

In The Name Of Allah

Chapter 2

Algebraic Methods for the Analysis and Synthesis of Logic Circuits

Department of Electrical Engineering

Islamic Azad University

Qazvin Branch

2.1 Fundamentals of Boolean Algebra (1)

In 1849, George Boole presented

2.1.1 Basic Postulates

- **Postulate 1 (Definition):** A Boolean algebra is a closed algebraic system containing a set K of two or more elements and the two operators \cdot and $+$
- **Postulate 2 (Existence of 1 and 0 element):**
 - (a) $a + 0 = a$ (identity for $+$), (b) $a \cdot 1 = a$ (identity for \cdot)
- **Postulate 3 (Commutativity):**
 - (a) $a + b = b + a$, (b) $a \cdot b = b \cdot a$
- **Postulate 4 (Associativity):**
 - (a) $a + (b + c) = (a + b) + c$ (b) $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
- **Postulate 5 (Distributivity):**
 - (a) $a + (b \cdot c) = (a + b) \cdot (a + c)$ (b) $a \cdot (b + c) = a \cdot b + a \cdot c$
- **Postulate 6 (Existence of complement):**
 - (a) $a + \bar{a} = 1$ (b) $a \cdot \bar{a} = 0$
- Normally \cdot is omitted.

2.1 Fundamentals of Boolean Algebra (2)

2.1.2 Venn diagrams for postulates

2.1.3 Duality

The dual expression is found by replacing all + with . And all ones with zeros and all zeros with ones.

2.1.4 Fundamental Theorems of Boolean Algebra

➤ Theorem 1 (Idempotency):

$$(a) \ a + a = a \qquad (b) \ aa = a$$

➤ Theorem 2 (Null element):

$$(a) \ a + 1 = 1 \qquad (b) \ a0 = 0$$

➤ Theorem 3 (Involution)

$$\overline{\overline{a}} = a$$

➤ Properties of 0 and 1 elements (Table 2.1):

OR	AND	Complement
$a + 0 = 0$	$a0 = 0$	$0' = 1$
$a + 1 = 1$	$a1 = a$	$1' = 0$

2.1 Fundamentals of Boolean Algebra (3)

➤ Theorem 4 (Absorption)

$$(a) \quad a + ab = a \qquad (b) \quad a(a + b) = a$$

➤ Examples:

$$\blacksquare (X + Y) + (X + Y)Z = X + Y \quad [\text{T4(a)}]$$

$$\blacksquare AB'(AB' + BC) = AB' \quad [\text{T4(b)}]$$

➤ Theorem 5

$$(a) \quad a + a'b = a + b \qquad (b) \quad a(a' + b) = ab$$

➤ Examples:

$$\blacksquare B + AB'CD = B + AC'D \quad [\text{T5(a)}]$$

$$\blacksquare (X + Y)((X + Y)' + Z) = (X + Y)Z \quad [\text{T5(b)}]$$

2.1 Fundamentals of Boolean Algebra (4)

➤ *Theorem 6*

$$(a) \quad ab + ab' = a \quad (b) \quad (a + b)(a + b') = a$$

➤ *Examples:*

$$\blacksquare ABC + AB'C = AC \quad [T6(a)]$$

$$\blacksquare (W + X' + Y' + Z')(W' + X' + Y' + Z)(W' + X' + Y + Z')(W' + X' + Y + Z)$$

$$= (W' + X' + Y')(W' + X' + Y + Z')(W + X' + Y + Z) \quad [T6(b)]$$

$$= (W' + X' + Y')(W' + X' + Y) \quad [T6(b)]$$

$$= (W' + X') \quad [T6(b)]$$

2.1 Fundamentals of Boolean Algebra (5)

➤ Theorem 7

$$(a) \quad ab + ab'c = ab + ac$$

$$(b) \quad (a + b)(a + b' + c) = (a + b)(a + c)$$

➤ Examples:

$$\begin{aligned} \blacksquare \quad wy' + wx'y + wxyz + wxz' &= wy' + wx'y + wxy + wxz' && [T7(a)] \\ &= wy' + wy + wxz' && [T7(a)] \\ &= w + wxz' && [T7(a)] \\ &= w && [T7(a)] \\ \blacksquare \quad (x'y + z)(w + x'y + z) &= (x'y + z)(w + x'y) && [T7(b)] \end{aligned}$$

2.1 Fundamentals of Boolean Algebra (6)

➤ Theorem 8 (DeMorgan's Theorem)

$$(a) \quad (a + b)' = a' b' \qquad (b) \quad (ab)' = a' + b'$$

➤ Generalized DeMorgan's Theorem

$$(a) \quad (a + b + \dots + z)' = a' b' \dots z' \quad (b) \quad (ab \dots z)' = a' + b' + \dots + z'$$

➤ Examples:

$$\begin{aligned} \blacksquare (a + bc)' &= (a + (bc))' \\ &= a'(bc)' && [T8(a)] \\ &= a'(b' + c') && [T8(b)] \\ &= a'b' + a'c' && [P5(b)] \end{aligned}$$

$$\blacksquare \text{Note: } (a + bc)' \neq a'b' + c'$$

2.1 Fundamentals of Boolean Algebra (7)

➤ More Examples for DeMorgan's Theorem

- $$\begin{aligned} \blacksquare (a(b + z(x + a')))' &= a' + (b + z(x + a'))' && \text{[T8(b)]} \\ &= a' + b' (z(x + a'))' && \text{[T8(a)]} \\ &= a' + b' (z' + (x + a'))' && \text{[T8(b)]} \\ &= a' + b' (z' + x'(a'))' && \text{[T8(a)]} \\ &= a' + b' (z' + x'a) && \text{[T3]} \\ &= a' + b' (z' + x') && \text{[T5(a)]} \end{aligned}$$
- $$\begin{aligned} \blacksquare (a(b + c) + a'b)' &= (ab + ac + a'b)' && \text{[P5(b)]} \\ &= (b + ac)' && \text{[T6(a)]} \\ &= b'(ac)' && \text{[T8(a)]} \\ &= b'(a' + c') && \text{[T8(b)]} \end{aligned}$$

