Combinational Logic Circuits

Instructor: Saraju P Mohanty

(Part 1)

- Binary Logic and Logic Gates (AND, OR, AND)
- Boolean Algebra (Identities, Manipulation, Complement of a Function)
- Standard Forms of Boolean Function (Minterms, Maxterms, SOP, POS)
- Map Specification (Karnaugh Map or K-map)

Digital circuits

- Digital circuits are hardware components manipulating binary information.
- Circuits are implemented using transistors and interconnections in complex semiconductor devices called integrated circuits.
- Each basic circuit is referred to as a logic gate.
- The internal electronics of individual gates are not of our concern, we are only interested in external logic properties.
- Each gate performs specific logic operation and transform input to the output.

Binary Logic

- Binary logic deals with binary variables (that assumes two discrete values) and with the operations applied to these variables.
- Binary variables are designated by alphabets, such as A, B, C,….., Z.
- The three basic logic operations are AND, OR, and NOT.

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

Table 2-1 Truth Tables for the Three Basic Logical Operations

Logic Gates

•Logic gates are electronic circuits that operate on one or more input signal to produce an output signal.

•Gates can be specified in three different ways: Graphic Symbols, Truth Tables, and Timing Diagrams.

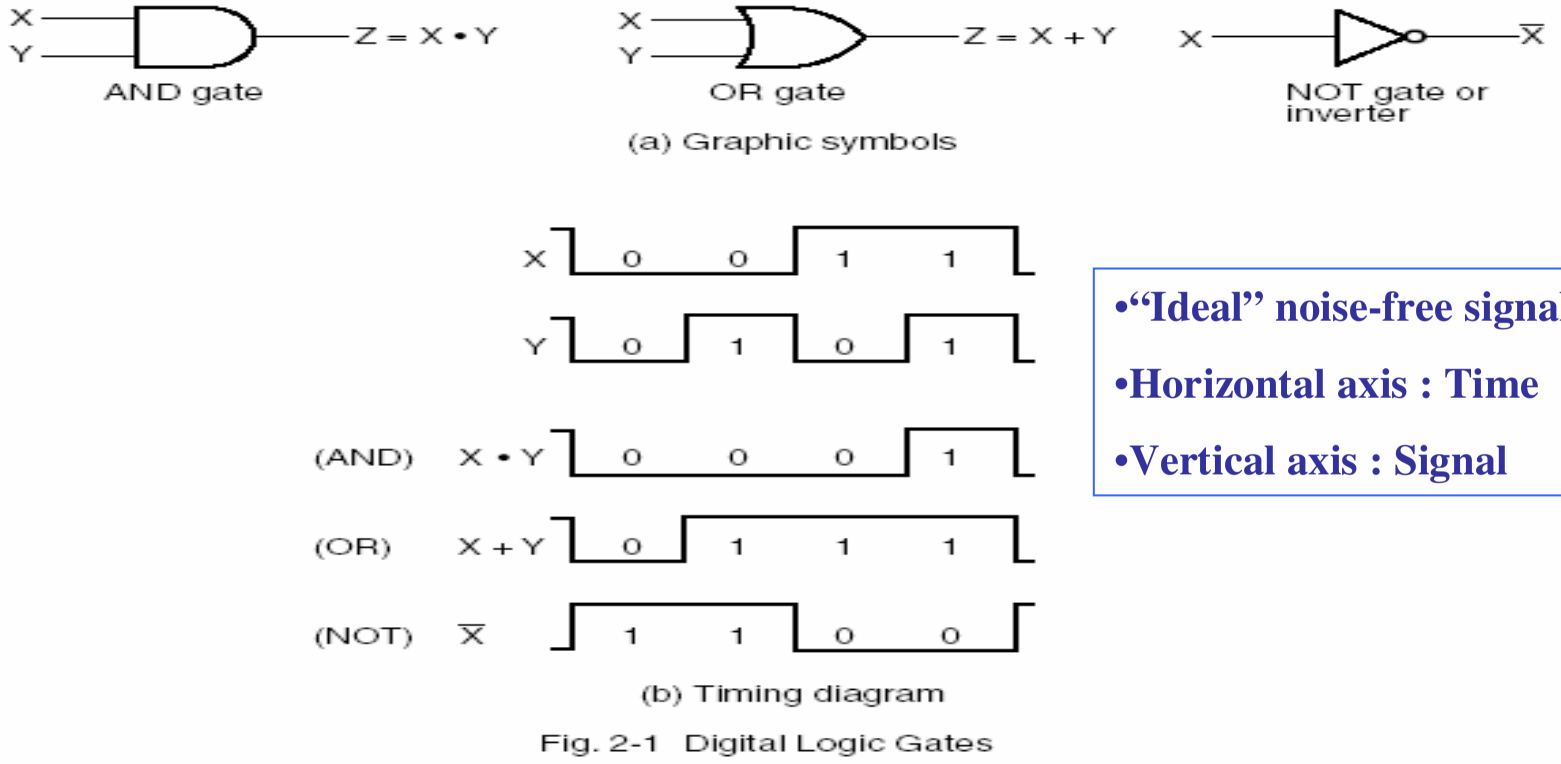


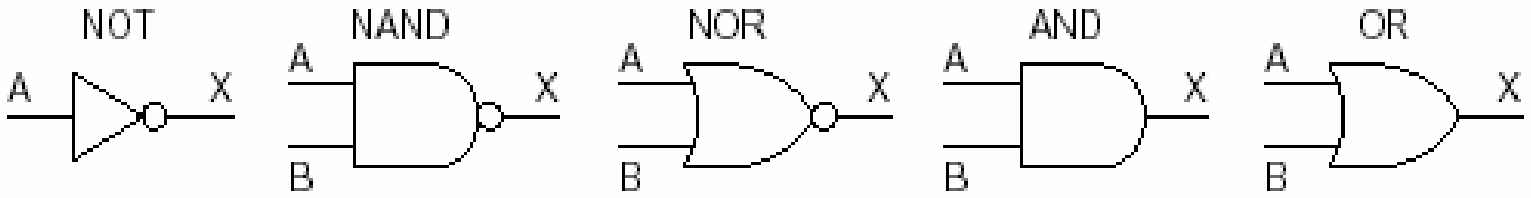
Fig. 2-1 Digital Logic Gates

•"Ideal" noise-free signals

•Horizontal axis : Time

•Vertical axis : Signal

Logic Gates: All basic gates, with NAND and NOR



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

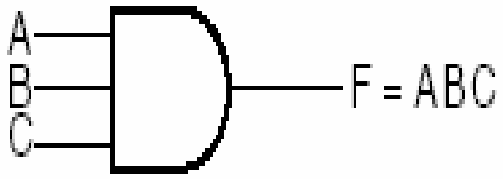
(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

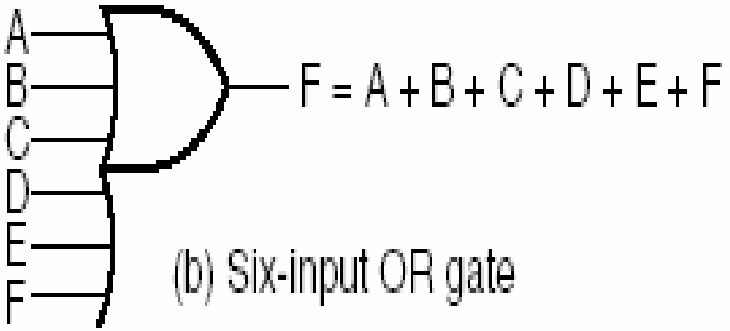
(e)

Figure 3-2. The symbols and functional behavior for the five basic gates.

