Digital Design

CS302

Ref: Digital Design. M. Morris Mano , and Michael D. Ciletti. Pearson, FIFTH EDITION, 2013

Boolean Algebra and Logic Gates

CHAPTER 2

Introduction

Because binary logic is used in all of today's digital computers and devices, the cost of the circuits that implement it, is an important factor addressed by designers—be they computer engineers, electrical engineers, or computer scientists.

Mathematical methods that simplify circuits in digital computers rely primarily on Boolean algebra.

Therefore, this chapter provides a brief foundation in Boolean algebra .

Basic Definitions

- Boolean algebra may be defined with
  - a set of elements,
  - a set of operators,

  and
  - a number of unproved axioms or postulates.

Axiomatic Definition of Boolean Algebra

- In 1854, George Boole developed an algebraic system now called Boolean algebra.
- In 1938, Claude E. Shannon introduced a two-valued Boolean algebra called switching algebra that represented the properties of bistable electrical switching circuits.
- For the formal definition of Boolean algebra, we shall employ the postulates formulated by E. V. Huntington in 1904.

Axiomatic Definition of Boolean Algebra

- **Binary variables** take one of two values.
- **Logical operators** operate on binary values and binary variables.
- **Basic logical operators** : AND (·), OR (+) and NOT **(‘)**.
- **Logic gates** implement logic functions.
- **Boolean Algebra**: a mathematical system used for designing and analyzing digital systems

Axiomatic Definition of Boolean Algebra

Boolean algebra (1854) is an algebraic structure defined by a set of elements, *B, together* with two binary operators, + and · provided

1. (a) The structure is closed with respect to **+**.

   (b) The structure is closed w.r.t. ·

2. (a) The element 0 is an identity element w.r.t. +:

   $x + 0 = 0 + x = x.$

   (b) The element 1 is an identity element w.r.t. · :

   $x \cdot 1 = 1 \cdot x = x.$

3. (a) Commutativity w.r.t. +; $x + y = y + x.$

   (b) Commutativity w.r.t.· ; that is, $x \cdot y = y \cdot x.$

Axiomatic Definition of Boolean Algebra

4. (a) The operator $\cdot$ is distributive over +;

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

(b) The operator + 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 exist at least two elements $x, y \in B$ *such that* $x \neq y$

Two-valued Boolean algebra

Two-valued Boolean algebra (1938) is defined on a set of 2 elements B = {0, 1} with + , · and the complement as shown in the following operator tables

AND

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

OR

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

NOT

x	x'
0	1
1	0

Basic Theorems and Properties of Boolean Algebra

Duality

- **The dual of an algebraic expression is obtained by interchanging + and · and interchanging 0's and 1's.**

- **The duality principle:** state that a Boolean expression remains valid if we take the dual of the expression on both sides.

Example:

The Boolean statement: $1 + 0 = 1$

The dual of this statement is: $0 \cdot 1 = 0$

Basic Theorems

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, commutativ	(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)	$x(x + y) = x$

Basic Laws and Theorems

Commutative Law	A + B = B + A	A.B = B.A
Associative Law	A + (B + C) = (A + B) + C	A . (B . C) = (A . B) . C
Distributive Law	A.(B + C) = AB + AC	A + (B . C) = (A + B) . (A + C)
Null Elements	A + 1 = 1	A . 0 = 0
Identity	A + 0 = A	A . 1 = A
Idempotence	A + A = A	A . A = A
Complement	A + A' = 1	A . A' = 0
Involution	A" = A	
Absorption (Covering)	A + AB = A	A . (A + B) = A
Simplification	A + A'B = A + B	A . (A' + B) = A . B
DeMorgan's Rule	(A + B)' = A'.B'	(A . B)' = A' + B'
Logic Adjacency (Combining)	AB + AB' = A	(A + B) . (A + B') = A
Consensus	AB + BC + A'C = AB + A'C	(A + B) . (B + C) . (A' + C) = (A + B) . (A' + C)

Basic Theorems

THEOREM 1(a):

$x + x = x.$

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

Basic Theorems

THEOREM 1(b):

$x \cdot x = x.$

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

Basic Theorems

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

Statement	Justification
$x + xy = x \cdot 1 + xy$	postulate 2(b)
$= x(1 + y)$	4(a)
$= x(y + 1)$	3(a)
$= x \cdot 1$	2(a)
$= x$	2(b)

THEOREM 6(b):

$x(x + y) = x$ by duality.

