

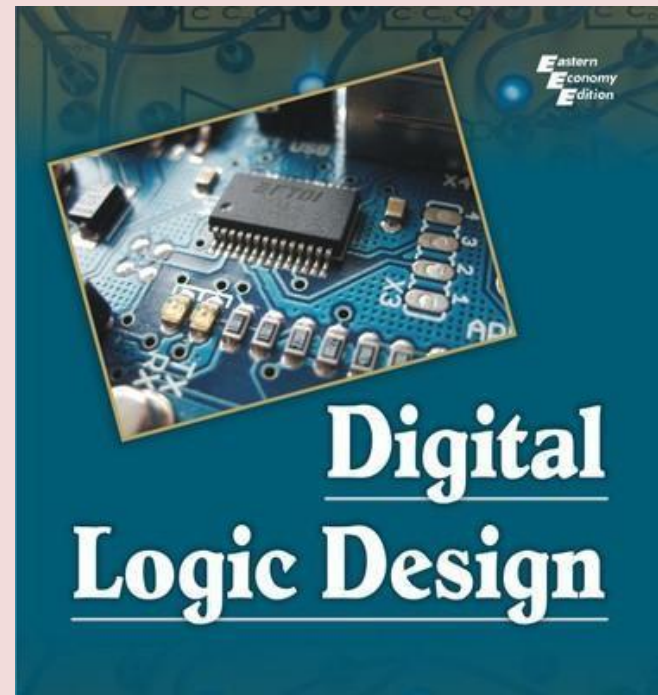
BOOLEAN ALGEBRA AND LOGIC GATES

Lec 4

Dr. Safa Saad

Software Department

2018-2019



Introduction

- Binary logic deals with variables that have two discrete values: **1** for **TRUE** and **0** for **FALSE**.
- A **simple switching circuit** containing active elements such as a **diode** and **transistor** can demonstrate the binary logic, which can either be **ON** (switch closed) or **OFF** (switch open).
- The **switching functions** can be expressed with **Boolean equations**. Complex Boolean equations can be simplified by a new kind of algebra, which is popularly called **Switching Algebra** or **Boolean Algebra**.

BASIC DEFINITIONS

Boolean algebra, like any other mathematical system, may be defined with :

- set of elements, set of operators, and number of assumptions and postulates.

Various algebraic structures are formulated on the basis of the most common postulates, which are described as follows:

1. Closer:

A set is closed with respect to a binary operator if, for every pair of elements of S , the binary operator specifies a rule for obtaining a unique element of S . For example, the set of natural numbers $N = \{1, 2, 3, 4, \dots\}$ is said to be closed with respect to the binary operator plus $(+)$ by the rules of arithmetic addition, since for any $X, Y \in N$ we obtain a unique element $Z \in N$ by the operation $X + Y = Z$.

However, note that the set of natural numbers is **not closed** with respect to the binary operator **minus** ($-$) by the rules of arithmetic subtraction because for:

$1 - 2 = -1$, where -1 is not of the set of natural numbers.

2. Associative Law: الارتباط

A binary operator $*$ on a set S is said to be associated whenever $(A*B)*C = A*(B*C)$ for all $A, B, C \in S$.

3. Commutative Law: التبادل

A binary operator $*$ on a set S is said to be commutative Whenever $A*B = B*A$ for all $A, B \in S$.

4. Identity Element:

A set S is to have an identity element with respect to a binary operation $*$ on S , if there exists an element $E \in S$ with the property $E * A = A * E = A$.

Example: The element 0 is an identity element with respect to the binary operator $+$ on the set of integers $I = \{.... -4, -3, -2, -1, 0, 1, 2, 3, 4,\}$ as $A + 0 = 0 + A = A$.

Similarly, the element 1 is the identity element with respect to the binary operator \times as $A \times 1 = 1 \times A = A$.

