EXPT NO.: 8

DESIGN OF JK, SR, D AND T FLIP-FLOPS

AIM:

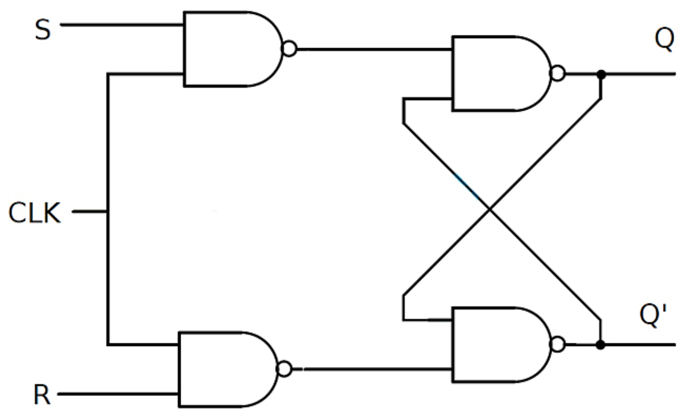
To design and simulate JK, SR, D and T flip-flops.

SR FLIP-FLOP:

The SR flip flop is also known as SR latch is one of the basic sequential logic circuit types of flip flop. It has two input ‘S’ and ‘R’ and two output Q and $\bar{Q}$. If Q is ‘1’ the latch is said to be SET and if Q is 0 the latch is said to be RESET. The design of SR flip flop by cross coupled NAND gates or NOR gate.

When the device is ‘SET’ means output is ‘1’, and is labelled ***S*** and when the output is ‘RESET’ means the output is ‘0’, labelled ***R***. the SR stands for ***‘Set-Reset’***. The reset input reset the flip flop to its original state with an output Q that will be either ‘1’ or ‘0’ logic depending upon the set/reset condition.

The truth table and logic diagram of SR flip-flop is given below:



INPUTS			OUTPUT
S	**R**	Q_n	Q_{n+1}
0	**0**	**0**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**0**	**1**
1	**0**	**1**	**1**
1	**1**	**0**	**X**
1	**1**	**1**	**X**

The possible cases:

Case 1: When both the inputs ***S and R are LOW***, then Q returns its previous state value i.e., it holds the previous data.

Case 2: When ***S is LOW and R is HIGH***, then flip-flop will be in Reset state i.e., $Q = 0, \bar{Q} = 1$.

Case 3: When ***S is HIGH and R is LOW***, then flip-flop will be in Set state i.e., $\boldsymbol{Q = 1, \overline{Q} = 0}$.

Case 4: When both the inputs ***S and R are HIGH***, then flip-flop is in Toggle state. This means that the next state will be intermediate (X).

The i/o ports needed to be declared for the formation of SR flip-flop is given below:

Port Name	INPUT/OUTPUT	Bus
S	In	No
R	In	No
Reset	In	No
Clk	In	No
Q	Out	No
$QBar$	Out	No

NB: Use temporary variable where ever necessary.

Syntax for if-then-else in VHDL:

```
if condition_1 then
        sequential statements
elsif condition2 then
        sequential statements
else
        sequential statements
end if;
```

VHDL Code for SR Flip-flop:

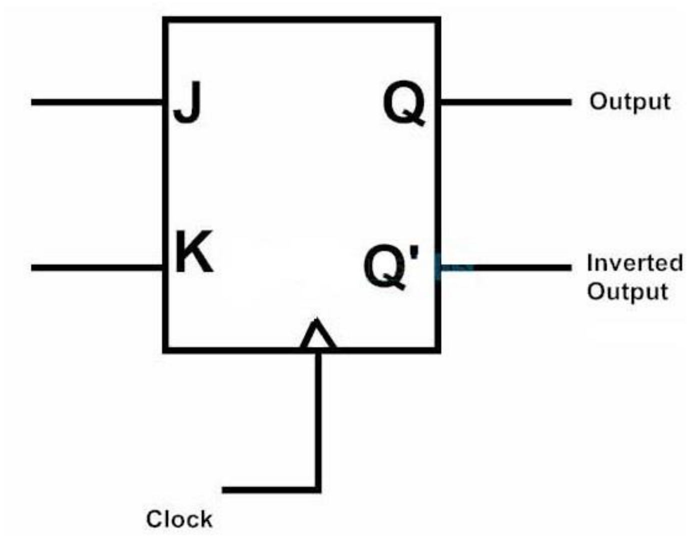
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;        // These two libraries are to be included for internal calculations.
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity srflipflop is
   Port ( S : in  STD_LOGIC;
          R : in  STD_LOGIC;
          Reset : in STD_LOGIC;
          Clk : in  STD_LOGIC;
          Q : out  STD_LOGIC;
          QBar : out  STD_LOGIC);
end srflipflop;
```

