EXPERIMENT-9

Objective: To design different types of flip flops using VHDL.

Resources Required:

Hardware Requirement: Computer

Software Requirement: XILINX 8.2 Software

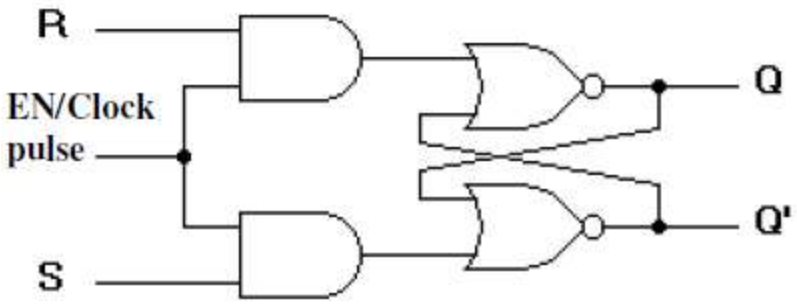
Theory:

Flip Flop: Flip flop is used to store one single bit of the information. Since Flip Flop is a sequential circuit so its input is based upon two parameters, one is the current input and other is the output from previous state.It has two outputs, both are complement of each other. It may be in one of two stable states, either 0 or 1.

SR Flip Flop: The SR flip flop is a 1-bit memory bistable device having two inputs, i.e., SET and RESET. The SET input 'S' set the device or produce the output 1, and the RESET input 'R' reset the device or produce the output 0. The SET and RESET inputs are labeled as S and R, respectively.

The SR flip flop stands for "Set-Reset" flip flop. The reset input is used to get back the flip flop to its original state from the current state with an output 'Q'. This output depends on the set and reset conditions, which is either at the logic level "0" or "1".

The NAND gate SR flip flop is a basic flip flop which provides feedback from both of its outputs back to its opposing input. This circuit is used to store the single data bit in the memory circuit. So, the SR flip flop has a total of three inputs, i.e., 'S' and 'R', and current output 'Q'. This output 'Q' is related to the current history or state. The term "flip-flop" relates to the actual operation of the device, as it can be "flipped" to a logic set state or "flopped" back to the opposing logic reset state.



Truth Table:

