Name of Experiment : 4-bit Counter Date : 13/12/22

Experiment No : 5 Experiment Result : Verified

4-bit Counter

Truth Table

Res	clk	Up Counter (dir=0)	Down Counter (dir=1)
X	0	No change	No change
0	↑	0000	1111
0	↑	0001	1110
0	↑	0010	1101
0	↑	0011	1100
0	↑	0100	1011
0	↑	0101	1010
0	↑	----	----
0	↑	----	----
0	↑	1111	0000
1	X	0000	0000

Code :-

```
'timescale 1ns / 1ps
module cntr (cnt, rst, dir, clk);
input clk, rst, dir;
output [3:0] cnt;
reg [3:0] cnt;
always @ (rst)
cnt = 0;
```

Teacher's Signature : _____

Name of Experiment : 4-bit counter Date : 13/12/22

Experiment No : 5 Experiment Result : Verified Page No. 28

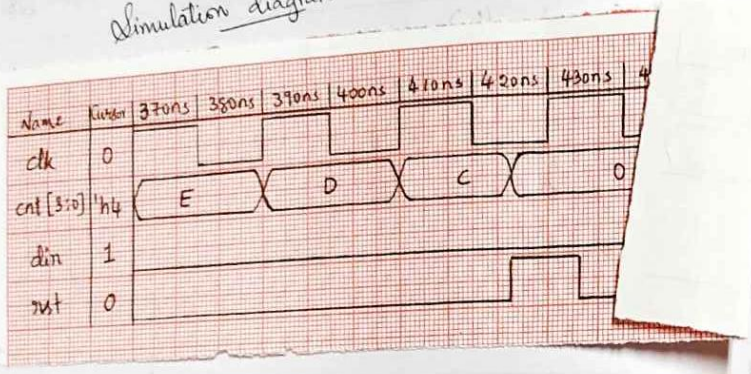
```
always @ (posedge clk)
begin
if (!rst)
begin
if (dir)
   cnt = cnt + 1;
else
   cnt = cnt - 1;
end
end
endmodule
```

Testbench

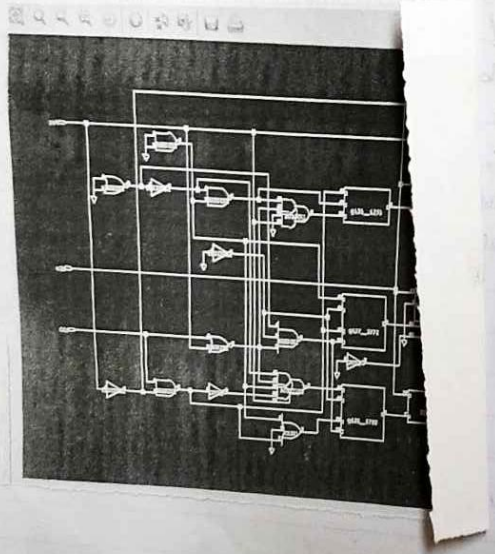
```
`timescale 1ns / 1ps
module cntr_tb;
reg clk, dir, rst;
wire [3:0] cnt;
cntruut (cnt, rst, dir, clk);
initial clk = 1'b0;
always
#5 clk = ~clk;
initial
begin
rst = 1; #10;
rst = 0; dir = 0; #205;
rst = 1; #10;
rst = 0; dir = 1; #200;
```

Teacher's Signature : _____

Simulation diagram



Schematic



Name of Experiment : 4-bit counter Date : 13/12/22

Experiment No : 5 Experiment Result : Verified Page No. 29

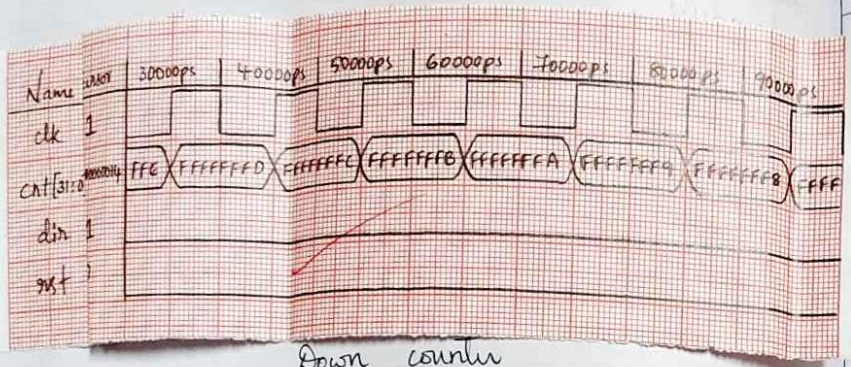
```
$finish;
end
endmodule
```

Result :-

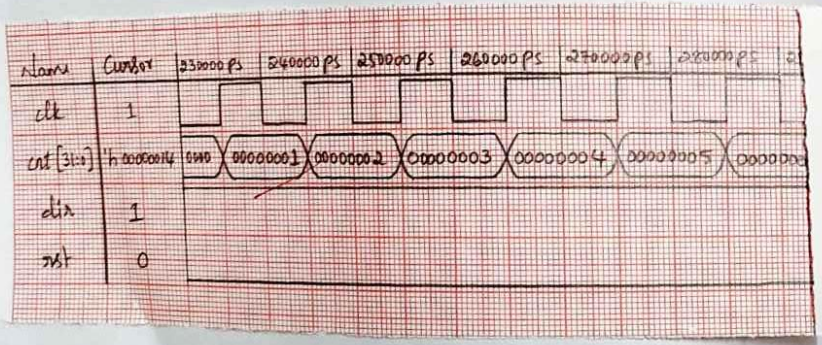
Implementation of 4-bit Counter has been verified