5. Inverse:

If a set S has the identity element E with respect to a binary operator $*$, there exists an element $B \in S$, which is called the inverse, for every $A \in S$, such that $A * B = E$.

Example: In the set of integers I with $E = 0$, the inverse of an element A is $(-A)$ since

$$A + (-A) = 0.$$

The multiplication inverse of A is $1/A$, i.e., $A \cdot 1/A = 1$.

6. Distributive Law:

The only distributive law applicable is that of (\cdot) over $+$

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

If summarized, for the field of real numbers, the operators and postulates have the following meanings:

The binary operator $+$ defines **addition**.

The **additive identity** is 0 .

The **additive inverse** defines **subtraction**.

The binary operator $(.)$ defines **multiplication**.

The **multiplication identity** is 1 .

The **multiplication inverse** of A is $1/A$, defines **division** *i.e.*, $A \cdot 1/A = 1$.

The **only distributive** law applicable is that of $(.)$ over $+$ whereas $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$

The following Huntington postulates are satisfied for the definition of Boolean algebra on a set of elements S together with two binary operators (+) and (.):

1. (a) Closer with respect to the operator (+).
(b) Closer with respect to the operator (.).
2. (a) An identity element with respect to + is designated by 0 *i.e.*, $A + 0 = 0 + A = A$.
(b) An identity element with respect to . is designated by 1 *i.e.*, $A.1 = 1.A = A$.
3. (a) Commutative with respect to (+), *i.e.*, $A + B = B + A$.
(b) Commutative with respect to (.), *i.e.*, $A.B = B.A$.
4. (a) (.) is distributive over (+), *i.e.*, $A.(B+C) = (A.B) + (A.C)$.
(b) (+) is distributive over (.), *i.e.*, $A + (B.C) = (A + B).(A + C)$.
5. For every element $A \in S$, there exists an element $A' \in S$ (called the **complement** of A) such that:
 $A + A' = 1$ and $A . A' = 0$.

TWO-VALUED BOOLEAN ALGEBRA

- Two-valued Boolean algebra is defined on a set of only two elements, $S = \{0,1\}$, with rules for two binary operators (+) and (.) and inversion or complement as shown in the following operator tables :

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

Figure 2-1

A	B	$A.B$
0	0	0
0	1	0
1	0	0
1	1	1

Figure 2-2

A	A'
0	1
1	0

Figure 2-3

- These rules are exactly the same for as the logical OR, AND, and NOT operations, respectively.

1. **Closure** is obviously valid, as from the table it is observed that the result of each operation is either 0 or 1 and $0, 1 \in S$.

2. From the tables, we can see that:

(i) $0 + 0 = 0$ $0 + 1 = 1 + 0 = 1$

(ii) $1 \cdot 1 = 1$ $0 \cdot 1 = 1 \cdot 0 = 0$

which verifies the two identity elements 0 for (+) and 1 for (.)

3. **The commutative** laws are confirmed by the symmetry of binary operator tables.

4. The distributive laws of $(.)$ over $(+)$ i.e.,

$A . (B+C) = (A . B) + (A . C)$, and $(+)$ over

$(.)$ i.e., $A + (B . C) = (A+B) . (A+C)$ can be shown to be applicable with the help of the truth tables considering all the possible values of A, B, and C .

(a) Operator $(.)$ over $(+)$

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

(b) Operator (+) over (.)

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

Figure 2-5

(c) $A + A' = 1$, since $0 + 0' = 1$ and $1 + 1' = 1$.

(d) $A \cdot A' = 0$, since $0 \cdot 0' = 0$ and $1 \cdot 1' = 0$.

BASIC PROPERTIES AND THEOREMS OF BOOLEAN ALGEBRA

- **DeMorgan's Theorem**

Two theorems that were proposed by DeMorgan play important parts in Boolean algebra.

