Mahavir Polytechnic, Nashik

Department of Artificial Intelligence and Machine Learning

Year: SY **Subject: DTE (313303)**

UNIT 2: Logic Gates and Boolean Algebra **Marks: 12**

Course Outcome 2: Apply Boolean laws to minimize complex Boolean function.

Syllabus:

2.1 Logic Families: Characteristics Parameters of logic Families- Noise margin, Power dissipation, Figure of merit, Fan in, Fan out, Speed of operation Current and Voltage parameters, Comparison of TTL, CMOS and ECL logic family

2.2 Introduction to positive and negative logic systems, Logic Gates: Symbol, Truth table of Basic logic gates (AND, OR, NOT) Universal gates (NAND, NOR) and Special purpose gates (EX-OR, EX-NOR)

2.3 Buffer: Tristate logic, Unidirectional and Bidirectional

2.4 Boolean algebra: Laws of Boolean algebra, Duality Theorem, De-Morgan's theorem

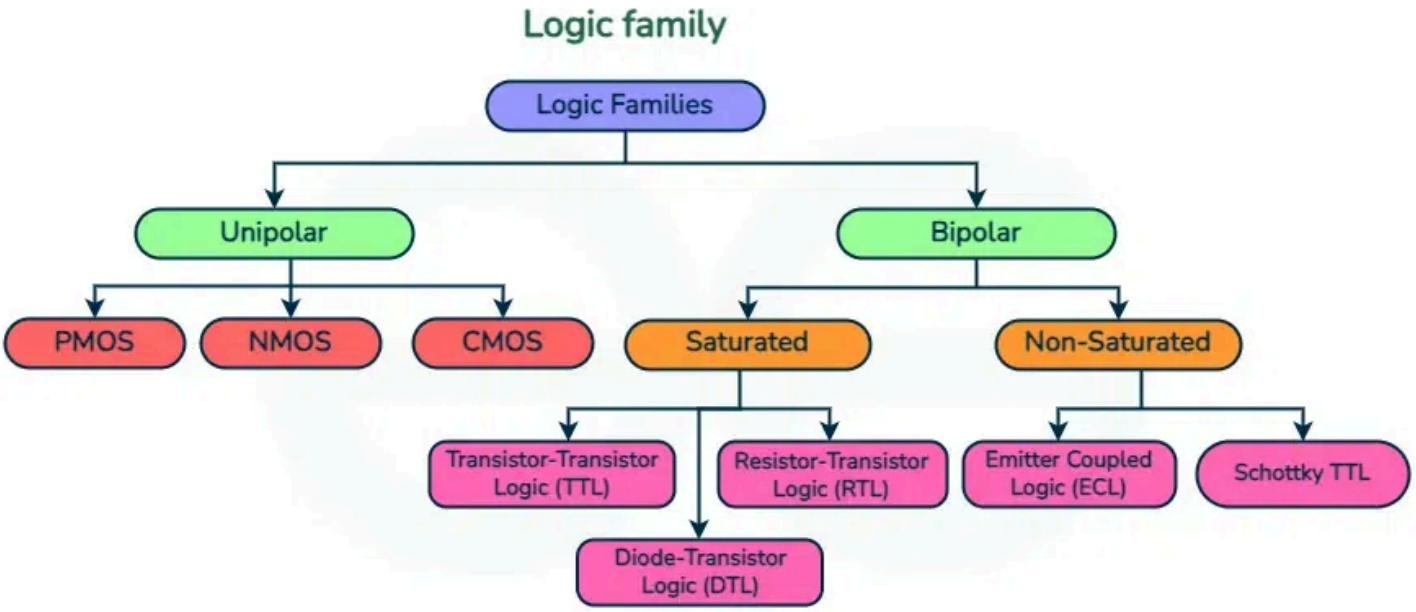
---

2.1 Logic Families

Logic families refer to groups of electronic logic gates that are constructed using similar techniques and have compatible electrical characteristics. These families play a crucial role in digital circuits and systems, as they determine the speed, power consumption, and voltage levels of the circuits. OR

Logic families are different types of technologies being used to build different logic gates. Logic gates are digital circuits that perform basic logic operations like AND, OR, NOT, NAND, and NOR. In other words, it is a group of compatible ICs with the same logic levels and supply voltages fabricated for performing various logical functions. Here, when we say that ICs have the same logic level, we are referring to two types of logic levels that exist.

- In positive logic, 0 is formed by a low voltage level, and a high voltage level forms 1. It means the ON state refers to high voltage as input or output while OFF means low voltage as input or output.
- In negative logic, 0 is formed by a high voltage level, and 1 is formed by a low voltage level. Here, the situation is reversed to that of positive logic. ON means a low voltage input or output while OFF means high voltage as input or output.



Some of the most common logic families detailed Characteristics:

- **TTL (Transistor-Transistor Logic)**

Speed: Moderate to high.

Power Consumption: Moderate to high.

Voltage Levels: Typically 5V.

Applications: General-purpose digital circuits, computers, and early microprocessors.