Teacher's Signature : _____

Simulation waveform



Down counter



Up counter

Name of Experiment : 32-bit Counter Date : 13/12/22

Experiment No : 5 Experiment Result : Verified Page No. 30

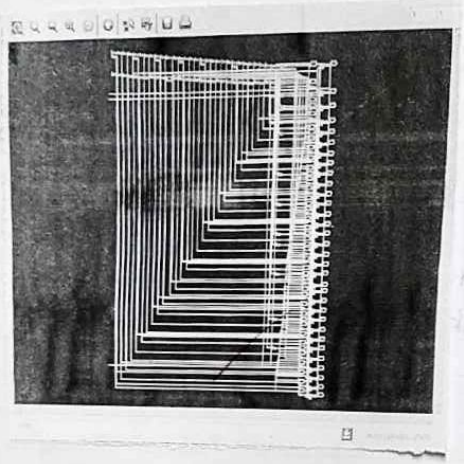
32-bit Counter

Code :-

```
`timescale 1ns / 1ps
module cntr (cnt, rst, dir, clk);
input clk, rst, dir;
output [31:0] cnt;
reg [31:0] cnt;
always @ (rst)
cnt = 31'hFFFFFFFF;
always @ (posedge clk)
begin
if (!rst)
begin
if (dir)
    cnt = cnt + 1;
else
    cnt = cnt - 1;
end
end
endmodule
```

Teacher's Signature : _____

Schematic



Name of Experiment : 32-bit Counter Date : 13/12/22

Experiment No : 5 Experiment Result : Verified Page No. 31

Test bench

```
`timescale 1ns/1ps
module cntr_tb;
reg clk, dir, rst;
wire [31:0] cnt;
cntruut (cnt, rst, dir, clk);
initial clk = 1'b0;
always
#5 clk = ~clk;
initial
begin
rst = 1; #10
rst = 0; dir = 0; #1000;
rst = 1; #10
rst = 0; dir = 1; #1000;
$finish
end
endmodule
```

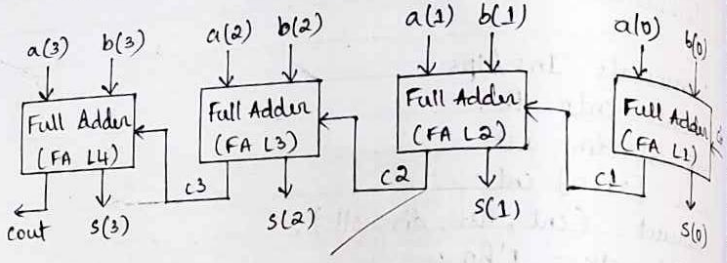
Result :-

Implementation of 32-bit counter has been verified.

25/25 A 13/12/22

Teacher's Signature :