The first theorem states that the complement of a product is equal to the sum of the complements. That is, if the variables are A and B, then:

$$(A.B)' = A' + B'$$

The second theorem states that the complement of a sum is equal to the product of the complements. In equation form, this can be expressed as:

$$(A + B)' = A' . B'$$

BASIC PROPERTIES AND THEOREMS OF BOOLEAN ALGEBRA

DeMorgan's theorems are applicable to **any number of variables**. Such as for three variables A, B, and C, the equations are:

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

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

(a) $A + 0 = A$	(b) $A.1 = A$
(a) $A + A' = 1$	(b) $A.A' = 0$
(a) $A + A = A$	(b) $A.A = A$
(a) $A + 1 = 1$	(b) $A.0 = 0$
$(A')' = A$	
(a) $A + B = B + A$	(b) $A.B = B.A$
(a) $A + (B + C) = (A + B) + C$	(b) $A.(B.C) = (A.B).C$
(a) $A(B + C) = A.B + A.C$	(b) $A + B.C = (A + B).(A + C)$
(a) $(A + B)' = A'.B'$	(b) $(A.B)' = A' + B'$
(a) $A + A.B = A$	(b) $A.(A + B) = A$

The following is the complete list of postulates and theorems useful for two-valued Boolean algebra.

Postulate 2	$(a) A + 0 = A$	$(b) A.1 = A$
Postulate 5	$(a) A + A' = 1$	$(b) A.A' = 0$
Theorem 1	$(a) A + A = A$	$(b) A.A = A$
Theorem 2	$(a) A + 1 = 1$	$(b) A.0 = 0$
Theorem 3, Involution	$(A')' = A$	
Theorem 3, Commutative	$(a) A + B = B + A$	$(b) A.B = B.A$
Theorem 4, Associative	$(a) A + (B + C) = (A + B) + C$	$(b) A.(B.C) = (A.B).C$
Theorem 4, Distributive	$(a) A(B + C) = A.B + A.C$	$(b) A + B.C = (A + B).(A + C)$
Theorem 5, DeMorgan	$(a) (A + B)' = A'.B'$	$(b) (A.B)' = A' + B'$
Theorem 6, Absorption	$(a) A + A.B = A$	$(b) A.(A + B) = A$

Boolean Functions

A Boolean function is an expression formed with binary variables, the two binary operators **AND** and **OR**, one unary operator **NOT**.

The **value of a function may be 0 or 1**.

- For example, if a **Boolean function** is expressed algebraically as: $F = AB'C$

Then the value of F will be 1, when $A = 1$, $B = 0$, and $C = 1$.
For other values of A , B , C the value of F is 0.

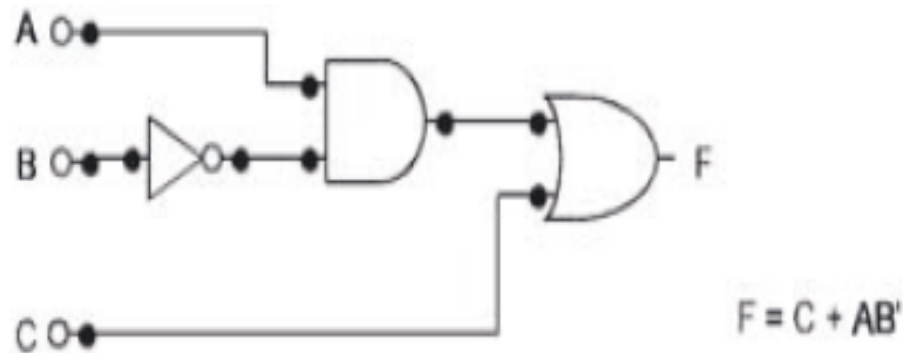
- **Boolean functions** can also be represented by truth tables. For an n number of variables, **$2n$** combinations of 1s and 0s

Boolean Functions

Boolean function $F = AB + C$ truth table can be written as :

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

A **Boolean function** from an algebraic expression can be realized to a logic diagram composed of logic gates.



