+

Chapter 11 Digital Logic

William Stallings, Computer Organization and Architecture, 9th Edition

Objectives

- What are the basis of digital circuits?
- What are the basic electronic components?
- How can minimize a combinational circuits?
- After studying this chapter, you should be able to:
  - Understand the basic operations of Boolean algebra.
  - Use a Karnaugh map to simplify a Boolean expression.

Contents

11.1- Boolean Algebra

- Mathematical discipline (môn) used to design and analyze the behavior of the digital circuitry in digital computers and other digital systems
- **Named after George Boole**
  - English mathematician
  - Proposed basic principles of the algebra in 1854
- Claude Shannon suggested Boolean algebra could be used to solve problems in relay-switching circuit design
- **Is a convenient tool:**
  - **Analysis**
    - It is an economical way of describing the function of digital circuitry
  - **Design**
    - Given a desired function, Boolean algebra can be applied to develop a simplified implementation of that function

+

Boolean Algebra

- Investigated Set:

  B = { False, True } = { F, T} = {0,1}

- Basic Operator: AND (.), OR (+), NOT
- Other operators: NAND (Not And), NOR (Not Or), XOR (Exclusive OR)
- Representation:

$$
\begin{aligned}
A \text{ AND } B &= A \cdot B \\
A \text{ OR } B &= A + B \\
\text{NOT } A &= \overline{A}
\end{aligned}
$$

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

$$
\begin{aligned}
A \text{ NAND } B &= \text{NOT}(A \text{ AND } B) = \overline{AB} \\
A \text{ NOR } B &= \text{NOT}(A \text{ OR } B) = \overline{A + B}
\end{aligned}
$$

Boolean Variables and Operations

- **Makes use of variables and operations**
  - Are logical
  - A variable may take on the value 1 (TRUE) or 0 (FALSE)
  - Basic logical operations are AND, OR, and NOT
- **AND**
  - Yields true (binary value 1) if and only if both of its operands are true
  - In the absence of parentheses the AND operation takes precedence over the OR operation
  - When no ambiguity will occur the AND operation is represented by simple concatenation instead of the dot operator
- **OR**
  - Yields true if either or both of its operands are true
- **NOT**
  - Inverts the value of its operand

Table 11.1- Boolean Operators

Table 11.1 Boolean Operators

(a) Boolean Operators of Two Input Variables

P	Q	NOT P $(\overline{P})$	P AND Q $(P \cdot Q)$	P OR Q $(P + Q)$	P NAND Q $(\overline{P \cdot Q})$	P NOR Q $(\overline{P + Q})$	P XOR Q $(P \oplus Q)$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0

(b) Boolean Operators Extended to More than Two Inputs (A, B, . . .)

Operation	Expression	Output = 1 if
AND	$A \cdot B \cdot \ldots$	All of the set {A, B, ...} are 1.
OR	$A + B + \ldots$	Any of the set {A, B, ...} are 1.
NAND	$\overline{A \cdot B \cdot \ldots}$	Any of the set {A, B, ...} are 0.
NOR	$\overline{A + B + \ldots}$	All of the set {A, B, ...} are 0.
XOR	$A \oplus B \oplus \ldots$	The set {A, B, ...} contains an odd number of ones.

Table 11.2: Basic Identities of Boolean Algebra

Table 11.2 Basic Identities of Boolean Algebra

Basic Postulates		
$A \cdot B = B \cdot A$	$A + B = B + A$	Commutative Laws
$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$	Distributive Laws
$1 \cdot A = A$	$0 + A = A$	Identity Elements
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$	Inverse Elements
Other Identities		
$0 \cdot A = 0$	$1 + A = 1$	
$A \cdot A = A$	$A + A = A$	
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$	Associative Laws
$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$	DeMorgan's Theorem

11.2- Basic Logic Gates

An electronic switch that is the elementary component of a digital circuit. It produces an electrical output signal that represents a binary 1 or 0 and is related to the states of one or more input signals by an operation of Boolean logic, such as AND, OR, or NOT (Microsoft Computer Dictionary)

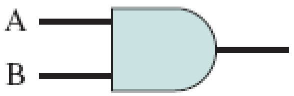
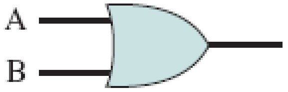
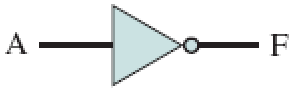
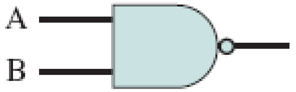
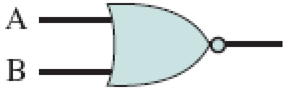
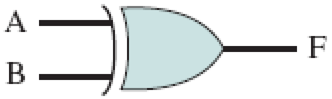
Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \bullet B$ or $F = AB$	A B F 0 0 0 0 1 0 1 0 0 1 1 1
OR		$F = A + B$	A B F 0 0 0 0 1 1 1 0 1 1 1 1
NOT		$F = \overline{A}$ or $F = A'$	A F 0 1 1 0
NAND		$F = \overline{AB}$	A B F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$F = \overline{A + B}$	A B F 0 0 1 0 1 0 1 0 0 1 1 0
XOR		$F = A \oplus B$	A B F 0 0 0 0 1 1 1 0 1 1 1 0