Logic Gates: Gates with more than two inputs

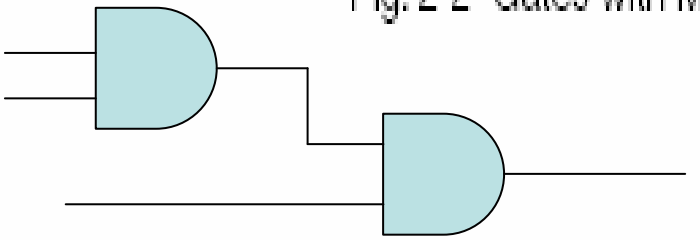


(a) Three-input AND gate



(b) Six-input OR gate

Fig. 2-2 Gates with More than Two Inputs



Alternative two input gate design for F=ABC

Logic Gates: More on gates…

NOTE: Operationally, **NOR = AND** and **NAND = OR**

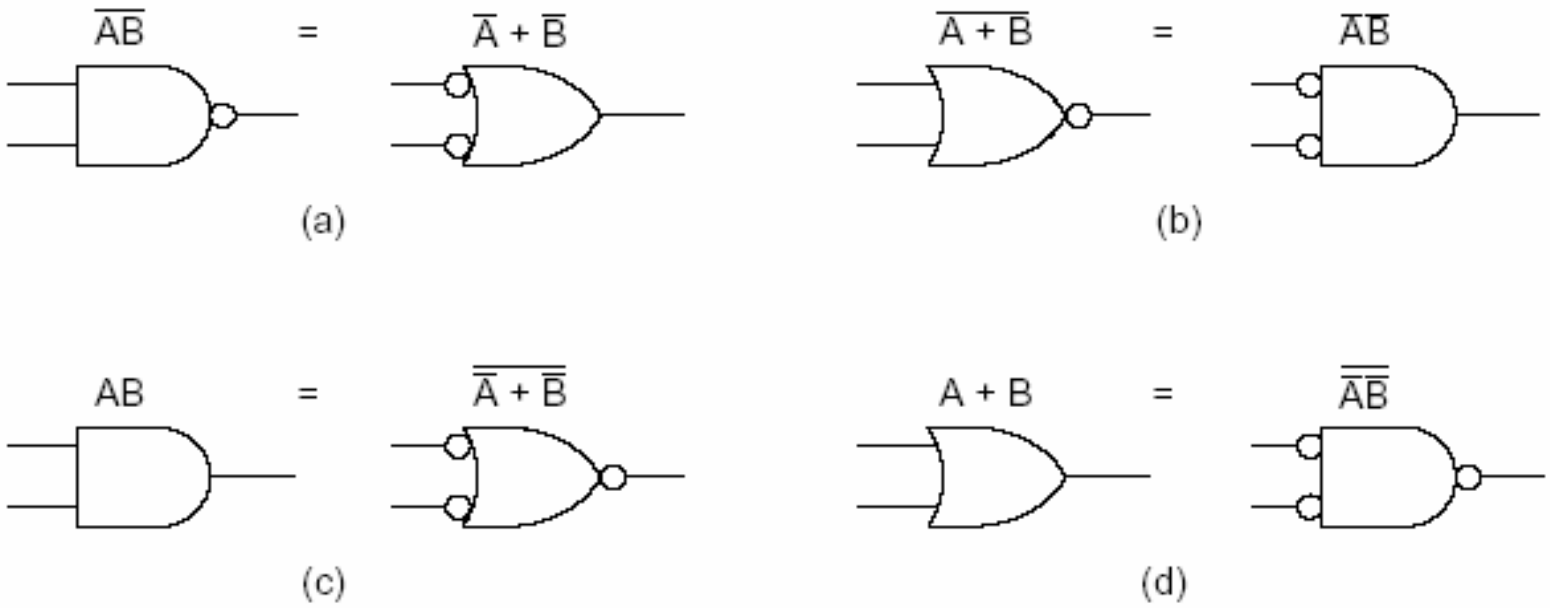


Figure 3-7. Alternative symbols for some gates: (a) NAND. (b) NOR. (c) AND. (d) OR.

Gates at Transistor Level : BJT

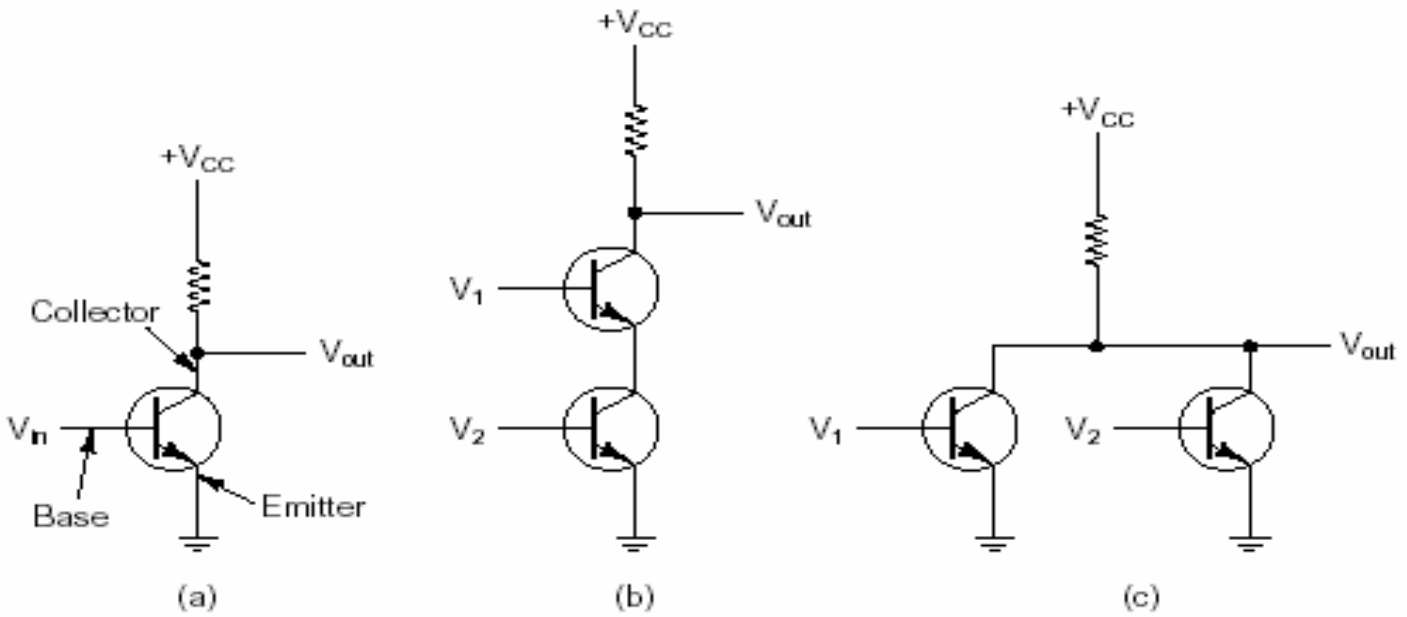
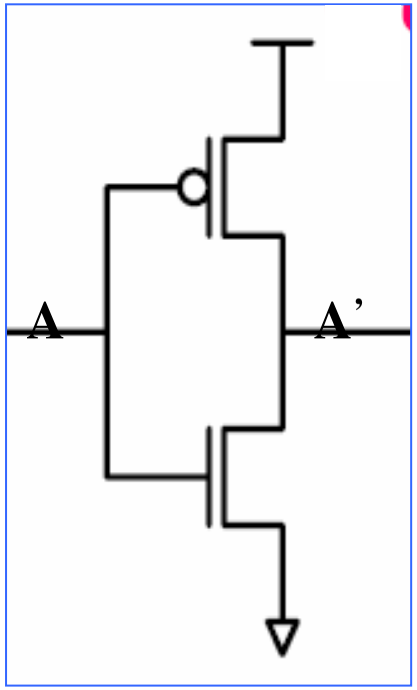


