Ch1: Digital Systems and Binary Numbers
**Ch2: Boolean Algebra and Logic Gates**
Ch3: Gate-Level Minimization
Ch4: Combinational Logic
Ch5: Synchronous Sequential Logic
Ch6: Registers and Counters

# Chapter 2
# Boolean Algebra and Logic Gates

Switching Theory & Logic Design
1403271-4

Prof. Adnan Gutub

Main Ref: M. Morris Mano and Michael D. Ciletti, Digital Design, Prentice Hall

## Content

## 2.1 INTRODUCTION
## Binary Logic and Gates

- Binary variables take on one of two values.
- Logical operators operate on binary values and binary variables.
- Basic logical operators are the logic functions AND, OR and NOT.
- Logic gates implement logic functions.
- Boolean Algebra: a useful mathematical system for specifying and transforming logic functions.
- We study Boolean algebra as a foundation for designing and analyzing digital systems!




## Binary Variables

- Recall that the two binary values have different names:
  - True/False
  - On/Off
  - Yes/No
  - 1/0
- We use 1 and 0 to denote the two values.
- Variable identifier examples:
  - A, B, y, z, or $X_1$ for now
  - RESET, START_IT, or ADD1 later

# Logical Operations

- The three basic logical operations are:
  - AND
  - OR
  - NOT
- AND is denoted by a dot (·).
- OR is denoted by a plus (+).
- NOT is denoted by an overbar ( ¯ ), a single quote mark (') after, or (~) before the variable.





# 2.2 Basic Definitions

- *Closure*
- *Associative law* : *(x * y) * z = x * (y * z)*
- *Commutative law : x * y = y * x*
- *Identity element :*
- *Inverse*
- *Distributive law : x * (y . z) = (x * y) . (x * z)*

## 2.3 Axiomatic Definition of Boolean Algebra

- In 1854, George Boole developed an algebraic system now called *Boolean algebra*
- Boolean algebra is an algebraic structure with two binary operators: + and .

7

## Logic Diagrams and Expressions

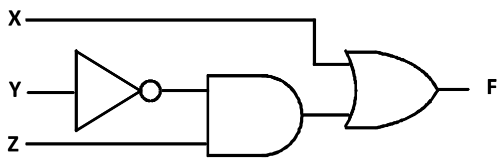
**Truth Table**

| X Y Z | $F = X + \overline{Y} \cdot Z$ |
|---|---|
| 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 |

**Equation**

$$F = X + \overline{Y}\ Z$$

**Logic Diagram**




- **Boolean equations, truth tables and logic diagrams describe the same function!**
- **Truth tables are unique; expressions and logic diagrams are not. This gives flexibility in implementing functions.**

Chapter 2 - 8

# Boolean Algebra

An algebraic structure defined on a set of at least two elements, B, together with three binary operators (denoted +, · and ) that satisfies the following basic identities:

| | | | |
|---|---|---|---|
| 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. | $(XY)Z = X(YZ)$ | 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 |





# Notation Examples

- Examples:
  - Y = A · B is read "Y is equal to A AND B."
  - z = x + y is read "z is equal to x OR y."
  - X = $\overline{A}$ is read "X is equal to NOT A."

Note: The statement:

1 + 1 = 2 (read "one <u>plus</u> one equals two")

is not the same as

1 + 1 = 1 (read "1 <u>or</u> 1 equals 1").

# Operator Definitions

Operations are defined on the values "0" and "1" for each operator:

| AND | OR | NOT |
|---|---|---|
| $0 \cdot 0 = 0$ | $0 + 0 = 0$ | $\overline{0} = 1$ |
| $0 \cdot 1 = 0$ | $0 + 1 = 1$ | $\overline{1} = 0$ |
| $1 \cdot 0 = 0$ | $1 + 0 = 1$ | |
| $1 \cdot 1 = 1$ | $1 + 1 = 1$ | |




# Truth Tables

- ***Truth table*** **– a tabular listing of the values of a function for all possible combinations of values on its arguments**
- **Example: Truth tables for the basic logic operations:**

| AND | | |
|---|---|---|
| X | Y | $Z = X \cdot Y$ |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| OR | | |
|---|---|---|
| X | Y | $Z = X + Y$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| NOT | |
|---|---|
| X | $Z = \overline{X}$ |
| 0 | 1 |
| 1 | 0 |

