CHAPTER 4
BOOLEAN ALGEBRA

4.1 Definition

Boolean algebra is the mathematics of digital logic in which the values of the variables are the truth values true and false, usually denoted 1 and 0, respectively.

Basic knowledge of Boolean algebra is indispensable to the study and analysis of logic circuits. *Variable, complement,* and literal are terms used in Boolean algebra. A variable is a symbol (usually an italic uppercase letter or word) used to represent an action, a condition, or data. Any single variable can have only a 1 or a 0 value. The complement is the inverse of a variable and is indicated by a bar over the variable (overbar).

There are four connecting symbols used in Boolean algebra:

1. Equal sign (=): This refers to the sign of equality as in mathematics.

2. Multiplication sign (·): It refers to the AND operation.

3. Plus sign (+): This refers to the OR operation.

4. Inversion sign (‘) or (–): This operation performs a complement of the input given to the logic gate.

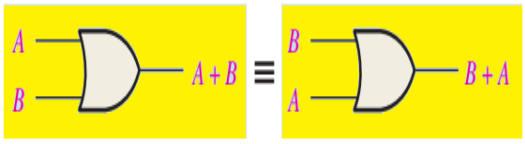
4.2 Laws of Boolean Algebra

The basic laws of Boolean algebra—the commutative laws for addition and multiplication, the associative laws for addition and multiplication, and the distributive law—are the same as in ordinary algebra. Each of the laws is illustrated with two or three variables, but the number of variables is not limited to this.

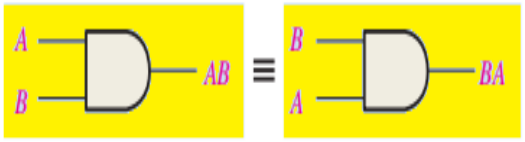
4.2.1 Commutative Laws

The commutative laws of addition and multiplication for two variables are written as:

$$A + B = B + A$$



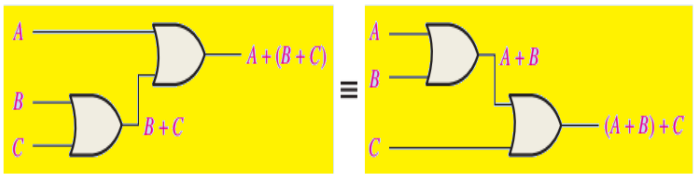
$$AB = BA$$



4.2.2 Associative Laws

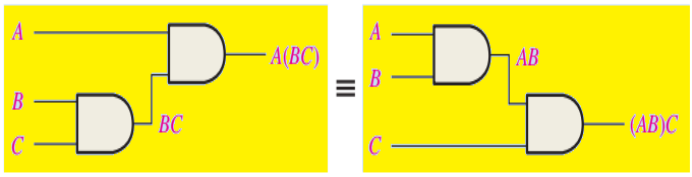
This law states that when ORing more than two variables, the result is the same regardless of the grouping of the variables. The figure below illustrates this law as applied to 2-input OR gates. The associative law of addition is written as follows for 3 variables:

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



This law states that it is no difference in the order in which the variables are grouped when ANDing more than two variables. The figure below illustrates this law as applied to 2-input AND gates. The associative law of multiplication is written as follows for 3 variables:

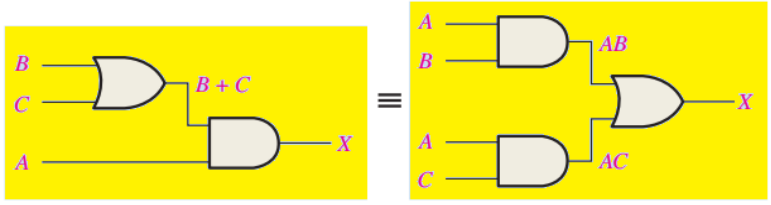
$$A\,(BC) = (AB)\,C$$



4.2.3 Distributive Law

The distributive law is written for three variables as follows:

$$A\,(B + C) = AB + AC$$



$X = A(B + C)$ $\qquad$ $X = AB + AC$

4.3 Rules of Boolean Algebra

The table below lists 12 basic rules that are useful in manipulating and simplifying Boolean expressions. A, B, or C can represent a single variable or a combination of variables.

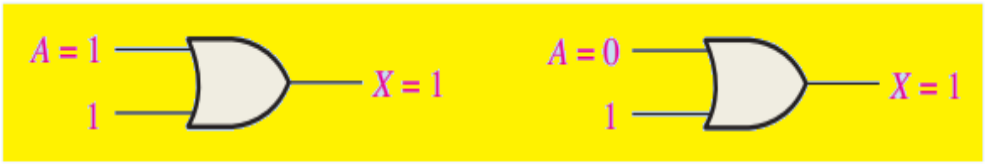
Basic rules of Boolean algebra

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

Rule 1: A variable ORed with 0 is always equal to the variable. If the input variable A is 1, the output variable X is 1, which is equal to A. If A is 0, the output is 0, which is also equal to A.

