Module -4

Contents:

i. Introduction to VHDL

ii. VHDL Models for Multiplexers

iii. VHDL Modules

iv. Set Reset Latch

V. Gated Latches

Vi. Edge Triggered D Flip-Flop

Vii. S R Flip Flop

Viii . J K Flip Flop

ix. T Flip Flop

X. Flip- Flop with additional inputs

Introduction to VHDL:

VHDL stands for VHSIC hardware description language, VHSIC stands for very high speed integrated Circuit. VHDL is general- purpose hardware description language which can be used to describe and simulate the operation of digital circuits. VHDL can describe a digital system at different levels, behavioral, data flow, and structural.

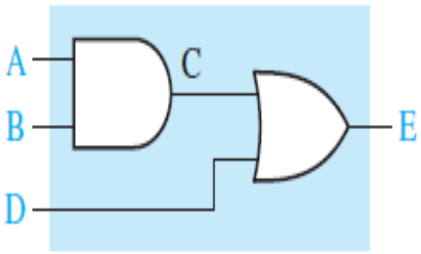
Structure of VHDL program:

VHDL program has two parts, **Entity part** and **Architecture Part,** Entity block is used to declare all input and output signals, architecture block is used to describe the internal operation of the circuit or module.

Syntax of VHDL program:

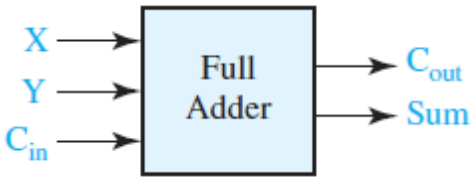
```
entity entity_name is
        port (inputs; outputs);
end entity_name;

architecture architecture_name of entity_name is
        signal internal signals;
begin
        statements;
end architecture_name;
```

Write VHDL program for the circuit shown below.




```
entity two_gates is
        port (A,B,D: in bit; E: out bit);
end two_gates;
architecture gates of two_gates is
        signal C:bit;
begin
        C <= A and B;
        E <= C or D;
end gates;
```

Write a program for Full adder



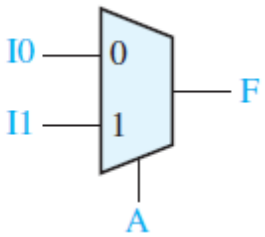

```
entity FullAdder is
        port (X,Y,Cin : in bit;
                Cout, Sum: out bit);
end FullAdder;

architecture Equations of FullAdder is
begin
        Sum <= X xor Y xor Cin;
        Cout <= (X and Y) or (X and Cin) or (Y and Cin);
end Equations;
```

Conditional Signal Assignment:

Syntax**:**

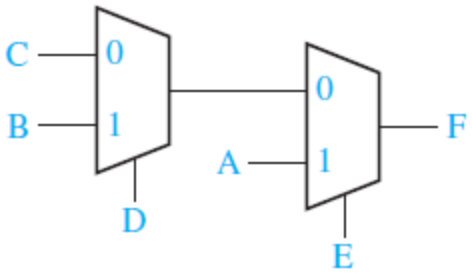
```
Signal_name <= expression1 when condition1
        else expression2 when condiiton2
        else expression3;
```

Write VHDL program for 2 to 1 Multiplexer using conditional signal assignment



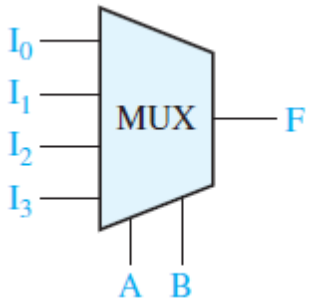

```
entity Mux is
        port(I0, I1,A: in bit; F:out bit);
end Mux;

architecture Equations of Mux is
begin
        F <= I0 when A='0'
        else I1;
end Equations;
```