INPUTS		OUTPUTS	
S	R	Q	Qb
0	0	Q	Qb

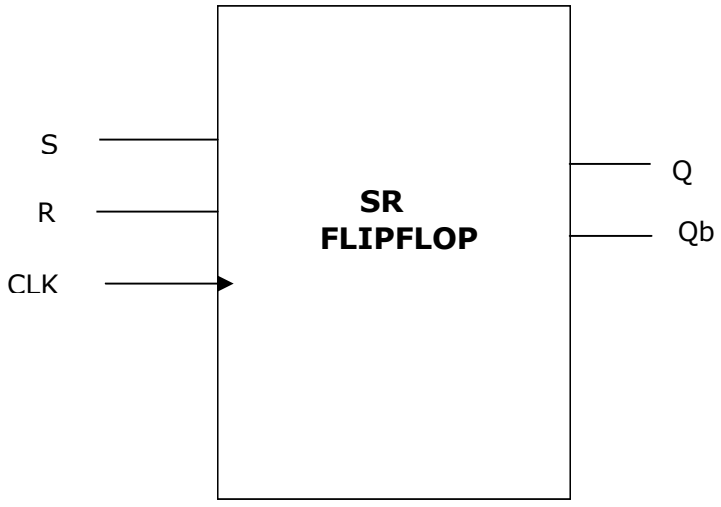
0	1	0	1
1	0	1	0
1	1	X	X

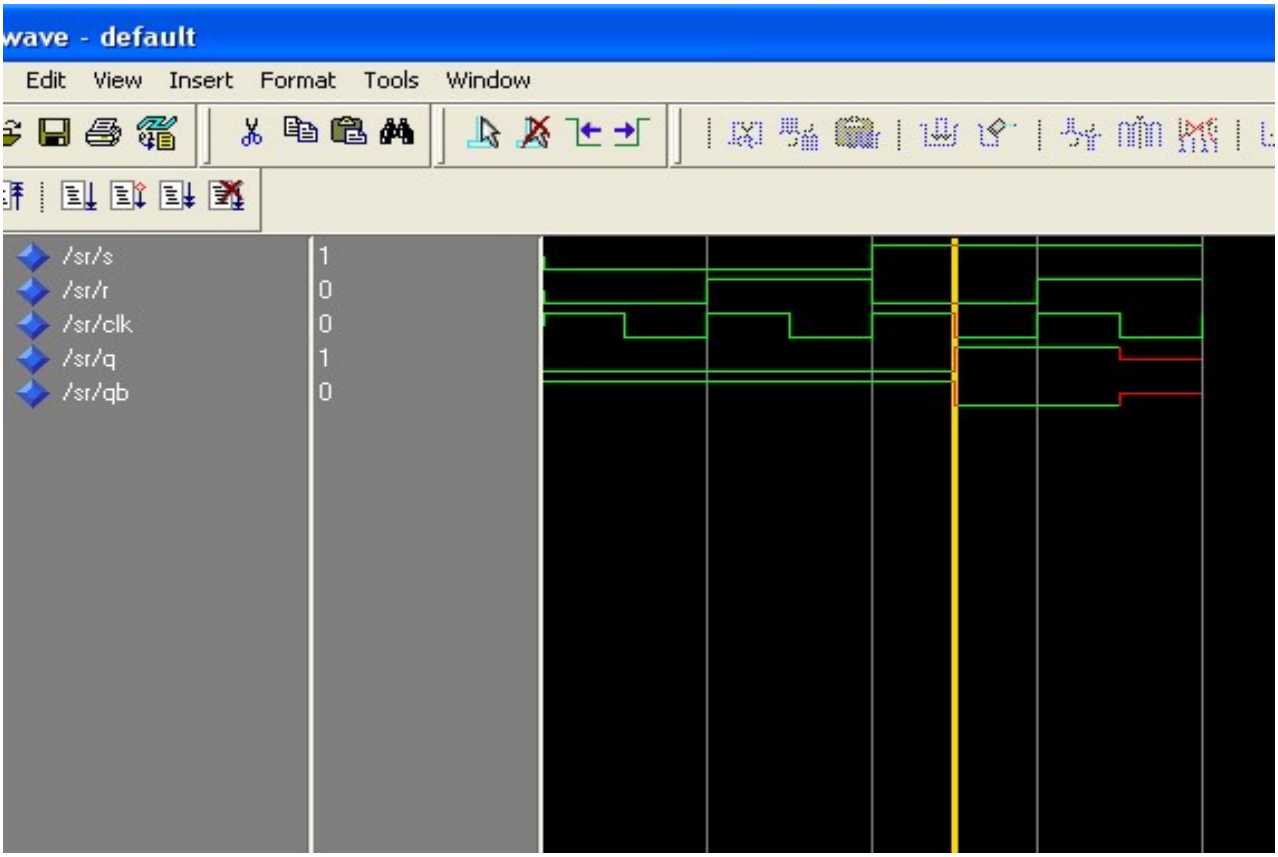
VHDL Code:

```
library ieee;
use ieee.std_logic_1164.all;
entity SR is
port(S,R,clk: in std_logic;Q:inout std_logic:='0';Qb:inout std_logic:='1');
end SR;
architecture ff of SR is
begin
process(S,R,clk)
variable t,tb: std_logic;
begin
t:=Q;
tb:=Qb;
if (clk='0'and clk'event) then if(S='0'and R='0') then t:=t;tb:=tb;
elsif(S='0'and R='1') then t:='0';tb:='1';
elsif(S='1'and R='0') then t:='1';tb:='0';
elsif(S='1'and R='1') then t:='U';tb:='U'; end if;
Q<=t;
Qb<=tb; end if;
end process; end ff;
```

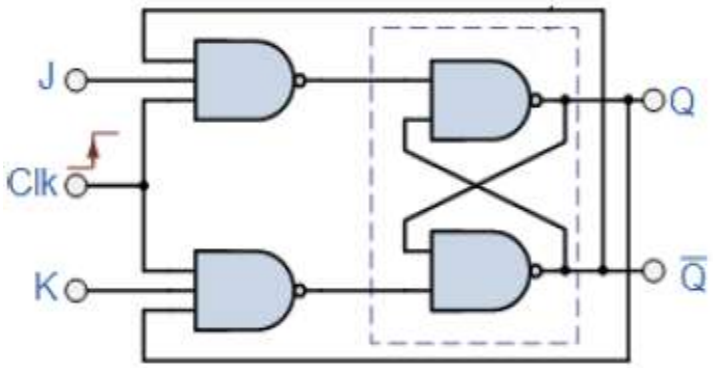
Output:

RTL Schematic:





JK Flip Flop: The JK flip flop (JK means Jack Kilby, a Texas instrument engineer, who invented it) is the most versatile flip-flop, and the most commonly used flip flop. Like the RS flip-flop, it has two data inputs, J and K, and an EN/clock pulse input (CP). Note that in the following circuit diagram NAND gates are used instead of NOR gates. It has no undefined states, however. The fundamental difference of this device is the feedback paths to the AND gates of the input, i.e. Q is AND-ed with K and CP and Q' with J and CP.



Truth Table:

INPUTS		OUTPUTS	
J	K	Q	Qb
0	0	Q	Qb
0	1	0	1
1	0	1	0
1	1	$\overline{Q}$	$\overline{Qb}$

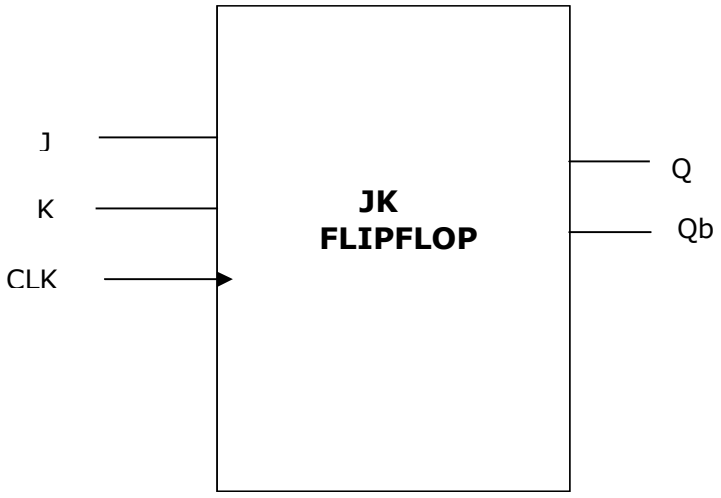
VHDL Code:

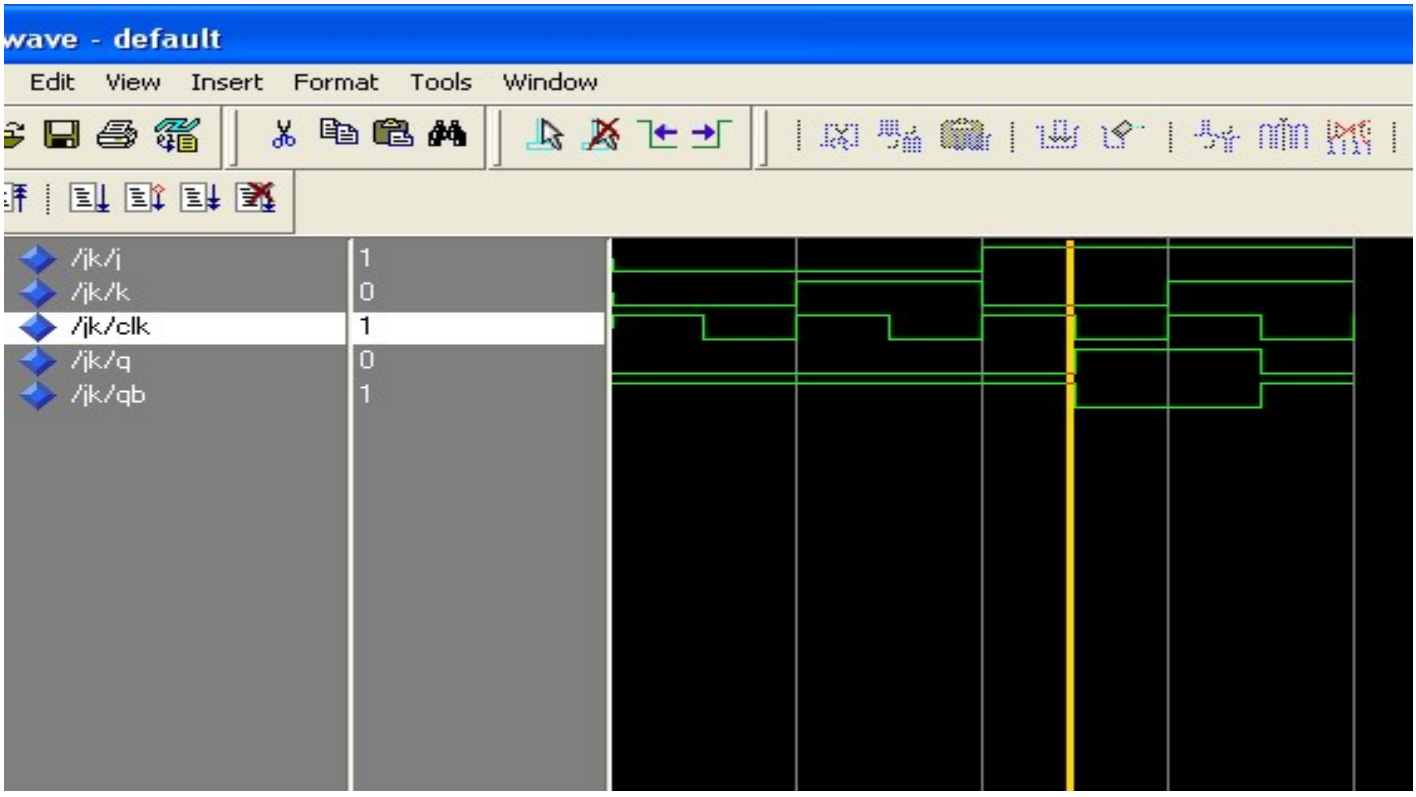
```
library ieee;
use ieee.std_logic_1164.all;
entity JK is
port(J,K,clk: in std_logic;Q:inout std_logic:='0';Qb:inout std_logic:='1');
end JK;
architecture ff of JK is begin
process(J,K,clk) variable t,tb: std_logic; begin
t:=Q;
tb:=Qb;
if (clk='0'and clk'event) then if(J='0'and K='0') then t:=t;tb:=tb;
elsif(J='0'and K='1') then t:='0';tb:='1';
elsif(J='1'and  K='0')  then  t:='1';tb:='0';  elsif(J='1'and  K='1')  then
t:=not t;tb:=not tb;

end if; end if; Q<=t;
Qb<=tb; end process; end ff;
```

Output:

RTL Schematic:

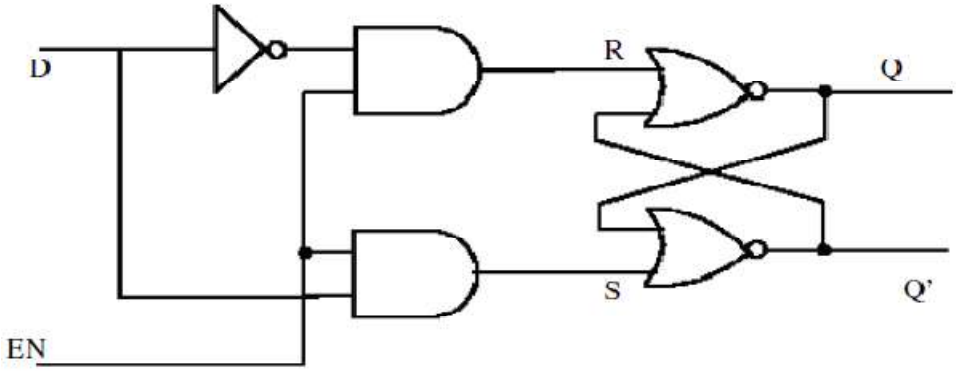




D Flip Flop:

An RS flip-flop is rarely used in actual sequential logic because of its undefined outputs for inputs R= S= 1. It can be modified to form a more useful circuit called D flip-flop, where D stands for data. The D flip-flop has only a single data input D as shown in the circuit diagram. That data input is connected to the S input of an RS flip-flop, while the inverse of D is connected to the R input. To allow the flip-flop to be in a holding state, a D-flip flop has a second input called Enable, EN. The Enableinput is AND-ed with the D-input.

- When EN=0, irrespective of D-input, the R = S = 0 and the state is held.
- When EN= 1, the S input of the RS flip-flop equals the D input and R is the inverse of D. Hence, output Q follows D, when EN= 1.
- When EN returns to 0, the most recent input D is ‘remembered'. The circuit operation is summarized in the characteristic table for EN=1.



