

Chapter 2

Boolean Algebra and Logic Gates

Boolean Algebra and Logic Gates

2.1 Basic Definitions

2.2 Axiomatic Definition of Boolean Algebra

2.3 Basic Theorems and Properties of Boolean Algebra

2.4 Boolean Functions

2.5 Canonical and Standard Forms

2.6 Other Logic Operations

2.7 Digital Logic Gates

2.8 IC Digital Logic Families

2.1 Basic Definitions

1.Closure A set S 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 .

Example:-

The set of natural numbers $N = \{1, 2, 3, 4, \dots\}$ is closed with respect to the binary operator plus (+) by the rules of arithmetic addition, since for any $a, b \in N$ we obtain a unique $c \in N$ by the operation $a + b = c$.

2.1 Basic Definitions

2. Associative law

- A binary operator $*$ on a set S is said to be associative whenever:
 $(x * y) * z = x * (y * z)$ for all $x, y, z \in S$

3. Commutative law

- A binary operator $*$ on a set S is said to be commutative whenever: $x * y = y * x$ for all $x, y \in S$

2.1 Basic Definitions

4. Identity element

A set S is said to have an identity element with respect to a binary operation $*$ on S if there exists an element $e \in S$ with the property: $e * x = x * e = x$ for every $x \in S$

Example:

The element 0 is an identity element with respect to operation $+$ on the set of integers $I = \{ \dots, -3, -2, -1, 0, 1, 2, 3, \dots \}$ since: $x + 0 = 0 + x = x$ for any $x \in I$. The set of natural numbers N has no identity element since 0 is excluded from the set.

2.1 Basic Definitions

5. Inverse

- A set S having the identity element e with respect to a binary operator $*$ is said to have an inverse whenever, for every $x \in S$, there exists an element $y \in S$ such that: $x * y = e$ Example: In the set of integer I with $e = 0$, the inverse of an element a is $(-a)$ since $a + (-a) = 0$.

6. Distributive law

- If $*$ and \bullet are two binary operators on a set S , $*$ is said to be distributive over \bullet whenever: $x * (y \bullet z) = (x * y) \bullet (x * z)$

2.2 Axiomatic Definition of Boolean Algebra

Boolean algebra is an algebraic structure defined on a set of elements B together with two binary operators $+$ and \bullet provided the following (Huntington) postulates are satisfied:

1. (a) Closure with respect to the operator $+$.
(b) Closure with respect to the operator \bullet .
2. (a) An identity element with respect to $+$, designated by 0 : $x + 0 = 0 + x = x$.
(b) An identity element with respect to \bullet , designated by 1 : $x \bullet 1 = 1 \bullet x = x$.

2.2 Axiomatic Definition of Boolean Algebra

3. (a) Commutative with respect to $+$: $x + y = y + x$.
(b) Commutative with respect to \cdot : $x \cdot y = y \cdot x$.
4. (a) \cdot is distributive over $+$: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$.
(b) $+$ is distributive over \cdot : $x + (y \cdot z) = (x + y) \cdot (x + z)$.
5. For every element $x \in B$, there exists an element $x' \in B$ (called the complement of x) such that: (a) $x + x' = 1$ and (b) $x \cdot x' = 0$.
6. There exists at least two elements $x, y \in B$ such that $x \neq y$.

2.2.2 Two-Valued Boolean Algebra

A two-valued Boolean algebra is defined on a set of two elements, $B = \{0, 1\}$, with rules for the two binary operators $+$ and \cdot as shown in the following operator tables (the rule for the complement operator is for verification of postulate 5):

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

2.2.2 Two-Valued Boolean Algebra

These rules are exactly the same as the AND, OR, and NOT operations, respectively, defined in Table 1-6. We must now show that the Huntington postulates are valid for the set $B = \{0, 1\}$ and the two binary operators defined above.

1. Closure is obvious from the tables since the result of each operation is either 1 or 0 and $1, 0 \in B$.
2. From the tables we see that: (a) $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$
(b) $1 \cdot 1 = 1$, $1 \cdot 0 = 0 \cdot 1 = 0$ which establishes the two identity elements 0 for $+$ and 1 for \cdot as defined by postulate 2.

2.2.2 Two-Valued Boolean Algebra

3. The commutative laws are obvious from the symmetry of the binary operator tables.

4. (a) The distribute law $x \bullet (y + z) = (x \bullet y) + (x \bullet z)$ can be shown to hold true from the operator tables by forming a truth table of all possible values of x , y , and z . For each combination, we derive $x \bullet (y + z)$ and show that the value is the same as $(x \bullet y) + (x \bullet z)$.

(b) The distributive law of $+$ over \bullet can be shown to hold true by means of a truth table similar to the one above.

