Unit-2
Boolean Algebra and Logic Gates

Unit-2
Boolean Algebra and Logic Gates

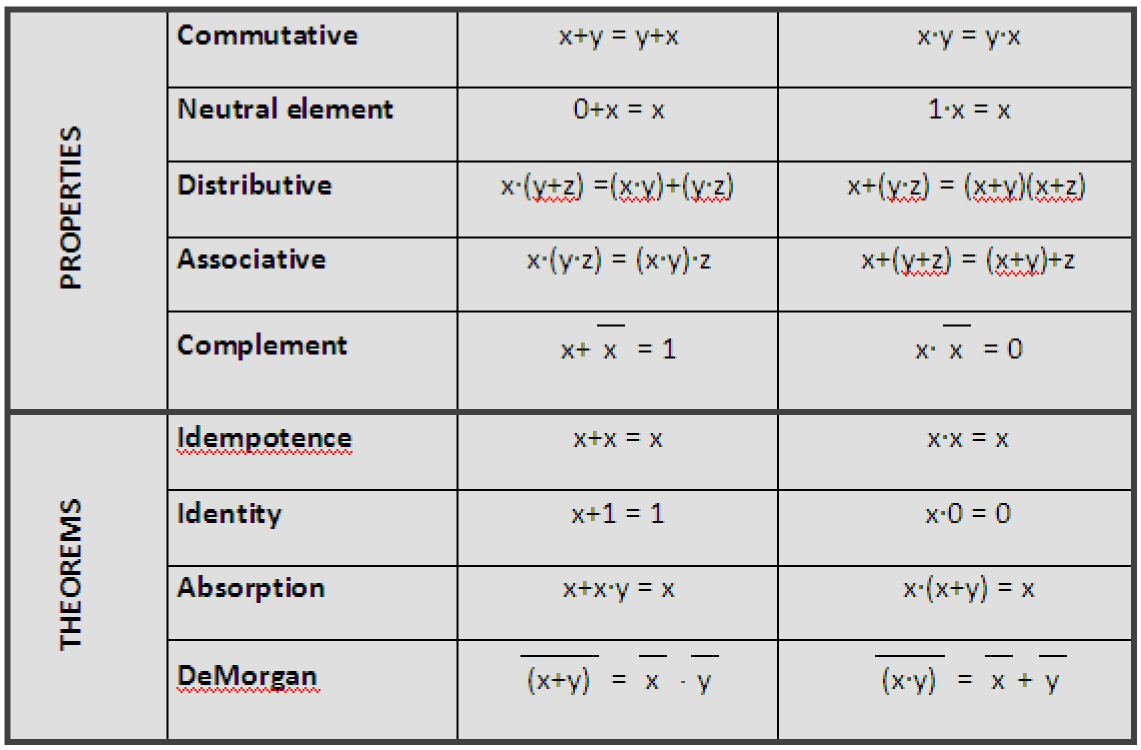
Introduction

- In 1854 George Boole introduced a systematic treatment of logic and developed for this purpose an algebraic system known as symbolic logic, or **Boolean algebra**.
- Boolean algebra is a branch of mathematics and it can be used to describe the manipulation and processing of **binary** information. The two-valued Boolean algebra has important application in the design of modern computing systems.

Boolean Algebra

- Boolean algebra is algebra for the manipulation of objects that can take on only **two** values, typically true and false.
- It is common to interpret the digital value **0** as false and the digital value **1** as true.
- A two-valued Boolean algebra is defined on a set of 2 elements B = {0, 1} with 3 binary operators OR (+), AND (•), and NOT (').

Basic Theorem and Properties of Boolean Algebra



PROPERTIES	**Commutative**	$x+y = y+x$	$x\cdot y = y\cdot x$
	Neutral element	$0+x = x$	$1\cdot x = x$
	Distributive	$x\cdot(y+z) = (x\cdot y)+(y\cdot z)$	$x+(y\cdot z) = (x+y)(x+z)$
	Associative	$x\cdot(y\cdot z) = (x\cdot y)\cdot z$	$x+(y+z) = (x+y)+z$
	Complement	$x+\overline{x} = 1$	$x\cdot\overline{x} = 0$
THEOREMS	**Idempotence**	$x+x = x$	$x\cdot x = x$
	Identity	$x+1 = 1$	$x\cdot 0 = 0$
	Absorption	$x+x\cdot y = x$	$x\cdot(x+y) = x$
	DeMorgan	$\overline{(x+y)} = \overline{x}\cdot\overline{y}$	$\overline{(x\cdot y)} = \overline{x}+\overline{y}$

