CHAPTER ONE

INTRODUCTION

The computer lies at the heart of computing. Without it most of the computing disciplines today would be a branch of theoretical mathematics. To be a professional in any field of computing today, one should acquire some understanding and appreciation of a computer system's functional components, their characteristics, their performance, and their interactions. We need to understand computer architecture in order to structure a program so that it runs more efficiently on a real machine. In selecting a system to use, they should be able to understand the tradeoff among various components, such as CPU clock speed vs. memory size.

Suppose a graduate enters the industry and is asked the most cost-effective computer for use throughout a large organization. An understating of the implications of spending more for various alternatives, such as larger cache or a higher processor clock rate, is essential to making the decision.

Digital computers use the binary number system, which has two digits; 0 and 1. *Because of the physical restriction of components., Inside the computer, there are integrated circuits with thousands of transistors. These transistors are made to operate on two-state. By this design, all the input and output voltages are either HIGH or LOW. Low voltage represents binary 0 and high voltage represents binary 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 and to develop complete sets of instructions for performing various types of computations.

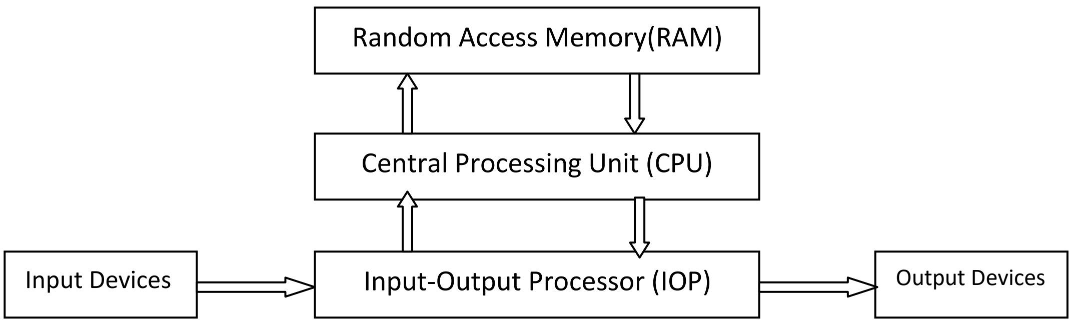
A computer system is subdivided into two functional entities: Hardware and Software.

The hardware of the computer consists of all the electronic components and electromechanical devices that comprise the physical entity of the device.

Computer software consists of the instructions and data that the computer manipulates to perform various data-processing tasks.

A sequence of instructions for the computer is called a Program.

The hardware of the computer is usually divided into three major parts.


Random Access Memory(RAM)
Central Processing Unit (CPU)
Input Devices
Input-Output Processor (IOP)
Output Devices

--Block diagram of a digital computer—

The **central processing unit (CPU)** contains an arithmetic and logic unit for manipulating (alter, edit, or move (text or data) on a computer.) data, a number of registers for storing data, and control circuits for fetching and executing instructions.

The memory of a computer contains storage for instructions and data. It is called a **Random Access Memory(RAM**) because the CPU can access any location in memory at random and retrieve the binary information within a fixed interval of time.

The **Input and output processor (IOP)** contains electronic circuits for communicating and controlling the transfer of information between the computer and the outside world. The input and output devices connected to the computer include keyboards, printers, terminals, and other communication devices.

Computer Architecture

Those attributes of the system that is **visible to a programmer**. It is concerned with the **structure and behavior of the computer** as seen by the user. Those attributes that have a **direct impact** on the execution of a program.

- Instruction sets
- Data representation – number of bits used to represent data
- Input/output mechanisms
- Memory addressing techniques

Computer Organization

The **operational units** and their **interconnections** that realize the architectural specifications. It is concerned with the **way the hardware components operate** and the way they are connected together to form the computer system.
Those hardware attributes that are transparent to the programmer.

- Control signals
- Interfaces between the computer and peripherals
- Memory technology

Computer Design

It is concerned with **hardware design of the computer**.

Once the computer specifications are formulated, it is the task of the designer to develop hardware for the system. This is sometimes referred to as computer implementation.

LOGIC GATES

A logic gate is an elementary **building block of a digital circuit**. It is a circuit with **one output and one or more inputs**. At any given moment, logic gate takes one of the two binary conditions low (0) or high (1), represented by different voltage levels.