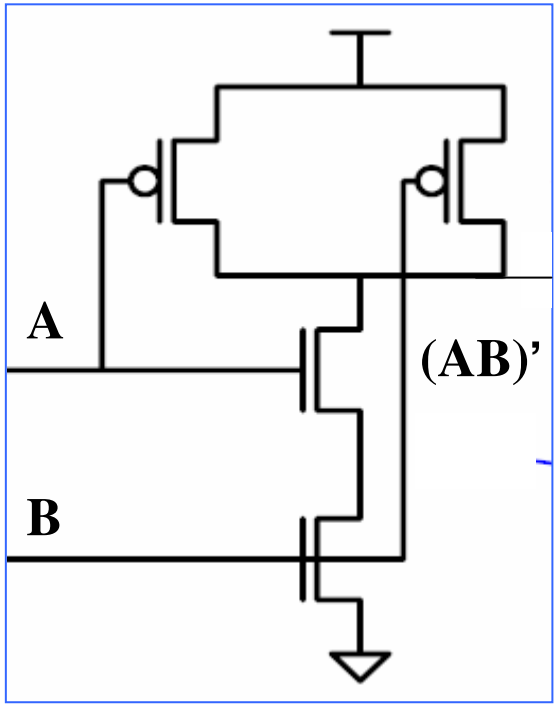
Figure 3-1. (a) A transistor inverter. (b) A NAND gate. (c) A NOR gate.

BJT : Bipolar Junction Transistor

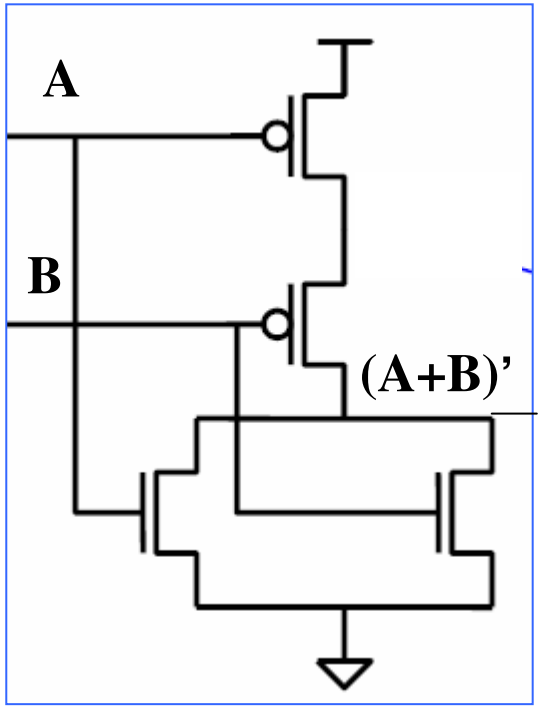
Gates at Transistor Level : CMOS FET



Inverter



NAND



NOR

CMOS FET : Complementary Metal Oxide Semiconductor Field Effect Transistor

Boolean Algebra

- To describe the operational properties of digital circuits mathematical notation specifying each gate operation is introduced, which is a class of mathematical system called Boolean Algebra.
- For simplicity, the algebra dealing with binary variables and logic properties is Boolean Algebra.
- A Boolean function has binary variables, an equal sign and algebraic expression.
- Boolean Function can be represented as a truth table as well.
- For n variable function the number of rows of the truth table will be 2^n.

Boolean Algebra : Boolean Function ……..

- For a given value of binary variables the Boolean function can be equal to either “0” or “1”.
- Example : F = X + Y’Z

X	Y	Z	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

Table 2-2 Truth Table for the Function $F = X + \overline{Y}Z$

- The two parts of the algebraic expression, such as “X” and “Y’Z” are called terms of the function.

Combinational Logic Circuit ??

- In logic circuits the variables are given as input and the functional value is taken as output.
- The gates are interconnected by wires that carry logic signals.
- Thus, the variables are combined by the logic operations, which gives rise to the name “combinational logic”.
- NOTE: Sequential variables are stored over time and as well as combined.

Boolean Algebra : Some Identities

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

Some identities of Boolean algebra.

Boolean Algebra : Some Identities

- Nine identities involve single variable.
- Seven identities involve two or more than two variables.
- The duals of an algebraic expression is obtained by interchanging the OR and AND operations, and replacing "1"s by "0"s and "0"s by "1"s.

1.	$X+0=X$	2.	$X\cdot 1=X$	
3.	$X+1=1$	4.	$X\cdot 0=0$	
5.	$X+X=X$	6.	$X\cdot X=X$	
7.	$X+\overline{X}=1$	8.	$X\cdot\overline{X}=0$	
9.	$\overline{\overline{X}}=X$			
10.	$X+Y=Y+X$	11.	$XY=YX$	Commutative
12.	$X+(Y+Z)=(X+Y)+Z$	13.	$X(YZ)=(XY)Z$	Associative
14.	$X(Y+Z)=XY+XZ$	15.	$X+YZ=(X+Y)(X+Z)$	Distributive
16.	$\overline{X+Y}=\overline{X}\cdot\overline{Y}$	17.	$\overline{X\cdot Y}=\overline{X}+\overline{Y}$	DeMorgan's

Table 2-3 Basic Identities of Boolean Algebra

Left and right column expressions are duals of each other, for example 10 and 11 are duals each other.

Boolean Algebra : DeMorgan's Theorem

A) X	Y	X + Y	$\overline{X+Y}$	B) X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X} \cdot \overline{Y}$
0	0	0	1	0	0	1	1	1
0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	0	1	0
1	1	1	0	1	1	0	0	0

Table 2-4 Truth Tables to Verify DeMorgan's Theorem

DeMorgan's Theorem for *n* variables

- $(X_1+X_2+X_3+\ldots\ldots+X_n)' = X_1'X_2'\ldots..X_n'$
- $(X_1X_2X_3\ldots\ldots X_n)' = X_1'+X_2'+\ldots..+X_n'$

Boolean Algebra: Algebraic Manipulations

•When Boolean expressions is implemented with logic gates, each term requires a gate, and each variable within the term becomes an input to the gate.

•Example: Lets consider the implementation of X+Y'Z

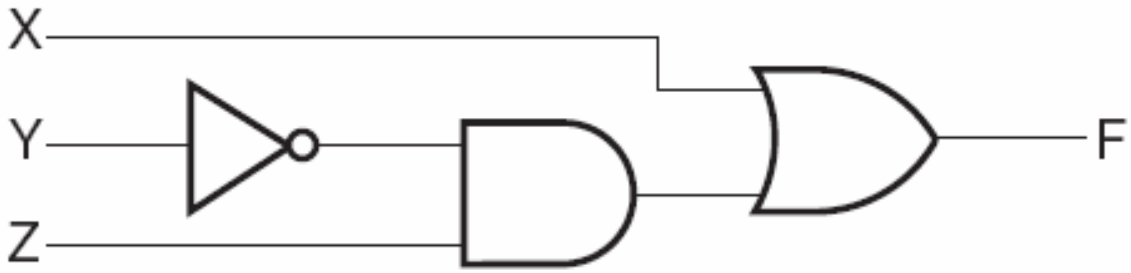
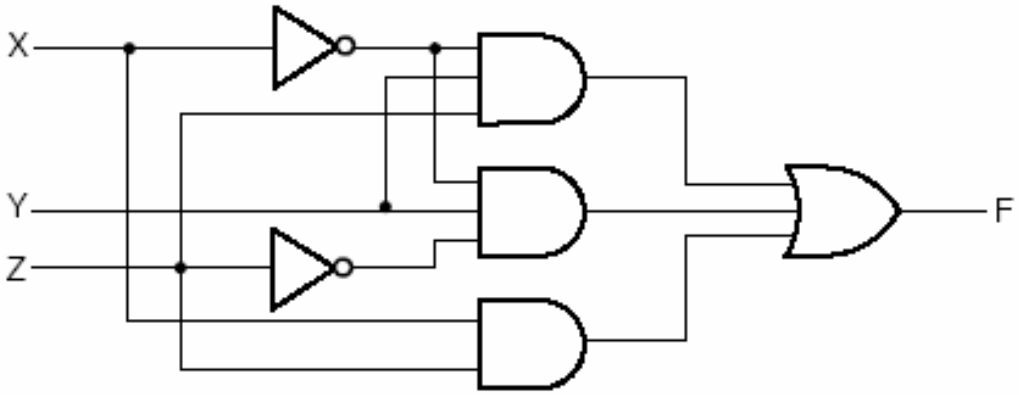