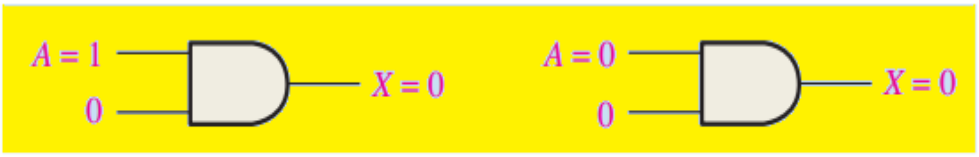
$X = A + 0 = A$

Rule 2: A variable ORed with 1 is always equal to 1. A 1 on an input to an OR gate produces a 1 on the output, regardless of the value of the variable on the other input.



$X = A + 1 = 1$

Rule 3: A variable ANDed with 0 is always equal to 0. Any time one input to an AND gate is 0, the output is 0, regardless of the value of the variable on the other input.



$$X = A \cdot 0 = 0$$

Rule 4: A variable ANDed with 1 is always equal to the variable. If A is 0, the output of the AND gate is 0. If A is 1, the output of the AND gate is 1 because both inputs are now 1s.

$$X = A \cdot 1 = A$$

Rule 5: A variable ORed with itself is always equal to the variable. If A is 0, then 0 + 0 = 0; and if A is 1, then 1 + 1 = 1.

$$X = A + A = A$$

Rule 6: A variable ORed with its complement is always equal to 1. If A is 0, then 0 + 0 = 0 + 1 = 1. If A is 1, then 1 + 1 = 1 + 0 = 1.

$$X = A + \bar{A} = 1$$

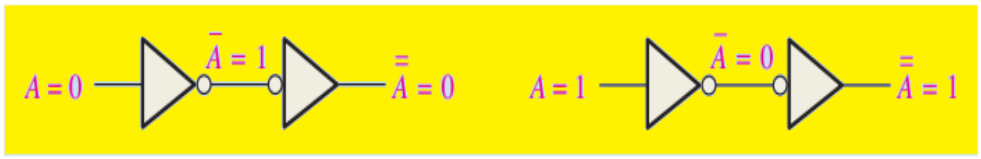
Rule 7: A variable ANDed with itself is always equal to the variable. If A = 0, then 0 #0 = 0; and if A = 1, then 1 #1 = 1.

$X = A \cdot A = A$

Rule 8: A variable ANDed with its complement is always equal to 0. Either A or A will always be 0; and when a 0 is applied to the input of an AND gate, the output will be 0 also.

$X = A \cdot \bar{A} = 0$

Rule 9: The double complement of a variable is always equal to the variable. If you start with the variable A and complement (invert) it once, you get A. If you then take A and complement (invert) it, you get A, which is the original variable.



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

Rule 10: This rule can be proved by applying the distributive law, rule 2, and rule 4 as follows:

$A + AB = A \,.\, 1 + AB = A\,(1 + B)$ Factoring (distributive law)

$= A \,.\, 1$ Rule 2: $(1 + B) = 1$

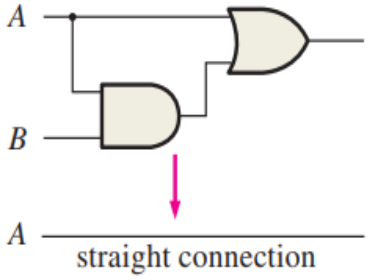
$= A$ Rule 4: $A \,.\, 1 = A$

The proof is shown in below, which shows the truth table and the resulting logic circuit simplification.

Rule 10: A+AB = A

A	*B*	*AB*	*A + AB*
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

equal



Rule 11: This rule can be proved as follows:

$A + \overline{A}B = (A + AB) + \overline{A}B$ Rule 10: A = A + AB

$=(AA + AB) + \overline{A}B$ Rule 7: A = AA

$=AA + AB + A\overline{A} + \overline{A}B$ Rule 8: adding $A\overline{A} = 0$

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

$= 1 . (A + B)$ Rule 6: A + A = 1

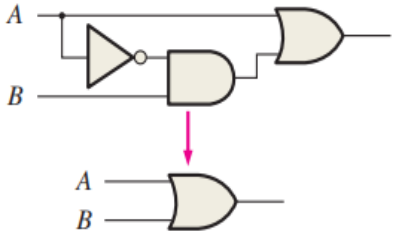
$= A + B$ Rule 4: drop the 1

The proof is shown in the table below, which shows the truth table and the resulting logic circuit simplification.