A voltage level will represent each of the two logic values. For example +5V might represent a logic 1 and 0V might represent a logic 0.

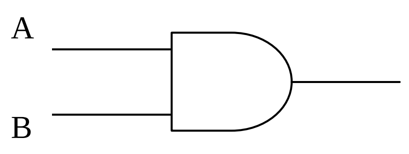
There are three fundamental logic gates namely, AND, OR and NOT. Also we have other logic gates like NAND, NOR, XOR and XNOR. Out of these NAND and NOR gates are called the **Universal Gates**. The circuit symbol and the truth table of these logic gates are explained here.

<u>AND Gates</u>

The AND Gate has two or more input signals but only one output signal. All the inputs must be high to get a high output. If we have two inputs to this AND gate and both the inputs are high then the output will be high otherwise the output will be low. All the possible inputs and outputs are shown in the following table.

Truth Table

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



Y = A . B

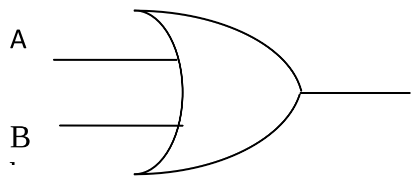
AND Gate symbol **AND function**

<u>OR Gates</u>

The OR gates has two or more input signals but only one output signal. If any input signal is high, the output signal is high. If we have two inputs to this OR gate and any of the two inputs is high then the output will be high. This can be shown in a table below with all the possible inputs and corresponding outputs.

Truth Table

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



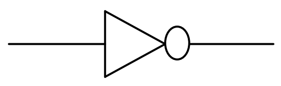
:A+B

OR Gate symbol **OR** function

<u>NOT or Inverter:</u>

A Low input produces a high output, and a high input produces low output. In binary format if the input is 0 the output will be 1 and if the input is 1 then the output will be 0. The table shows the input and output possibilities.

Input	Output
0	1
1	0



$\mathbf{Y = \overline{A}}$

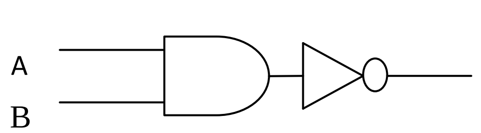
NOT Gate symbol **NOT** function

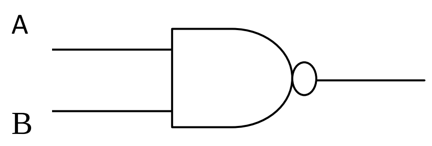
<u>NAND Gate</u>

NAND Gate is a combination of an AND gate with an inverter. An AND Gate followed by an inverter. Whatever the output of the AND gate, it will be inverted by the inverter.

This is the formation of NAND gate. The sign and the table is shown below.

The NAND operation is called Universal Operation or gate.





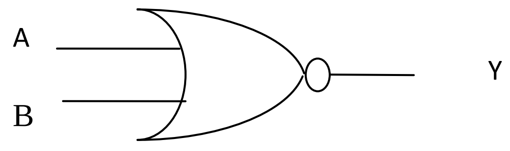
NAND Sig

$Y = \overline{AB}$

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

NOR Gate

NOR Gate is a combination of an OR gate with an inverter. An NOR Gate followed by an inverter. The NOR operation is also a Universal Operation or Gate.



NOR Sign

$Y = \overline{A+B}$

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

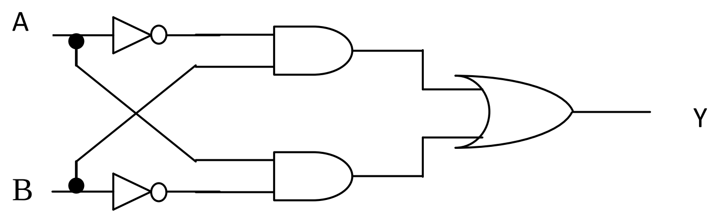
Exclusive OR Gate

An OR Gate recognizes with one or more 1s as inputs and gives output as 1. The Exclusive-OR is different; it recognizes only that have odd number of 1s. The following table shows different inputs and outputs.

