takes the high logic level (or the value 1) if and only if the input variables are both at the high logic level (or the value 1). In all other cases, the output is set to the low logic level (or the value 0). Table 2.2 shows the truth table for the AND gate.

Table 2.2. *Truth table. Inputs: A, B; Output: C*

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

2.2.3. *OR gate*

The OR function, which is also called logic addition, is represented by a plus (+).

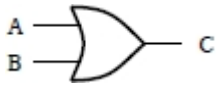

Figure 2.7. *OR gate C = A + B*

Table 2.3. *Truth table. Inputs: A, B; output: C*

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

Figure 2.7 depicts the OR gate, which has two gates. As illustrated by the truth table in Table 2.3, the output takes the logic level 1 if at least one of the two inputs is at the logic level 1, it takes logic level 0 if both the inputs are at the logic level 0.

2.2.4. *XOR gate*

The XOR (exclusive OR) function is represented by a plus within a circle (⊕).

Figure 2.8 depicts the symbolic representation of an XOR gate having two inputs. According to the truth table shown in Table 2.4, the output takes either logic level 1, when only one of the inputs is at logic level 1, or logic level 0, when both inputs are either at logic level 0 or at logic level 1.

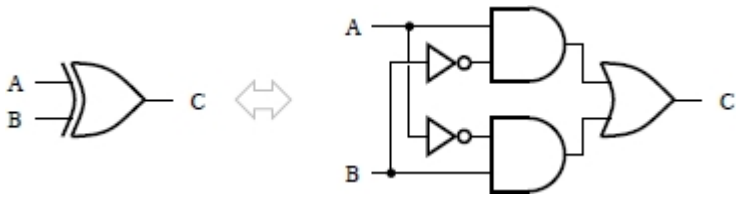

Figure 2.8. *XOR gate (exclusive OR)* $C = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$

Table 2.4. *Truth table. Inputs: A, B; Output: C*

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

2.2.5. *Complementary logic gates*

The NAND (NOT AND), NOR (NOT OR), and XNOR (NOT exclusive OR or inclusive AND) gates are said to be complementary and correspond, respectively, to the AND, OR and XOR gates when followed by a NOT gate. They are characterized by the following logic equations:

– NAND gate: $C = \overline{A \cdot B}$;

– NOR gate: $C = \overline{A + B}$;

– XNOR gate: $C = \overline{A \oplus B} = A \cdot B + \overline{A} \cdot \overline{B} = A \odot B$.

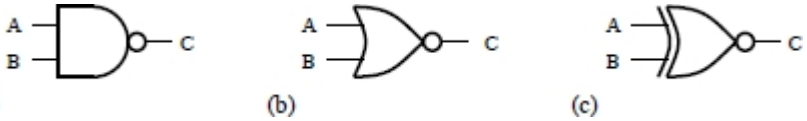

Figure 2.9. *NAND (NOT AND) (a), NOR (NOT OR) (b) and XNOR (NOT exclusive OR) (c) gates*

The NAND and NOR gates are considered to be universal gates. This means that any logic function can be implemented using just NAND or NOR gates. It must be noted that neither the XOR gate nor the XNOR gate is universal.

2.3. Three-state buffer

A three-state buffer works as a signal-controlled switch. An enable signal is used to control whether the input signal is transferred toward the output or isolated from the output, which is then held in a high-impedance state.

The output from the circuit shown in Figure 2.10 can take any of the following three states: high (1), low (0) and high impedance (z).

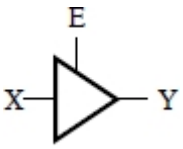

Figure 2.10. *Three-state buffer*

When E = 0, the output is held in the high impedance state.

When E = 1, the output is at the same state as the input (0 or 1).

Table 2.5 shows the truth table for the three-state buffer.

Table 2.5. *Truth table*

E	X	Y
0	x	z
1	0	0
1	1	1

Using the three-state buffer makes it possible to link the outputs of logic gates set up in parallel through a common line.

2.4. Logic function

A logic function is completely defined when, for all combinations of input variables, the function value is defined. The number of these combinations is 2^n for n variables.

A function is incompletely defined when there is at least one combination of input variables for which the logic level is unknown.

When a logic system is implemented, two cases may occur:

– one of the possible combinations will never exist in the normal functioning of the system. This is called a *forbidden condition* and is denoted by - (hyphen);

– one of the combinations exists but can take either the state 0 or 1. This is called the *do not care condition* and is represented by the symbol x or #.

In general, a logic function can thus take four states: 0, 1, x and -.

A function with n variables may be represented by a truth table having $n + 1$ columns and a maximum of 2^n lines.

2.5. The correspondence between a truth table and a logic function

Let $X(A, B, C)$ be a logic function with three variables, which is defined by a truth table.

DEFINITION 2.1.– *Based on the truth table in Table 2.6, the function X may be written as the following sum of products:*

$$X(A,B,C) = \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot B\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot B\cdot\overline{C} + A\cdot B\cdot C \qquad [2.1]$$
$$= \sum m(0,3,4,6,7)$$

Table 2.6. *Truth table (sum of products)*

	A	B	C	X	
0	0	0	0	1	$\leftrightarrow \overline{A}\cdot\overline{B}\cdot\overline{C}$
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\leftrightarrow \overline{A}\cdot B\cdot C$
4	1	0	0	1	$\leftrightarrow A\cdot\overline{B}\cdot\overline{C}$
5	1	0	1	0	
6	1	1	0	1	$\leftrightarrow A\cdot B\cdot\overline{C}$
7	1	1	1	1	$\leftrightarrow A\cdot B\cdot C$

With three variables, $\overline{A}\cdot B\cdot C$ corresponds to a minterm, while $\overline{A}\cdot B$ is not a minterm. A minterm must contain all function variables.