```
architecture Behavioral of srflipflop is
signal temp : STD_LOGIC := '0';

begin
        Q <= temp;
        QBar <= (not temp);

        process(S,R,Reset,Clk)
        begin
                if (Reset = '1') then
                        temp <= '0';

                elsif (RISING_EDGE(Clk)) then   //RISING_EDGE(Clk) represents Clk = 1
                        if (S = '0' and R = '0') then
                                temp <= temp;
                        elsif (S = '1' and R = '1') then
                                temp <= 'Z';          // Z is used to represent the intermediate condition
                        elsif (S /= R) then         // S/=R represents S ≠ R
                                temp <= S;
                        end if;
                end if;
        end process;
end Behavioral;
```

JK FLIP-FLOP:

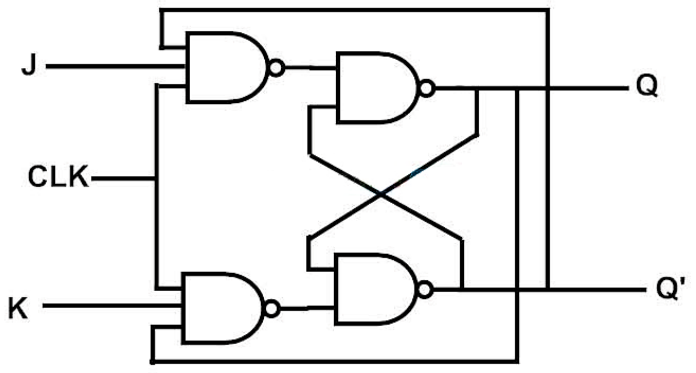
A JK flip-flop is the modification of SR flip-flop with no illegal state. In this the J input is similar to the SET input of SR flip-flop and the K input is similar to the RESET input of SR flip-flop.

The symbol of JK flip-flop is shown below:



The design of the JK flip-flop is such that the three inputs to one NAND gate are J, clock signal along with a feedback signal from $\bar{Q}$ and the three inputs to the other NAND are K, clock signal along with a feedback signal from Q.

The truth table and logic diagram of JK flip-flop is given below:



INPUTS		OUTPUTS	
J	**K**	Q	$\bar{Q}$
0	**0**	**REMEMBERS**	
0	**1**	**0**	**1**
1	**0**	**1**	**0**
1	**1**	**TOGGLES**	

The possible cases:

Case 1: When both the inputs ***J and K are LOW***, then Q returns its previous state value i.e., it holds the previous data.

When we apply a clock pulse to the J K flip-flop and the J input is low then irrespective of the other NAND gates, the NAND gate-1 output becomes HIGH. In the same manner, if the K input is low then output of NAND gate-2 is also HIGH. So thus, the output remains in the same state i.e., no change in the state of flip-flop.

Case 2: When ***J is LOW and K is HIGH***, then flip flop will be in Reset state i.e., $Q = 0, \bar{Q} = 1$.