2.2.2 Two-Valued Boolean Algebra

x y z	$y + z$	$x \cdot (y + z)$	$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
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

2.2.2 Two-Valued Boolean Algebra

5. From the complement table it is easily shown that: (a) $x + x' = 1$, since $0 + 0' = 0 + 1 = 1$ and $1 + 1' = 1 + 0 = 1$ (b) $x \cdot x' = 0$, since $0 \cdot 0' = 0 \cdot 1 = 0$ and $1 \cdot 1' = 1 \cdot 0 = 0$ which verifies postulate 5.

6. Postulate 6 is satisfied because the two-valued Boolean algebra has two distinct elements 1 and 0 with $1 \neq 0$.

2.3 Basic Theorems and Properties of Boolean Algebra

2.3.1 Duality

2.3.2 Basic Theorems

2.3.3 Operator Precedence

2.3.4 Venn Diagram

2.3.1 Duality

This important property of Boolean algebra is called the duality principle. It states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged. In a two-valued Boolean algebra, the identity elements and the elements of the set B are the same: 1 and 0. The duality principle has many applications. If the dual of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's

2.3.1 Duality

Table 2–1 Postulates and theorems of Boolean algebra

Postulate 2	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
Postulate 5	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Theorem 1	(a) $x + x = x$	(b) $x \cdot x = x$
Theorem 2	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Postulate 3, commutative	(a) $x + y = y + x$	(b) $xy = yx$
Theorem 4, associative	(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$
Postulate 4, distributive	(a) $x(y + z) = xy + xz$	(b) $x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a) $(x + y)' = x' y'$	(b) $(xy)' = x' + y'$
Theorem 6, absorption	(a) $x + xy = x$	(b) $x(x + y) = x$

THEOREM 1(a): $x + x = x$.

$x + x = (x + x) \cdot 1$	by postulate: 2(b)
$= (x + x)(x + x')$	5(a)
$= x + xx'$	4(b)
$= x + 0$	5(b)
$= x$	2(a)

2.3.2 Basic Theorems

THEOREM 1(b): $x \bullet x = x$.

$$x \bullet x = xx + 0$$

by postulate: 2(a)

$$= xx + xx'$$

5(b)

$$= x(x + x')$$

4(a)

$$= x \bullet 1$$

5(a)

$$= x$$

2(b)

2.3.2 Basic Theorems

THEOREM 2(a): $x + 1 = 1$.

$x + 1 = 1 \cdot (x + 1)$	by postulate: 2(b)
$= (x + x') (x + 1)$	5(a)
$= x + x' \cdot 1$	4(b)
$= x + x'$	2(b)
$= 1$	5(a)

THEOREM 2 (b): $x \cdot 0 = 0$ by duality.

2.3.2 Basic Theorems

2.3.2 Basic Theorems

THEOREM 3: $(x')' = x$.

From postulate 5, we have $x + x' = 1$ and $x \bullet x' = 0$, which defines the complement of x . The complement of x' is x and is also $(x')'$. Therefore, since the complement is unique, we have that

$$(x')' = x.$$

2.3.2 Basic Theorems

THEOREM 6(a): $x + xy = x$.

$$\begin{aligned}x + xy &= x \cdot 1 + xy \\&= x (1 + y) \\&= x (y + 1) \\&= x \cdot 1 \\&= x\end{aligned}$$

by postulate 2(b)

by postulate 4(a)

by postulate 3(a)

by theorem 2(a)

by postulate 2(b)

x	y	xy	$x + xy$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

THEOREM 6(b): $x(x + y) = x$ by duality.

x	y	$x + y$	$(x + y)'$	x'	y'	$x' y'$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

The first De Morgan's theorem $(x + y)' = x' y'$

2.3.3 Operator Precedence

The operator precedence for evaluating Boolean expressions is

- Parentheses
- NOT
- AND
- OR.

2.3.3 Operator Precedence (Example)

EXAMPLE 2-1: Using basic Boolean theorem prove:

(a) $(x + y)(x + z) = x + yz$

Soln: $(x + y)(x + z)$

$$= x \cdot x + x \cdot z + y \cdot x + y \cdot z \quad \text{since}$$

$$x \cdot x = x$$

$$= x + xz + yx + yz$$

$$= x(1 + z) + yx + yz$$

$$\text{since } (1 + z) = 1$$

$$= x + yx + yz$$

$$= x(1 + y) + yz$$

$$\text{since } (1 + y) = 1$$

$$= x + yz \text{ (Proved)}$$

2.3.4 Venn Diagram

A helpful illustration that may be used to visualize the relationships among the variables of a Boolean expression is the Venn diagram.