On referring to the truth table in Table 2.7, the product-of-sums form of the function X can be obtained as follows:

$$X(A,B,C) = (A+B+\overline{C}) \cdot (A+\overline{B}+C) \cdot (\overline{A}+B+\overline{C}) = \prod M(1,2,5) \quad [2.2]$$

Table 2.7. *Truth table (product of sums)*

	A	B	C	X	
0	0	0	0	1	
1	0	0	1	0	$\leftrightarrow A+B+\overline{C}$
2	0	1	0	0	$\leftrightarrow A+\overline{B}+C$
3	0	1	1	1	
4	1	0	0	1	
5	1	0	1	0	$\leftrightarrow \overline{A}+B+\overline{C}$
6	1	1	0	1	
7	1	1	1	1	

With three variables, $A+B+\overline{C}$ represents a *maxterm*, while $B+\overline{C}$ is not a *maxterm*.

The *canonical form* corresponds to the Boolean expression of a logic function using only *minterms* or *maxterms*. It is unique for each logic function.

The *complement* of the function *X* is given by:

$$\overline{X}(A,B,C) = \prod M(0,3,4,6,7) = \sum m(1,2,5) \quad [2.3]$$

In the case of four variables, the logic function *X* is considered to be defined by the truth table in Table 2.8.

The function *X* can be written as the following sum of products:

$$X(A,B,C,D) = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C \cdot D + A \cdot B \cdot C \cdot D = \sum m(1,5,7,15) \quad [2.4]$$

Generally, the following relationships exist between the *minterms*, m_i, and the *maxterm*, M_i, of a logic function of *n* variables:

$$\overline{m_i} = M_{2^n-1-i} \quad \text{or} \quad \overline{M_i} = m_{2^n-1-i} \quad (0 \leq i \leq 2^n-1) \quad [2.5]$$

Let *n* be the number of variables of a logic function:

– the sum of all the 2^n *minterms* of a function is equal to 1;

– the product of all the 2^n *maxterms* of a function is equal to 0.

Table 2.8. *Truth table (sum of products)*

	A	B	C	D	X	
0	0	0	0	0	0	
1	0	0	0	1	1	$\leftrightarrow \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D$
2	0	0	1	0	0	
3	0	0	1	1	0	
4	0	1	0	0	0	
5	0	1	0	1	1	$\leftrightarrow \overline{A} \cdot B \cdot \overline{C} \cdot D$
6	0	1	1	0	0	
7	0	1	1	1	1	$\leftrightarrow \overline{A} \cdot B \cdot C \cdot D$
8	1	0	0	0	0	
9	1	0	0	1	0	
10	1	0	1	0	0	
11	1	0	1	1	0	
12	1	1	0	0	0	
13	1	1	0	1	0	
14	1	1	1	0	0	
15	1	1	1	1	1	$\leftrightarrow A \cdot B \cdot C \cdot D$

The product of two different minterms is equal to 0, while the sum of two different maxterms is equal to 1.

2.6. Boolean algebra

Boolean algebra is applied to operations and functions on logic variables.

Let X and Y be logic (or Boolean) functions, whose values can only be 0 or 1. The following properties are verified:

1) commutativity: $X + Y = Y + X$ and $X \cdot Y = Y \cdot X$;

2) associativity: $X + (Y + Z) = (X + Y) + Z$ and $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$;

3) distributivity: $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$ and $(X + Y)(X + Z) = X + Y \cdot Z$;

4) $\overline{X + Y} = \overline{X} \cdot \overline{Y}$ (DeMorgan's theorem – NOR);

5) $\overline{X \cdot Y} = \overline{X} + \overline{Y}$ (DeMorgan's theorem – NAND).

EXAMPLE 2.1.– Implement an XOR gate from NAND logic gates and an XNOR gate from NOR logic gates.

The logic equation for the XOR gate is given by:

$$C = A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B \qquad [2.6]$$

and

$$C = \overline{\overline{C}} = \overline{\overline{A \cdot \overline{B}} \cdot \overline{\overline{A} \cdot B}} \qquad [2.7]$$

Because $A \cdot \overline{B} = A \cdot \overline{A \cdot B}$ and $\overline{A} \cdot B = \overline{A \cdot B} \cdot B$, equation [2.7] takes the form:

$$C = \overline{\overline{A \cdot \overline{A \cdot B}} \cdot \overline{\overline{A \cdot B} \cdot B}} \qquad [2.8]$$

Equation [2.8] can then be implemented as illustrated in Figure 2.11.

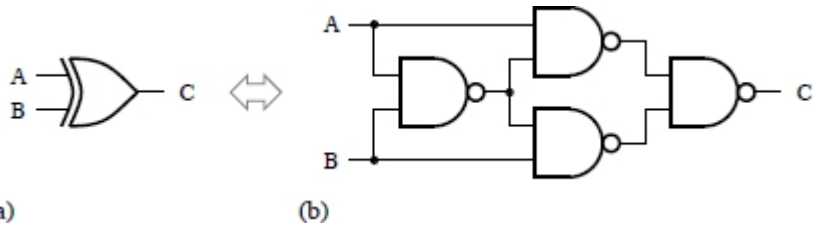

Figure 2.11. *XOR gate: a) symbol; b) construction using NAND gates*

For the XNOR gate, we have:

$$C = \overline{A \oplus B} = \overline{\overline{A} \cdot B + A \cdot \overline{B}} \qquad [2.9]$$

Because

$$\overline{A} \cdot B = \overline{\overline{\overline{A} \cdot B}} = \overline{\overline{A(A+B)}} = \overline{A + \overline{A+B}} \qquad [2.10]$$

and

$$A \cdot \overline{B} = \overline{\overline{A \cdot \overline{B}}} = \overline{\overline{(A+B)\overline{B}}} = \overline{\overline{A+B} + B} \qquad [2.11]$$

equation [2.9] becomes:

$$C = \overline{\overline{A + \overline{A+B}} + \overline{\overline{A+B} + B}} \qquad [2.12]$$

Figure 2.12 depicts the logic circuit corresponding to equation [2.12].