Prove the following:

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

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

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

Basic Theorems

The theorem of Boolean Algebra can be proven by means of truth tables (Theorem 6a)

x	y
0	0
0	1
1	0
1	1

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

x
0
0
1
1

Basic Theorems

DeMorgan's Theorem

$$(x + y)' = x'\,y', \qquad (x \cdot y)' = x' + y',$$

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

x'	y'	$x'y'$
1	1	1
1	0	0
0	1	0
0	0	0

Operator Precedence

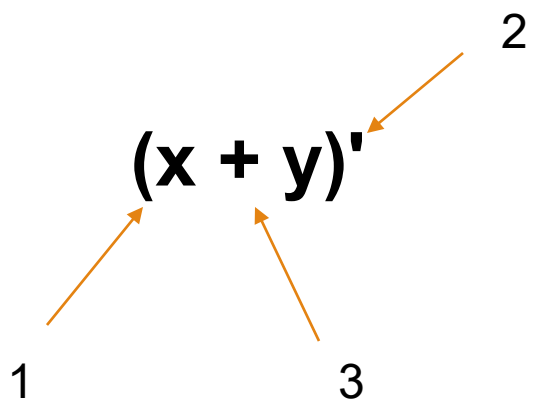
The operator precedence for evaluating Boolean expressions is:

(1) parentheses,

(2) NOT,

(3) AND, and

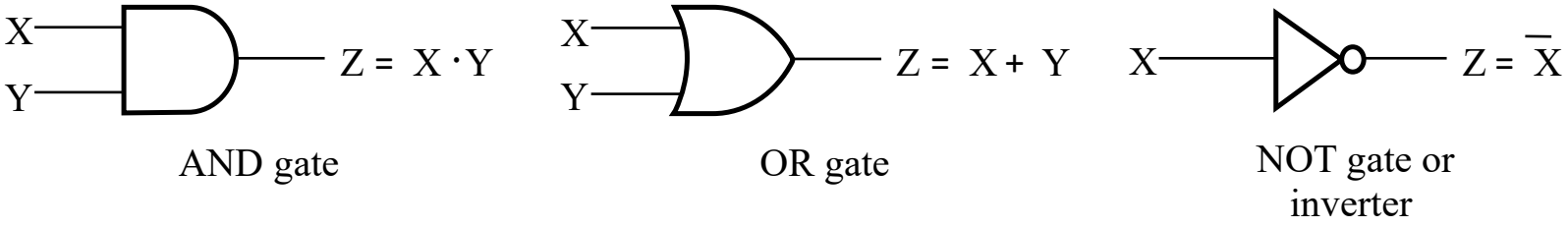
(4) OR.



<u>Exercise:</u> F = A(B + C)'(C + D)

Boolean Functions

Logic Gate Symbols



A Boolean function is described by an algebraic expression

Algebraic Manipulation

We define a *literal to* be a single variable within a term, in complemented or uncomplemented form.

