Fundamentals Of Computer Organization (cc-201)

UNIT-01

Logic circuits and components of Digital computers

What is Computer Organization?

- COMPUTER ORGANIZATION AND ARCHITECTURE provides in-depth knowledge of
  - Internal working,
  - structuring, and
  - implementation of a computer system.

- **COMPUTER ORGANIZATION**:

  Computer organization is concerned with the way the hardware components operate and the way they are connected together to form the computer system. The various components are assumed to be in place and the task is to investigate the organizational structure to verify that the computer parts operate as intended.

COMPUTER ARCHIETECTURE	COMPUTER ORGANIZATION
what computer does?	How computer implements?
Functional behavior	Structural behavior
Deals with high level design issue	Deals with Low level design issue
Designing-first approach	Designing-second approach
Instruction set	Circuit design signals

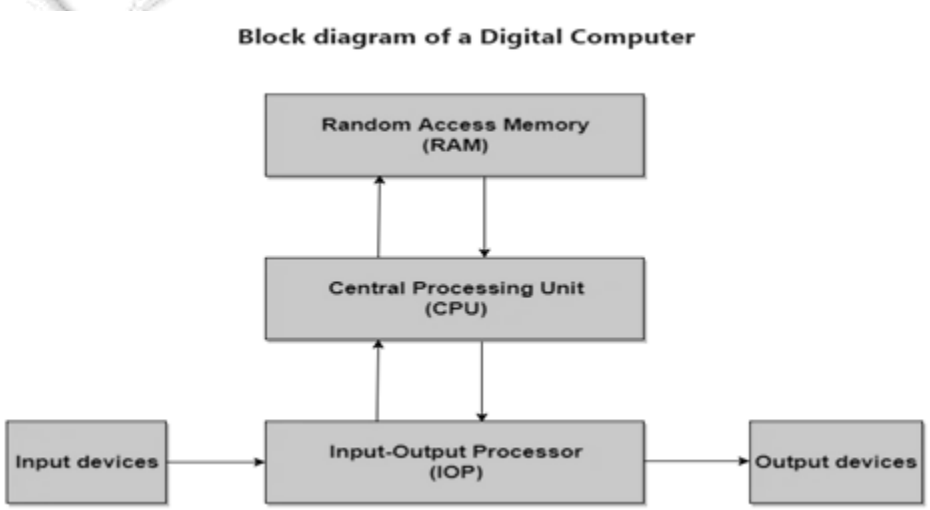
Digital Computers

- Digital computer is most commonly used type of computer.
- A computer that performs calculations and logical operations with quantities represented as digit, usually in the binary number system.
- Digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a **bit.**
- Information is represented in digital computers in groups of bits. By using various coding techniques, groups of bits can be made to represent not only binary numbers but also other discrete symbols, such as decimal digits or letters of the alphabet.

- The main three **components** of digital computer are,
  - **Input** : The user gives a set of input data.
  - **Processing** : The input data is processed by well defined and finite sequence of steps.
  - **Output**: Some data available from the processing step are output to the user.

Structure of a computer system

- The structure of a computer system as being composed of the following components:
- **ALU**: The **Arithmetic-Logic unit** that performs the computer's computational and logical functions.
- **RAM**: Temporary Memory; more specifically, the computer's main, or fast, memory, also known as **Random Access Memory(RAM).**
- **Control Unit**: This is a component that directs other components of the computer to perform certain actions, such as directing the fetching of data or instructions from memory to be processed by the ALU.
- **Man-machine interfaces**; i.e. input and output devices, such as keyboard for input and display monitor for output.



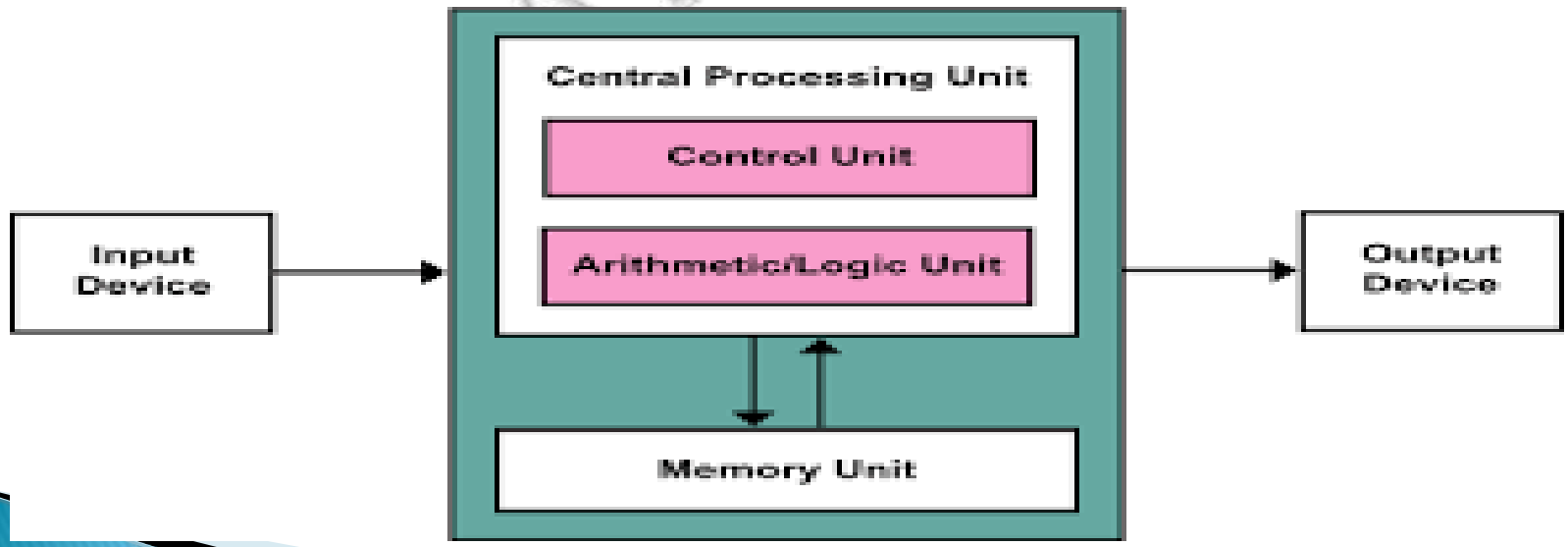
Central Processing Unit

- The central processing unit (CPU) is the brain of the computer.
- The CPU in a computer is usually a single chip. It organizes and carries out instructions that come from either the user or from the software.
- The CPU has three important subunits.
  1. Arithmetic-Logic unit
  2. Control Unit
  3. Memory Unit
- The heart of every computer is an Arithmetic Logic Unit (ALU).
- This is the part of the computer which performs arithmetic operations on numbers, e.g. addition, subtraction, etc. In this lab you will use the Verilog language to implement an ALU having 10 functions. Use of the case structure will make this job easy.

1. **Arithmetic-Logic Unit:**

- An **arithmetic logic unit (ALU)** is a digital circuit used to perform arithmetic and logic operations.
- It represents the fundamental building block of the **central processing unit (CPU)** of a computer.

