Number Systems, Codes & Binary Arithmetic

Number Systems :

The radix or base of a number system is the number of digits or basic symbols used in that system. Commonly used number systems are:

Type	Radix or Base	Symbols used	Example
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	15
Binary	2	0, 1	1001
Octal	8	0, 1, 2, 3, 4, 5, 6, 7	15
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B, C, D, E, F	A2F

Binary Number System : -

The number system with base two is called as the binary number system. Only 0 & 1 are the symbols used in this system. 0 & 1are called as binary digits or bits. The left most bit is called the Most Significant Bit (MSB) The right most bit is called the least significant Bit (LSB).

A group of character (either digits or letters) write one after another is called string. In binary number system. Group of 4 bits called Nibble & a group of 8 bits is called a Byte.

In following table Binary number with their equivalent decimal number is given.

Binary Number	Decimal Number	Binary Number	Decimal Number
0	0	101	5
1	1	110	6
01	2	111	7
11	3	1000	8
100	4	1001	9

1) Binary to Decimal conversion : (Stream lined Method)

Any Binary number can be converted in to it's equivalent decimal number using weights assigned to each bit position as using following steps.

a) **Write binary number**
b) **Write 1, 2, 4, 8, 16…. (2^0,2^1,2^2,2^3,2^4…) directly under binary bits from right to left from Binary Point**
c) **When zero appears in binary, do not count its value.**
d) **Add the remaining number to get decimal equivalent**

Ex: $10010_{(2)}$

1	0	0	1	0
2^4	2^3	2^2	2^1	2^0

$16 + 2 = 18_{(10)}$

For Fractions :

a) **Write the binary number after binary point**
b) **Write ½, ¼, 1/8, 1/16…. (2^{-1},2^{-2}, 2^{-3},2^{-4}….) directly under binary bits from left to right after binary point**
c) **When zero appears in binary do not count it's value**
d) **Add the remaining number to get decimal equivalent**

Ex : $0.1101_{(2)}$

1	1	0	1
2^{-1}	2^{-2}	2^{-3}	2^{-4}

$0.1101_{(2)} = 0.8125_{(10)}$

EX : $1101.0010_{(2)}$ convert to decimal .

Sol : Consider integer part of given number

1101 =	1	1	0	1
	2^3	2^2	2^1	2^0
	8	4	0	1

$8+4+0+1 = 13_{(10)}$

Now, consider Fractional Part,

.0010	**=**	**0**	**0**	**1**	**0**
		2^{-1}	2^{-2}	2^{-3}	2^{-4}
		0	**0**	**0.125**	**0**

$0+0+0.125+0 = 0.125_{(10)}$

Therefore ,

$1101.0010_{(2)} = 13.125_{(10)}$

2) <u>Binary to Decimal conversion</u> :

Decimal to binary conversion (Double Dabble method)

This method is a way of converting any decimal number to its binary equivalent. In this method for integers divide successively by 2& write down each quotient & its remainder till we get quotient as zero. Lastly remainder write in reverse order to get required binary number.

Ex: Convert $25_{(10)}$ to Binary

Here, Base is 2 therefore divide 25 by2 .
Now write number in R column in reverse order to get equivalent binary number.
Now, consider R column in Reverse order as shown $11001_{(2)}$ is the equivalent binary number. For $25_{(10)}$

D	Q	R
25 ÷ 2	12	1
12 ÷ 2	6	0
6 ÷ 2	3	0
3 ÷ 2	1	1
1 ÷ 2	0	1

<u>Fractions</u> : For fractional number , multiply given decimal number by 2 & then write down products & record the carry in the integer position. The carries written in forward order gives equivalent binary fractional number. Process of multiplication by 2 is continuous till we get necessary accuracy.

Ex. : $0.55_{(10)}$ convert in to binary.

Write binary carry in the direction as shown by arrow.
We get required binary fractional number.