Figure 11.1 Basic Logic Gates

+

Uses of NAND Gates

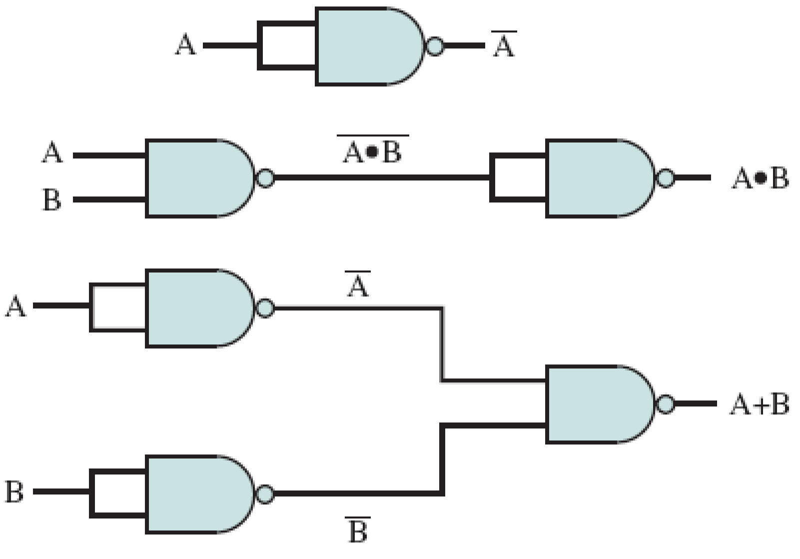


Figure 11.2 Some Uses of NAND Gates

Uses of NOR Gates

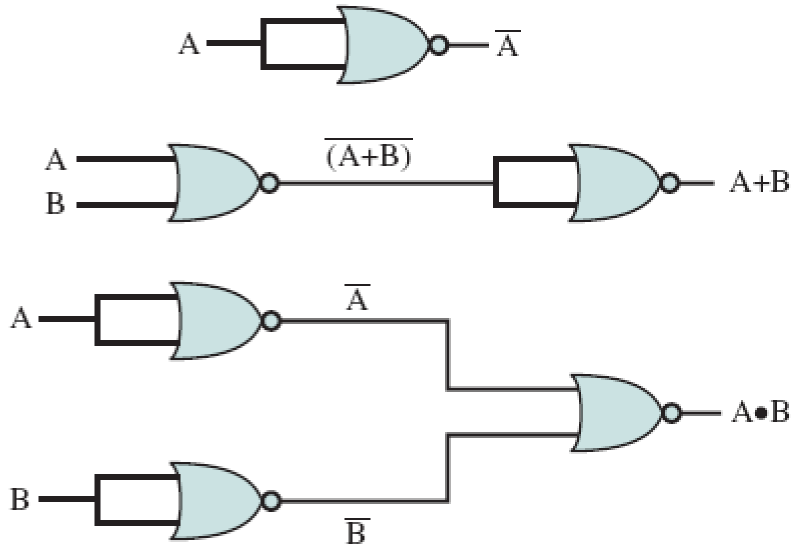
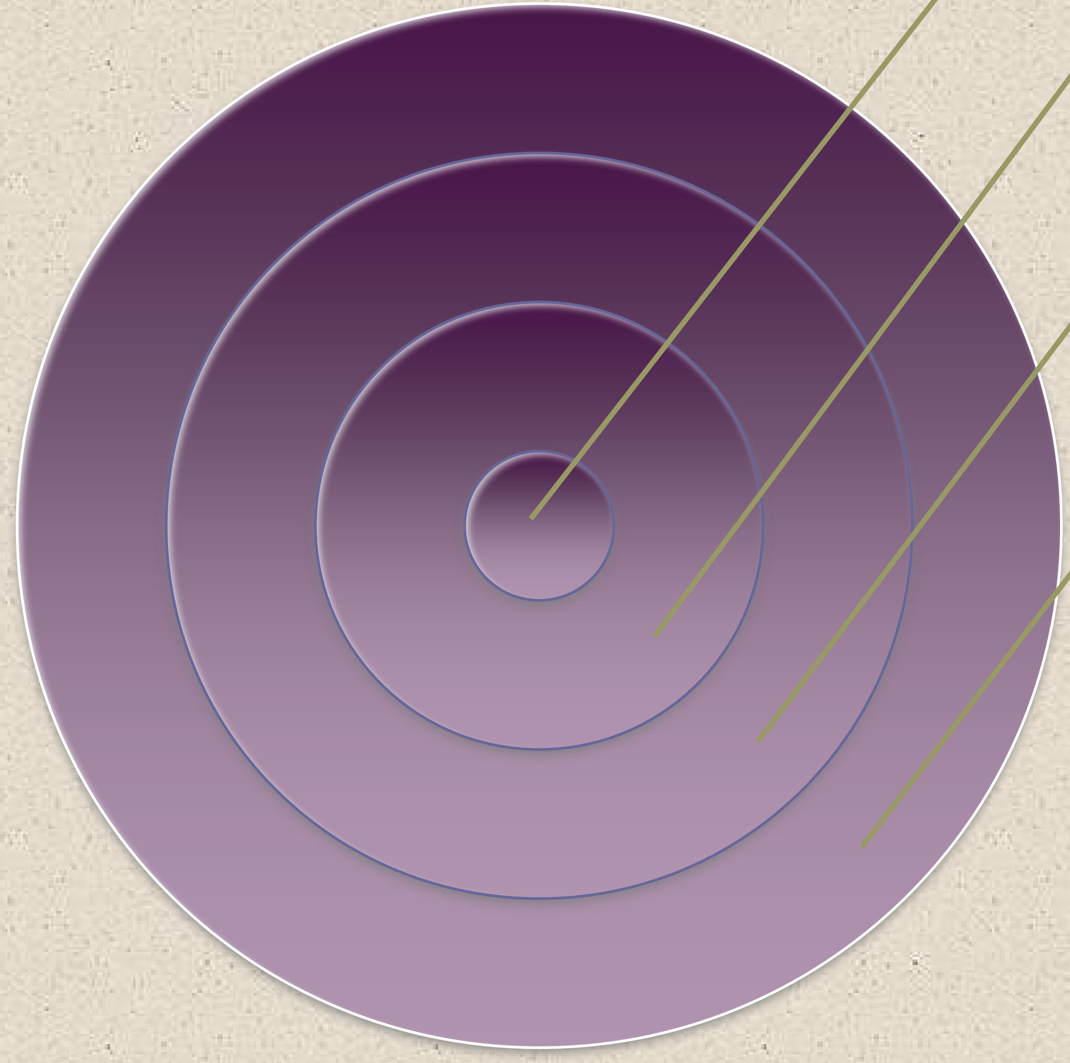


