SIMULATION AND IMPLEMENTATION SEQUENTIAL LOGIC CIRCUITS

AIM

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

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.

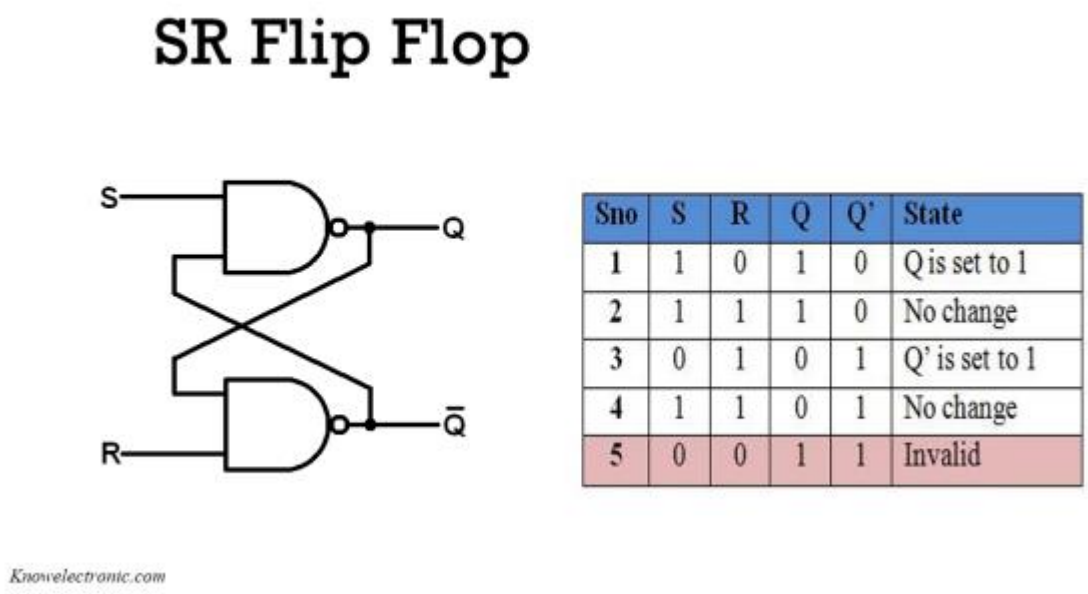
STEP:8 Select Check Syntax from the Synthesize XST Process. Double Click in the FloorplanArea/IO/Logic-Post Synthesis process in the User Constraints process group. UCF(User constraint File) is obtained.

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



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

VERILOG CODE

```
module sr_ff(clk,q,rst,s,r);
input s,r,clk,rst;
output reg q;
always@(posedge clk)
begin
if(rst==1)
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;
```