$0.55_{(10)} = 0.10001_{(2)}$

M	Product Carry
0.55 * 2	1.10 1
0.10 * 2	0.20 0
0.20 * 2	0.40 0
0.40 * 2	0.80 0
0.80 * 2	1.60 1

<u>Hexadecimal Number System :</u>

In this system a base is 16. This system uses a set of a 16 unique symbols 0 - - - - 9 decimal digits having same significance in decimal number system & six symbols A, B, C, D, E & F which represents decimal 10, 11, 12, 13, 14 & 15 respectively.

Ex : $A5_{(16)}$, $10_{(16)}$

<u>Hexadecimal to Decimal conversion:</u> By using positional value method i.e. multiply each hexadecimal digit by its weight & add the resulting products.

Ex : $A5_{(16)}$ convert to equivalent decimal.

$$A5_{(16)} = A * 16^1 + 5 * 16^0$$
$$= 10 * 16 + 5 * 1$$
$$= 160 + 5$$
$$= 165_{(10)}$$

For fractions :

Ex : $0.B2_{(16)}$ convert to equivalent decimal.

$$0.B2_{(16)} = B * 16^{-1} + 2 * 16^{-2}$$
$$= 11/16 + 2/256$$
$$= 0.0625 * 11 + 2 * 0.00390625$$
$$= 0.6875 + 0.0078125$$
$$0.B2_{(16)} = 0.6953125_{(10)}$$

Decimal to Hexadecimal conversion :

In this conversion divide the decimal number by 16, write quotient & remainder again, quotient divided by 16 this process is continued till remainder is zero. Then write remainder in reverse order to get hexadecimal number.

Ex :

$125_{(10)}$ convert to equivalent hexadecimal.

	D	Q	R
Here, Base is 16 therefore divide 125 by 16.			
Now write number in R column in reverse	**125 ÷ 16**	**7**	**13**
Order to get equivalent Hexadecimal number.	**7 ÷ 16**	**0**	**7**

Now, consider R column in Reverse order

as shown $7D_{(16)}$ is the equivalent hexadecimal number. For $125_{(10)}$

Binary to Hexadecimal conversion :

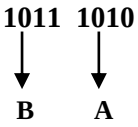
This conversion is done by combing the given binary digit / bits in a group of 4 & writing hexadecimal equivalent of each group. The grouping should start from LSB towards MSB in case of whole numbers & from binary point towards right in case of fraction numbers.

Ex : 10111010(2) convert to Hexadecimal equivalent.

i. **Write given binary number as 10111010**

ii. **Made group of 4 bits from LSB to MSB as 1011 1010**

iii. **For each group write hexadecimal equivalent**



iv. **$10111010_{(2)} = BA_{(16)}$**

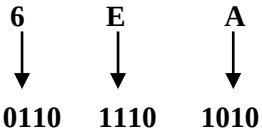
Hexadecimal to Binary conversion :

In this conversion each hexadecimal digit is converted into its 4 bit binary equivalent.

Ex. : $6EA_{(16)}$ convert in to binary equivalent.

i. **Write given Hexadecimal number as $6EA_{(16)}$**

ii. **For each Hexadecimal digit write 4 bit binary equivalent as**



iii. **Therefore , $6EA_{(16)} = 011011101010_{(2)}$**

LOGIC GATES

Introduction:

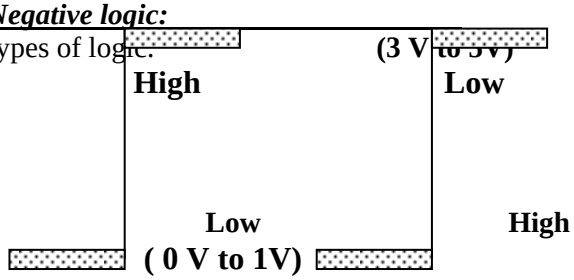
In our thinking & logic we differentiate the answer to 2 values only i.e. **true or false, right or wrong** Our thinking trying to find the answer to 2 valued questions. Logic next attracted mathematics. D-Morgan sets the relation ship between logic & mathematics. But, George Boolean invented new kind of algebra. He gives symbolic logic & this logic is known by name as Boolean Algebra.