Rule 11: A+$\overline{A}$B = A+B

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

equal



Rule 12: This rule can be proved as follows:

(A + B)(A + C) = AA + AC + AB + BC Distributive law

= A + AC + AB + BC Rule 7: AA = A

= A(1 + C) + AB + BC Factoring (distributive law)

= A .1 + AB + BC Rule 2: 1 + C = 1

= A(1 + B) + BC Factoring (distributive law)

= A . 1 + BC Rule 2: 1 + B = 1

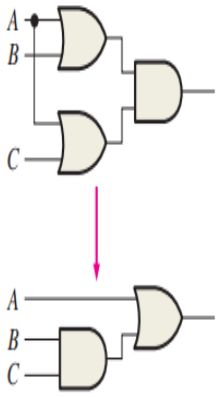
= A + BC Rule 4: A. 1 = A

The proof is shown in the table below, which shows the truth table and the resulting logic circuit simplification.

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

A	B	C	A + B	A + C	(A + B)(A + C)	BC	A + BC
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

equal



H.W:

1. Apply the associative law of addition to the expression A + (B + C + D).
2. Apply the distributive law to the expression A(B + C + D).

4.4 DeMorgan's Theorems

DeMorgan, a mathematician, proposed two theorems that are an important part of Boolean algebra. In practical terms, DeMorgan's theorems provide mathematical verification of the equivalency of the NAND and negative-OR gates and the equivalency of the NOR and negative-AND gates.

4.4.1 DeMorgan's first theorem

The complement of a product of variables is equal to the sum of the complements of the variables.

Stated another way,

The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.

The formula for expressing this theorem for two variables is

$$\overline{XY} = \bar{X} + \bar{Y} \quad \ldots\ldots\ldots \quad (1)$$

4.4.2 DeMorgan's second theorem

The complement of a sum of variables is equal to the product of the complements of the variables.

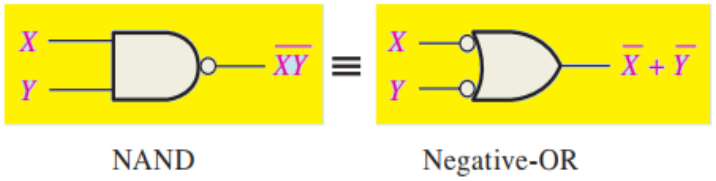
Stated another way,

The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables.

The formula for expressing this theorem for two variables is

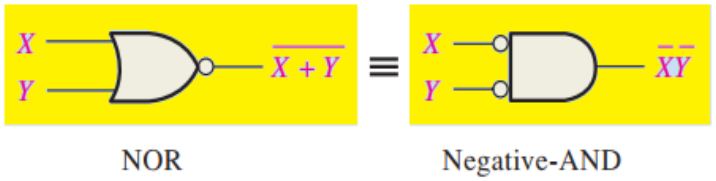
$$\overline{X + Y} = \bar{X}\bar{Y} \quad \ldots\ldots\ldots \quad (2)$$

Figure below shows the gate equivalencies and truth tables for Equations 1 and 2.



NAND Negative-OR

Inputs		Output	
X	Y	$\overline{XY}$	$\overline{X}+\overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0



NOR Negative-AND

Inputs		Output	
X	Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Example**:** Apply DeMorgan's theorems to the expressions $\overline{XYZ}$ and $\overline{X+Y+Z}$.

Sol.

$$\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$$
$$\overline{X+Y+Z} = \overline{X}\,\overline{Y}\,\overline{Z}$$

Example**:** Apply DeMorgan's theorems to the expressions $\overline{WXYZ}$ and $\overline{W+X+Y+Z}$.

Sol.

$$\overline{WXYZ} = \overline{W} + \overline{X} + \overline{Y} + \overline{Z}$$
$$\overline{W+X+Y+Z} = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$$

Example: Prove that $A+AB = A$

Sol.

$$A+AB = A\,(1+B)$$
$$= A \cdot 1 = A$$

Example: Prove that $A + \bar{A}B = A + B$

Sol.

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

Example: Apply DeMorgan's theorems to each expression:

(a) $\overline{\overline{(A + B)} + \overline{C}}$

(b) $\overline{\overline{(A} + B) + CD}$

(c) $\overline{(A + B)\overline{C}\,\overline{D} + E + \overline{F}}$

Sol.