Block diagram



Working

						cin	In Hexa
a	=		0	1	0	1	0
b	=		1	0	1	0	5
	=	0	1	1	1	1	A
		cout			sum		F

						cin	In Hexa
a	=		0	1	0	1	1
b	=		1	0	1	1	5
s	=	1	0	0	0	0	A
		cout			sum		0

Name of Experiment : 4-bit adder Date : 13/12/22

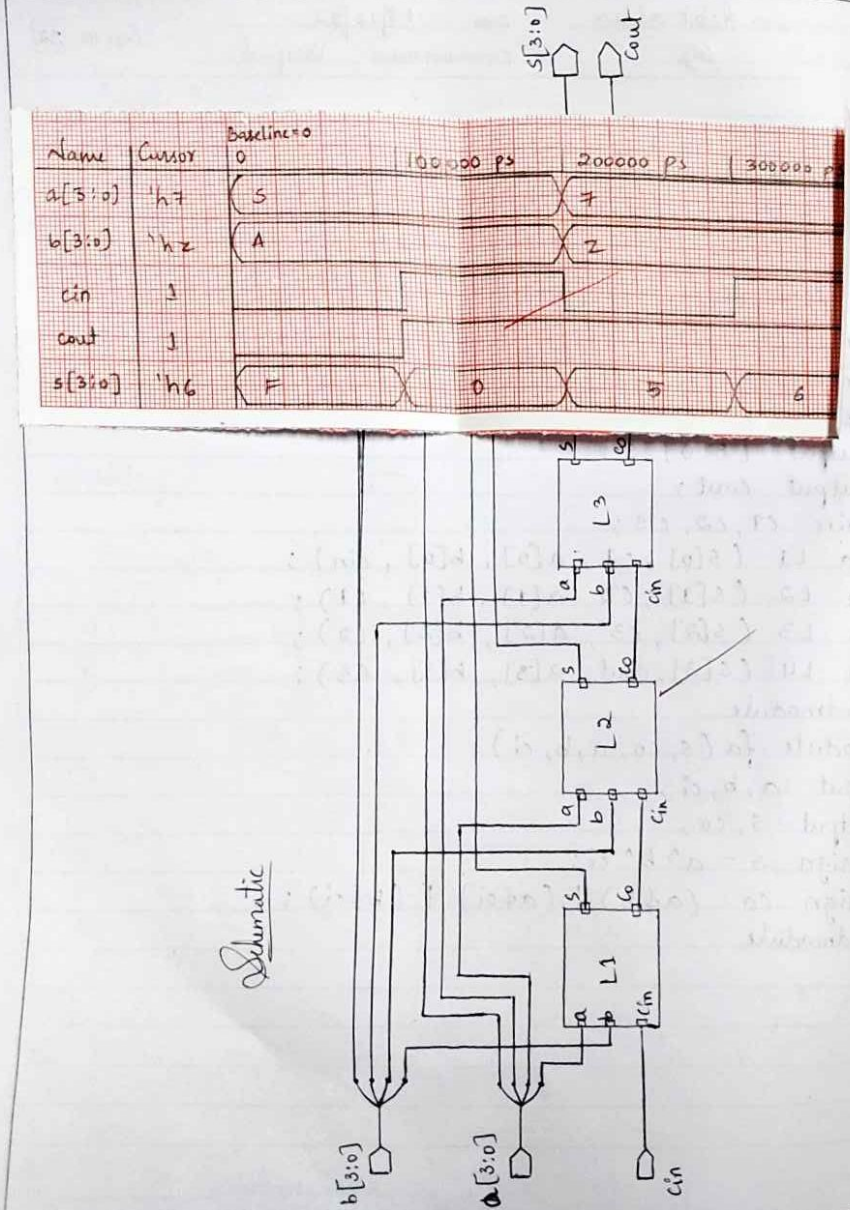
Experiment No : 6 Experiment Result : Verified Page No. 32

4-bit Parallel Adder

Code :-

```
`timescale 1ns / 1ps
module adder (s, cout, a, b, cin);
input [3:0] a,b;
input cin;
output [3:0] s;
output cout;
wire c1, c2, c3;
fa L1 (s[0], c1, a[0], b[0], cin);
fa L2 (s[1], c2, a[1], b[1], c1);
fa L3 (s[2], c3, a[2], b[2], c2);
fa L4 (s[3], cout, a[3], b[3], c3);
endmodule
module fa (s, co, a, b, ci);
input a, b, ci;
output s, co;
assign s = a^b^ci;
assign co = (a&b) | (a&ci) | (b&ci);
endmodule
```

Teacher's Signature :



Name of Experiment : 4-bit adder Date : 13/12/22

Experiment No : 6 Experiment Result : Verified Page No. 33

Test bench

```
'timescale 1ns/1ps
module adder_tb;
reg [3:0] a,b;
reg cin;
wire cout;
wire [3:0] s;
adder uut (s, cout, a, b, cin);
initial
begin
a = 4'b0101; b = 4'b1010; cin = 0; #100;
a = 4'b0101; b = 4'b1010; cin = 1; #100;
a = 4'b0111; b = 4'b1110; cin = 0; #100;
a = 4'b0111; b = 4'b1110; cin = 1; #100;
end
endmodule
```

Result :-

Implementation of 4-bit parallel adder is verified

25/25 A 13/12/22

Teacher's Signature : _____

Name of Experiment : 32-bit ALU Date : 13/12/22

Experiment No : 8 Experiment Result : Verified

32-bit ALU

Truth Table

Opcode	Res
000	a+b
001	a-b
010	a/b
011	a*b
100	a|b
101	a&b
110	~(a|b)
111	~(a&b)

Teacher's Signature : _____

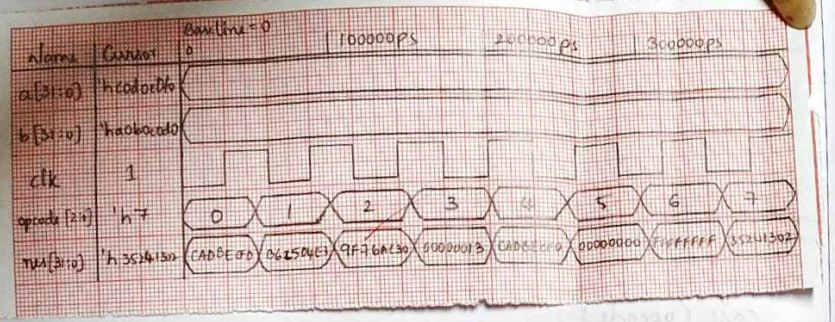
Name of Experiment : 32-bit ALU Date : 13/12/22

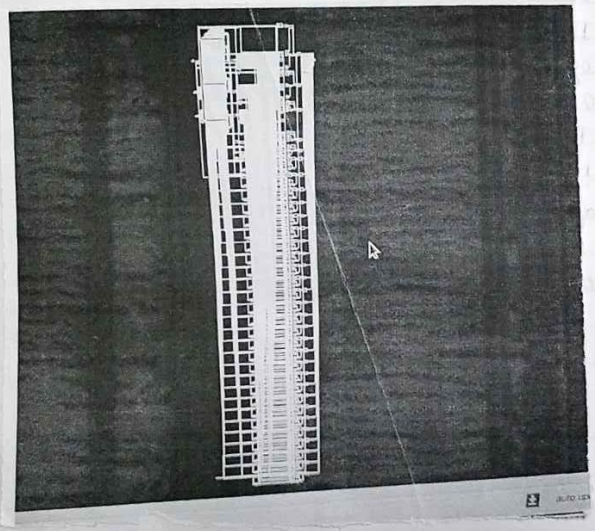
Experiment No : 8 Experiment Result : Verified Page No. 35