Duality Principle

It states that "***Every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged***". In a two-valued Boolean algebra, the identity elements and the elements of the set *B* are the same: 1 and 0. If the ***dual*** of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

E.g. $X \cdot Y + Z' = (X' + Y') \cdot Z$

De-Morgans Theorem

De Morgan’s theorem is used to convert OR type of expression into AND type and vice-versa.It is further divided into two different types;

1st law:
It state that the total complement of sum is equal to the product of individual complement.
i.e. (A+B)’=A’ ⋅B’

Proof:

Input		Output	
A	B	(A+B)’	A’ ⋅B’
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

2nd law:
It state that the total complement of the product is equal to the sum of individual complement.
i.e. (A⋅B)’ =A’+B’

Proof:

Input		Output	
A	B	(A⋅B)’	A’+B’
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Boolean Function

A binary variable can take a value of 0 or, 1. A Boolean function is an expression formed with binary variables, the two binary operators OR and AND, unary operator NOT, parenthesis and an equal sign. For a given value of variables, the function either can 0 or 1.

E.g.

$F_1 = xyz'$

$F_2 = x + y'z$

$F_3 = x'y'z + x'yz + xy'$

$F_4 = xy' + x'z$

Truth Table of Boolean functions:

A **truth table** shows the **relationship**, in tabular form, between the input values and the result of a specific Boolean operator or function on the input variables.

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

Here, F_3 and F_4 are same. Two functions of n binary variables are said to be equal if they have same values for all possible 2^n combinations of the n variables.

Algaberic Manipulation

When a Boolean function is implemented with logic gates, each literal in the function designates an input to a gate and each term is implemented with a gate. The minimization of the number of literals and the number of terms results is a circuit with less equipment.

Q. Simplify the following Boolean functions to a minimum number of literals.

1. $x(x' + y)$
   $= xx' + xy$
   $= 0 + xy$
   $= xy$

2. $x + x'y$
   $= (x + x')(x + y)$
   $= 1(x + y)$
   $= x + y$

3. $(x + y)(x + y')$
   $= x + xy + xy' + yy'$
   $= x(1 + y + y')$
   $= x$

4. $xy + x'z + yz$
   $= xy + x'z + yz(x + x')$
   $= xy + x'z + xyz + x'yz$
   $= xy(1 + z) + x'z(1 + y)$
   $= xy + x'z$

Q. Prove that: $(\boldsymbol{x} + \boldsymbol{y})(\overline{\boldsymbol{x}} + \boldsymbol{y}) = \boldsymbol{y}$

Sol^n:

Proof	Identity Name
$(x+y)(\overline{x}+y) \quad = x\overline{x}+xy+y\overline{x}+yy$	Distributive Law
$= 0+xy+y\overline{x}+yy$	Inverse Law
$= 0+xy+y\overline{x}+y$	Idempotent Law
$= xy+y\overline{x}+y$	Identity Law
$= y(x+\overline{x})+y$	Distributive Law (and Commutative Law)
$= y(1)+y$	Inverse Law
$= y+y$	Identity Law
$= y$	Idempotent Law

Note: To prove the equality of two Boolean expressions, you can also create the truth tables for each and compare. If the truth tables are **identical**, the expressions are **equal**.