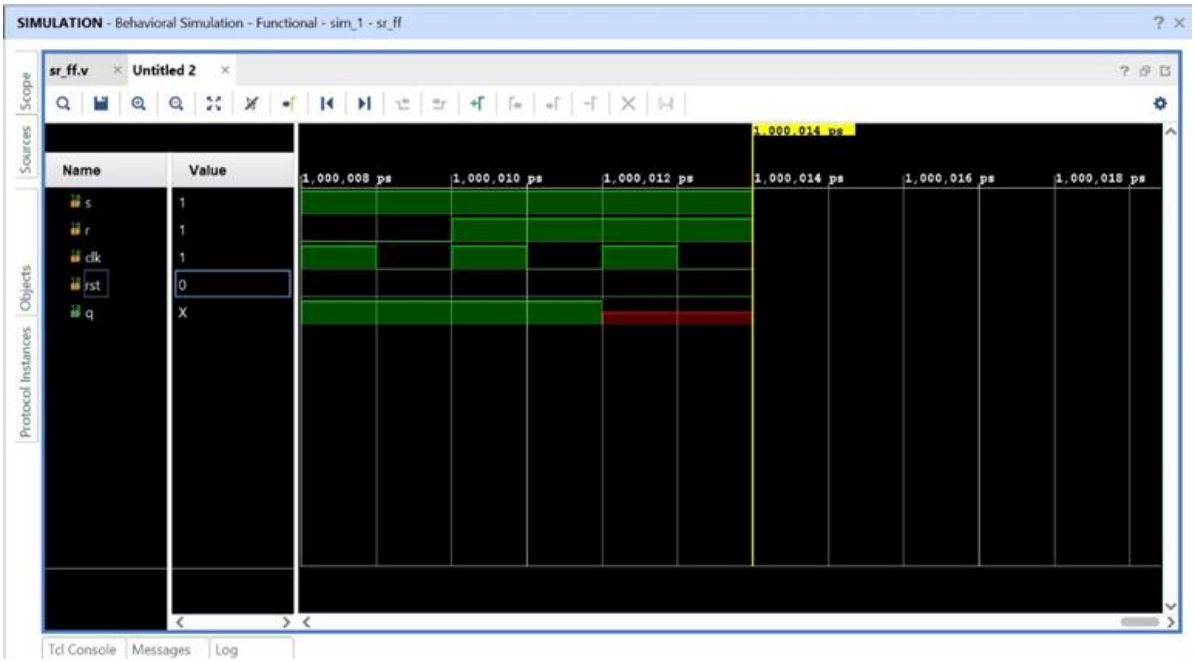
endcase

end

end

endmodule

OUTPUT



LOGIC DIAGRAM

\\ 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 jk_ff(clk,q,rst,j,k);

input j,k,clk,rst;

output reg q;

always@(posedge clk)

begin

if(rst==1)

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;

endcase

end

end

endmodule
```

OUTPUT

LOGIC DIAGRAM

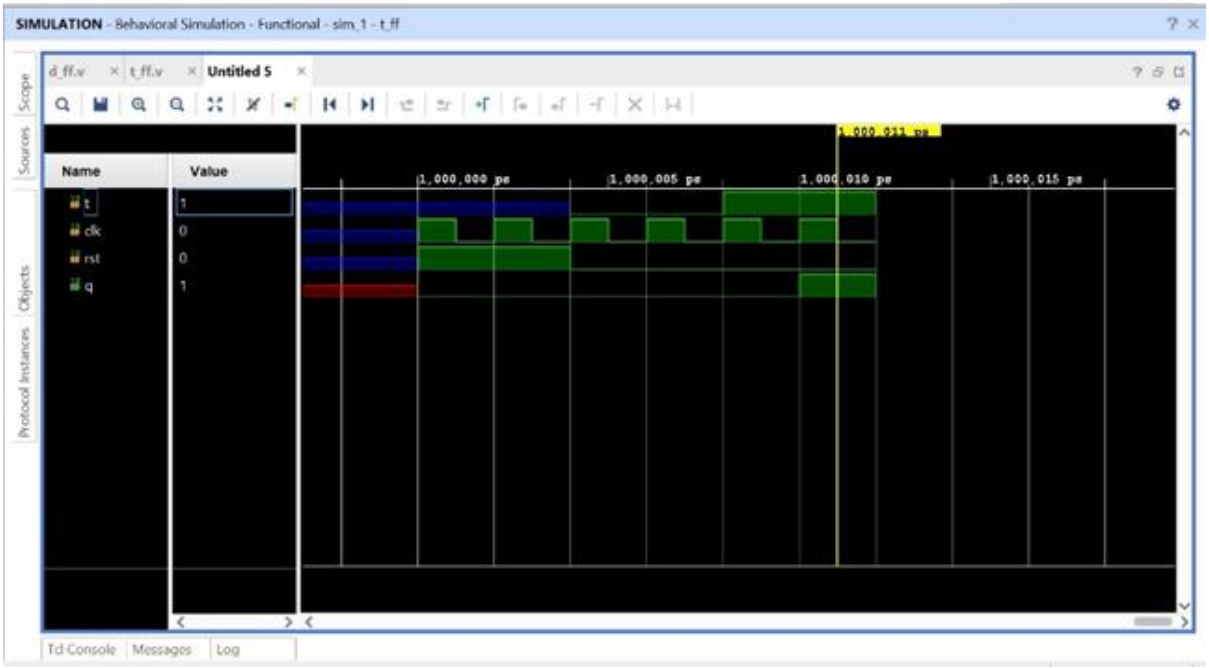
\\T FLIPFLOP\\

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 t_ff(clk,q,rst,t);
input t,clk,rst;
output reg q;
always@(posedge clk)
begin
if(rst==1)
q=1'b0;
else
if(t==0)
q=q;
else
q=~q;
end
endmodule
```