Boolean Algebra contains only two valued functions i.e. true or false. It uses binary numbering system, which uses 0 & 1 only.

Logic-Positive Logic & Negative logic:

In digital system we use two types of logic.

1) Positive Logic
2) Negative Logic



When '1' or high state represents more positive and '0' or low state represents more negative then logic is positive logic. When the voltage range is in the range of 3.5 V to 5 V it will be considered high level or level '1'. 0 V to 1 V range will be consider as low level or level '0'.

i.e. In Positive logic high level represented by '1' & low level represented by '0'.

In negative logic '1' or high state represented more negative & '0' or low state represents more positive therefore, logic is negative logic.As shown in above logic level diagram, the voltage range of 3.5 V to 5 V it will be considered low level & represented by '0'. The voltage range 0 V to 1 V will be considered as high level & represented by '1'.

i.e. In Negative logic high level represented by '0' & low level represented by '1'.

- **Gate: -**

A Gate is logic circuit with one or more input signals & has only single output. The output of the logic gate depending on the combination of input signals. The total number of combinations is given by 2^n. Where n is the number of inputs.

There are two types of gates Basic gates OR, AND, NOT & derived gates NAND, NOR, XOR and XNOR.

- ***Basic Gates: -***

1) OR Gate: -

This gate is known as inclusive OR gate, which has 2 inputs A & B and Y is the single output.

Logic: "If A is true or B is true then Y is true."

Logic equation of the OR gate is Y = A + B

For 'n' number of inputs Y = A + B + C + D +…. + n

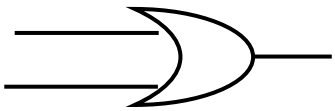
The OR operation is also called as Or addition. The rules of OR addition are

Y = A + B 1) 0 + 0 = 0 2) 0 + 1 = 1 3) 1 + 0 = 1 4) 1 + 1 = 1

So, truth table for OR gate is,

Truth Table **SYMBOL**

INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



Definition: "**OR** gate has high output if any one of the input is high"

2) AND Gate: -

AND gate has two or more input signals and only one output signal. A & B are the input signals and Y is the output.

Logic: "If A and B are true then Y is true"

Logic equation: For AND gate equation is $Y = A * B$

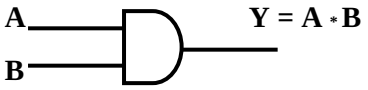
For 'n' number of inputs $Y = A * B * C * D *\ldots * n$

The AND operation is called as AND multiplication. The rules of AND multiplication are:

$Y = A * B$

1) $0 * 0 = 0$ 2) $0 * 1 = 0$ 3) $1 * 0 = 0$ 4) $1 * 1 = 1$

SYMBOL



Truth Table

INPUTS		OUTPUT
A	**B**	**Y**
0	0	0
0	1	0
1	0	0
1	1	1

Definition: "The gate which has high output if & only if all the inputs are high is called AND Gate."

3) NOT gate: A NOT gate is a gate with one input & one output. It is also called inverter because the Output State is always opposite the input state.

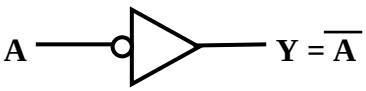
Logic: "If A is true then Y is false and if A is false then Y is true"

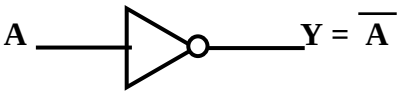
Logic equation: $Y = \overline{A}$

The rules of the inversion are:

1) $\overline{0} = 1$ 2) $\overline{1} = 0$

SYMBOL





❖ **Derived Gates :**

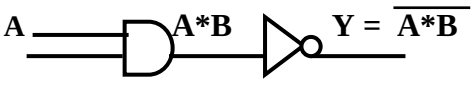
By using combination of two more basic gates we can construct the gates called derived gates. NAND, NOR & XOR are derived gates.

