MODULE 4

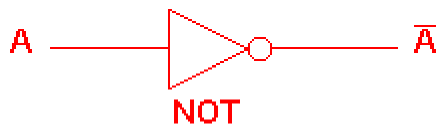
BOOLEAN ALGEBRA AND GATE NETWORKS

LOGIC GATES

1. NOT Gate

It is also called inverter gate. It performs the basic logic functions called inversion or complementation. The purpose of this circuit is to change one level to the opposite

Logic Diagram



Truth table

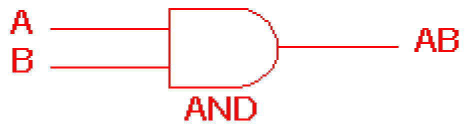
input	output
A	$\bar{A}$
0	1
1	0

Description: If logic 1 is applied to the NOT gate, the result will be 0. If logic 0 is applied to the NOT gate, the result will be 1

2. AND Gate

The AND gate perform the logical multiplication of inputs. It accepts two or more inputs and generates only one output

Logic diagram



Truth table

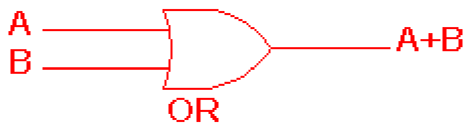
Input		Output
A	B	AB
1	1	1
1	0	0
0	1	0
0	0	0

Description: The result of AND gate is 1, if and only if all the inputs are 1.otherwise the result will be 0

3. OR Gate

OR gate performs the logical addition of inputs. It accepts two or more inputs and generates only one output

Logic diagram



Truth table

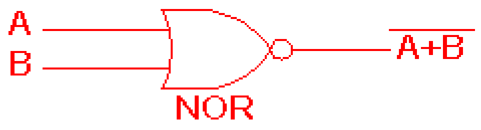
Input		Output
A	B	A+B
1	1	1
1	0	1
0	1	1
0	0	0

Description: The result of OR operation is 0 if and only if all the inputs are 0.Otherwise the result will be 1

4. NOR Gate

It is a combination of OR gate and NOT gate. Ie. It implies the OR function with complemented output. This is an OR gate with it's output are inverted by a NOT gate

Logic diagram



Truth table

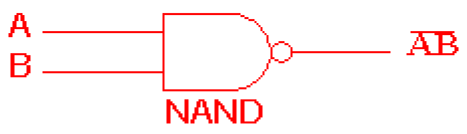
Input		Output
A	B	$\overline{A+B}$
1	1	0
1	0	0
0	1	0
0	0	1

Description: The result of NOR gate is 1 if and only if all the inputs are 0.otherwise the result will be 0

5. NAND Gate

It is a combination of AND gate and NOT gate. Ie. It implies the AND function with complemented output. This is an AND gate with it's output are inverted by a NOT gate

Logic diagram



Truth table

Input		Output
A	B	$\overline{AB}$
1	1	0
1	0	1
0	1	1
0	0	1

Description: the result of NAND gate is 0 , if and only if all the inputs are 1.otherwise the result will be 1

<u>Rules and Laws of Boolean Algebra</u>

<u>Rules of Boolean Addition</u>

0 + 0 =0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 1

Rules of Boolean Multiplication

0 * 0 =0
1 * 0 = 0
0 * 1 = 0
1 * 1 = 1

BASIC LAWS OF BOOLEAN ALGEBRA

There are 3 basic laws

1. Commutative Law

2. Associative Law

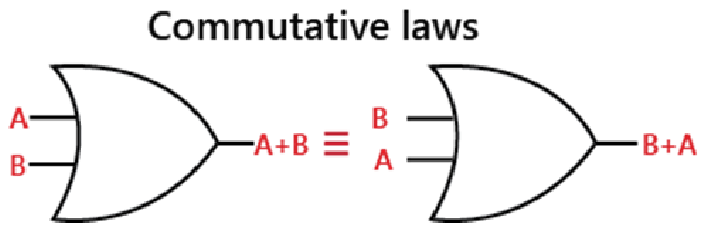
3. Distributive Law

1. Commutative Law

The commutative law of addition of 2 variables is written as

A + B = B + A

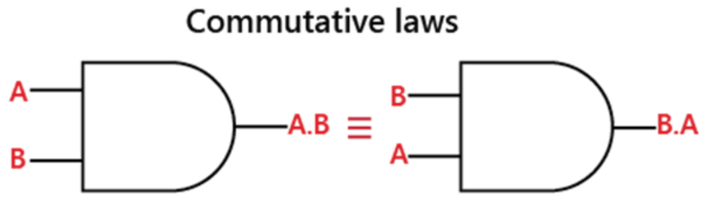
This law states that the order in which the variables are ORed make no difference in results



The commutative law of multiplivation of 2 variables is written as

A B = B A

This law states that the order in which the variables are ANDed make no difference in results