(a) $\overline{\overline{(A + B)} + \overline{C}} = (\overline{\overline{A + B}})\overline{\overline{C}} = (A + B)C$

(b) $\overline{\overline{(A} + B) + CD} = (\overline{\overline{A} + B})\overline{CD} = (\overline{\overline{A}}\,\overline{B})(\overline{C} + \overline{D}) = A\overline{B}(\overline{C} + \overline{D})$

(c) $\overline{(A + B)\overline{C}\,\overline{D} + E + \overline{F}} = \overline{((A + B)\overline{C}\,\overline{D})}\,\overline{(E + \overline{F})} = (\overline{A}\,\overline{B} + C + D)\overline{E}F$

Example: Prove that $(A + B)(A + C) = A + BC$

Sol.

$$(A+B)(A+C) = AA + AC + AB + BC$$
$$= A + AC + AB + BC$$
$$= A + (1+C) + AB + BC$$
$$= A.1 + AB + BC$$
$$= A + AB + BC$$
$$= A(1 + B) + BC$$
$$= A.1 + BC$$
$$= A + BC$$

H.W.:

Apply DeMorgan's theorems to the following expressions:

1. $\overline{ABC} + (\overline{\overline{D} + E})$ 2. $\overline{(A + B)C}$ 3. $\overline{A + B + C} + \overline{\overline{D}E}$

2. Apply DeMorgan's theorems to the expression $\overline{\overline{ABC} + D + E}$.

4.5 Logic Simplification Using Boolean Algebra

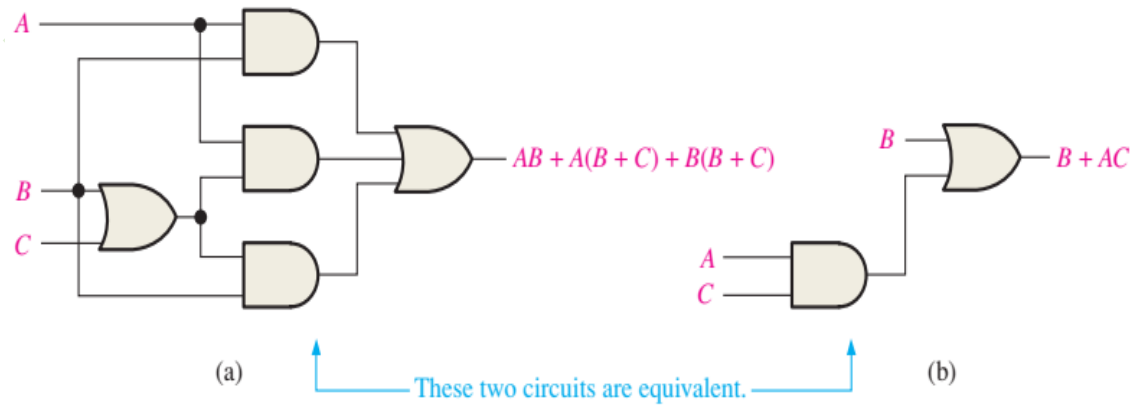
A logic expression can be reduced to its simplest form or changed to a more convenient form to implement the expression most efficiently using Boolean algebra.

Example**:** Using Boolean algebra techniques, simplify this expression:
AB + A(B + C) + B(B + C)

Sol.

AB + AB + AC + BB + BC distributive law to the second and third terms
AB + AB + AC + B + BC (BB = B) rule 7
AB + AC + B + BC (AB + AB = AB) rule 5
AB + AC + B (B + BC = B) rule 10
B + AC (AB + B = B) rule 10

The figure below shows that the simplification process significantly reduced the number of logic gates required to implement the expression. Part (a) shows that 5 gates are required to implement the expression in its original form; however, only 2 gates are needed for the simplified expression, shown in part (b). It is important to realize that these two gate circuits are equivalent.



Example**:** Simplify the following Boolean expression:

$$\overline{A\bar{B} + AC} + \bar{A}\bar{B}C$$

Sol.

$(\overline{A\bar{B}})(\overline{AC}) + \bar{A}\bar{B}C$ DeMorgan's theorem

$(\bar{A} + \bar{B})(\bar{A} + \bar{C}) + \bar{A}\bar{B}C$ DeMorgan's theorem

$\bar{A}\bar{A} + \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{B}C$ distributive law to the two terms

$\bar{A}\bar{B} + \bar{A}\bar{B}C = \bar{A}\bar{B}(1 + C) = \bar{A}\bar{B}$ rule 7 ($\bar{A}\bar{A} = \bar{A}$) = A) to the first term

$\bar{A} + \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B}\bar{C}$ rule 10 to the third and last terms.

$\bar{A} + \bar{A}\bar{C} = \bar{A}(1 + \bar{C}) = \bar{A}$ rule 10 to the first and second terms

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

$\bar{A} + \bar{A}\bar{B} = \bar{A}(1 + \bar{B}) = \bar{A}$ rule 10 to the first and second terms.