## Example 1: Boolean Algebraic Proof

- $A + A \cdot B = A$ (Absorption Theorem)

| Proof Steps | Justification (identity or theorem) |
|---|---|
| $A + A \cdot B$ | |
| $= A \cdot 1 + A \cdot B$ | $X = X \cdot 1$ |
| $= A \cdot (1 + B)$ | $X \cdot Y + X \cdot Z = X \cdot (Y + Z)$ (Distributive Law) |
| $= A \cdot 1$ | $1 + X = 1$ |
| $= A$ | $X \cdot 1 = X$ |

- Our primary reason for doing proofs is to learn:
  - Careful and efficient use of the identities and theorems of Boolean algebra, and
  - How to choose the appropriate identity or theorem to apply to make forward progress, irrespective of the application.





## Example 2: Boolean Algebraic Proofs

- $AB + \overline{A}C + BC = AB + \overline{A}C$ (Consensus Theorem)

| Proof Steps | Justification (identity or theorem) |
|---|---|
| $AB + \overline{A}C + BC$ | |
| $= AB + \overline{A}C + 1 \cdot BC$ | ? |
| $= AB + \overline{A}C + (A + \overline{A}) \cdot BC$ | ? |
| $=$ | |

## Example 3: Boolean Algebraic Proofs

- $(\overline{X+Y})Z + X\overline{Y} = \overline{Y}(X+Z)$

**Proof Steps** **Justification (identity or theorem)**

$(\overline{X+Y})Z + X\overline{Y}$

$=$





2.4 Basic Theorems and Properties of Boolean Algebra

## Useful Theorems

- $x \cdot y + \overline{x} \cdot y = y$ $(x+y)(\overline{x}+y) = y$ **Minimization**
- $x + x \cdot y = x$ $x \cdot (x+y) = x$ **Absorption**
- $x + \overline{x} \cdot y = x + y$ $x \cdot (\overline{x}+y) = x \cdot y$ **Simplification**
- $x \cdot y + \overline{x} \cdot z + y \cdot z = x \cdot y + \overline{x} \cdot z$ **Consensus**
  $(x+y) \cdot (\overline{x}+z) \cdot (y+z) = (x+y) \cdot (\overline{x}+z)$
- $\overline{x+y} = \overline{x} \cdot \overline{y}$ $\overline{x \cdot y} = \overline{x} + \overline{y}$ **DeMorgan's Laws**

## 2.4 Basic Theorems and Properties of Boolean Algebra

# Theorems

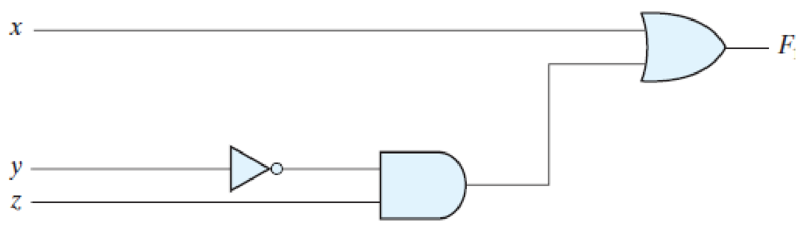
**Table 2.1**
*Postulates and Theorems of Boolean Algebra*

| | | | | |
|---|---|---|---|---|
| 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, 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) | $x(x + y) = x$ |

Chapter 2 - 17

# 2.5 Boolean Functions

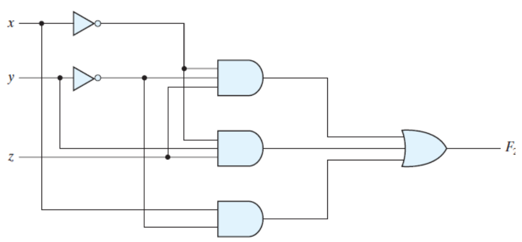
- Study: *F = x + y' z*
  - *Derive Truth Table*
  - *Draw Logic Circuit*



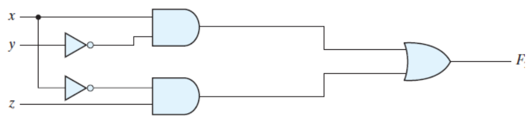

18

# Boolean Simplification

*F = x'y'z + x'yz + xy'*




*F = x'y'z + x'yz + xy'*
*= x'z(y' + y) + xy'*
*= x'z + xy'*




19

# 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. $x + x'y = (x + x')(x + y) = 1(x + y) = x + y.$
3. $(x + y)(x + y') = x + xy + xy' + yy' = x(1 + y + y') = x.$
4. $xy + x'z + yz = xy + x'z + yz(x + x')$
   $= xy + x'z + xyz + x'yz$
   $= xy(1 + z) + x'z(1 + y)$
   $= xy + x'z.$
5. $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$, by duality from function 4.