1) NAND Gate :

NAND gate means NOT – AND gate. It has two or more input but only one output. NAND gate can be constructed with AND gate followed by NOT gate. The complement output of AND gate forms NAND gate.

Definition :- If all the inputs of the gate are high the gate has low output then gate is called NAND gate.

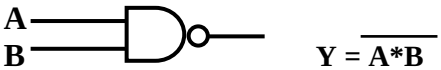
Logic meaning of NAND gate



Truth Table

INPUTS		OUTPUT
A	**B**	**Y**
0	0	1
0	1	1
1	0	1
1	1	0

Logic Symbol

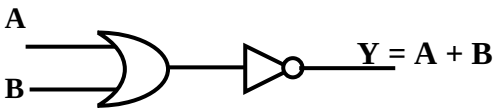


2) NOR Gate :-

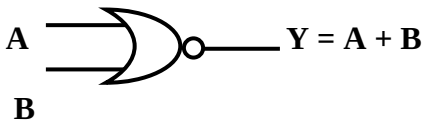
The NOR gate has one or more inputs & has only one output. NOR gate is the combination of NOT gate and OR gate. I.e. NOR gate can be constructed with OR gate followed by NOT gate. The

complimented result of OR gate forms the NOR gate. The words NOT – OR are contacted to the word NOR.
Logic equation of NOR gate is $Y = \overline{A + B}$

Logic meaning of NOR gate



Logic Symbol



Definition :- When any input of the gate is high output becomes low the gate is NOR gate.

Truth Table

INPUTS		OUTPUT
A	**B**	**Y**
0	0	1
0	1	0
1	0	0
1	1	0

****NAND & NOR Gates as Universal Building Blocks :**

OR, AND & NOT are the basic building blocks of the digital circuits. For combinational circuits we are use these three basic gates, but by using NAND gate or NOR gate we can construct any basic and derived gates so, NAND & NOR gates are called universal building blocks or universal gates.

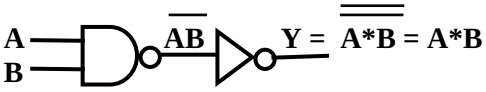
± NAND Gate as Universal Gate :-

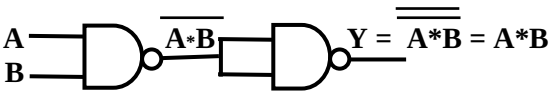
(A) NAND gate as NOT gate :-

By connecting all the inputs of NAND gate together we gate NOT gate.

(B) NAND gate as AND gate :-

NAND gate followed by NOT gate gives AND gate as shown :

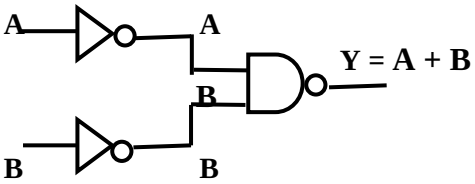


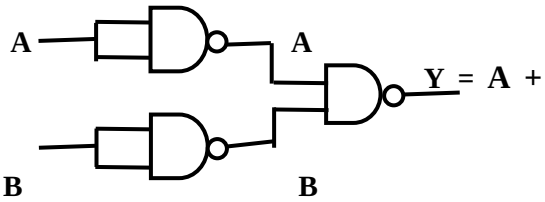


(C) NAND gate as OR gate :-

Boolean Equation for OR gate is $Y = A+B = \overline{\overline{A} + \overline{B}} = \overline{\overline{A} * \overline{B}}$

From above equation we construct the logic diagram as :

