Computer

A **computer** is an electronic device that receives input, stores or processes the input as per user instructions and provides output in desired format. Computers have become an integral part of our lives because they can accomplish easy tasks repeatedly without getting bored and complex ones repeatedly without committing errors. In this tutorial we will discuss in detail about the different parts of computer that enable it to carry out tasks efficiently and correctly. We will also discuss about microprocessors, the brain of computers, which actually do all the assigned tasks.

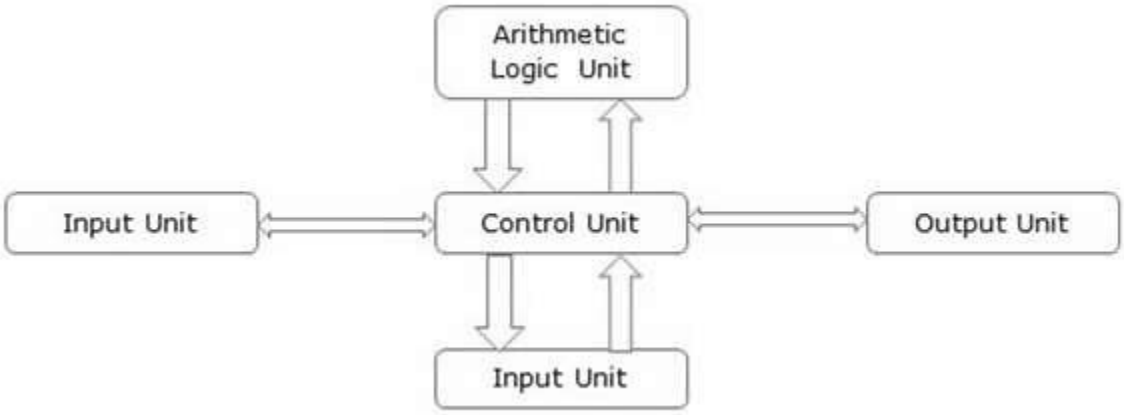
Input-Process-Output Model

Computer input is called **data** and the output obtained after processing it, based on user's instructions is called **information**. Raw facts and figures which can be processed using arithmetic and logical operations to obtain information are called **data**.

The processes that can be applied to data are of two types −

- **Arithmetic operations** − Examples include calculations like addition, subtraction, differentials, square root, etc.
- **Logical operations** − Examples include comparison operations like greater than, less than, equal to, opposite, etc.

The corresponding figure for an actual computer looks something like this −



The basic parts of a computer are as follows −

- **Input Unit** − Devices like keyboard and mouse that are used to input data and instructions to the computer are called input unit.
- **Output Unit** − Devices like printer and visual display unit that are used to provide information to the user in desired format are called output unit.
- **Control Unit** − As the name suggests, this unit controls all the functions of the computer. All devices or parts of computer interact through the control unit.
- **Arithmetic Logic Unit** − This is the brain of the computer where all arithmetic operations and logical operations take place.
- **Memory** − All input data, instructions and data interim to the processes are stored in the memory. Memory is of two types – **primary memory** and **secondary memory**. Primary memory resides within the CPU whereas secondary memory is external to it.

Control unit, arithmetic logic unit and memory are together called the **central processing unit** or **CPU**. Computer devices like keyboard, mouse, printer, etc. that we can see and touch are the **hardware** components of a computer. The set of instructions or programs that make the computer function using these hardware parts are called **software**. We cannot see or touch software. Both hardware and software are necessary for working of a computer.

Functionalities of a Computer

If we look at it in a very broad sense, any digital computer carries out the following five functions −

Step 1 − Takes data as input.

Step 2 − Stores the data/instructions in its memory and uses them as required.

Step 3 − Processes the data and converts it into useful information.

Step 4 − Generates the output.

Step 5 − Controls all the above four steps.

Characteristics of Computer

To understand why computers are such an important part of our lives, let us look at some of its characteristics −

- **Speed** − Typically, a computer can carry out 3-4 million instructions per second.
- **Accuracy** − Computers exhibit a very high degree of accuracy. Errors that may occur are usually due to inaccurate data, wrong instructions or bug in chips – all human errors.
- **Reliability** − Computers can carry out same type of work repeatedly without throwing up errors due to tiredness or boredom, which are very common among humans.
- **Versatility** − Computers can carry out a wide range of work from data entry and ticket booking to complex mathematical calculations and continuous astronomical observations. If you can input the necessary data with correct instructions, computer will do the processing.
- **Storage Capacity** − Computers can store a very large amount of data at a fraction of cost of traditional storage of files. Also, data is safe from normal wear and tear associated with paper.

Advantages of Computers

Following are certain advantages of computers.

High Speed

- Computer is a very fast device.
- It is capable of performing calculation of very large amount of data.
- The computer has units of speed in microsecond, nanosecond, and even the picosecond.

- It can perform millions of calculations in a few seconds as compared to man who will spend many months to perform the same task.

Accuracy

- In addition to being very fast, computers are very accurate.
- The calculations are 100% error free.
- Computers perform all jobs with 100% accuracy provided that the input is correct.

Storage Capability

- Memory is a very important characteristic of computers.
- A computer has much more storage capacity than human beings.
- It can store large amount of data.
- It can store any type of data such as images, videos, text, audio, etc.

Diligence

- Unlike human beings, a computer is free from monotony, tiredness, and lack of concentration.
- It can work continuously without any error and boredom.
- It can perform repeated tasks with the same speed and accuracy.

Versatility

- A computer is a very versatile machine.
- A computer is very flexible in performing the jobs to be done.
- This machine can be used to solve the problems related to various fields.
- At one instance, it may be solving a complex scientific problem and the very next moment it may be playing a card game.

Reliability

- A computer is a reliable machine.
- Modern electronic components have long lives.
- Computers are designed to make maintenance easy.

Automation

- Computer is an automatic machine.

- Automation is the ability to perform a given task automatically. Once the computer receives a program i.e., the program is stored in the computer memory, then the program and instruction can control the program execution without human interaction.

Reduction in Paper Work and Cost

- The use of computers for data processing in an organization leads to reduction in paper work and results in speeding up the process.
- As data in electronic files can be retrieved as and when required, the problem of maintenance of large number of paper files gets reduced.
- Though the initial investment for installing a computer is high, it substantially reduces the cost of each of its transaction.

1.1 Von-Neumann Architectures

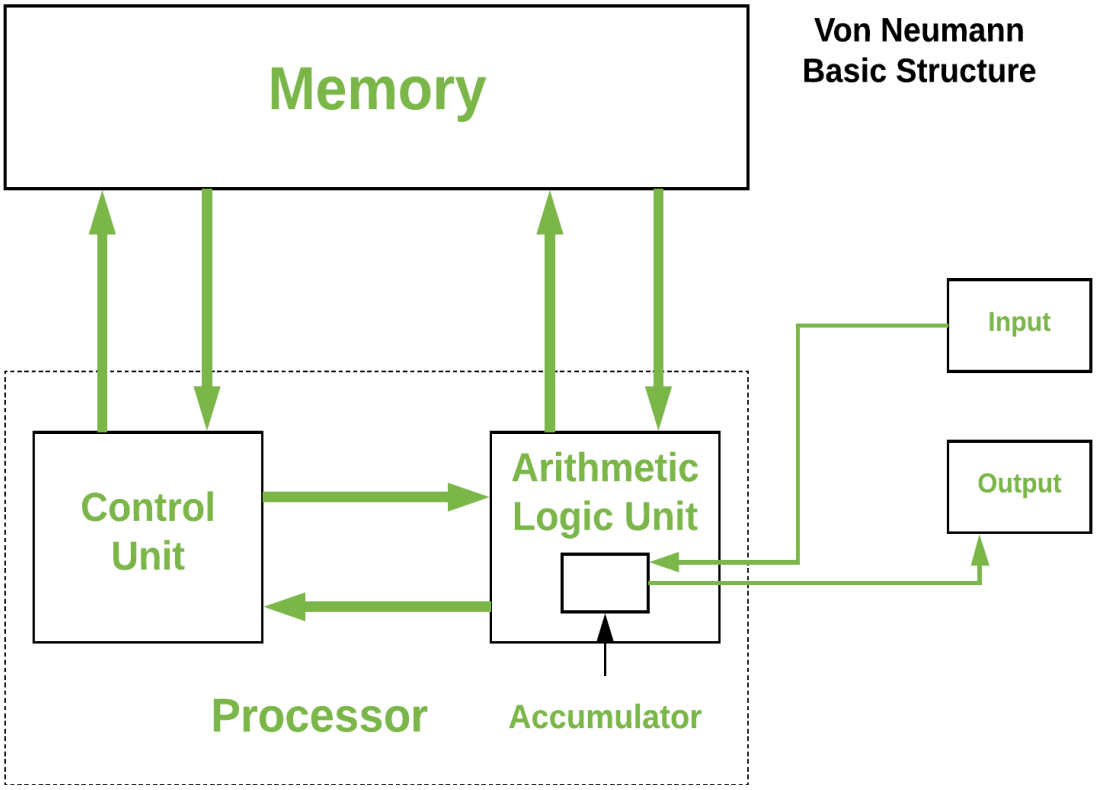
The **von Neumann architecture**—also known as the **von Neumann model** or **Princeton architecture**—is a computer architecture based on a 1945 description by the mathematician and physicist John von Neumann and others in the First Draft of a Report on the EDVAC.

Historically there have been 2 types of Computers:

1. **Fixed Program Computers –** Their function is very specific and they couldn't be programmed, e.g. Calculators.
2. **Stored Program Computers –** These can be programmed to carry out many different tasks, applications are stored on them, hence the name.

The modern computers are based on a stored-program concept introduced by John Von Neumann. In this stored-program concept, programs and data are stored in a separate storage unit called memories and are treated the same. This novel idea meant that a computer built with this architecture would be much easier to reprogram.

The basic structure is like,



It is also known as **IAS** computer and is having three basic units:

1. The Central Processing Unit (CPU)
2. The Main Memory Unit
3. The Input/Output Device

Let's consider them in details.