Write VHDL program for cas



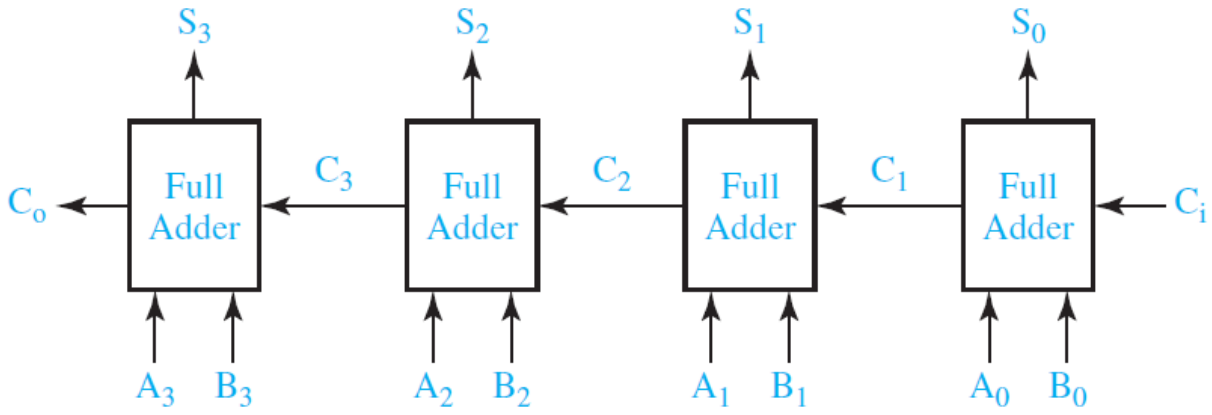

```
entity Mux is
        port(C,B,D,A,E: in bit; F:out bit);
end Mux;

architecture Equations of Mux is
begin
        F<=A when E='1'
                else B when D='1'
                else C;
end Equations;
```

Write VHDL program for 4 to 1 Multiplexer using conditional signal assignment




```
entity Mux is
        port(I0, I1,I2,I3,A,B: in bit; F:out bit);
end Mux;

architecture Equations of Mux is
begin
        F<=I0 when A&B = "00"
        else I1 when A&B = "01"
        else I2 when A&B = "10"
        else I3;
end Equations;
```

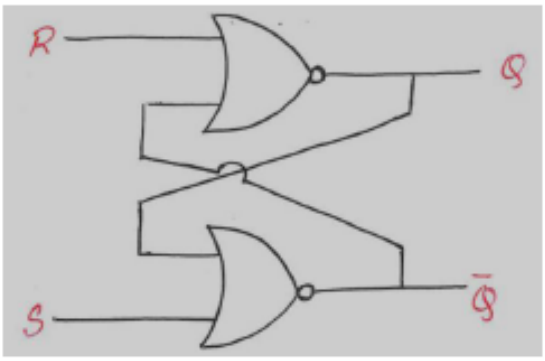
Write a VHDL program for 4-bit Binary Adder



In the above circuit, four full adders are used, hence while writing a program we should declare full adder as a component in body of architecture.

```
entity  Adder4 is
        port (A, B: in bit_vector (3 downto 0); Ci : in bit;
              S: out bit_vector(3 downto 0); CO : out bit);
End Adder4;

architecture Structure of Adder4 is
component FullAdder
        port (X, Y, Cin : in bit;
              Cout, Sum : out bit);
end component;

begin
        FA0: FullAdder port map (A(0), B(0), Ci, C(1), S(0));
        FA1: FullAdder port map (A(1), B(1), C(1), C(2), S(1));
        FA2: FullAdder port map (A(2), B(2), C(2), C(3), S(2));
        FA3: FullAdder port map (A(3), B(3), C(3), CO, S(3));
End Structure;
```

Set Reset Latch:



R	S	Q	Action
0	0	Last state	No change
0	1	1	SET
1	0	0	RESET
1	1	?	Forbidden

In NOR-gate Latch, two NOR gates are used, one of the inputs for NOR gates is S and R respectively, the other two inputs are fed back from the outputs

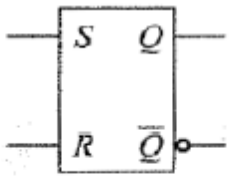
Case1: When R=0 and S=1, the output of upper NOR gate is 1 and lower NOR gate output is 0, hence the outputs **Q=1** $\overline{\mathbf{Q}}$**= 0**

Case2: When R=0 and S=0, the inputs for upper gate are **00** hence the output **Q=1** and inputs for lower gate are **10** the output $\overline{\mathbf{Q}}$**=0** (outputs of Flip flop did not change)

Case3: When **R=1 and** S=0 the inputs for upper gate are **10** hence the output **Q=0** and inputs **for** Lower gate are **00** hence the output $\overline{\mathbf{Q}}$ **=1**

Case4: when R=1 and S=1, the output of both NOR gates become **zero,** which violates the basic definition of flip flop($\overline{Q}$ should be complement of Q), hence R=1 and S=1 input combination is violated in NOR latch