2.1 Fundamentals of Boolean Algebra (8)

➤ *Theorem 9 (Consensus)*

$$(a) \quad ab + a'c + bc = ab + a'c$$

$$(b) \quad (a + b)(a' + c)(b + c) = (a + b)(a' + c)$$

➤ *Examples:*

$$\blacksquare AB + A'CD + BCD = AB + A'CD \quad [T9(a)]$$

$$\blacksquare (a + b')(a' + c)(b' + c) = (a + b')(a' + c) \quad [T9(b)]$$

$$\blacksquare ABC + A'D + BD + CD = ABC + (A' + B)D + CD \quad [P5(b)]$$

$$= ABC + (AB)'D + CD \quad [T8(b)]$$

$$= ABC + (AB)'D \quad [T9(a)]$$

$$= ABC + (A' + B')D \quad [T8(b)]$$

$$= ABC + A'D + B'D \quad [P5(b)]$$

➤ *Theorem 10 : will be presented later*

2.2 Switching Functions (1)

- **Switching algebra:** Boolean algebra with the set of elements $K = \{0, 1\}$
- If there are n variables, we can define 2^{2^n} switching functions.
- Sixteen functions of two variables (Table 2.3):

AB	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	1	2	3	4	5
00	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
10	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

- A switching function can be represented by a table as above, or by a switching expression as follows:
- $f_0(A,B) = 0$, $f_6(A,B) = AB' + A'B$, $f_{11}(A,B) = AB + A'B + A'B' = A' + B$,
...
- Value of a function can be obtained by plugging in the values of all variables:

The value of f_6 when $A = 1$ and $B = 0$ is: $1 \cdot 0' + 0 \cdot 1' = 1 + 0 = 1$.

2.2 switching Functions (2)

2.2.1 Truth Table

- Shows the value of a function for all possible input combinations.
- Truth tables for OR, AND, and NOT (Table 2.4):

ab	$f(a,b)=a+b$	ab	$f(a,b)=ab$	a	$f(a)=a'$
00	0	00	0	0	1
01	1	01	0	1	0
10	1	10	0		
11	1	11	1		

2.2 switching Functions (3)

➤ Truth tables for $f(A,B,C) = AB + A'C + AC'$
(Table 2.5)

<i>ABC</i>	<i>f(A,B,C)</i>	<i>ABC</i>	<i>f(A,B,C)</i>
000	0	<i>FFF</i>	<i>F</i>
001	1	<i>FFT</i>	<i>T</i>
010	0	<i>FTF</i>	<i>F</i>
011	1	<i>FTT</i>	<i>T</i>
100	1	<i>TFF</i>	<i>T</i>
101	0	<i>TFT</i>	<i>F</i>
110	1	<i>TTF</i>	<i>T</i>
111	1	<i>TTT</i>	<i>T</i>

2.2.2 Algebraic Forms of Switching Functions (1)

- ***Literal***: A variable, complemented or uncomplemented.
- ***Product term***: A literal or literals ANDed together
- ***Sum term***: A literal or literals ORed together.
- ***SOP (Sum of Products)*** :
 - ORing product terms
 - $f(A, B, C) = ABC + A'C + B'C$
- ***POS (Product of Sums)***
 - ANDing sum terms
 - $f(A, B, C) = (A' + B' + C)(A + C')(B + C)$

2.2.2 Algebraic Forms of Switching Functions (2)

- A *minterm* is a product term in which all the variables appear exactly once either complemented or uncomplemented.
- *Canonical Sum of Products (canonical SOP)*:
 - Represented as a sum of minterms only.
 - *Example*: $f_1(A,B,C) = A'BC' + ABC' + A'BC + ABC$ (2.1)
- Minterms of three variables:

Minterm	Minterm Code	Minterm Number
$A'B'C'$	000	m_0
$A'B'C$	001	m_1
$A'BC'$	010	m_2
$A'BC$	011	m_3
$AB'C'$	100	m_4
$AB'C$	101	m_5
ABC'	110	m_6
ABC	111	m_7

2.2.2 Algebraic Forms of Switching Functions (3)

- Compact form of canonical SOP form:

$$f_7(A,B,C) = m_2 + m_3 + m_6 + m_7 \quad (2.2)$$

- A further simplified form:

$$f_7(A,B,C) = \Sigma m(2,3,6,7) \text{ (minterm list form)} \quad (2.3)$$

- The ***order of variables*** in the functional notation is important.

- Deriving truth table of $f_7(A,B,C)$ from minterm list:

Row No. (i)	Inputs ABC	Outputs $f_1(A,B,C) = \Sigma m(2,3,6,7)$	Complement $f_1'(A,B,C) = \Sigma m(0,1,4,5)$
0	000	0	1 ← m_0
1	001	0	1 ← m_1
2	010	1 ← m_2	0
3	011	1 ← m_3	0
4	100	0	1 ← m_4
5	101	0	1 ← m_5
6	110	1 ← m_6	0
7	111	1 ← m_7	0

2.2.2 Algebraic Forms of Switching Functions (4)

➤ *Example:*

➤ Given $f(A, B, Q, Z) = A'B'Q'Z' + A'B'Q'Z + A'BOQZ' + A'BOQZ$, express $f(A, B, Q, Z)$ and $f'(A, B, Q, Z)$ in minterm list form.

$$\begin{aligned} f(A, B, Q, Z) &= A'B'Q'Z' + AB'Q'Z + A'BOQZ' + A'BOQZ \\ &= m_0 + m_1 + m_6 + m_7 \\ &= \Sigma m(0, 1, 6, 7) \end{aligned}$$

$$\begin{aligned} f'(A, B, Q, Z) &= m_2 + m_3 + m_4 + m_5 + m_8 + m_9 + m_{10} + m_{11} + m_{12} \\ &\quad + m_{13} + m_{14} + m_{15} \\ &= \Sigma m(2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15) \end{aligned}$$

➤
$$\sum_{i=0}^{2^n-1} m_i = 1 \quad (2.6)$$

➤ $AB + (AB)' = 1$ and $AB + A' + B' = 1$, but $AB + A'B' \neq 1$.