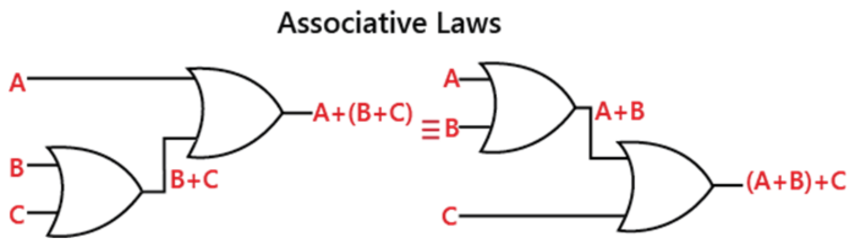


2. Associative Law

The associative law of addition is written as

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

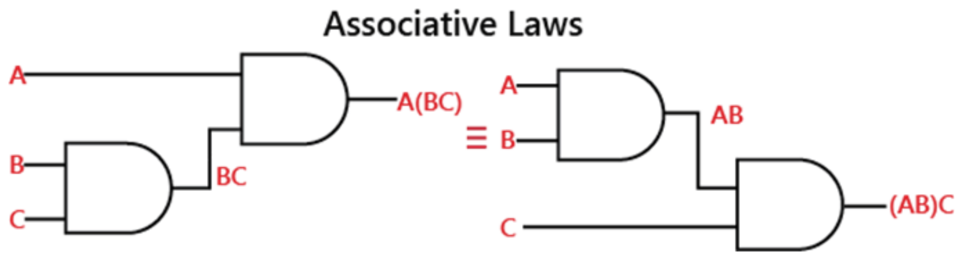
This law states that, there is no differences for the results in what order the variables are grouped when performing the OR operation



The associative law of multiplication is written as

A (BC) = (AB) C

This law states that, there is no differences for the results in what order the variables are grouped when performing the AND operation

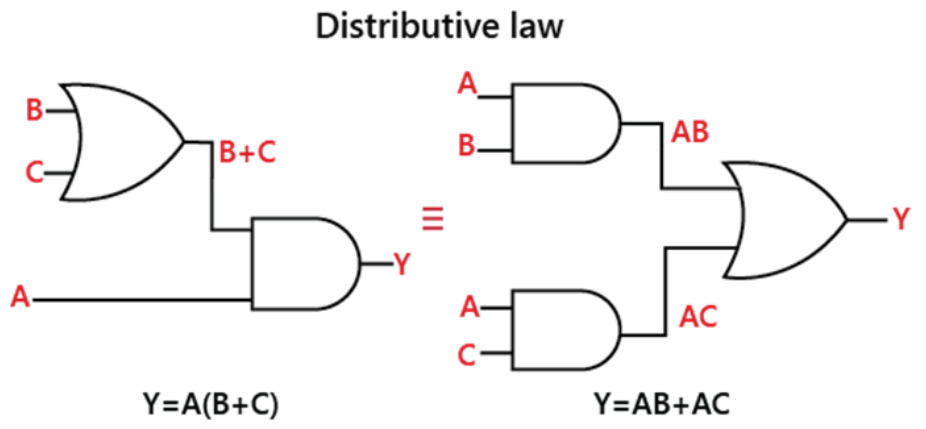


3. Distributive Law

Distributive law states that

A(B + C) = AB + AC

This law states that, the OR of several variables and ANDed the result with a single variable is equal to AND the single variable with each of the several variables and OR the products



RULES OF BOOLEAN ALGEBRA (POSTULATES OF BOOLEAN ALGEBRA)

1.	A+0=A	**7.**	A.A=A
2.	A+1=1	**8.**	A.A'=0
3.	A.0=0	**9.**	A''=A
4.	A.1=A	**10.**	A+AB=A
5.	A+A=A	**11.**	A+A'B=A+B
6.	A+A'=1	**12.**	(A+B)(A+C)=A+BC

PROOF

Rule 1: A + 0 = A

If A=0 then

0+0=0 ->A

If A=1 then

1+0=1 ->A

Rule 2: (A + 1) = 1

If A=0 then

0+1=1 ->1

If A=1 then

1+1=1 ->1

Rule 3: (A.0) = 0

If A=0 then

0.0=0 ->0

If A=1 then

1.0=0 ->0

Rule 4: (A.1) = A

If A=0 then

0.1=0 ->A

If A=1 then

1.1=1 ->A

Rule 5: (A + A) = A

If A=0 then

0+0=0 ->A

If A=1 then

1+1=1 ->A

Rule 6: (A + A') = 1

If A=0 then A’=1

0+1=1 ->1

If A=1 then A’=0

1+0=1 ->1

Rule 7: (A.A) = A

If A=0 then

0.0=0 ->A

If A=1 then

1.1=1 ->A

Rule 8: (A.A') = 0

If A=0 then A’=1

0.1=0 ->0

If A=1 then A’=0

1.0=0 ->0

Rule 9: (A')'=A

If A=1

A’=0

(A')'=1 -> A

If A=0

A’=1

(A')'=0 -> A

Rule 10: (A + AB) = A

A + AB = A(1 + B)
= A.1 Rule 2: (1 + B)= 1
= A Rule 4: A .1 = A

Rule 11: A + A’B = A + B

L.H.S = A+AB

A+AB+A’B rule 10

A.A+AB+A’B rule 7

A.A+AB+A’B+A.A’

A(A+B)+A’(A+B)

(A+A’)(A+B)

1.(A+B)

A+B =R.H.S

Rule 12: (A + B)(A + C) = A + BC

A.A + AC + AB + BC Distributive law
= A + AC + AB + BC Rule 7: AA = A
= A(1 + C)+ AB + BC Rule 2: 1 + C = 1
= A.1 + AB + BC Factoring (distributive law)
= A+AB+BC A+AB=A
= A+ BC =R.H.S