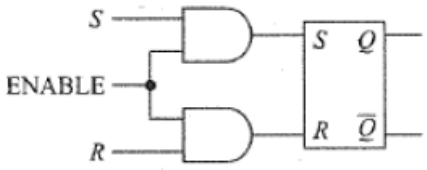
Symbol of SR Latch:



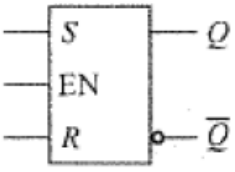
Logic symbol

Gated SR Latch:

Logic Symbol



(a) Logic diagram



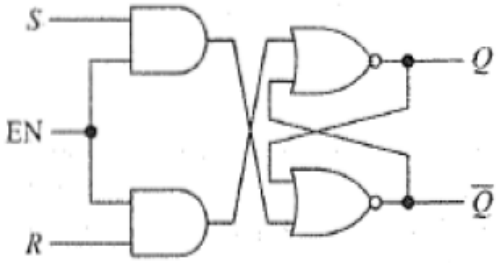
Operation:

- When ENABLE=HIGH, outputs are dependent on R and S, and Latch is said to be enabled.
- When ENABLE=LOW, Latch is disabled, outputs do not change with change in inputs R and S, Latch holds the previous state.

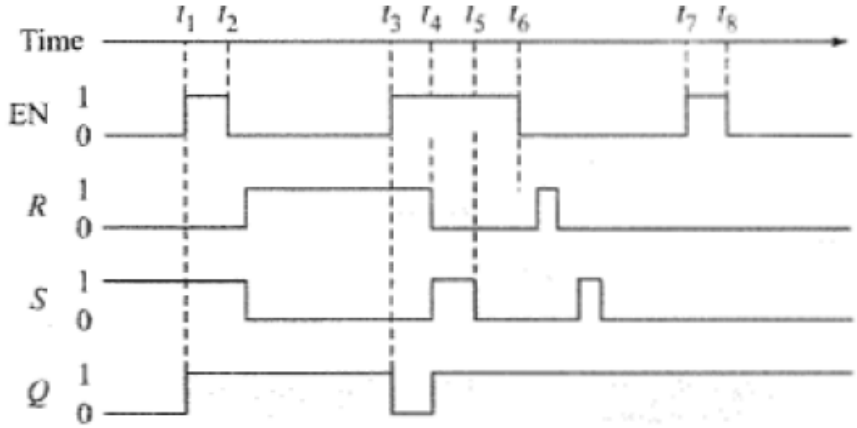
Truth table of Clocked RS Flip-Flop:

EN	S	R	Q_{n+1}
1	0	0	Q_n (no change)
1	0	1	0
1	1	0	1
1	1	1	? (Illegal)
0	X	X	Q_n (no change)

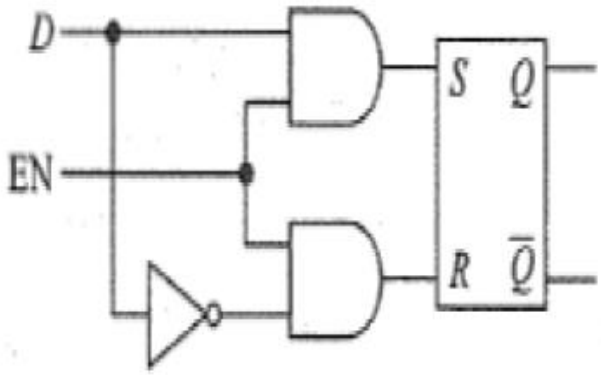
Logic Circuit:

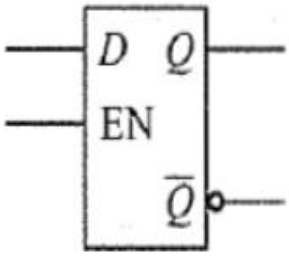


Timing Diagram of Clocked RS Flip-Flop:



Gated D Latch





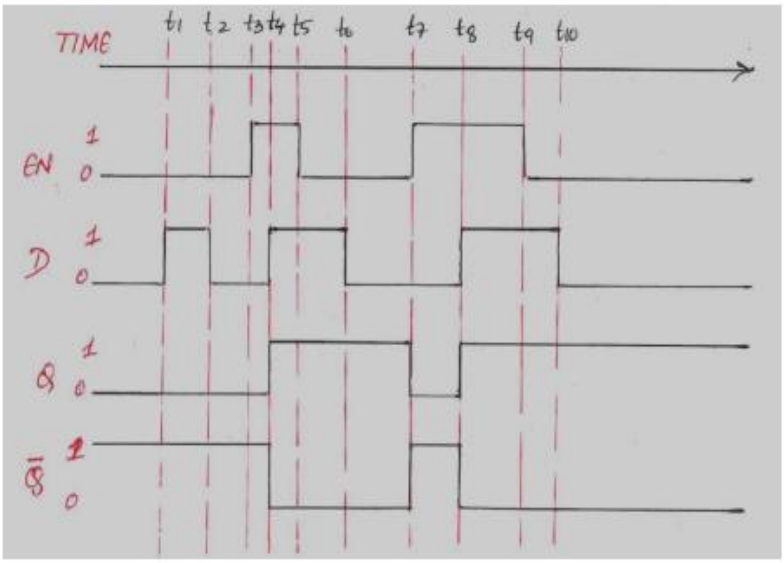
Operation:

- When EN=0, flip flop is disabled, irrespective of D input, flip flop remains in previous state(Q=Last state)
- When EN=1, flip-flop is transparent i.e., output depends on D (when D=0 Q=0, When D=1 Q=1)

Truth table of D-Flip-Flop:

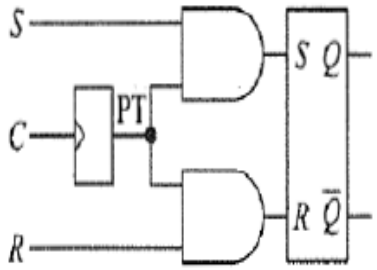
EN	D	Q_{n+1}
0	X	Q_n (last state)
1	0	0
1	1	1

Timing Diagram of Clocked D-Flip-Flop:

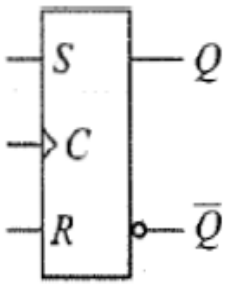


Positive Edge triggered SR Flip-Flop:

Logic Diagram:



Symbol:



The Clock (c) is applied to positive pulse forming circuit, pos
to gated RS Flip-Flop. The small triangle inside the symbol indicates that Q changes during positive edge of the clock.

Operation:

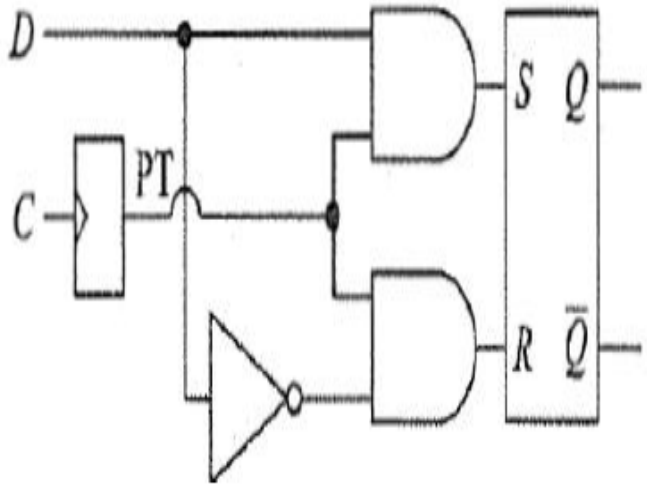
Case 1: When C=0 , output of both And gates are 0, hence flip-flop output Q_{n+1}=Last state.
Case 2: When C=1 S=0 R=1, output of upper AND gate is 0, and lower AND gate output is 1, hence flip flop output Q_{n+1}=0
Case3: When C=1 S=1 R=0, output of upper AND gate is 1, and lower AND gate output is 0, hence flip flop output Q_{n+1}=1
Case 4: When C=1 S=1 R=1, output of upper AND gate is 1, and lower AND gate output is 1, hence flip flop output Q_{n+1}=forbidden

Truth Table of Positive Edge-Triggered RS Flip-Flop:

C	S	R	Q_{n+1}	Action
↑	0	0	Q_n	No change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	?	Illegal