Truth Table:

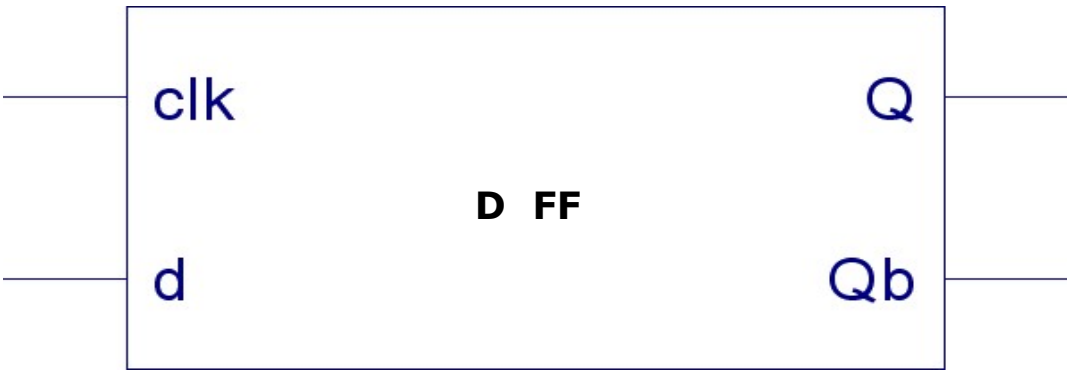
INPUTS	OUTPUTS	
D	Q	Qb
0	0	1
1	1	0

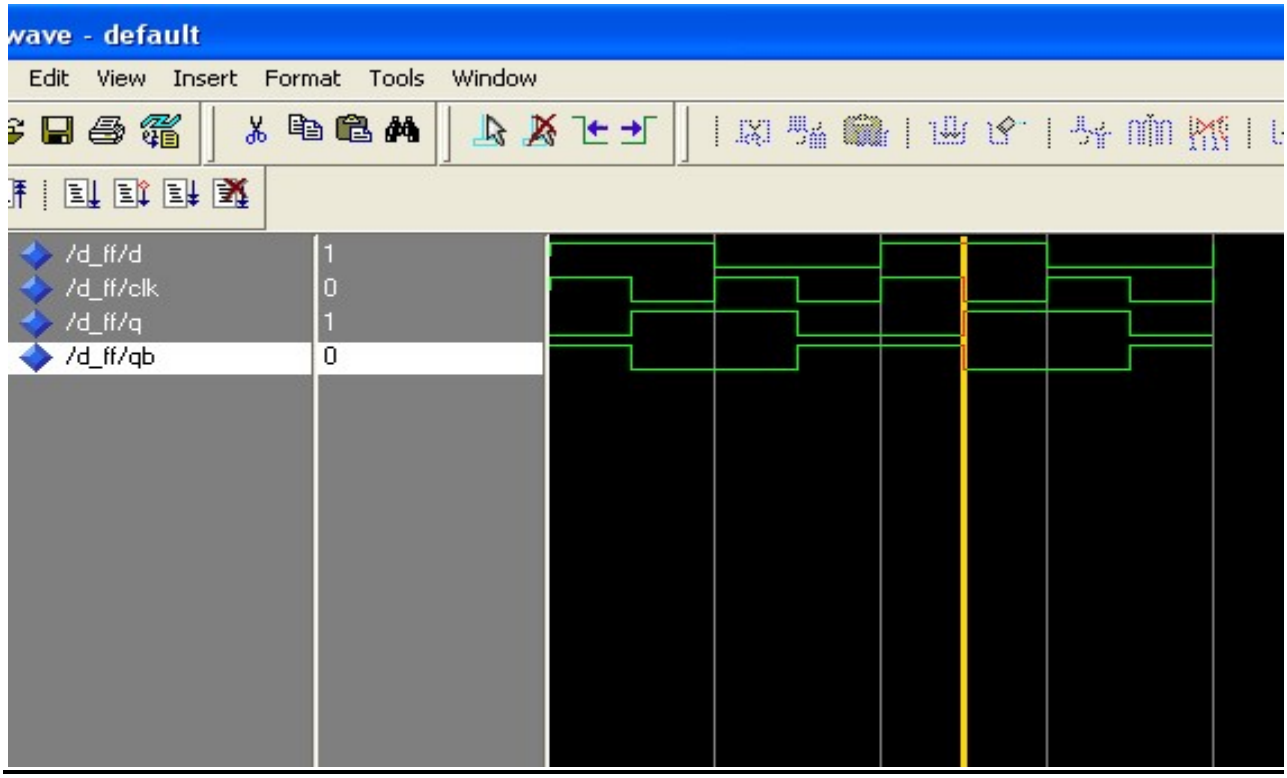
VHDL Code:

```
library ieee;
use ieee.std_logic_1164.all; entity d_ff is
port(d,clk:in  std_logic;  Q:inout  std_logic:='0';Qb:inout  std_logic:='1');
end d_ff;
architecture behaviour of d_ff is begin
process(d,clk) begin
if (clk='0' and clk'event)then q<=d;
qb<=not(d); end if;
end process; end behaviour;
```