Q. Prove the Boolean expression: $\boldsymbol{AB + AB'C + A'BC = AB + AC + BC}$

Sol^n:

$$
\begin{aligned}
& AB + AB'C + A'BC \\
&= A(B + B'C) + A'BC \\
&= A(B + C) + A'BC \qquad\qquad [\therefore A + A'B = A + B] \\
&= AB + AC + A'BC \\
&= AB + (A + A'B)C \\
&= AB + (A + B)C \\
&= AB + AC + BC
\end{aligned}
$$

Q. Minimize the expression: $\boldsymbol{AB + A\overline{C} + BC}$ ***using theorem and properties of Boolean Algebra.***

Sol^n:

$$
\begin{aligned}
AB + A\overline{C} + BC &= AB(C + \overline{C}) + A\overline{C} + BC \\
&= ABC + AB\overline{C} + A\overline{C} + BC \\
&= BC(1 + A) + A\overline{C}(1 + B) \\
&= BC + A\overline{C}
\end{aligned}
$$

Q. Prove that: $\boldsymbol{\overline{\overline{A\overline{B}C} + \overline{ACD} + B\overline{C}} = A\overline{B}CD}$

Sol^n:

$$
\begin{aligned}
\overline{\overline{A\overline{B}C} + \overline{ACD} + B\overline{C}} &= \overline{(\overline{A} + B + \overline{C}) + (\overline{A} + \overline{C} + \overline{D}) + B\overline{C}} \\
&= \overline{(\overline{A} + B + \overline{C} + \overline{D}) + B\overline{C}} \\
&= \overline{(\overline{A} + B + \overline{C} + \overline{D})} \\
&= A\overline{B}CD
\end{aligned}
$$

Complement of a function

The complement of a function F is F' and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of F. The complement of a function may be derived algebraically through De Morgan's theorem.

$$\begin{aligned}
(A+B+C)' &= (A+X)' && \text{let } B+C=X \\
&= A'X' && \text{by theorem 5(a) (DeMorgan)} \\
&= A'\cdot(B+C)' && \text{substitute } B+C=X \\
&= A'\cdot(B'C') && \text{by theorem 5(a) (DeMorgan)} \\
&= A'B'C' && \text{by theorem 4(b) (associative)}
\end{aligned}$$

Generalized theorems for finding complement:

$$(A+B+C+D+\cdots+F)' = A'B'C'D'\cdots F'$$

$$(ABCD\cdots F)' = A'+B'+C'+D'+\cdots+F'$$

Q. Find the Complement of the functions $F_1 = x'yz' + x'y'z$ and $F_2 = x(y'z' + yz)$.

Sol^n^:

By applying DeMorgan's theorem as many times as necessary, the complements are obtained as follows:

$$F_1' = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x+y'+z)(x+y+z')$$

$$\begin{aligned}
F_2' &= [x(y'z'+yz)]' = x' + (y'z'+yz)' = x' + (y'z')'\cdot(yz)' \\
&= x' + (y+z)(y'+z')
\end{aligned}$$

<u>**Logic Gates**</u>

- A logic gate is an electronic device that produces **a result** based on one or more input values.
- In reality, gates consist of one to six **transistors**, but digital designers think of them as a single unit.
- Integrated circuits contain collections of gates suited to a particular purpose.

Logic gate can be categories as follows:

1. Basic Gate

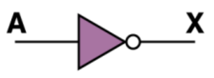
a. <u>NOT gate</u>

A NOT gate accepts one input value and produces one output value.

Boolean Expression

$X = A'$

Logic Diagram Symbol



Truth Table

A	X
0	1
1	0

Fig: Various representations of a NOT gate

By definition, if the input value for a NOT gate is 0, the output value is 1, and if the input value is 1, the output is 0.

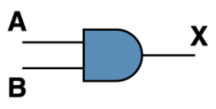
b. <u>AND gate</u>

An AND gate accepts two input signals. If the two input values for an AND gate are both 1, the output is 1; otherwise, the output is 0.

Boolean Expression

$X = A \cdot B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Fig: Various representations of a AND gate

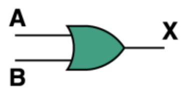
c. <u>OR gate</u>

If the two input values are both 0, the output value is 0; otherwise, the output is 1.

Boolean Expression

$X = A + B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Fig: Various representations of a OR gate

2. Universal Gate

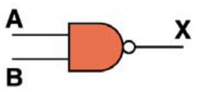
a. <u>NAND gate</u>

NAND gate is the combination of NOT gate and AND gate. If the two input values for an NAND gate are both 1, the output is 0; otherwise, the output is 1.