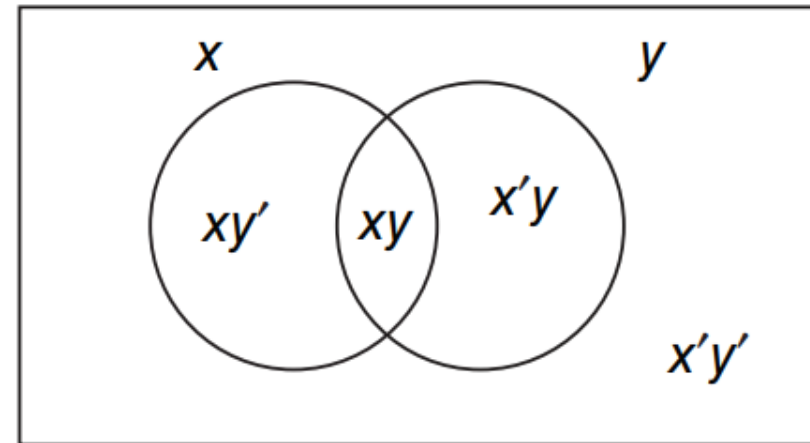
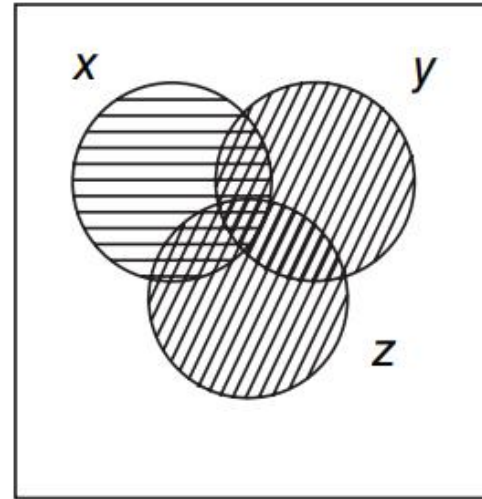
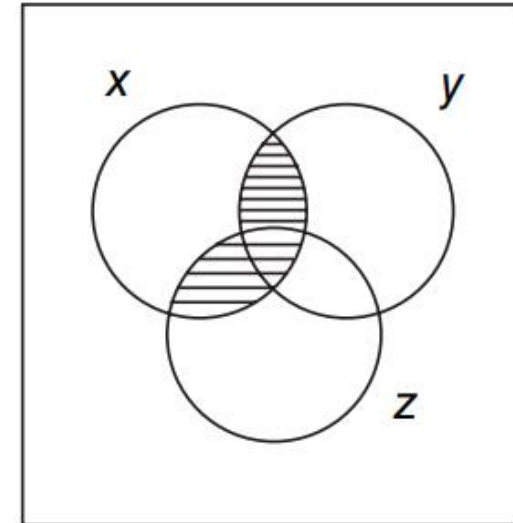


Figure 2.1 Venn diagram for two variables

Distributive law
 $x(y + z) = xy + xz$
(Venn Diagram)



$x(y + z)$



$xy + xz$

Figure 2.3 Venn diagram illustration of the distributive law

2.4 Boolean Functions

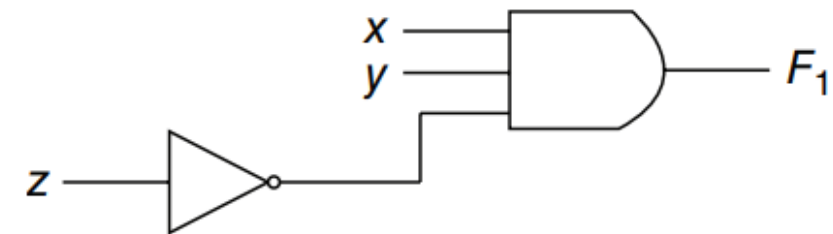
Table 2-2 Truth tables for $F_1 = xyz'$, $F_2 = x + y'z$, $F_3 = x'y'z + x'yz + xy'$, and $F_4 = xy' + x'z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

2.4 Boolean Functions (continued)

Table 2-2 Truth tables for $F_1 = xyz'$, $F_2 = x + y'z$, $F_3 = x'y'z + x'yz + xy'$, and $F_4 = xy' + x'z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

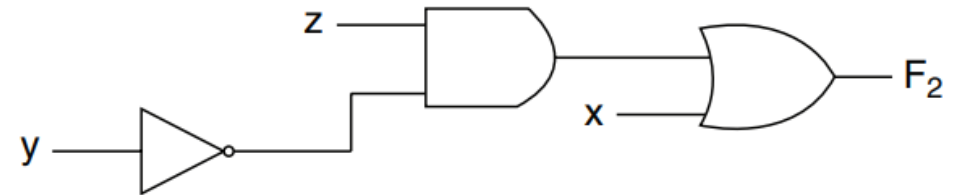


(a) $F_1 = xyz'$

2.4 Boolean Functions (continued)

Table 2-2 Truth tables for $F_1 = xyz'$, $F_2 = x + y'z$, $F_3 = x'y'z + x'yz + xy'$, and $F_4 = xy' + x'z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