OUTPUT



LOGIC DIAGRAM

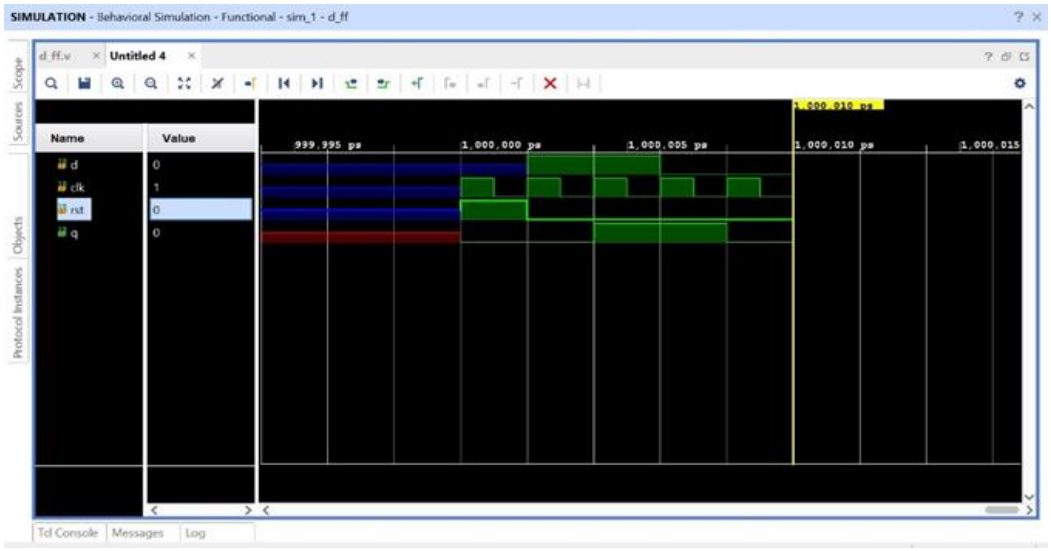
\\ D FLIPFLOP\\

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

VERILOG CODE

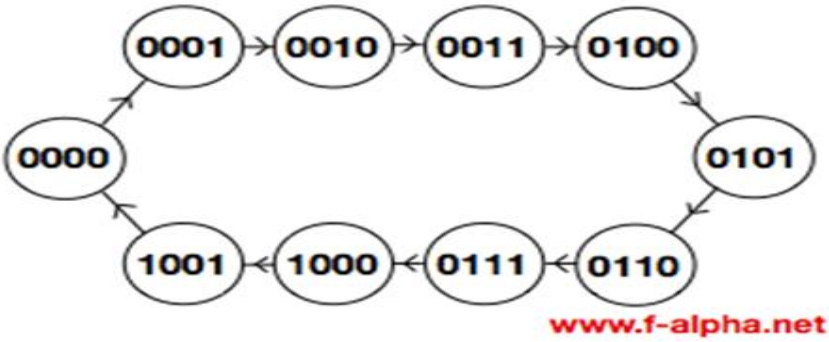
```
module d_ff(clk,q,rst,d);
input d,clk,rst;
output reg q;
always@(posedge clk)
begin
if(rst==1)
q=1'b0;
else
q=d;
end
endmodule
```