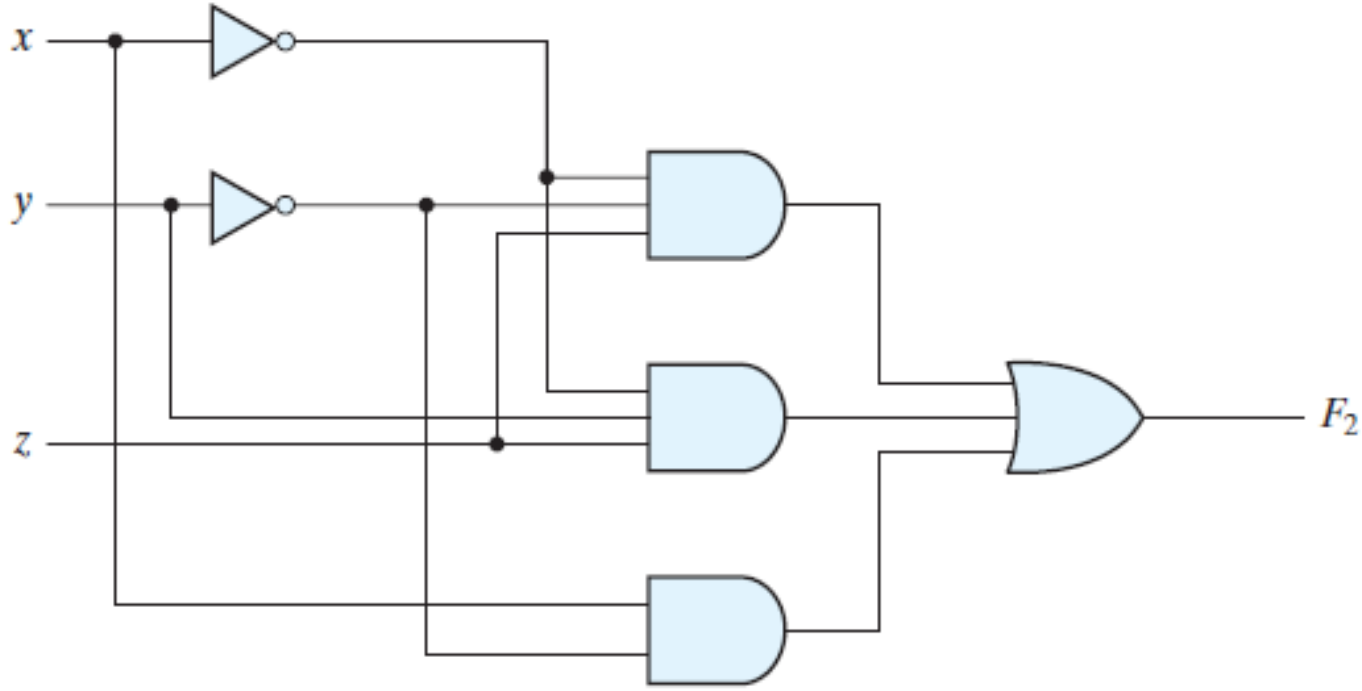
The function

$$F2 = x'y'z + x'yz + xy'$$

has three terms and eight literals,

And depends on 3 variables.

Algebraic Manipulation

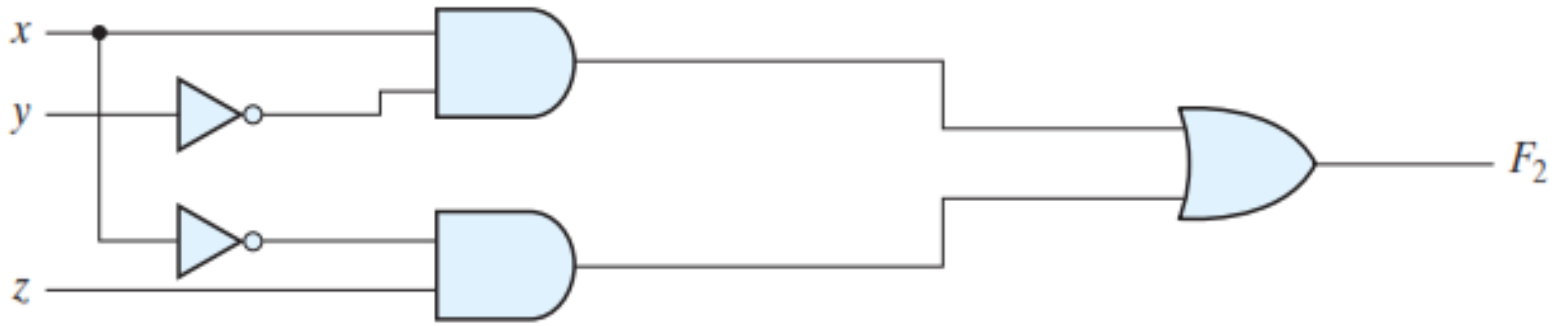


(a) $F_2 = x'y'z + x'yz + xy'$

Algebraic Manipulation

By reducing the number of terms, the number of literals, or both in a Boolean expression, it is possible to obtain a simpler circuit.