Output:

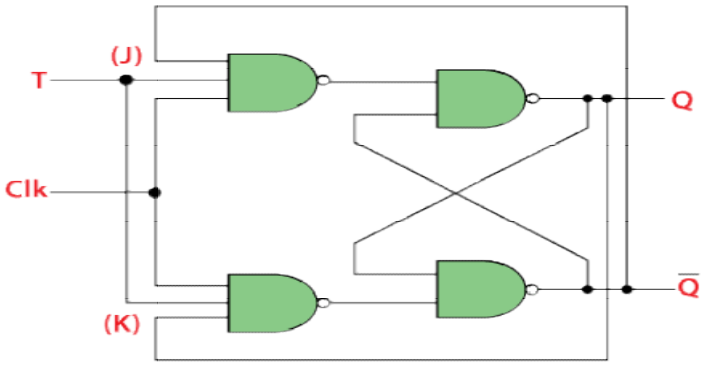
RTL Schematic:





T Flip Flop:

The T flip-flop is a single input version of the JK flip-flop. The T flip-flop is obtained from the JK type if both inputs are tied together.



Truth Table:

INPUTS	OUTPUTS	
T	Q	Qb
0	Q	Qb
1	Qb	Q

VHDL Code:

```
COMPONENT JK:-
library ieee;
```

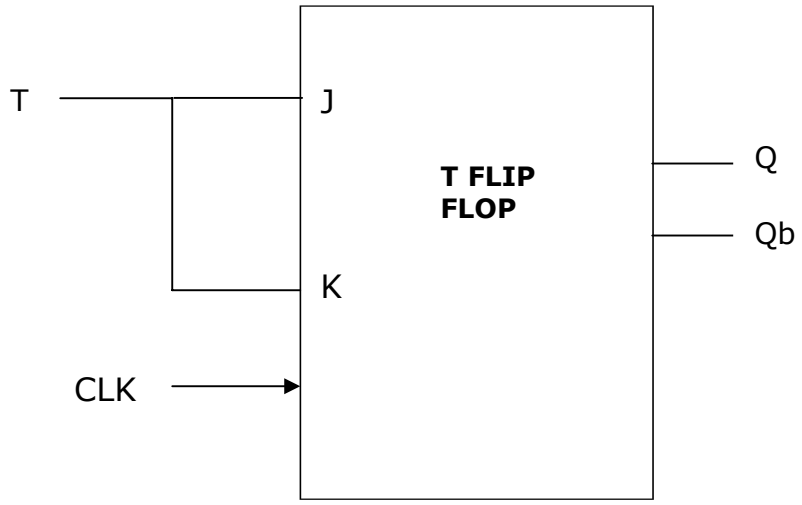
```
use ieee.std_logic_1164.all;
entity JK is
port(J,K,clk: in std_logic;Q:inout std_logic:='0';Qb:inout std_logic:='1');
end JK;
architecture ff of JK is begin
process(J,K,clk) variable t,tb: std_logic; begin
t:=Q;
tb:=Qb;
if (clk='0'and clk'event) then if(J='0'and K='0') then t:=t;tb:=tb;
elsif(J='0'and K='1') then t:='0';tb:='1';
elsif(J='1'and  K='0')  then  t:='1';tb:='0';  elsif(J='1'and  K='1')  then
t:=not t;tb:=not tb; end if;
end if; Q<=t;
Qb<=tb; end process; end ff;
```