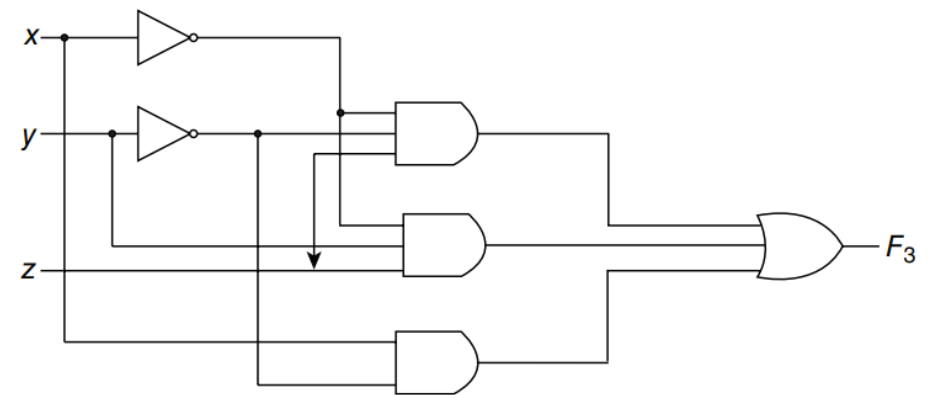


(b) $F_2 = x + y'z$

2.4 Boolean Functions (continued)

Table 2-2 Truth tables for $F_1 = xyz'$, $F_2 = x + y'z$, $F_3 = x'y'z + x'yz + xy'$, and $F_4 = xy' + x'z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

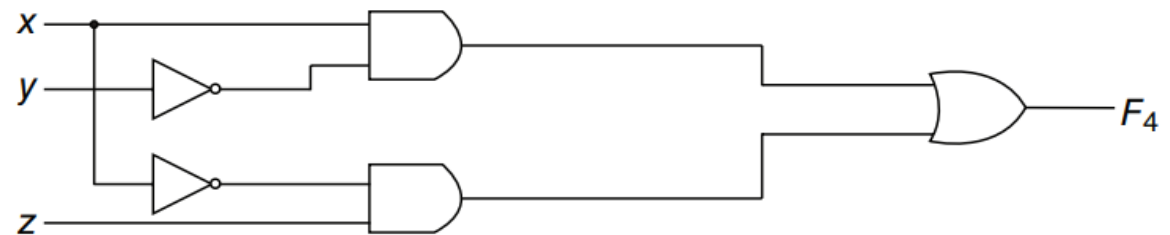


(c) $F_3 = x'y'z + x'yz + xy'$

2.4 Boolean Functions (continued)

Table 2-2 Truth tables for $F_1 = xyz'$, $F_2 = x + y'z$, $F_3 = x'y'z + x'yz + xy'$, and $F_4 = xy' + x'z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