20

# Complement of Function

$$
\begin{aligned}
(A + B + C)' &= (A + x)' && \text{let } B + C = x \\
&= A'x' && \text{by theorem 5(a) (DeMorgan)} \\
&= A'(B + C)' && \text{substitute } B + C = x \\
&= A'(B'C') && \text{by theorem 5(a) (DeMorgan)} \\
&= A'B'C' && \text{by theorem 4(b) (associative)}
\end{aligned}
$$

- Example 2.2: Find complement of F1 and F2
- F1 = x'yz'+x'y'z
- F2 = x(y'z'+yz)

21

## Some Properties of Identities & the Algebra

- **Unless it happens to be self-dual, the dual of an expression does not equal the expression itself.**
- **Example: $F = (A + \overline{C}) \cdot B + 0$**

  **dual $F = (A \cdot \overline{C} + B) \cdot 1 = A \cdot \overline{C} + B$**
- **Example: $G = X \cdot Y + (\overline{W + Z})$**

  **dual G =**
- **Example: $H = A \cdot B + A \cdot C + B \cdot C$**

  **dual H =**
- **Are any of these functions self-dual?**



Chapter 2 - 22

# Expression Simplification

- **An application of Boolean algebra**
- **Simplify to contain the smallest number of literals (complemented and uncomplemented variables):**

$$A B + \overline{A} C D + \overline{A} B D + \overline{A} C \overline{D} + A B C D$$

$$= AB + ABCD + \overline{A} C D + \overline{A} C \overline{D} + \overline{A} B D$$

$$= AB + AB(CD) + \overline{A} C (D + \overline{D}) + \overline{A} B D$$

$$= AB + \overline{A} C + \overline{A} B D = B(A + \overline{A}D) + \overline{A}C$$

$$= B (A + D) + \overline{A} C \quad \text{5 literals}$$





# Boolean Function Evaluation

$$F1 = x y \overline{z}$$
$$F2 = x + \overline{y} z$$
$$F3 = \overline{x}\,\overline{y}\,\overline{z} + \overline{x}\, y\, z + x \overline{y}$$
$$F4 = x \overline{y} + \overline{x} z$$

| x | y | z | F1 | F2 | F3 | F4 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 1 | 0 | 1 | | |
| 0 | 1 | 0 | 0 | 0 | | |
| 0 | 1 | 1 | 0 | 0 | | |
| 1 | 0 | 0 | 0 | 1 | | |
| 1 | 0 | 1 | 0 | 1 | | |
| 1 | 1 | 0 | 1 | 1 | | |
| 1 | 1 | 1 | 0 | 1 | | |

## 2.6 Canonical and Standard Forms

- **What are Canonical Forms?**
- **Minterms and Maxterms**
- **Index Representation of Minterms and Maxterms**
- **Sum-of-Minterm (SOM) Representations**
- **Product-of-Maxterm (POM) Representations**
- **Representation of Complements of Functions**
- **Conversions between Representations**




## Canonical Forms

- **It is useful to specify Boolean functions in a form that:**
  - **Allows comparison for equality.**
  - **Has a correspondence to the truth tables**
- **Canonical Forms in common usage:**
  - **Sum of Minterms (SOM)**
  - **Product of Maxterms (POM)**

# Minterms

- <u>Minterms</u> are AND terms with every variable present in either true or complemented form.
- Given that each binary variable may appear normal (e.g., x) or complemented (e.g., $\overline{x}$), there are $2^n$ minterms for $n$ variables.
- <u>Example:</u> Two variables (X and Y)produce 2 x 2 = 4 combinations:
  - $XY$ (both normal)
  - $X\overline{Y}$ (X normal, Y complemented)
  - $\overline{X}Y$ (X complemented, Y normal)
  - $\overline{X}\,\overline{Y}$ (both complemented)
- Thus there are <u>four minterms</u> of two variables.