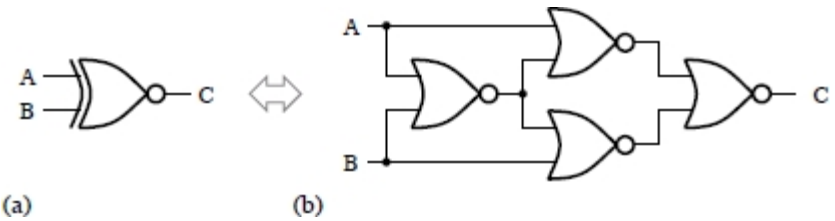

Figure 2.12. *XNOR gate: a) symbol; b) construction using NOR gates*

2.6.1. *Boolean algebra theorems*

Boolean algebra is governed by a certain number of theorems (or properties). The algebraic method of simplification uses properties of Boolean algebra to make it possible to minimize Boolean expressions (or logic functions), thus reducing the material cost for practical implementation.

2.6.1.1. *NOT, AND and OR functions*

Table 2.9 gives the basic properties for the NOT, AND and OR operations.

Table 2.9. *Basic properties for the NOT, AND and OR operations*

NOT	AND	OR
$\overline{0}=1$	$0\cdot X=0$	$0+X=X$
$\overline{1}=0$	$1\cdot X=X$	$1+X=1$
$\overline{\overline{X}}=X$	$X\cdot X=X$	$X+X=X$
	$X\cdot\overline{X}=0$	$X+\overline{X}=1$

In general, for the Boolean functions *X*, *Y* and *Z*, it is possible to establish the following theorems:

– *simplification theorem:*

$$X+X\cdot Y=X$$
$$X(X+Y)=X$$
$$X\cdot Y+X\cdot\overline{Y}=X$$
$$(X+Y)(X+\overline{Y})=X$$

– *absorption theorem:*

$$X+\overline{X}\cdot Y=X+Y$$
$$X(\overline{X}+Y)=X\cdot Y$$

– *factorization and multiplication theorem:*

$$(X+Y)(\overline{X}+Z) = X \cdot Z + \overline{X} \cdot Y$$
$$X \cdot Y + \overline{X} \cdot Z = (X+Z)(\overline{X}+Y)$$

– *consensus theorem:*

$$X \cdot Y + \overline{X} \cdot Z + Y \cdot Z = X \cdot Y + \overline{X} \cdot Z$$
$$(X+Y)(\overline{X}+Z)(Y+Z) = (X+Y)(\overline{X}+Z)$$

SHANNON'S EXPANSION THEOREM.– Let $F(x_0, x_1, \ldots, x_i, \ldots, x_{n-1})$ be a Boolean logic function of n variables. Shannon's expansion theorem can be written as follows:

$$F(x_0, x_1, \ldots, x_i, \ldots, x_{n-1}) = \overline{x_i} \cdot F(x_0, x_1, \ldots, 0, \ldots, x_{n-1}) + x_i \cdot F(x_0, x_1, \ldots, 1, \ldots, x_{n-1}) \quad [2.13]$$

In practice, applying Shannon's expansion theorem makes it possible to decompose a function of five variables, for example, to give rise to two functions of four variables. Thus:

$$F(A,B,C,D,E) = \overline{E} \cdot F(A,B,C,D,0) + E \cdot F(A,B,C,D,1) \quad [2.14]$$

where in fact $F(A, B, C, D, 0)$ and $F(A, B, C, D, 1)$ represent functions of four variables.

By iteratively applying Shannon's expansion theorem, a Boolean function can be expressed from the different values obtained for the different combinations of input variables as given in the truth table.

Table 2.10. *Truth table*

A	B	F(A,B)
0	0	1
0	1	0
1	0	0
1	1	1

For the two-variable function, $F(A, B)$, with the truth table in Table 2.10, we have:

$$\begin{aligned} F(A,B) &= \overline{A} \cdot F(0,B) + A \cdot F(1,B) \\ &= \overline{A}[\overline{B} \cdot F(0,0) + B \cdot F(0,1)] + A[\overline{B} \cdot F(1,0) + B \cdot F(1,1)] \\ &= \overline{A} \cdot \overline{B} \cdot F(0,0) + \overline{A} \cdot B \cdot F(0,1) + A \cdot \overline{B} \cdot F(1,0) + A \cdot B \cdot F(1,1) \end{aligned} \quad [2.15]$$

where $F(0, 0) = 1$, $F(0, 1) = 0$, $F(1, 0) = 0$ and $F(1, 1) = 1$.

2.6.1.2. *XOR (exclusive OR) and XNOR (inclusive AND) functions*

The XOR (OR exclusive) function of two variables is defined by:

$$X \oplus Y = X\overline{Y} + \overline{X} \cdot Y \quad [2.16]$$

Furthermore, we can write:

$$X \oplus Y = \overline{X} \oplus \overline{Y} \quad [2.17]$$

The definition of the XNOR (AND inclusive) function with two variables is given by:

$$X \odot Y = X \cdot Y + \overline{X} \cdot \overline{Y} \quad [2.18]$$

It must be noted that:

$$X \odot Y = \overline{X} \odot \overline{Y} = \overline{X \oplus Y} = \overline{X} \oplus Y = X \oplus \overline{Y} \quad [2.19]$$

Let us consider the Boolean functions, X, Y, and Z. Table 2.11 shows the truth table for the function $X \oplus (Y \cdot Z)$ and $(X \oplus Y)(X \oplus Z)$. We can thus observe that $X \oplus (Y \cdot Z)$ is different from $(X \oplus Y)(X \oplus Z)$.