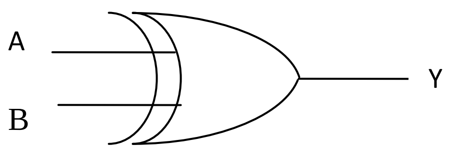
Y= A XOR B
Y= $A \oplus B$
Ex-OR sign

A	B	Y (Output)
0	0	0
0	1	1
1	0	1
1	1	0

XOR GATE Input Output

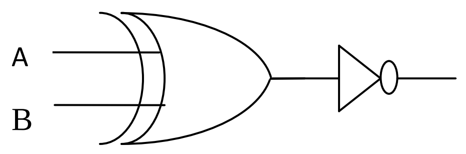


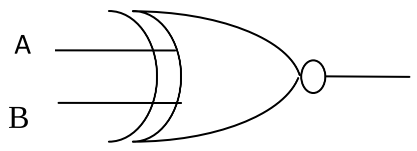
$Y = \bar{A}B + A\bar{B}$



Exclusive NOR Gate or XNOR

Exclusive NOR Gate is abbreviated as XNOR. This is logically equivalent to and XOR gate followed by an inverter. Following figure shows the XNOR gate and the table of input and outputs.





Y= A ENOR B
Y= $\overline{A \oplus B}$
Ex-NOR sign

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

BOOLEAN ALGEBRA

Boolean Algebra is mathematical system **for formulating logical statements with symbols** so that problems can be solved in a manner to **ordinary algebra**.

Boolean algebra is the mathematics of digital systems

A basic knowledge in the Boolean algebra required to study and analysis of logic circuits.

It is a convenient and systematic way of expressing and analyzing the operations of logic circuits.

Rule Number	Boolean Expression
1	$A + 0 = A$
2	$A + 1 = 1$
3	$A . 0 = 0$
4	$A.1 = A$
5	$A + A = A$
6	$A + \bar{A} = 1$
7	$A . A = A$
8	$A . \bar{A} = 0$
9	$\bar{\bar{A}} = A$
10	$A + AB = A$
11	$A + \bar{A} B = A + B$
12	$(A + B)(A + C) = A + BC$

Commutative Law :	$A + B = B + A$	$AB = BA$
Associative Law :	$A+(B+C) = (A+B)+C$	$A(BC) = (AB)C$
Distributive law:	$A(B+C) = AB + AC$	$A + (BC) = (A+B). (A+C)$
De morgan'sTheorems :	$\overline{A + B} = \overline{A} . \overline{B}$	$\overline{A.B} = \overline{A} + \overline{B}$

Karnaugh Map Method

The Karnaugh map method is a **graphical technique for simplifying Boolean functions**. It is a two-dimensional of a Truth Table. It provides a simpler method for minimizing logic expressions. The map method is ideally suited for four or less variables.
A Karnaugh map for n variables is made up of 2^n squares. Each square designates a product term of a Boolean expression. For product terms which are present in the expression, 1s are written in the corresponding squares; 0s are written in those squares which correspond to product terms not present in the expression.

Consider a map of two variables:

	$\bar{A}$	A
$\bar{B}$	$\bar{A}\,\bar{B}$	$A\,\bar{B}$
B	$\bar{A}\,B$	$A\,B$

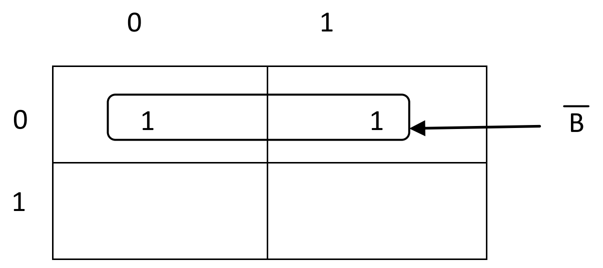
$B \backslash A$	0	1
0	$\bar{A}\,\bar{B}$	$A\,\bar{B}$
1	$\bar{A}\,B$	$A\,B$

In the map, 0 represents $\bar{A}$, and 1 represents A. Similarly, for variable B.

For example, for the Boolean functions $Y = A\bar{B} + AB$

	0	1
0		1
1		1

Example : simplify $Y = \bar{A}\,\bar{B} + A\bar{B}$



Two adjacent squares containing 1 have been grouped together. To show the grouping, they have been encircled. For simplification we have to see that which variable is common to both squares. In this case $\bar{B}$ is common to both as the 1st row is for $\bar{B}$.

