```
`timescale 1ns / 1ps
/**********************************************
**********************************
*
* Module: debounce
*
* Author: Eric Christie
* Class: ECEN 220, Section 1, Winter 2021
* Date: 3/16/21
*
* Description: Debounces noisy signal,
waiting 5 ms for any change in signal to
pass through
*
*
**********************************************
**********************************/

`default_nettype none
module debounce(
    input wire logic clk, reset, noisy,
    output logic debounced);

    logic clrTimer, timerDone;
```

```
    logic [18:0] timerCount;

    //           max value, bits     clk,
reset, inc, rollover, count
    Mod_Counter #(500000, 19) timer(clk,
clrTimer, 1'b1, timerDone, timerCount);
//counts to 5 ms then rolls over

    //assigning variables for FSM
    typedef enum logic[1:0] {s0, s1, s2,
s3, ERR='X} StateType;
    StateType ns, cs;

    always_comb begin
        ns = ERR; //default states
        debounced = 0;
        clrTimer = 0;

        if(reset) ns = s0;

        else //state transitions
            case(cs)
                s0: begin
                    clrTimer = 1; //OUTPUT
```

```
                    if(noisy) ns = s1;
//moves to next state
                    else
                        ns = s0; //stays
in state
                    end

                s1: if(!noisy) ns = s0;
//noisy goes low, goes back to s0
                    else if(noisy &&
timerDone) ns = s2; //moves to s2 if noisy
held until timer finishes
                    else ns = s1; //noisy
high, but waiting for timer

                s2: begin
                    debounced = 1; //OUTPUT
                    clrTimer = 1; //OUTPUT
                    if(!noisy) ns = s3;
//noisy goes low, move to s3 to evaluate
                    else ns = s2; //stays
                end

                s3: begin
```

```
                    debounced = 1; //OUTPUT
                    if(noisy) ns = s2;
//noisy goes high again, stops checking
                    else if(!noisy &&
timerDone) ns = s0; //noisy held low for 5
ms
                    else ns = s3;
//waiting for timer
                end
            endcase
    end

    always_ff @(posedge clk) cs <= ns;
//assigns next state at each clk
endmodule
```