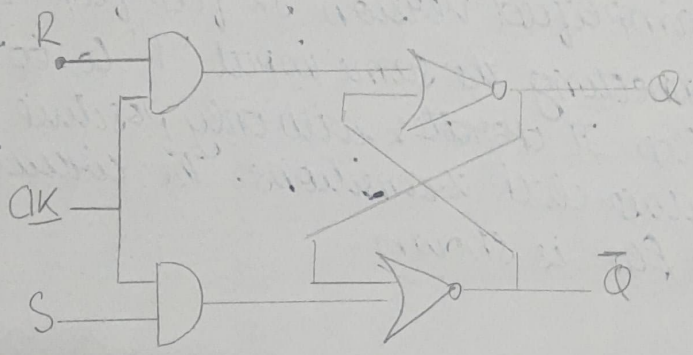
Experiment-6

Aim: Design
- SR flip flop
- JK flip flop
- D flip flop
- T flip flop ; using behavioural style of modelling.

Software Used: ModelSim PE student Edition 10.4a

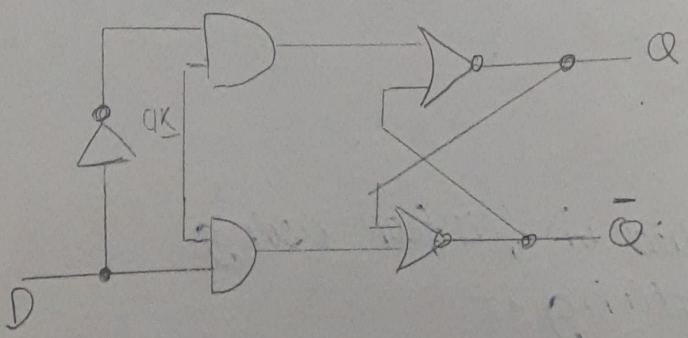
Theory: **SR Flip Flops:** These operate with only positive clock transitions or negative clock transition whereas, SR latch operates with enable signal. The circuit diagram of SR flip flop is shown



Truth Table:-

S	R	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	-

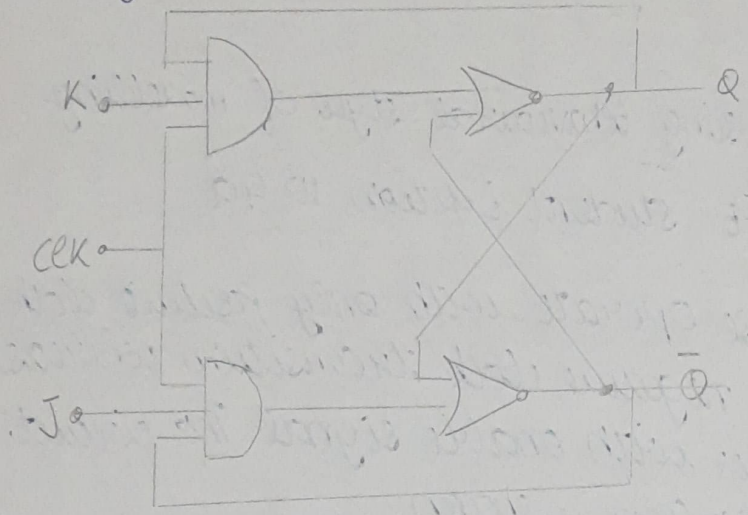
D Flip Flops: It operates with only positive clock transitions or negative transitions, whereas, D latch operates with enable signal. That means the output of D flip flop insensitive to the changes in the input, D except for active transition of the clock signal. The circuit diagram of D flip flop