So their simplification will result $Y = \bar{B}$
This can be verified also algebraically as follows:

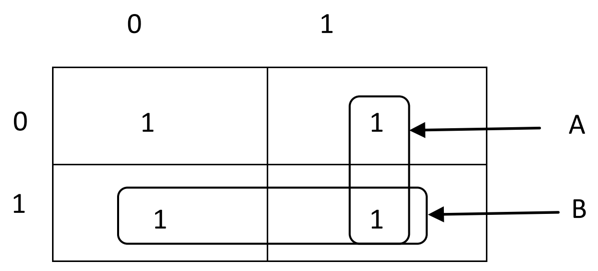
$$Y = \bar{A}\,\bar{B} + A\bar{B}$$

$$= \bar{B}\,(\bar{A} + A)$$

$= \overline{B}$

So the variable which is common to adjacent squares is selected, and the variable which is not common is discarded.

Example : simplify $Y = \overline{A}B + AB + A\overline{B}$



Simplification result $Y = \overline{A}B + AB + A\overline{B} = B + A$

The result obtained by map method can also be verified algebraically as follows:

$$Y = \overline{A}B + AB + A\overline{B}$$

$$= \overline{A}B + AB + AB + A\overline{B}$$

$$= B(\overline{A} + A) + A(B + \overline{B})$$

$$= B + A$$

Karnaugh Map for Three variables:

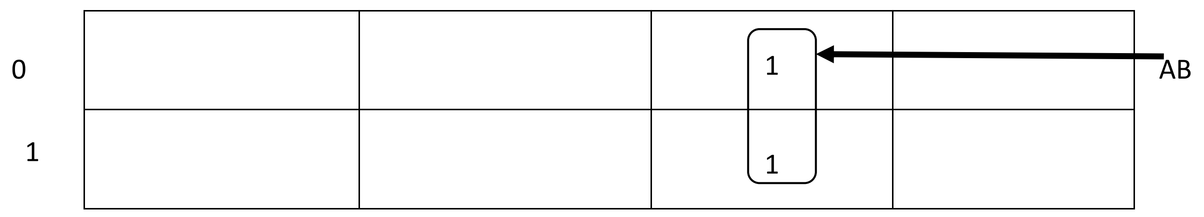
	$\overline{A}\,\overline{B}$	$\overline{A}B$	AB	$A\overline{B}$
$\overline{C}$	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\,\overline{C}$
C	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

The ordering of the variables, ie., 00,01,11,10 is in gray code.

C \ AB	00	01	11	10
0	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\,\overline{C}$
1	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

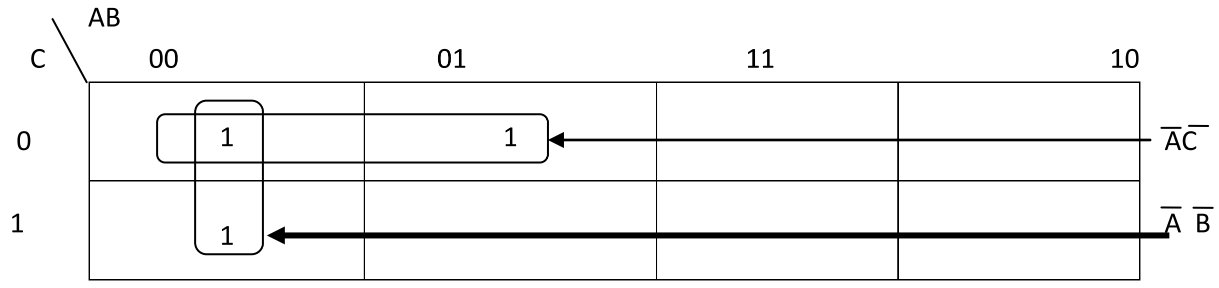
Example: simplifying the function $Y = AB\overline{C} + ABC$

C \ AB	00	01	11	10



The simplified function will be $Y = AB$

Example: simplifying the function $Y = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,B\,\bar{C} + \bar{A}\,\bar{B}\,C$



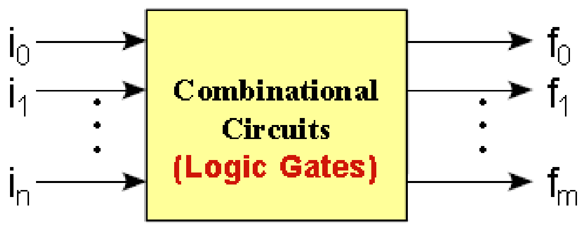
The simplified function will be $Y = \bar{A}\,\bar{B} + \bar{A}\,\bar{C}$