Fig. 2-3 Logic Circuit Diagram for $F = X + \overline{Y}Z$

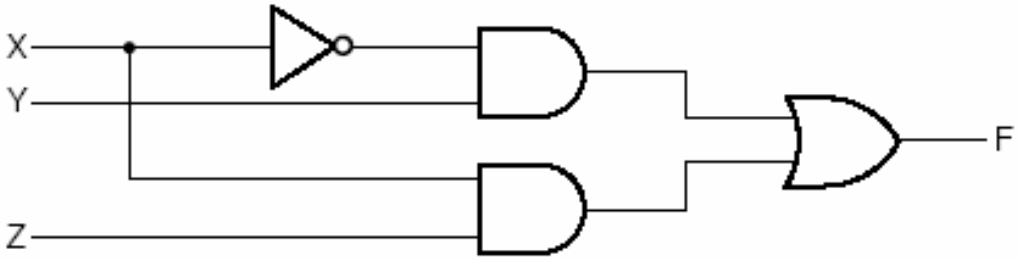
•Algebraic manipulations are useful for simplifying digital circuits.

Boolean Algebra: Terms and Literals

Literal is a single variable within a term that may or may not be complemented.



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$



(b) $F = \overline{X}Y + XZ$

Fig. 2-4 Implementation of Boolean Function with Gates

3 terms and 8 literals

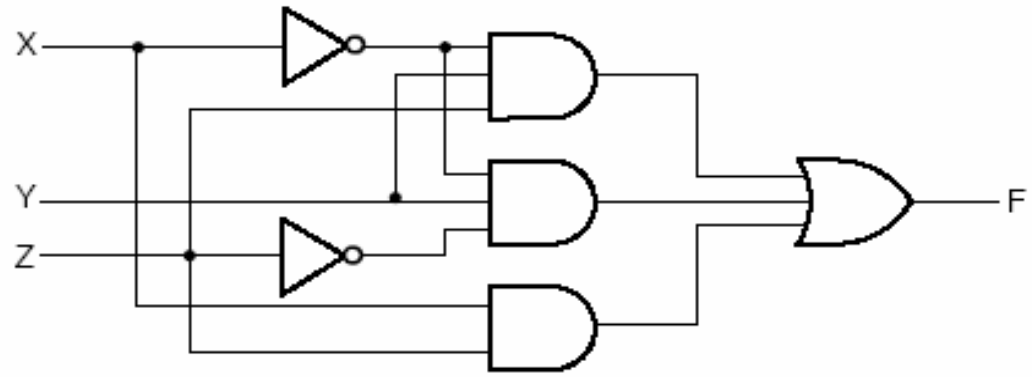
2 terms and 4 literals

Boolean Algebra: Simplifications

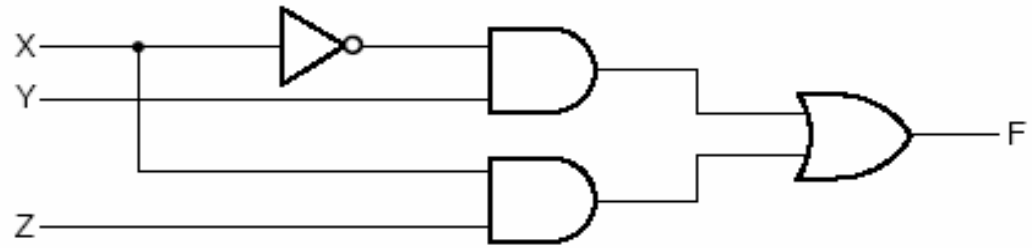
Both (a) and (b) give the same result F:

$$F = X'YZ+X'YZ'+XZ = X'Y(Z+Z')+XZ = X'Y+XZ$$

But, (b) needs fewer number of gates, so is area efficient.



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$



(b) $F = \overline{X}Y + XZ$

Fig. 2-4 Implementation of Boolean Function with Gates

Boolean Algebra: Complement of a Function

•Complement of a function is obtained by interchanging "0"s to "1"s and "1"s to "0"s.

•It can be obtained algebraically by applying DeMorgan's theorem.

•The general DeMorgan's theorem: Complement of a function can be obtained by interchanging AND and OR operations and complementing each variable and constant.

• Another method of getting complement: Take the dual of the function and complement each literal.

Boolean Algebra: Complementing Example

Find complement of the function, $F = X'YZ' + X'Y'Z$

Method 1: (Using DeMorgan's theorem)

$F' = (X'YZ' + X'Y'Z)' = (X'YZ')' \ (X'Y'Z)'$

$= (X + Y' + Z)\ (X + Y + Z')$

Method 2: (Using Duality)

We have, $F = X'YZ' + X'Y'Z = (X'YZ') + (X'Y'Z)$ (use parentheses to avoid confusion)

Dual of $F = (X' + Y + Z')\ (X' + Y' + Z)$

Complementing each literal : $(X + Y' + Z)\ (X + Y + Z') = F'$

Standard Forms of Boolean Functions

- Standard forms help in simplifying the Boolean expressions and frequently results in more desirable logic circuits.
- Standard forms contain product terms and sum terms.
  - **product terms** (example X'YZ)
  - **sum terms** (example X+Y+Z')
- In Boolean algebra, product means logical AND operation and sum means logical OR operation.
- Two standard forms Standard forms of Boolean functions are:
  - **Sum-of-Products (SOP)**
  - **Product-of-Sums (POS)**
- Product terms are called minterms and sum terms are called maxterms.

Standard Forms: Minterms

- A product term in which all the variables appear exactly once, either complemented or uncomplemented is called a minterm.
- Property: It represents exactly one combination of the binary variables in a truth table. It has the value *1* for that combination and *0* for all others.
- For n variables there are 2^n distinct minterms.
- Example: Given two variables X, Y, the four minterms are X'Y', X'Y, XY' and XY.
- The symbol for a minterm is m_j, where j denotes the decimal equivalent for which the minterm has the value of 1.

Standard Forms: Minterms (for 3 variables)

X	Y	Z	Product Term	Symbol	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	$\bar{X}\bar{Y}\bar{Z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\bar{X}\bar{Y}Z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\bar{X}Y\bar{Z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\bar{X}YZ$	m_3	0	0	0	1	0	0	0	0
1	0	0	$X\bar{Y}\bar{Z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$X\bar{Y}Z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$XY\bar{Z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	XYZ	m_7	0	0	0	0	0	0	0	1

Table 2-6 Minterms for Three Variables

The binary numbers from 000 to 111 are listed under variables. For each binary combination there is a related minterm. Each term is the product of exactly 3 literals. A literal is a complemented variable if the corresponding bit of the related binary combination is 0 and it is uncomplemented variable if it is 1. The table is shown above can be extended for any *n* variables.

Standard Forms: Maxterms

- A sum term that contains all the variables in complemented or uncomplemented form is called a maxterm.
- For n variables there are 2^n distinct maxterms.
- Each maxterm is the logical sum of the variables, with each variable being complemented if the corresponding bit of the binary number is 1 and uncomplemented if it is 0.
- The symbol for a maxterm is M_j where j denotes the decimal equivalent of the binary combination for which the maxterm has the value 0.
- Example: Given two variables X, Y, the four maxterms are (X+Y), (X+Y'), (X'+Y), and (X'+Y').