Table 2.11. *Truth table for* $X \oplus (Y \cdot Z)$ *and* $(X \oplus Y)(X \oplus Z)$

X	Y	Z	$Y \cdot Z$				
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	0	1	0
1	1	0	0	0	1	1	0
1	1	1	1	0	0	0	0

For the functions $X \oplus Y \oplus (X + Y)$ and $X \oplus Y \oplus X + X \oplus Y \oplus Y$, the truth table is represented in Table 2.12. Expressing all combinations of the variables X and Y where each function takes the logic level 1, we have:

$$X \oplus Y \oplus (X + Y) = X \cdot Y \qquad [2.20]$$

and

$$\begin{aligned} X \oplus Y \oplus X + X \oplus Y \oplus Y &= X \cdot Y + X \cdot \overline{Y} + \overline{X} \cdot Y \\ &= X(Y + \overline{Y}) + \overline{X} \cdot Y \\ &= X + \overline{X} \cdot Y \\ &= X + Y \end{aligned} \qquad [2.21]$$

Table 2.12. *Truth table for* $X \oplus Y \oplus (X + Y)$ *and* $X \oplus Y \oplus X + X \oplus Y \oplus Y$

X	Y	$X + Y$	$X \oplus Y$	$X \oplus Y \oplus (X + Y)$	$X \oplus Y \oplus X + X \oplus Y \oplus Y$
0	0	0	0	0	0
0	1	1	1	0	1
1	0	1	1	0	1
1	1	1	0	1	1

Therefore, functions $X \oplus Y \oplus (X + Y)$ and $X \oplus Y \oplus X + X \oplus Y \oplus Y$ are not equal.

The basic properties for the XOR and XNOR operations are given in Table 2.13.

Table 2.13. *Basic properties for the XOR and XNOR operations*

XOR	XNOR
$0 \oplus X = X$	$0 \odot X = \overline{X}$
$1 \oplus X = \overline{X}$	$1 \odot X = X$
$X \oplus X = 0$	$X \odot X = 1$
$X \oplus \overline{X} = 1$	$X \odot \overline{X} = 0$

The following theorems are verified for the Boolean functions X, Y and Z:

– *commutativity:*

$$X \oplus Y = Y \oplus X$$
$$X \odot Y = Y \odot X$$

– *associativity:*

$$X \oplus (Y \oplus Z) = (X \oplus Y) \oplus Z = X \oplus Y \oplus Z$$
$$X \odot (Y \odot Z) = (X \odot Y) \odot Z = X \odot Y \odot Z$$

– *factorization and distributivity:*

$$X \oplus (Y \oplus Z) = (X \oplus Y) \oplus Z = X \oplus Y \oplus Z$$
$$X \odot (Y \odot Z) = (X \odot Y) \odot Z = X \odot Y \odot Z$$

– *absorption:*

$$X \cdot (\overline{X} \oplus Y) = X \cdot Y$$
$$X + (\overline{X} \odot Y) = X + Y$$

– *consensus:*

$$(X \cdot Y) \oplus (\overline{X} \cdot Z) + Y \cdot Z = (X \cdot Y) \oplus (\overline{X} \cdot Z)$$
$$(X + Y) \odot (\overline{X} + Z) \cdot (Y + Z) = (X + Y) \odot (\overline{X} + Z)$$

For two logic functions, X and Y , it may be established that:

– if $X \cdot Y$=0, then $X + Y = X \oplus Y$;

– if $X + Y = 1$, then $X \cdot Y = X \odot y$.

NOTE 2.1.– Boolean algebra theorems (or properties) possess two forms, one deduced from the other by replacing:

– all plus signs (+) with a point (·), and vice versa;

– all circled plus signs $(\oplus)$ with a circled point $(\odot)$, and vice versa;

– any logic level 1 by logic level 0, and vice versa.

EXAMPLE 2.2.– Show that:

$$(X \cdot Y) \oplus (X + Y) = X \oplus Y$$
$$X \odot Y \odot (X \cdot Y) = X + Y$$
$$(\overline{X} + Y) \odot (X \oplus Y) = \overline{X} \cdot Y$$
$$X \cdot Y + Y \cdot Z + X \cdot Z = X \cdot Y \oplus Y \cdot Z \oplus X \cdot Z$$

Using Boolean algebra theorems (or properties), we can write:

$$\begin{aligned}(X\cdot Y)\oplus(X+Y) &= X\cdot Y(\overline{X+Y})+\overline{X\cdot Y}(X+Y)\\ &= X\cdot Y(\overline{X+Y})+\overline{X\cdot Y}(X+Y)\\ &= X\cdot Y(\overline{X}\cdot\overline{Y})+(\overline{X}+\overline{Y})(X+Y)\\ &= X\cdot\overline{Y}+\overline{X}\cdot Y\\ &= X\oplus Y\end{aligned} \quad [2.22]$$

$$\begin{aligned}X\odot Y\odot(X\cdot Y) &= (X\odot Y)(X\cdot Y)+(\overline{X\odot Y})(\overline{X\cdot Y})\\ &= (X\cdot Y+\overline{X}\cdot\overline{Y})(X\cdot Y)+(X\oplus Y)(\overline{X}+\overline{Y})\\ &= X\cdot Y+(X\cdot\overline{Y}+\overline{X}\cdot Y)(\overline{X}+\overline{Y})\\ &= X\cdot Y+X\cdot\overline{Y}+\overline{X}\cdot Y\\ &= X(Y+\overline{Y})+\overline{X}\cdot Y\\ &= X+\overline{X}\cdot Y\\ &= X+Y\end{aligned} \quad [2.23]$$

$$\begin{aligned}(\overline{X}+Y)\odot(X\oplus Y) &= (\overline{X}+Y)\odot(X\cdot\overline{Y}+\overline{X}\cdot Y)\\ &= (\overline{X}+Y)\cdot(X\cdot\overline{Y}+\overline{X}\cdot Y)\\ &\quad \text{because}\quad (\overline{X}+Y)+(X\cdot\overline{Y}+\overline{X}\cdot Y)=1\\ &= \overline{X}\cdot Y\end{aligned} \quad [2.24]$$

$$\begin{aligned}X\cdot Y+Y\cdot Z+X\cdot Z &= X\cdot Y(Z+\overline{Z})+(X+\overline{X})Y\cdot Z+X\cdot(Y+\overline{Y})Z\\ &= X\cdot Y\cdot Z+X\cdot Y\cdot\overline{Z}+\overline{X}\cdot Y\cdot Z+X\cdot\overline{Y}\cdot Z\\ &= X\cdot Y\cdot Z+X\cdot Y\cdot\overline{Z}\oplus(\overline{X}\cdot Y\cdot Z+X\cdot\overline{Y}\cdot Z)\\ &\quad \text{because}\quad (X\cdot Y\cdot\overline{Z})(\overline{X}\cdot Y\cdot Z+X\cdot\overline{Y}\cdot Z)=0\\ &= X\cdot Y(Z+\overline{Z})\oplus(Y\oplus X)Z\\ &= X\cdot Y\oplus Y\cdot Z\oplus X\cdot Z\end{aligned} \quad [2.25]$$

2.6.2. *Karnaugh maps*

A semi-graphical method, which is based on the use of Karnaugh maps, is more appropriate for the simplification of more complex Boolean expressions.