Figure 11.3 Some Uses of NOR Gates

11.3- Combinational Circuit



An interconnected set of gates whose output at any time is a function only of the input at that time

The appearance of the input is followed almost immediately by the appearance of the output, with only gate delays

Consists of n binary inputs and m binary outputs

Can be defined in three ways:

- Truth table
  - For each of the 2^n possible combinations of input signals, the binary value of each of the m output signals is listed
- Graphical symbols
  - The interconnected layout of gates is depicted
- Boolean equations
  - Each output signal is expressed as a Boolean function of its input signals

Example: Using 3 ways for a Boolean Function of Three Variables

Sum of product (SOP)

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

Table 11.3 A Boolean Function of Three Variables

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

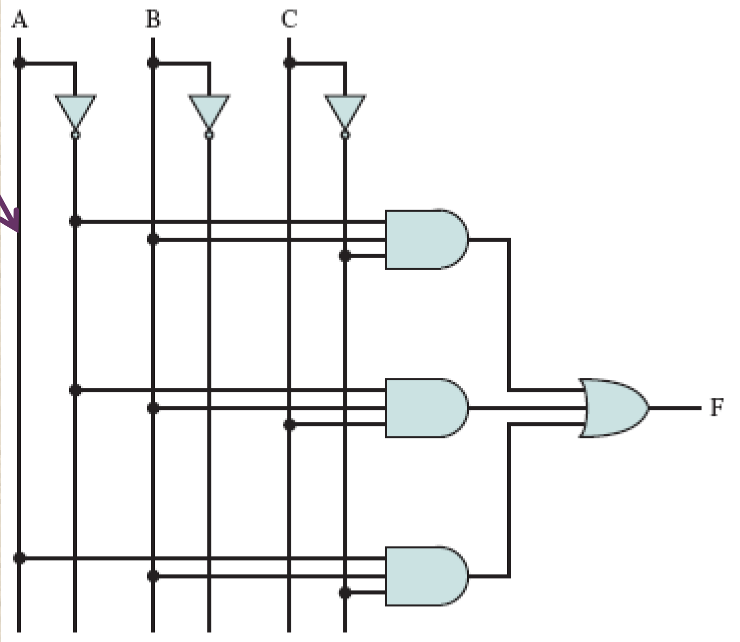


Figure 11.4 Sum-of-Products Implementation of Table 11.3

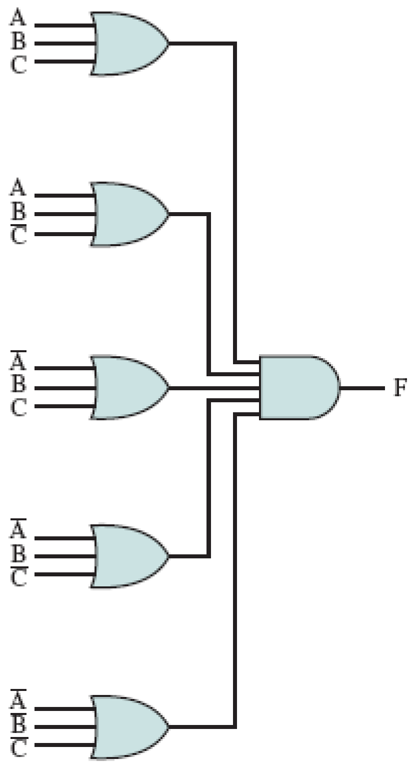


Figure 11.5 Product-of-Sums Implementation of Table 11.3

Product of Sum (POS)

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

Algebraic Simplication
Minimize a Boolean Function

- A Boolean function will be implemented as a combinational network → More complex function will cause a more complex network
- How to minimize a Bollean function?
  - Methods:
    - Karnaugh Map
    - Quine-McCluskey Method

Algebraic Simplification

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

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

Table 11.3 A Boolean Function of Three Variables

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

$$
\begin{aligned}
F &= \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C} \\
&= \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C} + \overline{A}B\overline{C} \\
&= \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C} + \overline{A}B\overline{C} \\
&= \overline{A}B\overline{C} + AB\overline{C} + \overline{A}BC + \overline{A}B\overline{C} \\
&= B\overline{C}(\overline{A} + A) + \overline{A}B(C + \overline{C}) \\
&= B\overline{C} + \overline{A}B \\
&= B(\overline{A} + \overline{C})
\end{aligned}
$$