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

H.W.:

1. Simplify the following Boolean expressions:
   a. $A + AB + A\bar{B}C$ b. $(\bar{A} + B)C + ABC$ c. $A\bar{B}C(BD + CDE) + A\bar{C}$
2. Implement each expression in Question 1 as originally stated with the appropriate logic gates. Then implement the simplified expression, and compare the number of gates.

4.6 Boolean Expressions For Truth Table

All Boolean expressions, regardless of their form, can be converted into either of two standard forms: the sum-of-products form or the product-of-sums form. Standardization makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier.

4.6.1 The Sum-of-Products (SOP) Form (Minterm)

This form is sometimes called "minterm". A product term that contains each of the n-variables factors in either complemented or uncomplemented form for output digits "1" only, is called SOP. For example for the truth table below:

Input			Output	
A	B	C	F	
0	0	0	1	$\bar{A}\bar{B}\bar{C}$
0	0	1	0	$\bar{A}\bar{B}C$
0	1	0	1	$\bar{A}B\bar{C}$
0	1	1	1	$\bar{A}BC$
1	0	0	0	$A\bar{B}\bar{C}$
1	0	1	0	$A\bar{B}C$
1	1	0	1	$AB\bar{C}$
1	1	1	1	ABC

The Logical SOP expression for the output digit "1" is written as"

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

This function com be put in another form such as:

$F = \sum 0, 2, 3, 6, 7$ Since F= 1 in rows 0, 2,3,6,7 only.

The second form is called the Canonical Sum of Products (Canonical SOP).

4.6.2 The Product-of-Sum (POS) Form (Maxterm)

A Logical equation can also be expressed as a product of sum (POS) form (sometimes this method is called "Maxterm". This is done by considering the combination for F=0 (output = 0). So for the above example from the truth table F=0 is in rows 1, 4, 5 hence:

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

$$F(A,B,C) = \bar{\bar{F}}(A,B,C) = \overline{\bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C}$$

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

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

$$F(A,B,C) = (A + B + \bar{C}) \cdot (\bar{A} + \bar{\bar{B}} + \bar{\bar{C}}) \cdot (\bar{A} + \bar{\bar{B}} + \bar{C})$$

This is POS form. POS form can be expressed as:

$$F = \prod 1, 4, 5$$

This form is called the Canonical Product of Sum (Canonical POS).

Example**:** Put F in SOP and POS form and simplifying it:

A	B	F
0	0	1
0	1	1
1	0	0
1	1	1

Sol.

$$SOP: \quad F(A,B) = \sum 0,1,3$$

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

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

$$F(A,B) = \bar{A} + B$$

$$POS: \quad F(A,B) = \prod 2$$

$$F(A,B) = \bar{A} + B$$

Example**:** Put in canonical SOP form

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

Sol.

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

$$101 \quad 011 \quad 111$$

$$F(A,B,C) = \sum 3,5,7$$

Example**:** Put in canonical POS form and draw the truth table, then determine canonical SOP and SOP form

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

Sol.

$$F(A,B,C) = \quad 001 \quad 010 \quad 111 \quad 110$$

$$M_1 \quad M_2 \quad M_3 \quad M_4$$

$$F(A,B,C) = \prod 1,2,6,7$$

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$F(A,B,C) = \sum 0,3,4,5$$

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

Example**:** Represent F_1, F_2 in SOP & POS forms then simplified F_1 and F_2 using Boolean algebra.

A	B	C	F_1	F_2
0	0	0	0	1
0	0	1	1	0
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Sol.

In SOP:

$$F_1(A,B,C) = \sum 1,2,3,5,6,7$$

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

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

$$= \bar{A}[\bar{B}C + B(\bar{C} + C)] + A[\bar{B}C + B(\bar{C} + C)]$$

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

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

$$F_1(A, B, C) = B + C$$

In POS:

$$F(A, B, C) = \prod 0,4$$

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

$$= A\bar{A} + AB + AC + \bar{A}B + BB + BC + C\bar{A} + CB + CC$$

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

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

$$= AB + AC + \bar{A}B + B + \bar{A}C + C$$

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

$$= B + C + B + C$$

$$F_1(A, B, C) = B + C$$

H.W.: Solution for F_2

4.6.3 Converting SOP to POS and Vice Versa

The binary values of the product terms in a given SOP expression aren't present in the equivalent POS expression. Therefore to convert from standard SOP to standard POS the following steps may be used:

Step 1: Evaluate each product term in the SOP expression that determines the binary numbers representing the product term.

Step 2: Determine all the binary numbers not included in the evaluation in step 1.

Step 3: Write the equivalent sum term for each binary number from step 2 and express it in POS form.