- **CMOS (Complementary Metal-Oxide-Semiconductor)**

Speed: Moderate to high (modern CMOS can be very fast).

Power Consumption: Very low (especially in static conditions).

Voltage Levels: Wide range (typically 3.3V, 5V, or lower in modern systems).

Applications: Microprocessors, microcontrollers, memory devices, and most modern digital circuits.

- **ECL (Emitter-Coupled Logic)**

Speed: Very high.

Power Consumption: Very high.

Voltage Levels: Typically -5.2V.

Applications: High-speed computing and communication systems.

Characteristics (Parameters) of Logic Families

Even though there are various logic families, the general characteristics, their definitions, nomenclature and terminologies used for all of them have been standardized.

1. Voltage and Current Parameters:

Voltage Parameters (Threshold Levels):

Ideally, the input voltage levels of 0V and +5V (for TTL) are called logic 0 and 1 levels respectively.

However, practically we will not always obtain voltage levels matching exactly to these values. Therefore it is necessary to define the worst-case input voltages.

$V_{IL(max)}$ – worst case low level input voltage:

This is the maximum value of input voltage which will be considered as a logic 0 level. If the input voltage is higher than VIL(max), then it will not be treated as a low (0) input level.

$V_{IH(min)}$ – worst case high level input voltage:

This is the minimum value of input voltage which will be considered as a logic 1 level. If the input voltage is lower than VIH(min), then it will not be treated as a high (1) input level.

$V_{OH(min)}$ – worst case high level output voltage:

This is the minimum value of output voltage which will be considered as a logic 1 level. If the output voltage is lower than VOH(min), then it will not be treated as a high (1) output level.

$V_{OL(max)}$ – worst-case low-level output voltage:

This is the maximum value of output voltage which will be considered as a logic 0 level. If the input voltage is higher than VOL(max), then it will not be treated as a low (0) output level.

Current Parameters:

IIL – Low-level input Current:

It is the current that flows into the input when a low level input voltage in the specified range is applied.

IIH – High-level input Current:

It is the current that flows into the input when a high level input voltage in the specified range is applied.

IOL – Low-level output Current:

It is the current that flows from the output when the output voltage is in the specified low voltage range and specified load is applied.

IOH – High-level Output Current:

It is the current that flows from the output when the output voltage is in the specified high voltage range and specified load is applied.

Fan-in and Fan-out:

Fan-in :

Fan in is defined as number of inputs a gate a has. For example, a two input gate will have a fan-in is equal to 2

Fan-out:

Fan-out is defined as the maximum number of output

Higher fan-out, higher the current supplying capacity of a gate. For example a fan-out of 5 indicates that the gate can drive (supply current to) at the most 5 inputs of the same IC family.

Noise Margin :

To understand the meaning of the term "Noise Margin" or "Noise Immunity", refer to the input and output voltage profiles

Noise is unwanted electrical disturbances that may induce some voltage in the connecting wires used between two gates or from a gate output to load.

Noise immunity is defined as the ability of a logic circuit to tolerate the noise without causing any unwanted changes in the output.

Propagation Delay(Speed of operation) :

The output of logic does not change its state instantaneously in response to the change in the state of the input.

There is a time delay between those two events, which is called the propagation delay.

Thus propagation delay is defined as the time delay between the instant of application of an input pulse and the instant of occurrence of the corresponding output pulse.

Power Dissipation :

As a result of applied voltage and currents flowing through the logic families (ICs), some power will be dissipated in it in the form of heat.

The power is in milliwatts.

Care should be taken to reduce the power dissipation taking place in the logic ICs in order to protect the ICS against damage due to excessive heat, to reduce the loading on power supplies, etc

Operating Temperature:

The temperature range acceptable for the consumer and industrial applications is 0^o and 70^o C and that for the military applications is -55^o to 125^o C.

The performance of gates will be in the specified limits over these temperature ranges.

Figure of Merit (Speed Power Product SPP) :

The figure of merit of a logical family is the product of power dissipation and propagation delay.

It is called the speed-power product. The speed is specified in seconds and power is specified in watts

Figure if Merit = Propagation delay time x Power dissipation

Practically, the value of the figure of merit should be as low as possible.

Comparison of TTL, CMOS and ECL logic family

Parameters	TTL	CMOS	ECL	RTL
Basic element	Transistors, diodes, and resistors	MOSFETs	Resistor and transistors	resistor and transistors
Fan out	moderate	highest(~50)	high	low
Propagation Delay	10ns	70ns	2ns	12ns
Noise margin	moderate	high	low	poor
Power dissipation	10mW	0.1mW	40-50mW	30mW
Circuit complexity	complex	moderately complex	complex	simple
Basic gate	NAND gate	NAND/NOR	OR/NOR	NOR gate
Application	Oscilloscopes, measurement devices	battery-powered circuits due to low power consumption, mobile equipments	high-speed switching application	practically obsolete due to poor noise margin