Standard Forms: Maxterms (for 3 variables)

X	Y	Z	Sum Term	Symbol	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	$X+Y+Z$	M_0	0	1	1	1	1	1	1	1
0	0	1	$X+Y+\overline{Z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$X+\overline{Y}+Z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$X+\overline{Y}+\overline{Z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\overline{X}+Y+Z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\overline{X}+Y+\overline{Z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\overline{X}+\overline{Y}+Z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\overline{X}+\overline{Y}+\overline{Z}$	M_7	1	1	1	1	1	1	1	0

Table 2-7 Maxterms for Three Variables

The binary numbers from 000 to 111 are listed under variables. For each binary combination there is a related maxterm. Each term is the sum of exactly 3 literals. A literal is a complemented variable if the corresponding bit of the related binary combination is 1 and it is uncomplemented variable if it is 0. The table is shown above can be extended for any *n* variables.

Note: $M_j = m_j'$

Standard Forms: Sum-of-Minterms

A function of three variables.

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A Boolean function can be expressed algebraically by logically summing all the minterms that produce a "1" in the function. This expression is called a "sum of minterms". The function shown in the left side, can be expressed as follows.

$$F = m_0 + m_2 + m_5 + m_7$$

$$= \Sigma\, m(0,2,5,7)$$

The symbol Σ denotes logical sum (OR).

Standard Forms: Product-of-Maxterms

A function of three variables.

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A Boolean function can be expressed algebraically by finding logical product of all the maxterms that produce a "0" in the function. This expression is called a "product of maxterms". The function shown in the left side, can be expressed as follows.

$$F = M_1 \cdot M_3 \cdot M_4 \cdot M_6$$

$$= \Pi\, M(1,3,4,6)$$

The symbol ⊓ denotes logical product (AND).

Standard Forms: Sum-of-Products and Product-of-Sums

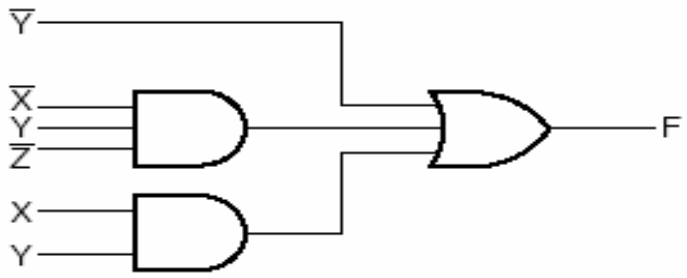


Fig. 2-5 Sum-of-Products Implementation

Sum-of-Products (SOP) is the simplified form of the Sum-of-Minterms. An SOP expression is equal to 1 only if one or more of the product terms in the expression is equal to one.

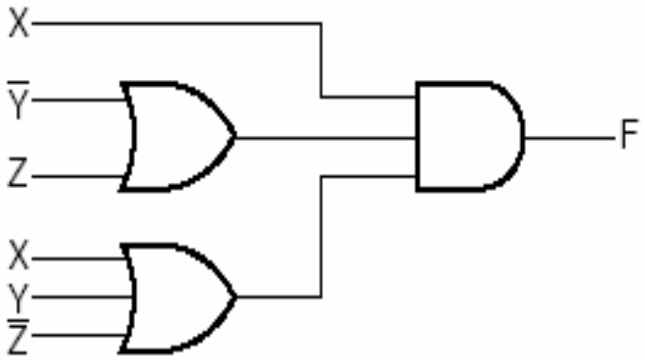


Fig. 2-7 Product-of-Sums Implementation

Product-of-Sums (POS) is the simplified form of Product-of-Maxterms. A POS expression is equal to 0 only if one or more of the sum terms in the expression is equal to zero.

Map Simplification

- We need to express a function in simplified form to reduce the number of gates needed for its implementation, thus reducing the overall area of the chip to be fabricated.
- A function has an unique truth table representation, but it can have various algebraic representations.
- Boolean expressions can be simplified by algebraic manipulations, but this is a complicated procedure.
- The map procedure is much simpler.
- The map is known as Karnaugh map or K-map.
- The K-map is a diagram made up of squares, with each square representing one minterm (or a maxterm) of the function.

Map Simplification: Two-Variable K-Map

There are four minterms for a Boolean function with two variables. Hence, the K-map consists of four squares, one for each minterm.

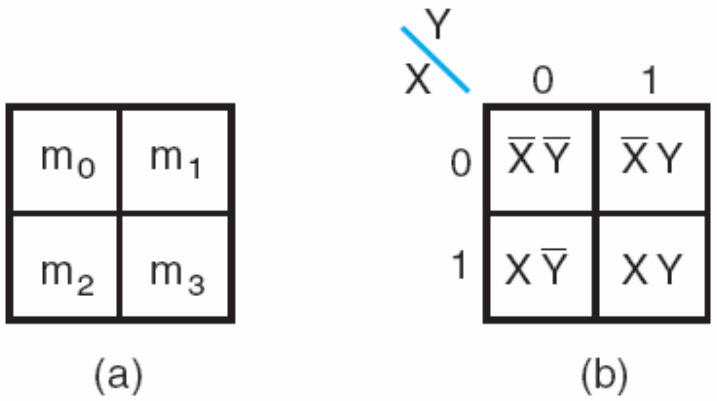


Fig. 2-8 Two-Variable Map

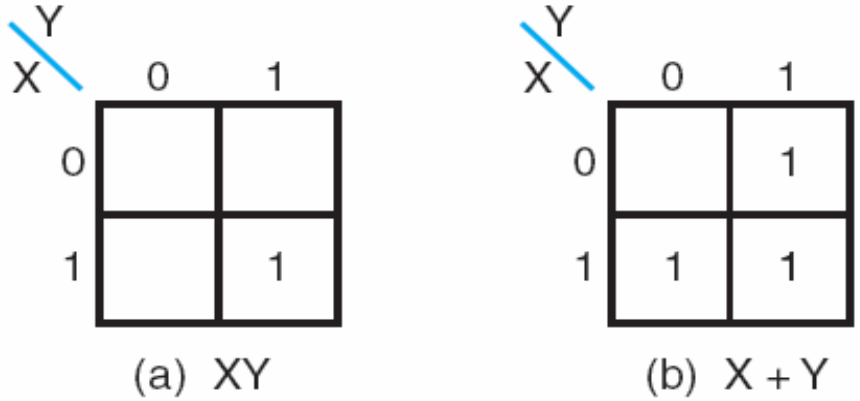


Fig. 2-9 Representation of Functions in the Map

X+Y in (b) obtained after simplification.

Map Simplification: Three Variable Map

There are eight minterms for a Boolean function with three variables. Hence, the K-map consists of eight squares, one for each minterm.

Fig. 2-10 Three-Variable Map

Map Simplification: Three Variable Map (To Remember)

- More than one squares are combined to obtain products with fewer literals.
- One square represents a minterm of three literals.
- A rectangle of two squares represents a product term of two literals.
- A rectangle/square of four squares represents a product term of one literal.
- A rectangle of eight squares covers the entire map and produces a function that is always 1.

NOTE: Similar rules can be formulated for two/four/n-variable K-map.