- **ALU** is responsible to perform the operation in the computer.
- The basic operations are implemented in hardware level.
- The ALU is typically designed in such a way that it has a direct input and output access to the processor main memory –the random access memory (RAM).
- Now, the input and outputs flow along an electronic path that is known as a bus.
- ALU is having collection of operations:
  1. Arithmetic operations
  2. Logic operations
  3. Bit shifting Operations



OPERATIONS CARRIED OUT BY THE ALU :

- **Arithmetic Operations**: This is basically addition and subtraction. Addition can be substituted for multiplication and subtraction for division.
- **Logical Operations:** This include NOR, NOT, OR, XOR etc.
- **Bit shifting Operations:** This entails shifting the position of bits by a certain number of places to either the right or left, which is considered a multiplication operation.

Control Unit:

- The control unit (CU) is a **component** of the central processing unit of the computer system that **controls the operations** of the processor.
- It informs the **arithmetic and logic unit**, the computer's **main memory** and the **output** and **input devices** how to respond to the command that have been sent to the processor.
- The control unit is taken to be the processor brain because it issues orders to everything and ensure that the best results are produced.

- The functions of the Control Unit include:
  - Interprets instructions, regulate and control processor timing.
  - Directs data flow through different components of the CPU.
  - Handle tasks such as decoding, fletching, executing the command and storing results.
  - Sends and receives control signals from other computer devices.
  - Directs sequential data flow.
  - Interprets commands and instructions.

3. Memory Unit:

- This unit can store instructions, data, and intermediate results. This unit supplies information to other units of the computer when needed. It is also known as internal storage unit or the main memory or the primary storage or Random Access Memory (RAM).
- Its size affects speed, power, and capability. Primary memory and secondary memory are two types of memories in the computer.
- Functions of the memory unit are :–

  – It stores all the data and the instructions required for processing.

  – It stores intermediate results of processing.

  – It stores the final results of processing before these results are released to an output device.

  – All inputs and outputs are transmitted through the main memory.

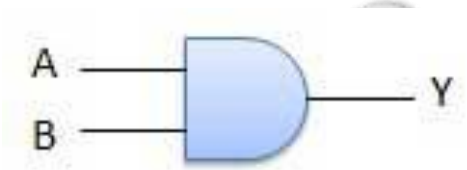
Logic Gates

- Logic gates are the basic **building blocks** of any digital system.
- The logic gates are the **main structural part** of a digital system.
- Logic Gates are a **block of hardware** that produces **signals of binary 1 or 0** when input logic requirements are satisfied.
- Each gate has a distinct **graphic symbol**, and its operation can be described by means of **algebraic expressions.**
- The seven basic logic gates includes: **AND, OR, XOR, NOT, NAND, NOR, and XNOR.**
- The relationship between the input-output binary variables for each gate can be represented in tabular form by a **truth table.**
- Each gate has one or two binary input variables designated by A and B and one binary output variable designated by x.
- It is an electronic circuit having one or more than one input and only one output.
- The relationship between the input and the output is based on a **certain logic.**

Standard Logic Gates

1. AND Gate

Logic Diagram



Y=A.B

Truth Table

INPUT		OUTPUT
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate:

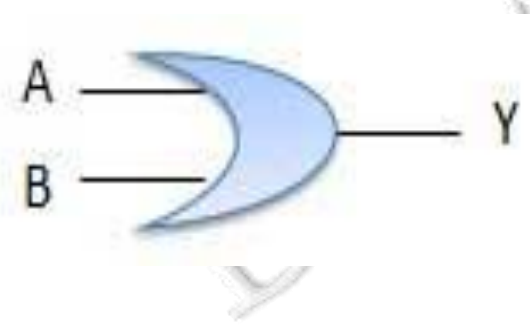
- The AND gate produces the AND logic function, that is,
- The output is 1 if input A and input B are both equal to 1; otherwise the output is 0.
- The algebraic symbol of the AND function is the same as the **multiplication** symbol of ordinary arithmetic.
- We can either use a **dot** between the variables or concatenate the variables without an operation symbol between them.
- AND gates may have more than two inputs

2. OR Gate

Truth Table

INPUT		OUTPUT
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Logic Diagram



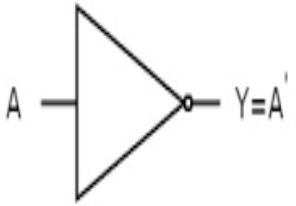
Y=A+B

OR Gate:

- The OR gate produces the inclusive-OR function;
- The output is 0 if input A or input B both inputs are 0; otherwise, the output is 1.
- The algebraic symbol of the OR function is +, similar to arithmetic **addition**.
- OR gates may have more than two inputs, and by definition, the output is 1 if any input is 1.

3. Not Gate

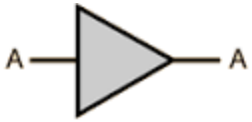
Logic Diagram



Truth Table

A	Y=A’
0	1
1	0

4. Buffer Gate



Buffer

In	Out
0	0
1	1

Not Gate (INVERTER):

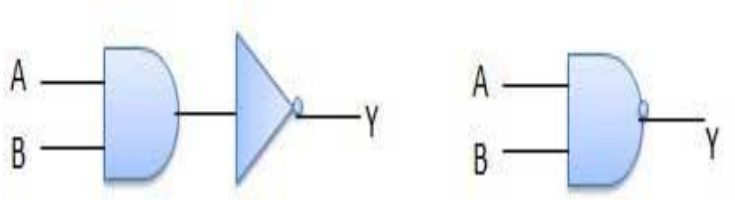
- The inverter circuit inverts the logic sense of a binary signal. It produces the NOT, or complement, function.
- The algebraic symbol used for the logic complement is either a prime or a bar over the variable symbol.

Buffer Gate:

- The Buffer circuit gives binary signal. It produces the same signal as output.

5. NAND Gate

Logic Diagram



$Y = \overline{AB}$

Truth Table

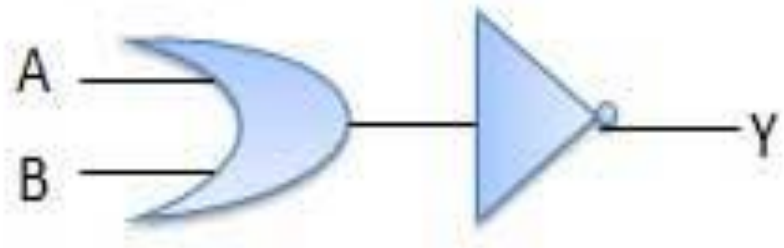
Input A	Input B	Output $Y = \overline{AB}$
0	0	1
1	0	1
0	1	1
1	1	0

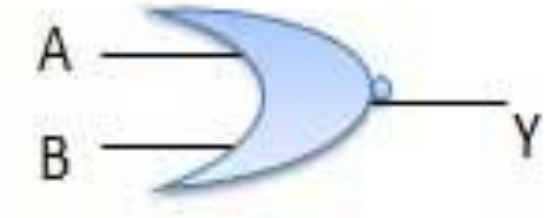
NAND Gate:

- A NOT-AND operation is known as NAND operation. It has n input (n >= 2) and one output.
- The NAND function is the complement of the AND function, as indicated by the graphic symbol, which consists of an AND graphic symbol followed by a small circle.
- The title NAND is derived from the abbreviation of NOT-AND.
- The NAND or "Not AND" function is a combination of the two separate logical functions, the AND function and the NOT function in series.
- The logic NAND function can be expressed by the Boolean expression of, $Y = \overline{A.B}$

6. NOR Gate

Logic Diagram





$Y = \overline{A + B}$

Truth Table

Input A	Input B	Output $Y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

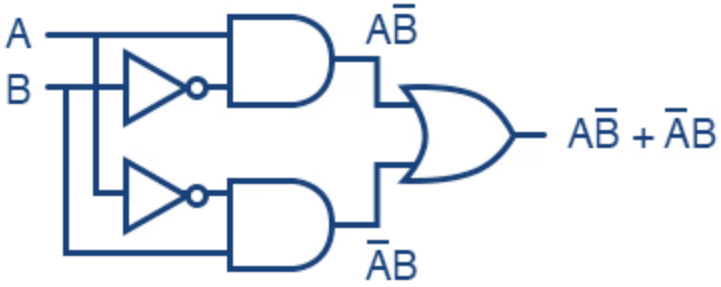
NOR Gate:

- A NOR gate represents an OR gate followed by an inverter. It has 2 or more input signals.
- It produces a 1 output when all the inputs are 0.
- It produces a 0 output when any or all of the inputs are 1.
- It therefore also acts as a negative (**Bubbled**) AND gate.

7. XOR Gate

Logic Diagram

. . . is equivalent to . .



$A \oplus B = A\bar{B} + \bar{A}B$

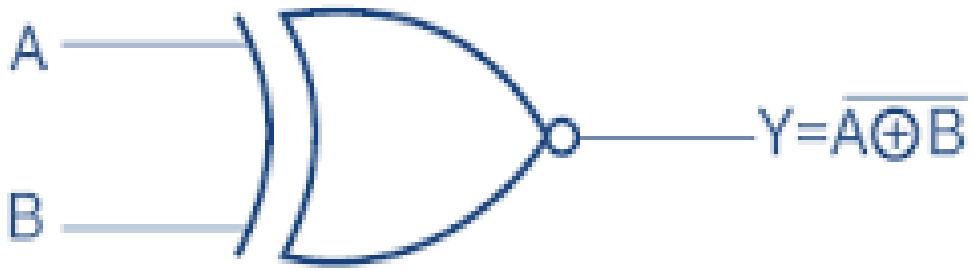
Truth Table

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

XOR Gate:

- An XOR gate (also known as an EOR, or EXOR gate) – pronounced as Exclusive OR gate
- XOR or Ex-OR gate is a special type of gate. It can be used in the half adder, full adder and subtraction.
- The exclusive-OR gate is abbreviated as EX-OR gate or sometime as X-OR gate.
- If two inputs are similar output become 0.
- If two inputs are dissimilar output become 1.
- digital logic gate that gives a true (i.e. a HIGH or 1) output when the number of true inputs is odd.
- **XOR represents the inequality function, i.e., the output is true if the inputs are not alike otherwise the output is false.**
- Any expression following the AB' + A'B form (two AND gates and an OR gate) may be replaced by a single Exclusive-OR gate.

8. XNOR Gate



$$Y = (\overline{A \oplus B}) = (A.B + \overline{A}.\overline{B})$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

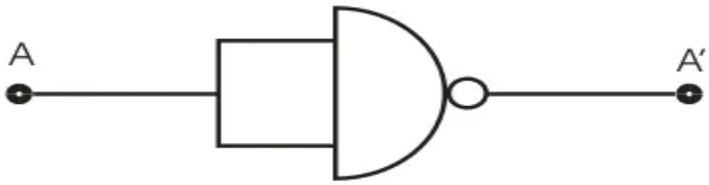
XNOR Gate:

- The XNOR gate (sometimes XORN'T, ENOR, EXNOR or NXOR and pronounced as Exclusive NOR) is a digital logic gate whose function is the logical complement of the exclusive OR (XOR) gate.
- An XNOR Gate is a type of digital logic gate that receives two inputs and produces one output.
- Both inputs are treated with the same logic, responding equally to similar inputs.
- Sometimes referred to as an "Equivalence Gate,"
- **The gate's output requires both inputs to be the same to produce a high output.**
- If both inputs are 0, the gate will produce a 1.
- If both inputs are 1, the gate will also produce a 1.
- However, if either input differs from the other, the gate will output a 0.

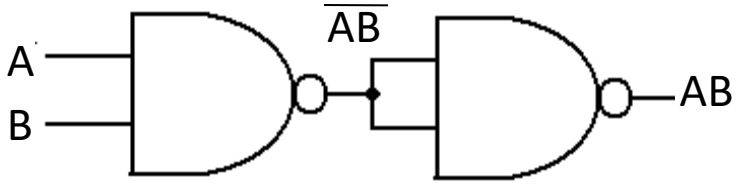
Universal Gates

- A **universal gate** is a logic gate which can implement any Boolean function without the need to use any other type of logic gate.
- The NOR gate and NAND gate are universal gates. This means that you can create any logical Boolean expression using only NOR gates or only NAND gates.
- **NAND Gate As A Universal Gate:**

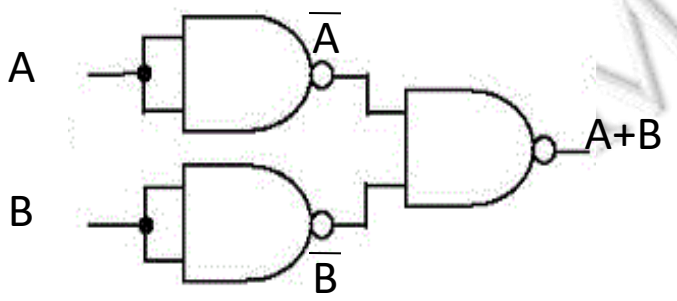
NOT Gate



AND Gate



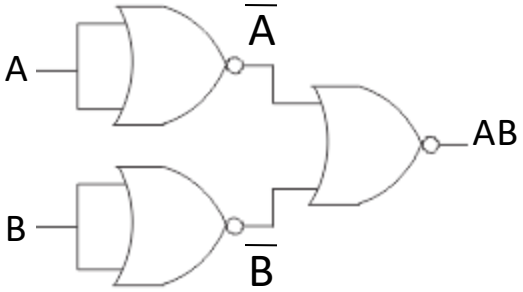
OR Gate



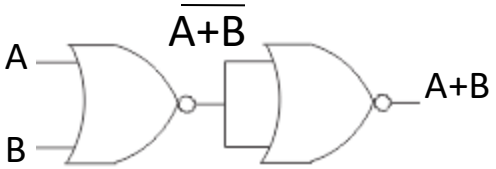
NOR Gate As A Universal Gate:

NOT Gate

AND Gate



AND Gate



Boolean Algebra

- Boolean algebra can be considered as an algebra that deals with binary variables and logic operations. Boolean algebraic variables are designated by letters such as A, B, x, and y. The basic operations performed are AND, OR, and complement.
- The Boolean algebraic functions are mostly expressed with binary variables, logic operation symbols, parentheses, and equal sign.
- It reduce complex expression to simpler expression that requires less number of gates which reduces the cost, power and space.
- It works with binary digits (bits): 0 and 1. While 1 represents true, 0 represents false. Computers can perform simple to extremely complex operations with the use of Boolean algebra. Boolean algebra and Boolean operations are the basis for computer logic.