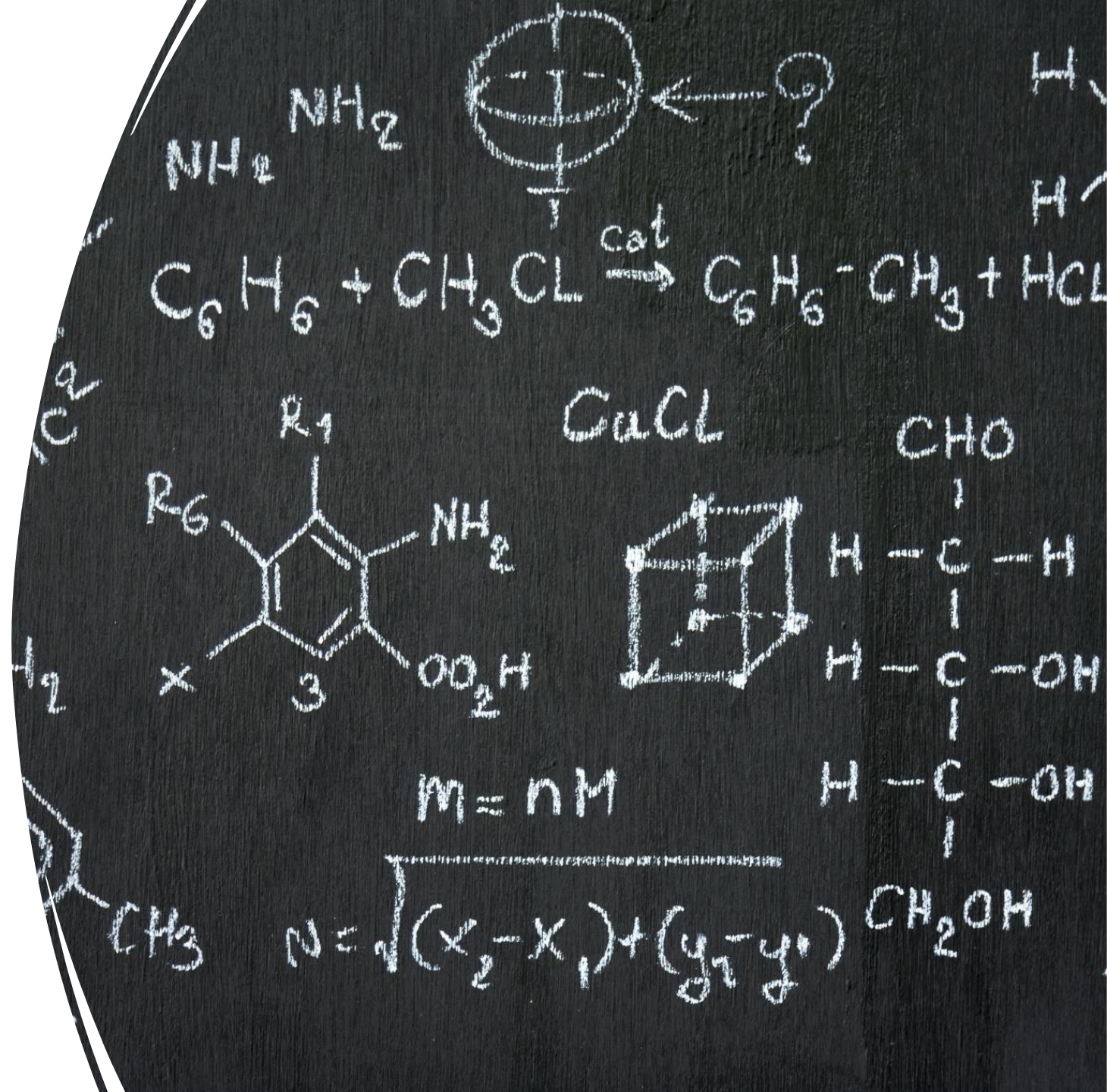


(d) $F_4 = xy' + x'z$

2.4.1 Algebraic Manipulation

Example: Simplify the following Boolean functions to a minimum number of literals.

$$\begin{aligned} 1. \quad & x + x' y \\ &= (x + x')(x + y) \\ &= 1 \bullet (x + y) \\ &= x + y \end{aligned}$$



2.4.1

Algebraic Manipulation

Example: Simplify the following Boolean functions to a minimum number of literals.

$$\begin{aligned} 2. \quad & x(x' + y) \\ &= xx' + xy \\ &= 0 + xy \\ &= xy \end{aligned}$$

2.4.1

Algebraic Manipulation

Example: Simplify the following Boolean functions to a minimum number of literals.

$$\begin{aligned} 3. \quad & x' y' z + x' y z + x y' \\ &= x' z(y' + y) + x y' \\ &= x' z + x y' \end{aligned}$$


2.4.1

Algebraic Manipulation

Example: Simplify the following Boolean functions to a minimum number of literals.

$$\begin{aligned} 4. \quad & xy + x'z + yz \\ &= xy + x'z + yz(x + x') \\ &= xy + x'z + xyz + x'yz \\ &= xy(1 + z) + x'z(1 + y) \\ &= xy + x'z \end{aligned}$$

Example: Simplify the following Boolean functions to a minimum number of literals.



2.4.1 Algebraic Manipulation

$$5. \quad (x + y) (x' + z) (y + z)$$

$= (x + y) (x' + z)$ by duality from function 4.

2.4.2 Complement of a Function

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

$$= (A + X)' \text{ let } B + C = X$$

$$= A' X' \text{ by theorem 5(a) (De Morgan)}$$

$$= A' \bullet (B + C)' \quad \text{substitute } B + C = X$$

$$= A' \bullet (B' C') \quad \text{by theorem 5(a) (De Morgan)}$$

$$= A' B' C' \quad \text{by theorem 4(b) (associative)}$$

2.4.2 Complement of a Function (Example)

EXAMPLE 2-3: Find the complement of the functions $F1 = x' yz' + x' y' z$ and $F2 = x (y'z' + yz)$.

Applying De Morgan's theorem as many times as necessary, the complements are obtained as follows:

$$\begin{aligned} F' 1 &= (x' yz' + x' y' z)' \\ &= (x' yz')'(x' y' z)' \\ &= (x + y' + z)(x + y + z') \end{aligned}$$

$$\begin{aligned} F' 2 &= [x (y' z' + yz)]' \\ &= x' + (y' z' + yz)' \\ &= x' + (y' z')' \bullet (yz)' \\ &= x' + (y + z)(y' + z') \end{aligned}$$

2.5 Canonical and Standard Forms

2.5.1 Minterms and Maxterms

2.5.2 Sum of Minterms

2.5.3 Product of Maxterms

2.5.4 Conversion between Canonical Forms

2.5.5 Standard Forms

2.5.1 Minterms and Maxterms

Table 2-3 Minterms and maxterms for three binary variables

x	y	z	Minterms		Maxterms	
			Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

$$\begin{aligned}
 f_1 &= x'y'z + xy'z' + xyz \\
 &= m_1 + m_4 + m_7 \\
 f_2 &= x'yz + xy'z + xyz' + xyz \\
 &= m_3 + m_5 + m_6 + m_7
 \end{aligned}$$

Table 2-4 Functions of three variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2.5.2 Sum of Minterms