- **Control Unit –**

  A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions and controlling how data moves around the system.

- **Arithmetic and Logic Unit (ALU) –**

  The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need, e.g. Addition, Subtraction, Comparisons. It performs Logical Operations, Bit Shifting Operations, and Arithmetic Operation.

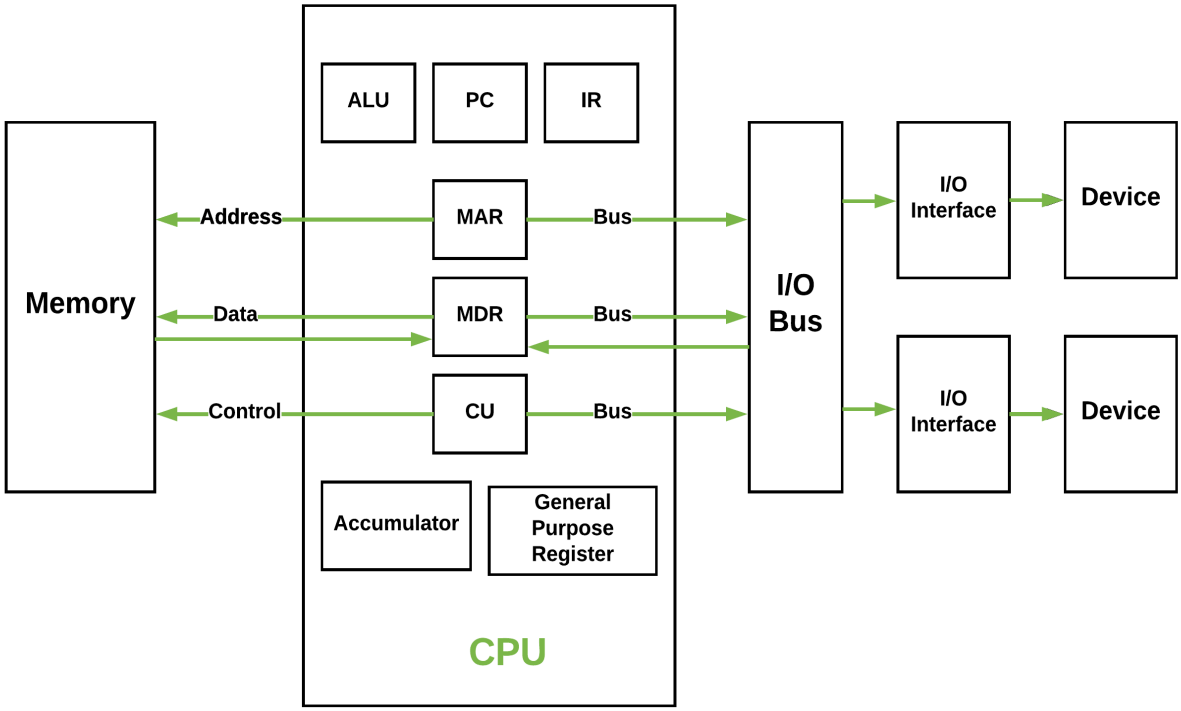


Figure – Basic CPU structure, illustrating ALU

- **Main Memory Unit (Registers) –**
  1. **Accumulator:** Stores the results of calculations made by ALU.
  2. **Program Counter (PC):** Keeps track of the memory location of the next instructions to be dealt with. The PC then passes this next address to Memory Address Register (MAR).
  3. **Memory Address Register (MAR):** It stores the memory locations of instructions that need to be fetched from memory or stored into memory.
  4. **Memory Data Register (MDR):** It stores instructions fetched from memory or any data that is to be transferred to, and stored in, memory.
  5. **Current Instruction Register (CIR):** It stores the most recently fetched instructions while it is waiting to be coded and executed.
  6. **Instruction Buffer Register (IBR):** The instruction that is not to be executed immediately is placed in the instruction buffer register IBR.
- **Input/Output Devices –** Program or data is read into main memory from the input device or secondary storage under the control of CPU input instruction. Output devices are used to output the information from a computer. If some results are evaluated by computer and it is stored in the computer, then with the help of output devices, we can present it to the user.

- **Buses –** Data is transmitted from one part of a computer to another, connecting all major internal components to the CPU and memory, by the means of Buses. Types:
    1. **Data Bus:** It carries data among the memory unit, the I/O devices, and the processor.
    2. **Address Bus:** It carries the address of data (not the actual data) between memory and processor.
    3. **Control Bus:** It carries control commands from the CPU (and status signals from other devices) in order to control and coordinate all the activities within the computer.

Von Neumann bottleneck –

Whatever we do to enhance performance, we cannot get away from the fact that instructions can only be done one at a time and can only be carried out sequentially. Both of these factors hold back the competence of the CPU. This is commonly referred to as the 'Von Neumann bottleneck'. We can provide a Von Neumann processor with more cache, more RAM, or faster components but if original gains are to be made in CPU performance then an influential inspection needs to take place of CPU configuration.
This architecture is very important and is used in our PCs and even in Super Computers.

1.2 Computer Bus

What Is Bus

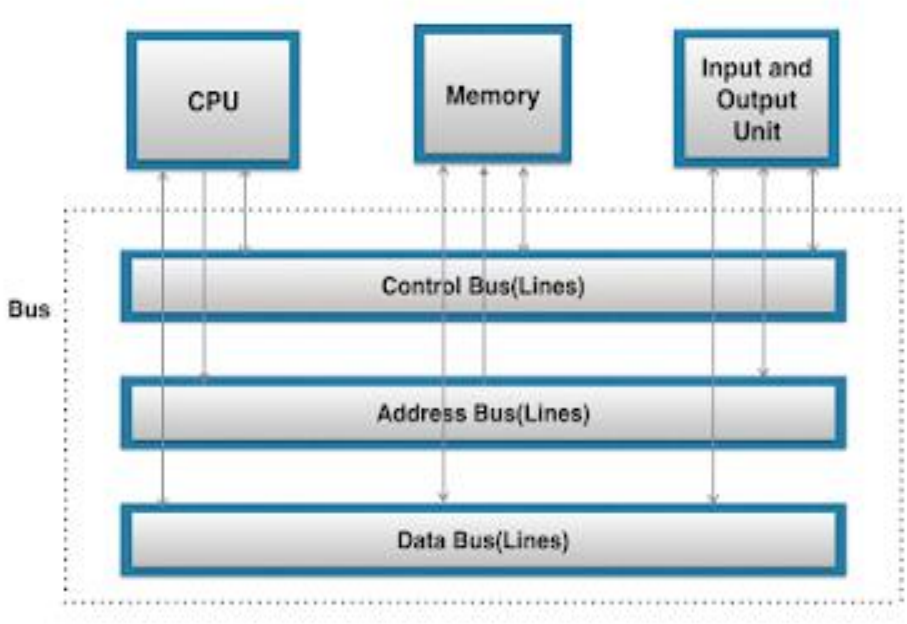
Bus is a subsystem that is used to transfer data and other information between devices.Means various devices in computer like(Memory, CPU, I/O and Other) are communicate with each other through buses.In general, a bus is said to be as the communication pathway connecting two or more devices.

A key characteristics of a bus is that it is a shared transmission medium,as multiple devices are attached to a bus.

Typically, a bus consists of multiple communication Pathways or lines which are either in the form of wires or metal lines etched in a card or board(Printed Circuit Board).Each line is capable of transmitting binary 1 and binary 0.

Computer System Contains a number of different buses that provide pathways between components at various levels of computer system hierarchy. But before discussing them i.e., types of buses , i will first describe here one of the most important aspect of buses which is described below-

Any Bus consists, typically of form about 50 to 100 of separate lines. And on any bus, the lines may generally be classified into three functional groups, as depicted in figure below:



Now,we will discussing about each in brief one by one.

- Data Lines:

Data Lines provide a path for moving data between system modules. It is bidirectional which means data lines are used to transfer data in both directions. As an example, CPU can read data on these lines drum memory as well as send data out of these lines to a memory location or to a port. And in any bus the no. of lines in data lines are either 8,16,32 or more depending on size of bus.These lines , collectively are called as data bus.

- Address Lines:

Address Lines are collectively called as address bus.In any bus, the no. of lines in address are usually 16,20,24, or more depending on type and architecture of bus.On these lines, CPU sends out the address of memory location on I/O Port that is to be written on or read from.In short, it is an internal channel from CPU to Memory across which the address of data(not data) are transmitted.Here the communication is one way that is, the address is send from CPU to Memory and I/O Port but not Memory and I/O port send address to CPU on that line and hence these lines are unidirectional.

- Control Lines:

Control Lines are collectively called as Control Bus.Control Lines are gateway used to transmit and receives control signals between the microprocessor and various devices attached to it.In other words, Control Lines are used by CPUs for Communicating with other devices within the computer.As an example-CPU sends signals on the Control bus to enable the outputs of address memory devices and port devices.Typical Control Lines signals are:

-Memory Read

-Memory Write

-I/O Read

-I/O Write

-Bus Request

-Bus Grant, etc.

Operation of Bus

The Operation of Bus is as follows:

- If One module wishes to send data to another, it must do two things:

  1.Obtain the use of module Bus

  2.Transfer for data to the Bus

- If one module wishes to request data from another module, it must:

  1.Obtain the use of Bus.

  2.Transfer a request to other module over the appropriate control and address lines. It must then wait for that second module to send the data.

Types Of Bus

There are variety of Buses, but here i will describe only those that are widely used.

- System Bus:

A Bus that connects major computer components (Processor, Memory, I/O) is called a System Bus.It is a single computer bus among all Buses that connects all these components of a computer system.And it is the only Bus, in which data lines, address, control lines all are present.It is also Known as "front side " Bus.It is faster than peripheral Bus(PCI, ISA, etc) but slower than backside Bus.