- The basic Laws of Boolean Algebra are as follows:
- **Commutative Law** : It states that the interchanging of the order of operands in a Boolean equation does not change its result. For example:
  - OR operator → A + B = B + A
  - AND operator → A * B = B * A
- **Associative Law**: It states that the AND operation are done on two or more than two variables. For example:
  A * (B * C) = (A * B) * C
- **Distributive Law**: It states that the multiplication of two variables and adding the result with a variable will result in the same value as multiplication of addition of the variable with individual variables. For example:
  A + (B.C) = (A + B) (A + C).
- **Annulment law**: According to this law a variable ANDed with 0 gives 0, while a variable ORed with 1 gives 1.
  A.0 = 0
  A + 1 = 1
- **Identity law**: According to this law a variable remain unchanged if it is ORed with 0 or ANDed with 1.
  A.1 = A
  A + 0 = A
- **Idempotent law**: According to this law a variable remain unchanged when it is ORed or ANDed with itself.
  A + A = A
  A.A = A
- **Complement law**: If complement is added to a variable it gives one, if a variable is multiplied with its complement it result in 0.
  A + A' = 1
  A.A'= 0
- **Double negation law/Inversion law**: The inversion law states that double inversion of a variable results in the original variable itself.
  ((A)')' = A
- **Absorption law**: This law deals with absorbing the similar variable.
  A.(A+B) = A
  A + A.B = A

De Morgan's Law:

- It is also known as De Morgan's theorem, works depending on the concept of Duality.
- Duality states that interchanging the operators and variables in a function, such as replacing 0 with 1 and 1 with 0, AND operator with OR operator and OR operator with AND operator.
- De Morgan stated 2 theorems, which will help us in solving the algebraic problems in digital electronics.
- The De Morgan's statements are:
- "The negation of a conjunction is the disjunction of the negations", which means that the complement of the product of 2 variables is equal to the sum of the compliments of individual variables.
  For example, (A.B)' = A' + B'.
- "The negation of disjunction is the conjunction of the negations", which means that compliment of the sum of two variables is equal to the product of the complement of each variable.
  For example, (A + B)' = A'B'.

- We will simplify this Boolean function on the basis of rules given by Boolean algebra.
- AB + A (B+C) + B (B+C)
- AB + AB + AC + BB + BC {Distributive law; A (B+C) = AB+AC, B (B+C) = BB+BC}
- AB + AB + AC + B + BC {Idempotent law; BB = B}
- AB + AC + B + BC {Idempotent law; AB+AB = AB}
- AB + AC +B {Absorption law; B+BC = B}
- B + AC {Absorption law; AB+B = B}
- Hence, the simplified Boolean function will be B + AC.

- Example: A'+B'+(A+B)'
- A'+B'+A'.B' (**De Morgan's** law(A+B)'=A'.B')
- A'(1+B')+B' (**Annulment law**: 1+B'=1)
- A'+B'

- Example:(A+B).(A+C)=A.1+B.c

Logic Circuits

- Logic circuits can broadly be classified in
  1. Combinational Logic circuits
  2. Sequential Logic circuits

1. **Combinational Logic circuits**:

- A circuit in which different types of logic gates are combined is known as a **combinational logic circuit**.
- The output of the combinational circuit is determined from the present combination of inputs, regardless of the previous input.
- The input variables, logic gates, and output variables are the basic components of the combinational logic circuit.
- There are different types of combinational logic circuits, such as Adder, Subtractor, Decoder, Encoder, Multiplexer, and De-multiplexer.
- Combinational Logic Circuits are made up from basic logic NAND, NOR or NOT gates that are "combined" or connected together to produce more complicated switching circuits.
- The **characteristics** of the combinational logic circuit:

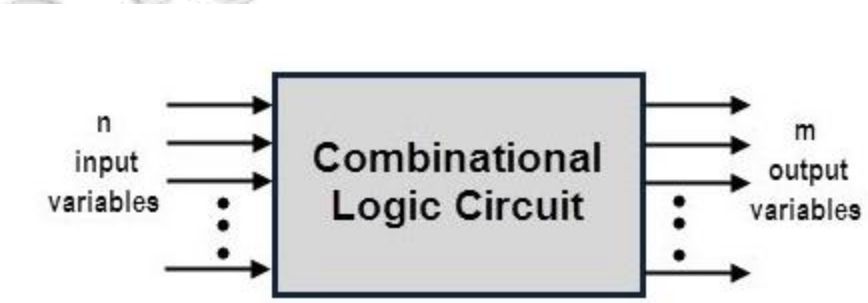
  - At any instant of time, the output of the combinational circuits depends only on the present input terminals.

  - The combinational circuit doesn't have any backup or previous memory. The present state of the circuit is not affected by the previous state of the input.

  - The n number of inputs and m number of outputs are possible in combinational logic circuits.

- The three main ways of specifying the function of a combinational logic circuit are:

1. **Boolean Algebra** – This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False that results in a logic "1" output.
2. **Truth Table** – A truth table defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible combination of input variable that the gate could encounter.
3. **Logic Diagram** – This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol, that implements the logic circuit.