2.2 Logic Gates: Symbol, Truth table

- Logic gates are the fundamental components of all digital circuits and systems.
- In digital electronics, there are seven main types of logic gates used to perform various logical operations.
- A logic gate is basically an electronic circuit designed by using components like diodes, transistors, resistors, capacitors, etc.,
- As the name implies, a logic gate is designed to perform logical operations in digital systems like computers, communication systems, etc

Types of logic gate are

There are 7 types of gates

1. Basic Logic Gates

There are three basic logic gates:

AND Gate

OR Gate

NOT Gate

2. Universal Logic Gates

In digital electronics, the following two logic gates are considered as universal logic gates:

NOR Gate

NAND Gate

3. Derived Logic Gates

The following two are the derived logic gates used in digital systems:

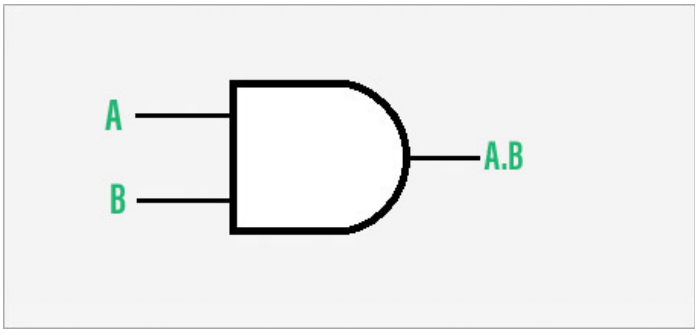
XOR Gate

XNOR Gate

2.2.1 Basic Logic Gates

1. AND Gate

- In digital electronics, the AND gate is one of the basic logic gate that performs the logical multiplication of inputs applied to it.
- Symbol:



Z=A.B

Where, A and B are inputs to the AND gate, while Z denotes the output of the AND gate.

- Truth Table:

Input		Output
A	**B**	**A AND B**
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate

- The primary function of the OR gate is to perform the logical sum operation.
- The boolean expression for a two input OR gate is given by,

$$Z = A + B$$

- Here, A, B, and C are inputs and Z is the output variables. We can extend this boolean expression to any number of input variables.
- Symbol:

- Truth Table:

Input		Output
A	**B**	**A OR B**
0	0	0
0	1	1
1	0	1
1	1	1

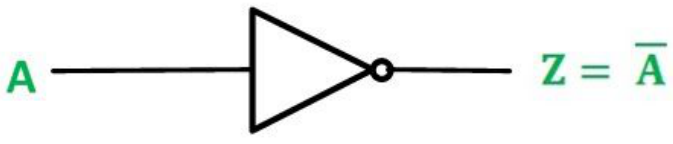
3. NOT gate

- In digital electronics, the NOT gate is another basic logic gate used to perform compliment of an input signal applied to it.
- It takes only one input and one output. The output of the NOT gate is complement of the input applied to it
- The NOT gate is also known as inverter, as it performs the inversion operation.

$$Z = \overline{A}$$

The bar over the input variable A represents the inversion operation.

- Symbol:



- Truth Table:

Input	Output
A	**NOT A**
0	1
1	0

4. NOR Gate

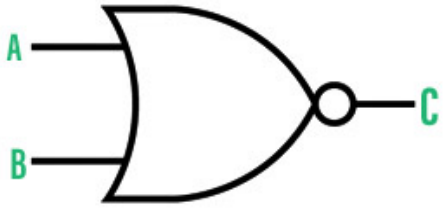
- The NOR gate is a type of universal logic gate that can take two or more inputs but one output.
- It is basically a combination of two basic logic gates i.e., OR gate and NOT gate. Thus, it can be expressed as,

NOR Gate = OR Gate + NOT Gate

- In other words, a NOR gate is an OR gate followed by a NOT gate
- The boolean expression of a two input NOR gate is given below:

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

- Symbol:



- Truth Table:

Input		Output
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

5. NAND Gate

- In digital electronics, the NAND gate is another type of universal logic gate used to perform logical operations.
- The NAND gate performs the inverted operation of the AND gate. Similar to NOR gate, the NAND gate can also have two or more input lines but only one output line.
- The NAND gate is also represented as a combination of two basic logic gates namely, AND gate and NOT gate. Hence, it can be expressed as

NAND Gate = AND Gate + NOT Gate

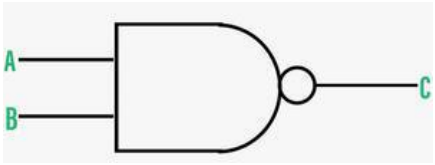
- Here is the boolean expression of a two input NAND gate.

$$C = \overline{AB}$$

In this expression, A and B are the input variables and C is the output variable.

We can extend this relation to any number of input variables like three, four, or more.

- Symbol:



- Truth Table:

Input		Output
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

6. XOR gate

- In digital electronics, there is a specially designed logic gate named, XOR gate, which is used in digital circuits to perform modulo sum.
- It is also referred to as Exclusive OR gate or Ex-OR gate.
- The XOR gate can take only two inputs at a time and give an output.
- The output of the XOR gate is high or logic 1 only when its two inputs are dissimilar.