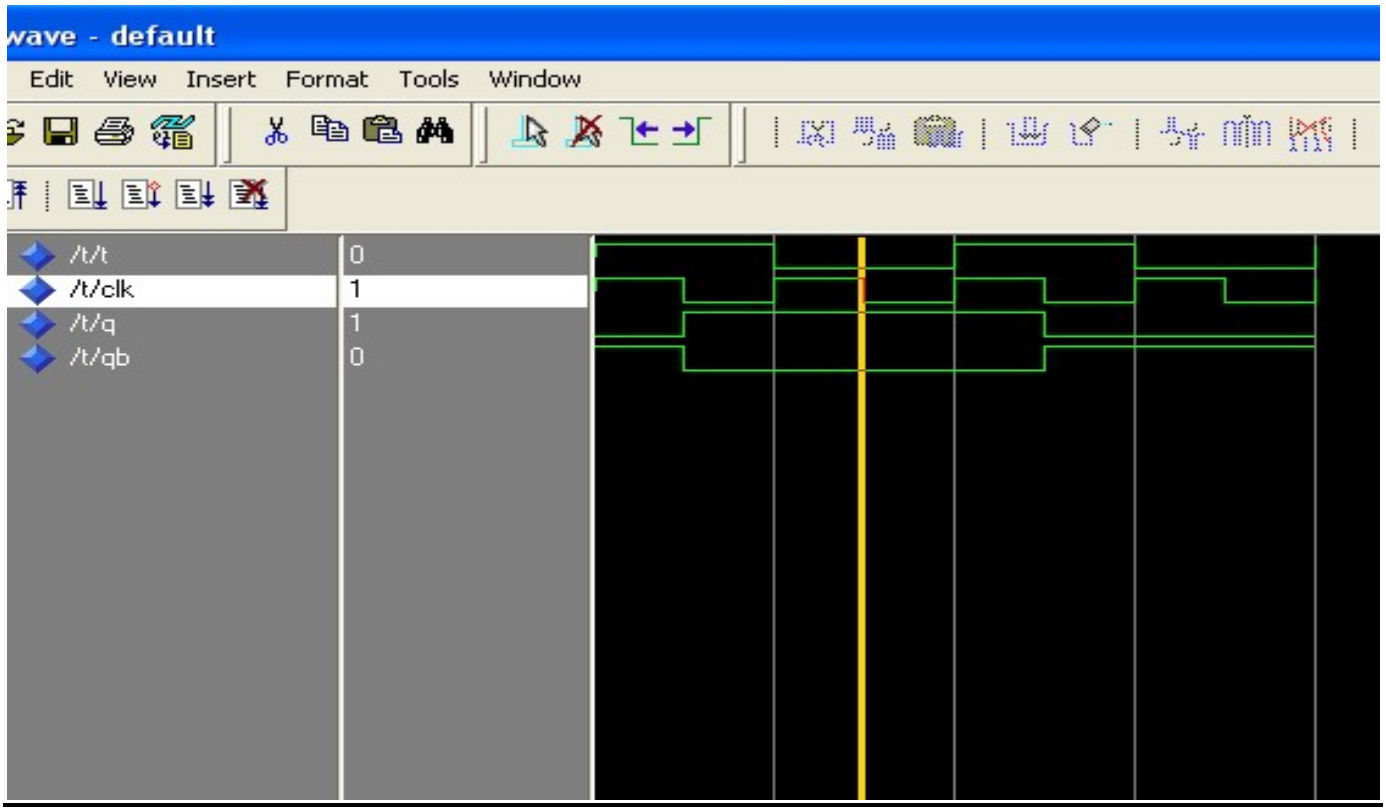
TOP MODULE:-

```
library ieee;
use ieee.std_logic_1164.all; entity T is
port(T,clk:  in  std_logic;Q:inout  std_logic:='0';Qb:inout  std_logic:='1');
end T;
architecture ff of T is component JK
port(J,K,clk: in std_logic;Q:inout std_logic:='0';Qb:inout std_logic:='1');
end component;
begin
X1: JK port map(T,T,clk,Q,Qb);
end ff;
```

Output:

RTL Schematic:





Results:VHDL codes of different flip flops are simulated & synthesized.