De-Morgan's Theorem

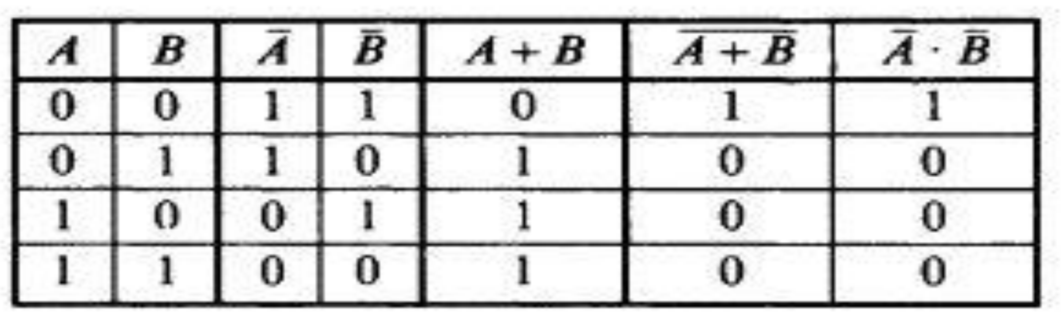
I. $\overline{(A+B)} = \overline{A} . \overline{B}$

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

De-Morgan's first theorem states that $\overline{(A+B)} = \overline{A} . \overline{B}$

The complement of sum is equal to the product of individual complements

Truth table



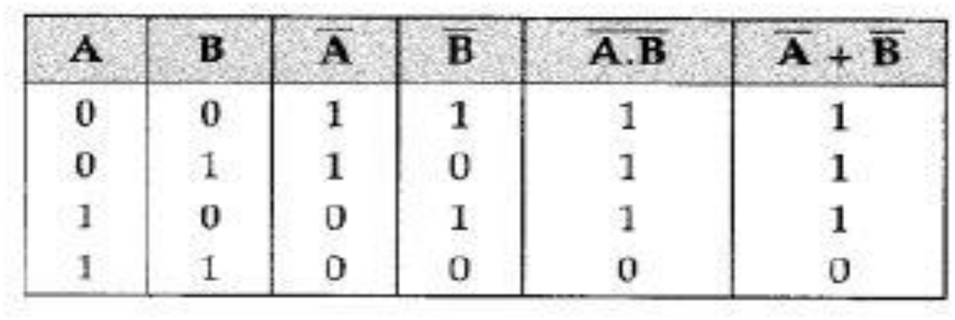
A	B	$\bar{A}$	$\bar{B}$	$A+B$	$\overline{A+B}$	$\bar{A}\cdot\bar{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

De-Morgan's second theorem states that $\overline{(AB)} = \overline{A} + \overline{B}$

The complement of product is equal to the sumt of individual complements

Truth table

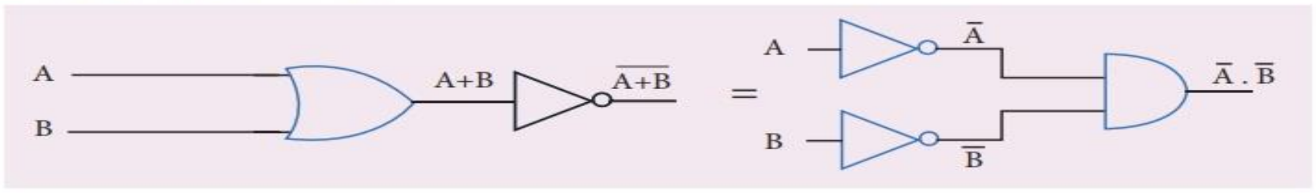
(ii) $\overline{A+B} = \overline{A} . \overline{B}$



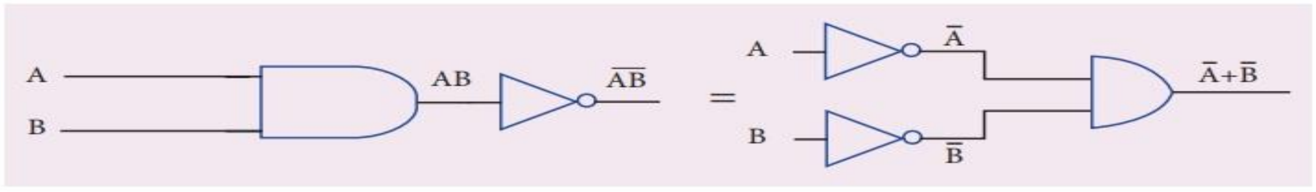
A	B	$\overline{A}$	$\overline{B}$	$\overline{A.B}$	$\overline{A}+\overline{B}$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Diagrams of theorem 1&2

De Morgan's first theorem



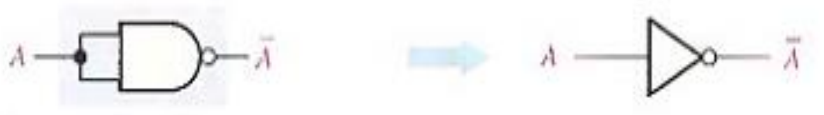
De Morgan's second theorem



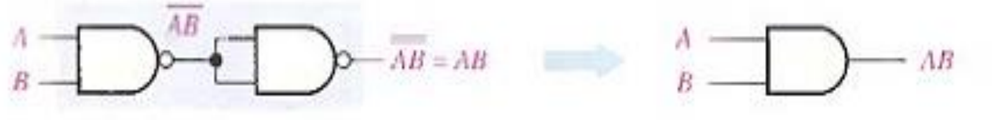
Universal gates

- NAND and NOR gates are collectively called universal gates.
- NAND gate is called universal gate because it can be used to generate NOT, AND, OR and NOR functions
- NOR gate is called universal gate because it can be used to generate NOT, AND, OR and NAND functions