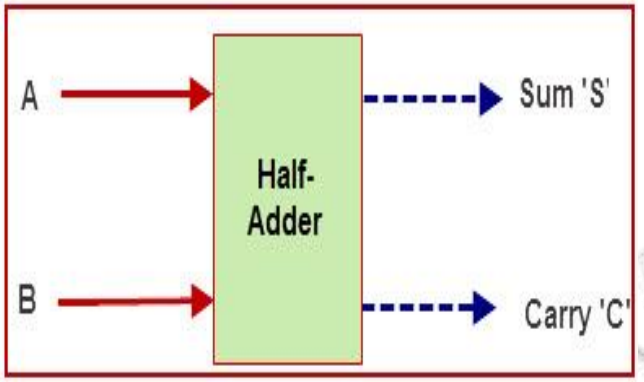


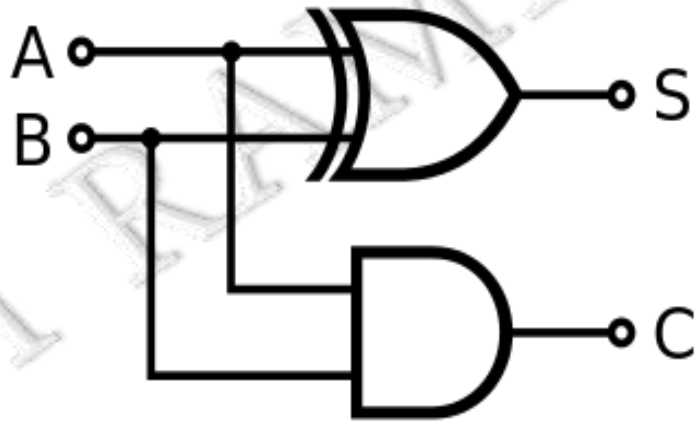
Block Diagram of Combinational circuit

Half Adder

- Half adder is a combinational logic circuit with two inputs and two outputs.
- The half adder circuit is designed to add two single bit binary number A and B.
- It is the basic building block for addition of two single bit numbers.
- This circuit has two outputs carry and sum.
- We can understand the function of a half-adder by formulating a truth table.
- Then the Boolean expression for a half adder is as follows.
- For the **SUM** bit: SUM = A XOR B = A $\oplus$ B
- For the **CARRY** bit: CARRY = A AND B = A.B
- One major disadvantage of the *Half Adder* circuit when used as a binary adder, is that there is no provision for a "Carry-in" from the previous circuit when adding together multiple data bits.

Half Adder





Sum = A XOR B
Carry = A AND B

INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder

- ‘A' and ‘B' are the two inputs, and S (Sum) and C (Carry) are the two outputs.
- The Carry output is '0' unless both the inputs are 1.
- 'S' represents the least significant bit(LSB) of the sum.

Full Adder

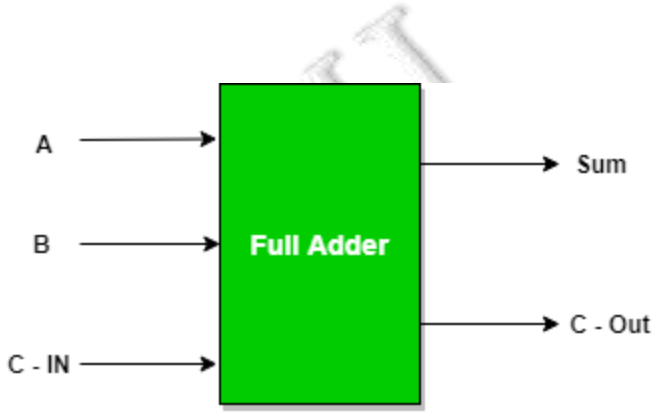
- A **full adder** is a logical circuit that performs an addition **operation** on three one-bit binary numbers.
- It is a three input and two output combinational circuit.
- The full adder is used to add three 1-bit binary numbers A, B, and carry C. The full adder has three input states and two output states i.e., sum and carry.
- The Boolean expression for a full adder is as follows.
- For the **SUM** (S) bit: SUM = (A XOR B) XOR Cin = $(A \oplus B) \oplus Cin$
- For the **CARRY-OUT** (Cout) bit:

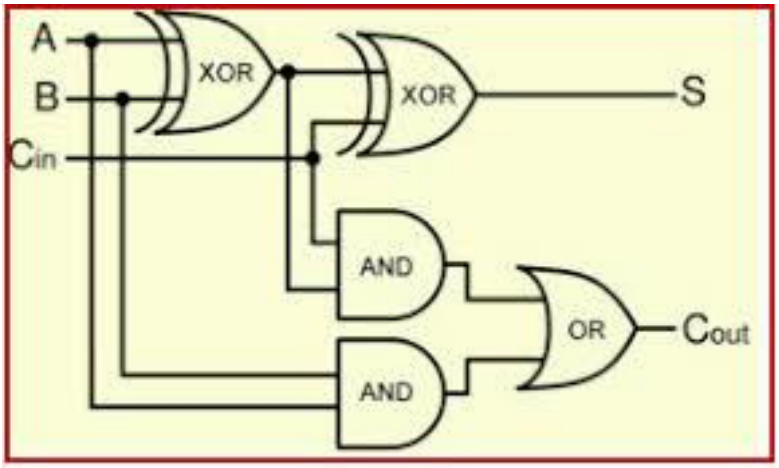
  CARRY-OUT = A AND B OR Cin(A XOR B) = $A.B + Cin(A \oplus B)$
- Specific to digital circuits (binary arithmetic), the addition of two 1's in binary creates a '2', i.e. 10.
- It is **used** in ALU in processor chip to perform arithmetic and logical operations.
- The main difference between the **Full Adder** and **Half Adder** is that a full adder has three inputs. The same two single bit data inputs A and B as before plus an additional *Carry-in* (C-in) input to receive the carry from a previous stage.

Full Adder

Full Adder Truth table

INPUTS			OUTPUTS	
A	B	C_{in}	SUM	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1





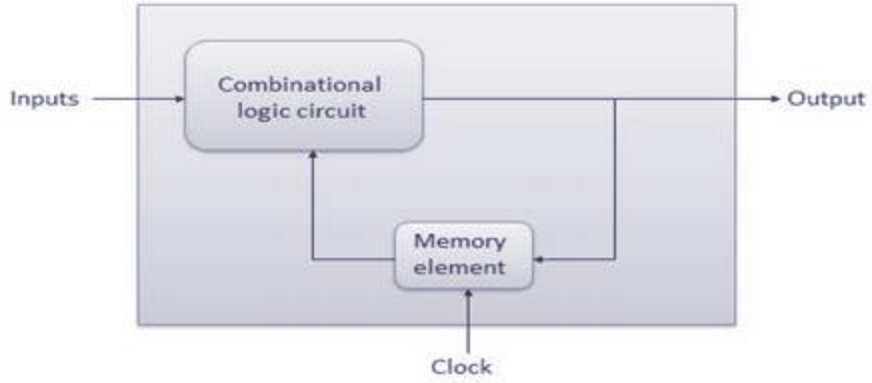
2. Sequential Logic circuits:

- The **combinational circuit does not use any memory.** Hence the previous state of input does not have any effect on the present state of the circuit.
- But **sequential circuit has memory** so output can vary based on input. This type of circuits uses previous input, output, clock and a memory element.

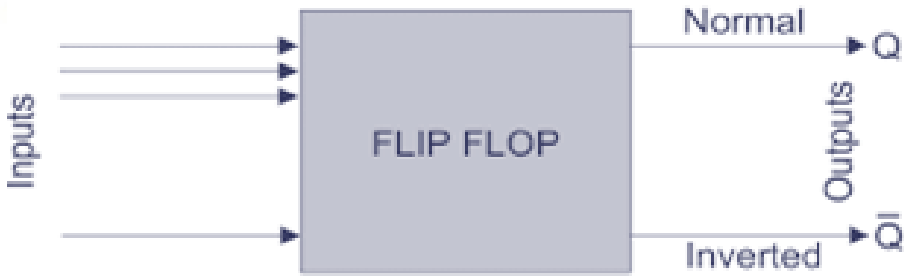
- **Flip flop** is a sequential circuit which generally samples its inputs and changes its outputs only at particular instants of time and not continuously.
- **Computers and calculators** use Flip-flop for their memory.
- A combination of number of flip flops will produce some amount of memory.
- **Flip-flops are used as data storage elements.**
- Flip flop is formed using logic gates, which are in turn made of transistors.
- Flip flop are basic building blocks in the memory of electronic devices.
- Each flip flop can **store one bit of data.**