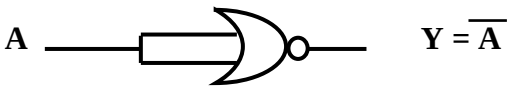




NOR Gate as Universal Gate :-

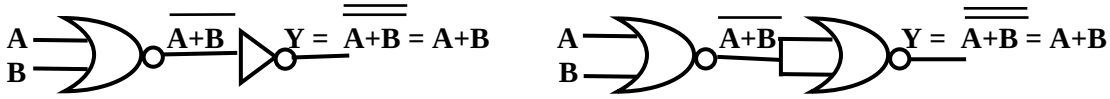
(A) NOR gate as NOT gate :-

By connecting all the inputs of NOR gate together we gate NOT gate.



(B) NOR gate as OR gate :-

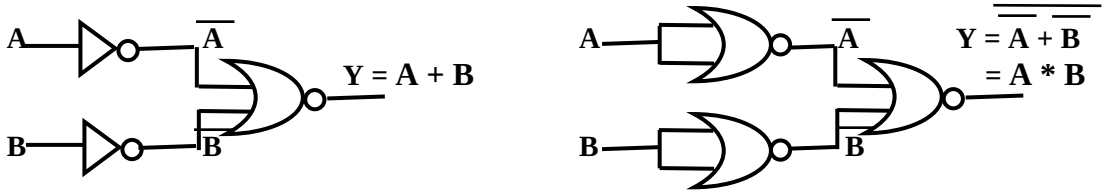
NOR gate followed by NOT gate gives OR gate as shown :



(C) NOR gate as AND gate :-

Boolean Equation for AND gate is $Y = A^* B = \overline{\overline{A} * \overline{B}} = \overline{\overline{A} + \overline{B}}$

From above equation we construct the logic diagram as :



Y Exclusive OR gate : - (XOR - gate):

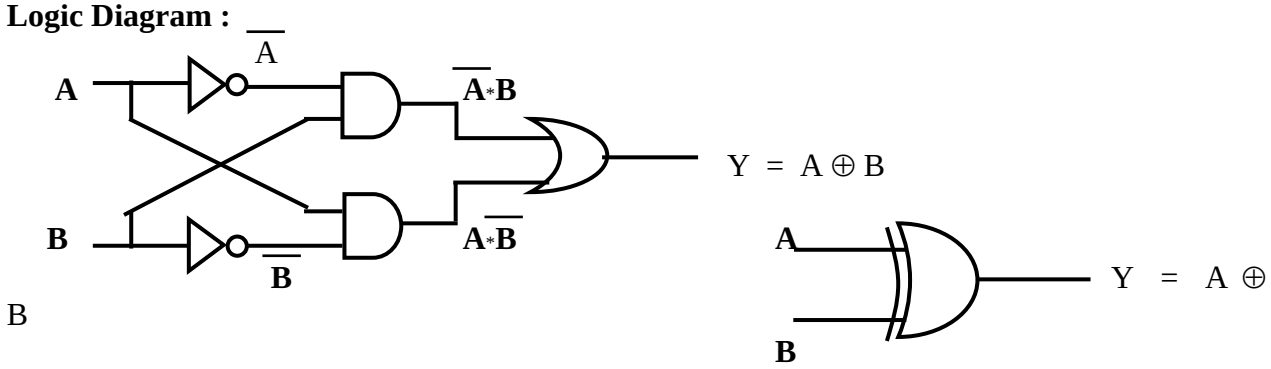
XOR gate is a special case of general OR gate X-OR gate has two or more inputs & one output. Two input XOR gate can be constructed as given in diagram. It requires 2 NOT gates, 2 AND gates, & an OR gate

The XOR gate is a logic circuit, which has a high output when it has odd number of inputs, are high.

