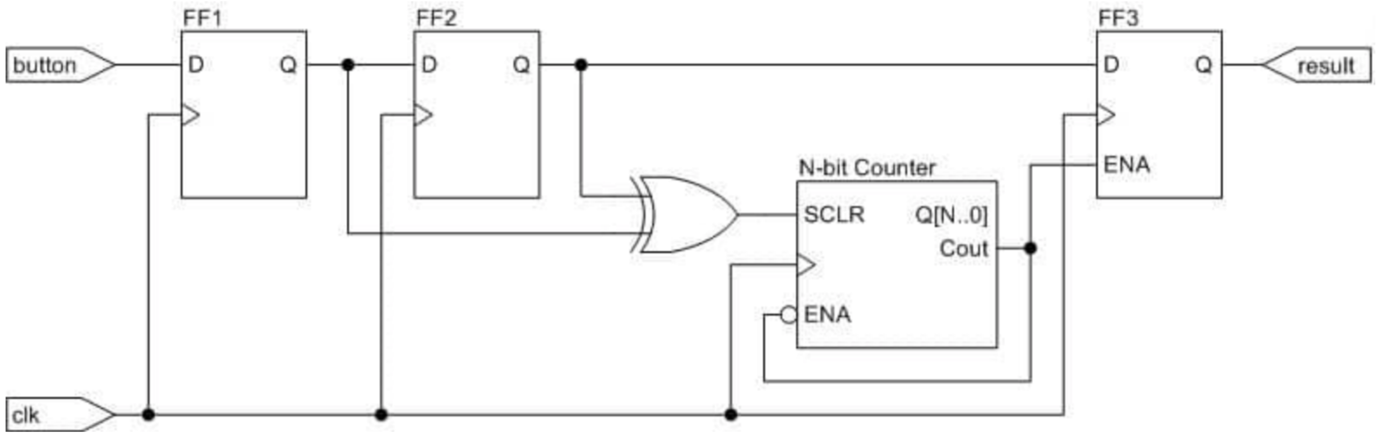
2 . Ans :

## Debounce Circuit

Using mechanical switches for a user interface is a ubiquitous practice. However, when these switches are actuated, the contacts often rebound, or bounce, off one another before settling into a stable state. There are various hardware and software techniques to eliminate this problem. The debounce component presented here is a simple digital logic circuit that addresses this temporary ambiguity (a common task when interfacing FPGAs or CPLDs with pushbuttons or other switches).



### Critical Specifications

The most critical feature of this debounce circuit is the *"debounce time period"* which basically depends upon the size of the counter.

Note that, here we are using a counter with a register size of N = 11, which is used for counting from 0 to $2^{11-1} = 1024$.

$$Critical\ \text{``}Debounce\text{''}\ Period \ = \ \frac{2^{N-1}}{f}$$

where $f$ = Frequency of the Clock used.
In this case :

$$Critical\ \text{``}Debounce\text{''}\ Period \ = \ \frac{2^{11-1}}{50\ MHz} \ = \ 20.48\ ms$$

This time period indicates the time for which the input has to remain stable for the output to be read at the output of the switch. [1]

### Verilog Code

```
//------------------------------ D Flipflop------------------------------------

`timescale 1ns / 1ps

module D_FF(Q,D,clk,rst);

```

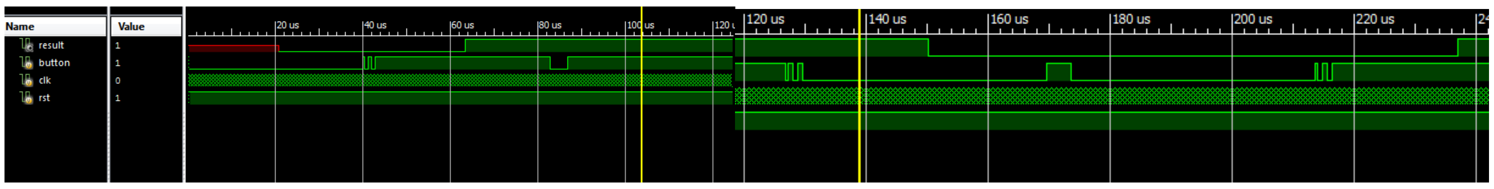
[1]This particular time period was also chosen for ease during simulation

```
input D,clk,rst;
output reg Q;

always@(posedge clk)
begin
if(rst == 1'b0) Q <= 1'b0;
else Q <= D;
end

endmodule


//-------------------------D Flipflop with enable-----------------------------

`timescale 1ns / 1ps

module D_FF_EN(Q,D,clk,en);

input D,clk,en;
output reg Q;

always @(posedge clk)
begin

if(en == 1'b1) Q <= D;
else Q <= Q;

end



endmodule


//------------------------------ Counter-------------------------------------


`timescale 1ns / 1ps

module counter(cout,sclr,en,clk,rst);

input sclr,en,clk,rst;
output cout;
parameter N = 11;

reg [N-1:0] q_present,q_next;

assign cout = q_present[N-1];

always @(posedge clk)
begin

if(rst == 1'b0) q_present <= {N{1'b0}};
else q_present <= q_next;

end

always @(sclr,q_present,en)
begin
```