Code using CASE Statement

```
`timescale 1ns/1ps
module alu (res, a, b, opcode, clk);
input [31:0] a, b;
input [2:0] opcode;
input clk;
output reg [31:0] res;
always @ (posedge clk)
begin
case (opcode)
3'b000 : res = a+b;
3'b001 : res = a-b;
3'b010 : res = a*b;
3'b011 : res = a/b;
3'b100 : res = a|b;
3'b101 : res = a&b;
3'b110 : res = ~(a&b);
3'b111 : res = ~(a|b);
endcase
end
endmodule
```

Teacher's Signature : _____





Name of [illegible] Date : 13/12/22

Experiment No : 8 Experiment Result : Verified Page No. 36

Test bench;

```
'timescale 1ns/1ps
module alu_tb;
reg [31:0] a,b;
reg [2:0] opcode;
reg clk;
wire [31:0] res;
alu uut (res, a,b, opcode, clk);
initial clk = 1'b0;
always
#5 clk = ~clk;
initial
 begin
a = 32'hc0d0e0f0;
b = 32'h0a0b0c0d;
opcode = 3'b000; #40;
opcode = 3'b001; #40;
opcode = 3'b010; #40;
opcode = 3'b011; #40;
opcode = 3'b100; #40;
opcode = 3'b101; #40;
opcode = 3'b110; #40;
opcode = 3'b111; #40;
$finish
end
endmodule
```

Teacher's Signature : _____

Name of Experiment : 32-bit ALU Date : 13/12/22

Experiment No : 8 Experiment Result : Verified Page No. 37

Code using IF Statement

```
`timescale 1ns / 1ps
module alu (res, a, b, opcode, clk);
input [31:0] a,b;
input [2:0] opcode;
input clk;
output reg [31:0] res;
always @ (posedge clk)
begin
if (opcode == 3'b000)
res = a+b;
else if (opcode == 3'b001)
res = a-b;
else if (opcode == 3'b010)
res = a*b;
else if (opcode == 3'b011)
res = a/b;
else if (opcode == 3'b100)
res = a|b;
else if (opcode == 3'b101)
res = a&b;
else if (opcode == 3'b110)
res = ~(a&b);
else if (opcode == 3'b111)
res = ~(a|b);
end
endmodule
```

Teacher's Signature : _____

Name of Experiment : D-flipflop Date : 20/12/22

Experiment No : 9 Experiment Result : Verified

Flipflops and Latches

D-flipflops

Truth Table:-

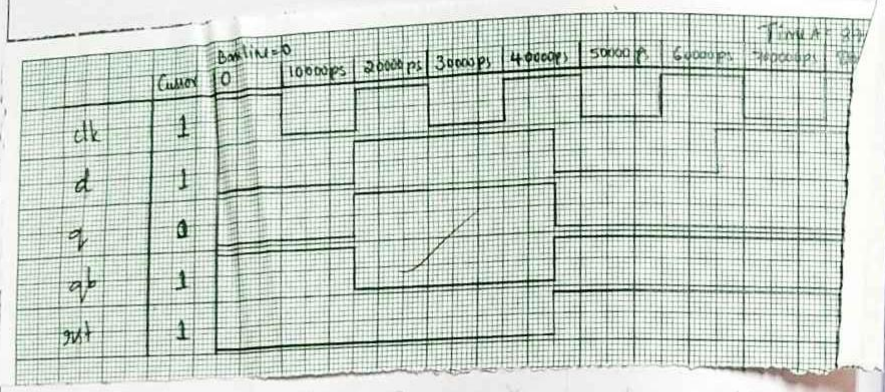
Inputs			Outputs	
clk	rst	D	q	qb
↑	0	0	0	1
↑	0	1	1	0
X	1	X	0	1

Code :

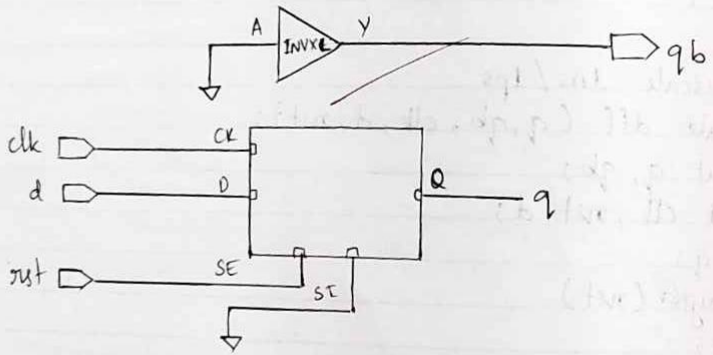
```
'timescale 1ns / 1ps
module dff (q, qb, clk, d, rst);
output q, qb;
input clk, rst, d;
reg q;
always@ (rst)
q=0;
always@ (posedgeclk)
begin
if(rst == 0)
q=d;
end
assign qb = ~q;
```