EXAMPLE 2-5: Express the Boolean function $F = A + B'C$ in a sum of minterms. The function has three variables A , B , and C . The first term A is missing two variables; therefore:

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

This is still missing one variable:

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

The second term $B'C$ is missing one variable:

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

Combining all terms, we have:

$$\begin{aligned} F &= A + B'C \\ &= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C \end{aligned}$$

But $AB'C$ appears twice, and according to theorem 1 ($x + x = x$), it is possible to remove one of them. Rearranging the minterms in ascending order, we finally obtain:

$$\begin{aligned} F &= A'B'C + AB'C' + AB'C + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \end{aligned}$$

2.5.3 Product of Maxterms

EXAMPLE 2-6: Express the Boolean function $F = xy + x'z$ in a product of maxterm form. First convert the function into OR terms using the distributive law:

$$\begin{aligned} F &= xy + x'z = (xy + x')(xy + z) \\ &= (x + x')(y + x')(x + z)(y + z) \\ &= (x' + y)(x + z)(y + z) \end{aligned}$$

The function has three variables: x , y , and z . Each OR term is missing one variable; therefore:

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x + y' + z) \\ y + z &= y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

Combining all the terms and removing those that appear more than once, we finally obtain:

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0 M_2 M_4 M_5 \end{aligned}$$

A convenient way to express this function is as follows:

$$F(x,y,z) = \prod(0, 2, 4, 5)$$

2.5.4

Conversion between Canonical Forms

- $F(A,B, C) = \sum(1,4,5,6,7)$
- $F'(A, B, C) = \sum (0, 2, 3) = m_0 + m_2 + m_3$
- $F = (m_0 + m_2 + m_3)' = m'_0 . m'_2 . m'_3$
 $= M_0 M_2 M_3 = \prod (0, 2, 3)$
- $M'_j = M_j$

Note: To convert from one canonical form to another, interchange the symbols \sum and \prod and list those numbers missing from the original form.

2.5.5 Standard Forms

- There are two types of standard forms: the sum of products and product of sums.
- **The sum of products** is a Boolean expression containing AND terms, called product terms, of one or more literals each.

$$F1 = y' + xy + x'yz'$$

- **A product of sums** is a Boolean expression containing OR terms, called sum terms. Each term may have any number of literals.

$$F2 = x(y' + z)(x' + y + z' + w)$$

2.6 Other Logic Operations

- When the binary operators AND and OR are placed between two variables x and y , they form two Boolean functions $x \cdot y$ and $x + y$, respectively.
- Each of the functions in Table 2-6 is listed with an accompanying name and a comment that explains the function in some way. The 16 functions listed can be subdivided into three categories:
 1. Two functions that produce a constant 0 or 1.
 2. Four functions with unary operations complement and transfer.
 3. Ten functions with binary operators that define eight different operations AND, OR, NAND, NOR, exclusive-OR, equivalence, inhibition, and implication.

Table 2-6 Boolean expressions for the 16 functions of two variables

Boolean functions	Operator symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$x \odot y$	Equivalence*	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

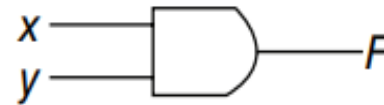
**Equivalence* is also known as *equality*, *coincidence*, and *exclusive-NOR*.

2.7 Digital Logic Gates

2.7.1 Extension to Multiple Inputs

Name	Graphic symbol	Algebraic function	Truth table
------	----------------	--------------------	-------------