COMBINATORIAL CIRCUITS AND FLIP FLOPS

Combinational Circuits:

A connected arrangement of logic gates with a set of inputs and outputs

Block diagram of a combinational circuit



Combinational circuits are those logic circuits whose operations can be completely described by a truth table / Boolean expression. A combinational circuit is realized using AND, OR, NOT gates (or NAND OR NOR gates). Examples of combinational circuits are: adder, subtractors, code converters, decoders, encoders, digital multiplexers, demultiplexers, programmable logic arrays, ROMs etc., Logic circuits for some important arithmetic operations are half-adder and full adder.

HALF ADDER:

A logic circuit which performs addition of two binary bits is called a half-adder.
Truth table for the addition of two binary bits.

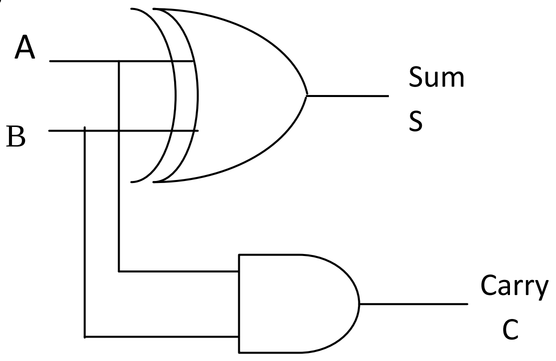
Inputs		Outputs	
A	B	Sum S	Carry C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

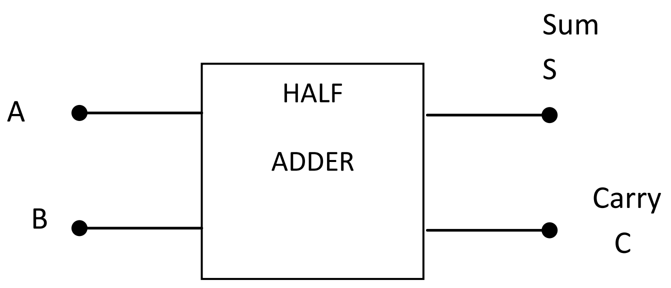
It is concluded that the sum is equal to A XOR B. It means that the outputs of an EXCLUSIVE-OR gate will give the sum. The carry is equal to A AND B. The output of an AND gate will give the carry.

$S = \overline{A}\ B + A\ \overline{B}$

$= A \oplus B$

$C = AB$





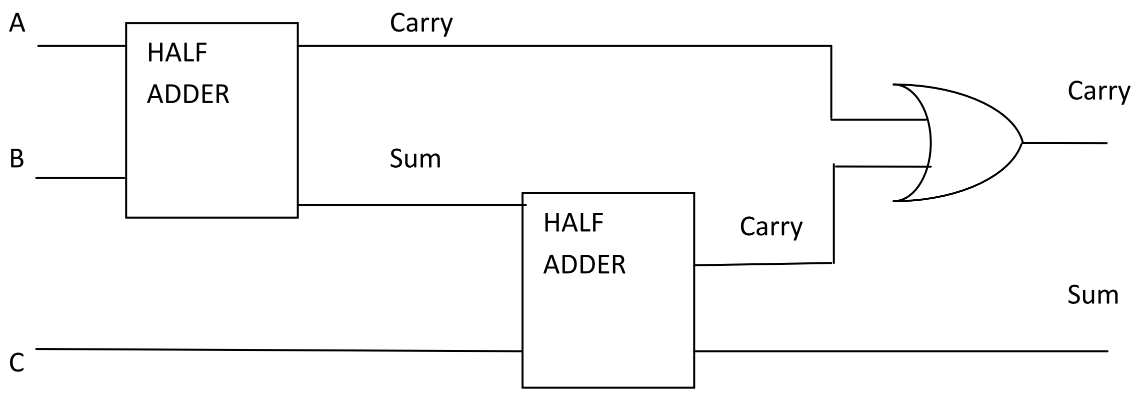
FULL ADDER

A logic circuit which performs addition of three binary bits is called a Full-adder.