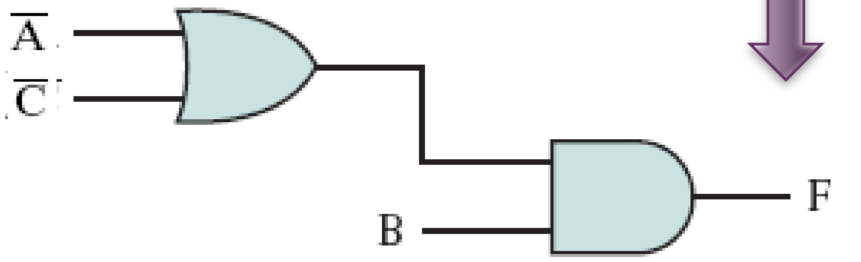


Figure 11.6 Simplified Implementation

Karnaugh Map

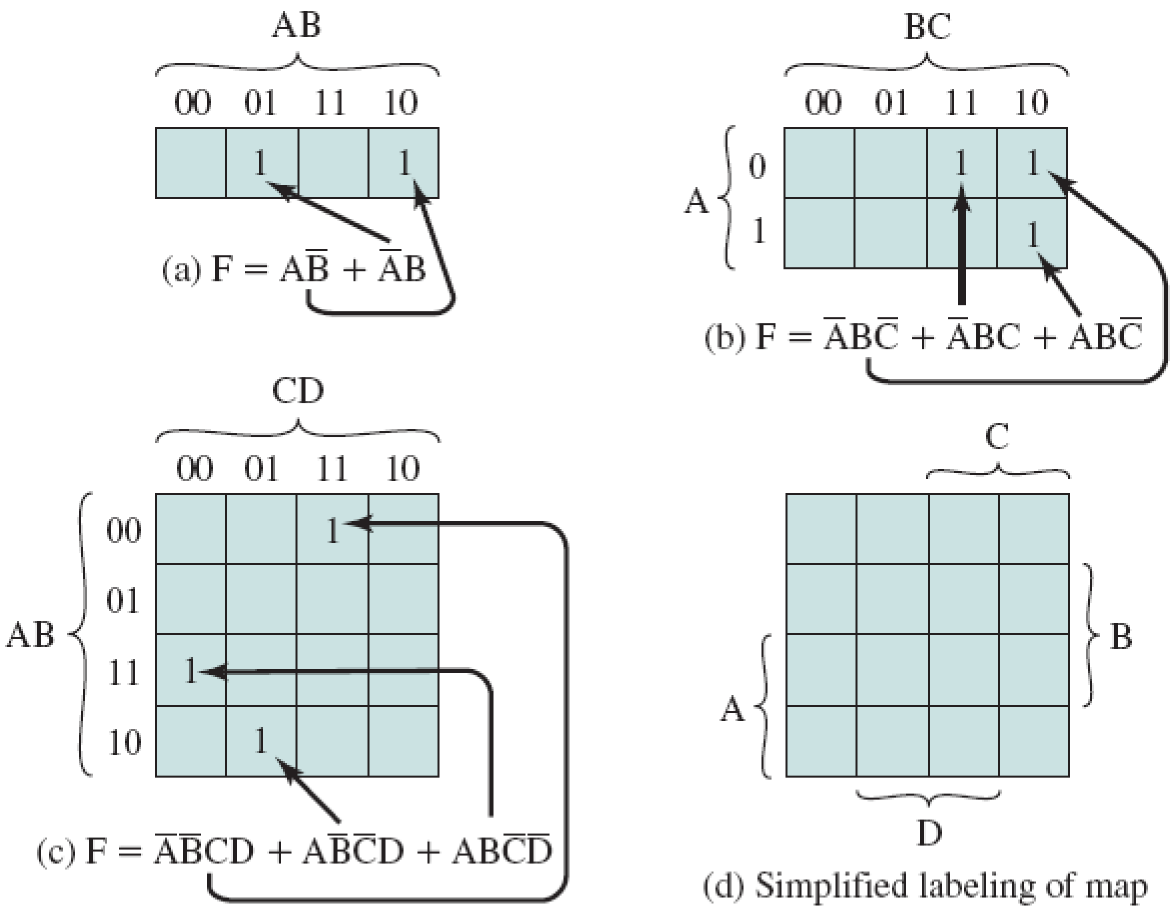


Figure 11.7 The Use of Karnaugh Maps to Represent Boolean Functions

- A convenient way of representing a Boolean function of a small number (up to four) of variables