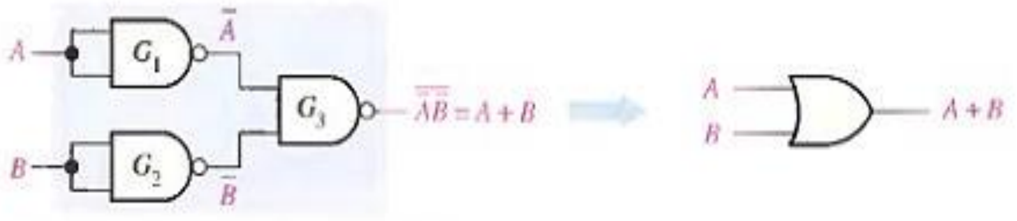
Universal property of NAND gate



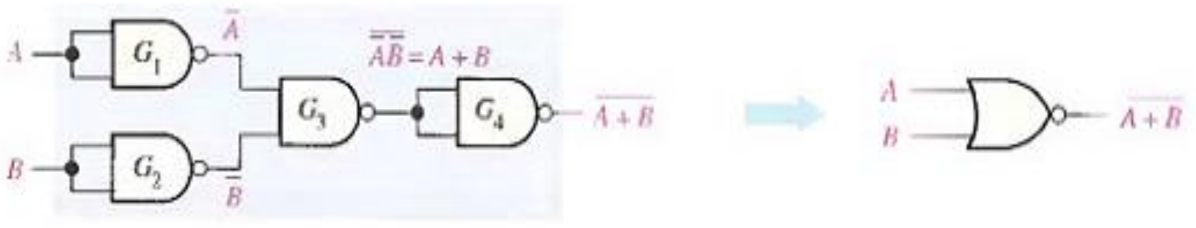
(a) One NAND gate used as an inverter



(b) Two NAND gates used as an AND gate



(c) Three NAND gates used as an OR gate

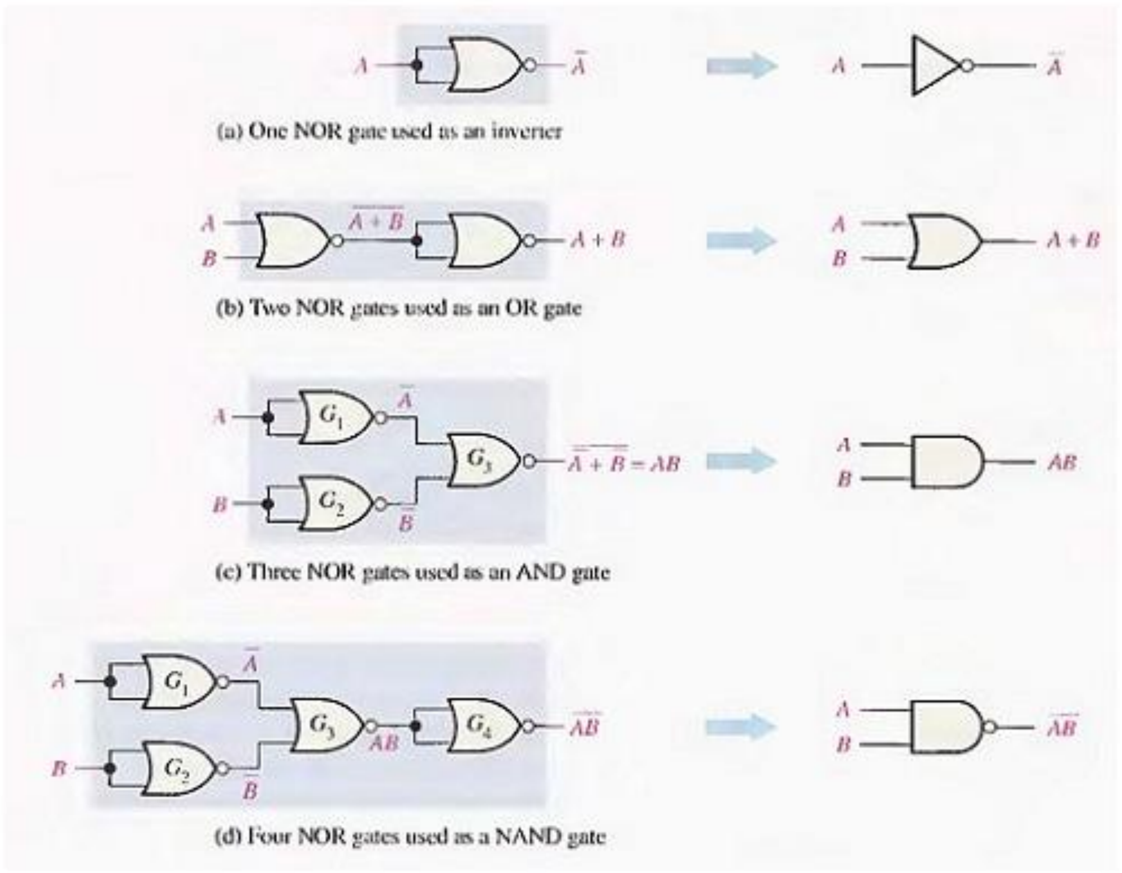


(d) Four NAND gates used as a NOR gate

Universal property of NOR gate

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

Fig.(6-10)(d) shows how NOR gates are used t0 form a NAND function.