Note: *A Standard SOP expression is one in which all the variables in the domain appear in each term of the expression. If any variable is missing from*

any term, we must add these missing variables to that term, by multiplying the term by the variables missing.

For example, if variable B is missing from the term AC, we must multiply this term AC, by $B + \bar{B}$ to make the expression standard SOP.

$$AC(B + \bar{B})$$

***Note**: using a similar procedure explained above (steps 1, 2, and 3) we can convert from standard POS to standard SOP. If there is missing any variable from any term, we must add the missing variable multiplied by its complement to that term.*

For example if variable A is missing from the term $(B + \bar{C})$ we must add $A\bar{A}$

$$[(B + \bar{C}) + A\bar{A}]$$
$$= (B + \bar{C} + A)(B + \bar{C} + \bar{A})$$

Example**:** Put in canonical POS form and draw the truth table, then determine canonical SOP and SOP form

$$F(A, B, C) = B + AC$$

Sol.

1st method

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

111 110 011 010 101

$$\therefore F(A, B, C) = \sum 2,3,5,6,7$$

$$\therefore F(A, B, C) = \prod 0,1,4$$

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

2nd method:

$F(A,B,C) = B + AC$

A	B	C	AC	F= B+AC
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

$\therefore F(A,B,C) = \sum 2,3,5,6,7$

$\therefore F(A,B,C) = \prod 0,1,4$

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

H.W.: Convert the POS form to SOP form and find these canonical:

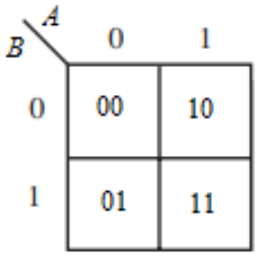
$F(A,B,C) = (A+B)(\bar{A}+C)(A+B+\overline{C})$

4.7 The Karnaugh Map (K-map)

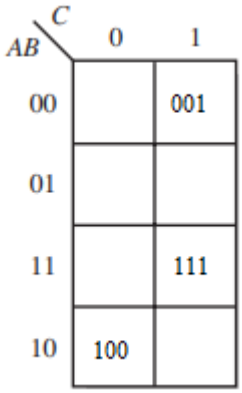
A K- map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression. As you have seen, the effectiveness of algebraic simplification depends on your familiarity with all the laws, rules, and theorems of Boolean algebra and on your ability to apply them. The K-map is an array of cells in which each cell represents a binary value of the input variables.

The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells. The K-maps can be used for expressions with two, three, four, and five variables, but we will discuss only 2, 3, and 4 variables. The number of cells in a K-map, as well as the number of rows in a truth table.

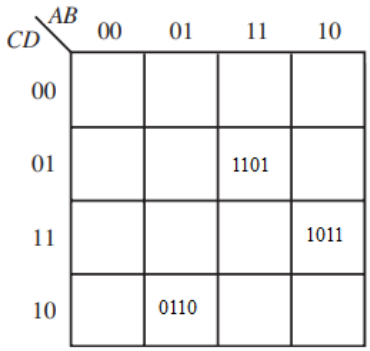
For 2 input variables, the number of cells is $2^2 = 4$ cells



For 3 input variables, the number of cells is $2^3 = 8$ cells



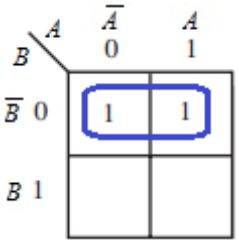
And for 4 input variables, the number of cells is $2^4 = 16$ cells



4.7.1 The 2-varieables K - map

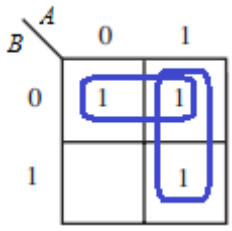
1. $F(A,B) = \bar{B}$ SOP

   $F(A,B) = B$ POS



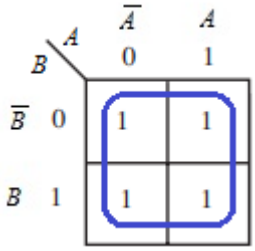
2. $F(A,B) = \bar{B} + A$ SOP

   $F(A,B) = A + \bar{B}$ POS



3.