The operation of the XOR gate can be described through a mathematical equation called its boolean expression. The following is the boolean expression for the output of the XOR gate**.**

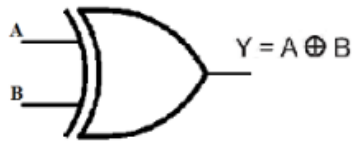
$$C = A \oplus B$$

Here, C is the output variable, and A and B are the input variables.

This expression can also be written as follows:

$$C = A\overline{B} + \overline{A}\,\mathbf{B}$$

- Symbol:



- Truth Table:

Input		Output
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

7. XNOR Gate

- The XNOR gate is another type of special purpose logic gate used to implement exclusive operation in digital circuits.
- It is used to implement the Exclusive NOR operation in digital circuits.
- It is also called the Ex-NOR or Exclusive NOR gate.
- It is a combination of two logic gates namely, XOR gate and NOT gate. Thus, it can be expressed as,

XNOR Gate = XOR Gate + NOT Gate

The output of an XNOR gate is high or logic 1 when its both inputs are similar. Otherwise the output is low or logic 0.

Here is the boolean expression of the XNOR gate.

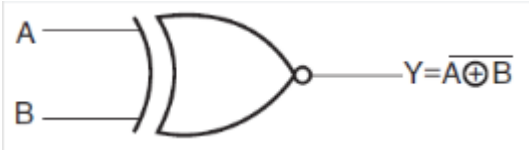
$$C = \overline{A \oplus B}$$

We can also write this expression as follows:

$$C = AB + \overline{AB}$$

Here, the A and B are inputs and Y is the output.

- Symbol:



- Truth Table:

Input		Output
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

Applications of Logic Gates

- Computers
- Microprocessors
- Microcontrollers
- Digital and smart watches

- Smartphones, etc.

2.3 Boolean algebra: Laws of Boolean algebra

- Boolean algebra is the category of algebra
- It is used to analyze and simplify digital circuits or digital gates.
- It is also called Binary Algebra or logical Algebra.
- It has been fundamental in the development of digital electronics and is provided for in all modern programming languages.

Laws of Boolean Algebra

There are six types of Boolean algebra laws. They are:

- Commutative law
- Associative law
- Distributive law
- AND law
- OR law
- Inversion law

1. Commutative law

- Any binary operation which satisfies the following expression is referred to as a commutative operation.
- Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

$$A. B = B. A$$

$$A + B = B + A$$

2. Associative Law

- It states that the order in which the logic operations are performed is irrelevant as their effect is the same.

$$(A. B). C = A . (B . C)$$

$$(A + B) + C = A + (B + C)$$

3. Distributive Law

- Distributive law states the following conditions:

$$\mathbf{A.\ (\ B + C) = (A.\ B) + (A.\ C)}$$

$$\mathbf{A + (B.\ C) = (A + B)\ .\ (\ A + C)}$$

4. AND Law

- These laws use the AND operation. Therefore they are called AND laws.

$$\mathbf{A\ .0 = 0}$$

$$\mathbf{A\ .\ 1 = A}$$

$$\mathbf{A.\ A = A}$$

$$A.\overline{A} = 0$$

5. OR Law

- These laws use the OR operation. Therefore they are called OR laws.

$$\mathbf{A\ + 0 = A}$$

$$\mathbf{A + 1 = 1}$$

$$\mathbf{A + A = A}$$

$$A + \overline{A} = 1$$

6. Inversion Law

- In Boolean algebra, the inversion law states that double inversion of variable results in the original variable itself.

$$\bar{\bar{A}} = A$$

De Morgan's First Law:

De Morgan's First Law states that

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

The first law states that the complement of the product of the variables is equal to the sum of their individual complements of a variable.

The truth table that shows the verification of De Morgan's First law is given as follows:

A	**B**	$\overline{A}$	$\overline{B}$	$\overline{(AB)}$	$\overline{(A} + \overline{B)}$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

The last two columns show that $\overline{(AB)} = \overline{(A} + \overline{B)}$

.

Hence, De Morgan's First Law is proved.

De Morgan's Second law:

De Morgan's Second law states that $\overline{(A + B)} = \overline{(A} . \overline{B)}$

The second law states that the complement of the sum of variables is equal to the product of their individual complements of a variable.

The following truth table shows the proof for De Morgan's second law.