# Maxterms

- <u>Maxterms</u> are OR terms with every variable in true or complemented form.
- Given that each binary variable may appear normal (e.g., x) or complemented (e.g., $\overline{x}$), there are $2^n$ maxterms for $n$ variables.
- <u>Example:</u> Two variables (X and Y) produce 2 x 2 = 4 combinations:
  - $X + Y$ (both normal)
  - $X + \overline{Y}$ (x normal, y complemented)
  - $\overline{X} + Y$ (x complemented, y normal)
  - $\overline{X} + \overline{Y}$ (both complemented)

# Maxterms and Minterms

- **Examples: Two variable minterms and maxterms.**

| Index | Minterm | Maxterm |
|---|---|---|
| 0 | $\overline{x}\,\overline{y}$ | $x + y$ |
| 1 | $\overline{x}\,y$ | $x + \overline{y}$ |
| 2 | $x\,\overline{y}$ | $\overline{x} + y$ |
| 3 | $x\,y$ | $\overline{x} + \overline{y}$ |

- **The index above is important for describing which variables in the terms are true and which are complemented.**




# Standard Order

- **Minterms and maxterms are designated with a subscript**
- **The subscript is a number, corresponding to a binary pattern**
- **The bits in the pattern represent the complemented or normal state of each variable listed in a standard order.**
- **All variables will be present in a minterm or maxterm and will be listed in the <u>same order</u> (usually alphabetically)**
- **Example: For variables a, b, c:**
  - **Maxterms: $(a + b + \overline{c})$, $(a + b + c)$**
  - **Terms: $(b + a + c)$, $a\,\overline{c}\,b$, and $(c + b + a)$ are NOT in standard order.**
  - **Minterms: $a\,\overline{b}\,c$, $a\ b\ c$, $\overline{a}\,\overline{b}\ c$**
  - **Terms: $(a + c)$, $\overline{b}\,c$, and $(\overline{a} + b)$ do not contain all variables**

## Purpose of the Index

- The index for the minterm or maxterm, expressed as a binary number, is used to determine whether the variable is shown in the true form or complemented form.
- For Minterms:
  - "1" means the variable is "Not Complemented" and
  - "0" means the variable is "Complemented".
- For Maxterms:
  - "0" means the variable is "Not Complemented" and
  - "1" means the variable is "Complemented".





## Index Example in Three Variables

- Example: (for three variables)
- Assume the variables are called X, Y, and Z.
- The standard order is X, then Y, then Z.
- The Index 0 (base 10) = 000 (base 2) for three variables). All three variables are complemented for minterm 0 ( $\overline{X}, \overline{Y}, \overline{Z}$) and no variables are complemented for Maxterm 0 (X,Y,Z).
  - Minterm 0, called $m_0$ is $\overline{X}\,\overline{Y}\,\overline{Z}$.
  - Maxterm 0, called $M_0$ is (X + Y + Z).
  - Minterm 6 ?
  - Maxterm 6 ?

## Index Examples – Four Variables

| Index $i$ | Binary Pattern | Minterm $m_i$ | Maxterm $M_i$ |
|---|---|---|---|
| 0 | 0000 | $\bar{a}\bar{b}\bar{c}\bar{d}$ | $a+b+c+d$ |
| 1 | 0001 | $\bar{a}\bar{b}\bar{c}d$ | ? |
| 3 | 0011 | ? | $a+b+\bar{c}+\bar{d}$ |
| 5 | 0101 | $\bar{a}b\bar{c}d$ | $a+\bar{b}+c+\bar{d}$ |
| 7 | 0111 | ? | $a+\bar{b}+\bar{c}+\bar{d}$ |
| 10 | 1010 | $a\bar{b}c\bar{d}$ | $\bar{a}+b+\bar{c}+d$ |
| 13 | 1101 | $ab\bar{c}d$ | ? |
| 15 | 1111 | $abcd$ | $\bar{a}+\bar{b}+\bar{c}+\bar{d}$ |