A Karnaugh map, like a truth table, provides a representation of logic functions. It is composed of a certain number of squares or cells, each of which is reserved for a term (minterm or maxterm) of a logic function. Figure 2.13 shows Karnaugh maps for a three-variable function and Figure 2.14 presents Karnaugh maps for a four-variable function. The variables can be represented in two ways. On each map, the combination of the variables are placed in accordance with the order

of Gray's encoding such that adjacent terms are in the neighboring cells or in the cells at map ends.

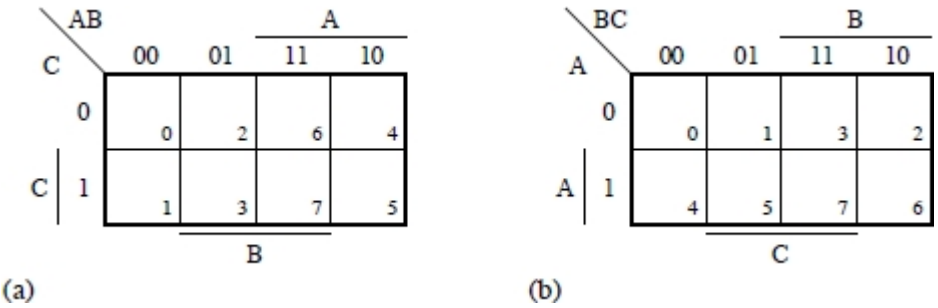

Figure 2.13. *Three-variable Karnaugh map*

Though Karnaugh maps can be used to reduce any logic function, they become difficult to manipulate when the number of variables exceeds six.

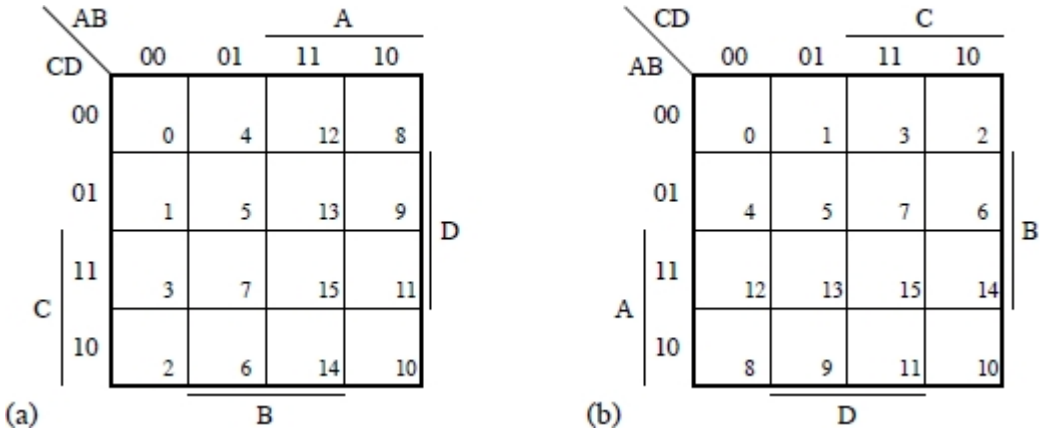

Figure 2.14. *Four-variable Karnaugh map*

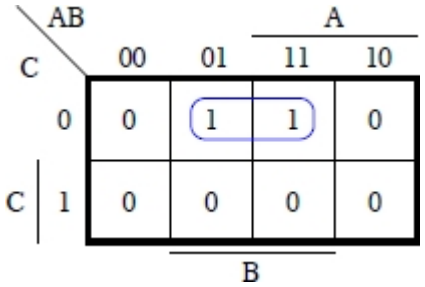

Figure 2.15. *Duad:* $X = \overline{A} \cdot B \cdot \overline{C} + A \cdot B \cdot \overline{C} = B \cdot \overline{C}$

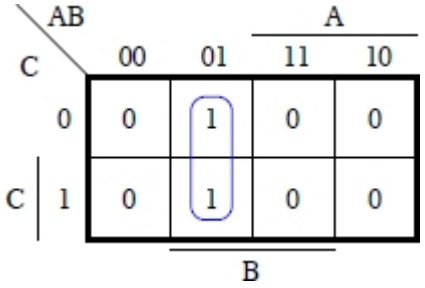

Figure 2.16. *Duad:* $X = \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C = \overline{A} \cdot B$

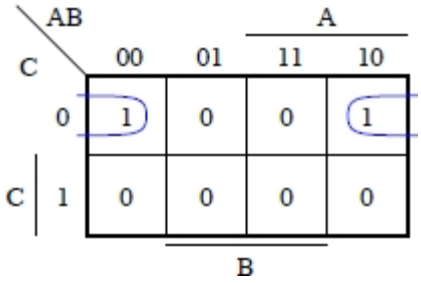

Figure 2.17. *Duad:* $X = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} = \overline{B} \cdot \overline{C}$

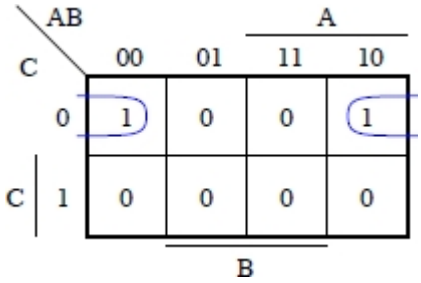

Figure 2.18. *Quad:* $X = \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C = \overline{B}$

