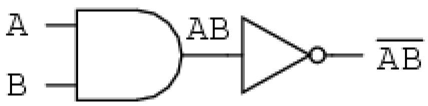
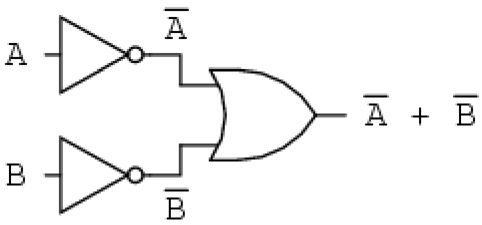
Demorgan's Theorems

A mathematician named DeMorgan developed a pair of important rules regarding group complementation in Boolean algebra. By group complementation.

You should recall from the chapter on logic gates that inverting all inputs to a gate reverses that gate’s essential function from AND to OR, or vice versa, and also inverts the output. So, an OR gate with all inputs inverted (a Negative-OR gate) behaves the same as a NAND gate, and an AND gate with all inputs inverted (a Negative-AND gate) behaves the same as a NOR gate. DeMorgan’s theorems state the same equivalence in “backward” form: that inverting the output of any gate results in the same function as the opposite type of gate (AND vs. OR) with inverted inputs:



. . . is equivalent to . . .



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

A long bar extending over the term AB acts as a grouping symbol, and as such is entirely different from the product of A and B independently inverted. In other words, (AB)’ is not equal to A’B’. Because the “prime” symbol (‘) cannot be stretched over two variables like a bar can, we are forced to use parentheses to make it apply to the whole term AB in the previous sentence. A bar, however, acts as its own grouping symbol when stretched over more than one variable. This has profound impact on how Boolean expressions are evaluated and reduced, as we shall see.

DeMorgan's theorem may be thought of in terms of breaking a long bar symbol. When a long bar is broken, the operation directly underneath the break changes from addition to multiplication, or vice versa, and the broken bar pieces remain over the individual variables.

The two theorems are :

1) $(\overline{x+y}) = \bar{x}.\bar{y}$
2) $(\overline{x.y}) = \bar{x} + \bar{y}$

Example :

Simplify the expression : $Z = \overline{(\bar{A}+C).(B+\bar{D})}$

<u>Solution:-</u>

$$Z = \overline{(\bar{A}+C)} + \overline{(B+\bar{D})}$$
$$= \bar{\bar{A}}.\bar{C} + \bar{B}.\bar{\bar{D}}$$
$$\therefore Z = A.\bar{C} + \bar{B}.D$$

Example :-

Simplify the expression : $Z = \overline{(A+BC).(D+EF)}$

<u>Solution:-</u>

$$Z = \overline{(A+BC)} + \overline{(D+EF)}$$
$$= \bar{A}.\overline{BC} + \bar{D}.\overline{EF}$$
$$= \bar{A}.(\bar{B}.\bar{C}) + \bar{D}.(\bar{E}+\bar{F})$$
$$= \bar{A}.\bar{B} + \bar{A}.\bar{C} + \bar{D}\bar{E} + \bar{D}\bar{F}$$

Example:-

Implement a circuit having the output expression:

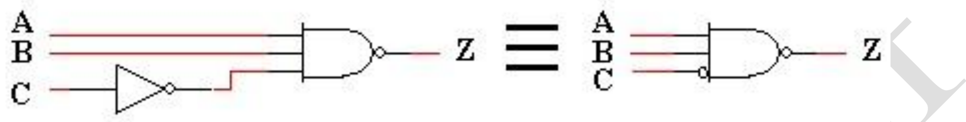
$$Z = \bar{A} + \bar{B} + C$$

Using NAND gate and an inverter.