AND



$$F = xy$$

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

2.7.1 Extension to Multiple Inputs

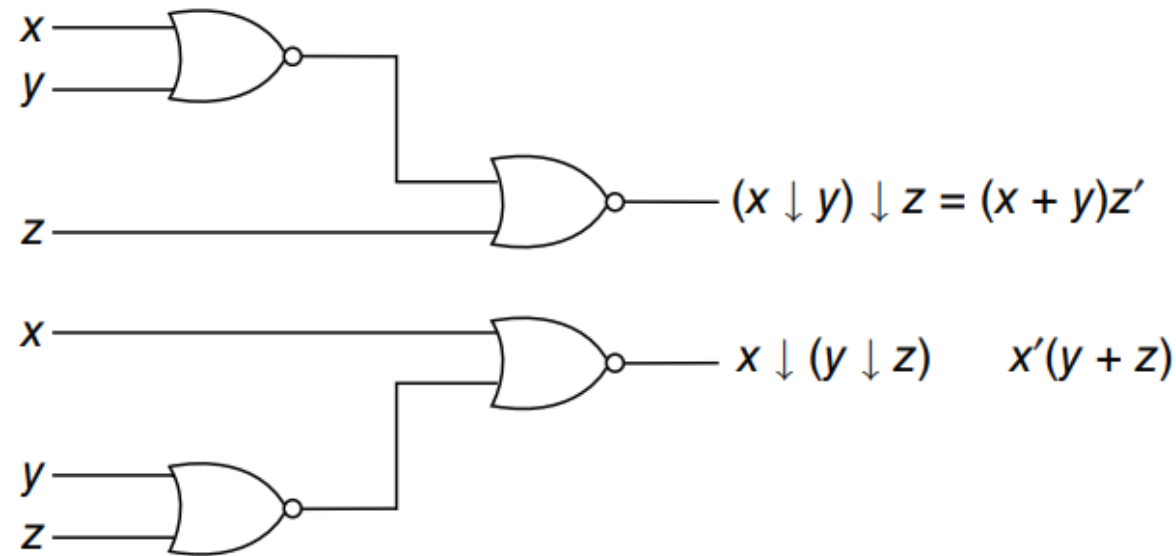


Figure 2.6 Demonstrating the nonassociativity of the NOR operator; $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

2.7.1 Extension to Multiple Inputs

The difficulty is that the NAND and NOR operators are not associative, i.e., $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$, as shown in Fig. 2-6 and below:

$$(x \downarrow y) \downarrow z = [(x + y)' + z]' = (x + y)z' = xz' + yz'$$

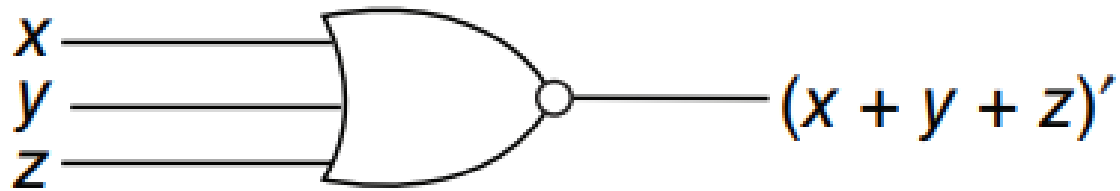
$$x \downarrow (y \downarrow z) = [x + (y + z)']' = x'(y + z) = x'y + x'z$$

To overcome this difficulty, we define the multiple NOR (or NAND) gate as a complemented OR (or AND) gate. Thus, by definition, we have:

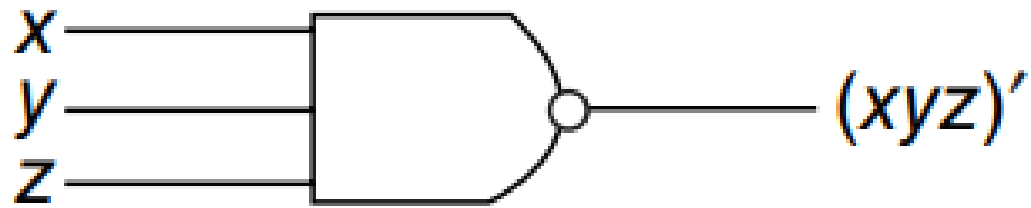
$$x \downarrow y \downarrow z = (x + y + z)'$$