A
$\overline{A}$
(a) One NOR gate used as an inverter
A
B
$\overline{A+B}$
$A+B$
(b) Two NOR gates used as an OR gate
G_1
G_2
G_3
$\overline{B}$
$\overline{\overline{A}+\overline{B}}=AB$
AB
(c) Three NOR gates used as an AND gate
G_4
$\overline{AB}$
(d) Four NOR gates used as a NAND gate

Fig.(6-10)

Express the expression using only NAND gate

Step1: Simplify the expression and represent it in Sum of product term if required

Step2: Each product term is represented by the NAND gate

Step3:Draw a single NAND gate to represent the sum of these terms

Express the expression using only NOR gate

Step1: Simplify the expression and represent it in Product of Sum term if required

Step2: Each sum term is represented by the NOR gate

Step3: Draw a single NOR gate to represent the product of these terms

Duality theorem (Dual expressions)

It states that starting with a Boolean relation we can obtain another relation by

1. Changing each OR sign to AND sign
2. Changing each AND sign to OR sign
3. Complementing any 0or1appearingin the relation

Duality theorem: Every algebraic expression deducible from the postulates of Boolean algebra and is remains valid if the operators and identity elements are interchanged

Example:

1. A+B dual= AB
2. (A+B)(0+C) dual=(AB)+(1C)

Product term (Min Term)

A product term is a single variable or logical product of several variables. It is also called min term. In min term the variables may or may not be complemented

Example: ABC, $\overline{A}$B

Sum term (Max Term)

A sum term is a single variable or logical sum of several variables. It is also called max term. In max term the variables may or may not be complemented

Example: A+B+$\overline{C}$, A+B

Product of sum term(POS Expression)

A POS expression is a Boolean expression in which max terms are logically multiplied

Example: (A+B)(C+D)

Sumt of Product term(SOP Expression)

A SOP expression is a Boolean expression in which min terms are logically added

Example: AB +CD

Canonical Expression

The Boolean expression which are expressed as sum of min terms or product of max terms are called canonical expressions

Example: AB+CD+EF -> SOP

(A+B+C) (D+E+F) ->POS

K-MAP(Karnaugh Map)

It is a graphical method used to simplify logical expression. The K-map is a diagram which made-up of squares. Each square represent one minterm.

Usually this method is employed for solving 2,3,4 variable simplification.

Once the Boolean expression is in SOP form, we can plot it in the K-map by placing 1 in each corresponding cell

If the map contains horizontally or vertically adjacent 1's, then we represent t it as a pair

If the map contain group of 4 one's, which are horizontally or vertically adjacent, then we can represent it as a Quad. The 1's may or may not be in end-to-end form

If the map contain group of 8 one's, which are horizontally or vertically adjacent, then we can represent it as an Octant. The 1's may or may not be in end-to-end form

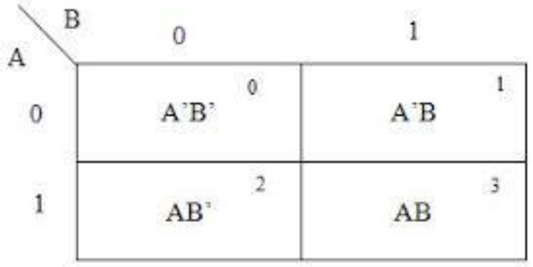
The map method is first proposed by veitch and is modified by karnaugh. So that the map is also called veitch diagram

Simplifying the expression using K-map

In order to simplify the expression, consider each group of 1's to create minterm. The variables that may appear both complemented and un-complemented form are eliminated from the SOP expression

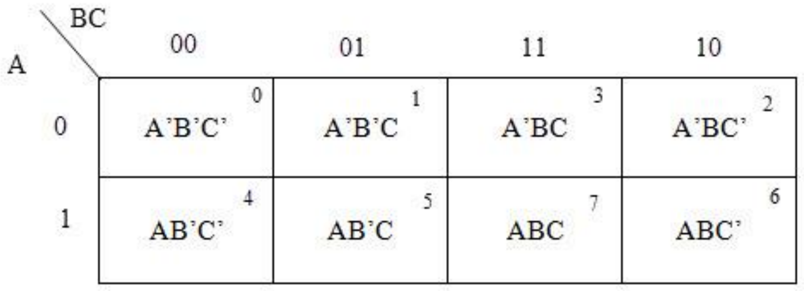
If the expression contains n terms, then the K-map contains 2^n cells