Logic Equation of XOR gate :- $Y = \overline{A}*B + A*\overline{B}$

$Y = A \oplus B$

Logic Diagram :



Logic Symbol :

<u>Laws & Theorems of Boolean Algebra</u>:

- **Identities**

1) A + 0 = A 2) A * 1 = A 3) A + 1 = 1
4) A * 0 = 0 5) A + A = A 4) A * A = A
7) A + A' = 1 8) A * A' = 0 9) (A')' = A

- **Commutative Laws**

1) A + B = B + A 2) A* B = B*A

- **Associative Laws**

1) A*(B*C) = (A*B)*C 2) A+(B+C) = (A+B)+C

- **Distributive Laws**

1) A*(A+B) = A*B + A*C 2) A + AB = A 7) A(A+B) = A

3) (A + B)' = A' * B' (De-Morgan's first theorem)
4) (A * B)' = A' + B'(De-Morgan's second theorem)
5) A + A'B = A + B 6) AB + A*B' = A

DeMorgan's Theorems :

❖ **De-Morgan's first theorem:**

Statement : De-Morgan's first theorem states that complement of sum is equal to the product of their complements.

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

Proof :

To prove this theorem, we consider all possible values of A & B

I) Let A = 0, B = 0

Consider, L.H.S
$\therefore$ L.H.S $= \overline{(A + B)}$
$= \overline{(0 + 0)}$
$= \overline{1}$

Consider R.H.S.
R.H.S. $= \overline{A} * \overline{B}$
$= \overline{0} * \overline{0}$
$= \overline{1}$

II) Let A = 0, B = 1

Consider, L.H.S
$\therefore$ L.H.S $= \overline{(A + B)}$
$= \overline{(0 + 1)}$
$= \overline{0}$

Consider R.H.S.
R.H.S. $= \overline{A} * \overline{B}$
$= \overline{0} * \overline{1}$
$= \overline{0}$

III) Let A = 1, B = 0

Consider, L.H.S
$\therefore$ L.H.S $= \overline{(A + B)}$
$= \overline{(1 + 0)}$
$= \overline{0}$

Consider R.H.S.
R.H.S. $= \overline{A} * \overline{B}$
$= \overline{1} * \overline{0}$
$= \overline{0}$

IV) Let A = 1, B = 1

Consider, L.H.S
$\therefore$ L.H.S $= \overline{(A + B)}$
$= \overline{(1 + 1)}$
$= \overline{0}$

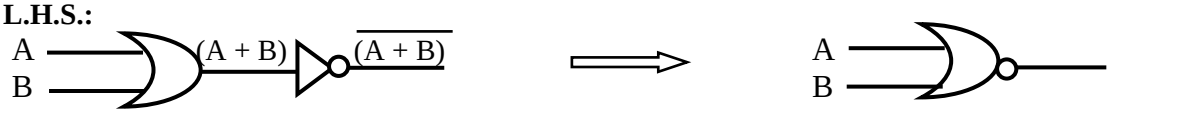
Consider R.H.S.
R.H.S. $= \overline{A} * \overline{B}$
$= \overline{1} * \overline{1}$
$= \overline{0}$

Thus, for all possible values of A & B **L.H.S. = R.H.S**

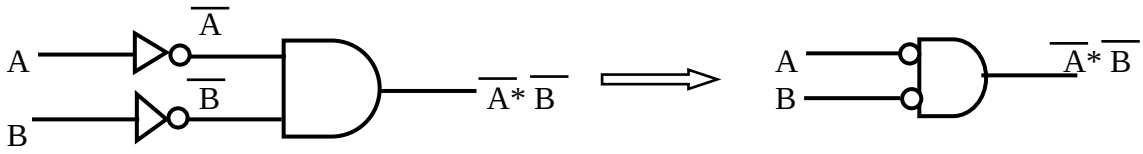
i.e. $\overline{(A + B)} = \overline{A} * \overline{B}$ Hence **DeMorgan's first Theorem is proved**.

Logic Diagram of DeMorgan's first Theorem:

± **L.H.S.:**



± **R.H.S.:**



❖ **De-Morgan's second theorem:**

Statement : De-Morgan's second theorem states that complement of product is equal to the sum of their complements.