- Peripheral Bus(I/O Bus /External Bus):

Peripheral Bus also known as "I/O Bus".It is data pathway that connects peripheral devices to the CPU.In other words , in computing, a peripheral bus is a computer bus designed to support computer peripheral like printers,hard drives.The PCI and USB buses are commonly used

Peripheral Buses, and are today used in commonly many PCs.Now we will discuss both of them in brief:

PCI(Peripheral Component Interconnect):

PCI Bus connects the CPU and expansion boards such as modem cards ,network cards and sound cards.These expansion boards are normally plugged into expansion slots on the motherboard.That's why PCI bus is also known as expansion bus or external Bus.

USB(Universal Serial Bus):

Universal Serial Bus is used to attach USB devices like Pen Drive, etc to CPU.

- Local Bus:

Local Bus are the traditional I/O(Peripheral) buses such as ISA,MCA, or EISA Buses.Now we will discuss about each in brief one by one.

ISA(Industry Standard Architecture Bus): The ISA Bus permit bus mastering i.e., it enabled peripheral connected directly to the bus to communicate directly with other peripherals without going through the processor.One of the consequences of bus mastering is Direct Memory Access.Up to end of 1990s almost all PCs computers were equipped with ISA Bus, but it was progressively replaced by the PCI Bus,which offer a better performance.

MCA(Micro Channel Architecture): It is an improved proprietary bus designed by IBM in 1987 to be used in their PS/2 lines of computers.

EISA(Extended Industry Standard Architecture): The EISA Bus use connectors that were same size as the ISA connectors but with 4 rows of contacts instead of 2 for 32 bit addressing.

- High Speed Bus:

High Speed Bus are specifically designed to support high capacity I/O devices.High Speed Bus brings high demand devices into closer integration with the processor.This Bus supports connection to high speed LANs, such as Fast Ethernet at 100 Mbps, video and graphic workstation, firewire etc.

What is Logic Gate

A logic gate is a basic building block of a digital circuit that has two inputs and one output.The relationship between the i/p and the o/p is based on a certain logic. These gates are implemented

using electronic switches like transistors, diodes. But, in practice basic logic gates are built using CMOS technology, FETS and MOSFET(Metal Oxide Semiconductor FET)s. Logic gates are used in microprocessors, microcontrollers, embedded system applications and in electronic and electrical project circuits.

Basic Gates

AND gate

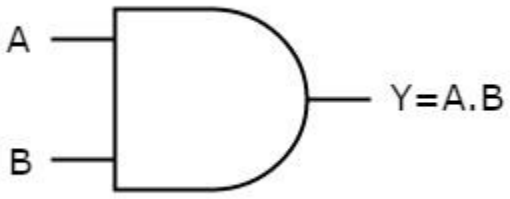
An AND gate is a digital circuit that has two or more inputs and produces an output, which is the **logical AND** of all those inputs. It is optional to represent the **Logical AND** with the symbol '.'.

The following table shows the **truth table** of 2-input AND gate.

A	**B**	**Y = A.B**
0	0	0
0	1	0
1	0	0
1	1	1

Here A, B are the inputs and Y is the output of two input AND gate. If both inputs are '1', then only the output, Y is '1'. For remaining combinations of inputs, the output, Y is '0'.

The following figure shows the **symbol** of an AND gate, which is having two inputs A, B and one output, Y.



This AND gate produces an output (Y), which is the **logical AND** of two inputs A, B. Similarly, if there are 'n' inputs, then the AND gate produces an output, which is the logical AND of all those inputs. That means, the output of AND gate will be '1', when all the inputs are '1'.

OR gate

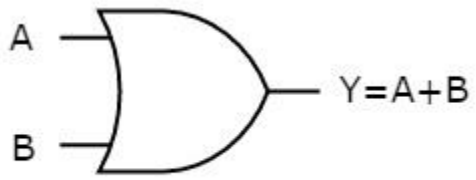
An OR gate is a digital circuit that has two or more inputs and produces an output, which is the logical OR of all those inputs. This **logical OR** is represented with the symbol '+'.

The following table shows the **truth table** of 2-input OR gate.

A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Here A, B are the inputs and Y is the output of two input OR gate. If both inputs are '0', then only the output, Y is '0'. For remaining combinations of inputs, the output, Y is '1'.

The following figure shows the **symbol** of an OR gate, which is having two inputs A, B and one output, Y.



This OR gate produces an output (Y), which is the **logical OR** of two inputs A, B. Similarly, if there are 'n' inputs, then the OR gate produces an output, which is the logical OR of all those inputs. That means, the output of an OR gate will be '1', when at least one of those inputs is '1'.

NOT gate

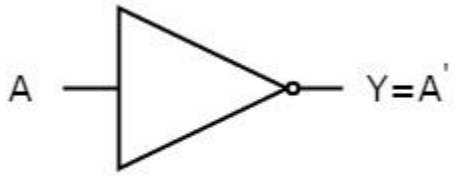
A NOT gate is a digital circuit that has single input and single output. The output of NOT gate is the **logical inversion** of input. Hence, the NOT gate is also called as inverter.

The following table shows the **truth table** of NOT gate.

A	Y = A'
0	1
1	0

Here A and Y are the input and output of NOT gate respectively. If the input, A is ‘0’, then the output, Y is ‘1’. Similarly, if the input, A is ‘1’, then the output, Y is ‘0’.

The following figure shows the **symbol** of NOT gate, which is having one input, A and one output, Y.



This NOT gate produces an output (Y), which is the **complement** of input, A.

Universal gates

NAND & NOR gates are called as **universal gates**. Because we can implement any Boolean function, which is in sum of products form by using NAND gates alone. Similarly, we can implement any Boolean function, which is in product of sums form by using NOR gates alone.

NAND gate

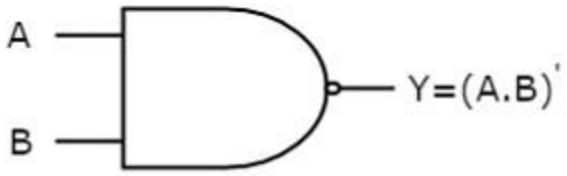
NAND gate is a digital circuit that has two or more inputs and produces an output, which is the **inversion of logical AND** of all those inputs.

The following table shows the **truth table** of 2-input NAND gate.

A	B	Y = (A.B)’
0	0	1
0	1	1
1	0	1
1	1	0

Here A, B are the inputs and Y is the output of two input NAND gate. When both inputs are ‘1’, the output, Y is ‘0’. If at least one of the input is zero, then the output, Y is ‘1’. This is just opposite to that of two input AND gate operation.

The following image shows the **symbol** of NAND gate, which is having two inputs A, B and one output, Y.



NAND gate operation is same as that of AND gate followed by an inverter. That's why the NAND gate symbol is represented like that.

NOR gate

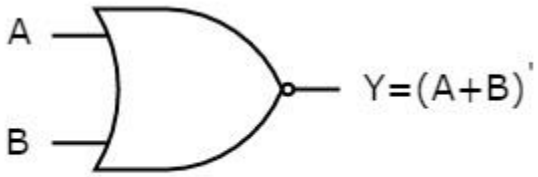
NOR gate is a digital circuit that has two or more inputs and produces an output, which is the **inversion of logical OR** of all those inputs.

The following table shows the **truth table** of 2-input NOR gate

A	B	Y = (A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

Here A, B are the inputs and Y is the output. If both inputs are '0', then the output, Y is '1'. If at least one of the input is '1', then the output, Y is '0'. This is just opposite to that of two input OR gate operation.

The following figure shows the **symbol** of NOR gate, which is having two inputs A, B and one output, Y.



NOR gate operation is same as that of OR gate followed by an inverter. That's why the NOR gate symbol is represented like that.

Special Gates

Ex-OR & Ex-NOR gates are called as special gates. Because, these two gates are special cases of OR & NOR gates.

Ex-OR gate

The full form of Ex-OR gate is **Exclusive-OR** gate. Its function is same as that of OR gate except for some cases, when the inputs having even number of ones.

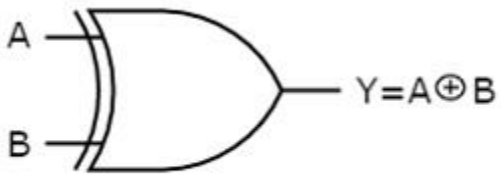
The following table shows the **truth table** of 2-input Ex-OR gate.

A	**B**	**Y = A⊕B**
0	0	0
0	1	1
1	0	1
1	1	0

Here A, B are the inputs and Y is the output of two input Ex-OR gate. The truth table of Ex-OR gate is same as that of OR gate for first three rows. The only modification is in the fourth row. That means, the output (Y) is zero instead of one, when both the inputs are one, since the inputs having even number of ones.

Therefore, the output of Ex-OR gate is '1', when only one of the two inputs is '1'. And it is zero, when both inputs are same.

Below figure shows the **symbol** of Ex-OR gate, which is having two inputs A, B and one output, Y.



Ex-OR gate operation is similar to that of OR gate, except for few combination(s) of inputs. That's why the Ex-OR gate symbol is represented like that. The output of Ex-OR gate is '1', when odd number of ones present at the inputs. Hence, the output of Ex-OR gate is also called as an **odd function**.

Ex-NOR gate

The full form of Ex-NOR gate is **Exclusive-NOR** gate. Its function is same as that of NOR gate except for some cases, when the inputs having even number of ones.

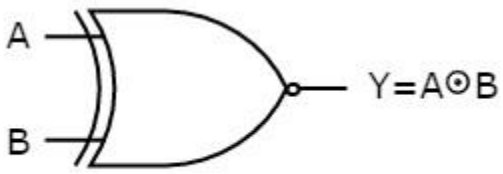
The following table shows the **truth table** of 2-input Ex-NOR gate.

A	B	Y = A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

Here A, B are the inputs and Y is the output. The truth table of Ex-NOR gate is same as that of NOR gate for first three rows. The only modification is in the fourth row. That means, the output is one instead of zero, when both the inputs are one.

Therefore, the output of Ex-NOR gate is '1', when both inputs are same. And it is zero, when both the inputs are different.

The following figure shows the **symbol** of Ex-NOR gate, which is having two inputs A, B and one output, Y.