- Following are different types of Flip – Flops:
  1. SR Flip-Flop
  2. D Flip-Flop
  3. JK Flip-Flop
  4. T Flip-Flop

Block diagram of sequential circuit

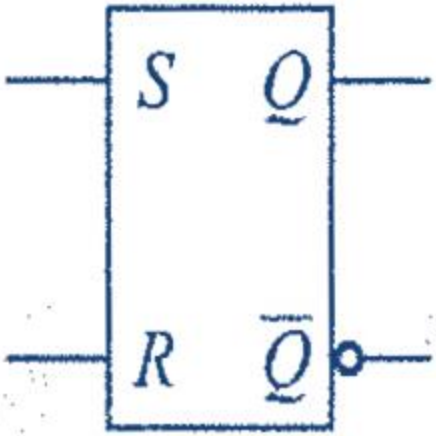


Block diagram of Flip Flop

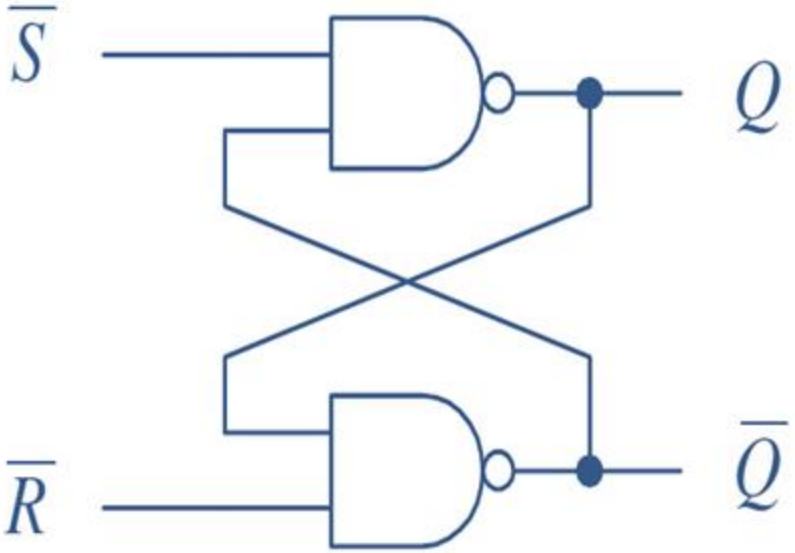


SR flip-flop

- The SR Flip-Flop stands for **"Set-Reset"**. It can be considered as one of the **most basic sequential logic circuit** possible.
- This simple flip-flop is basically a one-bit memory bi-stable device that has two inputs, one which will "SET" the device (meaning the output = "1"), and is labeled **S** and one which will "RESET" the device (meaning the output = "0"), labeled **R** and two outputs Q and Q'.
- The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level "1" or logic "0" depending upon this set/reset condition.
- The SR flip-flop can be made by cross coupling two inverting gates either NOR or NAND gates.
- There is feedback from each output to one of the other NAND gate inputs.



logic symbol



circuit of SR flip – flop

Truth Table

$\overline{S}$	$\overline{R}$	Q	State
1	1	Last State	No Change
0	1	1	Set
1	0	0	Reset
0	0	Invalid	Forbidden

• **An important point about NAND gate is that its dominating input is 0 i.e., if any of its input is Logic '0', the output is Logic '1', irrespective of the other input. The output is 0, only if all the inputs are 1.**

Case 1: R = 1 and S = 1

When both the S and R inputs are HIGH, the output remains in previous state i.e., it holds the previous data.

Case 2: R = 1 and S = 0

When R input is HIGH and S input is LOW, the flip flop will be in SET state. As R is HIGH, the output of NAND gate B i.e., Q becomes LOW. This causes both the inputs of NAND gate A to become LOW and hence, the output of NAND gate A i.e., Q becomes HIGH.

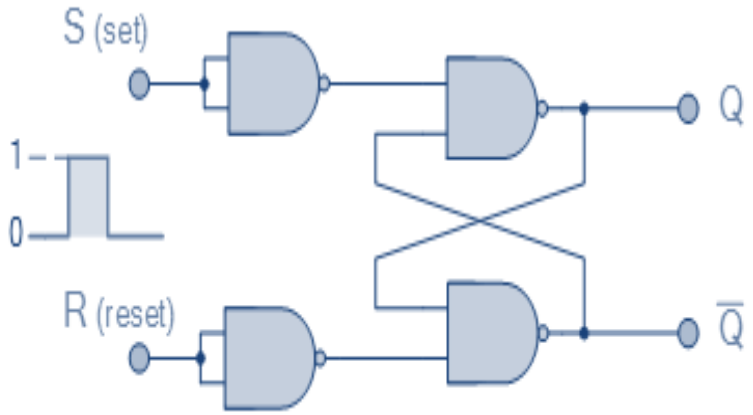
Case 3: R = 0 and S = 1

When R input is LOW and S input is HIGH, the flip flop will be in RESET state. As S is HIGH, the output of NAND gate A i.e., Q becomes LOW. This causes both the inputs of NAND gate B to become LOW and hence, the output of NAND gate A i.e., Q becomes HIGH.

Case 3: R = 0 and S = 0

When both the R and S inputs are LOW, the flip flop will be in undefined state. Because the low inputs of S and R, violates the rule of flip – flop that the outputs should complement to each other. So, the flip flop is in undefined state (or forbidden state).