Map Simplification: Three Variable Map Example **1**

Example: F(X,Y,Z) = Σ m(2,3,4,5)

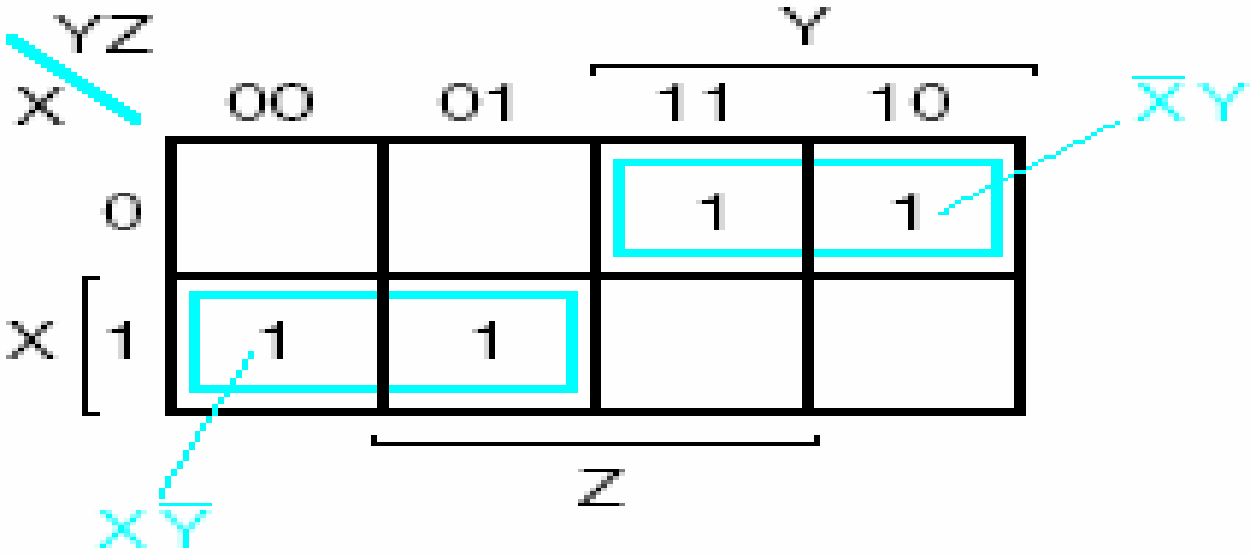


Fig. 2-11 Map for Example 2-3

After simplification we obtain, F(X,Y,Z) = X'Y + XY'

Map Simplification: Three Variable Map Example 2

Example: $F(X,Y,Z) = \Sigma\, m(0,2,4,6)$

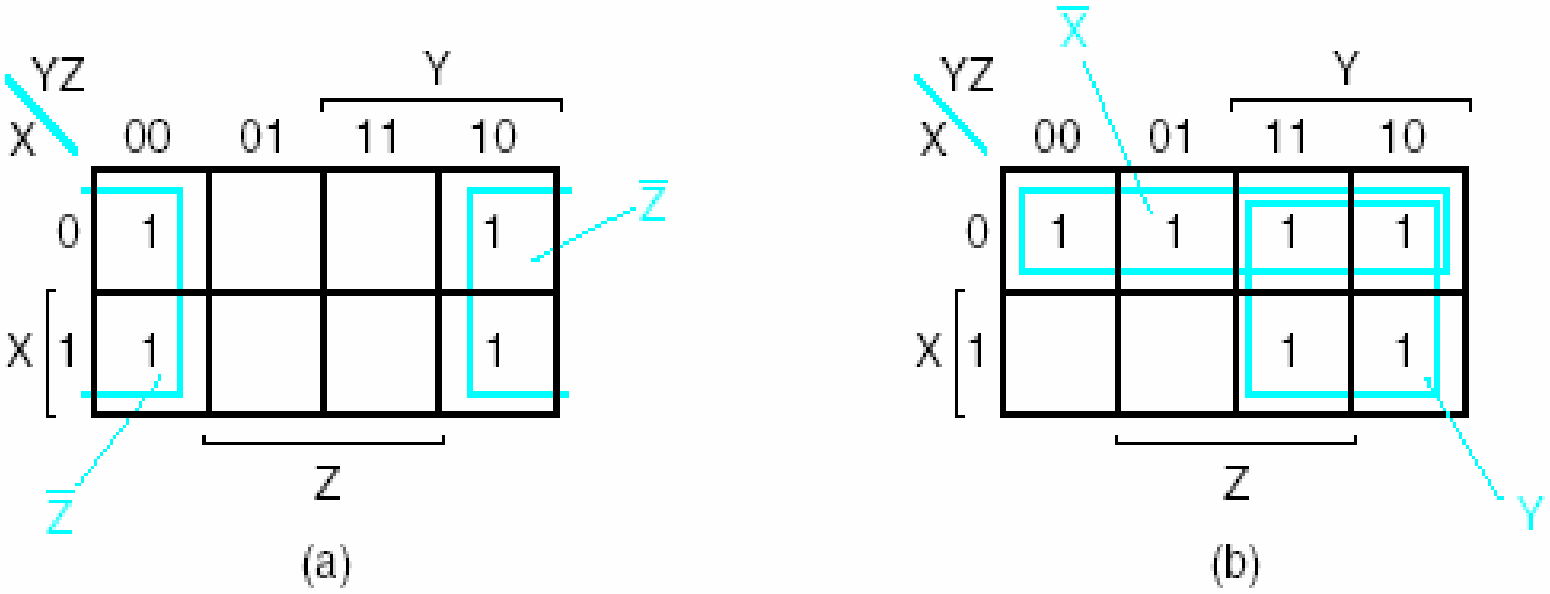
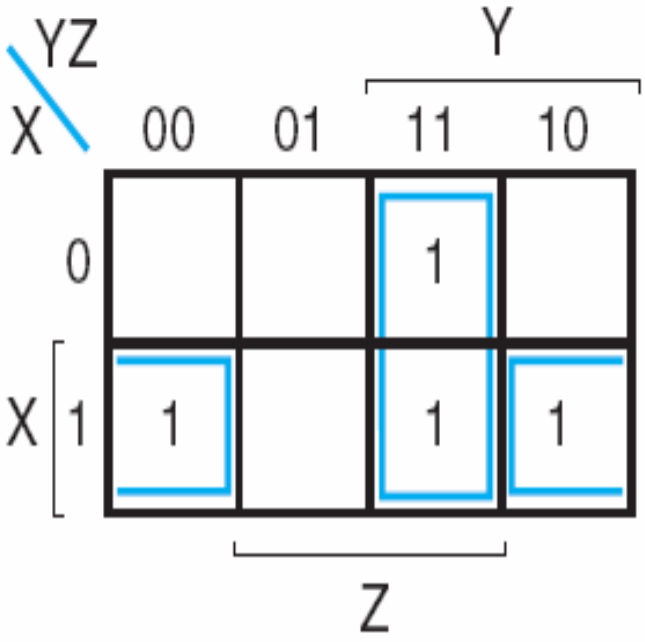


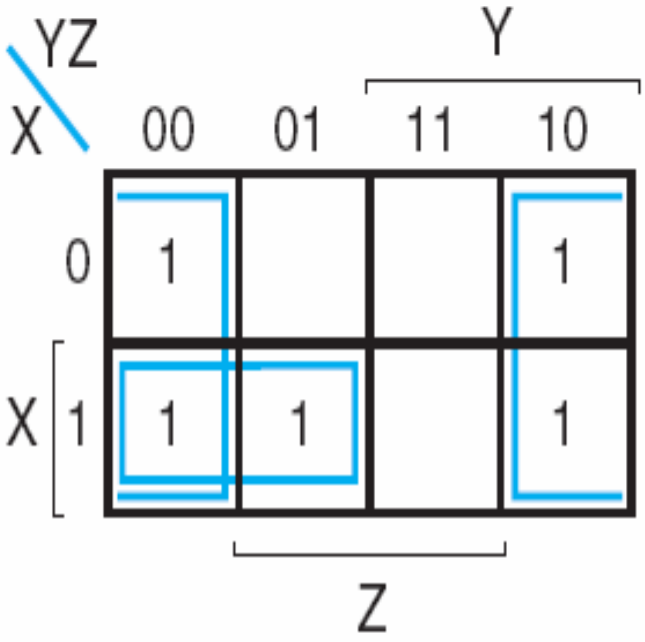
Fig. 2-13 Product Terms Using Four Minterms