## Minterm and Maxterm Relationship

- **Review: DeMorgan's Theorem**
  $\overline{x \cdot y} = \bar{x} + \bar{y}$ **and** $\overline{x + y} = \bar{x} \cdot \bar{y}$
- **Two-variable example:**
  $M_2 = \bar{x} + y$ **and** $m_2 = x \cdot \bar{y}$
  **Thus $M_2$ is the complement of $m_2$ and vice-versa.**
- **Since DeMorgan's Theorem holds for *n* variables, the above holds for terms of *n* variables**
- **giving:**

$$M_i = \overline{m}_i \text{ and } m_i = \overline{M}_i$$

**Thus $M_i$ is the complement of $m_i$.**

# Observations

- **In the function tables:**
  - **Each minterm has one and only one 1 present in the $2^n$ terms (a minimum of 1s). All other entries are 0.**
  - **Each maxterm has one and only one 0 present in the $2^n$ terms All other entries are 1 (a maximum of 1s).**
- **We can implement any function by "ORing" the minterms corresponding to "1" entries in the function table. These are called the minterms of the function.**
- **We can implement any function by "ANDing" the maxterms corresponding to "0" entries in the function table. These are called the maxterms of the function.**
- **This gives us two canonical forms:**
  - **Sum of Minterms (SOM)**
  - **Product of Maxterms (POM)**

**for stating any Boolean function.**





# Minterm Function Example

- **Example: Find $F_1 = m_1 + m_4 + m_7$**
- **$F1 = \overline{x}\ \overline{y}\ z + x\ \overline{y}\ \overline{z} + x\ y\ z$**

| x y z | index | m1 | + | m4 | + | m7 | = F1 |
|---|---|---|---|---|---|---|---|
| 0 0 0 | 0 | 0 | + | 0 | + | 0 | = 0 |
| 0 0 1 | 1 | 1 | + | 0 | + | 0 | = 1 |
| 0 1 0 | 2 | 0 | + | 0 | + | 0 | = 0 |
| 0 1 1 | 3 | 0 | + | 0 | + | 0 | = 0 |
| 1 0 0 | 4 | 0 | + | 1 | + | 0 | = 1 |
| 1 0 1 | 5 | 0 | + | 0 | + | 0 | = 0 |
| 1 1 0 | 6 | 0 | + | 0 | + | 0 | = 0 |
| 1 1 1 | 7 | 0 | + | 0 | + | 1 | = 1 |

# Minterm Function Example

- $F(A, B, C, D, E) = m_2 + m_9 + m_{17} + m_{23}$
- $F(A, B, C, D, E) =$




# Maxterm Function Example

- **Example: Implement F1 in maxterms:**

$F_1 = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$

$F_1 = (x + y + z) \cdot (x + \bar{y} + z) \cdot (x + \bar{y} + \bar{z}) \cdot (\bar{x} + y + \bar{z}) \cdot (\bar{x} + \bar{y} + z)$

| x y z | i | $M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$ = F1 |
|---|---|---|
| 0 0 0 | 0 | 0 · 1 · 1 · 1 · 1 = 0 |
| 0 0 1 | 1 | 1 · 1 · 1 · 1 · 1 = 1 |
| 0 1 0 | 2 | 1 · 0 · 1 · 1 · 1 = 0 |
| 0 1 1 | 3 | 1 · 1 · 0 · 1 · 1 = 0 |
| 1 0 0 | 4 | 1 · 1 · 1 · 1 · 1 = 1 |
| 1 0 1 | 5 | 1 · 1 · 1 · 0 · 1 = 0 |
| 1 1 0 | 6 | 1 · 1 · 1 · 1 · 0 = 0 |
| 1 1 1 | 7 | 1 · 1 · 1 · 1 · 1 = 1 |

## Maxterm Function Example

- $F(A, B, C, D) = M_3 \cdot M_8 \cdot M_{11} \cdot M_{14}$
- F(A, B,C,D) =





## Canonical Sum of Minterms

- Any Boolean function can be expressed as a <u>Sum of Minterms</u>.
  - For the function table, the <u>minterms</u> used are the terms corresponding to the 1's
  - For expressions, <u>expand</u> all terms first to explicitly list all minterms. Do this by “ANDing” any term missing a variable v with a term $(v + \overline{v})$.
- Example: Implement $f = x + \overline{x}\,\overline{y}$ as a sum of minterms.