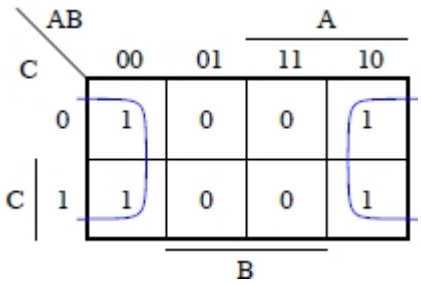

Figure 2.19. *Quad:* $X = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C + A \cdot B \cdot C + A \cdot \overline{B} \cdot C = C$

Using a Karnaugh map, the simplification of a logic function is carried out by grouping the adjacent cells that contain 1s. The number of cells in a group must be a power of 2, or of the form 2^n (n = 1, 2, 3, …):

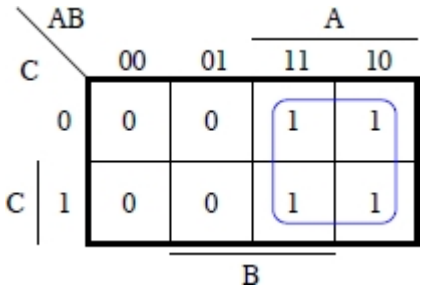

Figure 2.20. *Quad:* $X = A \cdot B \cdot C + A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot \overline{C} = A$

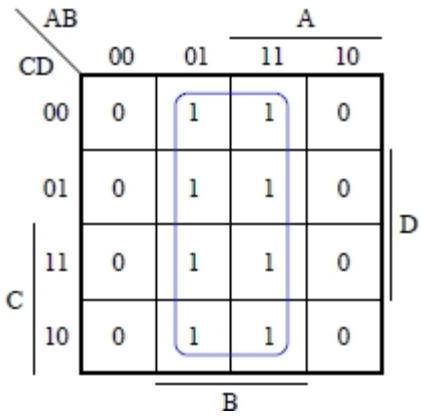

Figure 2.21. *Octad:* $X = B$

– duad of adjacent 1s: the variable that is both complemented and non-complemented can be eliminated (see Figures 2.15–2.17);

– quad of adjacent 1s: two variables that are both complemented and non-complemented can be eliminated (see Figures 2.18–2.20);

– octad of adjacent 1s: three variables that are both complemented and non-complemented can be eliminated (see Figures 2.21–2.24).

Only those variables that hold the same logic state in all the cells of a group appear in the simplified expression.

NOTE 2.2.– In the case of several possibilities for grouping the cells of a Karnaugh map, the minimization procedure can yield more than one logic expression.

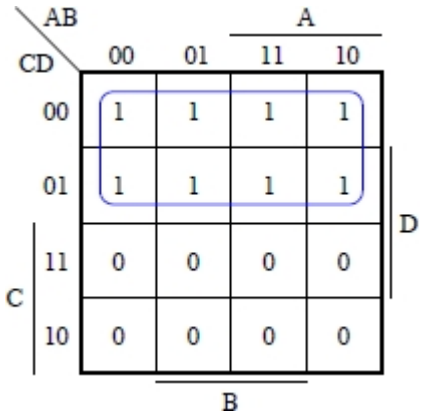

Figure 2.22. *Octad:* $X = \overline{C}$

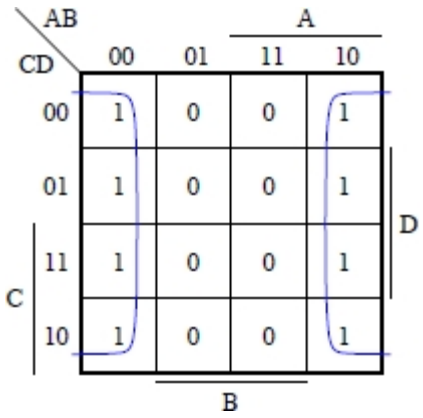

Figure 2.23. *Octad:* $X = \overline{B}$

The *minterms* that are not necessary for the desired application may be used to improve the simplification of the logic circuit. They are considered as *don't care terms*.

EXAMPLE 2.3.– Simplify the following expressions:

1) $X(A,B,C) = \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{B}\cdot C + \overline{A}\cdot B.$

2) $X(A,B,C) = \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot\overline{B}\cdot C + A\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot\overline{C}.$ We shall suppose that the input condition $A\cdot\overline{B}\cdot C$ does not affect the logic level of the function *X* (*do not care condition*).

3) $$X(A,B,C,D) = \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D + \overline{A}\cdot B\cdot\overline{C}\cdot D + \overline{A}\cdot B\cdot C\cdot D + \overline{A}\cdot B\cdot C\cdot\overline{D} + A\cdot B\cdot C\cdot\overline{D} + A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D} + A\cdot\overline{B}\cdot\overline{C}\cdot D + A\cdot\overline{B}\cdot C\cdot\overline{D}.$$

4) $$X(A,B,C,D) = \overline{A}\cdot\overline{B}\cdot C\cdot D + \overline{A}\cdot B\cdot\overline{C}\cdot\overline{D} + \overline{A}\cdot B\cdot\overline{C}\cdot D + \overline{A}\cdot B\cdot C\cdot D + \overline{A}\cdot B\cdot C\cdot\overline{D} + A\cdot B\cdot\overline{C}\cdot\overline{D} + A\cdot B\cdot\overline{C}\cdot D$$

5) $$X(A,B,C,D) = \overline{A}\cdot\overline{B}\cdot C\cdot D + \overline{A}\cdot B\cdot\overline{C}\cdot\overline{D} + \overline{A}\cdot B\cdot\overline{C}\cdot D + \overline{A}\cdot B\cdot C\cdot D + \overline{A}\cdot B\cdot C\cdot\overline{D} + A\cdot B\cdot\overline{C}\cdot\overline{D} + A\cdot B\cdot\overline{C}\cdot D$$

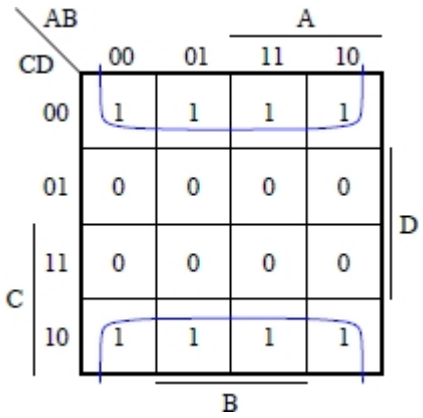

Figure 2.24. *Octad:* $X = \overline{D}$

Using Karnaugh maps represented in Figures 2.25–2.30, different solutions can be obtained.