Boolean Expression

$X = (A \cdot B)'$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Fig: Various representations of a NAND gate

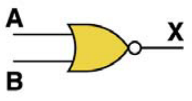
b. <u>NOR gate</u>

NOR gate is the combination of NOT gate and OR gate. If the two input values for NOR gate are both 0, the output value is 1; otherwise, the output is 0.

Boolean Expression

$X = (A + B)'$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Fig: Various representations of a NOR gate

3. Derived/Extended Gate

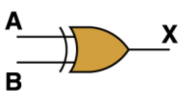
a. <u>Exclusive-OR gate (XOR)</u>

An XOR gate produces 0 if its two inputs are the same, and a 1 otherwise.

Boolean Expression

$$X = A \oplus B$$
$$= AB' + A'B$$

Logic Diagram Symbol



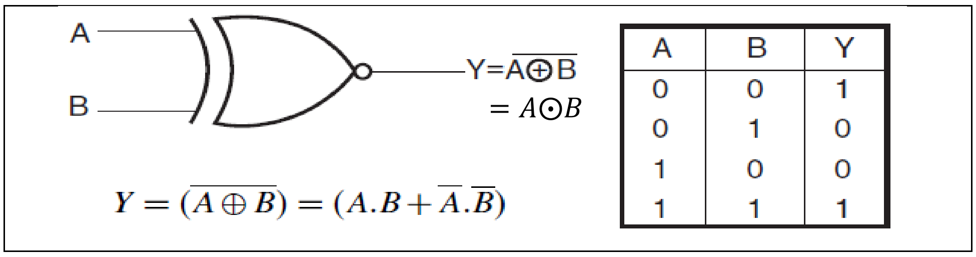
Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Fig: Various representations of a XOR gate

b. <u>Exclusive-NOR gate (X-NOR)</u>

X-NOR is the complement of X-OR. An X-NOR gate produces 1 if its two inputs are the same, and a 0 otherwise.



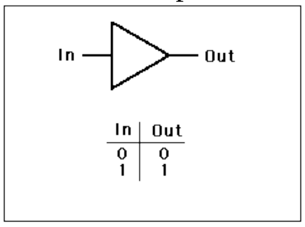
$$Y = (\overline{A \oplus B}) = (A.B + \overline{A}.\overline{B})$$

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

Fig: Various representations of a X-NOR gate

Buffer Gate

The buffer gate returns the same output as same as that of input.



In	Out
0	0
1	1

Universal Gate Realization

A universal gate is a gate which can implement any Boolean function without need to use any other gate type. The **NAND** and **NOR** gates are universal gates.

1. ***NAND gate as a Universal Gate***
   To prove that any Boolean function can be implemented using only NAND gates, we will show that the AND, OR, and NOT operations can be performed using only these gates.

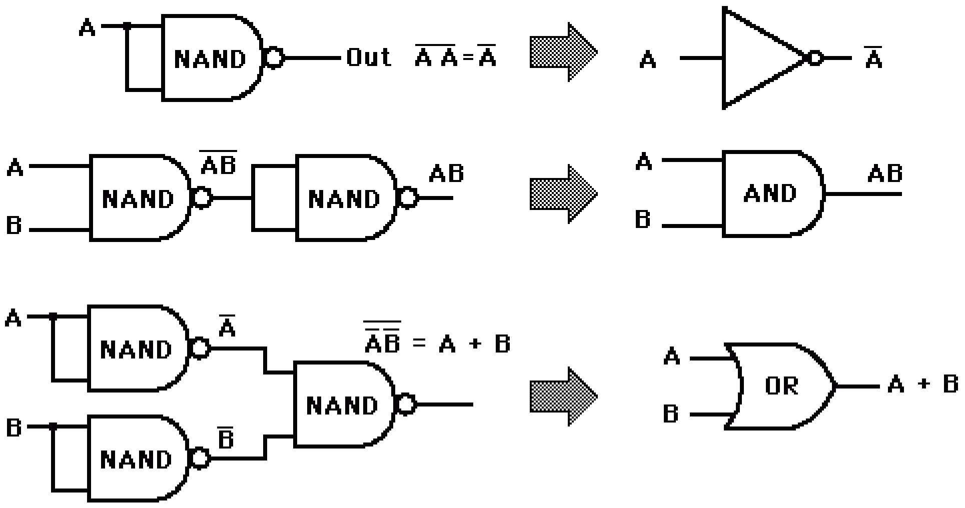


Fig: Three Circuits Constructed Using Only NAND Gates

Thus, the **NAND gate** is a universal gate since it can implement the AND, OR and NOT functions.