Canonical And Standard Forms الصيغ القانونية والقياسية

Logical functions are generally expressed in terms of different combinations of logical variables with their true forms as well as the complement forms.

logic function can be expressed in the following forms:

- (i) Sum of the Products (SOP)
- (ii) Product of the Sums (POS)

Product Term: *AND function is referred to as a product term or standard product.* For example, ABC' is a product term.

Sum Term: *An OR function is referred to as a sum term.*

For example, $A + B + C'$ is a sum term.

y

Sum of Products (SOP): The logical sum of two or more logical product terms is referred to as a sum of products expression. It is basically an OR operation on AND operated variables.

For example, $Y = AB + BC + AC$ or $Y = A'B + BC + AC'$ are sum of products expressions.

Product of Sums (POS): Similarly, the logical product of two or more logical sum terms is called a product of sums expression. It is an AND operation on OR operated variables.

For example, $Y = (A + B + C)(A + B' + C)(A + B + C')$ or $Y = (A + B + C)(A' + B' + C')$ are product of sums expressions.

Standard form: The standard form of the Boolean function is when it is expressed in sum of the products or product of the sums fashion. The examples stated above, like $Y = AB + BC + AC$ or $Y = (A + B + C)(A + B' + C)(A + B + C')$ are the standard forms.

However, Boolean functions are also sometimes expressed in nonstandard forms like $F = (AB + CD)(A'B' + C'D')$, which is neither a sum of products form nor a product of sums form. However, the same expression can be converted to a standard form with help of various Boolean properties, as:

$$F = (AB + CD)(A'B' + C'D') = A'B'CD + ABC'D'$$

Minterm

A product term containing all n variables of the function in either true or complemented form is called the minterm. Each minterm is obtained by an AND operation of the variables in their true form or complemented form.

For a two-variable function, four different combinations are possible, such as, $A'B'$, $A'B$, AB' , and AB . These product terms are called the fundamental products or standard products or minterms. In the minterm, a variable will have the value **1** if it is in **true** or uncomplemented form, whereas, it contains the value **0** if it is in complemented form. For three variables function, eight minterms are possible as listed in the following table in Figure 9. So, if the number of variables is n , then the possible number of minterms is 2^n .

A	B	C	Minterm
0	0	0	$A'B'C'$
0	0	1	$A'B'C$
0	1	0	$A'BC'$
0	1	1	$A'BC$
1	0	0	$AB'C'$
1	0	1	$AB'C$
1	1	0	ABC'
1	1	1	ABC

Minterm

Canonical Sum of Product Expression: When a Boolean function is expressed as the logical sum of all the minterms from the rows of a truth table, for which the value of the function is 1, it is referred to as the *canonical sum of product expression*.

The **canonical sum of products form** of a logic function can be **obtained** by using the following procedure:

1. Check each term in the given logic function. Retain if it is a minterm, continue to examine the next term in the same manner.
2. Examine for the variables that are missing in each product which is not a minterm. If the missing variable in the minterm is X, **multiply** that minterm with **(X+X')**.
3. Multiply all the products and discard the redundant terms.

Here are some examples to explain the above procedure.

Example 1 Obtain the canonical sum of product form of the function $F(A, B) = A + B$

Solution: The given function contains two variables A and B. The variable B is missing from the first term of the expression and the variable A is missing from the second term of the expression. Therefore, the first term is to be multiplied by $(B + B')$ and the second term is to be multiplied by $(A + A')$ as demonstrated below.

$$\begin{aligned} F(A, B) &= A + B \\ &= A.1 + B.1 \\ &= A(B + B') + B(A + A') \\ &= AB + AB' + AB + A'B \\ &= AB + AB' + A'B \quad (\text{as } AB + AB = AB) \end{aligned}$$

Hence the canonical sum of the product expression of the given function is $F(A, B) = AB + AB' + A'B$.