When we apply a clock pulse to the J K flip-flop and the inputs are J is low and K is high the output of the NAND gate connected to J input becomes 1. Then Q becomes 0. This will reset the flip-flop again to its previous state. So, the flip-flop will be in RESET state.

Case 3: When ***J is HIGH and K is LOW***, then flip-flop will be in Set state i.e., $Q = 1, \bar{Q} = 0$.

When we apply a clock pulse to the J K flip-flop and the inputs are J is high and K is low the output of the NAND gate connected to K input becomes 1. Then $\bar{Q}$ becomes 0. This will set the flip-flop with the high clock input. So, the flip-flop will be in SET state.

Case 4: When both the inputs J and K are HIGH, then flip-flop is in Toggle state. This means that the output will complement of the previous state.

Race around condition of JK Flip-Flop:

For high inputs of JK flip-flop, only the lower NAND gates are triggered by the outputs that are compliment to each other i.e., Q and $\bar{Q}$. So, while high inputs are connected to flip-flop, at any instant, one gate is enabled and the other gate will be disabled. If the upper gate is in disabled state, it will drive the flip-flop to SET state, later when the lower gate is enabled, it will drive the flip flop to RESET state which causes the toggling of output. This will cause the Race around condition in JK flip-flop.

Steps to avoid racing condition:

- We can avoid the Race around condition by setting up the clock-on time less than the propagation delay of the flip-flop. It can be achieved by edge triggering.
- By making the flip flop to toggle over one clock period. This concept is introduced in Master Slave JK flip-flop.

The i/o ports needed to be declared for the formation of JK flip-flop is given below:

Port Name	INPUT/OUTPUT	Bus
J	In	No
K	In	No
Reset	In	No
Clk	In	No
Q	Out	No
$QBar$	Out	No

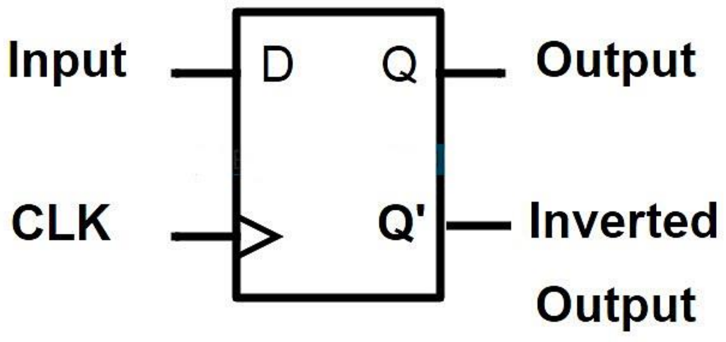
- **NB: Use temporary variable where ever necessary.**

D FLIP-FLOP:

D flip-flops are also called as '***Delay flip-flop***' or '***Data flip-flop***'. They are used to store 1-bit binary data. They are one of the widely used flip-flops in digital electronics. Apart from being the basic memory element in digital systems, D flip-flops are also considered as ***Delay line elements and Zero-Order Hold elements***.

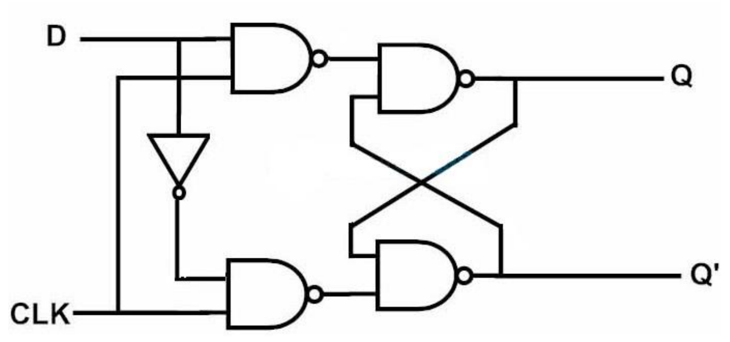
D flip-flop has two inputs, a clock (CLK) input and a data (D) input and two outputs, one is main output represented by Q and the other is complement of Q represented by $\bar{Q}$.

The symbol of a D flip-flop is shown below:



A D flip-flop is constructed by modifying an SR flip-flop. The S input is given with D input and the R input is given with inverted D input. Hence a D flip-flop is similar to SR flip-flop in which the two inputs are complement to each other, so there will be no chance of any intermediate state occurs. The major drawback of SR flip-flop is the race around condition which in D flip-flop is eliminated (because of the inverted inputs).

The truth table and logic diagram of D flip-flop is given below:



INPUT	OUTPUTS	
D	Q	$\bar{Q}$
0	0	1
1	1	0

The i/o ports needed to be declared for the formation of D flip-flop is given below:

Port Name	INPUT/OUTPUT	Bus
D	In	No
Reset	In	No
Clk	In	No
Q	Out	No
$QBar$	Out	No

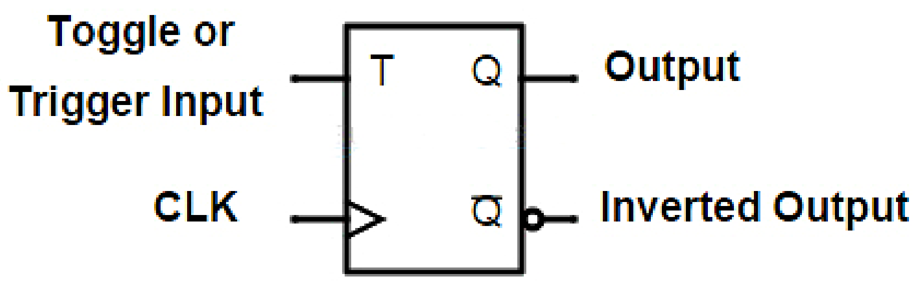
- **NB: Use temporary variable where ever necessary.**

T FLIP-FLOP:

In T flip flop, ***'T' defines the term 'Toggle'***. In SR flip-flop, we provide only a single input called 'Toggle' or 'Trigger' input to avoid an intermediate state occurrence. Now, this flip-flop work as a Toggle switch. The next output state is changed with the complement of the present state output. This process is known as 'Toggling'.

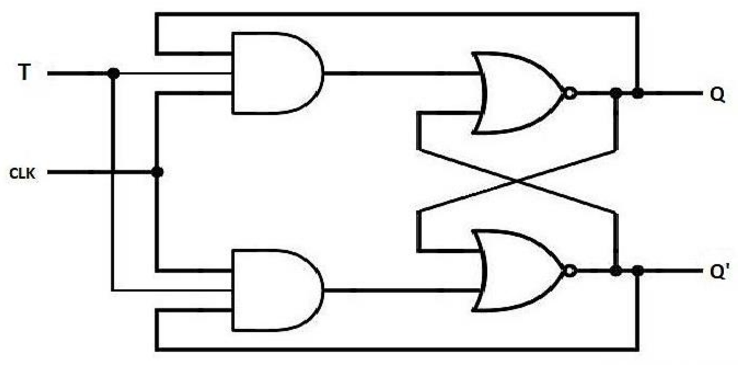
We can construct the 'T Flip Flop' by making changes in the 'JK Flip Flop'. The 'T Flip Flop' has only one input, which is constructed by connecting the input of JK flip-flop. This single input is called T. In simple words, we can construct the 'T Flip Flop' by converting a 'JK Flip Flop'. Sometimes the 'T Flip Flop' is referred to as single input 'JK Flip Flop'.

The symbol of a T flip-flop is shown below:



The T flip-flop is also called toggle flip-flop. It is a change of the JK flip-flop. The T flip flop is received by relating both inputs of a JK flip-flop. The T flip-flop is received by relating the inputs 'J' and 'K'. When T = 0, both AND gates are disabled. Therefore, there is no change in the output. When T= 1, the output toggles.

The truth table and logic diagram of T flip-flop is given below:



INPUT	OUTPUTS	
T	Q	$\overline{Q}$
0	1	0
1	0	1

The i/o ports needed to be declared for the formation of T flip-flop is given below:

Port Name	INPUT/OUTPUT	Bus
T	In	No
Reset	In	No
Clk	In	No
Q	Out	No
$QBar$	Out	No

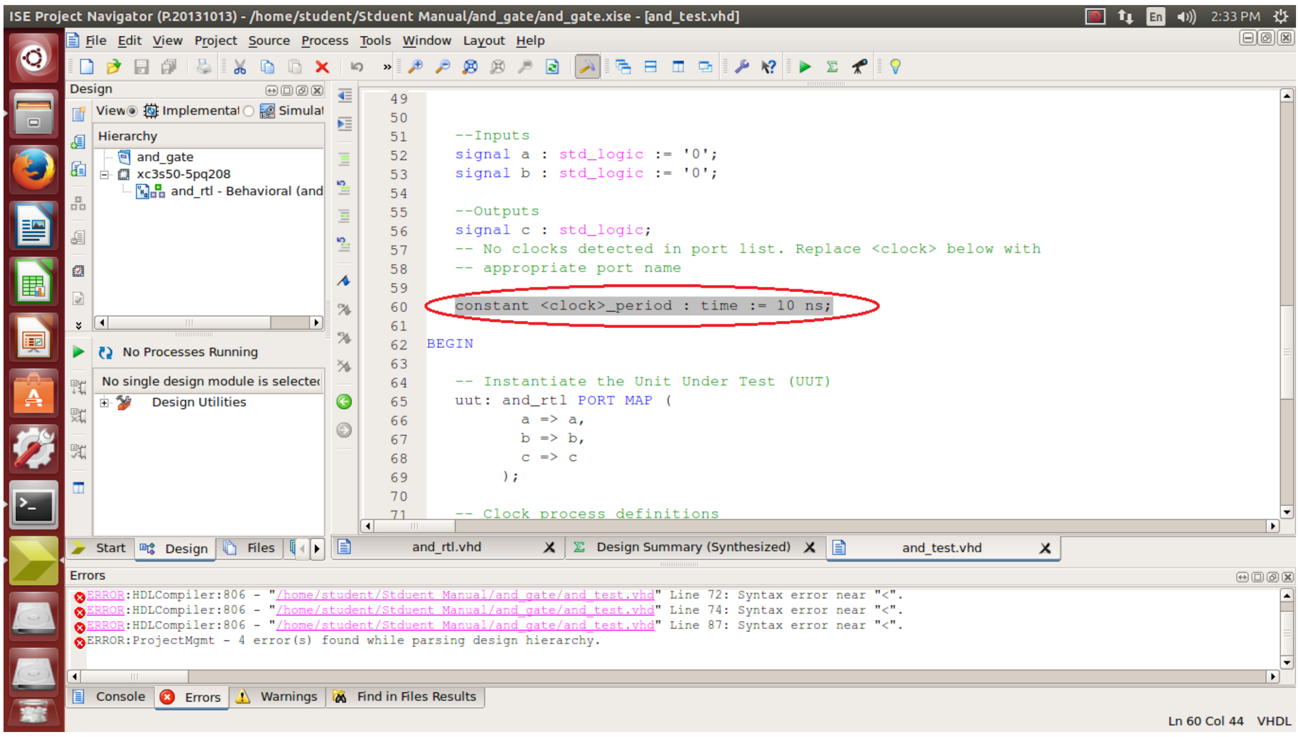
- **NB: Use temporary variable where ever necessary.**

- **In the test bench, for all kinds of flip-flops and other sequential circuits; look for:**

  constant Clk_period : time := 10 ns;

  and change it to:

  constant Clk_period : time := 2 ps;


ISE Project Navigator (P.20131013) - /home/student/Stduent Manual/and_gate/and_gate.xise - [and_test.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
and_gate
xc3s50-5pq208
and_rtl - Behavioral (and
No Processes Running
No single design module is selected
Design Utilities
Start Design Files
and_rtl.vhd
Design Summary (Synthesized)
and_test.vhd
Errors
Console Errors Warnings Find in Files Results
Ln 60 Col 44 VHDL

```

   --Inputs
   signal a : std_logic := '0';
   signal b : std_logic := '0';

   --Outputs
   signal c : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

   constant <clock>_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: and_rtl PORT MAP (
          a => a,
          b => b,
          c => c
        );

   -- Clock process definitions
```

```
ERROR:HDLCompiler:806 - "/home/student/Stduent Manual/and_gate/and_test.vhd" Line 72: Syntax error near "<".
ERROR:HDLCompiler:806 - "/home/student/Stduent Manual/and_gate/and_test.vhd" Line 74: Syntax error near "<".
ERROR:HDLCompiler:806 - "/home/student/Stduent Manual/and_gate/and_test.vhd" Line 87: Syntax error near "<".
ERROR:ProjectMgmt - 4 error(s) found while parsing design hierarchy.
```