Clocked SR flip-flop

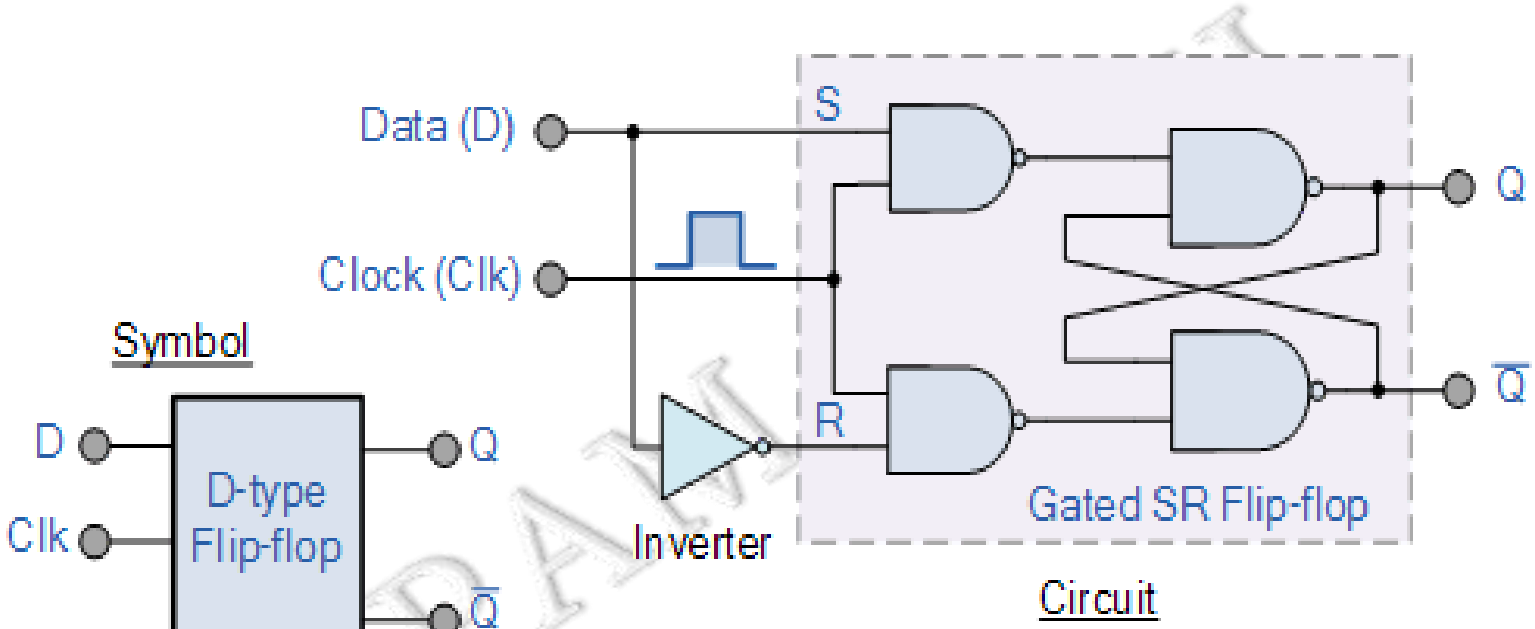


NAND Circuit

Truth Table

S	R	Q	$\overline{Q}$
0	0	No change	
0	1	0	1
1	0	1	0
1	1	X (Invalid)	X

D flip-flop



Truth Table

Clk	D	Q		Description
↓ » 0	X	Q	$\overline{Q}$	Memory no change
↑ » 1	0	0	1	Reset Q » 0
↑ » 1	1	1	0	Set Q » 1

D flip-flop

- The **D-type/ Delay flip-flop is a modified Set-Reset flip-flop with the addition of an inverter** to prevent the S and R inputs from being at the same logic level
- One of the main disadvantages of the basic SR NAND Gate Bistable circuit is that the indeterminate input condition of SET = "0" and RESET = "0" is forbidden.
- This state will force both outputs to be at logic "1", over-riding the feedback latching action and whichever input goes to logic level "1" first will lose control, while the other input still at logic "0" controls the resulting state of the latch.
- The **D Flip Flop** is by far the most important of the clocked flip-flops as it ensures that inputs S and R are never equal to one at the same time.
- The D-type flip flop are constructed from a gated SR flip-flop with an inverter added between the S and the R inputs to allow for a single D (Data) input.
- Then this single data input, labelled "D" and is used in place of the "Set" signal, and the inverter is used to generate the complementary "Reset" input thereby making a level-sensitive D-type flip-flop from a level-sensitive SR-latch as now S = D and R = not D as shown.

JK flip-flop

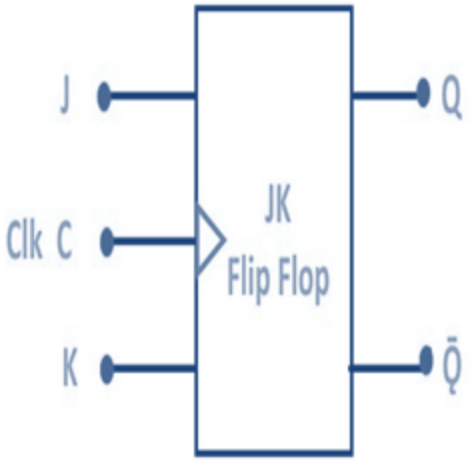
- The **JK Flip Flop** is the most widely used flip flop. It is considered to be a **universal flip-flop** circuit.
- The sequential operation of the JK Flip Flop is same as for the RS flip-flop with the same **SET** and **RESET** input but it suffers from two basic switching problems.

  1. The Set = 0 and Reset = 0 condition (S = R = 0) must always be avoided

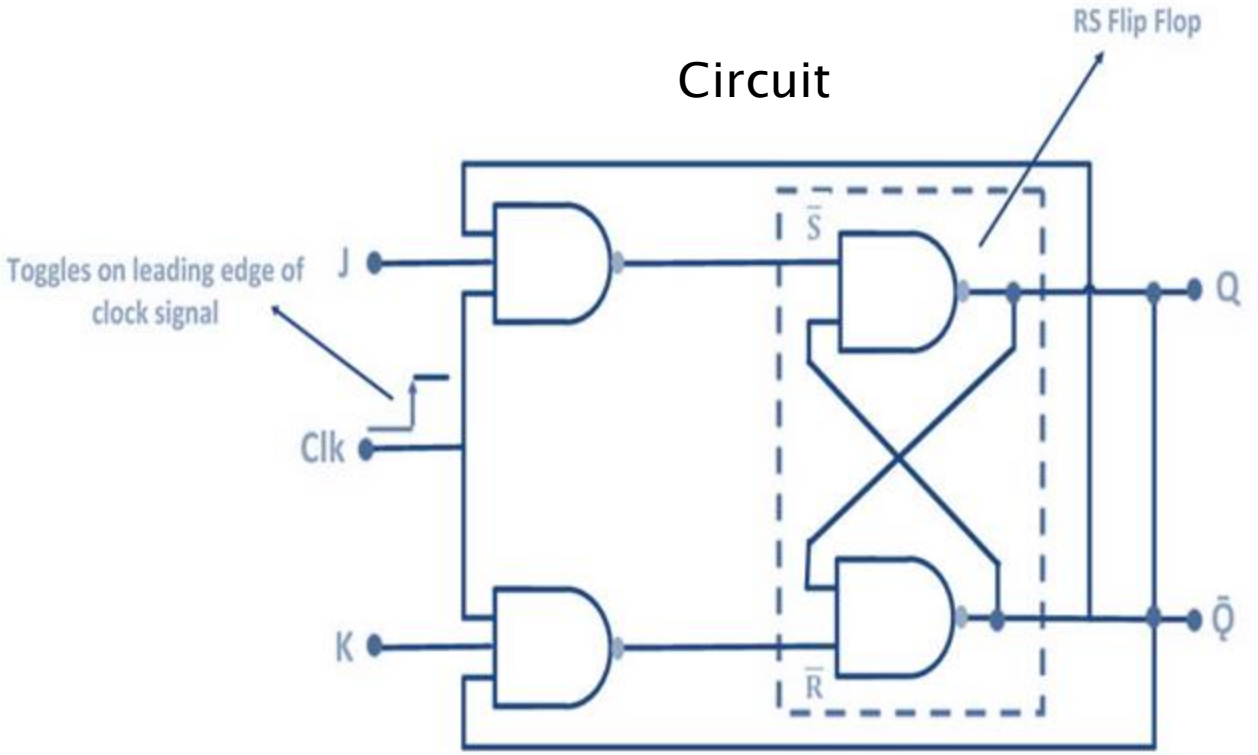
  2. If Set or Reset change state while the enable (EN) input is high the correct latching action may not occur
- Then to overcome these two fundamental design problems with the SR flip-flop design, the **JK flip Flop** was developed.
- The JK Flip Flop name has been kept on the inventor name of the circuit known as **Jack Kilby.**