Teacher's Signature :

Simulation waveform



Schematic



Name of Experiment : D-flipflop Date : 20/12/22

Experiment No : 9 Experiment Result : Verified


```
endmodule
```

Test bench

```
'timescale 1ns / 1ps
module dff_tb;
reg clk, rst, d;
wire q, qb;
dff n1 (q, qb, clk, d, rst);
initial
clk = 1'b1
always
#10 clk = ~clk;
initial
begin
rst = 1'b0;
d = 1'b0; #20;
d = 1'b1; #27;
rst = 1'b1;
d = 1'b0; #20;
d = 1'b1; #20;
$finish;
end
endmodule
```

Result: Implementation of D-flipflop is verified

Teacher's Signature : _____

Name of Experiment : SR-flip flop
Experiment No : 9
Date : 20|12|22
Experiment Result : Verified

SR - Flipflops

Truth Table

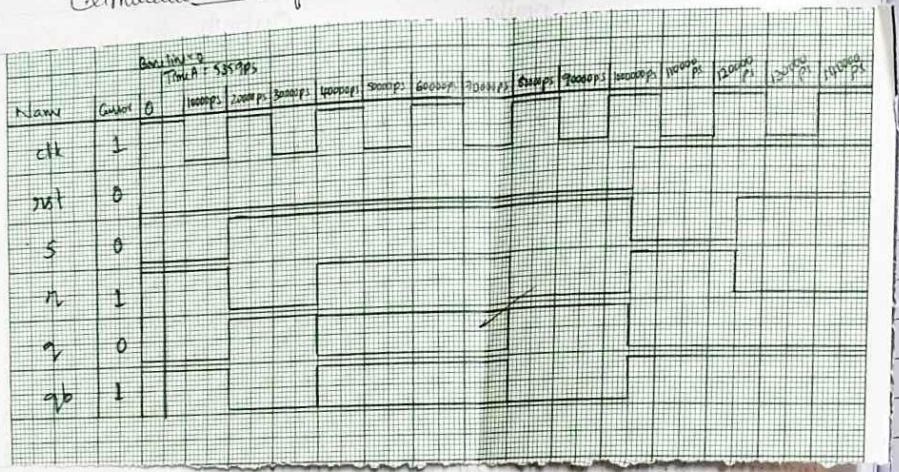
Inputs				Outputs	
clk	rst	S	R	q	qb
↑	0	0	0	No change	
↑	0	0	1	0	1
↑	0	1	0	1	0
↑	0	1	1	X	X
X	1	X	X	0	1

Code

```
'timescale 1ns/1ps
module srff (q, qb, clk, s, r, rst);
output q, qb;
input clk, rst, s, r;
reg q;
always@ (rst)
q=0;
always@ (posedge clk)
begin
if (!rst)
begin
if (s==1'b0 & r==1'b0)
q=q;
else if (s==1'b0 & r==1'b1)
```

Teacher's Signature : _____

Simulation waveform



Name of Experiment : SR flip-flop
Experiment No : 9
Date : 20/12/22
Experiment Result : Verified
Page No. 42

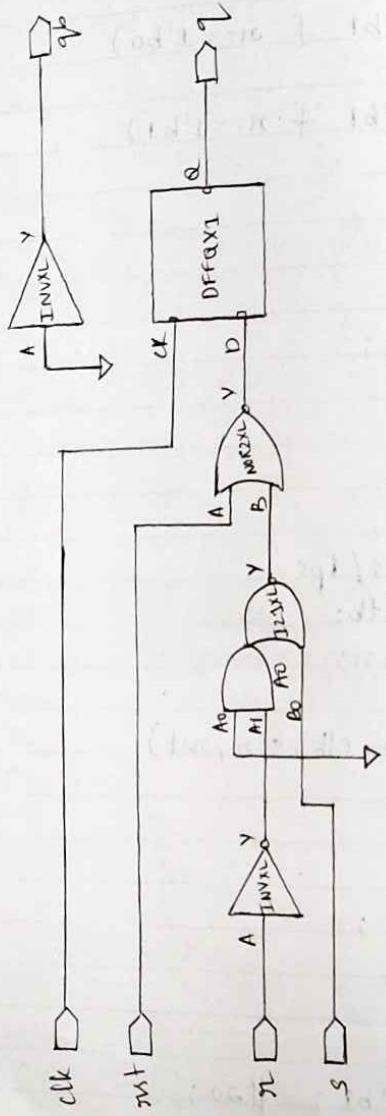
```
    q = 1'b0;
    else if (s==1'b1 & r==1'b0)
    q = 1'b1;
    else if (s==1'b1 & r==1'b1)
    q = 1'bx;
    end
    else q=1'b0;
    end
    assign qb=~q;
    endmodule
```

Test bench

```
' timescale 1ns/1ps
module srff_tb;
reg clk, rst, s, r;
wire q, qb;
srff n1 (q, qb, clk, s, r, rst)
initial
clk = 1'b1;
always
#10 clk = ~clk;
initial
begin
rst = 1'b0;
s = 1'b0 ; r = 1'b1 ; #20;
s = 1'b1 ; r = 1'b0 ; #20;
s = 1'b1 ; r = 1'b1 ; #40;
```

Teacher's Signature :