Maxterm

A sum term containing all n variables of the function in either true or complemented form is called the maxterm. Each maxterm is obtained by an OR operation of the variables in their true form or complemented form. Four different combinations are possible for a two-variable function, such as, $A' + B'$, $A' + B$, $A + B'$, and $A + B$. These sum terms are called the standard sums or maxterms. Note that, in the maxterm, a variable will have the value **0**, if it is in **true** or **uncomplemented** form, whereas, it contains the value **1**, if it is in **complemented** form. Like minterms, for a three-variable function, eight maxterms are also possible as listed in the following table in Figure 10.

A	B	C	Maxterm
0	0	0	$A + B + C$
0	0	1	$A + B + C'$
0	1	0	$A + B' + C$
0	1	1	$A + B' + C'$
1	0	0	$A' + B + C$
1	0	1	$A' + B + C'$
1	1	0	$A' + B' + C$
1	1	1	$A' + B' + C'$

Figure 10

Canonical Product of Sum Expression: When a Boolean function is expressed as the logical product of all the maxterms from the rows of a truth table, for which the value of the function is 0, it is referred to as the *canonical product of sum expression*.

The **canonical product of sums** form of a logic function can be **obtained** by using the following procedure.

1. Check each term in the given logic function. Retain it if it is a maxterm, continue to examine the next term in the same manner.
2. Examine for the variables that are missing in each sum term that is not a maxterm. If the missing variable in the maxterm is X, **add** that maxterm with **(X.X')**.
3. Expand the expression using the properties and postulates as described earlier and discard the redundant terms.

Example 3 Obtain the canonical product of the sum form of the following function.

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

Solution: In the above three-variable expression, C is missing from the first term, A is missing from the second term, and B is missing from the third term. Therefore, CC' is to be added with first term, AA' is to be added with the second, and BB' is to be added with the third term. This is shown below.

$$\begin{aligned} F(A, B, C) &= (A + B') (B + C) (A + C') \\ &= (A + B' + 0) (B + C + 0) (A + C' + 0) \\ &= (A + B' + CC') (B + C + AA') (A + C' + BB') \\ &= (A + B' + C) (A + B' + C') (A + B + C) (A' + B + C) (A + B + C') (A + B' + C') \\ &\quad [\text{using the distributive property, as } X + YZ = (X + Y)(X + Z)] \\ &= (A + B' + C) (A + B' + C') (A + B + C) (A' + B + C) (A + B + C') \\ &\quad [\text{as } (A + B' + C') (A + B' + C') = A + B' + C'] \end{aligned}$$

Hence the canonical product of the sum expression for the given function is

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

Deriving a Sum of Products (SOP) Expression from a Truth Table

The sum of products (SOP) expression of a Boolean function can be obtained from its truth table summing or performing OR operation of the product terms corresponding to the combinations containing a function value of 1.

Inputs			Output Y	Product terms	Sum terms
A	B	C			
0	0	0	0		$A + B + C$
0	0	1	0		$A + B + C'$
0	1	0	1	$A'BC'$	
0	1	1	0		$A + B' + C'$
1	0	0	1	$AB'C'$	
1	0	1	1	$AB'C$	
1	1	0	1	ABC'	
1	1	1	0		$A' + B' + C'$

Figure 11

The final sum of products expression (SOP) for the output Y is derived by summing or performing an OR operation of the four product terms as shown : $Y = A'BC' + AB'C' + AB'C + ABC'$




Deriving a Product of Sums (POS) Expression from a Truth Table




The product of sums (POS) expression of a Boolean function can also be obtained from its truth table by a similar procedure. Here, an AND operation is performed on the sum terms corresponding to the combinations containing a function value of 0.

