Expt. No. 7 - Design and synthesis of a sequence generator using negative edge triggered JK flip flop for sequence of digits in your mobile number	
NAME : S. Chandra Moulee	**Date :** 03/06/2022
ROLL No.: CB.EN.P2VLD21016	**Marks :** **out of 10**

AIM:

To Design and synthesis a sequence generator using negative edge triggered JK flip flop for sequence of digits in your mobile number. [if any digit repeats consider it only once - the first occurance] Use BSAYS 3 FPGA, the output should be displayed on the seven segment display. Demonstration to be shown before the next lab.

PROCEDURE:

Excitation table to obtain the following sequence [1000 - 0111 - 0101 - 0100 - 0001 - 1001 - 0011] using JK flip flops

Qd	Qc	Qb	Qa	Qd +1	Qc +1	Qb +1	Qa +1	Jd	Kd	Jc	Kc	Jb	Kb	Ja	Ka
0	0	0	0	1	0	0	0	1	x	0	x	0	x	0	x
0	0	0	1	1	0	0	1	1	x	0	x	0	x	x	0
0	0	1	0	1	0	0	0	1	x	0	x	x	1	0	x
0	0	1	1	1	0	0	0	1	x	0	x	x	1	x	1
0	1	0	0	0	0	0	1	0	x	x	1	0	x	1	x
0	1	0	1	0	1	0	0	0	x	x	1	0	x	x	1
0	1	1	0	1	0	0	0	1	x	x	1	x	1	0	x
0	1	1	1	0	1	0	1	0	x	x	0	x	1	x	0
1	0	0	0	0	1	1	1	x	1	1	x	1	x	1	x
1	0	0	1	0	0	1	1	x	1	0	x	1	x	x	0
1	0	1	0	1	0	0	0	x	0	0	x	x	1	0	x
1	0	1	1	1	0	0	0	x	0	0	x	x	1	x	1
1	1	0	0	1	0	0	0	x	0	x	1	0	x	0	x
1	1	0	1	1	0	0	0	x	0	x	1	0	x	x	1

1	1	1	0	1	0	0	0	x	0	x	1	x	1	0	x
1	1	1	1	1	0	0	0	x	0	x	1	x	1	x	1

Jd: Qcbar + Qb.Qabar

1	1	1	1
0	0	0	1
x	x	x	x
x	x	x	x

Kd: Qcbar.Qbbar

x	x	x	x
x	x	x	x
0	0	0	0
1	1	0	0

Jc: Qd.Qbbar.Qabar

0	0	0	0
x	x	x	x
x	x	x	x
1	0	0	0

Kc: Qd + Qabar

x	x	x	x
1	0	0	1
1	1	1	1
x	x	x	x

Jb: Qd.Qcbar

0	0	x	x
0	0	x	x

0	0	x	x
1	1	x	x

Kb: 1

x	x	1	1
x	x	1	1
x	x	1	1
x	x	1	1

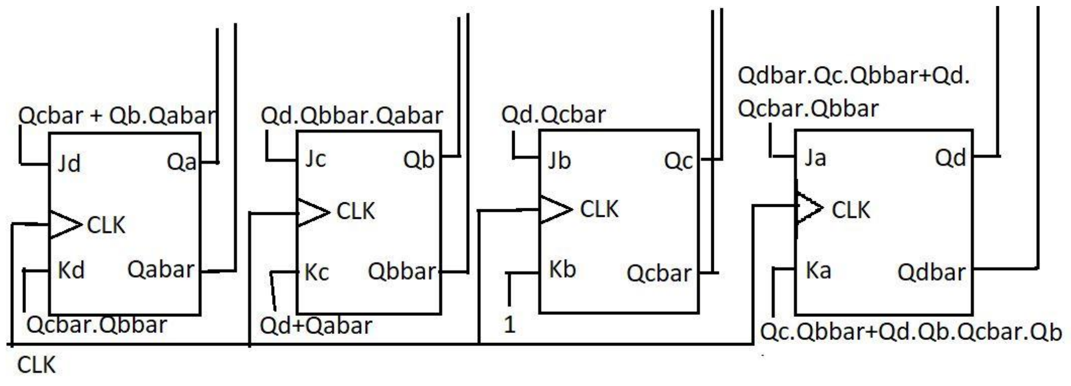
Ja: Qdbar.Qc.Qbbar + Qd.Qcbar.Qbbar

0	x	x	1
1	x	x	0
0	x	x	0
1	x	x	0

Ka: Qc.Qbbar + Qd.Qb + Qcbar.Qb

x	0	1	x
x	1	0	x
x	1	1	x
x	0	1	x

BLOCK/CIRCUIT DIAGRAM:

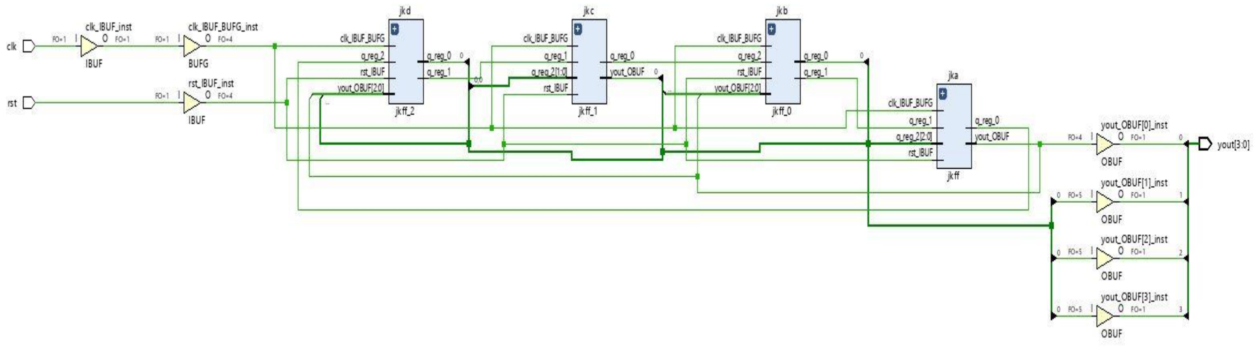


CODE:

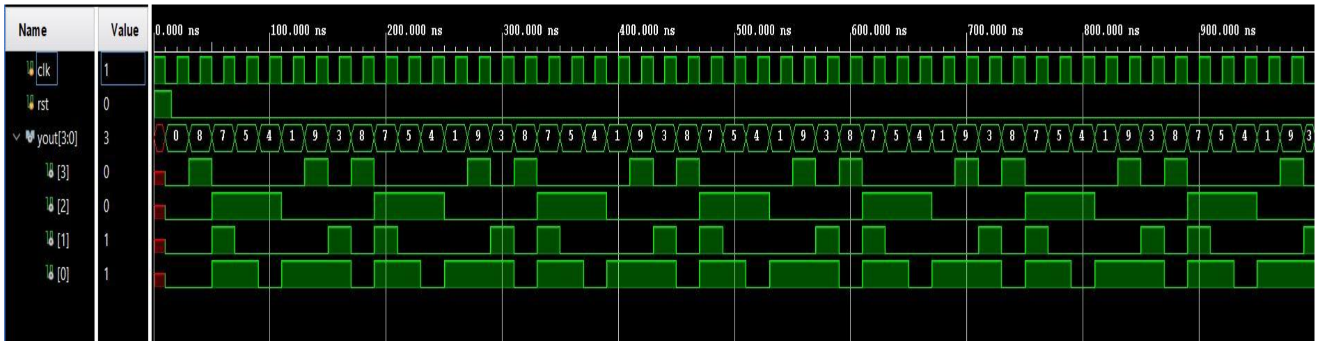
```
module jkff(clk, rst, j, k, q);
input clk, rst, j, k;
output reg q;
always @(negedge clk)
begin
if (rst==1'b1)
q=0;
else
begin
case({j,k})
2'b00: q<=q;
2'b01: q<=1'b0;
2'b10: q<=1'b1;
2'b11: q<=~q;
endcase
end
end
endmodule

module mobile(clk, rst, yout);
input clk, rst;
wire [3:0]j,k;
wire [3:0] q;
output [3:0] yout;
assign j[3]=(~q[2] | (q[1]&~q[0]));
assign k[3]=(~q[2]&~q[1]);
assign j[2]=(q[3] & ~q[1] & ~q[0] );
assign k[2]=(q[3] | ~q[0]);
assign j[1]=(q[3] & ~q[2]);
assign k[1]=(1'b1);
assign j[0]=((~q[3]&q[2]&~q[1]) | (q[3]&~q[2]&~q[1]));
assign k[0]=((q[2]&~q[1]) | (q[3]&q[1]) | (~q[2]&q[1]));

jkff jka(clk, rst, j[0], k[0], q[0]);
jkff jkb(clk, rst, j[1], k[1], q[1]);
jkff jkc(clk, rst, j[2], k[2], q[2]);
jkff jkd(clk, rst, j[3], k[3], q[3]);

assign yout[3]=q[3];
assign yout[2]=q[2];
assign yout[1]=q[1];
assign yout[0]=q[0];
endmodule

//////////////////////////////////////////////////////////////////////////////////////////////////////////
module tb_mobile_number();
reg clk, rst;
wire [3:0]yout;
always
begin
clk=1'b1;
#10;
clk=1'b0;
#10;
end
initial
begin
rst=1'b1;
#15;
```

```
rst=1'b0;
end
mobile inst0(.clk(clk), .rst(rst), .yout(yout));
endmodule
```

SCHEMATIC DIAGRAM:



OUTPUT:



INFERENCE:

- Framed an excitation table for next states starting from the first number
- Based on the table, input values for the J, K of the flip flop equations were formed
- Created a single JK flip flop module and made four instantiations to create 4 flip flops
- Flip flops were excited with the J and K values obtained from the next state table
- Triggered the flip flops with negedge clock and obtained the desired sequence in Vivado tool