$$F_2 = x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$$



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

Algebraic Manipulation

EXAMPLE 2.1

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

1. $x(x' + y) = xx' + xy$
$= 0 + xy = xy.$

2. $x + x'y = (x + x')(x + y)$
$= 1(x + y) = x + y.$

Algebraic Manipulation

3. $(x + y)(x + y') = x + xy + xy' + yy' = x(1 + y + y') = x.$

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

5. $$\begin{aligned} &(x + y)(x' + z)(y + z) \\ &\quad = (x + y)(x' + z), \text{ by duality from function 4.} \end{aligned}$$

Complement of a function

A simpler way **for deriving the complement of a function is to**

1. **take the dual of the function and**
2. **complement each literal.**

This method follows from the generalized forms of DeMorgan's theorems

Complement of a function

Example 2.3

Find the complement of the functions *F1 and F2*

F1 = x′y z ′+ x ′y ′z.

F2 = x (y ′ z ′ + yz).

Complement of a function

Example 2.3

Find the complement of the functions *F1 and F2*

$F1 = x'y\,z' + x'y'z.$

The dual of *F1* is $(x' + y + z')\,(x' + y' + z)$.

Complement each literal: $(x + y' + z)(x + y + z') = F1'$

$F2 = x\,(y'\,z' + yz)$.

The dual of *F2 is* $x + (y' + z')(y + z)$.

Complement each literal: $x' + (y + z)(y' + z') = F2'$

Canonical and Standard Forms

Minterms (*standard product*) are AND terms with every variable present in either true or complemented form.

There are 2^n minterms for *n* variables (each binary variable may appear normal or complemented .

Example: Two variables (X and Y) produce 2 x 2 = 4 combinations:

X Y (both normal)

X Y′ (X normal, Y complemented)

X′ Y (X complemented, Y normal)

X′ Y′ (both complemented)

Canonical and Standard Forms

Maxterms (*standard sums*) are OR terms with every variable in true or complemented form.

There are 2^n maxterms for *n* variables (each binary variable may appear normal or complemented) .

Example: Two variables (X and Y) produce
2 x 2 = 4 combinations:

X + Y (both normal)
X + Y′ (X normal, Y complemented)
X′ + Y (X complemented, Y normal)
X′ + Y′ (both complemented)

Table 2.3
Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	$M_0 = \overline{m_0}$
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

- It is clear that the following relation holds: $m'j = Mj$

Canonical and Standard Forms

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in *canonical form*

Any Boolean function can be expressed as a sum of minterms

Sum of minterms of entries that evaluate to '**1**'

x	y	z	F	Minterm
0	0	0	0	
0	0	1	**1**	$m_1 = x'y'z$
0	1	0	0	
0	1	1	0	
1	0	0	**1**	$m_4 = xy'z'$
1	0	1	0	
1	1	0	0	
1	1	1	**1**	$m_7 = x\,y\,z$

Focus on the '**1**' entries

$$F = m_1 + m_4 + m_7 = \sum(1, 4, 7) = x'y'z + xy'z' + x\,y\,z$$

To find the complement of a Boolean function :

1. **form a minterm for each combination that produces a 0 in the function and then**
2. **ORing those terms.**

Note:
Any Boolean function can be expressed as a sum of minterms (with "sum" meaning the ORing of terms).

Example

$F' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$

If we take the complement of *F'* we obtain the function *F:*

$F = (x + y + z)(x + y' + z)(x + y' + z')(x' + y + z')(x' + y' + z)$

$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$

EXAMPLE 2.4: Sum of Minterms

Express the function $F = A + B'C$ as a sum of minterms.