2.2.2 Algebraic Forms of Switching Functions (5)

- A *maxterm* is a sum term in which all the variables appear exactly once either complemented or uncomplemented.
- **Canonical Product of Sums (canonical POS):**
 - Represented as a product of maxterms only.
 - *Example:* $f_2(A,B,C) = (A+B+C)(A+B+C)(A'+B+C)(A'+B+C)$ (2.7)
- Maxterms of three variables:

Maxterm	Maxterm Code	Maxterm Number
$A+B+C$	000	M_0
$A+B+C'$	001	M_1
$A+B'+C$	010	M_2
$A+B'+C'$	011	M_3
$A'+B+C$	100	M_4
$A'+B+C'$	101	M_5
$A'+B'+C$	110	M_6
$A'+B'+C'$	111	M_7

2.2.2 Algebraic Forms of Switching Functions (6)

$$\blacktriangleright f_2(A, B, C) = M_0 M_1 M_4 M_5 \quad (2.8)$$

$$= \Pi M(0, 1, 4, 5) \text{ (maxterm list form)} \quad (2.9)$$

The truth table for $f_2(A, B, C)$:

Row No. (i)	Inputs ABC	M_0 $A+B+C$	M_1 $A+B+C'$	M_4 $A'+B+C$	M_5 $A'+B+C'$	Outputs $f_2(A, B, C)$
0	000	0	1	1	1	0
1	001	1	0	1	1	0
2	010	1	1	1	1	1
3	011	1	1	1	1	1
4	100	1	1	0	1	0
5	101	1	1	1	0	0
6	110	1	1	1	1	1
7	111	1	1	1	1	1

2.2.2 Algebraic Forms of Switching Functions (7)

➤ Truth tables of $f_1(A,B,C)$ of Eq. (2.3) and $f_2(A,B,C)$ of Eq. (2.7) are identical.

➤ Hence, $f_1(A,B,C) = \Sigma m(2,3,6,7) = f_2(A,B,C)$
 $= \Pi M(0,1,4,5)$ (2.10)

➤ **Example:** Given $f(A,B,C) = (A+B+C)(A+B+C)(A'+B+C)(A'+B+C)$, construct the truth table and express in both maxterm and minterm form.

▪ $f(A,B,C) = M_1M_3M_5M_7 = \Pi M(1,3,5,7) = \Sigma m(0,2,4,6)$

Row No. (i)	Inputs ABC	Outputs $f(A,B,C) = \Pi M(1,3,5,7) = \Sigma m(0,2,4,6)$
0	000	1 m_0
1	001	0 ← M_1
2	010	1 m_2
3	011	0 ← M_3
4	100	1 m_4
5	101	0 ← M_5
6	110	1 m_6
7	111	0 ← M_7

2.2.2 Algebraic Forms of Switching Functions (8)

➤ Relationship between minterm m_i and maxterm M_i :

- For $f(A,B,C)$, $(m_1)' = (A'B'C)' = A + B + C' = M_1$

- In general, $(m_i)' = M_i$ (2.11)

$$(M_i)' = ((m_i)')' = m_i \quad (2.12)$$

2.2.2 Algebraic Forms of Switching Functions (9)

➤ **Example:** Relationship between the maxterms for a function and its complement.

- For $f(A,B,C) = (A+B+C)(A+B+C)(A'+B+C)(A'+B+C)$
- The truth table is:

Row No. (i)	Inputs ABC	Outputs $f(A,B,C)$	Outputs $f'(A,B,C) = \Pi M(0,2,4,6)$
0	000	1	0 ← M_0
1	001	0	1
2	010	1	0 ← M_2
3	011	0	1
4	100	1	0 ← M_4
5	101	0	1
6	110	1	0 ← M_6
7	111	0	1

2.2.2 Algebraic Forms of Switching Functions (10)

- From the truth table

$$f'(A,B,C) = \prod M(0,2,4,6) \text{ and } f(A,B,C) = \prod M(1,3,5,7)$$

- Since $f(A,B,C) \times f'(A,B,C) = 0$,

$$(M_0M_2M_4M_6)(M_1M_3M_5M_7) = 0$$

- In general,
$$\prod_{i=0}^{2^n - 1} M_i = 0 \quad (2.13)$$

- Another observation from the truth table:

$$f(A,B,C) = \sum m(0,2,4,6) = \prod M(1,3,5,7)$$

$$f'(A,B,C) = \sum m(1,3,5,7) = \prod M(0,2,4,6)$$

2.2.3 Derivation of Canonical Forms (1)

➤ Derive canonical POS or SOP using switching algebra.

➤ **Theorem 10. Shannon's expansion theorem**

$$(a). f(x_1, x_2, \dots, x_n) = x_1 f(1, x_2, \dots, x_n) + (x_1)' f(0, x_2, \dots, x_n)$$

$$(b). f(x_1, x_2, \dots, x_n) = [x_1 + f(0, x_2, \dots, x_n)] [(x_1)' + f(1, x_2, \dots, x_n)]$$

➤ **Example:** $f(A, B, C) = AB + AC + A'C$

$$\blacksquare f(A, B, C) = AB + AC + A'C = A f(1, B, C) + A' f(0, B, C)$$

$$= A(1 \times B + 1 \times C + 1' \times C) + A'(0 \times B + 0 \times C + 0' \times C) = A(B + C) + A'C$$

$$\blacksquare f(A, B, C) = A(B + C) + A'C = B[A(1 + C) + A'C] + B[A(0 + C) + A'C]$$

$$= B[A + A'C] + B[AC + A'C] = AB + A'BC + ABC + A'BC$$

$$\blacksquare f(A, B, C) = AB + A'BC + ABC + A'BC$$

$$= C[AB + A'B \times 1 + AB \times 1' + A'B \times 1] + C[AB + A'B \times 0 + AB \times 0' + A'B \times 0]$$

$$= ABC + A'BC + A'BC + ABC + ABC$$

2.2.3 Derivation of Canonical Forms (2)

➤ **Alternative:** Use Theorem 6 to add missing literals.

➤ **Example:** $f(A,B,C) = AB + AC + A'C$ to canonical SOP form.