Ex-NOR gate operation is similar to that of NOR gate, except for few combination(s) of inputs. That's why the Ex-NOR gate symbol is represented like that. The output of Ex-NOR gate is '1', when even number of ones present at the inputs. Hence, the output of Ex-NOR gate is also called as an **even function**.

From the above truth tables of Ex-OR & Ex-NOR logic gates, we can easily notice that the Ex-NOR operation is just the logical inversion of Ex-OR operation.

Introduction to Flip Flops

Flip Flops

Do you know!! computers and calculators use Flip-flop for their memory. A combination of number of flip flops will produce some amount of memory.

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.

These are also called as sequential logic circuits. Also know these before learning about fliplfops.

- Sequential Logic circuits
- Latches

Flip – flops have two stable states and hence they are bistable multivibrators. The two stable states are High (logic 1) and Low (logic 0).

The term flip – flop is used as they can switch between the states under the influence of a control signal (clock or enable) i.e. they can 'flip' to one state and 'flop' back to other state.

- Flip – flops are a binary storage device because they can store binary data (0 or 1).
- Flip – flops are edge sensitive or edge triggered devices i.e. they are sensitive to the transition rather than the duration or width of the clock signal.
- They are also known as signal change sensitive devices which mean that the change in the level of clock signal will bring change in output of the flip flop.
- A Flip – flop works depending on clock pulses.
- Flip flops are also used to control the digital circuit's functionality. They can change the operation of a digital circuit depending on the state.

Some of the most common flip – flops are SR Flip – flop (Set – Reset), D Flip – flop (Data or Delay), JK Flip – flop and T Flip – flop.

Latches vs Flip-Flops

Latches and flip – flops are both 1 – bit binary data storage devices. The main difference between a latch and a flip – flop is the triggering mechanism. Latches are transparent when enabled ,whereas flip – flops are dependent on the transition of the clock signal i.e. either positive edge or negative edge.

The modern usage of the term flip – flop is reserved to clocked devices and term latch is to describe much simpler devices. Some of the other differences between latches and flip – flops are listed in below table.

LATCH	FLIP – FLOP
Latches do not require clock signal.	Flip – flops have clock signals
A latch is an asynchronous device.	A flip – flop is a synchronous device.
Latches are transparent devices i.e. when they are enabled, the output changes immediately if the input changes.	A transition from low to high or high to low of the clock signal will cause the flip – flop to either change its output or retain it depending on the input signal.
A latch is a Level Sensitive device (Level Triggering is involved).	A flip – flop is an edge sensitive device (Edge Triggering is involved).
Latches are simpler to design as there is no clock signal (no careful routing of clock signal is required).	When compare to latches, flip – flops are more complex to design as they have clock signal and it has to be carefully routed. This is because all the flip – flops in a design should have a clock signal and the delay in the clock reaching each flip – flop must be minimum or negligible.
The operation of a latch is faster as they do not have to wait for any clock signal.	Flip - flops are comparatively slower than latches due to clock signal.
The power requirement of a latch is less.	Power requirement of a flip – flop is more.
A latch works based on the enable signal.	A flip – flop works based on the clock signal.

Types of flip flops

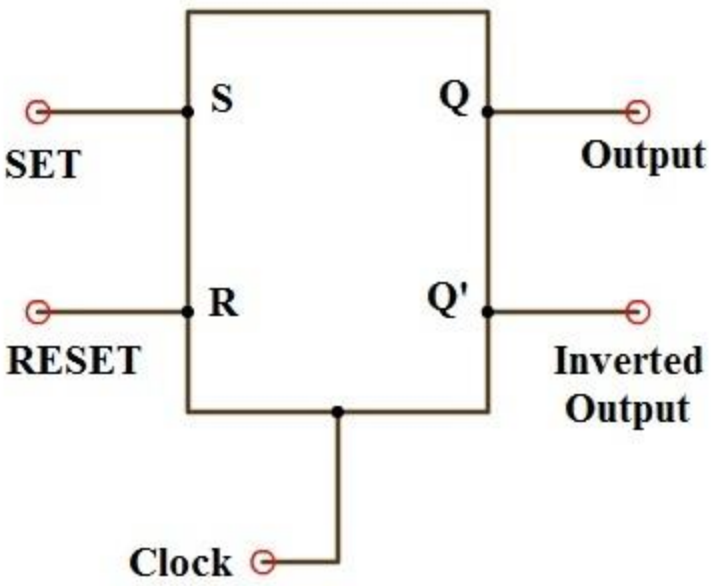
Based on their operations, flip flops are basically 4 types. They are

1. S-R flip flop
2. D flip flop
3. J-K flip flop
4. T flip flop

S-R Flip Flop

The S-R flip-flop is basic flip-flop among all the flip-flops. All the other flip flops are developed after SR-flip-flop.

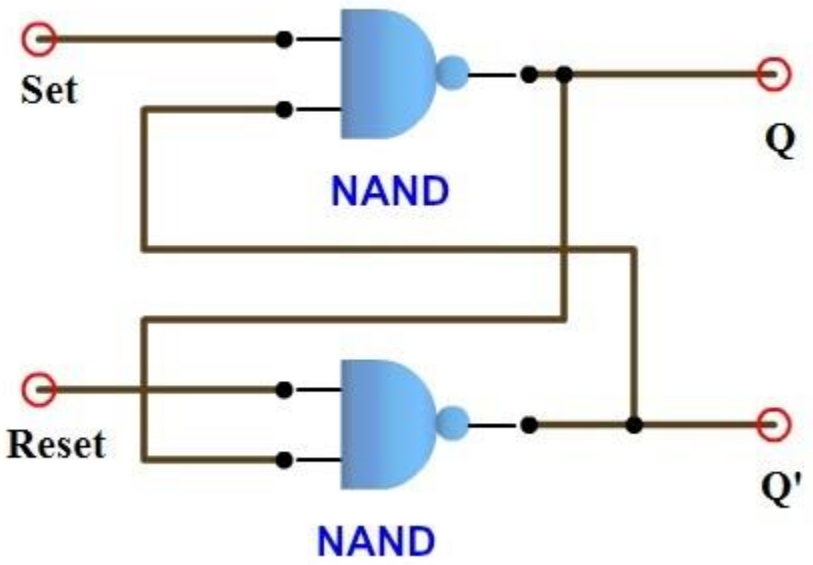
SR flip flop is represented as shown below.



S-R stands for SET and RESET. This can also be called RS flip-flop. Difference is RS is inverted SR flip-flop.

Any flip flop can be build using logic gates. NAND and NOR gates were used as they are universal gates.

Here is the SR flip-flop using NAND gates.



Truth Table of SR Flip Flop

Sno	S	R	Q	Q'	State
1	1	0	1	0	Q is set to 1
2	1	1	1	0	No change
3	0	1	0	1	Q' is set to 1
4	1	1	0	1	No change
5	0	0	1	1	Invalid

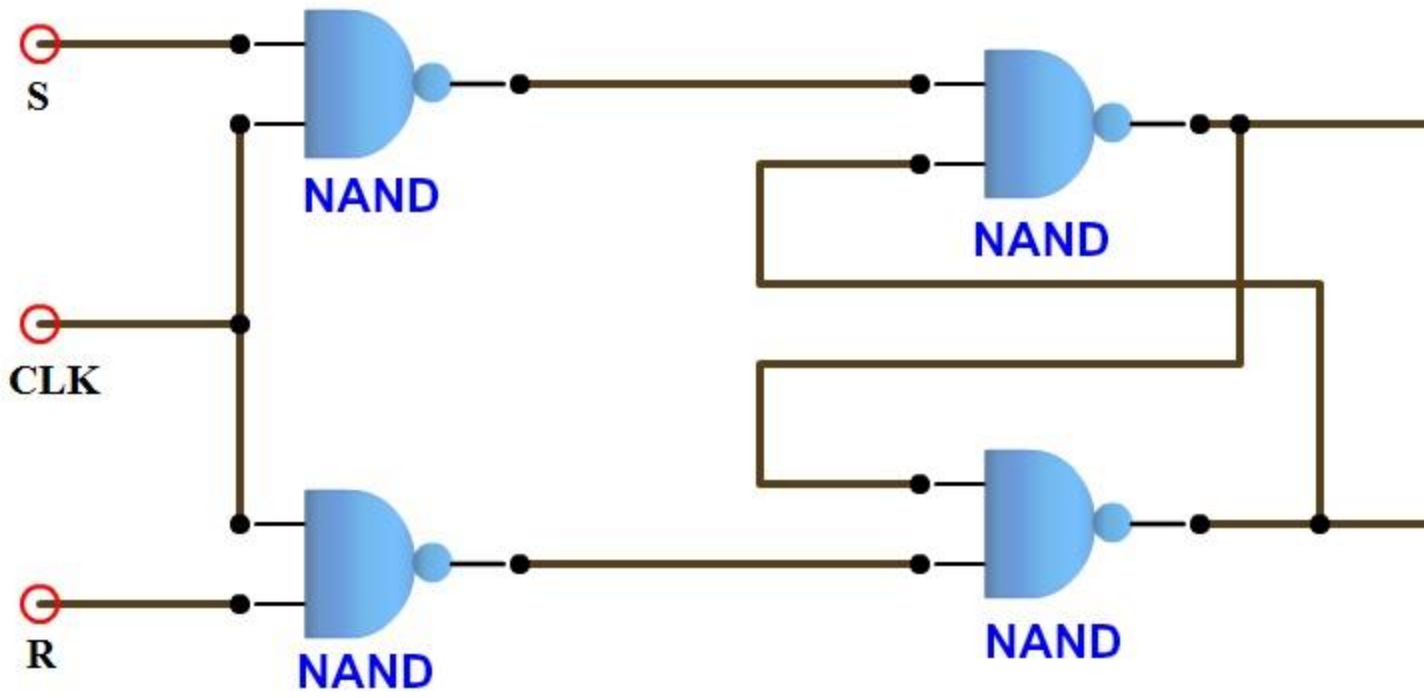
Working

From the above truth table it is clear that SR flip flop will be set or reset for four conditions.