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

This function is still missing one variable, so

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

$= ABC + ABC' + AB'C + AB'C'$

The second term $B'C$ is missing one variable; hence,

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

$F = A + B'C$

$= ABC + ABC' + \mathbf{AB'C} + AB'C' + \mathbf{AB'C} + A'B'C$

$= ABC + ABC' + AB'C + AB'C' + A'B'C$

$A'B'C + AB'C' + AB'C + ABC' + ABC$

$= m1 + m4 + m5 + m6 + m7 = \sum(1, 4, 5, 6, 7)$

Table 2.5

Truth Table for $F = A + B'C$

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

$A = 1$ and $BC = 01$

An alternative procedure for deriving the minterms of a Boolean function is to obtain the truth table of the function directly from the algebraic expression and then read the minterms from the truth table.

EXAMPLE 2.5: Product of Maxterms

Express the Boolean function $F = xy + x'z$ as a product of maxterms.

First, convert the function into OR terms by using the distributive law:

$$F = xy + x'z = (xy + x')(xy + z) \quad \text{(distrib. law)}$$

$$= (x' + x)(y + x')(x + z)(y + z) \quad \text{(distrib. law)}$$

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

The function has three variables: *x, y, and z.*

Each OR term is missing one variable

EXAMPLE 2.5: Product of Maxterms

$$F = (x' + y)(x + z)(y + z)$$

$$x' + y = x' + y + zz' = (x' + y + z)(x' + y + z')$$

$$x + z = x + z + yy' = (x + y + z)(x + y' + z)$$

$$y + z = y + z + xx' = (x + y + z)(x' + y + z)$$

$$F = (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z')$$

$$= M_0 M_2 M_4 M_5$$

This function can be express as follows:

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

The product symbol, $\prod$, denotes the ANDing of maxterms; the numbers are the indices of the maxterms of the function.

Conversion between Canonical Forms

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function.

Conversion between Canonical Forms

Example

Consider the function

$$F(A, B, C) = \sum(1, 4, 5, 6, 7)$$

This function has a complement that can be expressed as

$$F'(A, B, C) = \sum(0, 2, 3) = m0 + m2 + m3$$

Now, if we take the complement of F by DeMorgan's theorem, we obtain F in a different form:

$$F = (m0 + m2 + m3)' = m'0 \cdot m'2 \cdot m'3 = M0M2M3 = \prod(0, 2, 3)$$

Note that: mj'=Mj

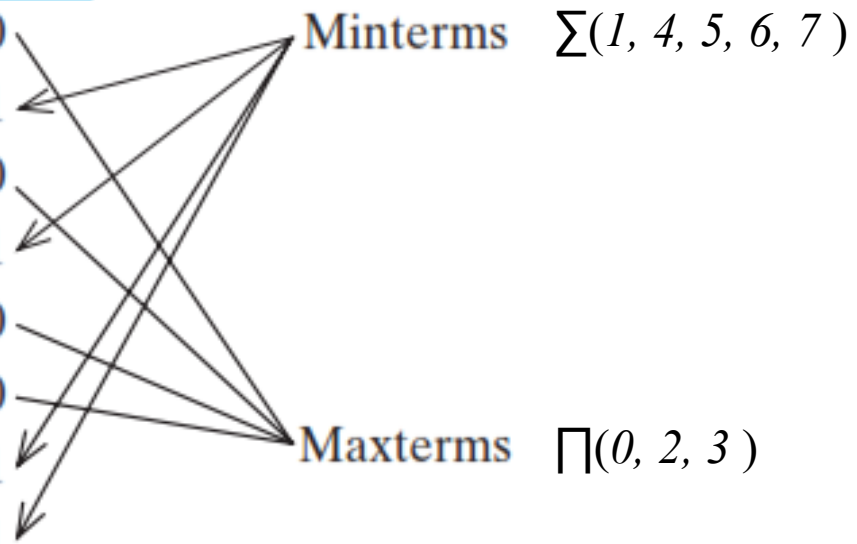
A general conversion procedure:

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

Table 2.6

Truth Table for F = xy + x′z

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



Standard Forms

The two canonical forms of Boolean algebra are very seldom because each minterm or maxterm must contain, by definition, *all the variables, either complemented or uncomplemented.*

Another way to express Boolean functions is in *standard form where* the terms that form the function may contain one, two, or any number of literals.