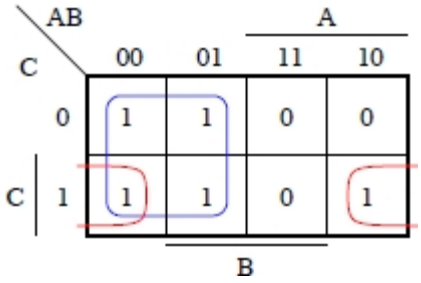

Figure 2.25. *Example 2.1(1):* $X = \overline{A} + \overline{B} \cdot C$

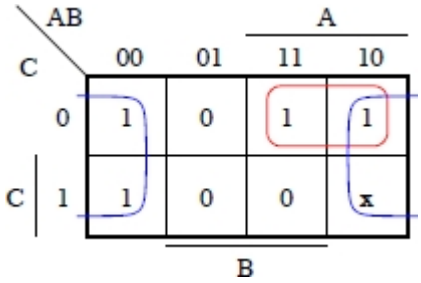

Figure 2.26. *Example 2.1(2):* $X = \overline{B} + A \cdot \overline{C}$

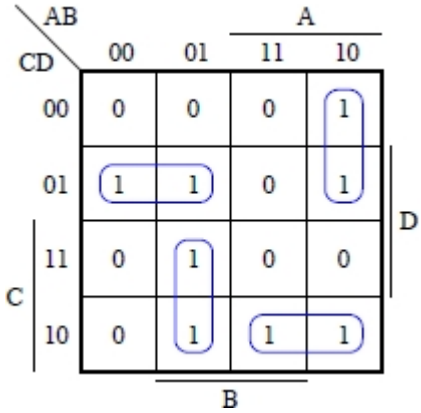

Figure 2.27. *Example 2.1(3) (case 1):* $X = \overline{A} \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot C \cdot \overline{D}$

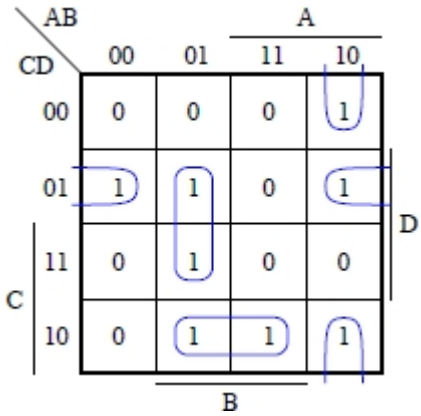

Figure 2.28. *Example 2.1(3) (case 2):* $X = \overline{A} \cdot B \cdot D + B \cdot C \cdot \overline{D} + \overline{B} \cdot \overline{C} \cdot D + A \cdot \overline{B} \cdot \overline{D}$

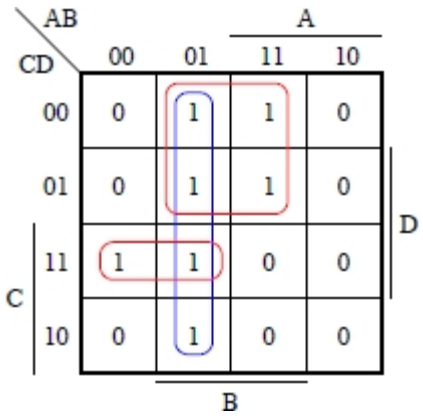

Figure 2.29. *Example 2.1(4):* $X = \overline{A} \cdot B + B \cdot \overline{C} + \overline{A} \cdot C \cdot D$

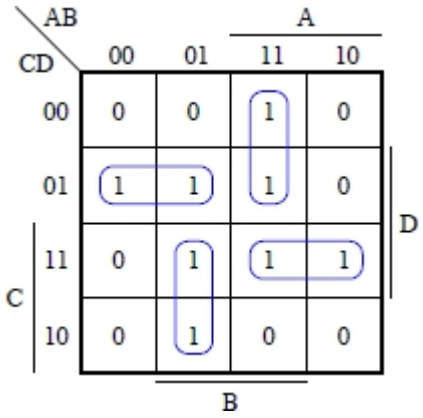

Figure 2.30. *Example 2.1(5):* $X = A \cdot B \cdot \overline{C} + \overline{A} \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C + A \cdot C \cdot D$

It should be noted that some logic functions can have many minimal expressions.

EXAMPLE 2.4.– Simplify the logic function *F* in the two following cases:

a) $F(A, B, C) = \sum m(1, 3, 4, 7)$;

b) $F(A, B, C) = \sum m(1, 3, 4, 7) + x(2, 5)$, where the *don't care terms* are represented by *x*.

The minimal expressions of the function *F* may be obtained from the Karnaugh maps represented in Figures 2.31 and 2.32.

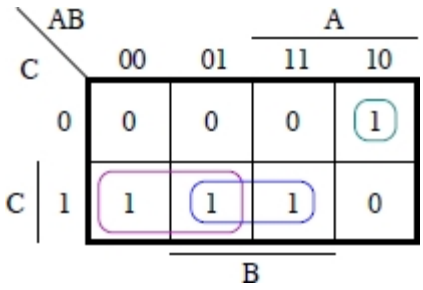

Figure 2.31. *Example 2.2(a):* $F = A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C + B \cdot C$

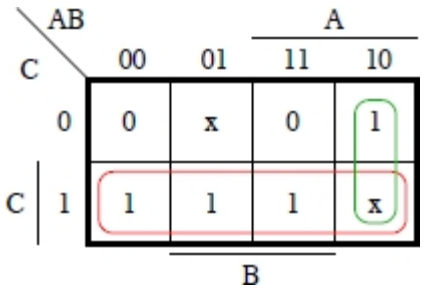

Figure 2.32. *Example 2.2(b):* $F = A \cdot \overline{B} + C$

It should be noted that a *don't care term* is taken into account only if it can contribute to the simplification of the logic function.

