```
1   `timescale 1ns / 1ps
2   //*************************************************************//
3   //  Class: CECS 361                                            //
4   //  Project: Project1-Cecs361                                  //
5   //                                                             //
6   //  File name: <debounce.v>                                    //
7   //  Abstract: Debounce will stabilize the output to show only the //
8   //            necessary transitions.                           //
9   //            Reference to Pong Chu's Debounce                 //
10  //  Created by       <Alina Suon> on <09-18-18>.                //
11  //  Copyright © 2018 <Alina Suon>. All rights reserved.         //
12  //
    //                                                                    //
13  //  In submitting this file for class work at CSULB            //
14  //  I am confirming that this is my work and the work          //
15  //  of no one else. In submitting this code I acknowledge that //
16  //  plagiarism in student project work is subject to dismissal.  //
17  //  from the class                                             //
18  //*************************************************************//
19
20  module debounce(clk, rst, sw, db);
21     input wire clk, rst;
22     input wire sw;
23     output reg db;
24
25     makePulse makePulse(clk, rst, pulseIt);
26
27  //makePulse    P1 (clk, rst, pulse);
28     localparam [2:0]
29              zero0   = 3'b000,
30              s1_01 = 3'b001,
31              s1_10 = 3'b010,
32              s1_11 = 3'b011,
33              one1    = 3'b100,
34              s0_01 = 3'b101,
35              s0_10 = 3'b110,
36              s0_11 = 3'b111;
37
38  //Counter bit = 20 to get 10ms tick
39     localparam n = 20;
40
41  //Declare signals
42        reg [n-1:0] qReg;
43        wire [n-1 : 0] q_ns ;
44        wire pulse;
45        reg [2:0] stateReg , stateNS ;
46
47  //10ms tick Counter
48     always @ (posedge clk, posedge rst)
49        if (rst)
50           qReg <= 0;
51        else
52           qReg <= q_ns;
53
54  //q next state
55     assign q_ns = qReg + 1;
56
```

```
57   //Output pulse / tick
58      assign pulse = (qReg == 0) ? 1'b1 : 1'b0;
59
60   // Finite state machine //
61   // State register
62      always @ ( posedge clk , posedge rst)
63         if (rst)
64            stateReg <= zero0;
65         else
66            stateReg <= stateNS ;
67
68      // next-state logic and output logic
69      always @ (*)
70         begin
71            stateNS = stateReg;  // default state: same state
72            db = 1'b0;              // default output: 0
73
74      //Case statement
75      case (stateReg)
76         zero0:
77               if(sw)
78                  stateNS = s1_01;
79                  s1_01:
80               if(~sw)
81                  stateNS = zero0;
82               else
83                  if(pulse)
84                  stateNS = s1_10;
85         s1_10:
86               if(~sw)
87                  stateNS = zero0;
88               else
89                  if (pulse)
90                  stateNS = s1_11;
91         s1_11:
92               if(~sw)
93                  stateNS = zero0;
94               else
95                  if (pulse)
96                  stateNS = one1;
97         one1:
98               begin
99               db = 1'b1;
100              if (~sw)
101                 stateNS = s0_01;
102              end
103        s0_01:
104              begin
105              db = 1'b1;
106              if (sw)
107                 stateNS = one1;
108              else
109                 if (pulse)
110                 stateNS = s0_10;
111              end
112        s0_10:
113              begin
```

```
114                db = 1'b1;
115                if (sw)
116                   stateNS = one1;
117                else
118                   if (pulse)
119                   stateNS = s0_11;
120                end
121          s0_11:
122                begin
123                db = 1'b1;
124                if (sw)
125                   stateNS = one1;
126                else
127                   if (pulse)
128                   stateNS = zero0;
129                end
130          default: stateNS = zero0;
131          endcase
132          end
133    endmodule
```