Two types of standard forms:

- **The sum of products and**
- **The product of sums.**

Sum of Products

The sum of products is a Boolean expression containing AND terms, called product terms, with one or more literals each.

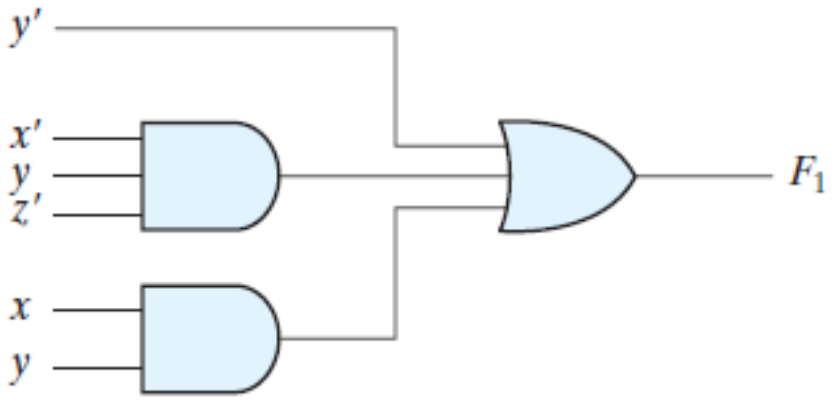
The sum denotes the ORing of these terms.

Note:

Any Boolean function can be expressed as a sum of minterms (with “sum” meaning the ORing of terms).

Example

F1 = y' + xy + x'yz'



(a) Sum of Products

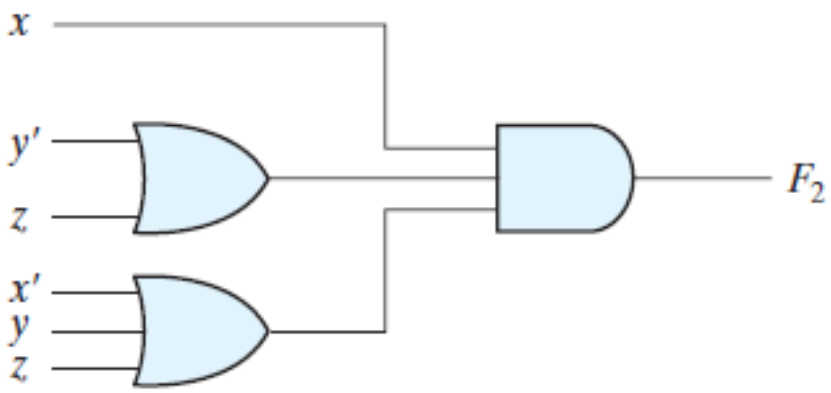
Product of Sums

A product of sums is a Boolean expression containing OR terms. Each term may have any number of literals.

The product denotes the ANDing of these terms.

Example

$F2 = x\,(\,y' + z\,)(x' + y + z'\,)$

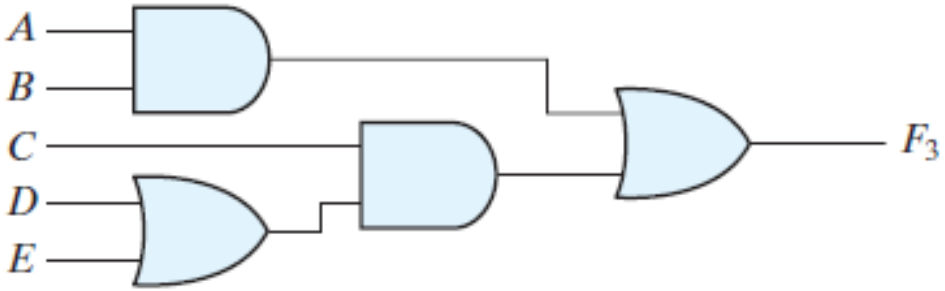


(b) Product of Sums

Standard forms

A Boolean function may be expressed in a nonstandard form. For example, the function F_3 is neither in sum-of-products nor in product-of-sums form.