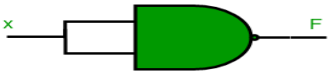
A	B	$\overline{A}$	$\overline{B}$	$\overline{(A + B)}$	$(\overline{A} . \overline{B})$
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

The last two columns show that $\overline{(A + B)} = (\overline{A} . \overline{B})$

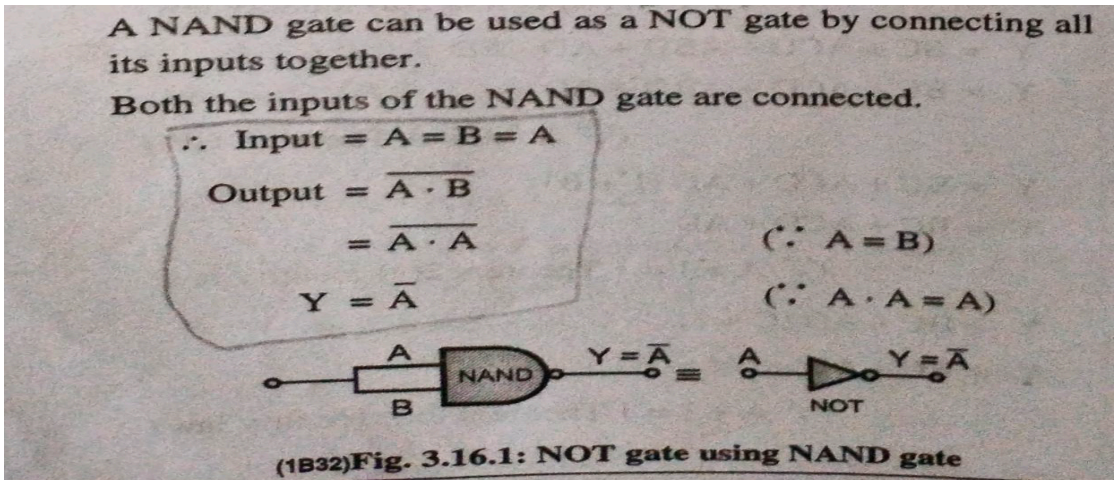
Hence, De Morgan's second law is proved.

2.5 Implementation of Basic Gate using Universal gates.

2.5.1 Implementation of NOT Gate using NAND gate:



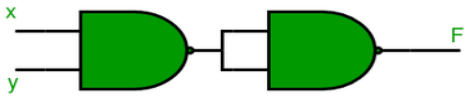
x	F
0	1
1	0



(1B32)Fig. 3.16.1: NOT gate using NAND gate

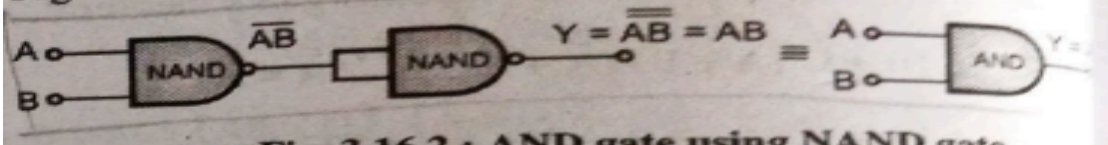
2.5.2 Implementation of AND Gate using NAND gate:

The AND gate can be implemented by using two NAND gates in the below fashion:



x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 3.16.2 shows AND gate using NAND.



(1B33)**Fig. 3.16.2 : AND gate using NAND gate**

- To construct an AND gate from NAND gates, we ne an inverter or NOT gate at the output of NAND gate
- The double inversion cancels out and final resul AND function as shown in Fig. 3.16.2.
- The Boolean expression for AND gate is,

$$Y = AB$$

The output expression of Fig. 3.16.2

$$Y = \overline{\overline{AB}} = AB \quad (\because \overline{\overline{A}} = A \text{ Theorem 3 Involut}$$

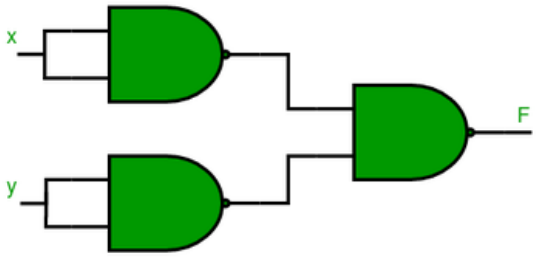
- Thus, from output we can see that AND functio realized by NAND gates.

☞ **Truth table**

Table 3.16.1

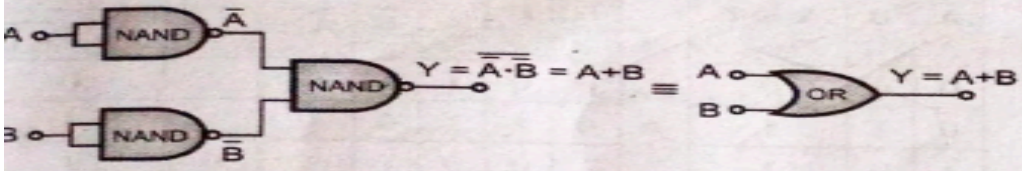
A	B	$\overline{A \cdot B}$	$Y = \overline{\overline{A \cdot B}}$	AB
0	0	1	0	0
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

2.5.3 Implementation of OR Gate using NAND gate:



x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 3.16.3 shows the construction of an OR gate using three NAND gates.



(1B34)**Fig. 3.16.3: OR gate using NAND gates**

- The first two NAND gates invert A and B inputs to produce outputs $\bar{A}$ and $\bar{B}$.
- The outputs are applied to inputs another NAND gate to obtain the OR function. Thus, by using three NAND gates we can realize an OR gate.