1. For last condition it will be in invalid state.
2. SR Flip-flop will be set when S=1 and R=0, if S=1 and R=1 then previous state is remembered by the flip flop.
3. Flip-flop will be reset when S=0 and R=1, if S=1 and R=1, then it will remember the previous state.
4. But when both the inputs are zeros, SR Flip flop will be in an uncertain state where both Q and Q' will be same. This is not same allowed..

This is indeterminate state is avoided by adding gates extra gates to the existing flip flop. This is called clocked or gated SR Flip flop. This produces the output only for the High clock pulse.

The circuit of a clocked SR flip – flop using NAND gates is shown below.

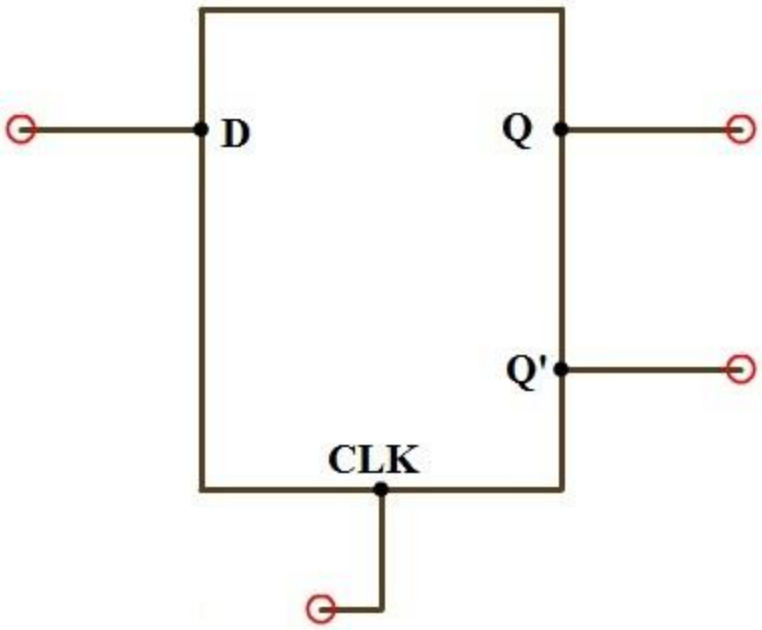


Know in detail about SR Flip FlopD flip flop

D flip flop

In the SR flip flop an uncertain state occurred. This can be avoided by using D flip flop. Here D stands for "Data".
It is constructed from SR flip flop. The two inputs (S &R) of the clocked SR flip flop are connected to an inverter.

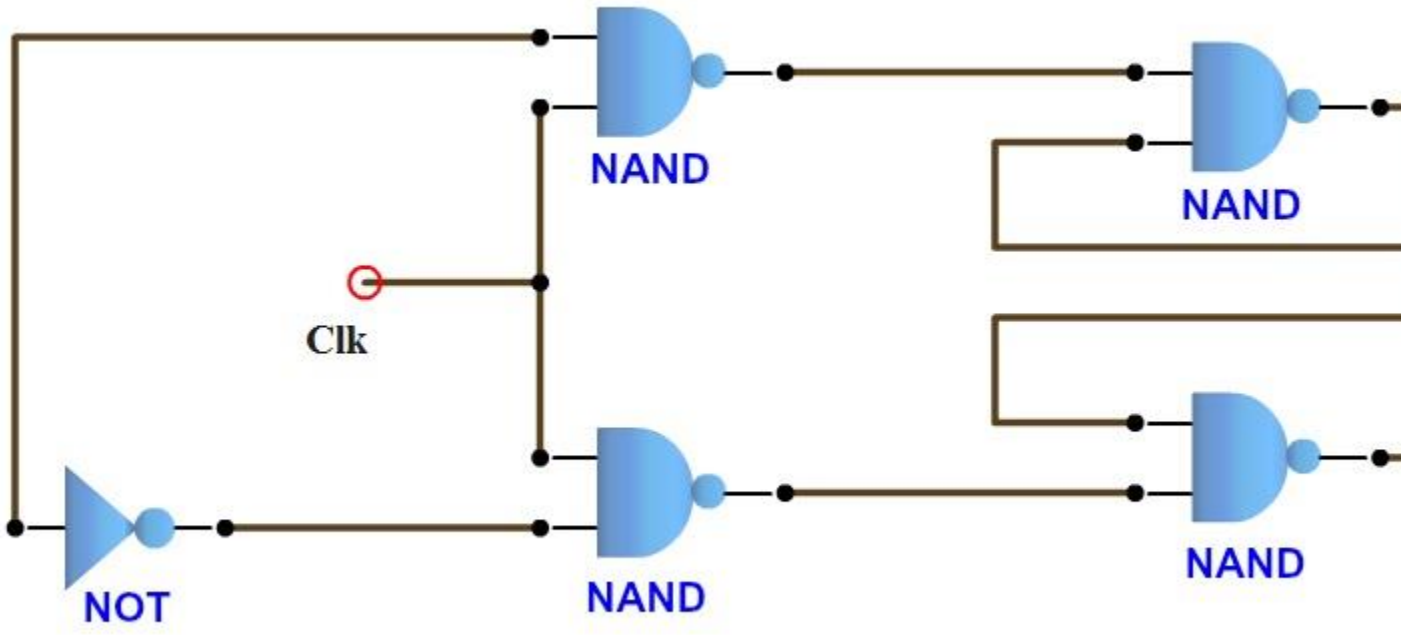
It is one of the most widely used flip – flops. It has a clock signal (Clk) as one input and Data (D) as other. There are two outputs and these outputs are complement to each other. The symbol of D flip – flop is shown below.



Truth table

Clk	D	Q	Q’	State
0	0	Q	Q’	No change in state
1	0	0	1	Resets Q to 0
1	1	1	1	Sets Q to 1

D flip – flop using NAND gates is shown below.



Working

- D flip flop will work depending on the clock signal.
- When the clock is low there will be no change in the output of the flip flop i.e. it remembers the previous state.
- When the clock signal is high and if it receives any data on its data pin, it Changes the state of output.
- When data is high Q reset to 0,while Q is set to 0 if data is low.

A master slave D flip flop can be constructed using D-flip flop.

Know in detail about D-flip flop.

J-K Flip Flop

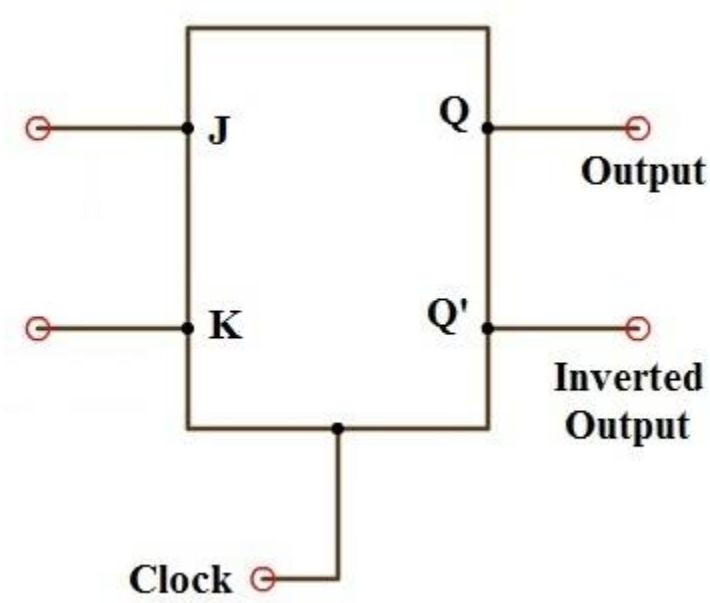
JK flip – flop is named after Jack Kilby, an electrical engineer who invented IC.

A JK flip – flop is a modification of SR flip – flop. 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 condition J = K = 1 which is not allowed in SR flip – flop (S = R = 1) is interpreted as a toggle command.

The JK flip flop has

- Two data inputs J and K.
- One clock signal input (CLK).
- Two outputs Q and Q'.

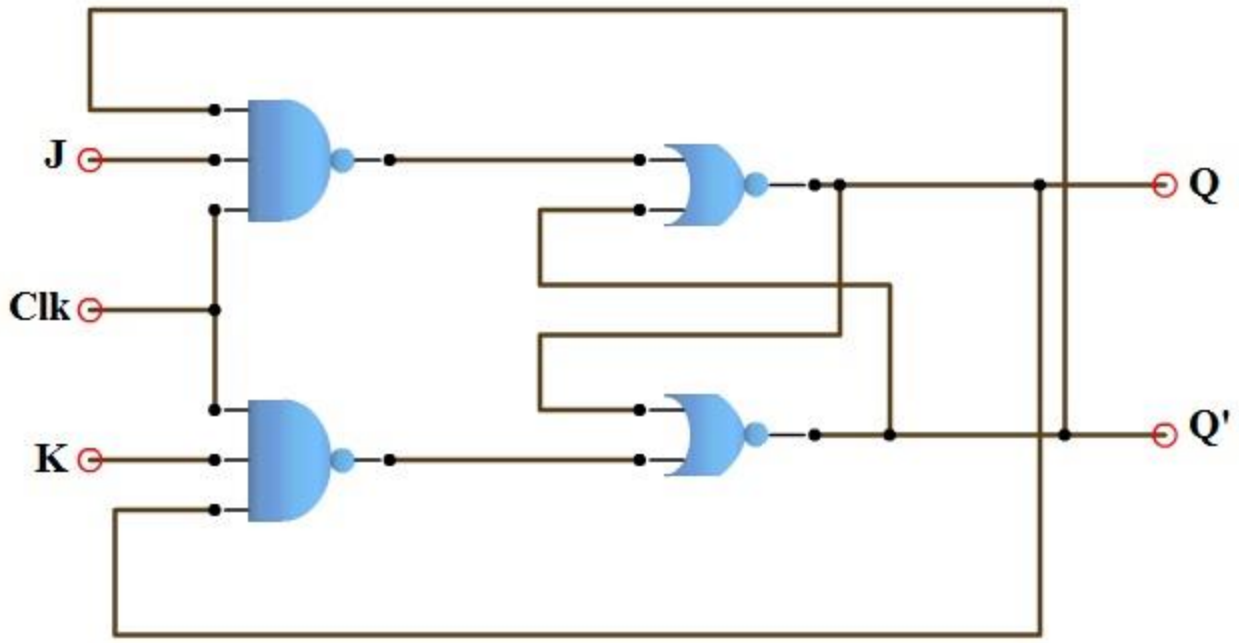
The symbol of a JK flip – flop is shown below.



