```
`timescale 1ns / 1ps
/*************************************************************************** *
* Module: debounced
*
* Author: Noah Hanks
* Class: ECEN 220, Section 3, Fall 2020 - ECEN 220, Section 1, Winter 2020 * Date: 3
November 2020
*
* Description: Uses a counter to determine how long a button has been pressed to
tell if the button bounced or not.
*
*
***************************************************************************/
`default_nettype none
module debounce(
 input wire logic clk, reset, noisy,
 output logic debounced
 );

 logic timerDone, clrTimer;
 logic[18:0] extra;
 typedef enum logic[1:0] {s0, s1, s2, s3, ERR='X} StateType;
 StateType ns, cs;

 mod_counter #(500000, 19) MC0(.clk(clk), .reset(clrTimer), .increment(1'b1),
.rolling_over(timerDone), .count(extra));

 //this block determines if the button is help long enough to be counted as a single
input.
 //otherwise, it says that the input was never made because the value of noisy
changed too fast.
 always_comb begin
    ns = ERR;
    debounced = 0;
    clrTimer = 0;
    if(reset)
    ns = s0;
    else
    case(cs)
    s0: begin
        clrTimer = 1'b1;
        if(noisy)
        ns = s1;
        else
        ns = s0;
        end
    s1: if(noisy & timerDone)
    ns = s2;
    else if(noisy & ~timerDone)
        ns = s1;
    else
        ns = s0;
    s2: begin
        debounced = 1'b1;
        clrTimer = 1'b1;
        if(noisy)
            ns = s2;
        else
            ns = s3;
        end
    s3: begin
    ns = s3;
    debounced = 1'b1;
    if(~noisy & ~timerDone)
        ns = s3;
    else if(noisy)
        ns = s2;
    else
        ns = s0;
    end
 endcase
 end

 //this block makes sure the current state is always being set to the next state
 always_ff @(posedge clk)
 cs <= ns;



endmodule
```