Boolean expression for the OR gate is

$$Y = A + B$$

Boolean expression for Fig. 3.16.3,

$$Y = \overline{\bar{A} \cdot \bar{B}}$$

$$Y = \bar{\bar{A}} + \bar{\bar{B}} \quad (\because \overline{AB} = \bar{A} + \bar{B} \text{ DeMorgan's law})$$

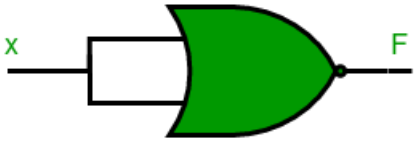
$$Y = A + B \quad (\because \bar{\bar{A}} = A \text{ Involution})$$

Truth table

Table 3.16.2 : Truth table for OR realization using NAND gates

A	B	$\bar{A}$	$\bar{B}$	$\bar{A}\cdot\bar{B}$	$\overline{\bar{A}\cdot\bar{B}}$		A	B	A + B
0	0	1	1	1	0		0	0	0
0	1	1	0	0	1	≡	0	1	1
1	0	0	1	0	1		1	0	1
1	1	0	0	0	1		1	1	1

2.5.4 Implementation of NOT Gate using NOR gate:



x	F
0	1
1	0

Fig. 3.17.1 shows NOT gate using NOR gate. A NOR gate can be used as a NOT gate by connecting all its inputs together.

Both the inputs of the NOR gate are connected.

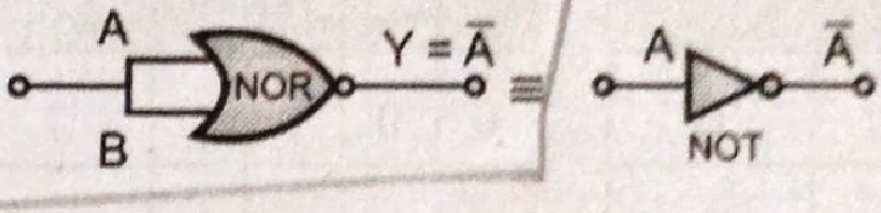
$\therefore$ Input A = B = A

Output $= \overline{A + B}$

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

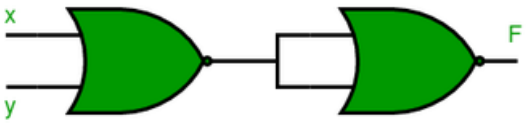
$Y = \bar{A}$ ($\because$ A + A = A Theorem 1(a))

This is the expression for NOT gate. Fig. 3.17.1 shows NOT gate using NOR gate.



(1B40) **Fig. 3.17.1: NOT gate using NOR gate**

2.5.5 Implementation of OR Gate using NOR gate:



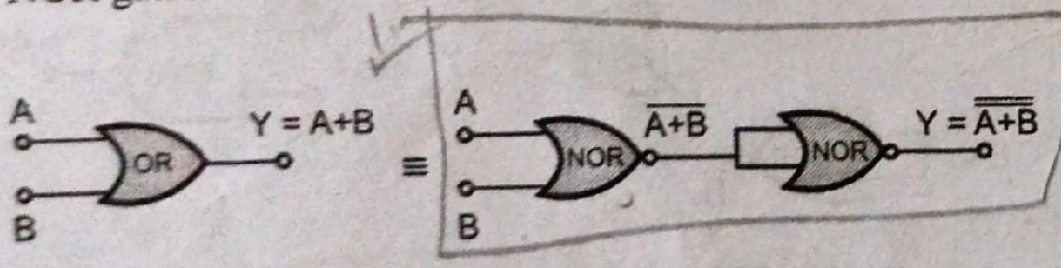
x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

☑ **Ans. :**

The Boolean expression for OR gate is,

$$Y = A + B$$

Fig. 3.17.3 shows the implementation of OR gate using NOR gates.



(1B42)**Fig. 3.17.3 : Implementation of OR gate using NOR gates**

The Boolean expression for circuit shown in Fig. 3.17.3 is

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

$$Y = A + B \qquad (\because \overline{\overline{A}} = A \text{ Involution law})$$

is the equation of OR gate.

☞ **Truth tables**

Truth table for OR gate

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

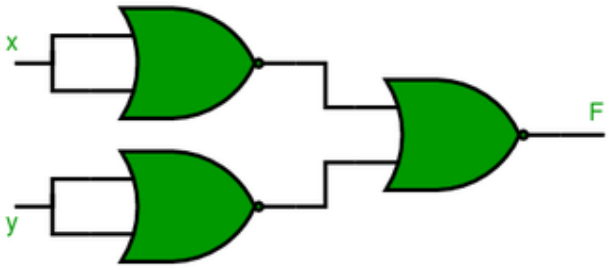
≡

Truth table for OR gate using NOR gates

A	B	$\overline{A+B}$	$\overline{\overline{A+B}}$
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