Truth Table:

clk	D	Qn+1
0	x	Qn
1	0	0
1	1	1

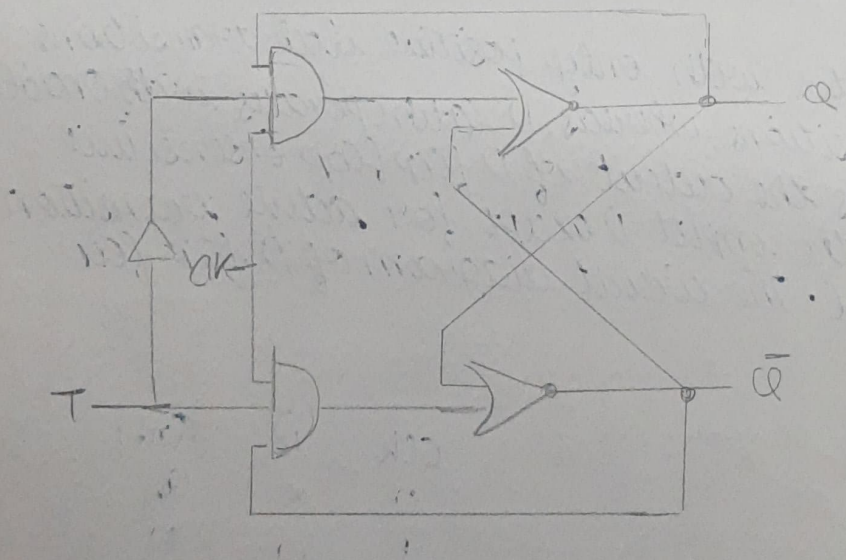
JK Flip Flop:- It is the modified version of SR flip flop. It operates with only positive clock transitions or negative clock transitions. The circuit diagram is shown.



Truth Table:

clk	J	K	Qn+1
0	x	x	Qn
1	0	0	Qn
1	0	1	0
1	1	0	1
1	1	1	$\overline{Qn}$

T Flip Flop:- It is the simplified version JK flip flop It is obtained by connecting the same input 'T' to both inputs of JK flip flop. It operates with only positive clock transitions or negative clock transitions. The circuit diagram of T flip flop is shown.



Truth Table:

clk	T	Qn+1
0	x	Qn
1	0	Qn
1	1	$\overline{Qn}$

Result:- Hence we have studied in flip flop using behavioral modelling.

Program Code:

SR Flip Flop:

```
library ieee;
use ieee.std_logic_1164.all;

entity srff is
        port(S, R, CLK: in std_logic;
             Q, Qbar: out std_logic);
end srff;

architecture arch_srff of srff is
        begin

        process(CLK)
          begin
            if (S /= R) and rising_edge(CLK) then
              Q <= S;
              Qbar <= R;

            elsif (S = '1') and rising_edge(CLK) then
              Q <= 'Z';
              Qbar <= 'Z';
            end if;

        end process;
end arch_srff;
```

D Flip Flop:

```
library ieee;
use ieee.std_logic_1164.all;

entity dff is
        port(D, CLK: in std_logic;
             Q, Qbar: out std_logic);
end dff;

architecture arch_dff of dff is
        begin

        process(CLK)
          begin
            if rising_edge(CLK) then
              Q <= D;
              Qbar <= not D;

            end if;
        end process;
end arch_dff;
```

JK Flip Flop:

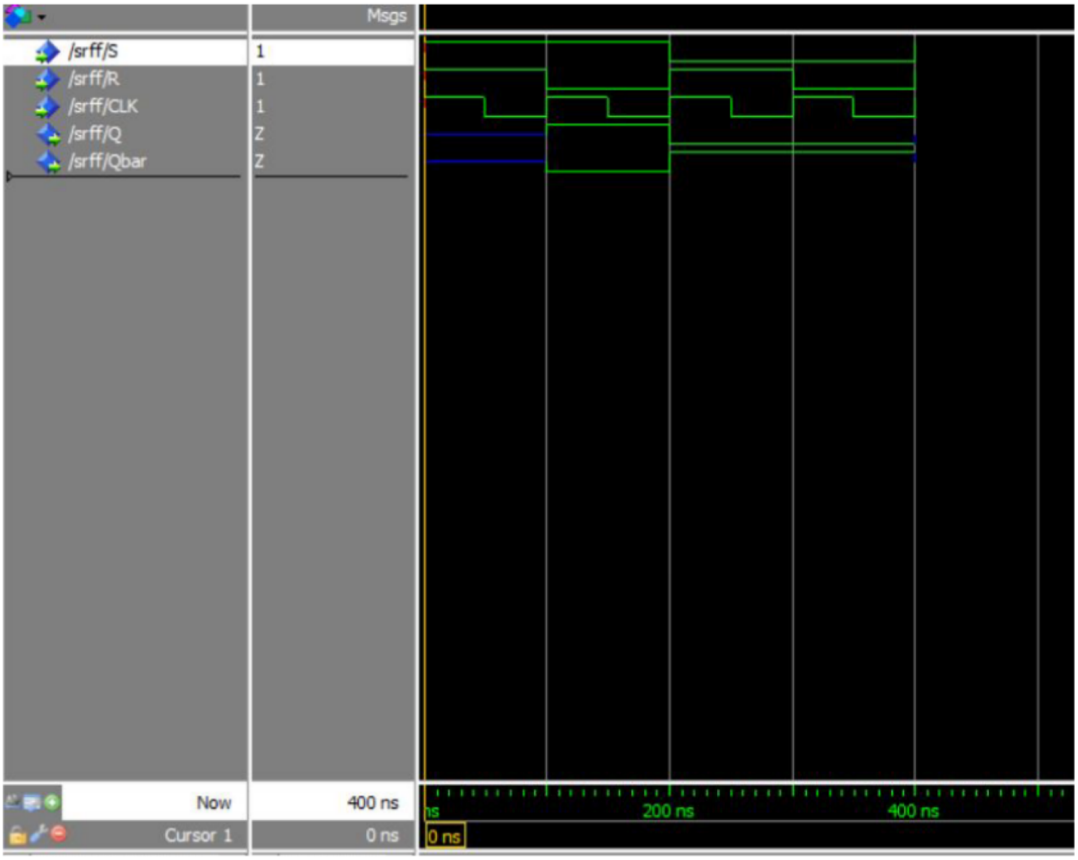
```
library ieee;
use ieee.std_logic_1164.all;

entity jkff is
        port(J, K, CLK: in std_logic;
             Q, Qbar: inout std_logic);
end jkff;

architecture arch_jkff of jkff is
        begin

        process(CLK)
          begin
            if (J /= K) and rising_edge(CLK) then
              Q <= J;
              Qbar <= K;

            elsif (J = '1') and rising_edge(CLK) then
              Q <= not Q;
              Qbar <= not Qbar;
            end if;

        end process;
end arch_jkff;
```

T Flip Flop:

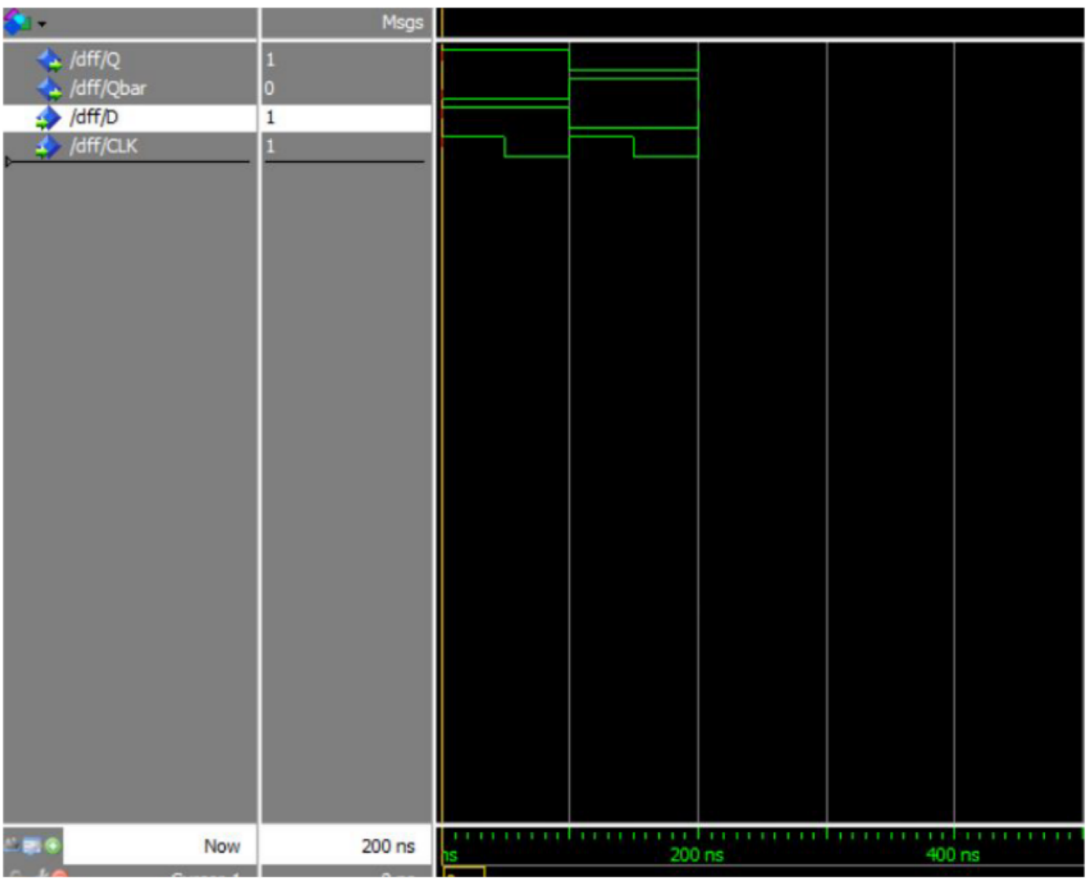
```
library ieee;
use ieee.std_logic_1164.all;

entity tff is
        port(T, CLK: in std_logic;
             Q, Qbar: inout bit);
end tff;

architecture arch_tff of tff is
        begin

        process(CLK)
          begin
            if (T = '1') and rising_edge(CLK) then
              Q <= not Q;
              Qbar <= not Qbar;
            end if;

        end process;
end arch_tff;
```

OUTPUT:

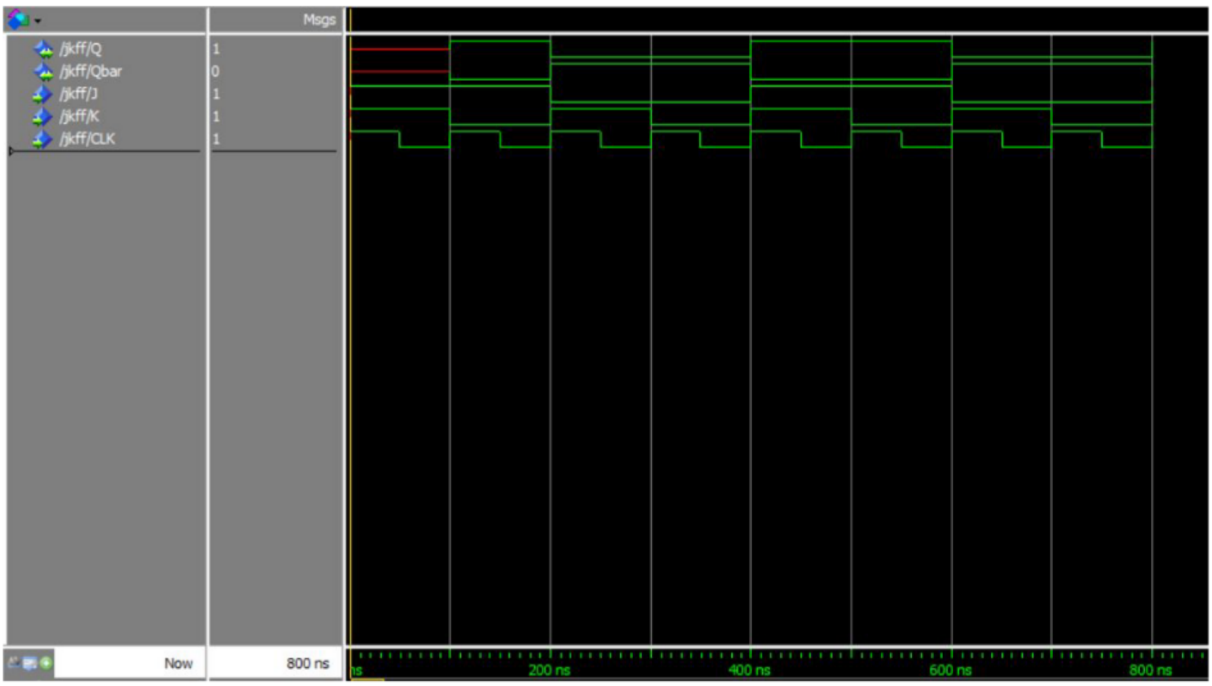
SR Flip Flop



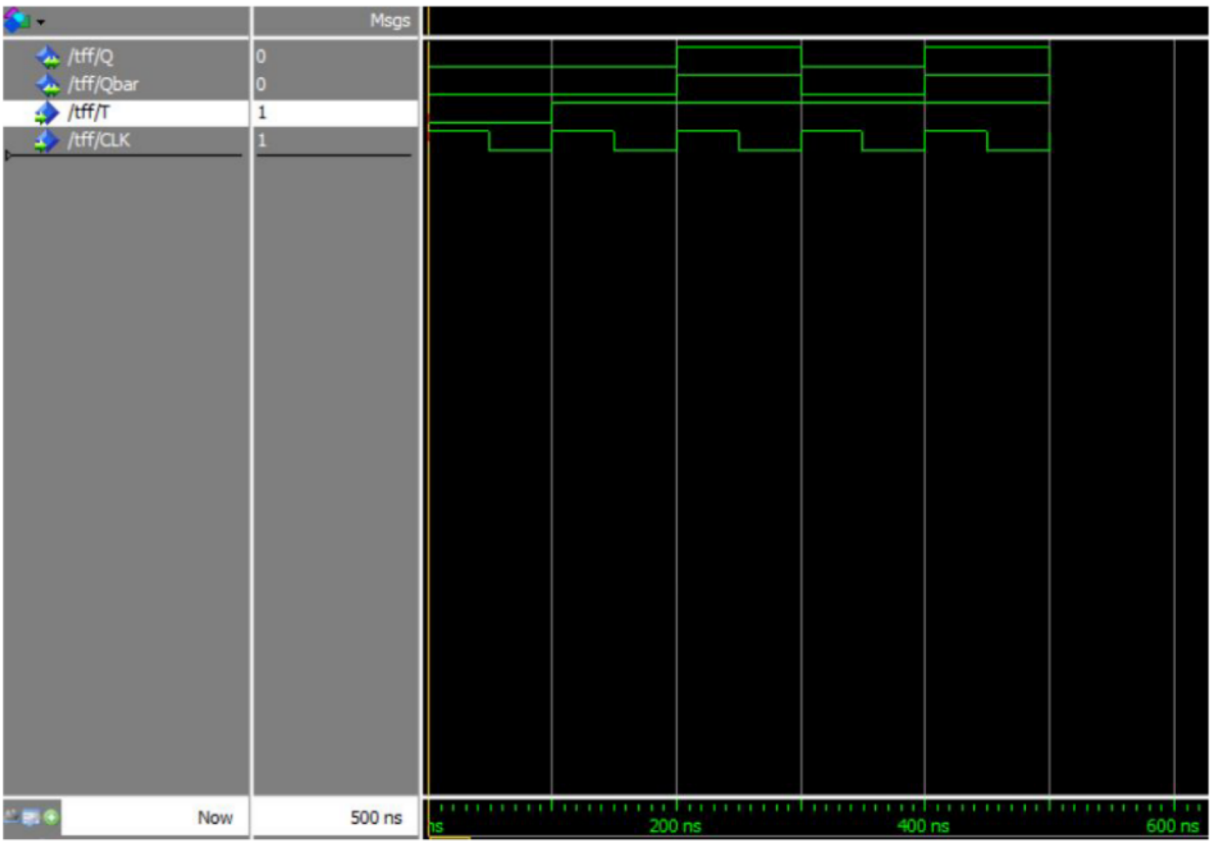
D Flip Flop



JK Flip Flop



T Flip Flop



Result: Hence we have studied and programmed the following flip flops in vhdl.