Schematic



Name of Experiment : SR-flip flop
Experiment No : 9
Date : 20/12/22
Experiment Result : Verified


```
s = 1'b1 ; r = 1'b0 ; #23;
rst = 1'b1 ;
s = 1'b0 ; r = 1'b1 ; #20;
s = 1'b1 ; r = 1'b0 ; #20;
$finish;
end
endmodule
```

Result:

Implementation of SR-flip flop is verified

Teacher's Signature :

Name of Experiment : Jk flip flops Date : 20/12/22

Experiment No : 9 Experiment Result : Verified Page No. 44

Jk - Flipflops

Truth Table

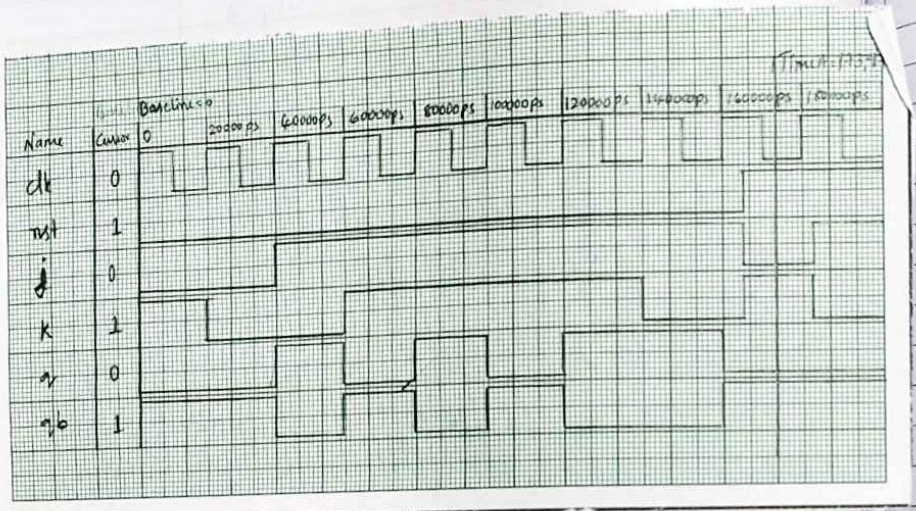
Inputh				Outputh	
clk	rst	J	K	q	qb
↑	0	0	0	No change	
↑	0	0	1	0	1
↑	0	1	0	1	0
↑	0	1	1	Toggle	
x	1	x	x	0	1

Code

```
'timescale 1ns / 1ps
module jkff (q, qb, clk, j, k, rst);
output q, qb;
input clk, rst, j, k;
reg q;
always@ (rst);
q=0;
always@ (posedge clk)
begin
if (!rst)
begin
if (j==1'b0 & k==1'b0)
q=q;
else if (j==1'b0 & k==1'b1)
```

Teacher's Signature :

Simulation waveform



Name of Experiment : JK flip flop
Experiment No : 9
Date : 20/12/22
Experiment Result : Verified

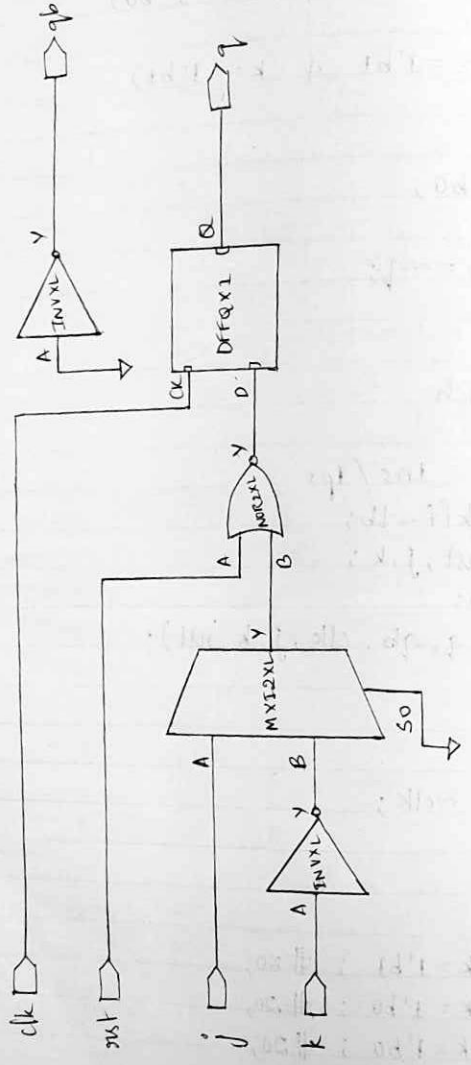

```
q = 1'b0;
else if (j == 1'b1 & k == 1'b0)
q = 1'b1;
else if (j == 1'b1 & k == 1'b1)
q = ~q;
end
else q = 1'b0;
end
assign qb = ~q;
endmodule
```

Test bench

```
'timescale 1ns/1ps
module jkff_tb;
reg clk, rst, j, k;
wire q, qb;
jkff n1 (q, qb, clk, j, k, rst);
initial
clk = 1'b1;
always
#10 clk = ~clk;
initial
begin
rst = 1'b0;
j = 1'b0 ; k = 1'b1 ; #20;
j = 1'b0 ; k = 1'b0 ; #20;
j = 1'b1 ; k = 1'b0 ; #20;
```