JK flip-flop

Symbol



Circuit



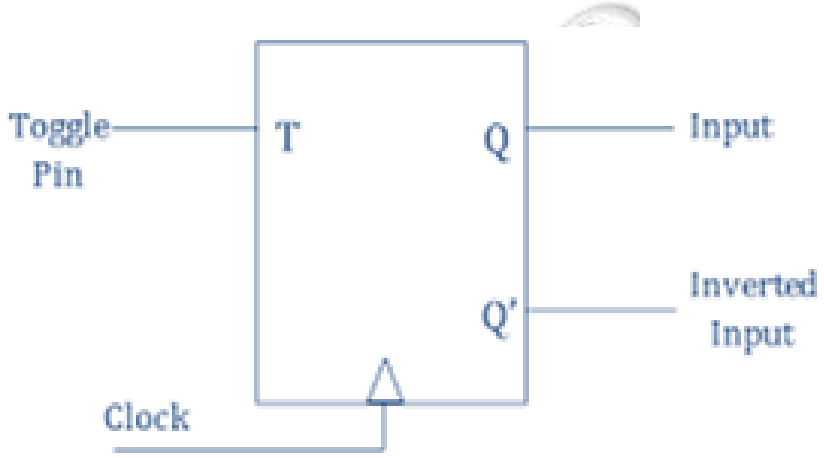
Truth Table

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

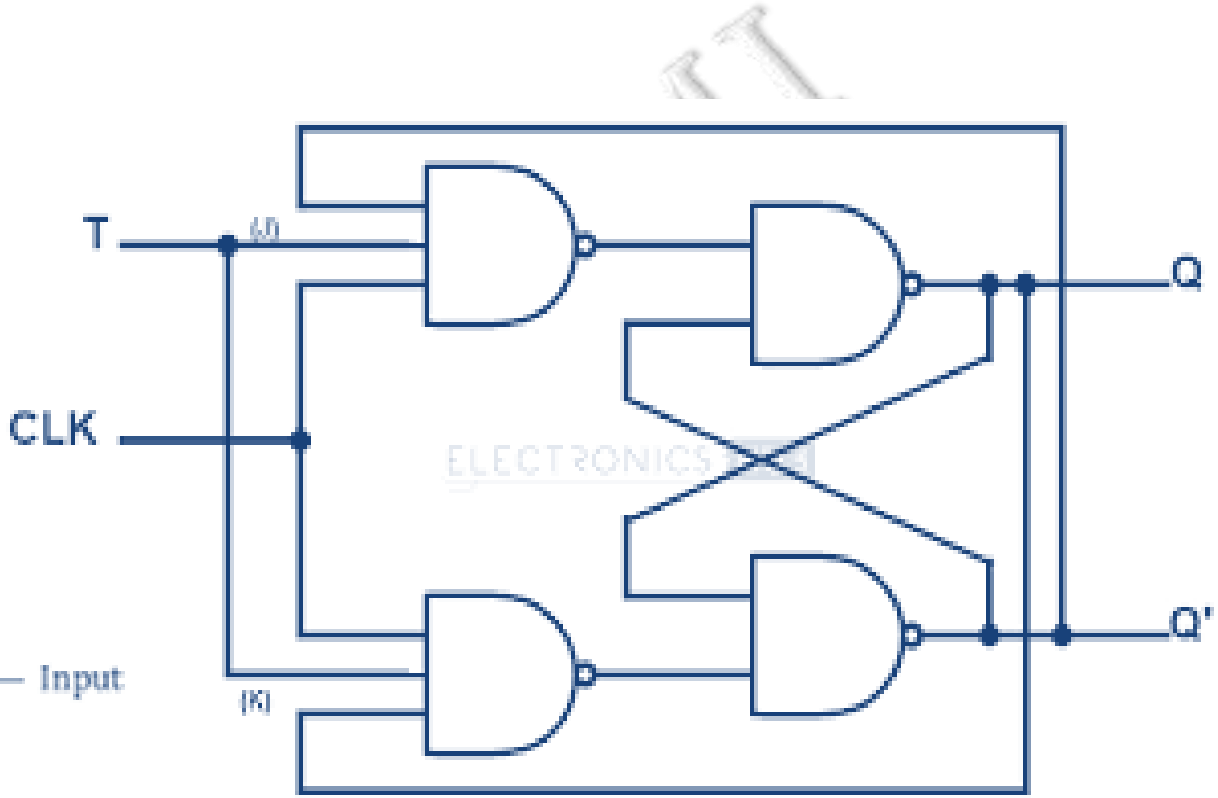
T Flip Flop

Truth Table

Q_n	T	$Q_n + 1$
0	0	0
0	1	1
1	0	1
1	1	0



Symbol: T Flip-flop



Circuit

T Flip Flop

- The **T** or "toggle" **flip-flop** changes its output on each clock edge, giving an output which is half the frequency of the signal to the **T** input. ... It can be made from a J-K **flip-flop** by tying both of its inputs high.
- To avoid the occurrence of intermediate state in SR flip – flop, we should provide only one input to the flip – flop called Trigger input or Toggle input (T). Then the flip – flop acts as a Toggle switch. Toggling means 'Changing the next state output to complement of the present state output'.
- We can design the T flip – flop by making simple modifications to the JK flip – flop.
- The T flip – flop is a single input device and hence by connecting J and K inputs together and giving them with single input called T
- we can convert a JK flip – flop into T flip – flop. So a T flip – flop is sometimes called as single input JK flip – flop.

Construction

- We can construct a T flip – flop by connecting AND gates as input to the NOR gate SR latch. And these AND gate inputs are feedback with the present state output Q and its complement Q' to each AND gate. A toggle input (T) is connected in common to both the AND gates as an input. The AND gates are also connected with common Clock (CLK) signal.
- In the T flip – flop, a pulse train of narrow triggers are provided as input (T) which will cause the change in output state of flip – flop. So these flip – flops are also called Toggle flip – flops.

Comparison of Circuit

Combinational Circuit	Sequential Circuit
The outputs of the combinational circuit depend only on the present inputs.	The outputs of the sequential circuits depend on both present inputs and present state(previous output).
The feedback path is not present in the combinational circuit.	The feedback path is present in the sequential circuits.
In combinational circuits, memory elements are not required.	In the sequential circuit, memory elements play an important role and require.
The clock signal is not required for combinational circuits.	The clock signal is required for sequential circuits.
The combinational circuit is simple to design.	It is not simple to design a sequential circuit.

Assignment:

1. Explain Structure of a computer system with block diagram.
2. Explain different logic Gates.
3. Create all basic gates with NOR Gate.
4. Draw and Explain Combinational Logic circuits.
5. Draw and Explain Sequential Logic circuits.