Diagram of 2 variables K-map



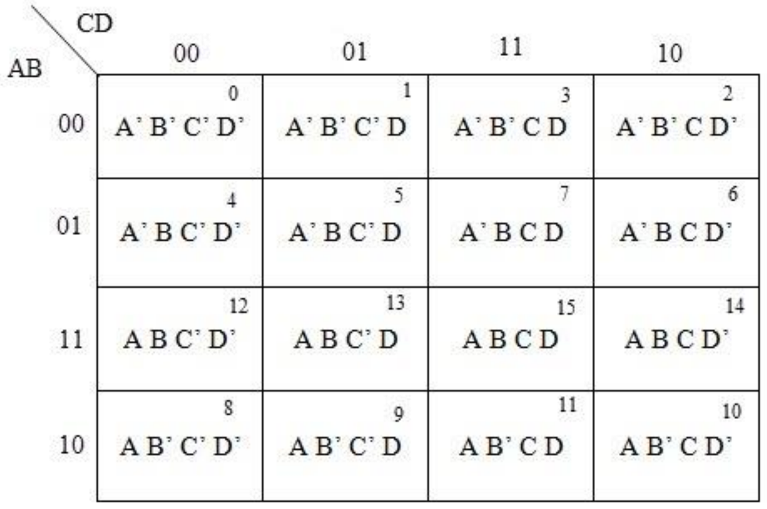
A \ B	0	1
0	A’B’ 0	A’B 1
1	AB’ 2	AB 3

Diagram of 3 variables K-map



A \ BC	00	01	11	10
0	A’B’C’ 0	A’B’C 1	A’BC 3	A’BC’ 2
1	AB’C’ 4	AB’C 5	ABC 7	ABC’ 6

Diagram of 4 variables K-map

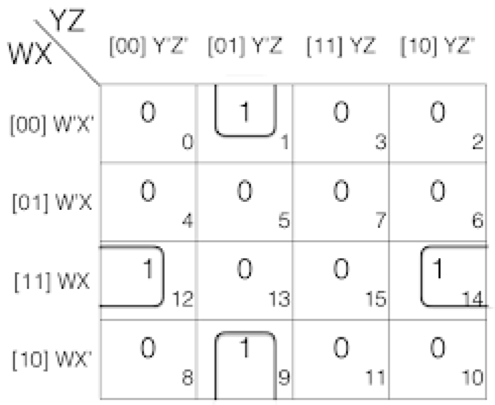


AB \ CD	00	01	11	10
00	A’ B’ C’ D’ 0	A’ B’ C’ D 1	A’ B’ C D 3	A’ B’ C D’ 2
01	A’ B C’ D’ 4	A’ B C’ D 5	A’ B C D 7	A’ B C D’ 6
11	A B C’ D’ 12	A B C’ D 13	A B C D 15	A B C D’ 14
10	A B’ C’ D’ 8	A B’ C’ D 9	A B’ C D 11	A B’ C D’ 10

[Problems Simplification]

Map Rolling

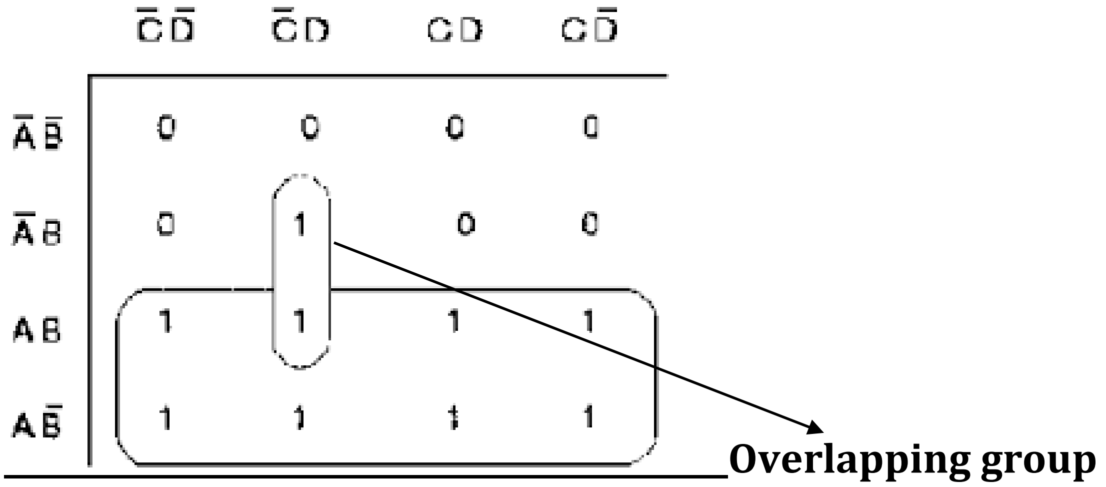
Map rolling means roll the map. That is, when solving K-map, we consider the combination of 1’s in different edges of the map. We check whether the left edges are touching the right edges and top edges are touching the bottom edges. Then we can say that map rolling is occurred in that k-map



WX \ YZ	[00] Y'Z'	[01] Y'Z	[11] YZ	[10] YZ'
[00] W'X'	0 (0)	1 (1)	0 (3)	0 (2)
[01] W'X	0 (4)	0 (5)	0 (7)	0 (6)
[11] WX	1 (12)	0 (13)	0 (15)	1 (14)
[10] WX'	0 (8)	1 (9)	0 (11)	0 (10)