Teacher's Signature : _____

Schematic



Name of Experiment : JK flip-flop
Experiment No : 9
Date : 20/12/2022
Experiment Result : Verified
Page No. 46

```
j = 1'b1 ; k = 1'b1 ; #80;
j = 1'b1 ; k = 1'b0 ; #23;

rst = 1'b1
j = 1'b0 ; k = 1'b1; #20;
j = 1'b1 ; k = 1'b0; #20;
$finish
end
endmodule
```

Result :-

Implementation of JK flip-flop is verified

Teacher's Signature : _____

Name of Experiment : D- latch Date : 20/12/22

Experiment No : 9 Experiment Result : Verified Page No. 47

D - Latch

Truth Table

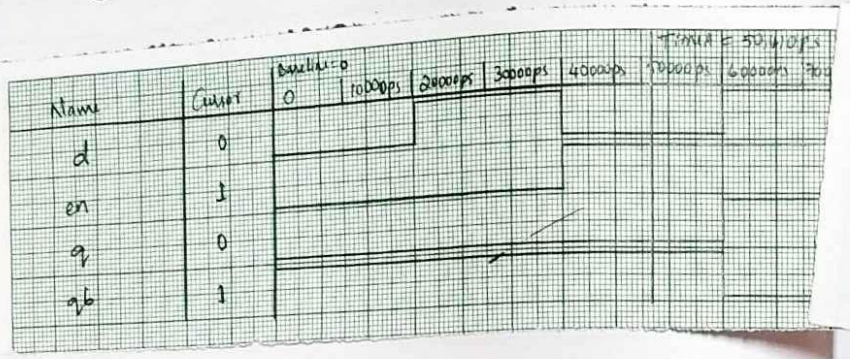
Inputs		Outputs	
en	D	q	qb
1	0	0	1
1	1	1	0
0	x	0	1

Code :

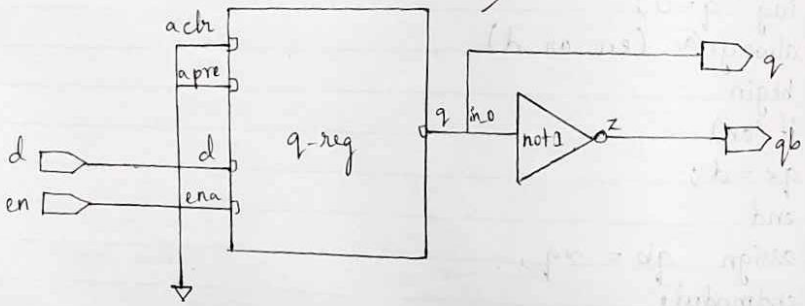
```
`timescale 1ns / 1ps
module dl (q, qb, d, en);
input en, d;
output q, qb;
reg q=0;
always@ (en or d)
begin
if (en)
q<=d;
end
assign qb = ~q;
endmodule
```

Teacher's Signature : _____

Simulation waveform



Schematic



Name of Experiment : D-latch
Experiment No : 9
Date : 20/12/22
Experiment Result : Verified
Page No. 48

Test bench

```
`timescale 1ns / 1ps
module dl_tb;
reg en, d;
wire q;
dl uut (q, qb, d, en);
initial
begin
en = 1'b0;
d = 1'b0; #20; d = 1'b1; #20;
en = 1'b1;
d = 1'b0; #20; d = 1'b1; #20;
end
endmodule
```

Result:

Implementation of D-latch is verified

Name of Experiment : SR latch Date : 20/12/22

Experiment No : 9 Experiment Result : Verified

SR - Latch

Truth Table

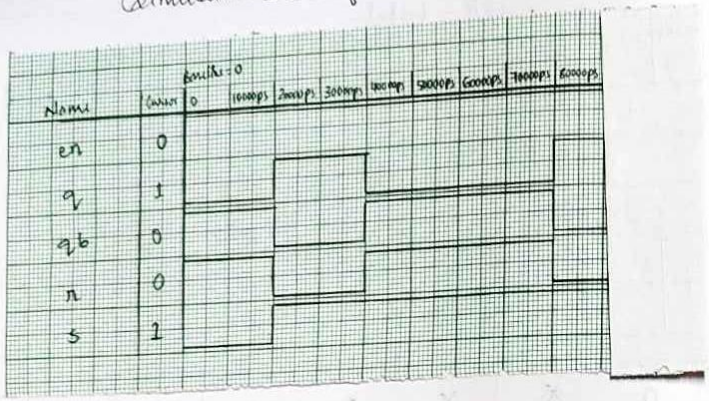
Inputh			Outputh	
en	S	R	q	qb
1	0	0	No change	
1	0	1	1	0
1	1	0	0	1
1	1	1	Undetermined	
0	X	X	0	1