Truth Table

Clk	J	K	Q	Q'	State
1	0	0	Q	Q'	No change in state
1	0	1	0	1	Resets Q to 0
1	1	0	1	0	Sets Q to 1
1	1	1	-	-	Toggles

The circuit of a JK flip – flop using gates is shown below. It is similar to a modified NAND SR flip – flop.



Working

- When J is low and K is low, then Q returns its previous state value i.e. it holds the current state.
- When J is low and K is high, then flip – flop will be in reset state i.e. Q = 0, Q' =1.
- When J is high and K is low then flip – flop will be in set state i.e. Q = 1, Q' =0.
- When J is high and K is high then flip – flop will be in Toggle state or flip state. This means that the output will complement to the previous state value.

To Know in detail about JK Flip Flop

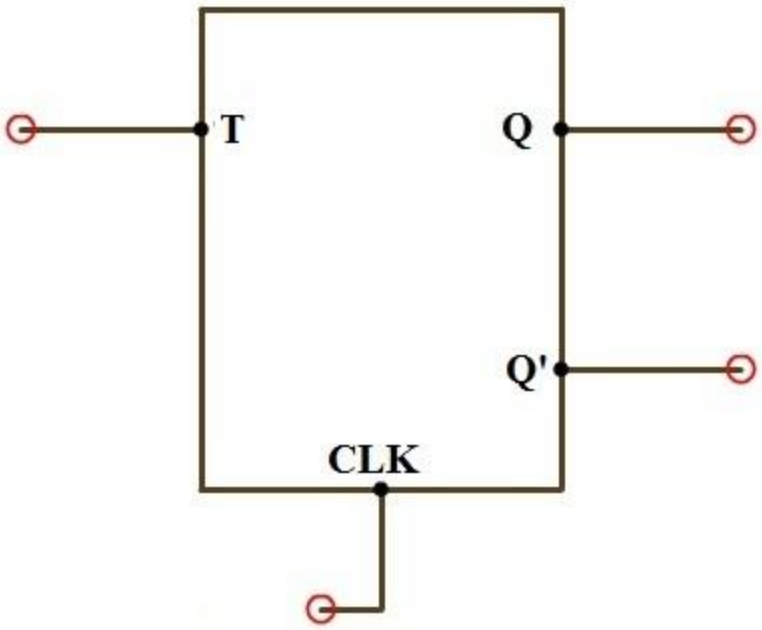
T Flip Flop

T flip flop is also known as "Toggle Flip – flop". Toggle is to change the output to complement of the previous state in the presence of clock input signal.

The T flip flop has

- T input.
- One clock signal input (CLK).
- Two outputs Q and Q'.

The symbol of a T flip – flop is shown below.



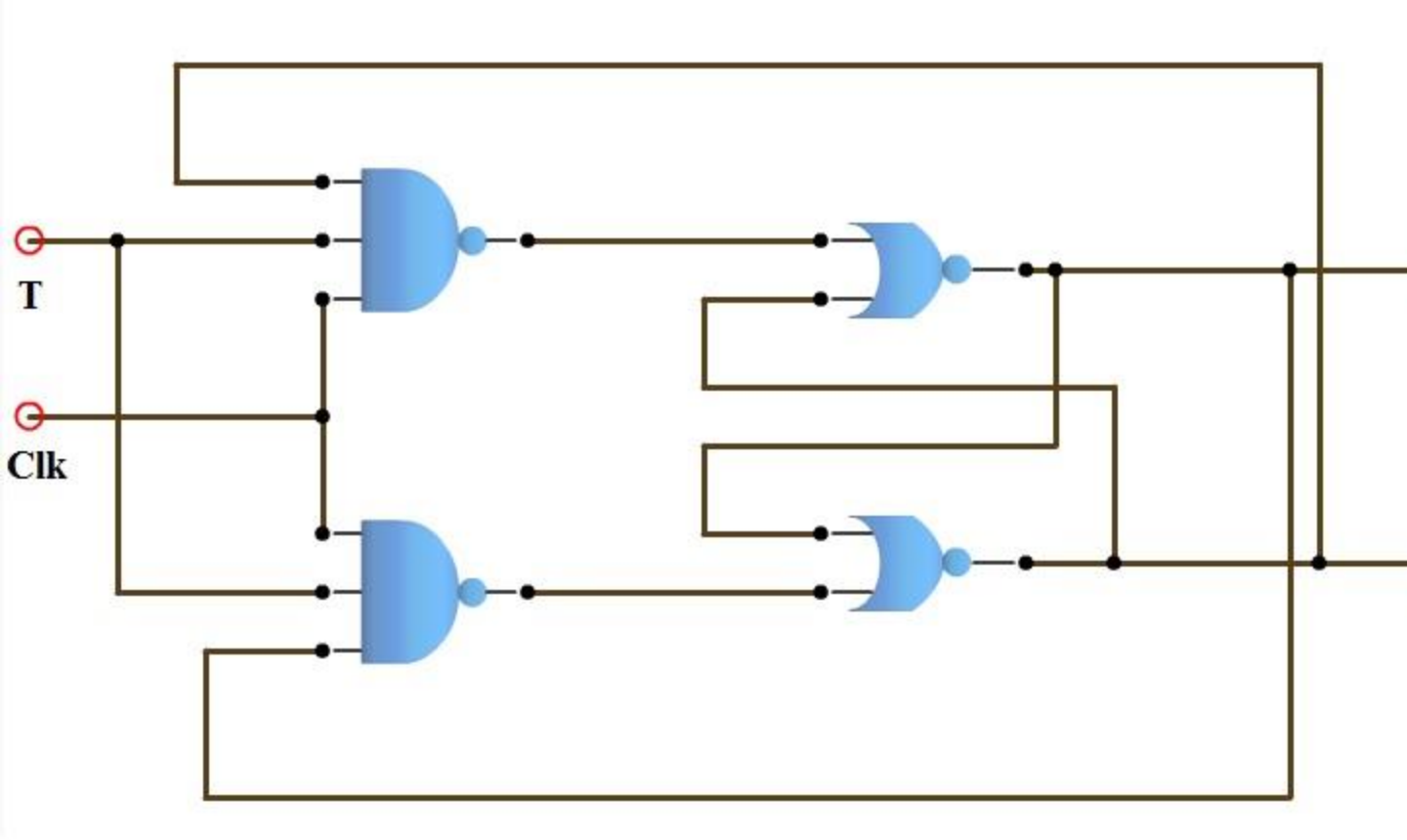
We can construct a T flip – flop by using any other flip – flops.

- SR flip – flop: By connecting the feedback of outputs of SR flip – flop to the inputs (S & R).
- D flip – flop: Connecting the Q' to its Data input of D flip – flop as feedback path.
- J K flip – flop: By combing the J & K inputs of JK flip – flop, to make as single input, we can design the T flip – flop.

Truth Table

T	Q	Q'
0	0	0
1	0	1
0	1	0
1	1	0

The circuit of a T flip – flop made from NAND JK flip – flop is shown below.



Working

The operation of the T flip flop is explained below.

When the T input is low, then the next sate of the T flip – flop is same as the present state i.e. it holds the current state.

- T = 0 and present state = 0 then the next state = 0.
- T = 0 and present state = 1 then the next state = 1.

When the T input is high, then the next sate of the T flip – flop is toggled i.e. it is same as the complement of present state on clock transition.

- T = 1 and present state = 0 then the next state = 1.
- T = 1 and present state = 1 then the next state = 0.

Know in detail about T-Flip flop

WHERE WE USE FLIP FLOPS??

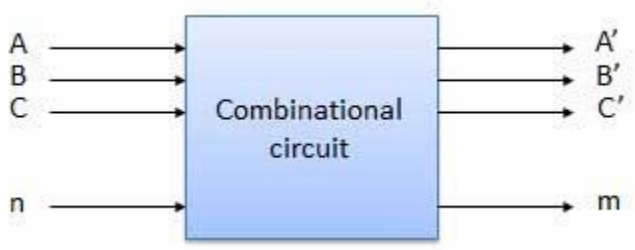
Flip flops are widely used in

- Registers: As the flip flops have two stable states, we use them in memory elements like registers, for data storage. Generally we use registers in electronic devices like computers.
- Counters: The groups of interconnected flip flops are uses as counters, to count the increment or decrement of an event occurrence.
- Frequency division: Flip flops are used as frequency division circuits, which divide the input frequency to exactly to its half. Frequency division circuits are used to regularize the frequency of electronic circuits.
- Data transfer: We use shift registers (A special-type of registers) to transfer the data from one flip flop to another, which are connected in a specific order.

Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer. Some of the characteristics of combinational circuits are following −

- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
- The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
- A combinational circuit can have an n number of inputs and m number of outputs.

Block diagram



We're going to elaborate few important combinational circuits as follows.

Half Adder

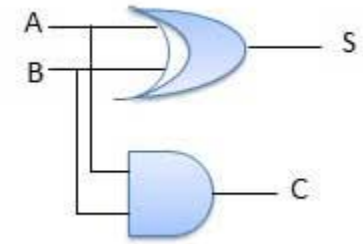
Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**.