  First expand terms: $f = x(y + \overline{y}) + \overline{x}\,\overline{y}$
  Then distribute terms: $f = xy + x\overline{y} + \overline{x}\,\overline{y}$
  Express as sum of minterms: $f = m_3 + m_2 + m_0$

# Another SOM Example

- Example: $F = A + \overline{B}\,C$
- There are three variables, A, B, and C which we take to be the standard order.
- Expanding the terms with missing variables:
- Collect terms (removing all but one of duplicate terms):
- Express as SOM:





# Shorthand SOM Form

- From the previous example, we started with:

  $F = A + \overline{B}\,C$
- We ended up with:

$F = m_1+m_4+m_5+m_6+m_7$
- This can be denoted in the formal shorthand:

  $F(A,B,C) = \Sigma_m(1,4,5,6,7)$
- Note that we explicitly show the standard variables in order and drop the "m" designators.

## Canonical Product of Maxterms

- Any Boolean Function can be expressed as a <u>Product of Maxterms (POM)</u>.
  - For the function table, the maxterms used are the terms corresponding to the 0's.
  - For an expression, expand all terms first to explicitly list all maxterms. Do this by first applying the second distributive law , "ORing" terms missing variable v with a term equal to $v \cdot \bar{v}$ and then applying the distributive law again.
- Example: Convert to product of maxterms:

$$f(x, y, z) = x + \bar{x}\,\bar{y}$$

Apply the distributive law:

$$x + \bar{x}\,\bar{y} = (x + \bar{x})(x + \bar{y}) = 1 \cdot (x + \bar{y}) = x + \bar{y}$$

Add missing variable z:

$$x + \bar{y} + z \cdot \bar{z} = (x + \bar{y} + z)(x + \bar{y} + \bar{z})$$

Express as POM: $f = M_2 \cdot M_3$





## Another POM Example

- Convert to Product of Maxterms:

$$f(A, B, C) = A\bar{C} + BC + \bar{A}\,\bar{B}$$

- Use $x + y\,z = (x+y)\cdot(x+z)$ with $x = (A\bar{C} + BC)$, $y = \bar{A}$, and $z = \bar{B}$ to get:

$$f = (A\bar{C} + BC + \bar{A})(A\bar{C} + BC + \bar{B})$$

- Then use $x + \bar{x}y = x + y$ to get:

$$f = (\bar{C} + BC + \bar{A})(A\bar{C} + C + \bar{B})$$

and a second time to get:

$$f = (\bar{C} + B + \bar{A})(A + C + \bar{B})$$

- Rearrange to standard order,

$$f = (\bar{A} + B + \bar{C})(A + \bar{B} + C) \text{ to give } f = M_5 \cdot M_2$$

# Function Complements

- **The complement of a function expressed as a sum of minterms is constructed by selecting the minterms missing in the sum-of-minterms canonical forms.**
- **Alternatively, the complement of a function expressed by a Sum of Minterms form is simply the Product of Maxterms with the same indices.**
- **Example: Given $F(x, y, z) = \Sigma_m(1,3,5,7)$**

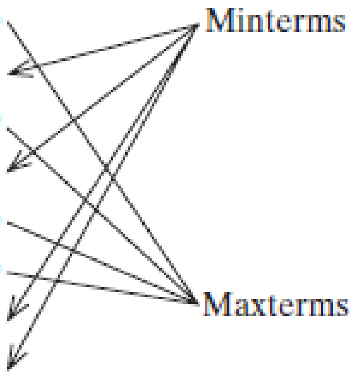
$$\overline{F}(x, y, z) = \Sigma_m(0,2,4,6)$$
$$\overline{F}(x, y, z) = \Pi_M(1,3,5,7)$$





**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 |



## Conversion Between Forms

- To convert between sum-of-minterms and product-of-maxterms form (or vice-versa) we follow these steps:
  - Find the function complement by swapping terms in the list with terms not in the list.
  - Change from products to sums, or vice versa.
- Example:Given F as before: $F(x,y,z) = \Sigma_m(1,3,5,7)$
- Form the Complement: $\overline{F}(x,y,z) = \Sigma_m(0,2,4,6)$
- Then use the other form with the same indices – this forms the complement again, giving the other form of the original function: $F(x,y,z) = \Pi_M(0,2,4,6)$




## Standard Forms

- Standard Sum-of-Products (SOP) form: equations are written as an OR of AND terms
- Standard Product-of-Sums (POS) form: equations are written as an AND of OR terms
- Examples:
  - SOP: $A\,B\,C + \overline{A}\,\overline{B}\,C + B$
  - POS: $(A+B)\cdot(A+\overline{B}+\overline{C})\cdot C$
- These "mixed" forms are neither SOP nor POS
  - $(A\,B + C)(A + C)$
  - $A\,B\,\overline{C} + A\,C\,(A+B)$