Code :-

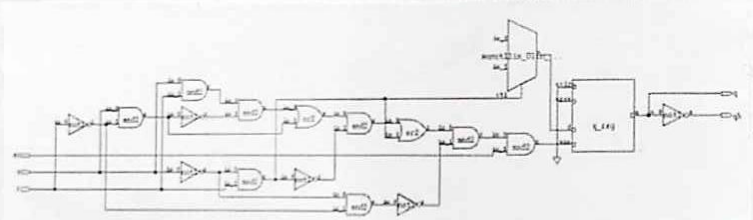
```
'timescale 1ns/1ps
module srl (q, qb, s, r, en);
input en, s, r;
output q, qb;
reg q=0;
always@ (en or s or r)
begin
if (en)
if (s==1'b0 & r==1'b0)
q=q;
else if (s==1'b0 & r==1'b1)
q=1'b0;
else if (s==1'b1 & r==1'b0)
q=1'b1;
```

Teacher's Signature : _____

Simulation waveform



Schematic



Name of Experiment : SR latch Date : 20/12/22

Experiment No : 9 Experiment Result : Verified Page No. 50

```
else if (s==1'b1 & r==1'b1)
q = 1'bX;
end
assign qb = ~q;
endmodule
```

Test bench

```
'timescale 1ns/1ps
module srff_tb;
reg en, s, r;
wire q, qb;
srl n1 (q, qb, s, r, en);
initial
begin
en = 1'b1;
s = 1'b0 ; r=1'b1 ; #20 ;
s = 1'b1 ; r=1'b0; #20 ;
s = 1'b1 ; r=1'b1; #40 ;
s = 1'b1 ; r=1'b0; #23 ;
en = 1'b0 ;
s = 1'b0 ; r=1'b1 ; #20;
s = 1'b1 ; r=1'b0 ; #20;
end
endmodule
```

Result: Implementation of SR-latch is verified

Teacher's Signature : _____

Name of Experiment : JK latch Date : 20/12/22

Experiment No : 9 Experiment Result : Verified Page No. 51

JK- Latch

Truth Table

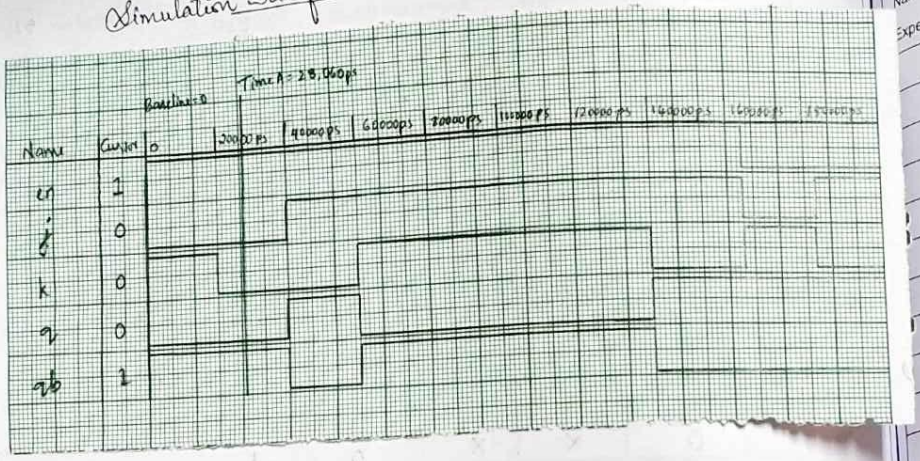
Inputs			Outputs	
en	J	K	q	qb
1	0	0	No change	
1	0	1	1	0
1	1	0	0	1
1	1	1	Toggle	
0	x	x	0	1

Code :-

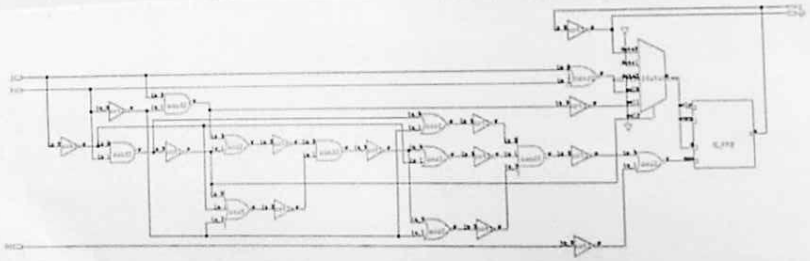
```
'timescale 1ns / 1ps
module jkl (q, qb, j, k, en);
input en, j, k;
output q, qb;
reg q=0;
always@ (en or j or k)
begin
if (en)
if (j== 1'b0 & k== 1'b0)
q=q;
else if ( j== 1'b0 & k== 1'b1)
q= 1'b0;
else if (j== 1'b1 & k== 1'b0)
q=1'b1;
```

Teacher's Signature : _____

Simulation waveform



Schematic



Name of Experiment : JK latch
Experiment No : 9
Date : 20/12/22
Experiment Result : Verified
Page No. 52

```
else if (j==1'b1 & k==1'b1)
q=~q;
end
assign qb=~q;
endmodule
```

Test bench

```
'timescale 1ns/1ps
module jkff_tb;
reg en,j,k;
wire q,qb;
jkl uut (q,qb,j,k,en);
initial
begin
en=1'b1;
j=1'b0; k=1'b1; #20;
j=1'b0; k=1'b0; #20;
j=1'b1; k=1'b0; #20;
j=1'b1; k=1'b1; #80;
j=1'b1; k=1'b0; #23;
en=0;
j=1'b0; k=1'b1; #20;
j=1'b1; k=1'b0; #20;
end
endmodule
```

Result: Implementation of JK latch is verified.

25/25

Teacher's Signature :