SIMULATION AND IMPLEMENTATION OF SEQUENTIAL LOGIC CIRCUITS

AIM: To simulate and synthesis SR, JK, T, D - FLIPFLOP, COUNTER DESIGN using Xilinx ISE.

APPARATUS REQUIRED:

Xilinx 14.7 Spartan6 FPGA

PROCEDURE:

STEP:1 Start the Xilinx navigator, Select and Name the New project.

STEP:2 Select the device family, device, package and speed.
STEP:3 Select new source in the New Project and select Verilog Module as the Source type.
STEP:4 Type the File Name and Click Next and then finish button. Type the code and save it. STEP:5 Select the Behavioral Simulation in the Source Window and click the check syntax.
STEP:6 Click the simulation to simulate the program and give the inputs and verify the outputs as per the truth table.
STEP:7 Select the Implementation in the Sources Window and select the required file in the Processes Window.

STEP:8 Select Check Syntax from the Synthesize XST Process. Double Click in the FloorplanArea/IO/Logic-Post Synthesis process in the User Constraints process group. UCF(User constraint File) is obtained.

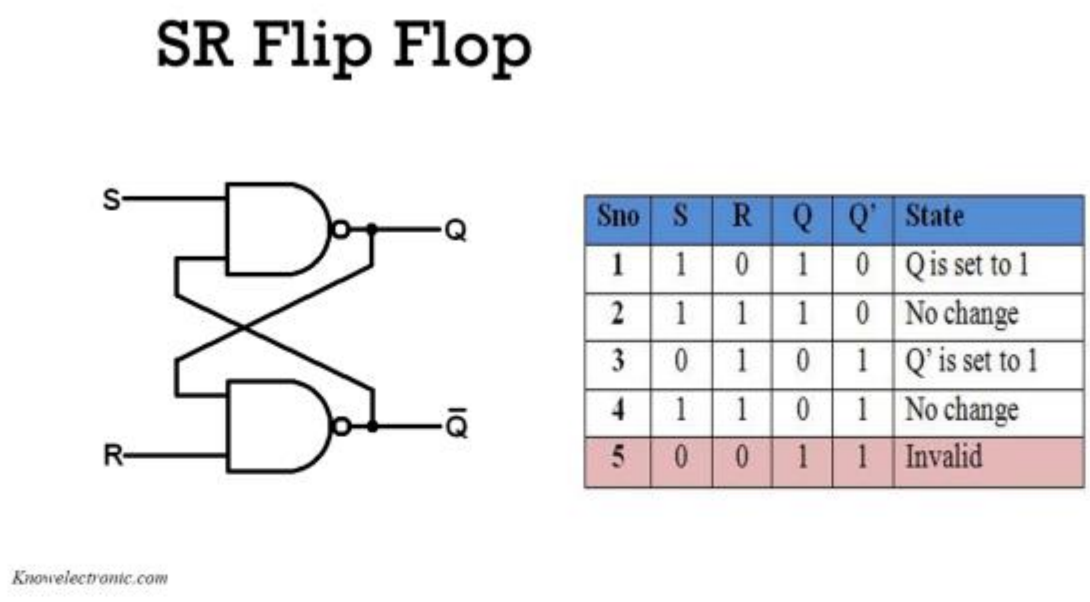
STEP:9 In the Design Object List Window, enter the pin location for each pin in the Loc column Select save from the File menu.

STEP:10 Double click on the Implement Design and double click on the Generate Programming File to create a bitstream of the design.(.v) file is converted into .bit file here.

STEP:11 On the board, by giving required input, the LEDs starts to glow light, indicating the output.

LOGIC DIAGRAM

SR FLIPFLOP:



SR Flip Flop

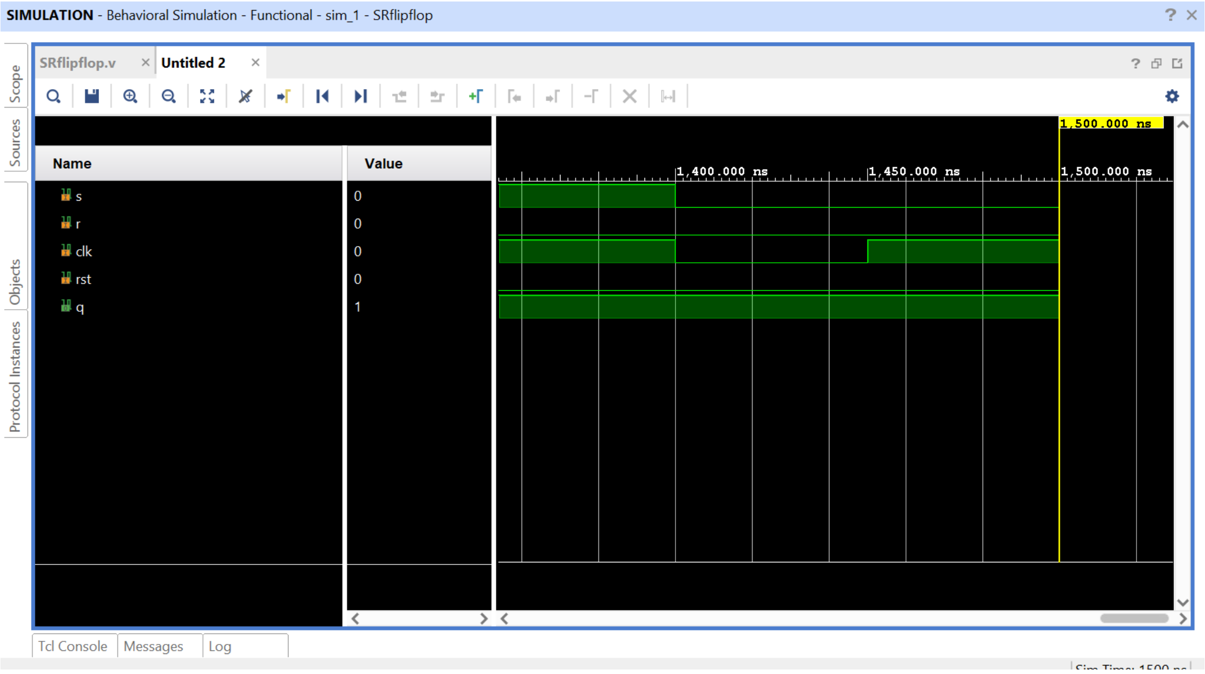
Sno	S	R	Q	Q'	State
1	1	0	1	0	Q is set to 1
2	1	1	1	0	No change
3	0	1	0	1	Q' is set to 1
4	1	1	0	1	No change
5	0	0	1	1	Invalid

Knowelectronic.com

VERILOG CODE

```
module SRflipflop(s,r,clk,rst,q);
input s,r,clk,rst;
output q;
reg q;
always@(posedge clk)
begin
 if(rst)
 q<=1'b0;
 else
  begin
    case({s,r})
     2'b00:q<=q;
     2'b01:q<=1'b0;
     2'b10:q<=1'b1;
     2'b11:q<=1'bx;
    default:q<=1'bx;
    endcase
    end
   end
 endmodule
```

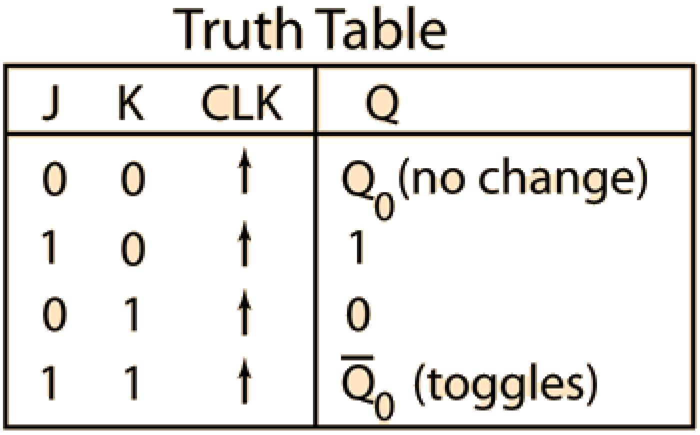
OUTPUT WAVEFORM



SIMULATION - Behavioral Simulation - Functional - sim_1 - SRflipflop

Name	Value
s	0
r	0
clk	0
rst	0
q	1

JK FLIPFLOP:



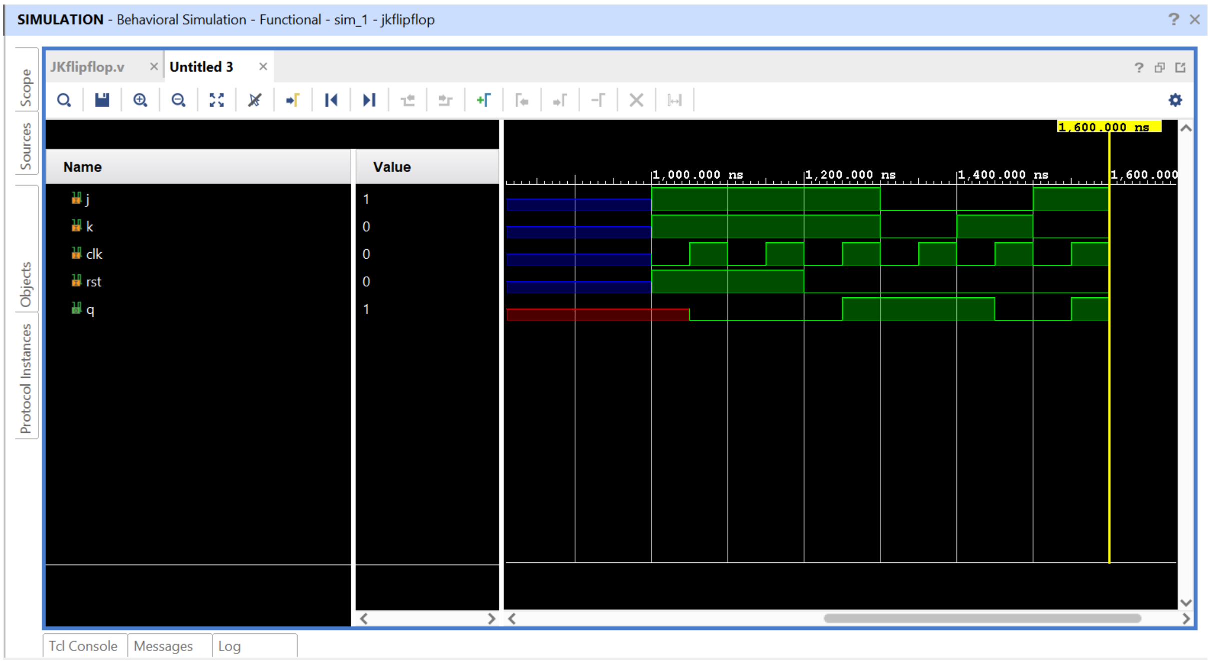
Truth Table

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

VERILOG CODE

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

OUTPUT WAVEFORM

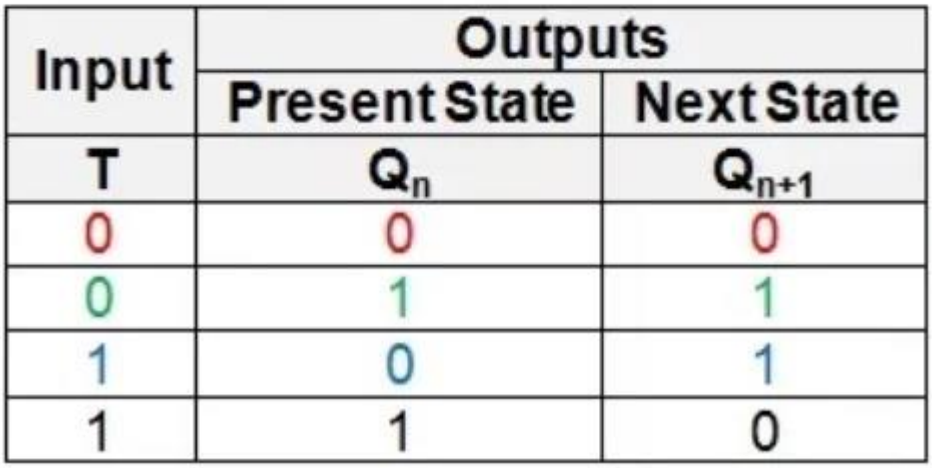


SIMULATION - Behavioral Simulation - Functional - sim_1 - jkflipflop

JKflipflop.v | Untitled 3

Name	Value
j	1
k	0
clk	0
rst	0
q	1

T FLIPFLOP:



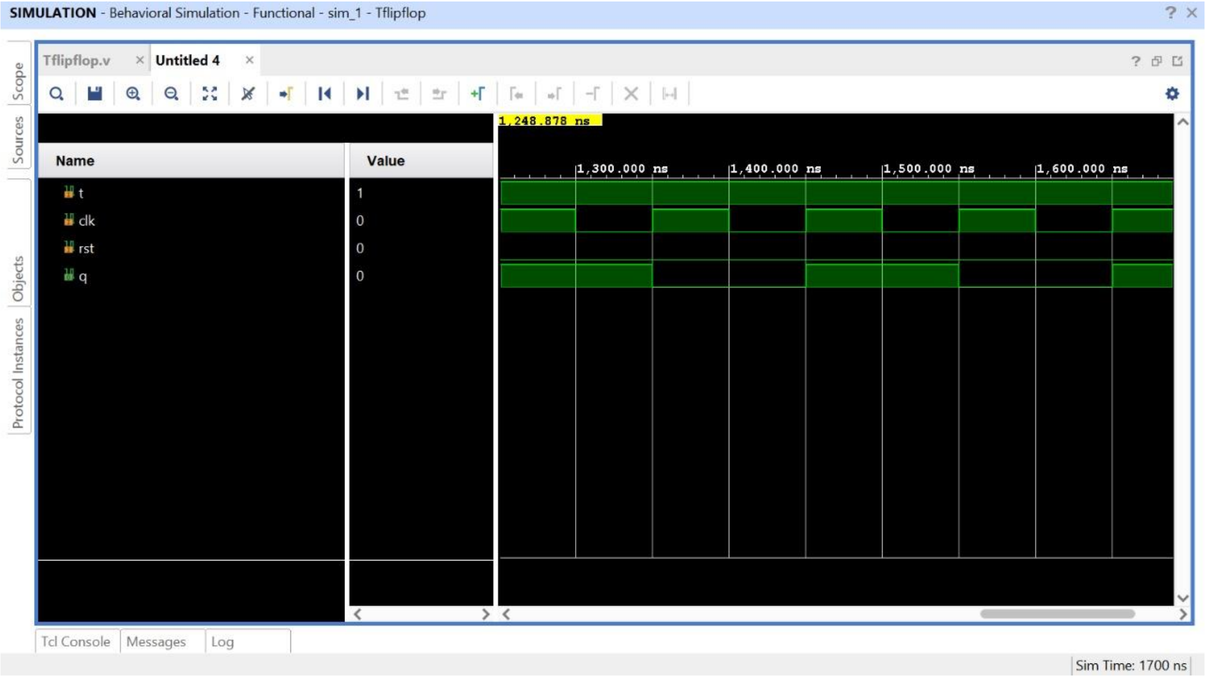
Input	Outputs	
	Present State	Next State
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