(c) Truth table

3.20 Positive Edge triggered D-Flip Flop:



D Flip-Flop has only one input (D) ,when clock C=0 the output **Q** remains in the last state, when Clock C is positive going(↑) output **Q** depends on **D**

Case1: when C=0 , outputs of both AND gates are zero, hence Q_{n+1} remains in the last state

Case 2: When C=1, D=0, output of upper AND gate =0 and output of Lower AND gate=1 hence Flip-flop output $Q_{n+1}=0$

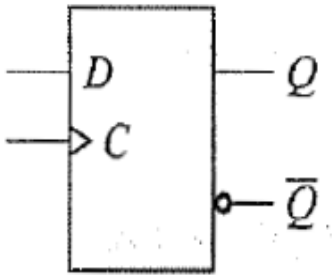
Case3: When C=1, D=1, output of upper AND gate =1 and output of Lower AND gate=0, hence Flip-flop output $Q_{n+1}=1$

Truth table of positive-Edge Triggered D-Flip Flop:

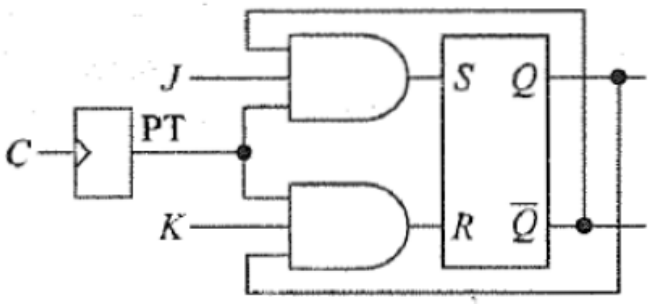
C	D	Q_{n+1}
0	X	Q_n (last state)
↑	0	0
↑	1	1

(c) Truth table

Symbol



3.21 Positive Edge Triggered JK flip Flop:



In Positive-Edge Triggered JK Flip-Flop, inputs are J and K the output Q is fed back to the Lower AND gate, and output $\overline{Q}$ is fed back to upper AND gate.

Operation:

Case1: when C=0, outputs of both AND gates are 0 hence, the output Q_{n+1} remains in last state.

Case2: When C is 1, J=0 K=0 both AND gate outputs are 0, hence Flip flop output Q_{n+1} remains in last state Q_n

Case3: When C is 1, J=0 K=1, Q_n=1 upper AND gate output is zero and Lower AND gate output is 1, hence Flip flop output Q_{n+1} is 0

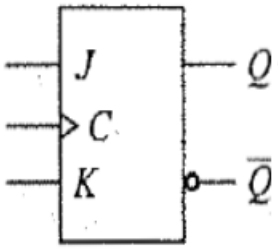
Case4: When C is 1, J=1 K=0, Q_n=0 upper AND gate output is 1 and Lower AND gate output is 0, hence Flip flop output Q_{n+1} is 1

Case 5: When C=1, J=1, K=1 the output $Q_{n+1}=\overline{Q_n}$ (outputs Toggle)

Truth Table of Positive Edge triggered JK Flip-Flop:

C	J	K	Q_{n+1}	Action
↑	0	0	Q_n (last state)	No change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	$\overline{Q}_n$ (toggle)	Toggle

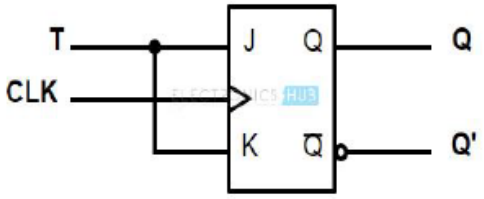
Symbol:



T Flip-Flop:

- T Flip-Flop is also called as Toggle Flip-Flop.
- T Flip-Flop has only one input.

Symbol:

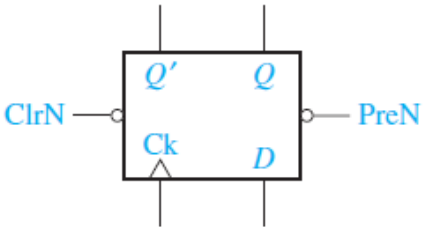


Truth Table:

CLK	T	Q_{n+1}
0	×	Q_n
↑	0	Q_n
↑	1	$\overline{Q_n}$

Flip-Flop with additional Inputs:

In a Flip-Flop, **preset** and **clear** inputs are additional inputs, these inputs are also called as asynchronous inputs since their operation does not depend on the clock.

