3. CODE

DESIGN CODE

```
module debouncing_cir (input logic clk,
input logic rst,
input logic d,
output logic q
);
logic [30:0] clk_div;
always@ (posedge clk)
begin
if (clk)
clk_div<=0;
else
clk_div<=clk_div+1;
end
logic d_ff;
always@ (posedge clk_div[30] or posedge rst)
begin
if (rst)
d_ff <= 1'b0;
else
d_ff <= d;
end
logic [3:0] counter;
always @ (posedge clk_div[30] or posedge rst)
begin
if (rst)
counter <= 4'b0000;
else if (counter == 4'b1111)
counter <= 4'b0000;
else
counter <= counter + 1;
```

```
end
logic debounce_signal;
assign debounce_signal = (counter == 4'b1111);
always @(posedge clk_div[30] or posedge rst)
begin
if (rst)
q <= 1'b0;
else if (debounce_signal)
q <= d_ff;
end
endmodule
```

TEST BENCH CODE

```
module debouncing_cir_tb();
logic clk;
logic rst;
logic d;
logic q;
debouncing_cir uut(clk,rst,d,q);
always begin
#20 clk = ~clk;
end
initial
begin
clk = 0;
rst = 1;
d = 0;
#50 rst = 0;
#50 d = 1;
#50 d = 0;
#50 d = 1;
#50 d = 0;
```

```
#50 d = 0;
#50 d = 0;
#50 d = 1;
#50 d = 1
#200
$finish;
end
endmodule
```