VERILOG CODE

```
module Tflipflop(t,clk,rst,q);
input t,clk,rst;
output q;
```

```
reg q;
always@(posedge clk)
begin
 if(rst)
 q<=0;
 else if(t)
 q<=~q;
 else
 q<=q;
end
endmodule
```

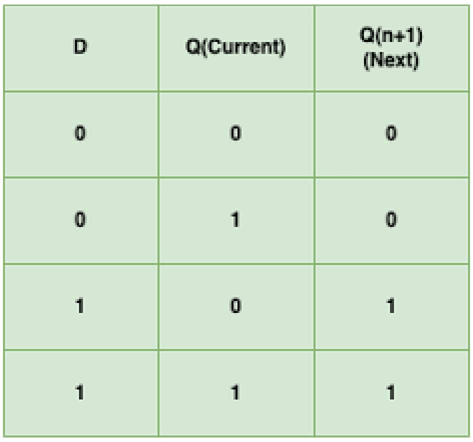
OUTPUT WAVEFORM



SIMULATION - Behavioral Simulation - Functional - sim_1 - Tflipflop

Name	Value
t	1
clk	0
rst	0
q	0

D FLIPFLOP:

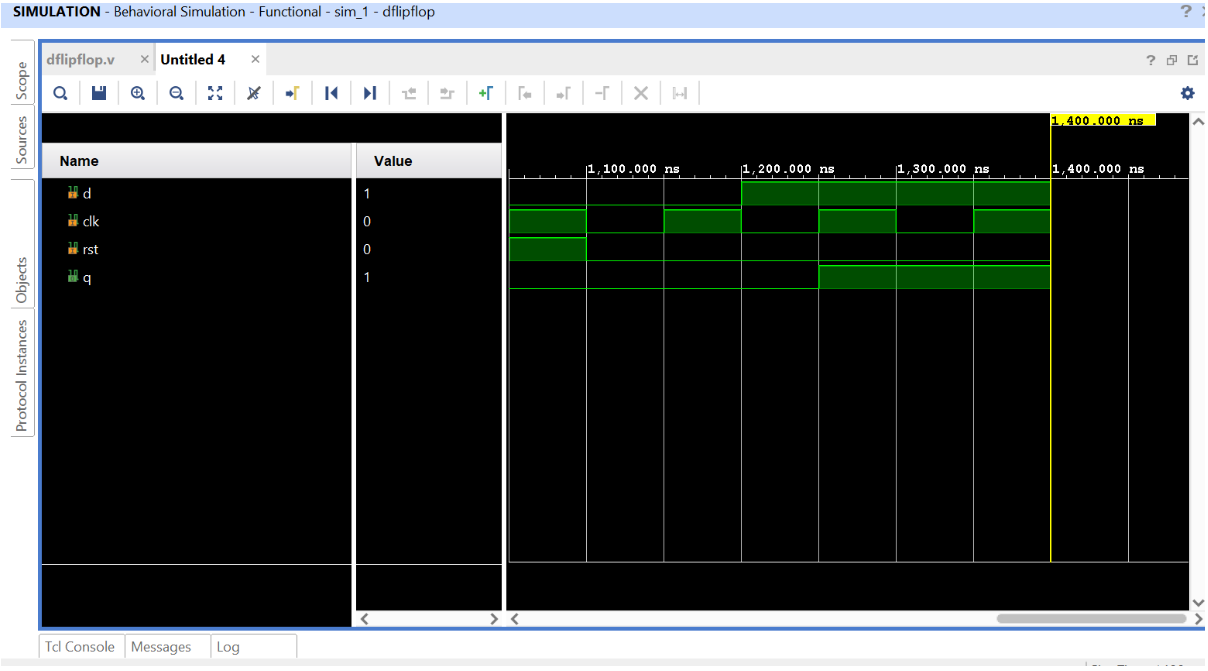


D	Q(Current)	Q(n+1) (Next)
0	0	0
0	1	0
1	0	1
1	1	1

VERILOG CODE

```
module dflipflop(d,clk,rst,q);
input d,clk,rst;
output reg q;
always @(posedge clk)
begin
if(rst)
q <=1'b0;
else
q <= d;
end
endmodule
```