After simplification we obtain, $F(X,Y,Z) = Z'$

Map Simplification: Three Variable Map Example **3**



(a) $F_1(X, Y, Z) = \Sigma m(3, 4, 6, 7)$
$= YZ + X\overline{Z}$



(b) $F_2(X, Y, Z) = \Sigma m(0, 2, 4, 5, 6)$
$= \overline{Z} + X\overline{Y}$

Fig. 2-14 Maps for Example 2-4

Map Simplification: Three Variable Map Example **4**

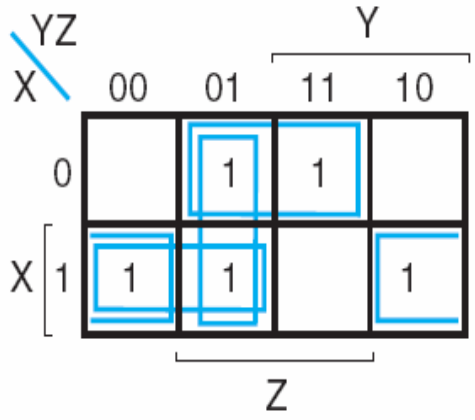


Fig. 2-15 $F(X,Y,Z) = \Sigma m(1, 3, 4, 5, 6)$

$$F(X,Y,Z) = X'Z + XZ' + XY'$$
$$= X'Z + XZ' + Y'Z$$

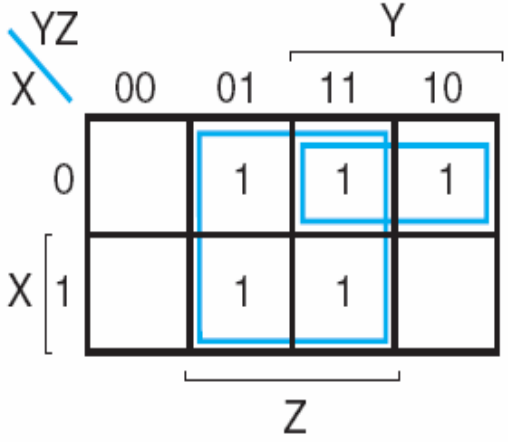


Fig. 2-16 $F(X,Y,Z) = \Sigma m\ (1,2,3,5,7)$

$$F(X,Y,Z) = Z + X'Y$$

Map Simplification: Four Variable Map

There are sixteen minterms for a Boolean function with four variables; the K-map consists of sixteen squares, one for each minterm.

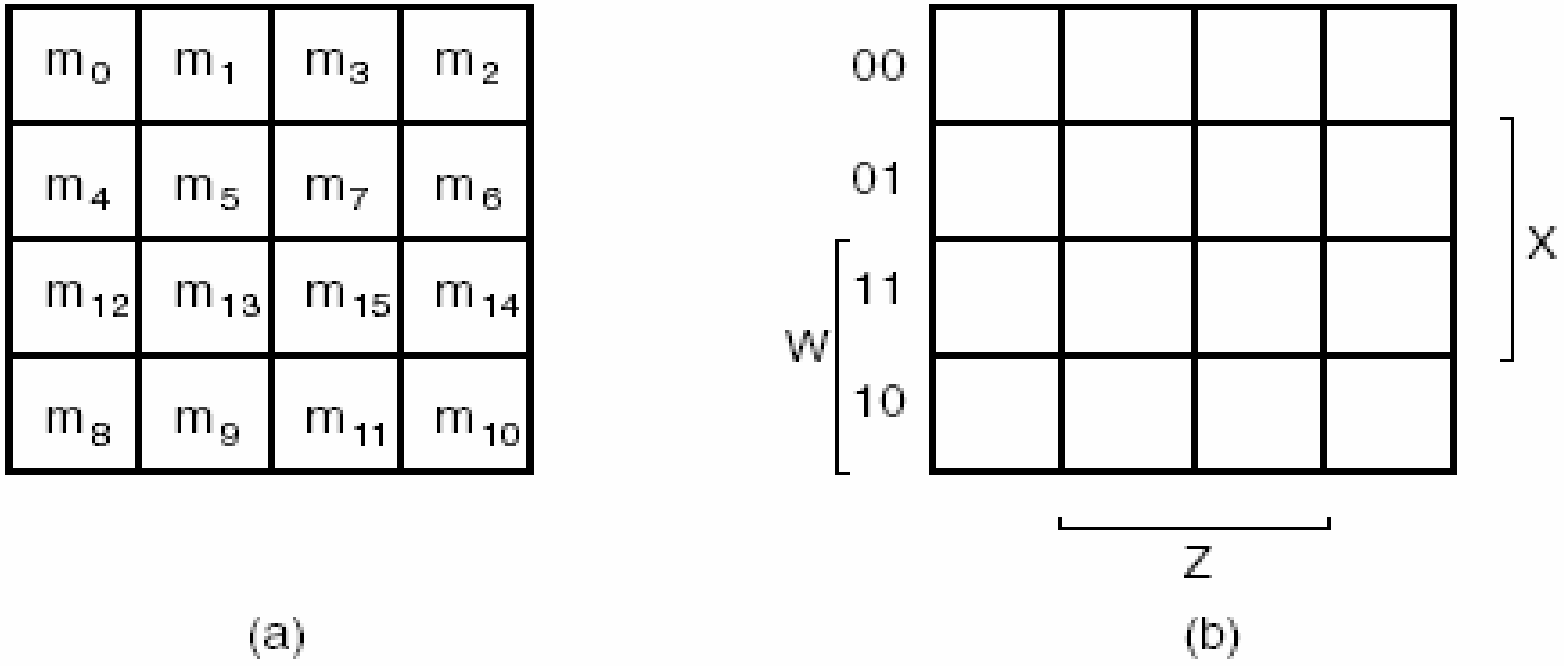
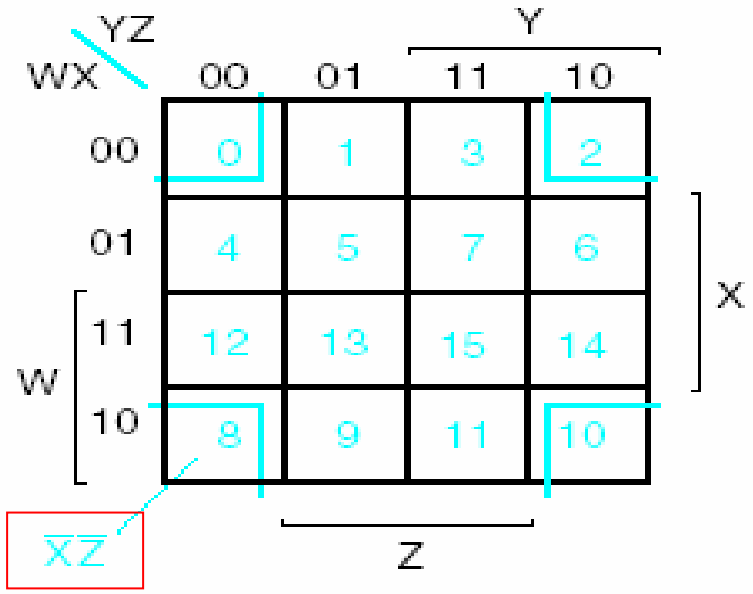


Fig. 2-17 Four-Variable Map

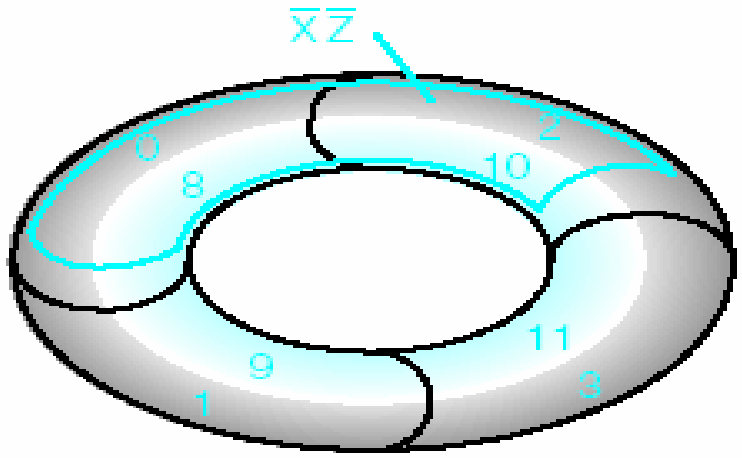
Map Simplification: Four Variable Map (To Remember)

- More than one squares are combined to obtain products with fewer literals.
- One square represents a minterm of four literals.
- A rectangle of two squares represents a product term of three literals.
- A rectangle/square of four squares represents a product term of two literal.
- A rectangle of eight squares represents a product term of one literals.
- A rectangle of sixteen squares covers the entire map and produces a function that is always 1.