# Standard Sum-of-Products (SOP)

- **A sum of minterms form for *n* variables can be written down directly from a truth table.**
  - **Implementation of this form is a two-level network of gates such that:**
  - **The first level consists of *n*-input AND gates, and**
  - **The second level is a single OR gate (with fewer than $2^n$ inputs).**
- **This form often can be simplified so that the corresponding circuit is simpler.**




# Standard Sum-of-Products (SOP)

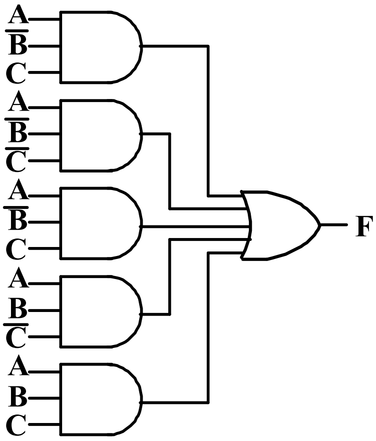
- **A Simplification Example:**
- $F(A,B,C) = \Sigma m(1,4,5,6,7)$
- **Writing the minterm expression:**

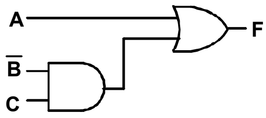
  $F = \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + AB\overline{C} + ABC$
- **Simplifying:**

  F =
- **Simplified F contains 3 literals compared to 15 in minterm F**

# AND/OR Two-level Implementation of SOP Expression

- The two implementations for F are shown below – it is quite apparent which is simpler!



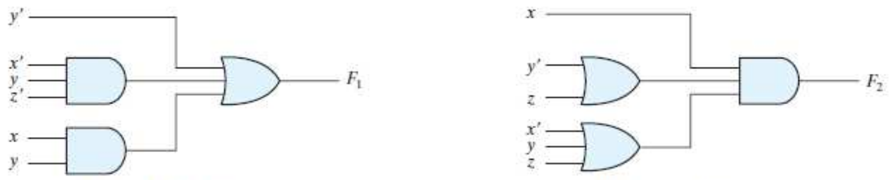





# SOP and POS Observations

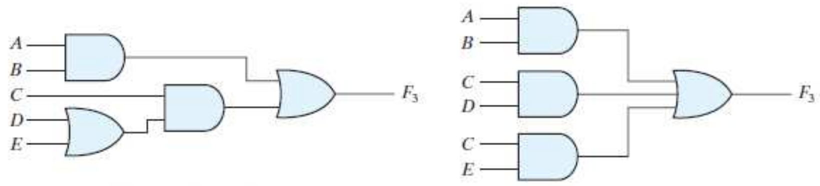
- The previous examples show that:
  - Canonical Forms (Sum-of-minterms, Product-of-Maxterms), or other standard forms (SOP, POS) differ in complexity
  - Boolean algebra can be used to manipulate equations into simpler forms.
  - Simpler equations lead to simpler two-level implementations
- Questions:
  - How can we attain a "simplest" expression?
  - Is there only one minimum cost circuit?
  - The next part will deal with these issues.

# Level's of implementation




(a) Sum of Products (b) Product of Sums

**FIGURE 2.3**
Two-level implementation




(a) $AB + C(D + E)$ (b) $AB + CD + CE$

**FIGURE 2.4**
Three- and two-level implementation

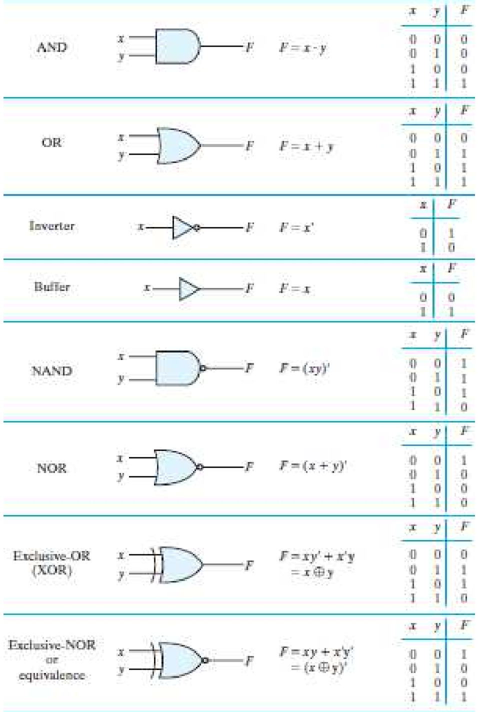
# 2.7 Other Logic Operations

**Table 2.8**
*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 |
| $F_9 = xy + x'y'$ | $(x \oplus y)'$ | 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 |