- $AB = ABC + ABC = m_6 + m_7$

- $AC = ABC + ABC = m_4 + m_6$

- $A'C = A'BC + A'BC = m_1 + m_3$

- Therefore,

$$f(A,B,C) = (m_6 + m_7) + (m_4 + m_6) + (m_1 + m_3) = \Sigma m(1, 3, 4, 6, 7)$$

➤ **Example:** $f(A,B,C) = A(A + C)$ to canonical POS form.

- $A = (A+B')(A+B) = (A+B'+C)(A+B'+C)(A+B+C)(A+B+C) = M_3M_2M_1M_0$

- $(A+C) = (A+B'+C)(A+B+C) = M_3M_1$

- Therefore,

$$f(A,B,C) = (M_3M_2M_1M_0)(M_3M_1) = \Pi M(0, 1, 2, 3)$$

2.2.4 Incompletely Specified Functions

- A switching function may be incompletely specified.
- Some minterms are omitted, which are called *don't-care minterms*.
- Don't cares arise in two ways:
 - Certain input combinations never occur.
 - Output is required to be 1 or 0 only for certain combinations.
- Don't care minterms: d_i Don't care maxterms: D_i
- **Example:** $f(A,B,C)$ has minterms $m_0, m_3,$ and m_7 and don't-cares d_4 and d_5 .
 - Minterm list is: $f(A,B,C) = \Sigma m(0,3,7) + d(4,5)$
 - Maxterm list is: $f(A,B,C) = \Pi M(1,2,6) \cdot D(4,5)$
 - $f'(A,B,C) = \Sigma m(1,2,6) + d(4,5) = \Pi M(0,3,7) \cdot D(4,5)$
 - $f(A,B,C) = A'BC + A'BC + ABC + d(ABC + AB'C)$
 $= BC + BC$ (use d_4 and omit d_5)

2.3 switching Circuits (1)

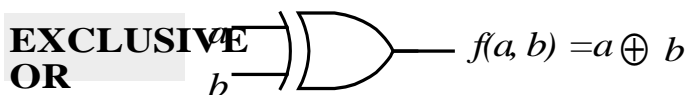
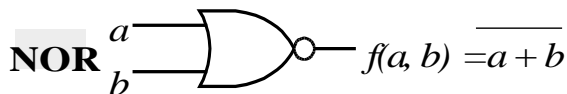
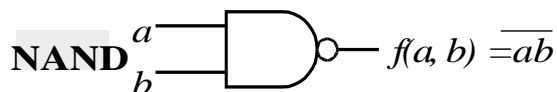
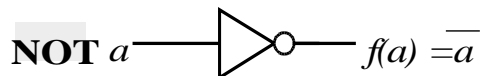
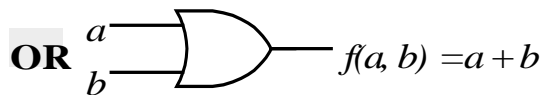
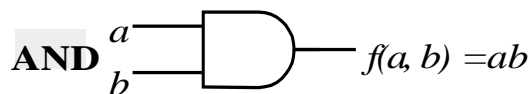
2.3.1 Electronic Logic Gates

➤ *Electrical Signals and Logic Values*

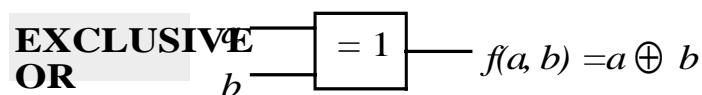
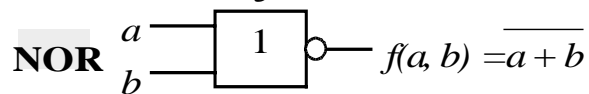
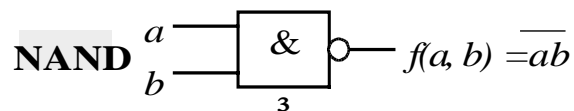
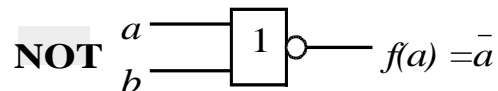
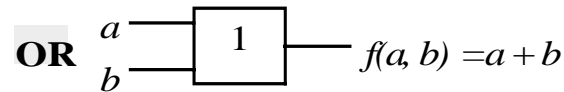
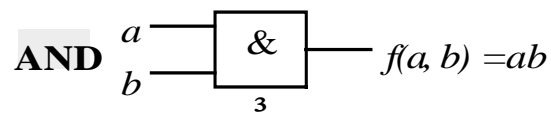
Electric Signal	Logic Value	
	Positive Logic	Negative Logic
High Voltage (H)	1	0
Low Voltage (L)	0	1

- A signal that is set to logic 1 is said to be *asserted*, *active*, or *true*.
- An *active-high* signal is asserted when it is high (positive logic).
- An *active-low* signal is asserted when it is low (negative logic).

2.3.1 Electronic Logic Gates (1)



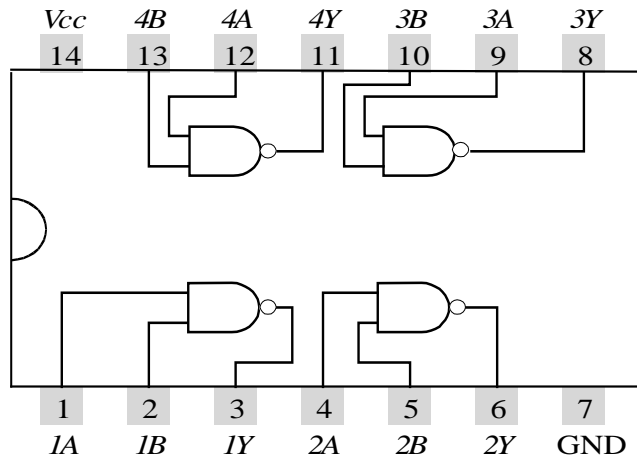
Symbol set 1



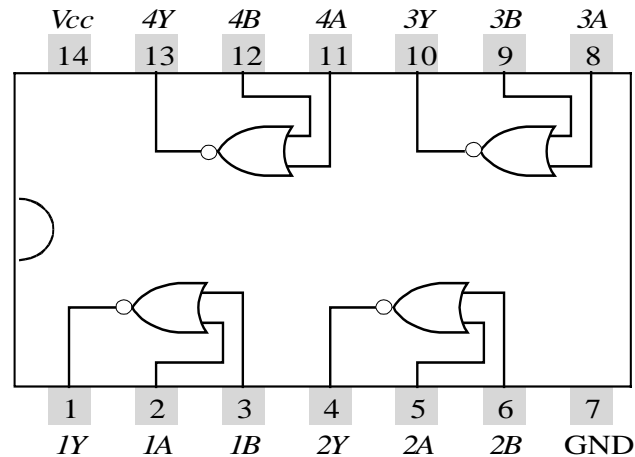
Symbol set 2

(ANSI/IEEE Standard 91-1984)