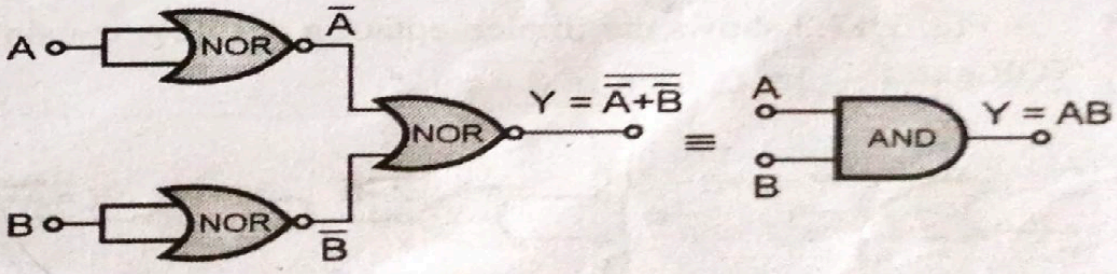
Thus, we can not realize OR gate using NOR gates.

2.5.6 Implementation of AND Gate using NOR gate:



x	y	F
0	0	0
1	0	0
1	1	1
0	1	0

Fig. 3.17.2 shows implementation of AND gate using NOR gates.



(1B41)**Fig. 3.17.2 : AND neither gate using NOR gate**

The Boolean expression for circuit shown in Fig. 3.17.2 is,

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

$$Y = \overline{\overline{AB}} \qquad (\because \bar{A} + \bar{B} = \overline{AB} \text{ DeMorgan's Theorem})$$

$$\mathbf{Y = AB} \qquad (\because \bar{\bar{A}} = 1 \text{ Involution law Theorem 3})$$

is the equation of AND gate.

☞ **Truth tables**

Truth table for AND gate

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

≡

Truth table for AND gate using NOR gates

A	B	$\bar{A}$	$\bar{B}$	$\bar{A}+\bar{B}$	$\overline{\bar{A}+\bar{B}}$
0	0	1	1	1	0
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	0	1

Thus, we can realize AND gate using NOR gates.

Example of Boolean Algebra:

Ex. 4.15.1 : Use De-Morgan's theorem to simplify the following Boolean expression.

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

☑ **Soln. :**

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

$$Y = \overline{\bar{A}B} \cdot \overline{A\bar{B}}$$

[De- Morgan's law : $\overline{(A + B)} = \bar{A} \cdot \bar{B}$]

$$Y = (\bar{\bar{A}} + \bar{B}) \cdot (\bar{A} + \bar{\bar{B}})$$

$$Y = (A + \bar{B}) \cdot (\bar{A} + B)$$ [Inversion law : $\bar{\bar{A}} = A$]

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

[AND law : $A \cdot \bar{A} = 0$, $B \cdot \bar{B} = 0$]

[Distributive law : $\bar{A} + AB = \bar{A} + B$]

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

Ex. 4.15.2 : Verify the following Boolean function

$$AB + \bar{A}C + BC = AB + \bar{A}C$$

☑ **Soln. :**

$$\text{L.H.S.} = AB + \bar{A}C + BC$$

$$= AB + \bar{A}C + BC \cdot 1$$

$$= AB + \bar{A}C + BC(A + \bar{A})$$

[OR law : $A + \bar{A} = 1$]

$$= AB + \bar{A}C + ABC + \bar{A}BC$$

$$= AB(1 + C) + \bar{A}C(1 + B)$$

[OR law : $(1 + C) = (1 + B) = 1$]

$$= AB + \bar{A}C = \text{R.H.S.}$$

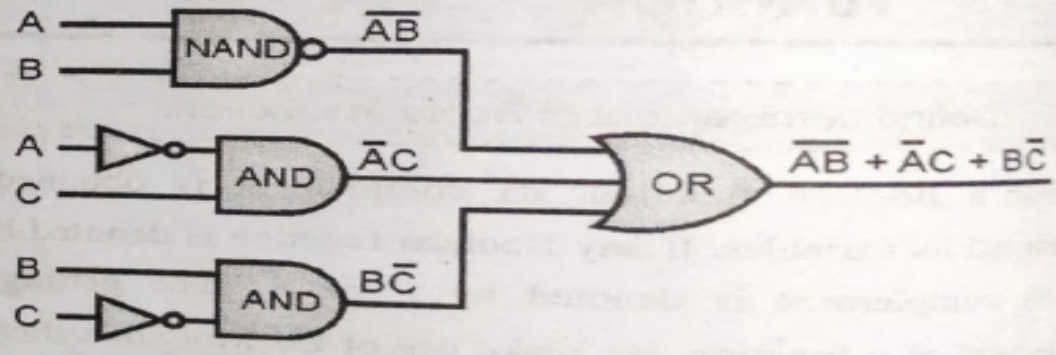
Hence Proved

Implement the logical expression using gates,

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

☑ **Soln. :**

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



UEx. 4.15.8 (MSBTE - Q. 2(b), S-15, 4 Marks)

Simplify the following equation using Boolean laws and realize it using basic gates only. $Y = ABC + A\bar{B}\,C + \bar{A}\,BC + \bar{A}\,\bar{B}\,C$.