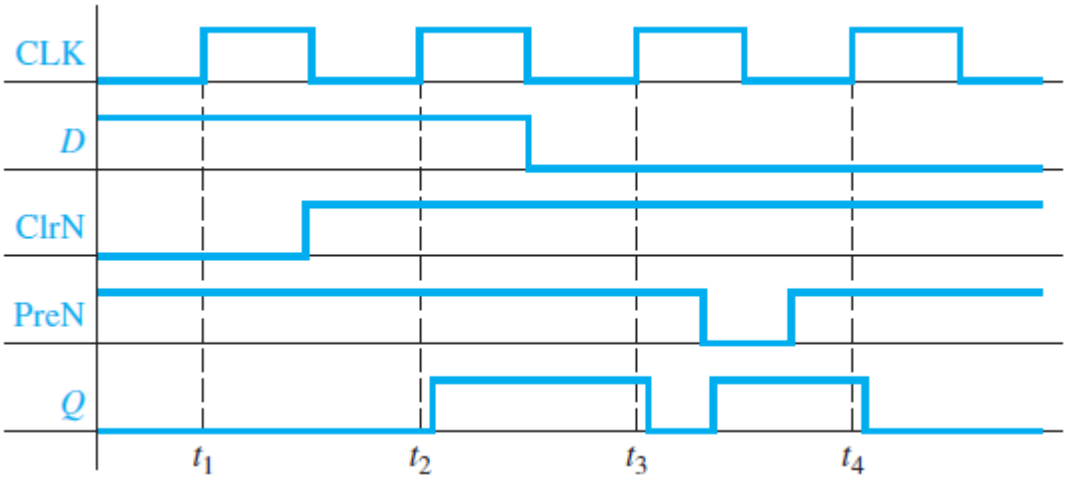


(a)

Ck	*D*	PreN	ClrN	Q^+
x	x	0	0	(not allowed)
x	x	0	1	1
x	x	1	0	0
↑	0	1	1	0
↑	1	1	1	1
0,1,↓	x	1	1	*Q* (no change)

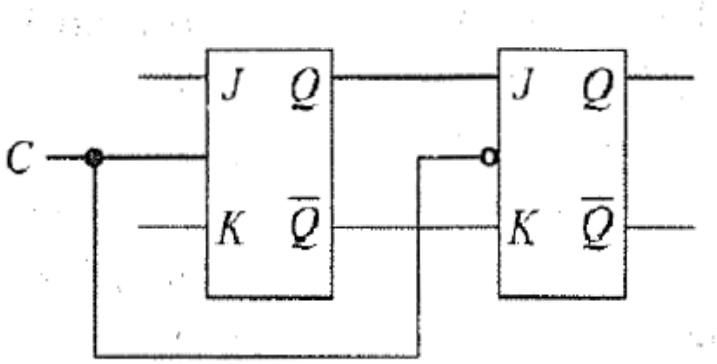
(b)

As shown in the above figure, the small circles on the inputs preN and clrN indicate that a logic 0 is required to clear or set the flip-flop, this type of inputs is often referred to as **active-low** inputs because a low voltage or logic 0 will activate the clear or preset inputs. A logic 0 applied to clear will reset the flip flop to Q=0 and a 0 applied to preset will set the flip flop to Q=1.



The above timing diagram depicts the operation of clear and preset inputs, at time t_1 , clrN=0 holds the Q output at 0, so the rising edge of the clock is ignored, At t2 and t3 normal state changes occur because clrN and preN are both 1, then Q is set to 1 by preN =0 but Q is cleared at t4 by the rising edge of the clock because D=0 at that time.

JK Master Slave Flip- Flop:



- Two flip-flops are used in JK master Slave Flip-Flop, first Flip-Flop is **Master,** Second Flip-Flop is **Slave,** Master is Positive Level-triggered Flip-Flop and Slave is Negative Level-Triggered Flip-Flop. Output of Master Flip flop is depends on inputs J and K when Clock is positive, output of master is connected to Slave, hence Slave follows the Master when clock is negative.