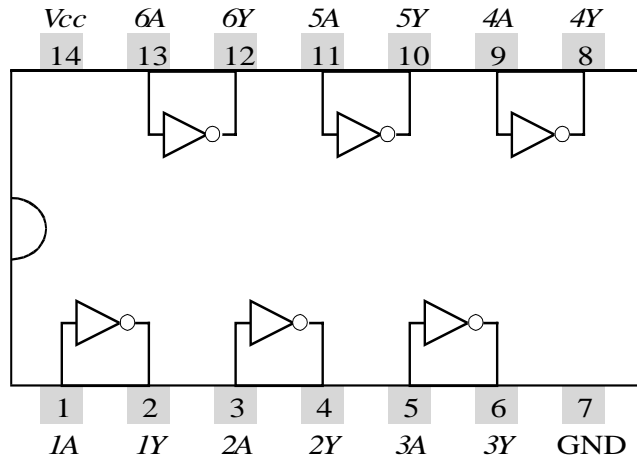
2.3.1 Electronic Logic Gates (2)



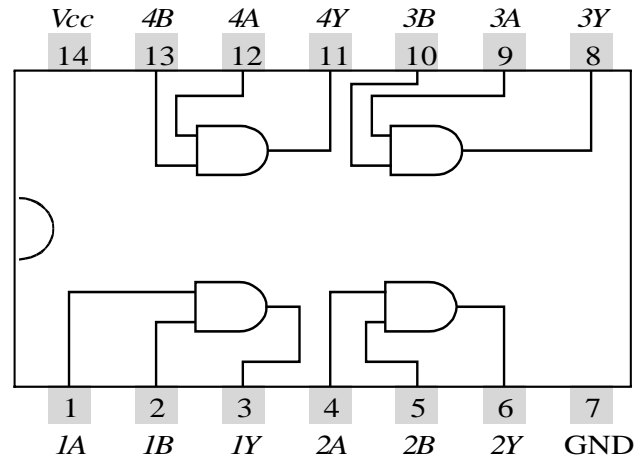
$7400Y = \overline{AB}$
Quadruple two-input NAND gates