☑ **Soln. :**

$$Y = ABC + A\bar{B}C + \bar{A}\,BC + \bar{A}\,\bar{B}\,C$$

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

$$Y = BC + \bar{B}C \quad [\text{OR law} : A + \bar{A} = 1]$$

$$Y = C\,(B + \bar{B}) \quad [\text{OR law} : B + \bar{B} = 1]$$

$$Y = C$$

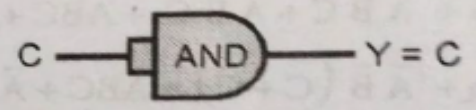


Fig. Ex. 4.15.8

UEx. 4.15.9 (MSBTE - Q. 3(a), S-15, 4 Marks)

Simplify using DeMorgans theorem and realize it using basic gates.

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

☑ **Soln. :**

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

$$Y = \overline{(A\bar{B} + \bar{A}\bar{B})} + \overline{(AB + \bar{A}B)}$$

[DeMorgan's law : $\overline{AB} = \bar{A} + \bar{B}$]

$$Y = \left(\overline{A\bar{B}}\right) \cdot \left(\overline{\bar{A}\,\bar{B}}\right) + \left(\overline{AB}\right) + \left(\overline{\bar{A}\,B}\right)$$

[DeMorgan's law : $\overline{A + B} = \bar{A} \cdot \bar{B}$]

$$Y = \left(\bar{A} + \bar{\bar{B}}\right) \cdot \left(\bar{\bar{A}} + \bar{\bar{B}}\right) + \left(\bar{A} + \bar{B}\right)\left(\bar{\bar{A}} + \bar{B}\right)$$

[DeMorgan's law : $\overline{AB} = \bar{A} + \bar{B}$]

$$Y = \left(\bar{A} + B\right) \cdot (A + B) + \left(\bar{A} + \bar{B}\right) \cdot \left(A + \bar{B}\right)$$

[Involution law : $\bar{\bar{A}} = A, \bar{\bar{B}} = B$]

Fig. Ex

UEx. 4.15.10 (MSBTE - Q. 5(c), S-15, 4 Marks)

Simplify the following and realize it.

$$Y = A + \bar{A}\,\bar{B}\,C + \bar{A}\,\bar{B}\,\bar{C} + ABC + \bar{A}\,\bar{B}.$$

☑ **Soln. :**

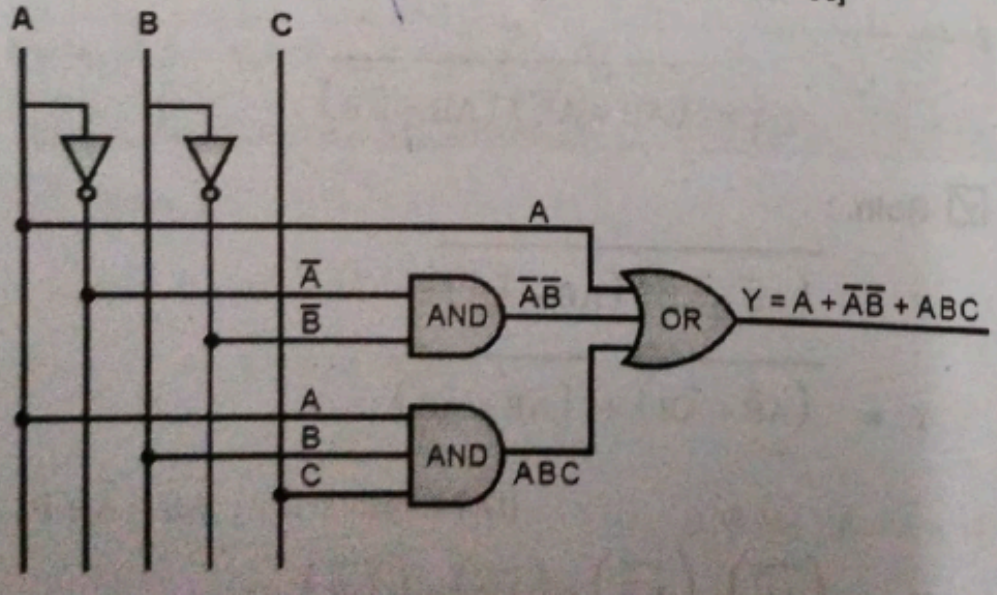
$$Y = A + \bar{A}\,\bar{B}\,C + \bar{A}\,\bar{B}\,\bar{C} + ABC + \bar{A}\,\bar{B}$$

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

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

[OR law : $C + \bar{C} = 1$]

$$Y = A + \bar{A}\,\bar{B} + ABC$$ [OR law : $A + A = A$]



UEx. 4.15.19 (MSBTE - Q. 4(b), W-17, 4 Marks)

Simplify following equation using boolean algebra and draw circuit diagram :

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

☑ **Soln. :**

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

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

$$= 0 + \bar{A}\bar{B} + \bar{A}\bar{B} + 0$$

[AND law : $\bar{A}A = \bar{B}B = 0$]

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

$$= \bar{A}\bar{B}$$ [OR law : $A + A = A$]