$$F_3 = AB + C(D + E)$$

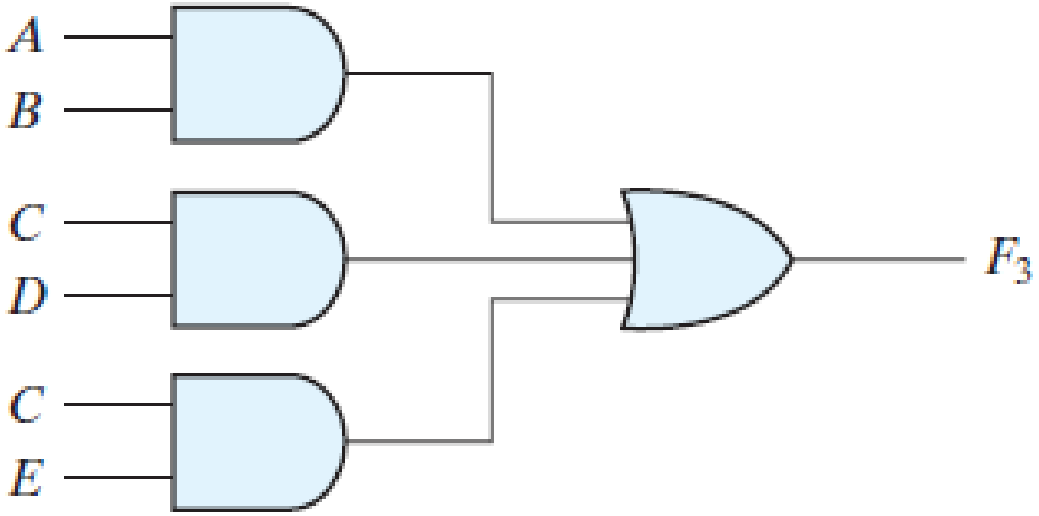


(a) $AB + C(D + E)$

FIGURE 2.4
Three- and two-level implementation

It can be changed to a standard form by using the distributive law to remove the parentheses:

$F_3 = AB + C(D + E) = AB + CD + CE$



(b) $AB + CD + CE$

In general, a two-level implementation is preferred because it produces the least amount of delay through the gates when the signal propagates from the inputs to the output.

OTHER LOGIC OPERATIONS

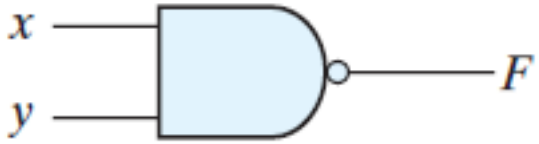
Table 2.7
Truth Tables for the 16 Functions of Two Binary Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

0 AND xy‘ x x'y y XOR OR NOR XNOR y' x+y‘ x' x’+y NAND 1

Digital Logic Gates

NAND

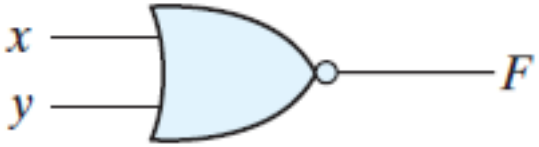


$F = (xy)'$

$= x \uparrow y$

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

NOR



$F = (x + y)'$

$x \downarrow y$

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

Digital Logic Gates

The exclusive-OR (XOR), denoted by the symbol $\oplus$, is a logical operation that performs the following Boolean operation:

$$x \oplus y = xy' + x'y$$

The exclusive-OR is equal to 1 if only *x is equal to 1 or if only y is equal to 1*

Exclusive-OR (XOR)

$F = xy' + x'y$
$= x \oplus y$

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

Homework

Page 69:

2.1 $\rightarrow$ 2.9, 2.15, 2.17, 2.20 $\rightarrow$ 2.22, 2.30, 2.31