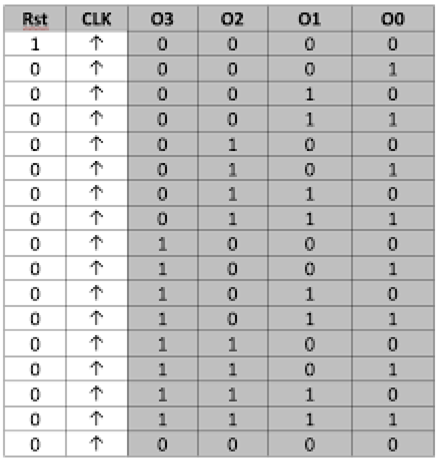
OUTPUT WAVEFORM



SIMULATION - Behavioral Simulation - Functional - sim_1 - dflipflop

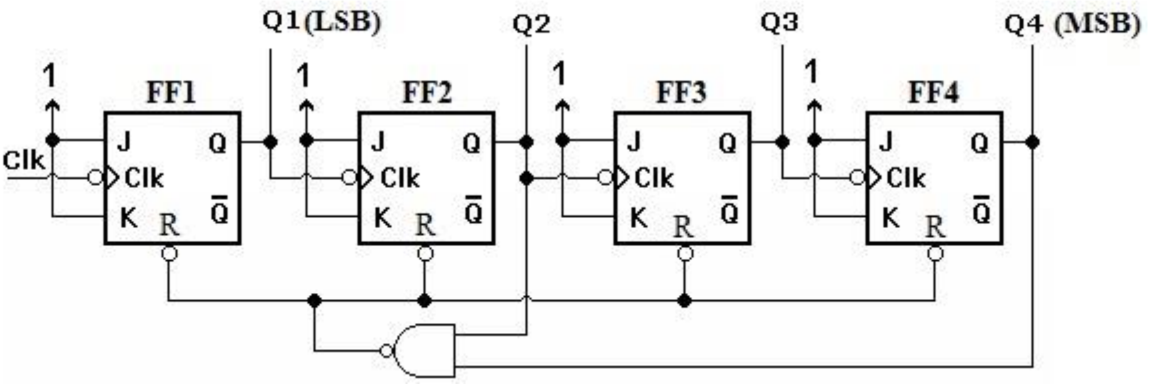
Name	Value
d	1
clk	0
rst	0
q	1

COUNTER



Rst	CLK	O3	O2	O1	O0
1	↑	0	0	0	0
0	↑	0	0	0	1
0	↑	0	0	1	0
0	↑	0	0	1	1
0	↑	0	1	0	0
0	↑	0	1	0	1
0	↑	0	1	1	0
0	↑	0	1	1	1
0	↑	1	0	0	0
0	↑	1	0	0	1
0	↑	1	0	1	0
0	↑	1	0	1	1
0	↑	1	1	0	0
0	↑	1	1	0	1
0	↑	1	1	1	0
0	↑	1	1	1	1
0	↑	0	0	0	0

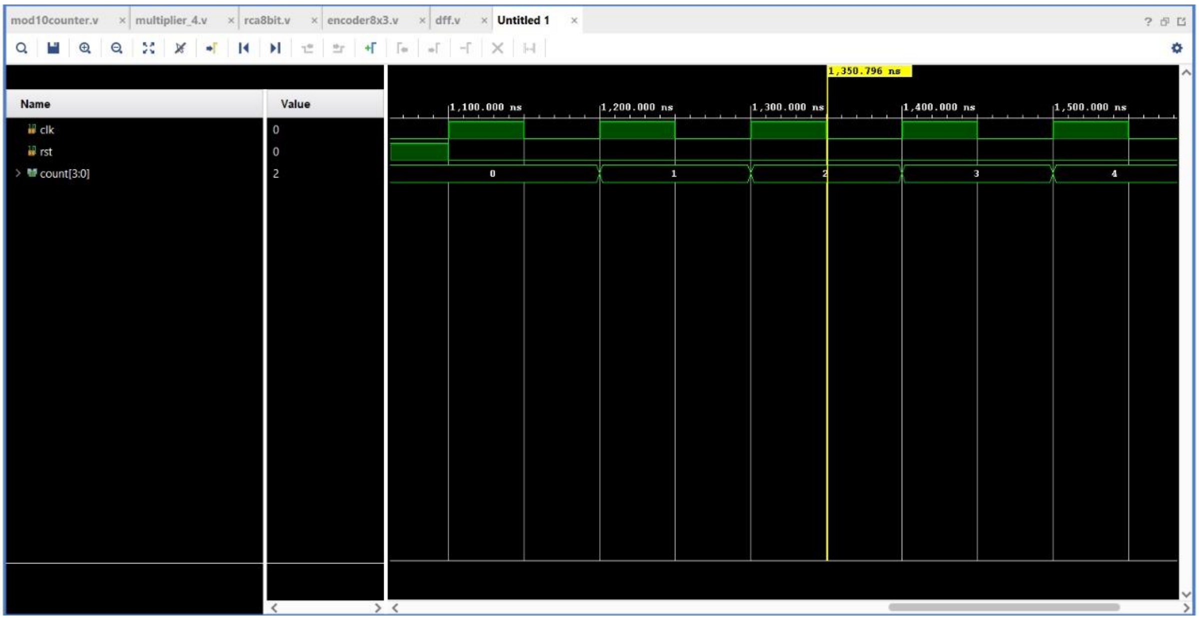
MOD10 COUNTER:



VERILOG CODE

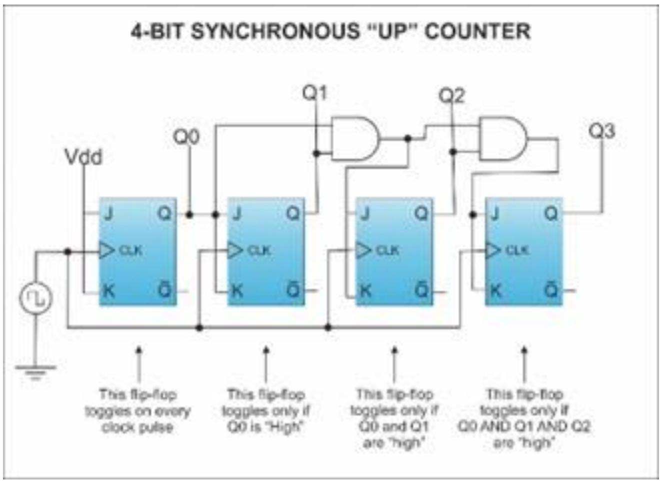
```
module mod10(clk,rst,count);
input clk,rst;
output[3:0]count;
reg[3:0]count;
always@(posedge clk)
begin
if(rst|count==4'b1001)
count<=4'b0;
else
count<=count+1;
end
endmodule
```

OUTPUT WAVEFORM



Name	Value
clk	0
rst	0
count[3:0]	2

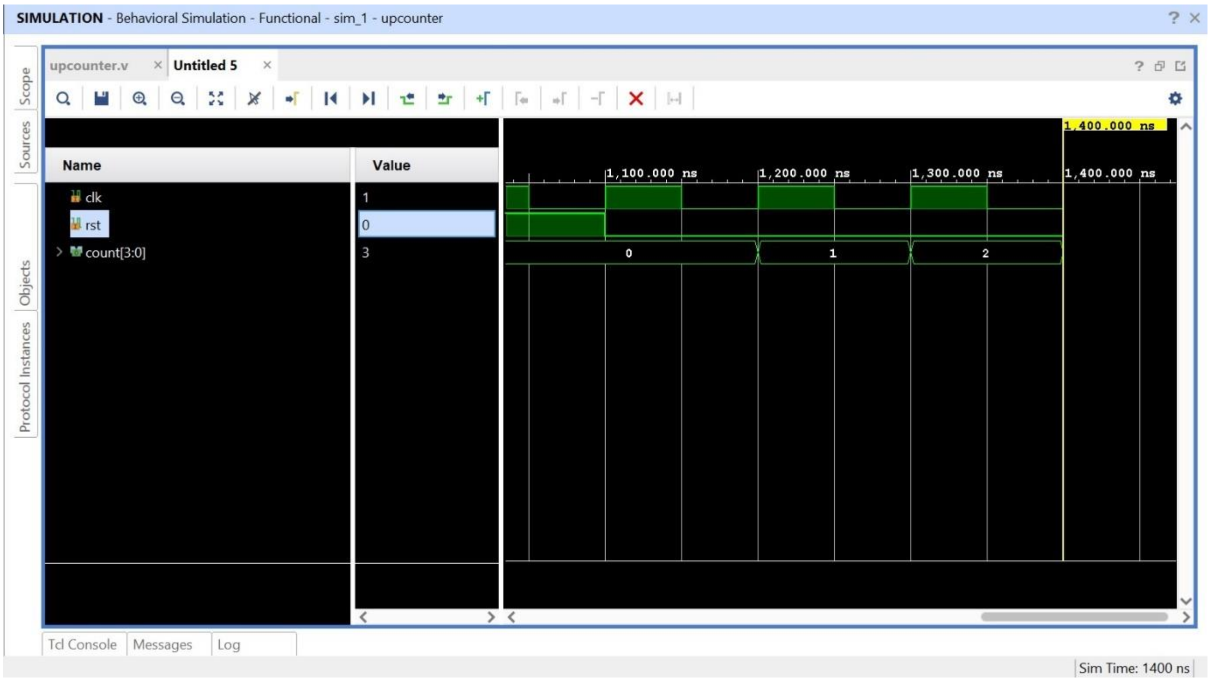
UP COUNTER:


4-BIT SYNCHRONOUS "UP" COUNTER
Q1
Q2
Q0
Q3
Vdd
J
Q
CLK
K
Q̄
This flip-flop toggles on every clock pulse
This flip-flop toggles only if Q0 is "High"
This flip-flop toggles only if Q0 and Q1 are "high"
This flip-flop toggles only if Q0 AND Q1 AND Q2 are "high"