$7402Y = \overline{A+B}$
Quadruple two-input NOR gates

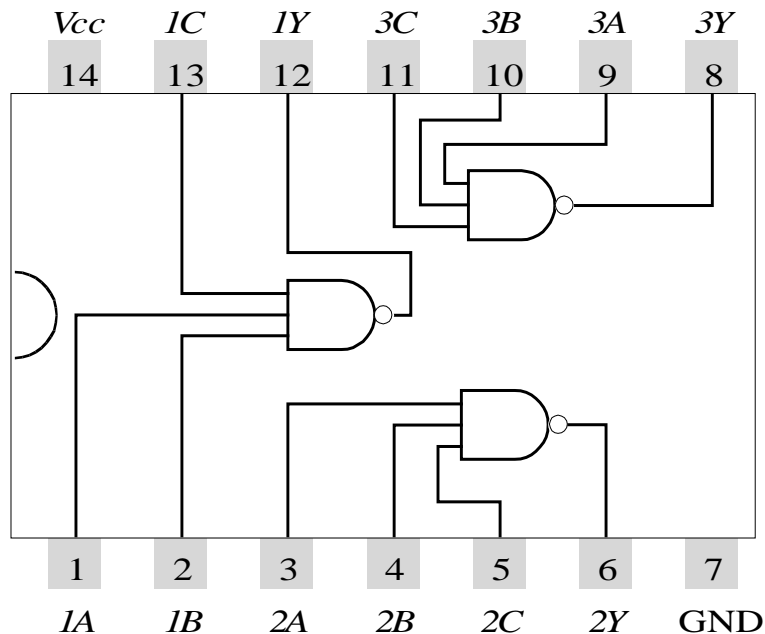


$7404Y = \overline{A}$
Hex inverters

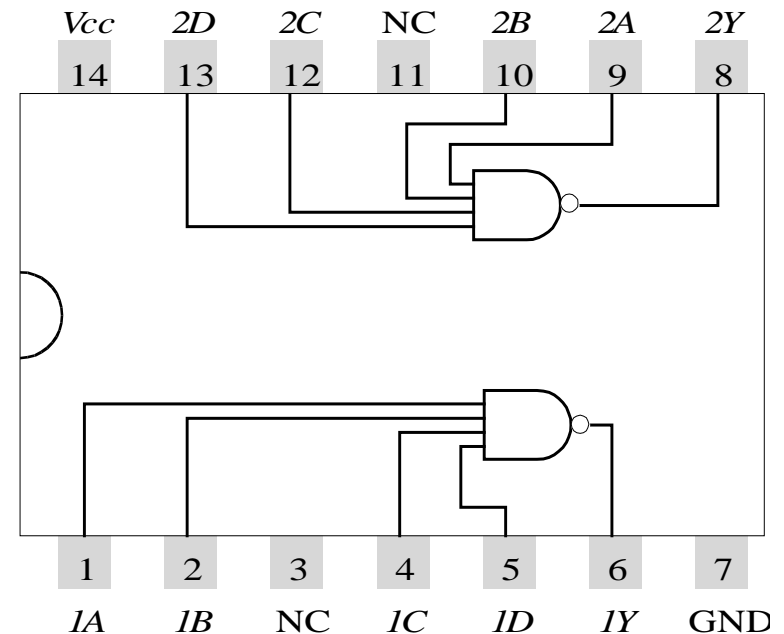


$7408Y = AB$
Quadruple two-input AND gates

2.3.1 Electronic Logic Gates (3)

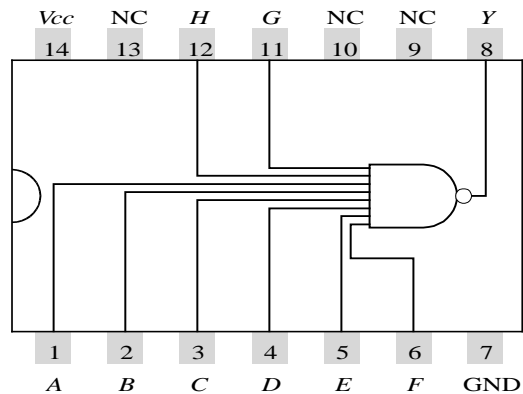


$7410Y = ABC$
Triple three-input NAND gates

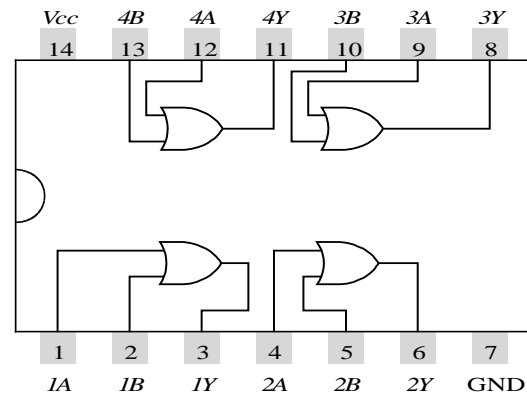


$7420Y = ABCD$
Dual four-input NAND gates

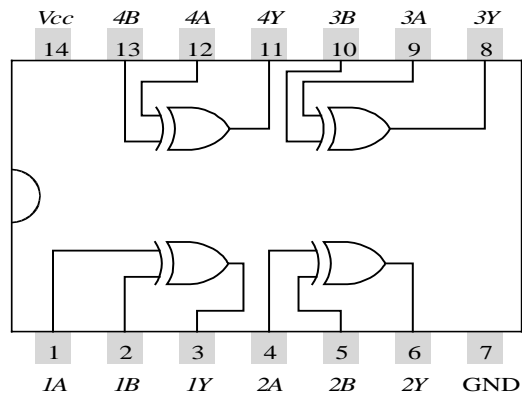
2.3.1 Electronic Logic Gates (4)



7430Y = ABCDEFGH
8-input NAND gate



7432Y = A + B
Quadruple two-input OR gates



7486Y = A ⊕ B
Quadruple two-input exclusive-OR gates

2.3.2 Basic Functional Components (1)

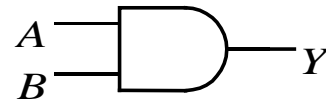
➤ AND

a	b	$f_{AND}(a, b) = ab$
0	0	0
0	1	0
1	0	0
1	1	1

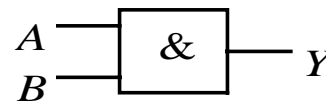
(a)

A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

(b)



(c)



(d)

- (a) AND logic function.
- (b) Electronic AND gate.
- (c) Standard symbol.
- (d) IEEE block symbol.

2.3.2 Basic Functional Components (2)

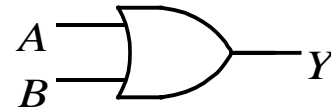
➤ OR

a	b	$f_{OR}(a, b) = a + b$
0	0	0
0	1	1
1	0	1
1	1	1

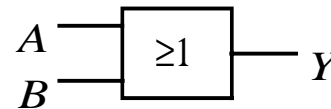
(a)

A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H

(b)



(c)



(d)

- (a) OR logic function.
- (b) Electronic OR gate.
- (c) Standard symbol.
- (d) IEEE block symbol.

2.3.2 Basic Functional Components (3)

- Meaning of the designation ≥ 1 in IEEE symbol:

ab	$\text{sum}(a, b)$	$\text{sum}(a, b) \geq 1$	$f_{OR}(a, b) = a + b$
00	0	False	0
01	1	True	1
10	1	True	1
11	2	True	1

2.3.2 Basic Functional Components (4)

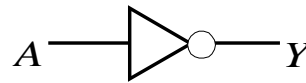
➤ NOT

a	$fNOT(a) = \bar{a}$
0	1
1	0

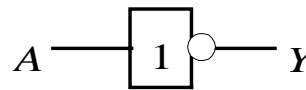
(a)

A	Y
L	H
H	L

(b)



(c)



(d)

- (a) NOT logic function.
- (b) Electronic NOT gate.
- (c) Standard symbol.
- (d) IEEE block symbol.

2.3.2 Basic Functional Components (5)

➤ *Positive Versus Negative Logic*

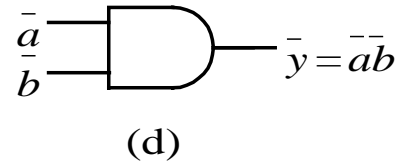
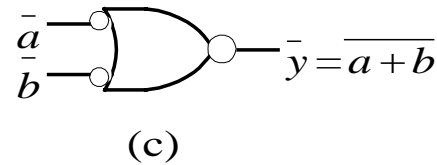
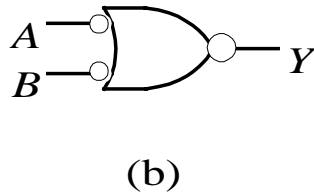
	Positive Logic	Negative Logic
1 is represented by	High Voltage	Low Voltage
0 is represented by	Low Voltage	High Voltage

2.3.2 Basic Functional Components (6)

➤ *AND Gate Usage in Negative Logic*

A	B	Y
1	1	1
1	0	1
0	1	1
0	0	0

(a)



- (a) AND gate truth table ($L = 1, H = 0$)
- (b) Alternate AND gate symbol (in negative logic)
- (c) Preferred usage
- (d) Improper usage

$$y = a \cdot b = \overline{\overline{a \cdot b}} = \overline{\overline{a} + \overline{b}} = \overline{f_{OR}(\overline{a}, \overline{b})} \quad (2.14)$$