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

Proof :

To prove this theorem, we consider all possible values of A & B

V) Let A = 0, B = 0

Consider, L.H.S

$\therefore$ L.H.S $= \overline{(A * B)}$
$= \overline{(0 * 0)}$
$= \overline{0}$
$= 1$

Consider R.H.S.

R.H.S. $= \overline{A} + \overline{B}$
$= \overline{0} + \overline{0}$
$= 1 + 1$
$= 1$

VI) Let A = 0, B = 1

Consider, L.H.S

$\therefore$ L.H.S $= \overline{(A * B)}$
$= \overline{(0 * 1)}$
$= \overline{0}$
$= 1$

Consider R.H.S.

R.H.S. $= \overline{A} + \overline{B}$
$= \overline{0} + \overline{1}$
$= 1 + 0$
$= 1$

VII) Let A = 1, B = 0

Consider, L.H.S

$\therefore$ L.H.S $= \overline{(A * B)}$
$= \overline{(1 * 0)}$
$= \overline{0}$
$= 1$

Consider R.H.S.

R.H.S. $= \overline{A} + \overline{B}$
$= \overline{1} + \overline{0}$
$= 0 + 1$
$= 1$

VIII) Let A = 1, B = 1

Consider, L.H.S

$\therefore$ L.H.S $= \overline{(A * B)}$
$= \overline{(1 * 1)}$
$= \overline{1}$
$= 0$

Consider R.H.S.

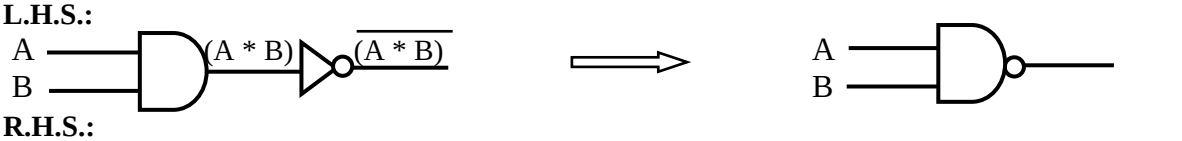
R.H.S. $= \overline{A} + \overline{B}$
$= \overline{1} + \overline{1}$
$= 0 + 0$
$= 0$

Thus, for all possible values of A & B **L.H.S. = R.H.S**

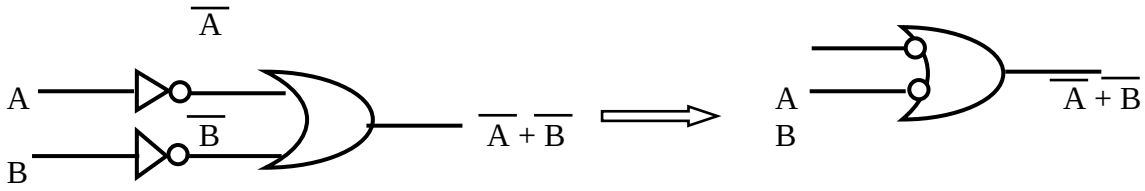
i.e. $\overline{(A * B)} = \overline{A} + \overline{B}$ Hence, **DeMorgan's second Theorem is proved**.

Logic Diagram of De Morgan's second Theorem:

± **L.H.S.:**



± **R.H.S.:**



❖ **Truth Table to prove De Morgan's Theorems.**

				First Theorem		Second Theorem	
A	**B**	$\overline{A}$	$\overline{B}$	$\overline{(A + B)}$ **(L.H.S)**	$\overline{A} * \overline{B}$ **(R.H.S)**	$\overline{(A * B)}$ **(L.H.S)**	$\overline{A} + \overline{B}$ **(R.H.S)**
0	0	1	1	1	1	1	1
0	1	1	0	1	1	0	0
1	0	0	1	1	1	0	0
1	1	0	0	0	0	0	0