Example

Karnaugh Maps

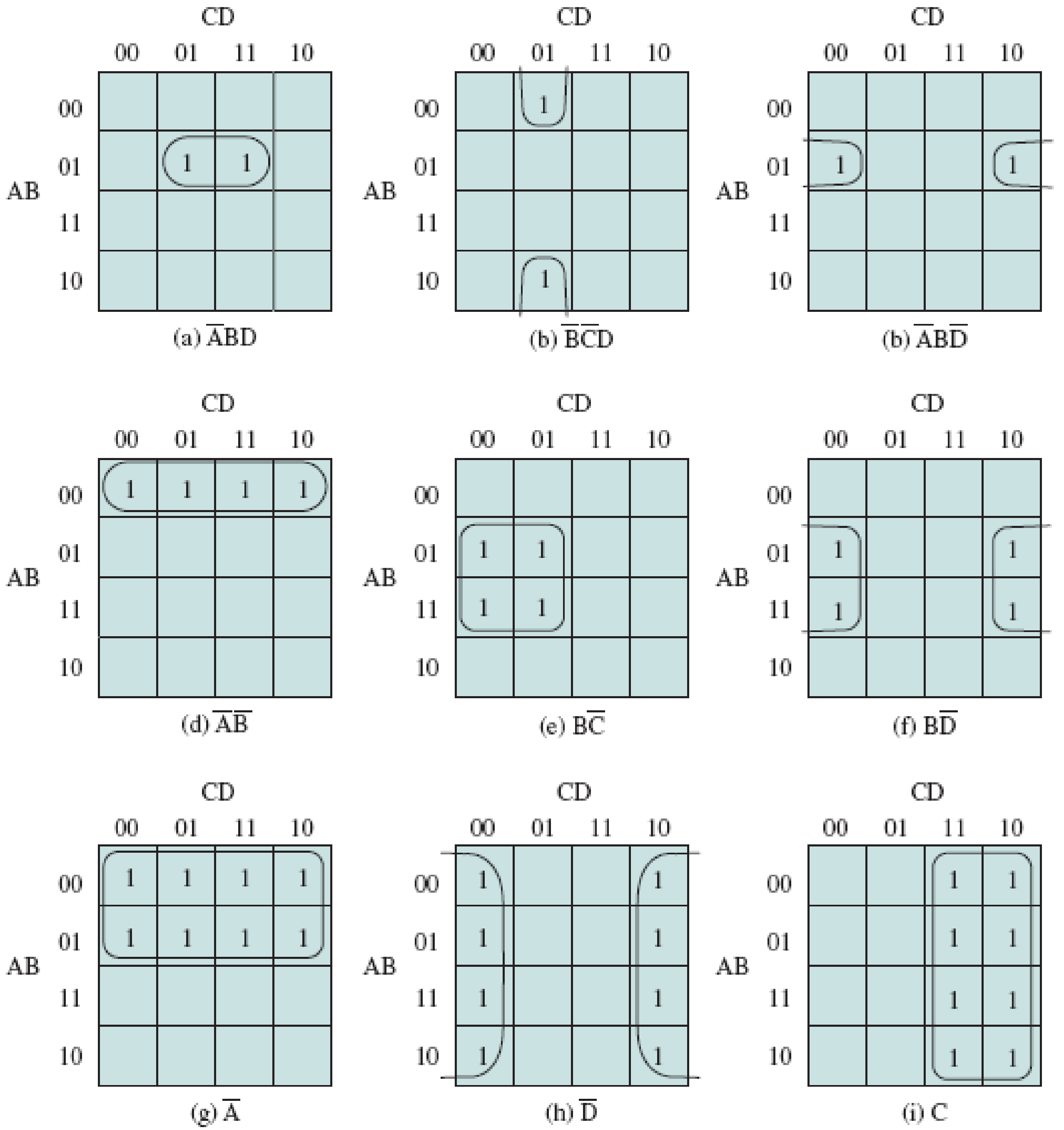


Figure 11.8 The Use of Karnaugh Maps

Overlapping Groups

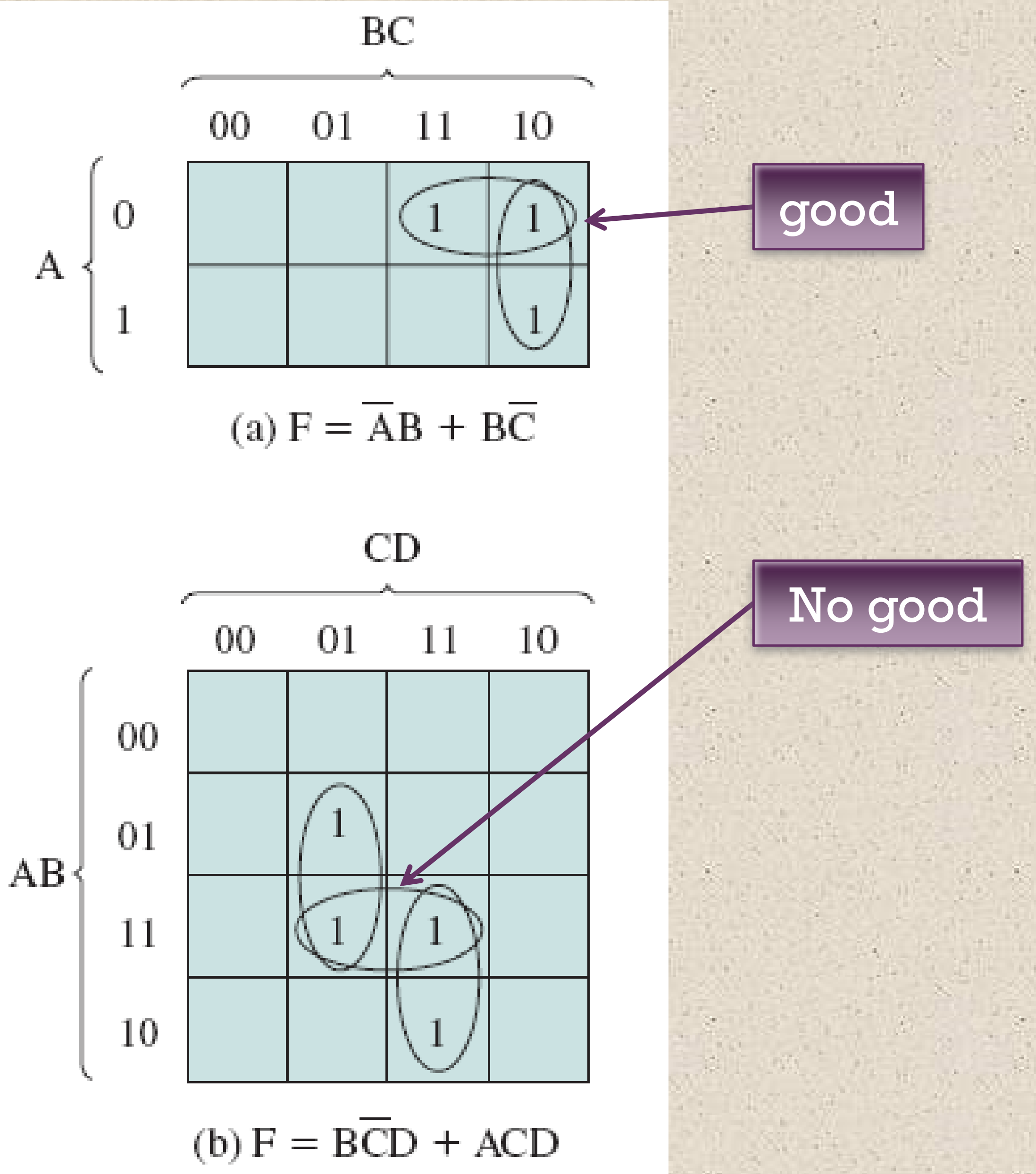


Figure 11.9 Overlapping Groups

Table 11.4- Truth Table for the One-Digit Packed Decimal Incrementer

Table 11.4 Truth Table for the One-Digit Packed Decimal Incrementer

	Input					Output			
