24                    end
25
26           WAITOFF: begin
27                       if (in)  begin
28                          ns = WAITOFF;
29                          out = 1'b0;
30                       end
31                       else     begin
32                          ns = NONE;
33                          out = 1'b0;
34                       end
35                    end
36        endcase
37     end
38
39     always_ff @(posedge clk) begin
40        if (reset)
41           ps <= NONE;
42        else
43           ps <= ns;
44     end
45  endmodule
46
47
48  module button_press_testbench ();
49     logic clk, reset, in, out;
50
51     button_press dut (clk, reset, in, out);
52     // Set up the clock.
53     parameter CLOCK_PERIOD=100;
54     initial begin
55     clk <= 0;
56     forever #(CLOCK_PERIOD/2) clk <= ~clk;
57     end
58
59     // Set up the inputs to the design. Each line is a clock cycle.
60     initial begin
61                                 @(posedge clk);
62        reset <= 1;              @(posedge clk);
63        reset <= 0; in <= 0;     @(posedge clk);
64                                 @(posedge clk);
65                                 @(posedge clk);
66                                 @(posedge clk);
67        in <= 1;                 @(posedge clk);
68                                 @(posedge clk);
69                                 @(posedge clk);
70                                 @(posedge clk);
71        in <= 0;                 @(posedge clk);
72                                 @(posedge clk);
73                                 @(posedge clk);
74                                 @(posedge clk);
75        $stop; // End the simulation.
76     end
```

```
77    endmodule
78
```