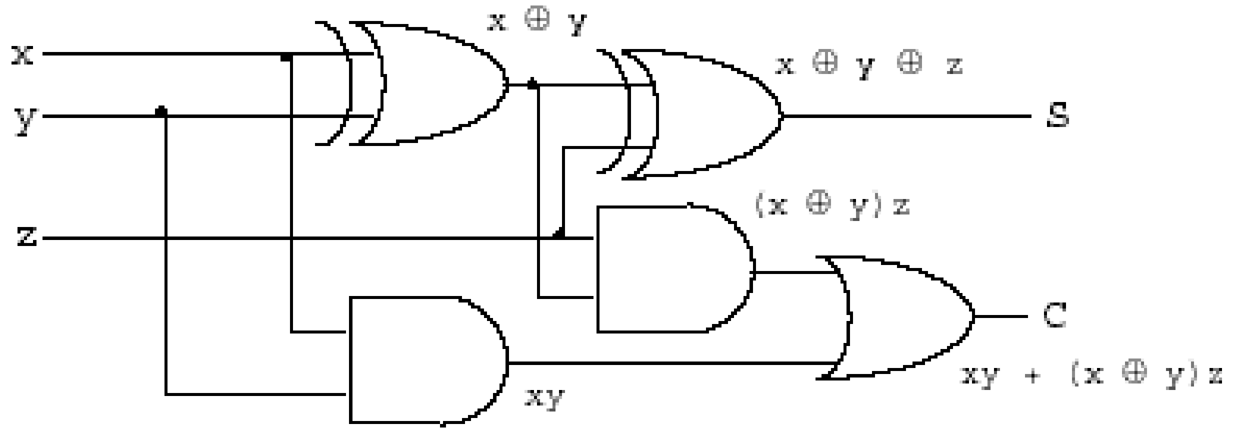
A full adder can be built using two half adders and an OR gate.

Inputs			Outputs	
A	B	C	Sum S	Carry C
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

$S = x \oplus y \oplus z$
$C = z\ (x \oplus y) + xy$



The logic diagram for the full adder



FLIP FLOPS:

Flip flops is a binary cell capable of storing one bit of information.

The flip flop is a **bistable** device. It exists in one of two states and, in the absence of input, remains in that state. Thus, the flip flop can function as a 1 bit memory.

The flip flop has two outputs, which are always the complementary of each other; these are generally labelled Q and $\overline{Q}$

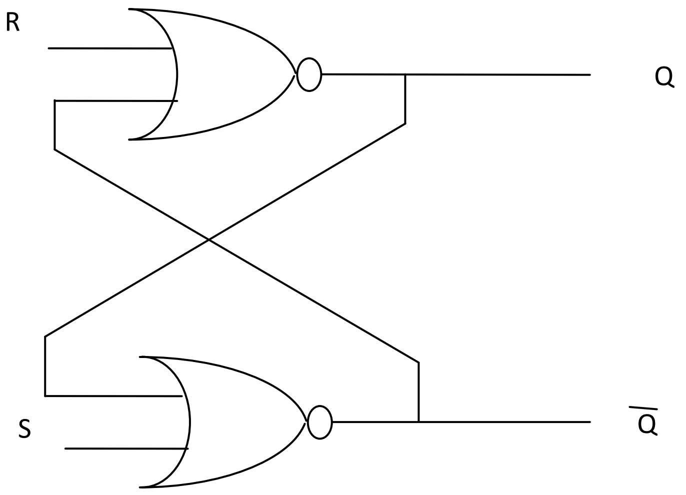
The S-R Flip flop:

The circuits has two inputs, s(Set) and R (Reset) and two outputs $\overline{Q}$ and Q and consists of two NOR gates connected in a feedback arrangement.

First, Let us show that the circuit is bistable. Assume that both S and R are 0 and that Q is 0. The inputs to the lower NOR gate are Q = 0 and S = 0. Thus, the output $\overline{Q}$ = 1 means that the inputs to the upper NOR gate are Q=1 and R=0, which has the output $\overline{Q}$ = 0.

Thus, the state of the circuit is internally consistent and remains stable as long as S = R = 0. A similar line of reasoning shows that the state Q = 1, $\overline{Q}$ = 0 is also stable for R = S = 0.