Number	**A**	**B**	**C**	**D**	**Number**	**W**	**X**	**Y**	**Z**
0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	2	0	0	1	0
2	0	0	1	0	3	0	0	1	1
3	0	0	1	1	4	0	1	0	0
4	0	1	0	0	5	0	1	0	1
5	0	1	0	1	6	0	1	1	0
6	0	1	1	0	7	0	1	1	1
7	0	1	1	1	8	1	0	0	0
8	1	0	0	0	9	1	0	0	1
9	1	0	0	1	0	0	0	0	0
Don't care condition	1	0	1	0		d	d	d	d
	1	0	1	1		d	d	d	d
	1	1	0	0		d	d	d	d
	1	1	0	1		d	d	d	d
	1	1	1	0		d	d	d	d
	1	1	1	1		d	d	d	d

Figure 11.10

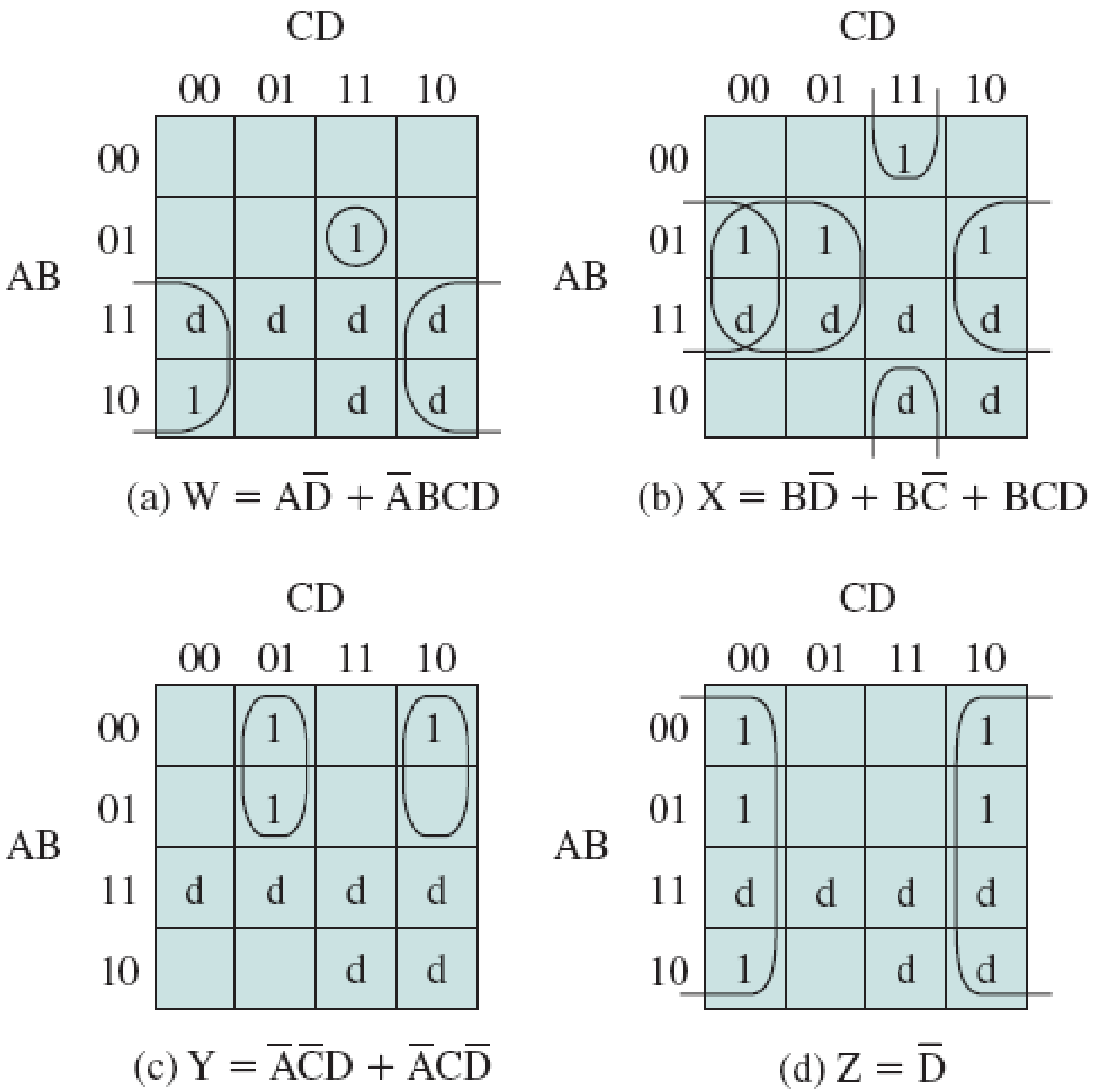


Figure 11.10 Karnaugh Maps for the Incrementer

Table 11.5: First Stage of Quine-McCluskey Method

Table 11.5 First Stage of **Quine–McCluskey Method**
(for $F = ABCD + AB\overline{C}D + AB\overline{C}\,\overline{D} + A\overline{B}CD + \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D$)