54

# 2.8 Digital Logic Gates




| | x y | F |
|---|---|---|
| AND | 00, 01, 10, 11 | 0, 0, 0, 1 |
| OR | 00, 01, 10, 11 | 0, 1, 1, 1 |
| Inverter | 0, 1 | 1, 0 |
| Buffer | 0, 1 | 0, 1 |
| NAND | 00, 01, 10, 11 | 1, 1, 1, 0 |
| NOR | 00, 01, 10, 11 | 1, 0, 0, 0 |
| XOR | 00, 01, 10, 11 | 0, 1, 1, 0 |
| XNOR | 00, 01, 10, 11 | 1, 0, 0, 1 |

55

# 2.9 Integrated Circuits

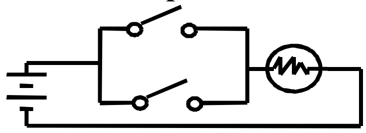
- In the earliest computers, switches were opened and closed by magnetic fields produced by energizing coils in relays. The switches in turn opened and closed the current paths.
- Later, vacuum tubes that open and close current paths electronically replaced relays.
- Today, transistors are used as electronic switches that open and close current paths.

56

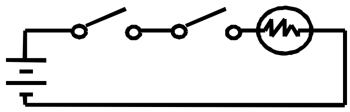
# Logic Function Implementation

- **Using Switches**
  - **For inputs:**
    - **logic 1 is switch closed**
    - **logic 0 is switch open**
  - **For outputs:**
    - **logic 1 is light on**
    - **logic 0 is light off.**
  - **NOT uses a switch such that:**
    - **logic 1 is switch open**
    - **logic 0 is switch closed**

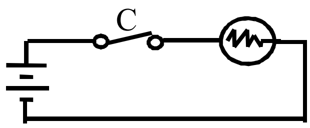
**Switches in parallel => OR**



**Switches in series => AND**



**Normally-closed switch => NOT**

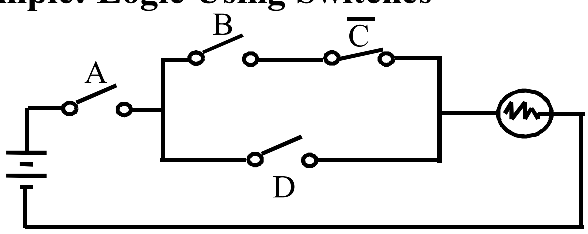







## Logic Function Implementation (Continued)

- **Example: Logic Using Switches**



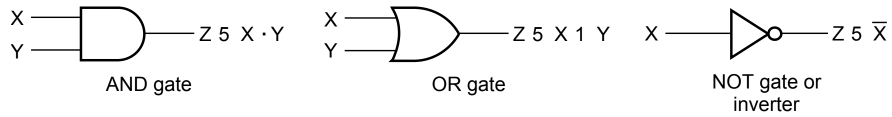

- **Light is on (L = 1) for**

  L(A, B, C, D) =

  **and off (L = 0), otherwise.**
- **Useful model for relay circuits and for CMOS gate circuits, the foundation of current digital logic technology**

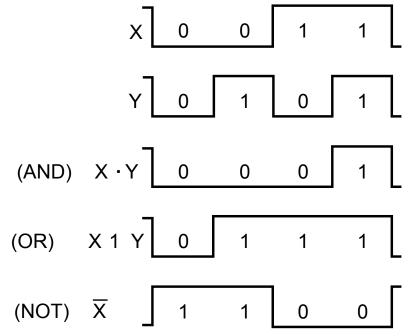
# Logic Gate Symbols and Behavior

- **Logic gates have special symbols:**




(a) Graphic symbols

- **And waveform behavior in time as follows:**




(b) Timing diagram





# Gate Delay

- **In actual physical gates, if one or more input changes causes the output to change, the output change does not occur instantaneously.**
- **The delay between an input change(s) and the resulting output change is the *gate delay* denoted by $t_G$:**