Thus, this circuit can function as a 1-bit memory. Suppose that S changes to the value 1. Now the inputs to the lower NOR gate are S =1 , Q =0. After some time delay t, the output of the lower NOR gate will be $\overline{Q}$ = 0.



characteristic Table

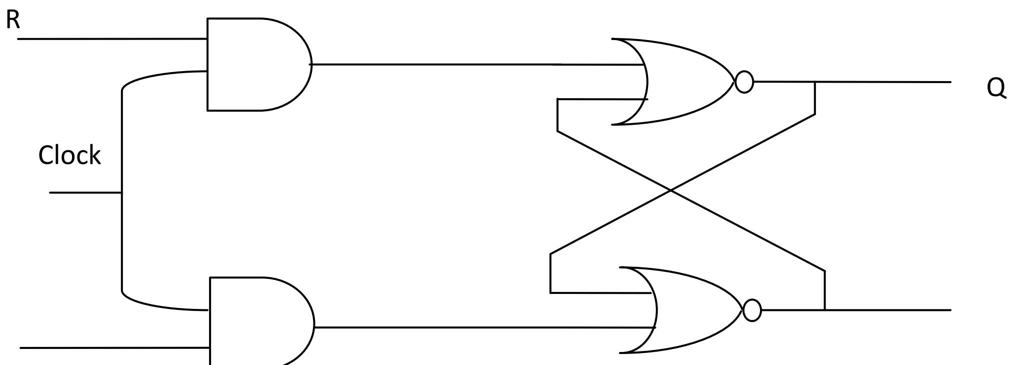
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	--

So, at this point in time, the inputs to the upper NOR gate become R=0 , $\overline{Q}$ = 0. After another gate delay of t, the output Q becomes 1. This is again a stable state. The inputs to the lower gate are now S = 1, $\overline{Q}$ =1, which maintain the output Q=0. As long as S =1 and R =0, the outputs will remain Q =1, Q = 0. Furthermore, if S returns to 0, the outputs will remain unchanged.

Observe that the inputs S = 1, R =1 are not allowed, because these would produce an inconsistent output (both $\overline{Q}$ and Q equal 0).

Clocked S-R Flip-Flop:

The output of the S-R latch changes, after a brief time delay, in response to a change in the input. This is referred to as asynchronous operation. More typically, events in the digital computer are synchronized to a clock pulse, so that changes occur only when a clock pulse occurs. This type of device is referred to as a clocked S-R flip-flop. Note that the R and S inputs are passed to the NOR gates only during the clock pulse.