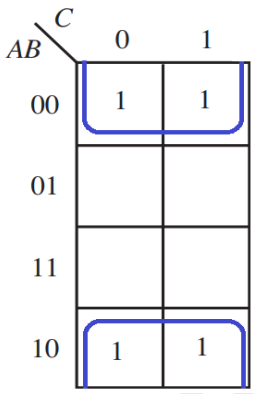
$F(A,B) = 1$



4.7.2 The 3-variebles K - map

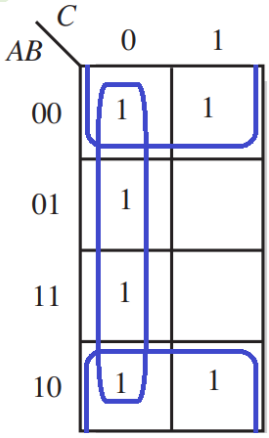
1. $F(A,B,C) = \bar{B}$ SOP

   $F(A,B,C) = \bar{B}$ POS



2. $F(A,B,C) = \bar{C} + \bar{B}$ SOP

   $F(A,B,C) = \bar{B} + \bar{C}$ POS

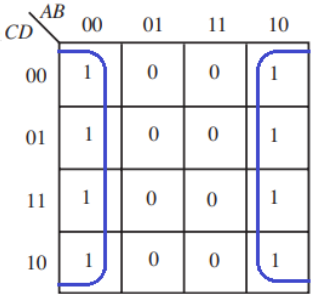


4.7.3 The 4-variebles K – map

1.

$F(A,B) = \bar{B}$ SOP

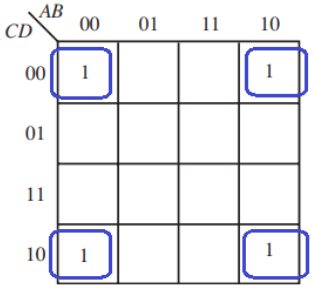
$F(A,B) = B$ POS



2.

$F(A,B) = \bar{B}\bar{D}$ SOP

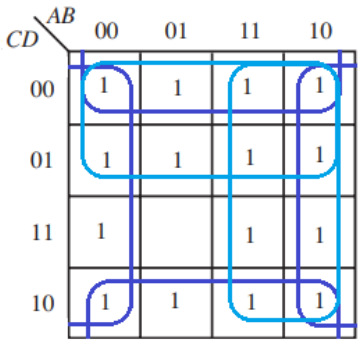
$F(A,B) = (\bar{B}) \cdot (\bar{D})$ POS



3.

$$F(A,B,C) = A + \bar{C} + \bar{B} + \bar{C} \qquad SOP$$

$$F(A,B,C) = A + \bar{C} + \bar{B} + \bar{C} \qquad POS$$



Note:

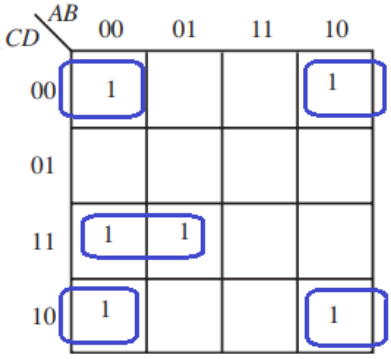
1. *Number of 1's or 0's in one group must be 1, 2, 4, 8, and 16.*
2. *We must take maximum number of 1's or 0's in one group.*

Example: Simplify the following SOP expression on a Karnaugh map:

$$F = \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}CD + A\bar{B}C\bar{D}$$

Sol.

$$F = \bar{B}\bar{D} + \bar{A}CD$$

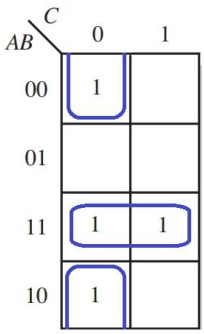


Example: Determine the simply expression by the truth table below using Karnaugh map method.

A	*B*	*C*	*F*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Sol.

$$F = AB + \bar{B}\bar{C}$$



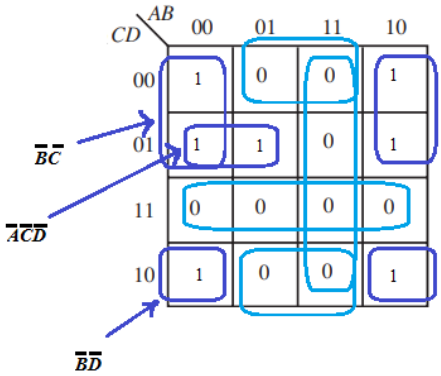
HW: Implement the Logic function specified in the above example.

Example: Simplify the following Boolean function in:

(a) SOP form (b) POS form

$F(A, B, C, D) = \sum(0, 1, 2, 5, 8, 9, 10)$

Sol.