$$x \uparrow y \uparrow z = (xyz)'$$

$$F = [(ABC)'(DE)']' = ABC + DE$$



(a) Three-input NOR gate



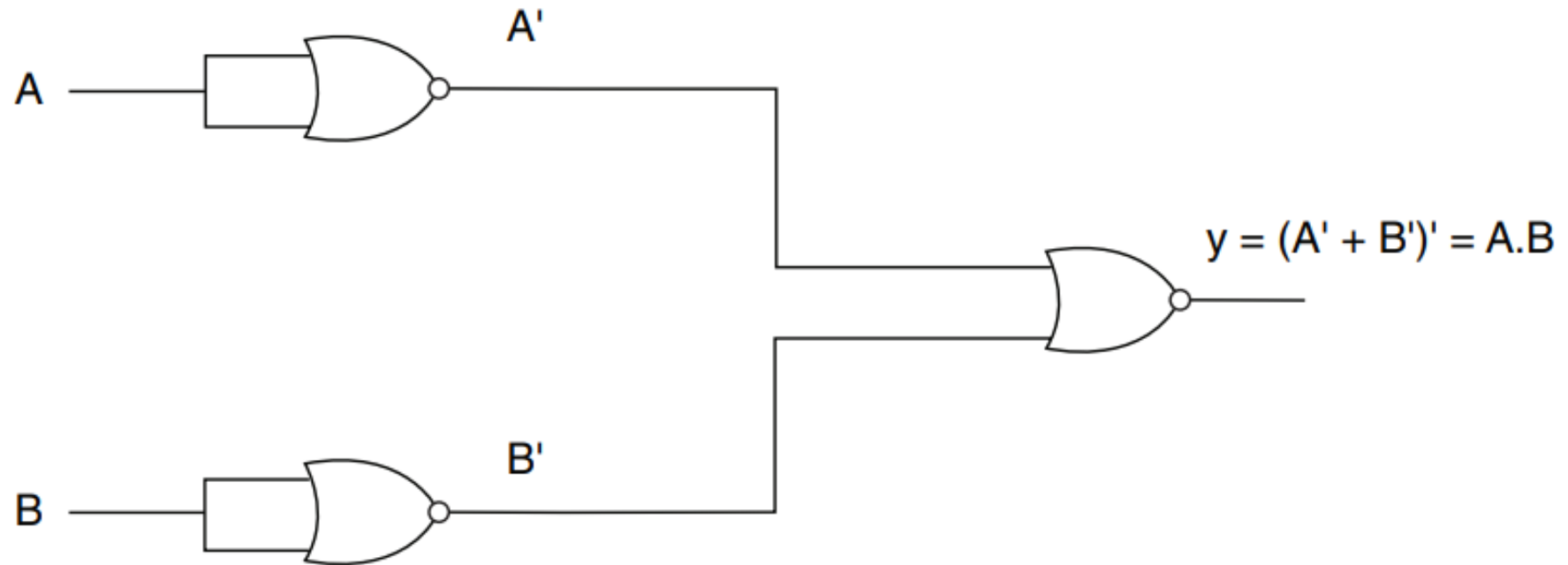
(b) Three-input NAND gate

2.7.1 Extension to Multiple Inputs

2.7.1 Extension to Multiple Inputs

EXAMPLE 2-7: Construct AND Gate using NOR Gate

Soln: $A.B = (A.B)'' = (A' + B')'$ using *D* Morgan's theorem



2.8 IC Digital Logic Families

TTL Transistor-transistor logic

ECL Emitter-coupled logic

MOS Metal-oxide semiconductor

CMOS Complementary metal-oxide
semiconductor

I²L Integrated-injection logic

2.8.1 Positive and Negative Logic

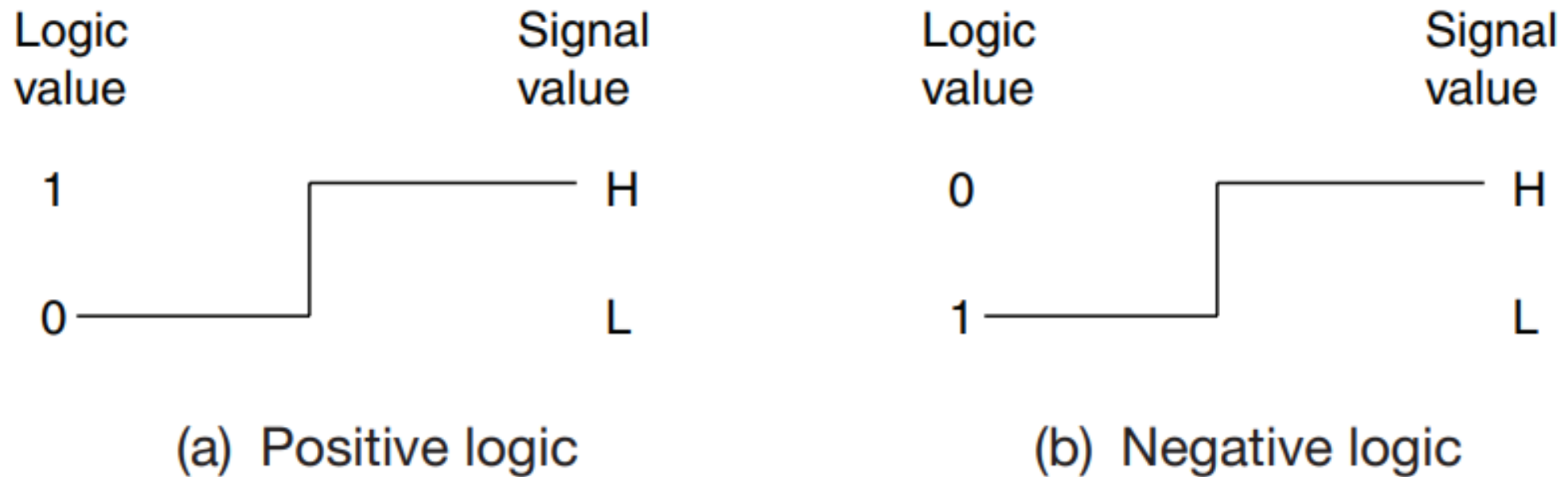


Figure 2.10 Signal-amplitude assignment and type of logic

2.8.1 Positive and Negative Logic

Table 2-7 H and L levels in IC logic families

IC family type	Voltage supply (V)	High-level voltage (V)		Low-level voltage (V)	
		Range	Typical	Range	Typical
TTL	$V_{CC} = 5$	2.4 – 5	3.5	0 – 0.4	0.2
ECL	$V_{EE} = -5.2$	-0.95 – -0.7	-0.8	-1.9 – -1.6	-1.8
CMOS	$V_{DD} = 3 - 10$	V_{DD}	V_{DD}	0 – 0.5	0
Positive logic:			logic-1		logic-0
Negative logic:			logic-0		logic-1

Thank You