Inputs			Output	Product terms	Sum terms
A	B	C			
0	0	0	0		$A + B + C$
0	0	1	0		$A + B + C'$
0	1	0	1	$A'BC'$	
0	1	1	0		$A + B' + C'$
1	0	0	1	$AB'C'$	
1	0	1	1	$AB'C$	
1	1	0	1	ABC'	
1	1	1	0		$A' + B' + C'$

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



DIGITAL LOGIC GATES

Name	Graphic Symbol	Algebraic Function	Truth Table		
AND		$F = AB$	A	B	F
			0	0	0
			0	1	0
			1	0	0
			1	1	1
OR		$F = A + B$	A	B	F
			0	0	0
			0	1	1
			1	0	1
			1	1	1
Inverter or NOT		$F = A'$		A	F
				0	1
				1	0

Buffer		$F = A$		A	F
				0	0
				1	1
NAND		$F = (AB)'$	A	B	F
			0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR		$F = (A + B)'$	A	B	F
			0	0	1
			0	1	0
			1	0	0
			1	1	0

DIGITAL LOGIC GATES

//

Exclusive-OR (XOR)		$F = AB' + A'B$ $= A \oplus B$	A	B	F
			0	0	0
			0	1	1
			1	0	1
			1	1	0
Equivalence Or Exclusive-NOR (XNOR)		$F = AB + A'B'$ $= A \odot B$	A	B	F
			0	0	1
			0	1	0
			1	0	0
			1	1	1

DIGITAL LOGIC GATES

AND and **OR** gates are both commutative and associative.

For the **AND** function, $AB = BA$ -commutative

And $(AB)C = A(BC) = ABC$. -associative

For the **OR** function, $A + B = B + A$ -commutative

And $(A + B) + C = A + (B + C)$. -associative

These indicate that the gate inputs can be interchanged and these functions can be extended to three or more variables very simply as shown in Figures 13(a) and 13(b).



Figure 13(a)

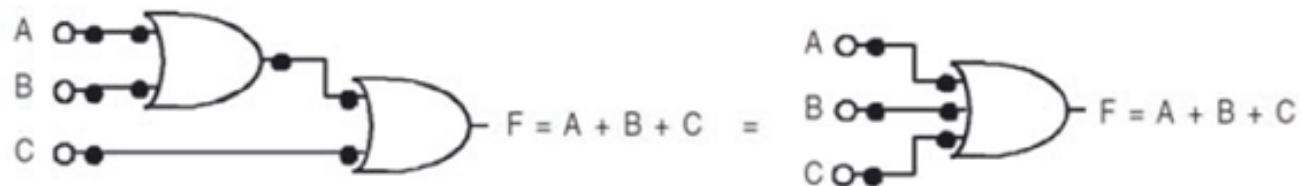


Figure 13(b)

DIGITAL LOGIC GATES

The **NAND** and **NOR** functions are the complements of **AND** and **OR** functions respectively. They are commutative, but not associative. So these functions can not be extended to multiple input variables very simply. However, these gates can be extended to multiple inputs with slightly modified functions as shown in Figures 14(a) and 14(b) below.

For **NAND** function, $(AB)' = (BA)'$.

-commutative

But, $((AB)'C)' \neq (A(BC)')'$.

-does not follow associative property.

As $((AB)'C)' = (AB) + C'$ and $(A(BC)')' = A' + BC$.

Similarly, for **NOR** function, $((A + B)' + C)' \neq (A + (B + C)')'$.

As, $((A + B)' + C)' = (A + B)C' = AC' + BC'$.

And $(A + (B + C)')' = A'(B + C) = A'B + A'C$.