VERILOG CODE

```
module upcounter(clk,rst,count);
input clk,rst;
output[3:0]count;
reg[3:0]count;
always@(posedge clk)
begin
if(rst)
count<=4'b0;
else
count<=count+1;
end
endmodule
```

OUTPUT WAVEFORM

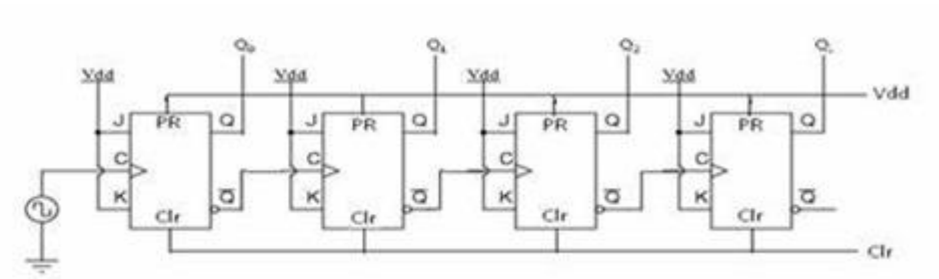


SIMULATION - Behavioral Simulation - Functional - sim_1 - upcounter

Name	Value
clk	1
rst	0
count[3:0]	3

Sim Time: 1400 ns

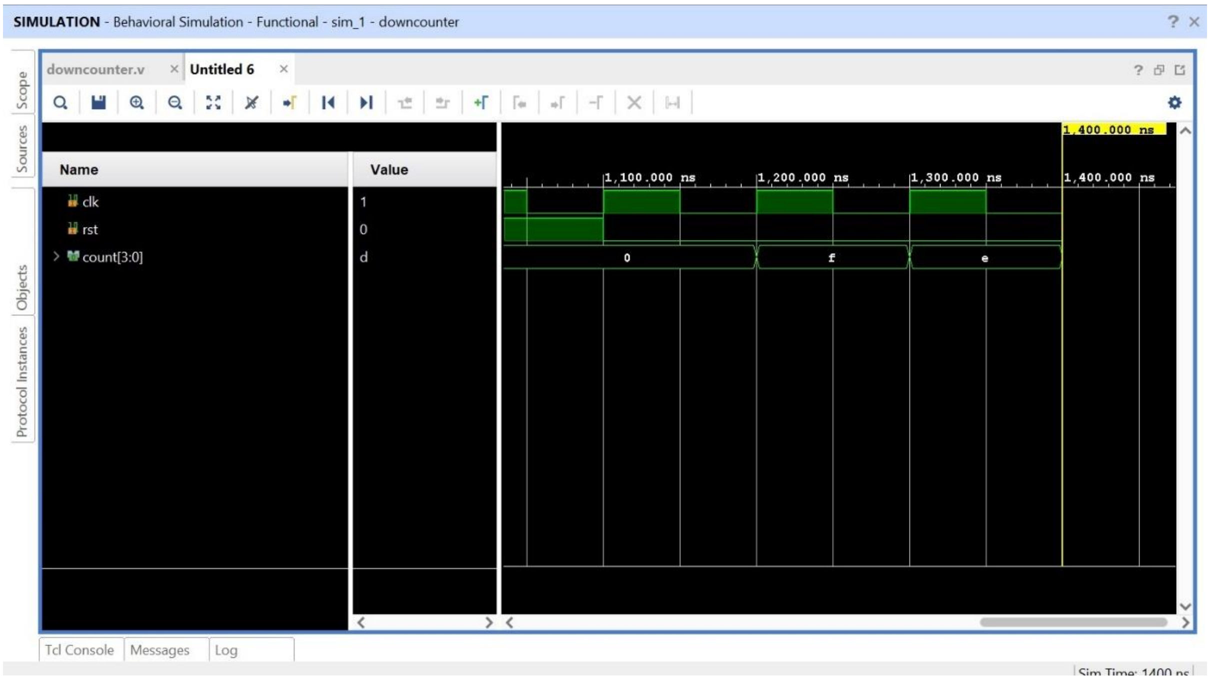
DOWN COUNTER:



VERILOG CODE

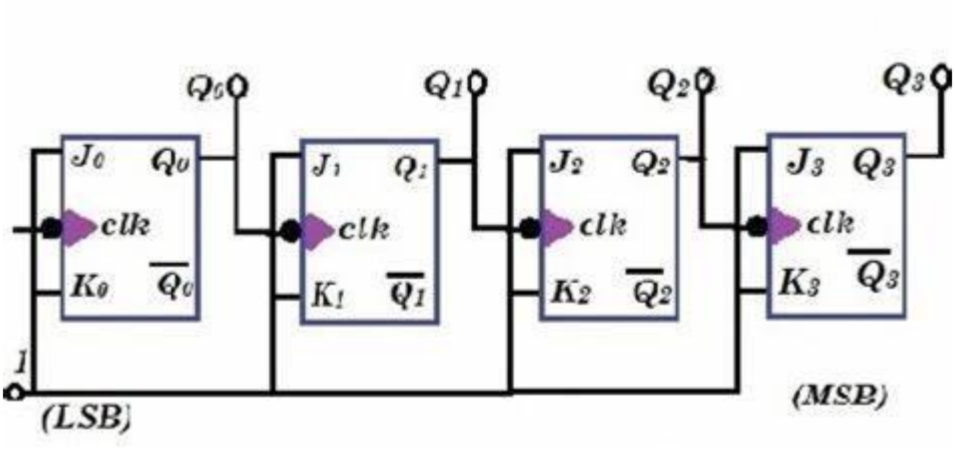
```
module downcounter(clk,rst,count);
input clk,rst;
output[3:0]count;
reg[3:0]count;
always@(posedge clk)
begin
if(rst)
count<=4'b0;
else
count<=count-1;
end
endmodule
```

OUTPUT WAVEFORM


SIMULATION - Behavioral Simulation - Functional - sim_1 - downcounter

Name	Value
clk	1
rst	0
count[3:0]	d

RIPPLECARRY COUNTER:

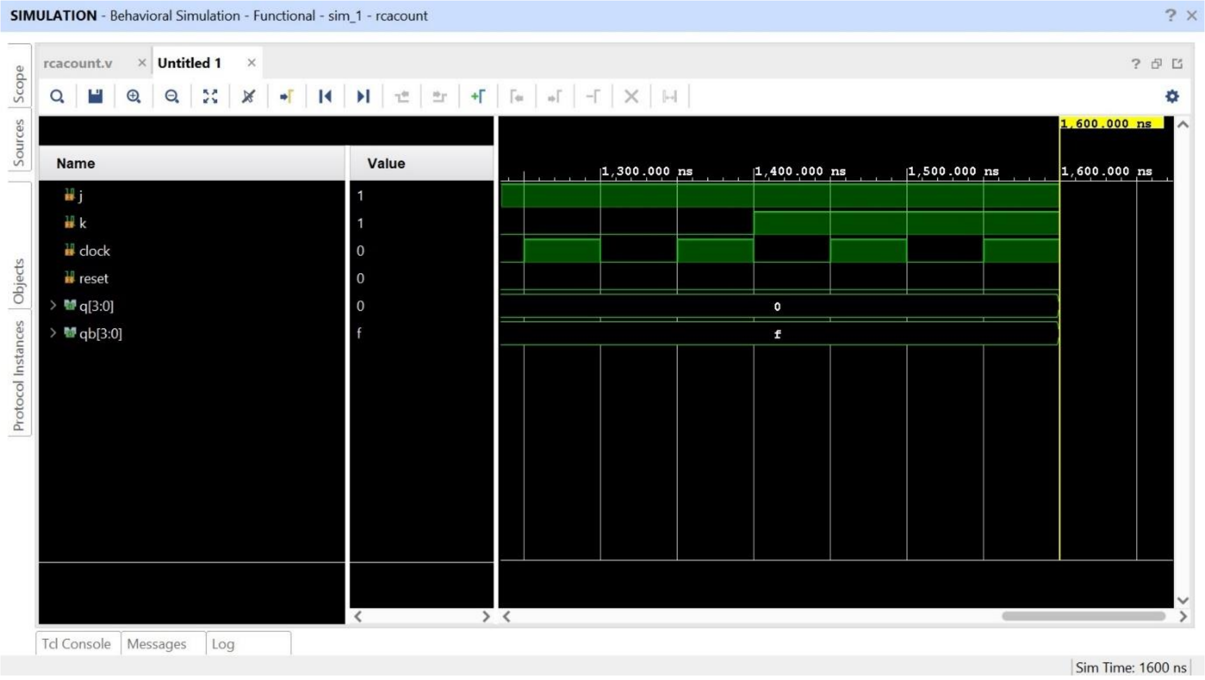


VERILOG CODE

```
module jkff(j,k,clock,reset,q,qb);
input j,k,clock,reset;
output reg q,qb;
always@(negedge clock)
begin
case({reset,j,k})
3'b100 :q=q;
3'b101 :q=0;
3'b110 :q=1;
3'b111 :q=~q;
default :q=0;
endcase
qb<=~q;
end
endmodule

module rcacount(j,k,clock,reset,q,qb);
input j,k,clock,reset;
output wire [3:0]q,qb;
jkff JK1(j,k,clock,reset,q[0],qb[0]);
jkff JK2(j,k,q[0],reset,q[1],qb[1]);
jkff JK3(j,k,q[1],reset,q[2],qb[2]);
jkff JK4(j,k,q[2],reset,q[3],qb[3]);
endmodule
```

OUTPUT WAVEFORM



Name	Value
j	1
k	1
clock	0
reset	0
q[3:0]	0
qb[3:0]	f

Sim Time: 1600 ns

RESULT:

Thus the simulation and implementation of sequential logic circuits done and output is verified successfully.