2. ***NOR gate as a Universal Gate***
   To prove that any Boolean function can be implemented using only NOR gates, we will show that the AND, OR, and NOT operations can be performed using only these gates.

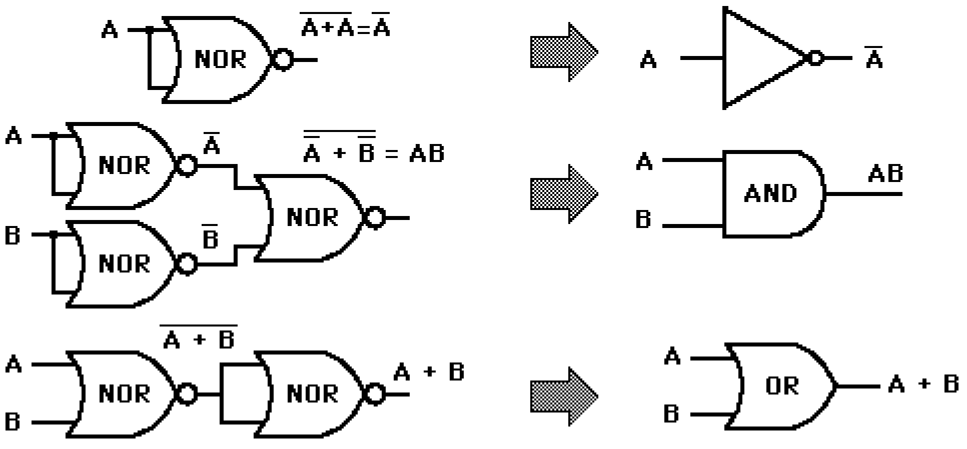


Fig: Three Circuits Constructed Using Only NOR Gates

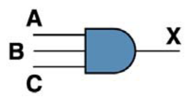
Thus, the **NOR gate** is a universal gate since it can implement the AND, OR and NOT functions.

Gates with More Inputs

- Gates can be designed to accept three or more input values
- A three-input AND gate, for example, produces an output of 1 only if all input values are 1.

Boolean Expression **Logic Diagram Symbol** **Truth Table**

$X = A \cdot B \cdot C$



A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

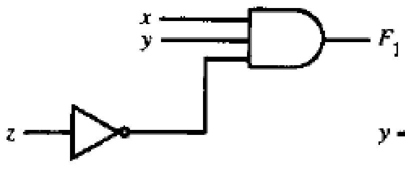
Fig: Various representations of a three input AND gate