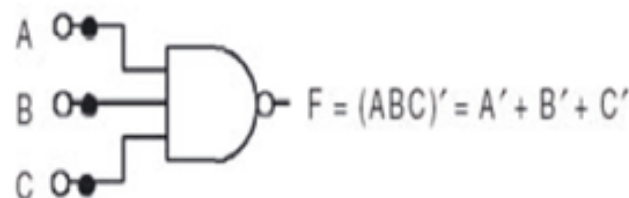
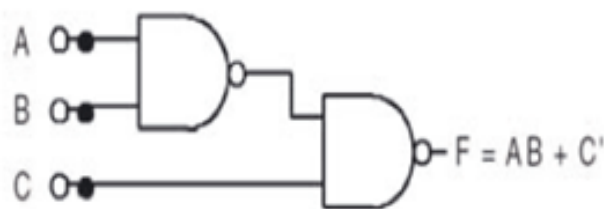
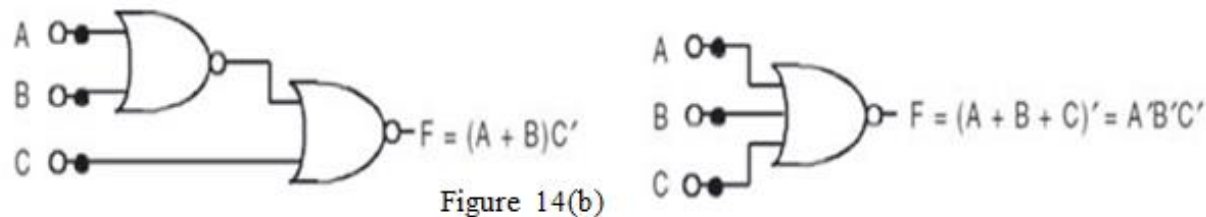
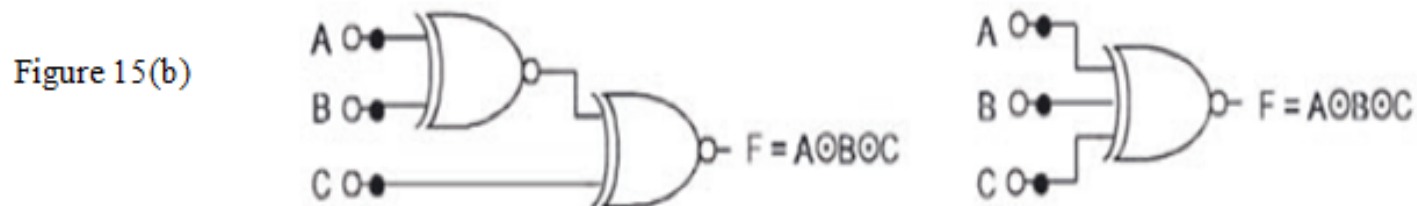
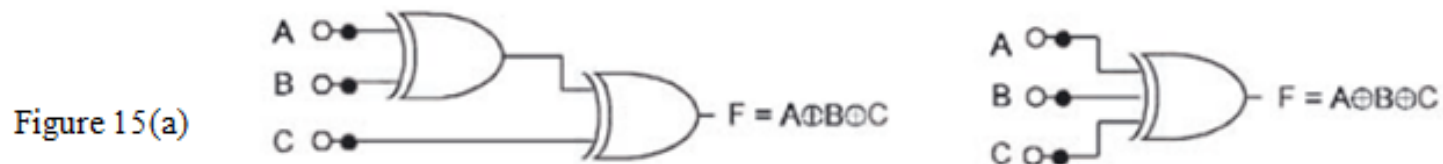


Figure 14(a)

DIGITAL LOGIC GATES



The **Exclusive-OR** gates and **equivalence** gates both have commutative and associative properties, and they can be extended to multiple input variables. For a multiple-input **Ex-OR (XOR)** gate output is low when *even* numbers of 1s are applied to the inputs, and when the number of 1s is *odd* the output is logic 0. **Equivalence** gate or **XNOR** gate is equivalent to XOR gate followed by NOT gate and hence its logic behavior is opposite to the XOR gate. However, multiple-input exclusive-OR and equivalence gates are uncommon in practice. Figures 15(a) and 15(b) describe the extension to multiple-input exclusive-OR and equivalence gates.



Thanks ...