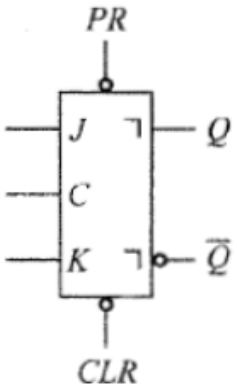
Case 1: when clock C=1, and inputs J=K=0 , master output remains in last state, slave follows the master when C=0, hence output Q remains in last state

Case2: When C=1, J=1 and K=0, master output is 1, slave follows the master when clock =0, hence Q=1

Case 3: when C=1, J=0, K=1 master output is 0, J and K inputs of slave are 0 and 1, when C=0, output of slave Q =0

Case 4: When C=1 J=1, K=1, Master Toggles, slave also toggles when clock goes Low

Symbol

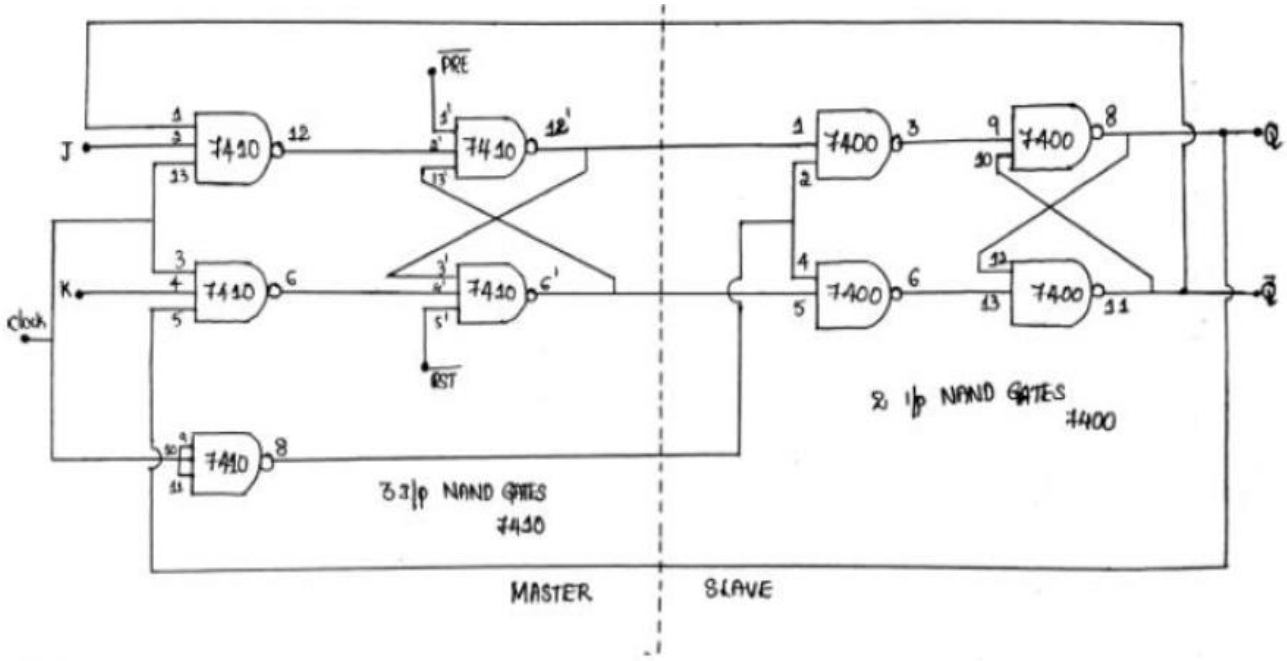


Truth Table:

C	J	K	Q_{n+1}	Action
⊓	L	L	Q_n	No change
⊓	L	H	L	RESET
⊓	H	L	H	SET
⊓	H	H	$\overline{Q}_n$	Toggle

The symbol ┐ appearing next to Q indicates Postponed Output. The master output is dependent on inputs J and K while clock is high, the state of master is shifted to slave when clock goes Low.

Draw NAND-NAND gate implementation of JK master slave Flip-Flop



Important Questions

1. Write VHDL program for 4 to 1 Multiplexer using conditional signal assignment.
2. Write a VHDL program for 4-bit Binary Adder.
3. Explain working principle of gated SR Latech.
4. With circuit diagram and truth table, explain the working of positive edge triggered JK Flip-Flop.
5. Explain working principle of D flip-flop.
6. Define Race-Around condition in JK flip-flop

   In JK Flip-flop, when J=1 and K=1 and clock is positive edge output Q and Qb toggle, this phenomenon is called as Race around condition.
7. Briefly explain about additional inputs of flip-flop with timing diagram.
8. Explain JK master slave flip flop and draw Nand-Nand gate implementation of JK master slave flip-flop.