Implementation of Boolean function using gates

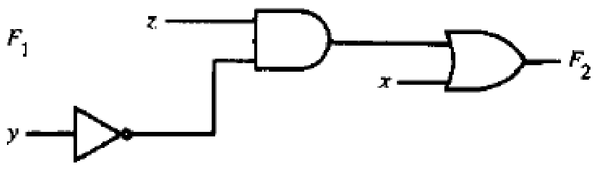
$F_1 = xyz'$

$F_2 = x + y'z$

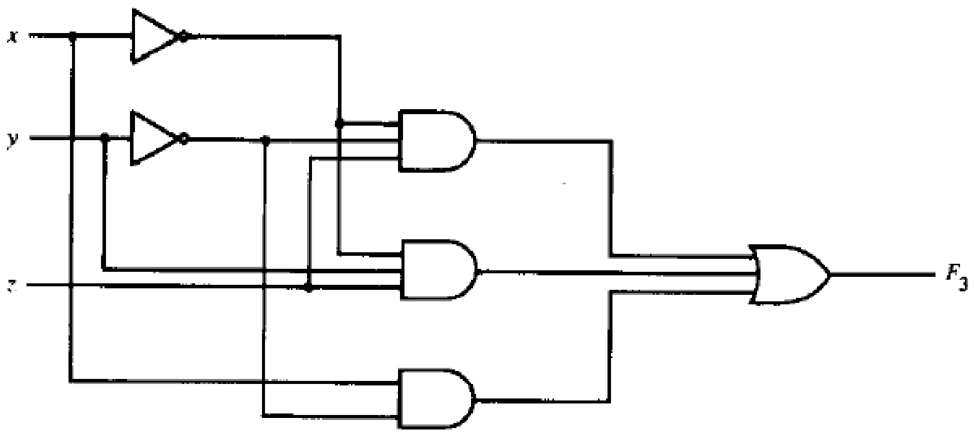
$F_3 = x'y'z + x'yz + xy'$



(a) $F_1 = xyz'$



(b) $F_2 = x + y'z$



(c) $F_3 = x'y'z + x'yz + xy'$

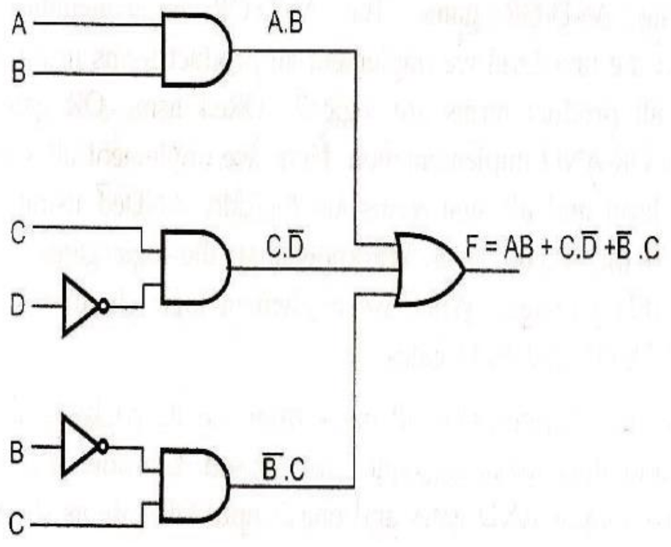
Q. Draw a logic gates that implements the following:

a. F = AB + C D' + B' C

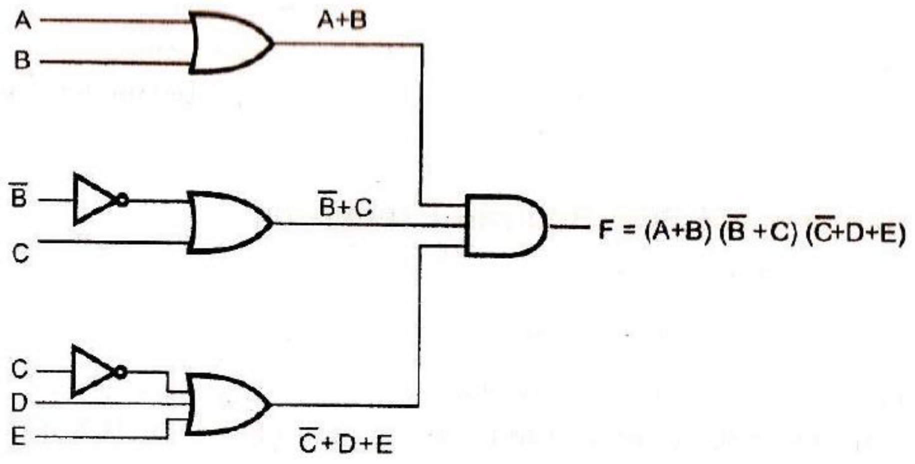
b. F = (A + B) (B' + C) (C' + D + E)

Sol^n:

a.



b.



<u>Integrated Circuits (ICs)</u>

An Integrated circuit is an association (or connection) of various electronic devices such as resistors, capacitors and transistors etched (or fabricated) to a semiconductor material such as silicon or germanium. It is also called as a **chip** or **microchip**. An IC can function as an amplifier, rectifier, oscillator, counter, timer and memory. Sometime ICs are connected to various other systems to perform complex functions.

ICs can be categorized into two types
- Analog or Linear ICs
- Digital or logic ICs

Further there are certain ICs which can perform as a combination of both analog and digital functions.

Analog or Linear ICs : They produce continuous output depending on the input signal. From the name of the IC we can deduce that the output is a linear function of the input signal. Op-amp (operational amplifier) is one of the types of linear ICs which are used in amplifiers, timers and counters, oscillators etc.