Block diagram

Truth Table

Inputs		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

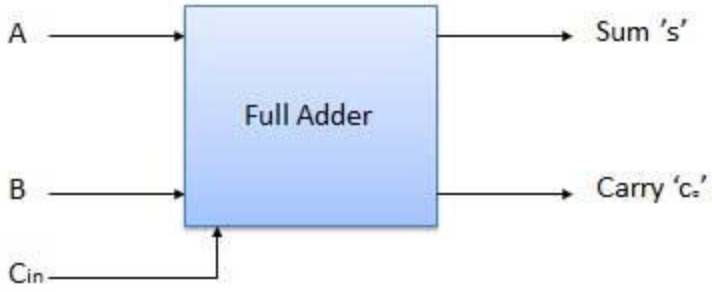
Circuit Diagram



Full Adder

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

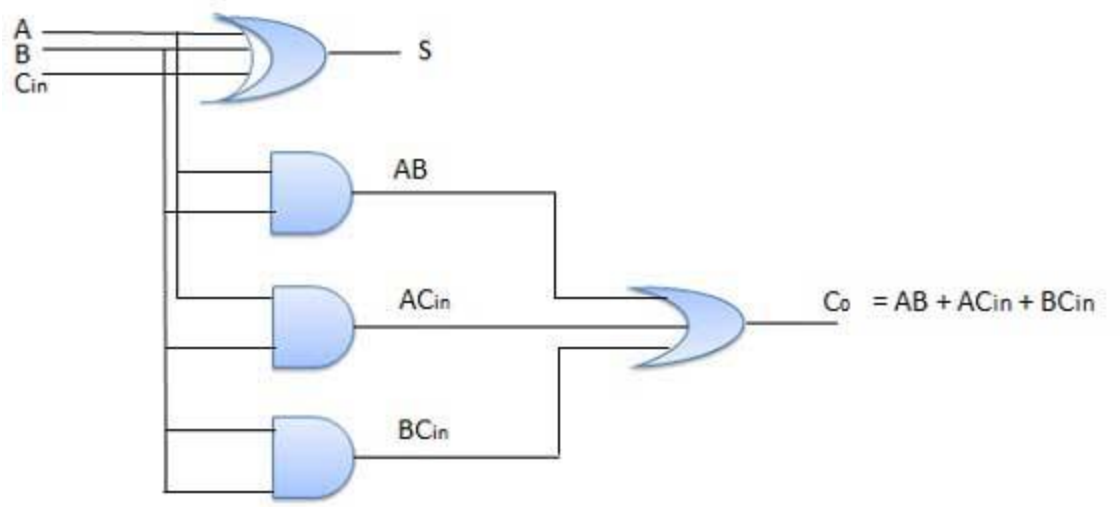
Block diagram



Truth Table

Inputs			Output	
A	B	Cin	S	Co
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

Circuit Diagram



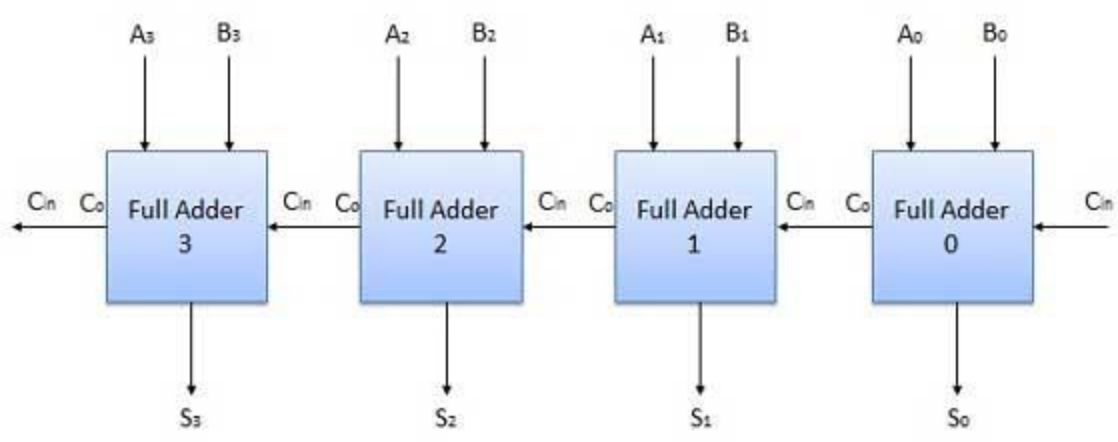
N-Bit Parallel Adder

The Full Adder is capable of adding only two single digit binary number along with a carry input. But in practical we need to add binary numbers which are much longer than just one bit. To add two n-bit binary numbers we need to use the n-bit parallel adder. It uses a number of full adders in cascade. The carry output of the previous full adder is connected to carry input of the next full adder.

4 Bit Parallel Adder

In the block diagram, A_0 and B_0 represent the LSB of the four bit words A and B. Hence Full Adder-0 is the lowest stage. Hence its C_{in} has been permanently made 0. The rest of the connections are exactly same as those of n-bit parallel adder is shown in fig. The four bit parallel adder is a very common logic circuit.

Block diagram



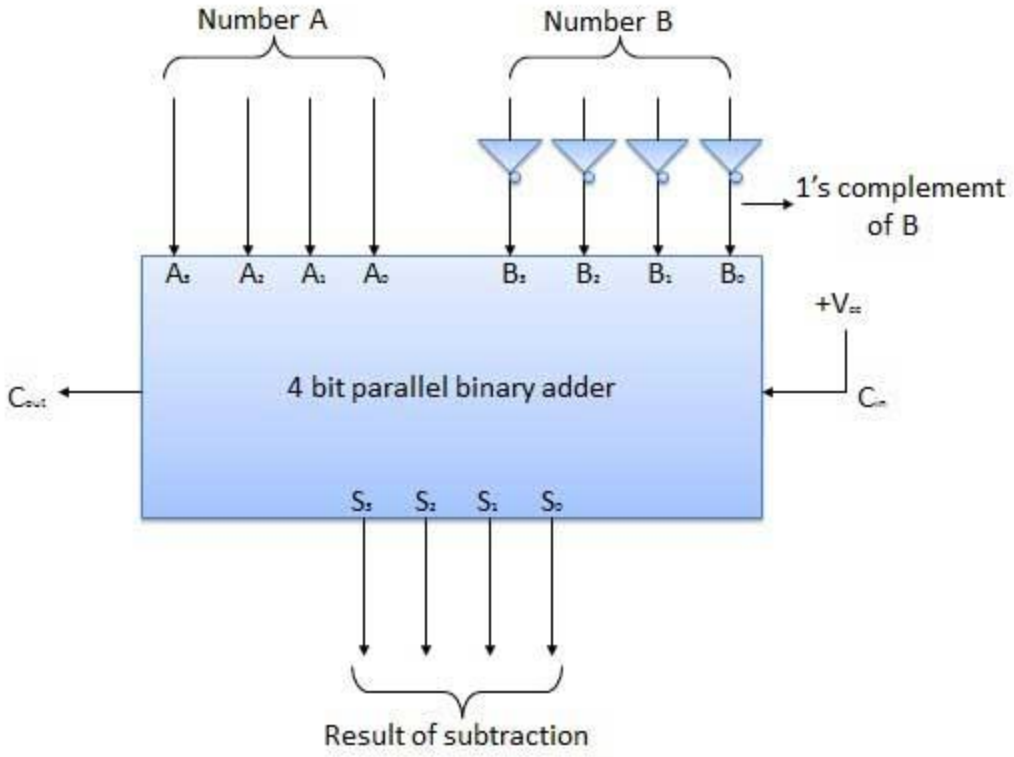
N-Bit Parallel Subtractor

The subtraction can be carried out by taking the 1's or 2's complement of the number to be subtracted. For example we can perform the subtraction (A-B) by adding either 1's or 2's complement of B to A. That means we can use a binary adder to perform the binary subtraction.

4 Bit Parallel Subtractor

The number to be subtracted (B) is first passed through inverters to obtain its 1's complement. The 4-bit adder then adds A and 2's complement of B to produce the subtraction. S_3 S_2 S_1 S_0 represents the result of binary subtraction (A-B) and carry output C_{out} represents the polarity of the result. If A > B then Cout = 0 and the result of binary form (A-B) then C_{out} = 1 and the result is in the 2's complement form.

Block diagram



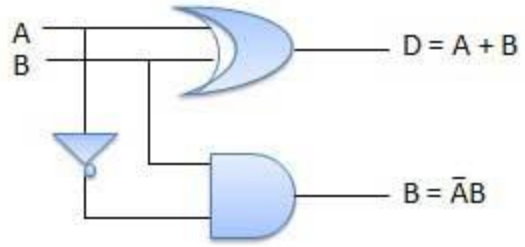
Half Subtractors

Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit.

Truth Table

Inputs		Output	
A	B	(A – B)	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Circuit Diagram



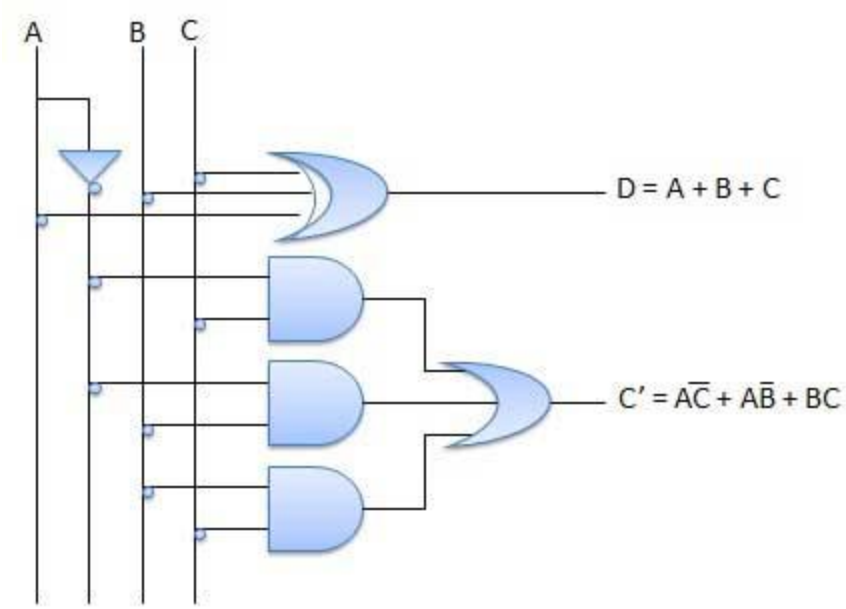
Full Subtractors

The disadvantage of a half subtractor is overcome by full subtractor. The full subtractor is a combinational circuit with three inputs A,B,C and two output D and C'. A is the 'minuend', B is 'subtrahend', C is the 'borrow' produced by the previous stage, D is the difference output and C' is the borrow output.