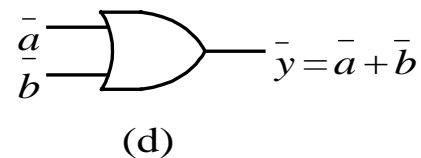
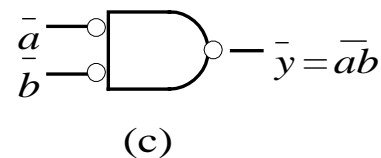
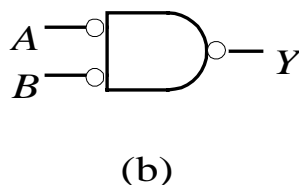
$$\overline{y} = \overline{\overline{\overline{a} + \overline{b}}} = \overline{\overline{a} + \overline{b}} = \overline{f_{OR}(a, b)} \quad (2.15)$$

2.3.2 Basic Functional Components (7)

➤ OR Gate Usage in Negative Logic

A	B	Y
1	1	1
1	0	0
0	1	0
0	0	0

(a)



- (a) OR gate truth table ($L = 1, H = 0$)
- (b) Alternate OR gate symbol (in negative logic)
- (c) Preferred usage
- (d) Improper usage
- $y = a + b = \overline{\overline{a + b}} = \overline{\bar{a} \cdot \bar{b}} = \bar{f}_{AND}(\bar{a}, \bar{b})$ (2.16)
- $\bar{y} = \overline{(\bar{a}) \cdot (\bar{b})} = \overline{\bar{a} \cdot \bar{b}} = \bar{f}_{AND}(a, b)$ (2.17)

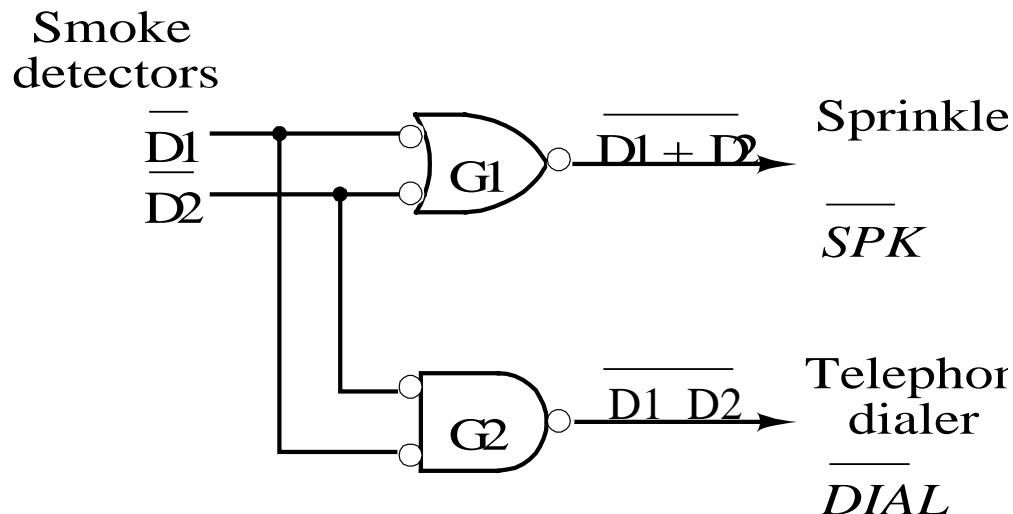
2.3.3 Basic Functional Components (8)

➤ *Example 2.32* : Building smoke alarm system

- Components: two smoke detectors, a sprinkler, and an automatic telephone dialer
- Behavior:
 - Sprinkler is activated if either smoke detector detects smoke.
 - When both smoke detector detect smoke, fire department is called.
- Signals:
 - $\overline{D1}, \overline{D2}$: Active-low outputs from two smoke detectors.
 - \overline{SPK} : Active-low input to the sprinkler
 - \overline{DIAL} : Active-low input to the telephone dialer.
- Logic equations
 - $$\overline{SPK} = \overline{D1 + D2} \quad (2.18)$$
 - $$\overline{DIAL} = \overline{D1 \cdot D2} \quad (2.19)$$

2.3.2 Basic Functional Components (9)

- Logic diagram of the smoke alarm system



2.3.2 Basic Functional Components (10)

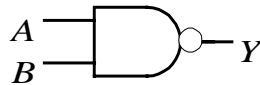
➤ *NAND*

a	b	$f_{NAND}(a, b) = \overline{ab}$
0	0	1
0	1	1
1	0	1
1	1	0

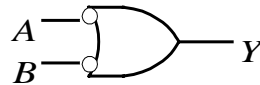
(a)

A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

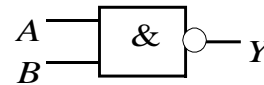
(b)



(c)



(d)



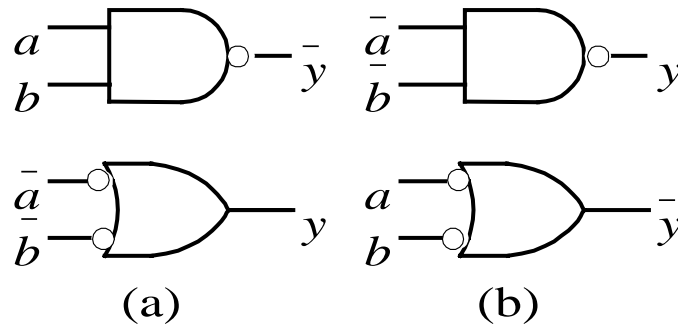
(e)

- (a) NAND logic function
- (b) Electronic NAND gate
- (c) Standard symbol
- (d) IEEE block symbol

2.3.2 Basic Functional Components (10)

➤ Matching signal polarity to NAND gate inputs/outputs

- (a) Preferred usage (b) Improper usage



➤ Additional properties of NAND gate:

$$f_{NAND}(a, a) = \overline{a \cdot a} = \bar{a} = f_{NOT}(a)$$

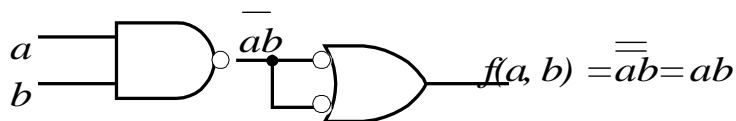
$$\bar{f}_{NAND}(a, b) = \overline{\overline{a \cdot b}} = a \cdot b = f_{AND}(a, b)$$

$$f_{NAND}(\bar{a}, \bar{b}) = \overline{\bar{a} \cdot \bar{b}} = a + b = f_{OR}(a, b)$$

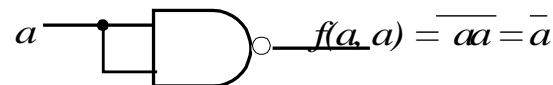
➤ Hence, NAND gate may be used to implement all three elementary operators.