Digital or Logic ICs : Unlike Analog ICs, Digital ICs never give a continuous output signal. Instead it operates only during defined states. Digital ICs are used mostly in microprocessor and various memory applications. Logic gates are the building blocks of Digital ICs which operate either at 0 or 1.

<u>Levels of Integration</u>:

Small Scale Integration (SSI) devices contain several independent gates in a single package. The inputs and outputs of the gates are connected directly to the pins in the package. The number of gates is usually fewer than 10 and is limited by number of pins available in the IC.

Medium-scale integration (MSI) devices have a complexity of approximately 10 to 100 gates in a single package. They usually perform specific elementary digital operations such as decoders, adders or multiplexers.

Large-scale integration (LSI) devices contain between 100 and a few thousand gates in a single package. They include digital systems such as processor, memory chips and programmable logic devices.

Very large-scale integration (VLSI) devices contain thousands of gates within a single package. Examples are large memory arrays and complex microcomputer chips. Because of their small size and low cost, VLSI devices have revolutionized the computer system design technology, giving the designer the capabilities to create structures that previously were uneconomical.

IC digital Logic Families

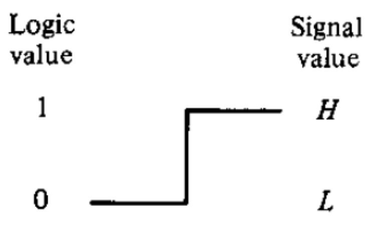
Many different logic families of digital IC's have been introduced commercially.

TTL	Transistor-transistor logic.
ECL	Emitter-coupled logic.
MOS	Metal-oxide semiconductor
CMOS	Complementary metal-oxide semiconductor.
I^2C	Integrated-injection logic

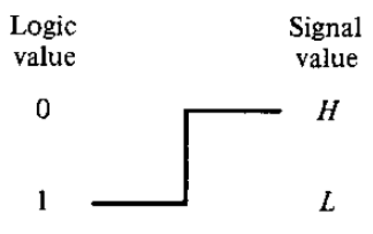
- ✓ **TTL** has extensive list of digital functions and currently the most popular logic family.
- ✓ **ECL** is used in systems requiring high-speed operations.
- ✓ **MOS** and **I^2C** are used in circuits requiring high component density and
- ✓ **CMOS** is used in systems requiring low power consumption.

Positive and Negative Logic

The binary signal at the inputs and outputs of any gate has one of two values, except during transition. One signal value represents logic-1 and the other logic-0. So there is a possibility of two different assignments of signal level to logic value, as shown in Fig.