	Product Term	Index	A	B	C	D	
0001	$\overline{A}\,\overline{B}\,\overline{C}D$	1	0	0	0	1	✓
0101	$\overline{A}B\overline{C}D$	5	0	1	0	1	✓
0110	$\overline{A}BC\overline{D}$	6	0	1	1	0	✓
1100	$AB\overline{C}\,\overline{D}$	12	1	1	0	0	✓
0111	$\overline{A}BCD$	7	0	1	1	1	✓
1011	$A\overline{B}CD$	11	1	0	1	1	✓
1101	$AB\overline{C}D$	13	1	1	0	1	✓
1111	$ABCD$	15	1	1	1	1	✓

A → 1, Not A → 0

ABCD → 1111 → Index=15

Table 11.6: Last Stage of Quine-McCluskey Method

Table 11.6 Last Stage of Quine–McCluskey Method
(for $F = ABCD + AB\overline{C}D + AB\overline{C}\,\overline{D} + A\overline{B}CD + \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}\,B\,\overline{C}D$)

	$ABCD$	$AB\overline{C}D$	$AB\overline{C}\,\overline{D}$	$A\overline{B}CD$	$\overline{A}BCD$	$\overline{A}BC\overline{D}$	$\overline{A}B\overline{C}D$	$\overline{A}\,\overline{B}\,CD$
BD	X	X			X		X	
$\overline{A}\,\overline{C}D$							$\boxed{X}$	⊗
$\overline{A}BC$					$\boxed{X}$	⊗		
$AB\overline{C}$		$\boxed{X}$	⊗					
ACD	$\boxed{X}$			⊗				

1111 + 1101 → 11-1

0111 + 0101 → 01-1

11-1+ 01-1 → -1-1 → BD

Exercises

11.1 Construct a truth table for the following Boolean expressions:

a. $ABC + \overline{A}\,\overline{B}\,\overline{C}$
b. $ABC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C}$
c. $A(B\overline{C} + \overline{B}C)$
d. $(A + B)(A + C)(\overline{A} + \overline{B})$

11.2 Simplify the following expressions according to the commutative law:

a. $A \cdot \overline{B} + \overline{B} \cdot A + C \cdot D \cdot E + \overline{C} \cdot D \cdot E + E \cdot \overline{C} \cdot D$
b. $A \cdot B + A \cdot C + B \cdot A$
c. $(L \cdot M \cdot N)(A \cdot B)(C \cdot D \cdot E)(M \cdot N \cdot L)$
d. $F \cdot (K + R) + S \cdot V + W \cdot \overline{X} + V \cdot S + \overline{X} \cdot W + (R + K) \cdot F$

11.3 Apply DeMorgan's theorem to the following equations:

a. $F = \overline{V + A + L}$
b. $F = \overline{A} + \overline{B} + \overline{C} + \overline{D}$

11.4 Simplify the following expressions:

a. $A = S \cdot T + V \cdot W + R \cdot S \cdot T$
b. $A = T \cdot U \cdot V + X \cdot Y + Y$
c. $A = F \cdot (E + F + G)$
d. $A = (P \cdot Q + R + S \cdot T)T \cdot S$
e. $A = \overline{\overline{D} \cdot \overline{D} \cdot E}$
f. $A = Y \cdot (W + X + \overline{\overline{Y} + \overline{Z}}) \cdot Z$
g. $A = (B \cdot E + C + F) \cdot C$

+
Summary

11.5 Construct the operation XOR from the basic Boolean operations AND, OR, and NOT.

11.6 Given a NOR gate and NOT gates, draw a logic diagram that will perform the three-input AND function.

11.7 Write the Boolean expression for a four-input **NAND gate**.

11.8 A combinational circuit is used to control a seven-segment display of decimal digits, as shown in Figure 11.35. The circuit has four inputs, which provide the four-bit code used in packed decimal representation ($0_{10} = 0000, \ldots, 9_{10} = 1001$). The seven outputs define which segments will be activated to display a given decimal digit. Note that some combinations of inputs and outputs are not needed.

a. Develop a truth table for this circuit.
b. Express the truth table in SOP form.
c. Express the truth table in POS form.
d. Provide a simplified expression.

Summary

Chapter 11

Digital Logic

- Boolean Algebra
- Gates
- Combinational Circuit
  - **Algebraic Simplification**
    - Karnaugh Map
    - Quine-McCluskey Method