Map Simplification: Four Variable Map Example **1**



(a)



(b)

Fig. 2-18 Four-Variable Map: Flat and on a Torus to Show Adjacencies

Map Simplification: Four Variable Map Example **2**

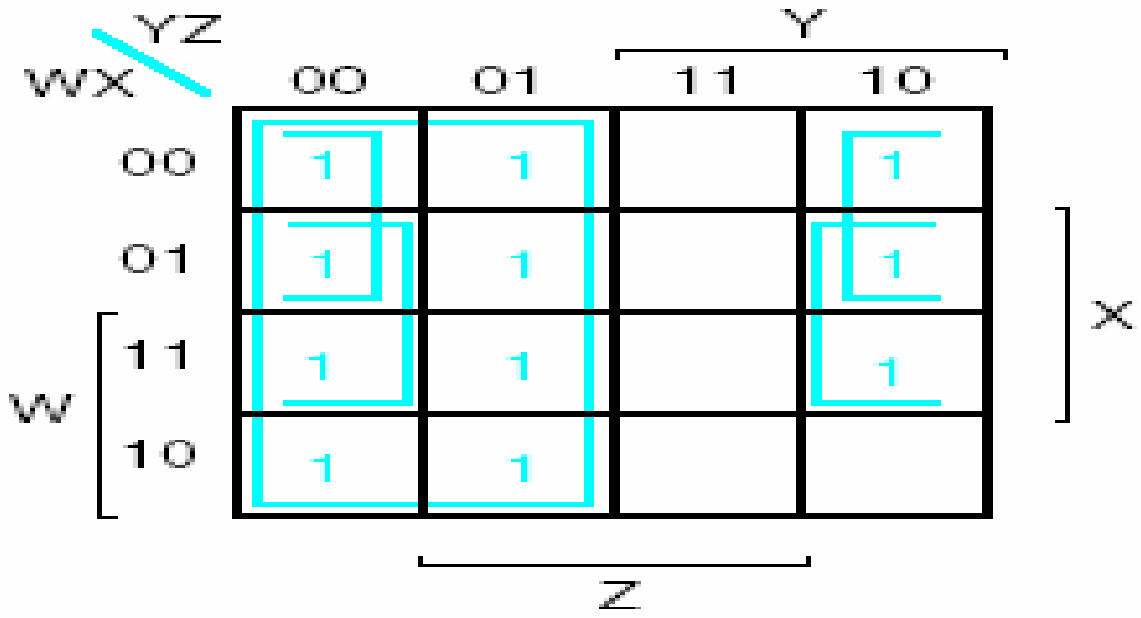


Fig. 2-19 Map for Example 2-5: $F = \overline{Y} + \overline{W}\overline{Z} + X\overline{Z}$

Map Simplification: Four Variable Map Example **3**

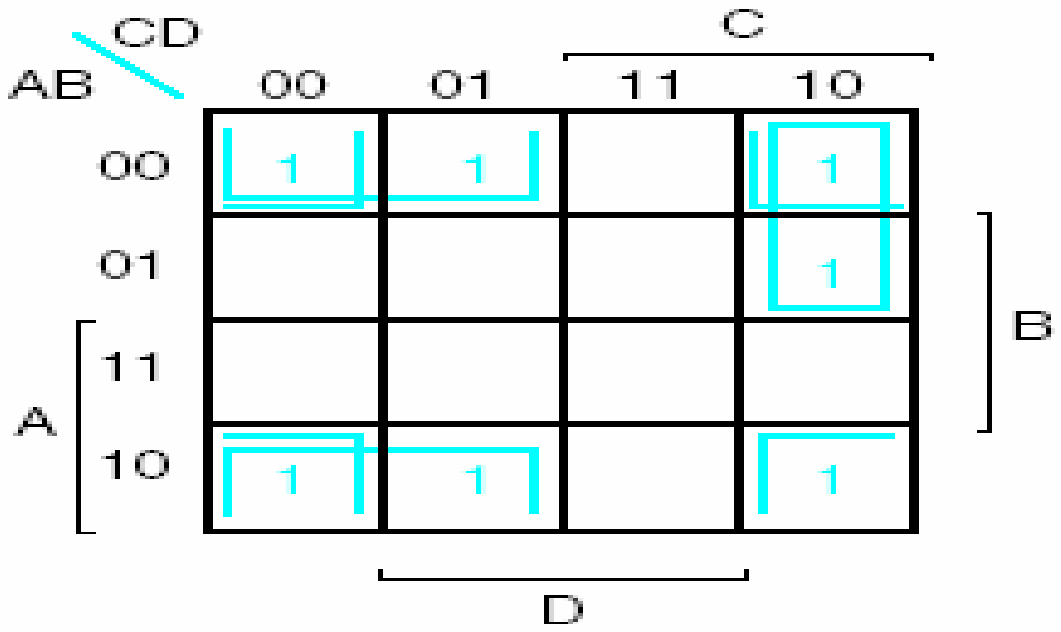


Fig. 2-20 Map for Example 2-6: $F = \overline{B}\overline{D} + \overline{B}\overline{C} + \overline{A}C\overline{D}$

Map Simplification: Four Variable Map Example **4**

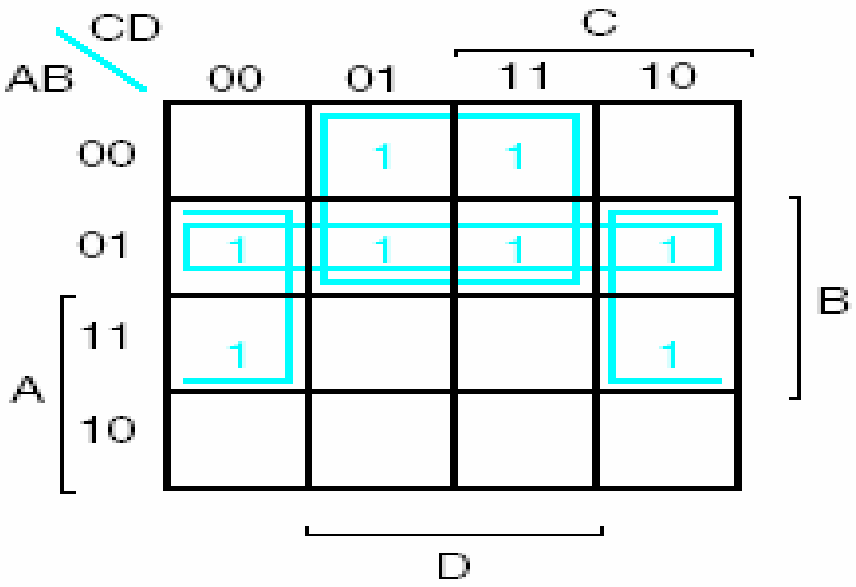


Fig. 2-21 Prime Implicants for Example 2-7: $\bar{A}D$, $B\bar{D}$, and $\bar{A}B$