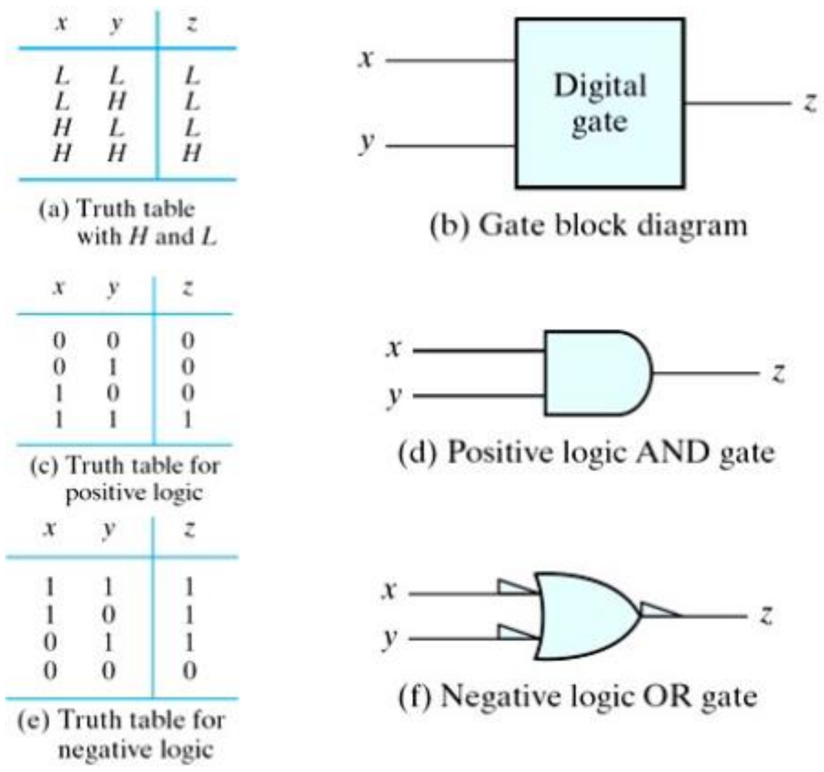
(a) Positive logic



(b) Negative logic

Positive logic: H = 1; L = 0

Negative Logic: H = 0; L = 1



x	y	z
L	L	L
L	H	L
H	L	L
H	H	H

(a) Truth table with H and L

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

(c) Truth table for positive logic

x	y	z
1	1	1
1	0	1
0	1	1
0	0	0

(e) Truth table for negative logic

Fig: Demonstration of positive and negative logic

<u>Special Characteristics</u>

The characteristics that describe the performance of IC digital logic families are: Fan-out, power dissipation, propagation delay and noise margin.

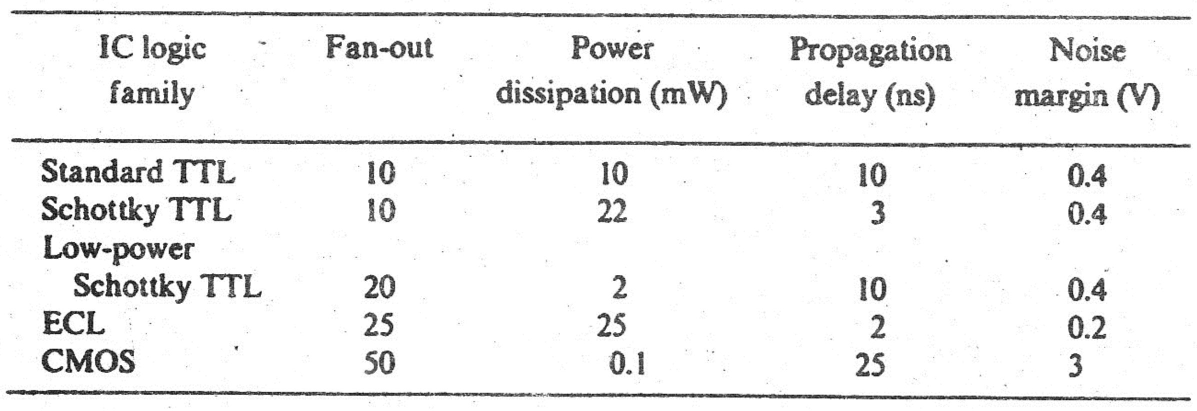
- **Fan-out** specifies the number of standard loads that the output of a gate can drive without impairing its normal operation. A standard load is usually defined as the amount of current needed by an input of another logic gate in the same IC family. Sometimes the term loading is used instead of fan-out. This term is derived from the fact that the output pin of a gate can supply limited current, above which it ceases to operate properly and is said to be overloaded.

- **Power dissipation** is the supplied power required to operate the gate. This parameter is expressed in milliwatts (mW) and represents the actual power dissipated in the gate.

- **Propagation delay** is the average transition delay time for a signal to propagate from input to output when the binary signals change in value. Propagation delay is expressed in nanoseconds (ns).

- **Noise margin** is the maximum noise voltage added to the input signal of a digital circuit that does not cause an undesirable change in the circuit output. There are two types of noise to be considered. ***DC noise*** is caused by a drift in the voltage level of a signal. ***AC noise*** is a random pulse that may be created by other switching signals. Noise margin is expressed in volts (V) and represent the maximum noise signal that can be tolerated by the gate.

Typical characteristics of IC logic family



IC logic family	Fan-out	Power dissipation (mW)	Propagation delay (ns)	Noise margin (V)
Standard TTL	10	10	10	0.4
Schottky TTL	10	22	3	0.4
Low-power Schottky TTL	20	2	10	0.4
ECL	25	25	2	0.2
CMOS	50	0.1	25	3

References:

- ***M. Morris Mano, "Digital Logic & Computer Design"***