2.6.3. *Simplification of logic functions with multiple outputs*

The simplification of logic functions with multiple outputs can be carried out in four steps:

1) Write the functions to be simplified in the sum of products;

2) The minterms being represented by *m* and the *don't care terms* by x, form the products of sums in a systematic manner in accordance with the following rules of the AND operation:

$$m_i \cdot m_i = m_i$$

$$m_i \cdot m_j = 0 \ (i \neq j)$$

$$m_i \cdot \mathrm{x}_i = m_i$$

$$\mathrm{x}_i \cdot \mathrm{x}_i = \mathrm{x}_i$$

$$m_i \cdot \mathrm{x}_j = \mathrm{x}_i \cdot \mathrm{x}_j = 0 \ (i \neq j);$$

3) Draw up a table containing the terms common to different functions;

4) Based on the Karnaugh map for each function, group the common terms and then simplify the remaining terms.

However, when a group of common terms is part of a larger group of 2^n (n = 1, 2, 3) terms, this group is only selected if it yields the simplest logic expression.

EXAMPLE 2.5.– Propose a circuit that implements the following circuit:

$$F(A,B,C,D) = \sum m(0,2,3,4,6,7,10,11)$$
$$G(A,B,C,D) = \sum m(0,4,8,9,10,11,12,13)$$

Taking into account the common terms (see the Karnaugh maps shown in Figures 2.33 and 2.34), we can obtain the circuit in Figure 2.35 that consists of six logic gates with a total of 16 inputs.

By independently simplifying the two functions, we arrive at:

$$F(A,B,C,D) = \overline{A}\cdot C + \overline{A}\cdot \overline{D} + \overline{B}\cdot C$$
$$G(A,B,C,D) = A\cdot \overline{C} + \overline{C}\cdot \overline{D} + A\cdot \overline{B}$$

In this case, eight logic gates with a total of 18 inputs are needed for the implementation of the functions F and G.

2.6.4. *Factorization of logic functions*

To reduce hardware implementation costs, it is often necessary to find terms common to several functions.

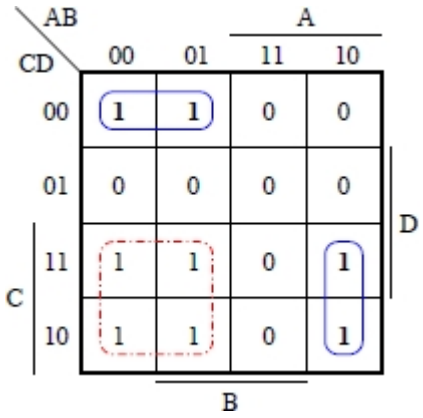

Figure 2.33. *Function F* $F = \overline{A}\cdot C + A\cdot \overline{B}\cdot C + \overline{A}\cdot \overline{C}\cdot \overline{D}$

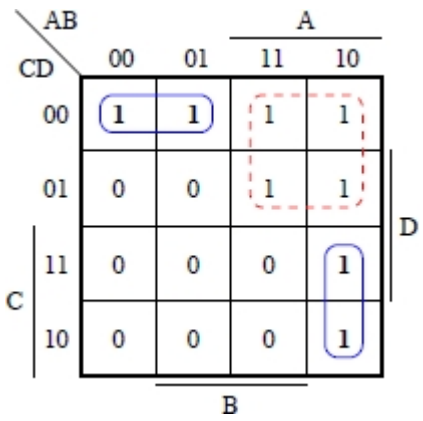

Figure 2.34. *Function G* $G = A\cdot \overline{C} + A\cdot \overline{B}\cdot C + \overline{A}\cdot \overline{C}\cdot \overline{D}$

Factorize (or decompose) the following Boolean expressions to yield the term $C + D$:

$$F(A,B,C,D) = A \cdot C + A \cdot D + B \cdot \overline{C} \cdot \overline{D}$$
$$G(A,B,C,D) = A \cdot B \cdot C + A \cdot B \cdot D + \overline{A} \cdot \overline{C} \cdot \overline{D} + \overline{B} \cdot \overline{C} \cdot \overline{D}$$

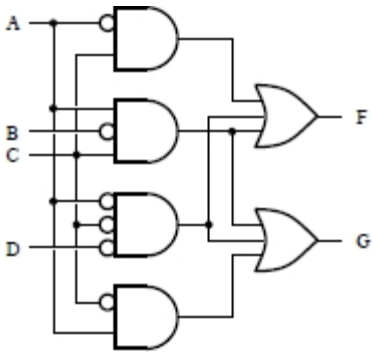

Figure 2.35. *Logic circuits for the implementation of F and G*

The factorization (or the decomposition) of functions F and G can be carried out as follows:

$$F = A \cdot C + A \cdot D + B \cdot \overline{C} \cdot \overline{D} \quad [2.26]$$
$$= A(C + D) + B \cdot \overline{C} \cdot \overline{D}$$

$$= A(C + D) + B(\overline{C + D}) \quad [2.27]$$

and

$$G = A \cdot B \cdot C + A \cdot B \cdot D + \overline{A} \cdot \overline{C} \cdot \overline{D} + \overline{B} \cdot \overline{C} \cdot \overline{D} \quad [2.28]$$
$$= A \cdot B(C + D) + \overline{A}(\overline{C + D}) + \overline{B}(\overline{C + D})$$

$$= A \cdot B(C + D) + \overline{AB}(\overline{C + D}) \quad [2.29]$$

2.7. Multi-level logic circuit implementation

Combinational logic circuits are generally designed using two-level logic networks. They arise directly from the sum-of-product expressions and are characterized by a high speed of operation (or fast response time) if the number of inputs is such that the load limit for each logic gate is not exceeded (or if the input fan-in specification is satisfied).

The design of multi-level circuits is based on the factorization