Overlapping Groups

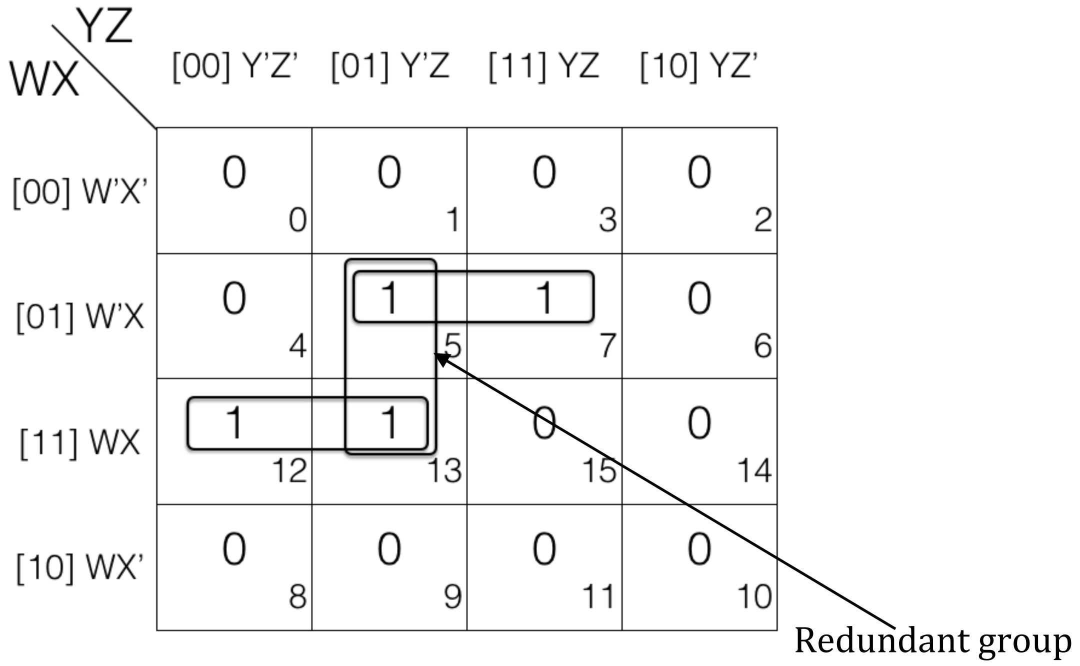
Overlapping means when grouping 1’s in a k-map, same 1 can be included in more than one group. Then that type of groups are called overlapping groups.



	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	1	0	0
AB	1	1	1	1
$A\bar{B}$	1	1	1	1

Redundant Groups

It is the group in which all the 1's in one group is overlapped by other groups. This type of redundant group are eliminated to get the simplified expression



Product of Sum reduction (POS Reduction)

In POS reduction, each square of k-map represent a max term. For the simplification we put 0's instead of 1's in K-map.

The major difference with SOP reduction is that, here the complemented letters represented 1 and the un-complemented letters represent 0.

[Problems Simplification]

Don't care conditions

This type of condition states that, we don't need to care about what value is assumed by the function. The function may either produce 0 or produce 1 as output. We don't need to care about what the function output is. These conditions are used only to fill up other bit combinations. Don't care conditions are marked with the symbol 'X' in the K-map

[Problems Simplification]

Parity Generator and Checker

Parity bit

It is used for error detection in data transmission. In digit al system, when binary data is transmitted and processed, data may be subjected to noise(disturbance, modification, alteration).Such a noise can alter 0's to 1's and vice versa. So that the parity bit is added to the message to detect errors.

Parity Generator

It is combinational logic circuit that generates the parity bit at the transmitter. It accepts n-1 bit data and generate the additional bit called parity bit, that is to be transmitted along with data

2 types of parity mechanisms

- Even parity mechanism
- Odd parity mechanism

Even parity mechanism: In this mechanism, the parity it is 0, if there are even number of 1's in the data .otherwise the parity bit is 1

Odd parity mechanism: In this mechanism, the parity it is 0, if there are odd number of 1's in the data .otherwise the parity bit is 1

Truth table of Even parity generator

3-bit message			Even parity bit generator (P)
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Truth table of Odd parity generator

3-bit message			Odd parity bit generator (P)
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Parity checker