Truth Table

Inputs			Output	
A	B	C	(A-B-C)	C'
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

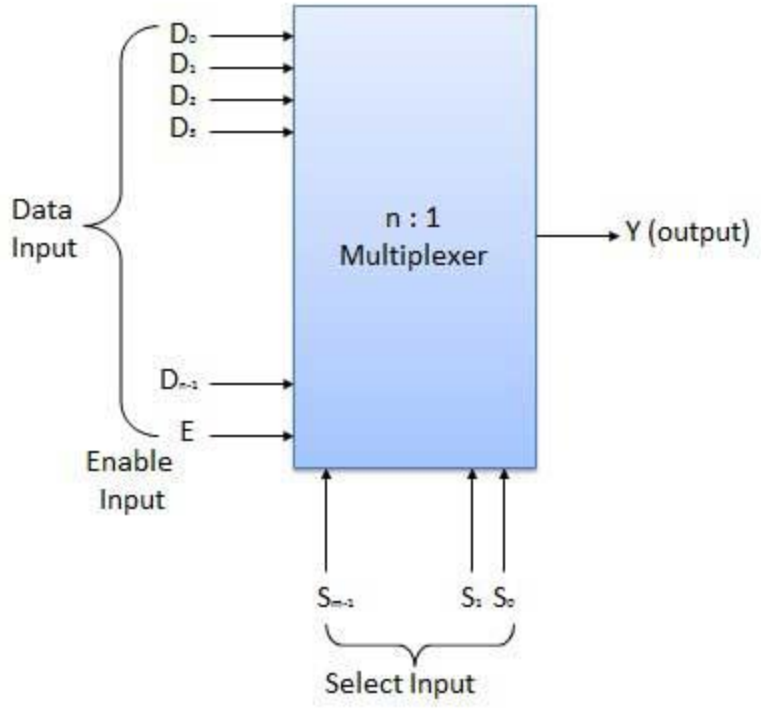
Circuit Diagram



Multiplexers

Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with 2m = n. It is a digital circuit which selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the selected inputs. Depending on the digital code applied at the selected inputs, one out of n data sources is selected and transmitted to the single output Y. E is called the strobe or enable input which is useful for the cascading. It is generally an active low terminal that means it will perform the required operation when it is low.

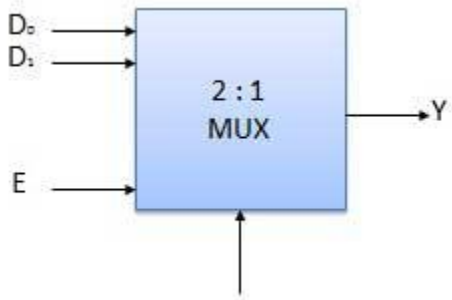
Block diagram



Multiplexers come in multiple variations

- 2 : 1 multiplexer
- 4 : 1 multiplexer
- 16 : 1 multiplexer
- 32 : 1 multiplexer

Block Diagram



Truth Table

Enable	Select	Output
E	S	Y
0	x	0
1	0	D_0
1	1	D_1

x = Don't care

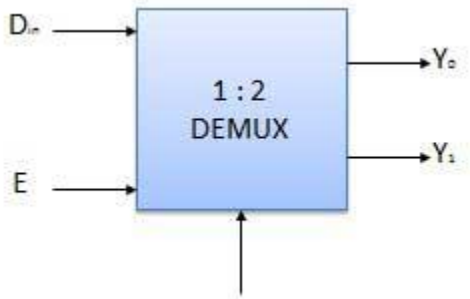
Demultiplexers

A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs. It has only one input, n outputs, m select input. At a time only one output line is selected by the select lines and the input is transmitted to the selected output line. A de-multiplexer is equivalent to a single pole multiple way switch as shown in fig.

Demultiplexers comes in multiple variations.

- 1 : 2 demultiplexer
- 1 : 4 demultiplexer
- 1 : 16 demultiplexer
- 1 : 32 demultiplexer

Block diagram



Truth Table

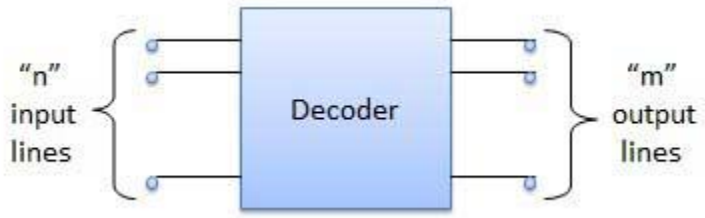
Enable	Select	Output	
E	S	Y0	Y1
0	x	0	0
1	0	0	D_{in}
1	1	D_{in}	0

x = Don't care

Decoder

A decoder is a combinational circuit. It has n input and to a maximum m = 2n outputs. Decoder is identical to a demultiplexer without any data input. It performs operations which are exactly opposite to those of an encoder.

Block diagram



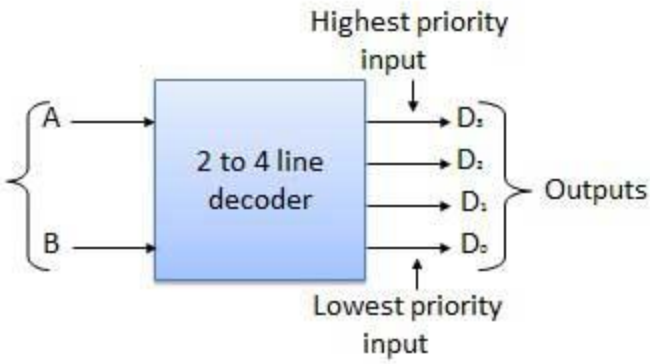
Examples of Decoders are following.

- Code converters
- BCD to seven segment decoders
- Nixie tube decoders
- Relay actuator

2 to 4 Line Decoder

The block diagram of 2 to 4 line decoder is shown in the fig. A and B are the two inputs where D through D are the four outputs. Truth table explains the operations of a decoder. It shows that each output is 1 for only a specific combination of inputs.

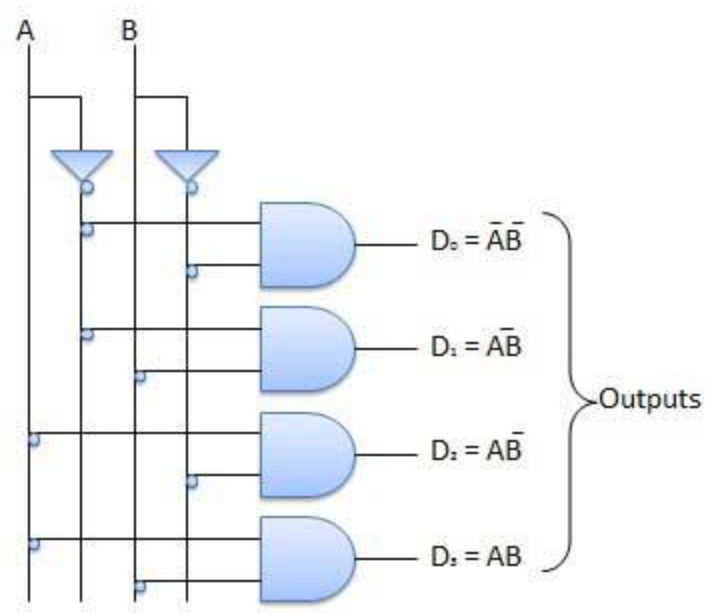
Block diagram



Truth Table

Inputs		Output			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

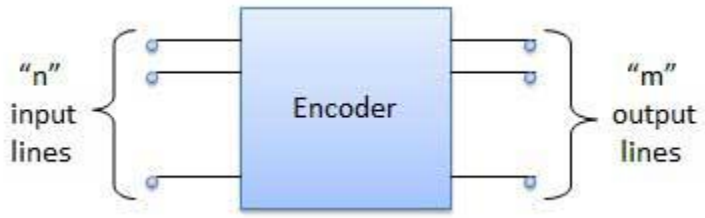
Logic Circuit



Encoder

Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder. An encoder has n number of input lines and m number of output lines. An encoder produces an m bit binary code corresponding to the digital input number. The encoder accepts an n input digital word and converts it into an m bit another digital word.

Block diagram



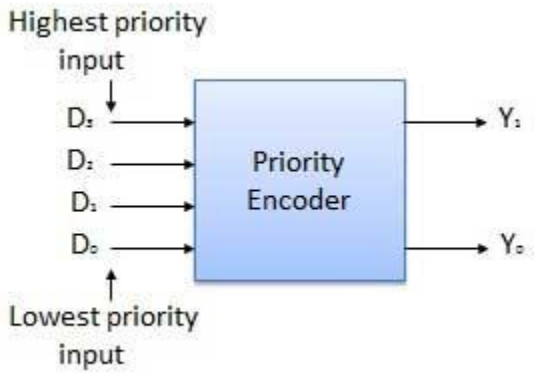
Examples of Encoders are following.

- Priority encoders
- Decimal to BCD encoder
- Octal to binary encoder
- Hexadecimal to binary encoder

Priority Encoder

This is a special type of encoder. Priority is given to the input lines. If two or more input line are 1 at the same time, then the input line with highest priority will be considered. There are four input D_0, D_1, D_2, D_3 and two output Y_0, Y_1. Out of the four input D_3 has the highest priority and D_0 has the lowest priority. That means if $D_3 = 1$ then $Y_1\ Y_1 = 11$ irrespective of the other inputs. Similarly if $D_3 = 0$ and $D_2 = 1$ then $Y_1\ Y_0 = 10$ irrespective of the other inputs.

Block diagram



Truth Table

Highest	Inputs		Lowest	Outputs	
D_3	D_2	D_1	D_0	Y_0	Y_1
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Logic Circuit