OUTPUT



LOGIC DIAGRAM

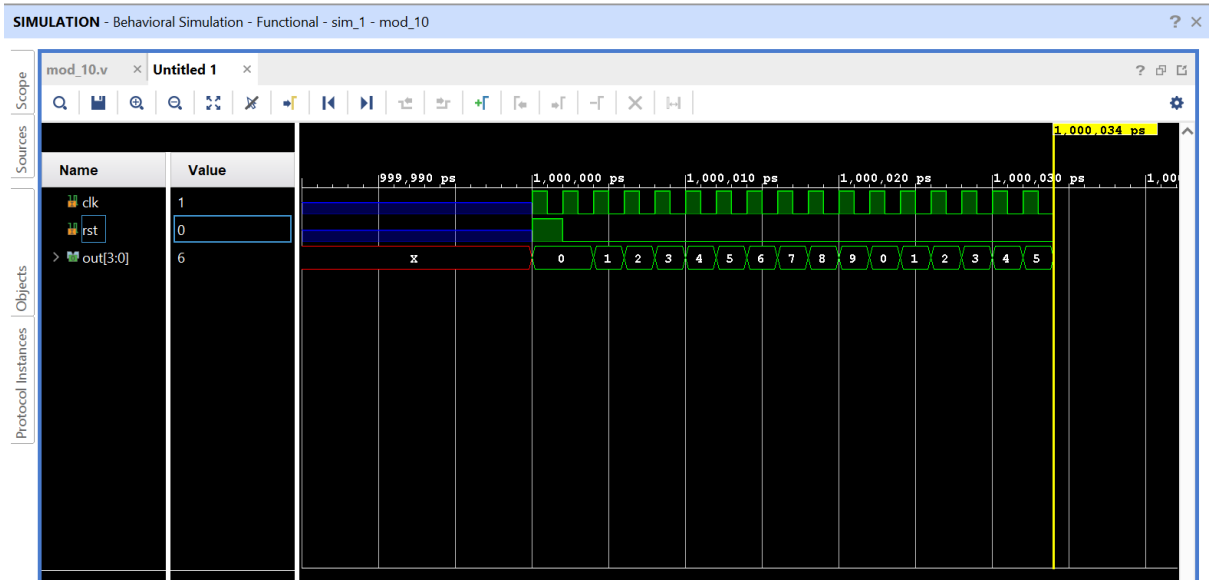
\\ MOD 10 COUNTER\\



VERILOG CODE

```
module mod_10(clk,rst,out);

input clk,rst;

output reg[3:0]out;

always@(posedge clk)

begin

if(rst==1|out==9)

out=4'b0;

else

out=out+1;

end

endmodule
```

OUTPUT



LOGIC DIAGRAM

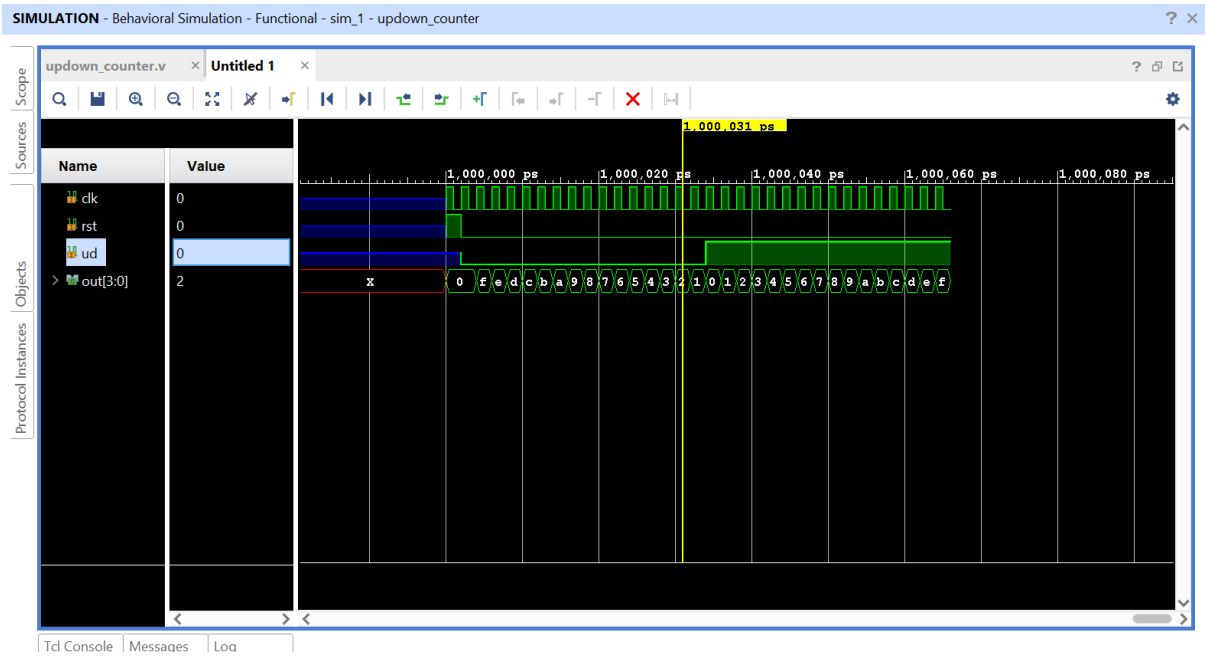
\\UPDOWN COUNTER\\

COUNT-UP Mode				*COUNT-DOWN Mode*			
States	Q_C	Q_B	Q_A	*States*	Q_C	Q_B	Q_A
0	0	0	0	7	1	1	1
1	0	0	1	6	1	1	0
2	0	1	0	5	1	0	1
3	0	1	1	4	1	0	0
4	1	0	0	3	0	1	1
5	1	0	1	2	0	1	0
6	1	1	0	1	0	0	1
7	1	1	1	0	0	0	0

VERILOG CODE

```
module updown_counter(clk,rst,ud,out);

input clk,rst,ud;

Output reg[3:0]out;

always@(posedge clk)

begin

if(rst==1)

out=4'b0;

else if (ud==1)

out=out+1;

else if(ud==0)

out=out-1;

end

endmodule
```

OUTPUT



RESULT

Thus The Simulate And Synthesis SR, Jk, T, D - Flipflop, Counter Design Using Xilinx Ise Are Verified Successfully