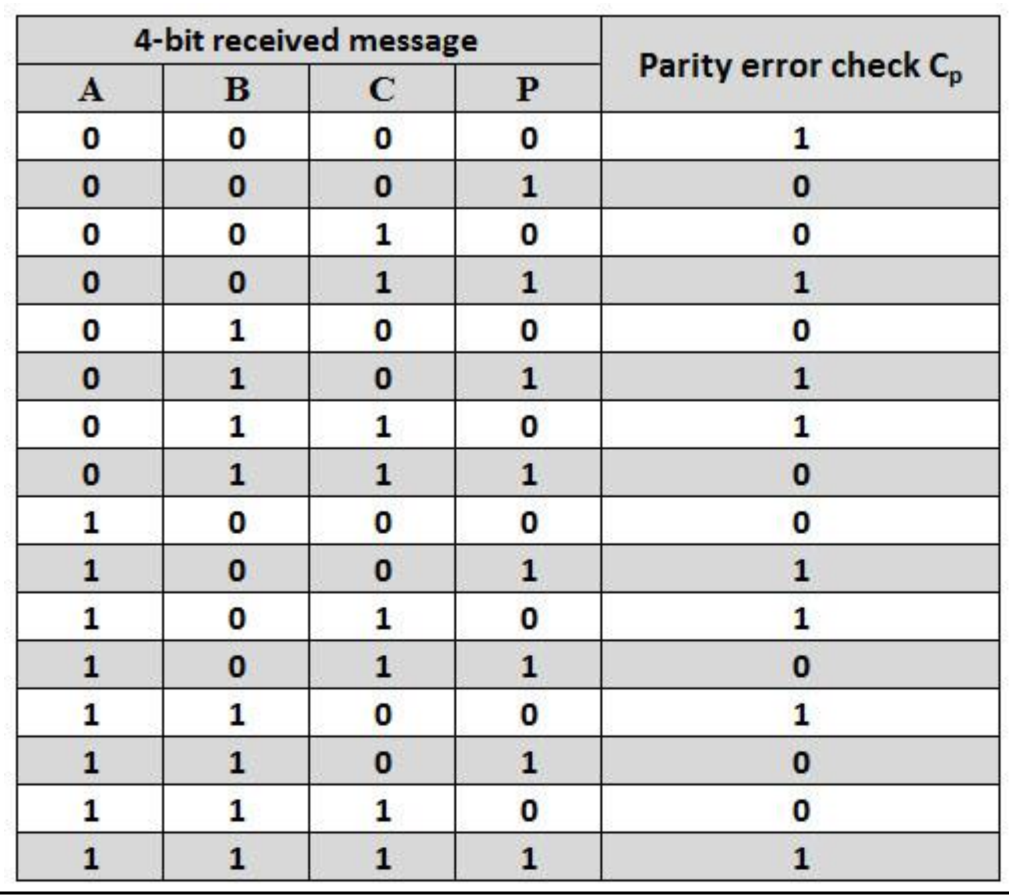
It is also combinational logic circuit which checks the data along with the parity bit. This circuit checks for possible errors in data transmission.

In data transmission, both receiver and transmitter should use same parity mechanism

Even parity checker checks the data along with the parity bit and PEC(parity error checker) is set to 0, if the received message has even number of 1's (no error).otherwise PEC is set to 1(presence of error)

odd parity checker checks the data along with the parity bit and PEC(parity error checker) is set to 0, if the received message has odd number of 1's (no error).otherwise PEC is set to 1(presence of error)

Truth table of odd parity checker

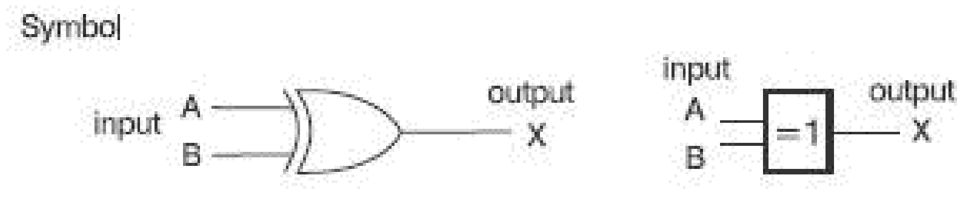


4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

XOR Gate

The output of XOR gate is 1 when both inputs are different. The output of XOR gate is 0 when both inputs are same.

Diagram & truth table



An XOR gate

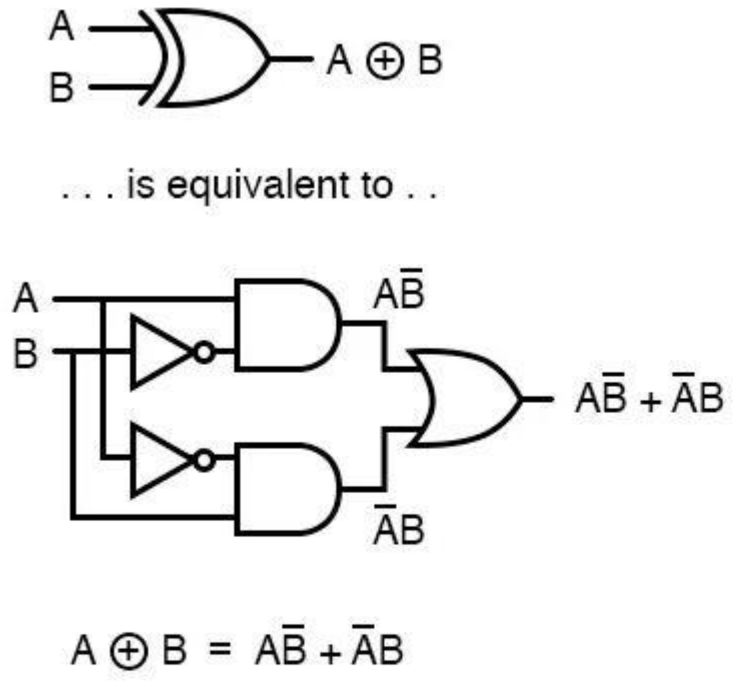
Truth table

input		output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

This is an inverted version of the XOR

Applications of XOR Gate

- Arithmetic operations
- To implement parity checker
- To implement controlled inverter
- Used to binary to gray and gray to binary conversion
- Used to minimize combinational circuits


A
B
$A \oplus B$
. . . is equivalent to . .
A
B
$A\overline{B}$
$\overline{A}B$
$A\overline{B} + \overline{A}B$
$A \oplus B = A\overline{B} + \overline{A}B$