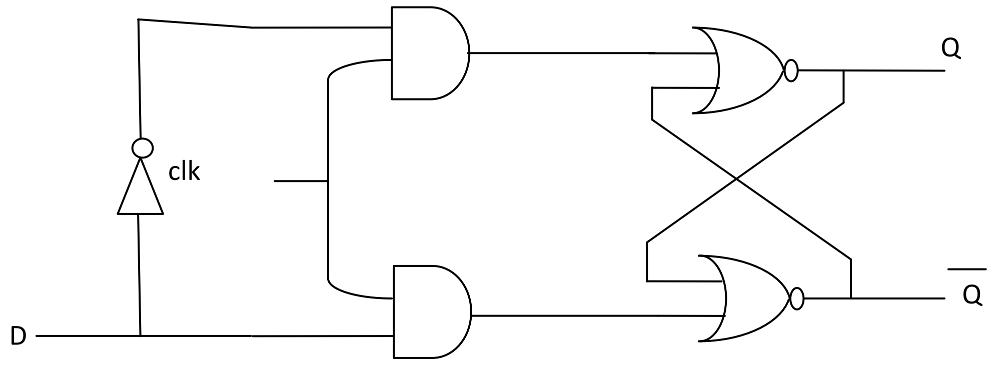
$\overline{Q}$

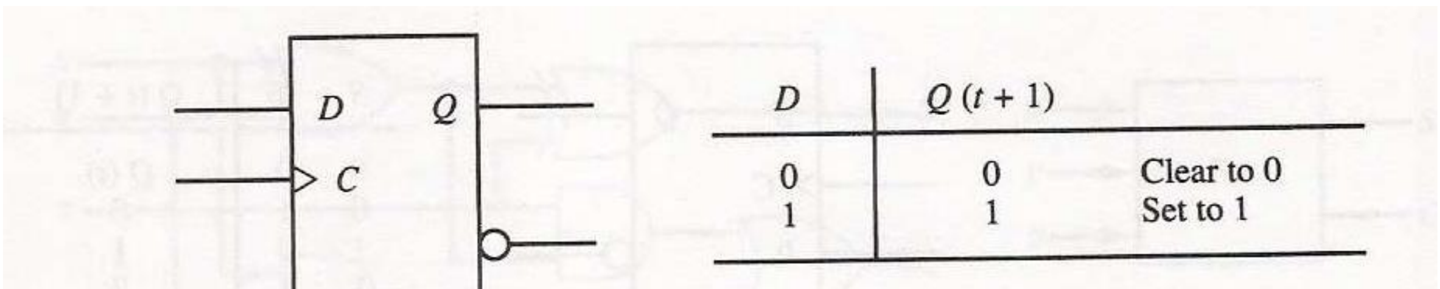
S

<u>D Flip – Flop:</u>

One problem with S-R flip flop is that the condition R = 1, S = 1 must be avoided. One way to do this is to allow just a single input. The D flip flop accomplishes this. By using an inverter, the two AND gates are guaranteed to be the opposite of each other.

The D flip flop is sometimes referred to as the data flip flop because it is, in effect, storage for one bit of data. the output of the D flip flop is always equal to the most recent value applied to the input. Hence, it remembers and produces the last input. It is also referred to as the delay flip flop, because it delays a 0 or 1 applied to its input for a single clock pulse.

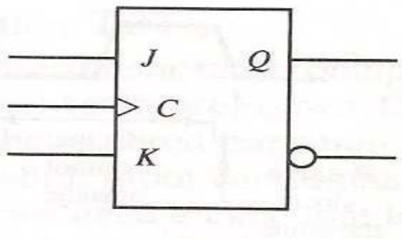




D	Q (t + 1)	
0	0	Clear to 0
1	1	Set to 1

J-K Flip-Flop:

Like the S-R flip flop, it has 2 inputs. However, in this case all possible combinations of input values are valid. In its characteristic table, we can note that the first three combinations are the same as for the S-R flip-flop. With no input, the output is stable. The J input alone performs a set function, causing the output to be 1; the K input alone performs a reset function, causing the output to be 0. When both J and K are 1, the function performed is referred to as the toggle function: the output is reversed.



(a) Graphic symbol

J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	$Q'(t)$	Complement

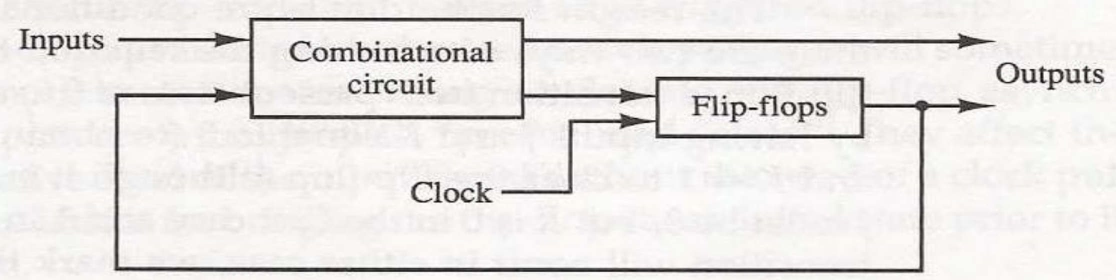
(b) Characteristic table

<u>SEQUENTIAL CIRCUITS</u>

A sequential circuit is **an interconnection of Flip-flops and Gate**. A sequential circuit consists of a combinational logic and storage elements. The output of a sequential circuit is not only a function of a present inputs but also of past inputs.
The state of the storage elements depends upon the preceding inputs and the preceding states of the elements. To realize sequential circuits in addition to AND, OR and NOT gates, flip-flops are also required.
Examples of sequential circuits are: registers, shift registers, counters, etc.,

The two major uses of sequential circuits in digital systems are:

1) As memories to store information while processing
2) As control circuits to generate control signals which are essential to select and enable a sequence of data transfer or data processing steps in the execution of multistep tasks.



The sequential circuits which employ clock are called **synchronous sequential circuits**. In a synchronous sequential circuit all memory elements are clocked latches or clocked flip flops. The design and operation of sequential circuits is greatly simplified by the use of clock signals.

The sequential circuits which do not employ clock are known as **unclocked or asynchronous sequential circuits**. Unclocked sequential circuits are difficult to design and therefore, they are relatively uncommon.