(a) The 1's marked in the map represent all minterm of the function. The cells marked with 0's represent the Maxterm not included in the function and therefore the function will be:

$$F = \bar{B}\,\bar{C} + \bar{B}\bar{D} + \bar{A}\bar{C}D$$

(b) If the squares marked with 0's are combined we obtain the simplified POS form or the complement of *F:*

$$\bar{F} = AB + CD + B\bar{D}$$

Applying DeMorgan's theorem by taking the complement of each side, we obtain the simplified function in POS form:

$$\bar{\bar{F}} = \overline{AB + CD + B\bar{D}}$$

$$\bar{\bar{F}} = (\overline{AB}) \cdot (\overline{CD}) \cdot (\overline{B\bar{D}})$$

$$\bar{\bar{F}} = (\bar{A} + \bar{B}) \cdot (\bar{C} + \bar{D}) \cdot (\bar{B} + \bar{\bar{D}})$$

$$\bar{\bar{F}} = (\bar{A} + \bar{B}) \cdot (\bar{C} + \bar{D}) \cdot (\bar{B} + D)$$

Note: *To use K-map for simplification a function expressed in POS form, follow these rules:*

1. *Take the complement of the function.*
2. *From the results write "0" in the Squares of POS form. Or convert the POS to SOP form, then follow the standard rules used to enter the 1's in the cells of K-map.*

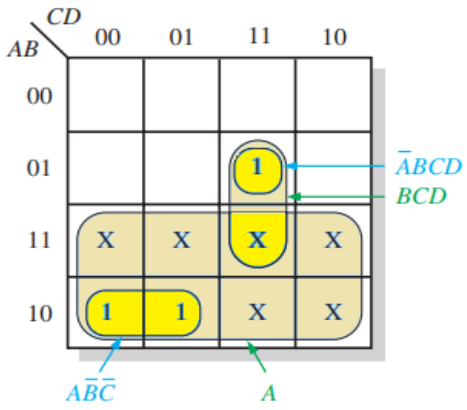
4.7.4 Don't Care Conditions

Sometimes a situation arises in which some input variable combinations are not allowed. For example, recall that in the BCD code, there are six invalid combinations: 1010, 1011, 1100, 1101, 1110, and 1111. Since these unallowed states will never occur in an application involving the BCD code, they can be treated as "don't care" terms with respect to their effect on the output. That is, for these "don't care" terms either a 1 or a 0 may be assigned to the output; it really does not matter since they will never occur.

The "don't care" terms can be used to advantage on the Karnaugh map. The figure below shows that for each "doesn't care" term, an X is placed in the cell. When grouping the 1s, the Xs can be treated as 1s to make a larger grouping or as 0s if they cannot be used to advantage. The larger a group, the simpler the resulting term will be.

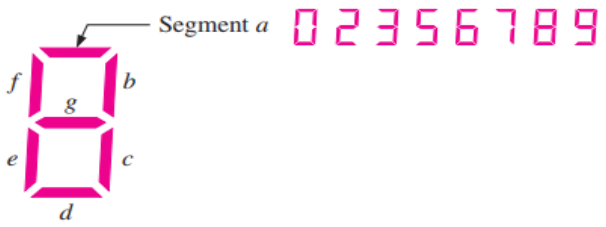
Inputs				Output
A	*B*	*C*	*D*	*Y*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Don't cares



The truth table describes a logic function that has a 1 output only when the BCD code for 7, 8, or 9 is present on the inputs. If the "don't care" are used as 1s, the resulting expression for the function is A + BCD, as indicated in K-map. If the "don't care" is not used as 1s, the resulting expression is ABC + ABCD; so you can see the advantage of using "don't care" terms to get the simplest expression.

Example: In a 7-segment display, each of the seven segments is activated for various digits. For example, segment-a is activated for the digits 0, 2, 3, 5, 6, 7, 8, and 9, as illustrated in the figure below. Since each digit can be represented by a BCD code, derive an SOP expression for segment-a using the variables ABCD and then minimize the expression using a K - map.

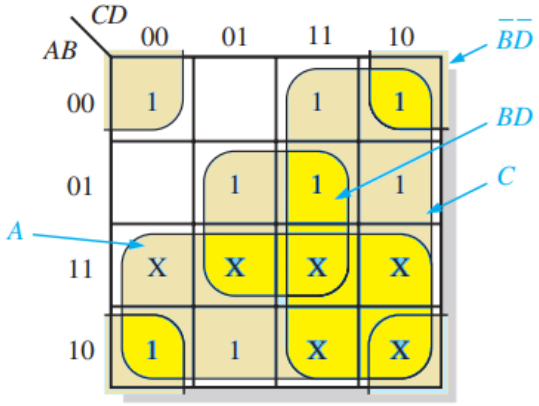


Sol.

The expression for segment-a is:

$$a = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D$$

Each term in the expression represents one of the digits in which segment-a is used. The Karnaugh map minimization is shown in the figure below. X's (don't care) are entered for those states that do not occur in the BCD code.



From the K - map, the minimized expression for segment-a is:

$$a = A + C + BD + \bar{B}\bar{D}$$