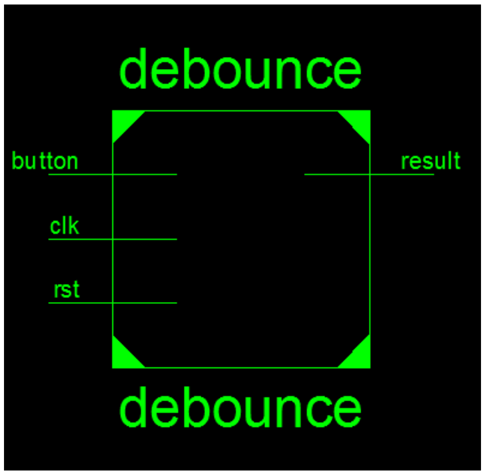
```

case({sclr,en})

2'b00 : q_next <= q_present;
2'b01 : q_next <= q_present + 1;
default : q_next <= {N{1'b0}};

endcase
end

endmodule

//---------------------------- Debounce Circuit--------------------------------

`timescale 1ns / 1ps
module debounce(result,button,clk,rst);

input button,clk,rst;
output result;
wire  w0,w1,w2,w3;

D_FF D1(w0,button,clk,rst);
D_FF D2(w1,w0,clk,rst);

xor XG(w2,w1,w0);

counter C1(w3,w2,~w3,clk,rst);

D_FF_EN DF(result,w1,clk,w3);

endmodule


//------------------------------ TestBench--------------------------------------

`timescale 1ns / 1ps
module test;

// Inputs
reg button;
reg clk;
reg rst;

// Outputs
wire result;

// Instantiate the Unit Under Test (UUT)
debounce uut (
.result(result),
.button(button),
.clk(clk),
.rst(rst)
);

initial begin

clk = 0;
forever #10 clk = ~clk;

```

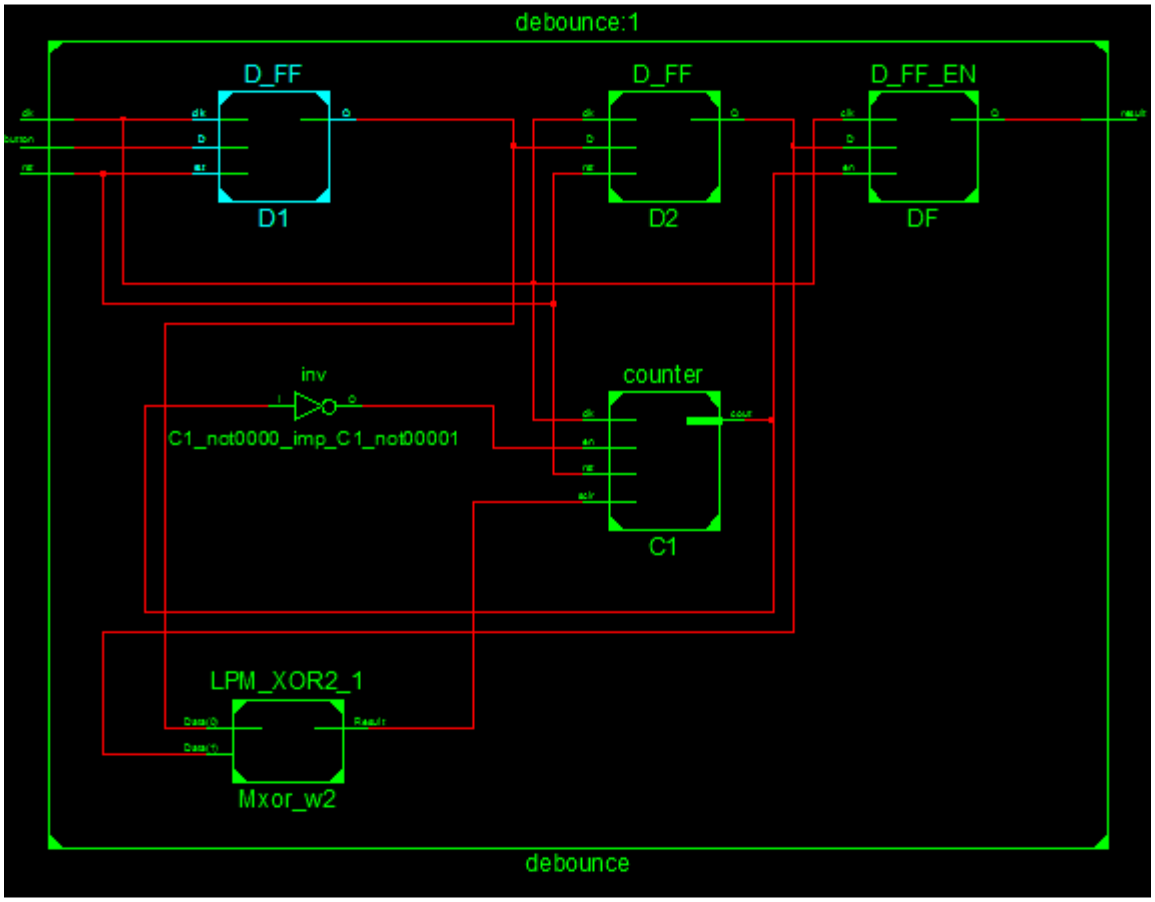
```
end

initial begin

rst = 1'b0;
#200
rst =1'b1;
button = 1'b0;
end

always
begin
#40000 button = 1'b1;

#400 button = 1'b0;

#800 button = 1'b1;

#800 button = 1'b0;

#800 button = 1'b1;

#40000 button = 1'b0;

#4000 button = 1'b1;

#40000 button = 1'b0;

#400 button = 1'b1;

#800 button = 1'b0;

#800 button = 1'b1;

#800 button = 1'b0;

#40000 button = 1'b1;

#4000 button = 1'b0;

end

endmodule
```

**Simulation Results :**



**Schematic :**


debounce
button
clk
rst
result
debounce


debounce:1
D_FF
D1
D_FF
D2
D_FF_EN
DF
inv
C1_not0000_imp_C1_not00001
counter
C1
LPM_XOR2_1
Mxor_w2
debounce