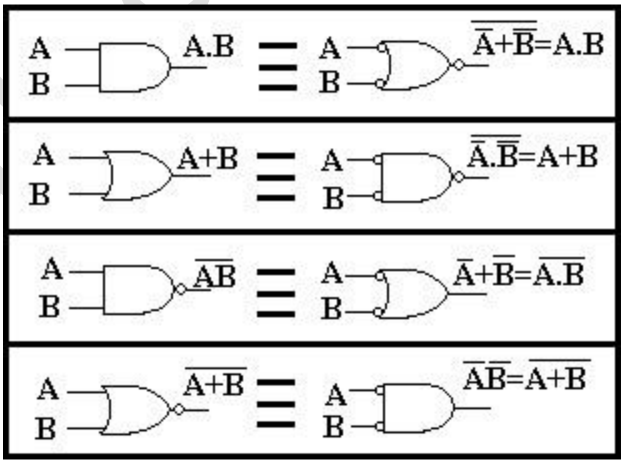
<u>Solution:-</u>

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

$$= \overline{A.B.\bar{C}}$$



<u>**Alternate Logic-Gates Representation :-**</u>



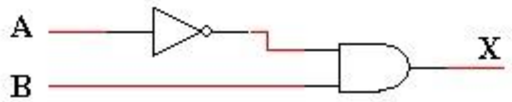
Designing Logic Circuits :-

1) **Sum of products methods :**
   The Boolean expression of any logic circuit can be derived from the truth.

Example:

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

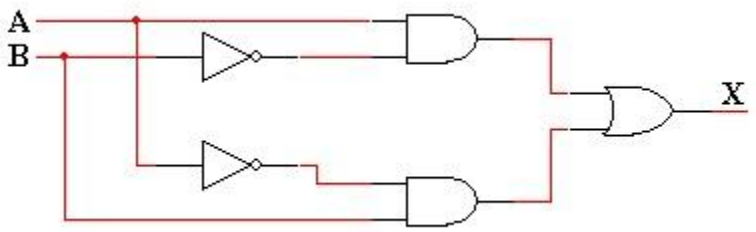
$X = \bar{A}B$



Example:

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

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



The general procedure for obtaining the output expression from truth table can be summarized as follows:

1- Write an AND term for each case in the table where the output is 1.
2- Each AND term contains each input variable in either inverted or non inverted form. If the variable is 0 for that particular case in the table, it is inverted in the AND term.
3- All the AND terms are then ORed together to produce the final expression in the output.

Example:-

Design a logic circuit that has three inputs A,B,C and whose output will be high only when a majority of the inputs is high.

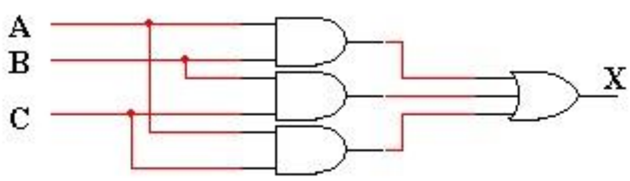
C	B	A	X	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$AB\bar{C}$
1	0	0	0	
1	0	1	1	$A\bar{B}C$
1	1	0	1	$\bar{A}BC$
1	1	1	1	ABC

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

This expression can be simplified as follows:

Rewrite the expression with the ABC term occurring three times (Note: A+A+A=A):

$$X = \bar{A}BC + ABC + A\bar{B}C + ABC + AB\bar{C} + ABC$$
$$X = BC(\bar{A} + A) + AC(\bar{B} + B) + AB(\bar{C} + C)$$
$$\therefore X = AB + BC + AC$$



H.W :-

1) A four logic-signal A,B,C,D are being used to represent a 4-bit binary number with A as the LSB and D as the MSB. The binary inputs are fed to a logic circuit that produces a logic 1 (HIGH) output only when the binary number is greater than $0110_2=6_{10}$. Design this circuit.

2) repeat problem 1 for the output will be 0 (LOW) when the binary input is less than $0111_2=7_{10}$.

2) Product of sums method :-

The product of sums method can be thought of as the dual of the sum of products. It is, in terms of logic functions. For instance, (A+B)(B+C) is a product of sums expression. Several other examples are:

(A+B)(B+C+D)
(A+B+C)(D+E+F)
(A+B+C)(D+E+F+G)(A+E+G)
$(A+B+\overline{C})(\overline{D}+E+F)(F+G+H)(A+\overline{F}+G)$

A product of sums expression can also contain a single variable term such as: A(B+C+D)(E+F+G).

This form also lends itself to straight forward implementation with logic gates because it simply involves ANDing two more OR terms. A two-level gate network will always result, as the following example will show.

Example:-

Construct the following function with logic gates:

(A+B)(C+D+E)(F+G+H+I)

Solution :