2.3.2 Basic Functional Components (11)

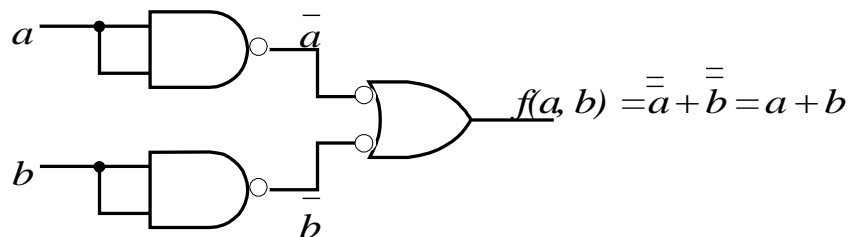
- AND, OR, and NOT gates constructed exclusively from NAND gates



AND gate



NOT gate



OR gate

2.3.2 Basic Functional Components (12)

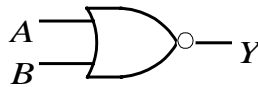
➤ *NOR*

a	b	$f_{NOR}(a, b) = \overline{a + b}$
0	0	1
0	1	0
1	0	0
1	1	0

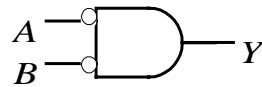
(a)

A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L

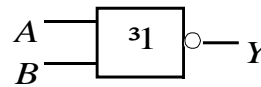
(b)



(c)



(d)



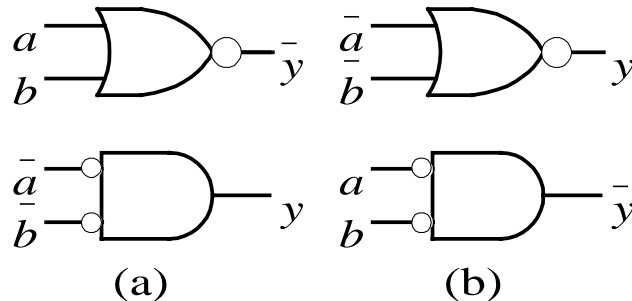
(e)

- (a) NAND logic function
- (b) Electronic NAND gate
- (c) Standard symbol
- (d) IEEE block symbol

2.3.2 Basic Functional Components (13)

➤ Matching signal polarity to NOR gate inputs/outputs

- (a) Preferred usage (b) Improper usage



➤ Additional properties of NAND gate:

$$f_{NOR}(a, a) = \overline{a + a} = \bar{a} = f_{NOT}(a)$$

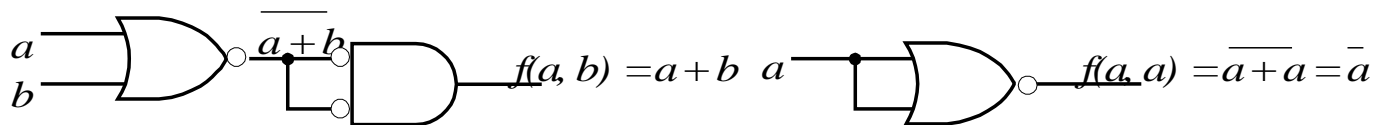
$$\bar{f}_{NOR}(a, b) = \overline{\overline{a + b}} = a + b = f_{OR}(a, b)$$

$$f_{NOR}(\bar{a}, \bar{b}) = \overline{\bar{a} + \bar{b}} = a \cdot b = f_{AND}(a, b)$$

➤ Hence, NAND gate may be used to implement all three elementary operators.

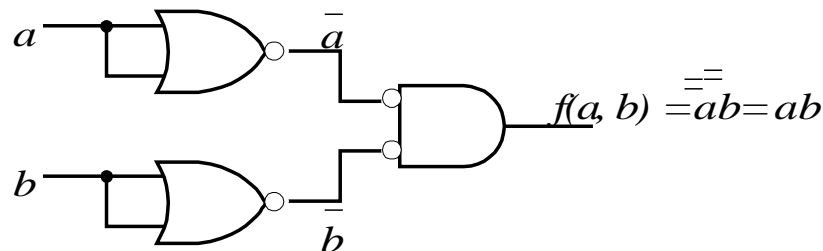
2.3.2 Basic Functional Components (14)

- AND, OR, and NOT gates constructed exclusively from NOR gates.



OR gate

NOT gate



AND gate

2.3.2 Basic Functional Components (15)

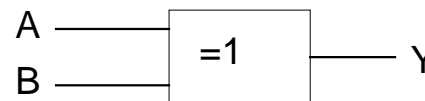
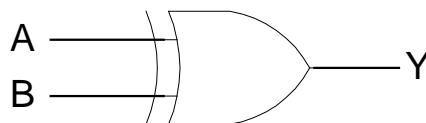
➤ Exclusive-OR (XOR)

$$f_{\text{XOR}}(a, b) = a \oplus b = \bar{a}b + a\bar{b} \quad (2.24)$$

$a b$	$f_{\text{XOR}}(a, b) = a \oplus b$	$A B$	Y
0 0	0	L L	L
0 1	1	L H	H
1 0	1	H L	H
1 1	0	H H	L

(a) XOR logic function

(b) Electronic XOR gate



(c) Standard symbol

(d) IEEE block symbol

2.3.2 Basic Functional Components (16)

➤ POS of XOR

$$\begin{aligned}
 a \oplus b &= \bar{a}b + a\bar{b} \\
 &= \bar{a}a + \bar{a}b + a\bar{b} + b\bar{b} && [P2(a), P6(b)] \\
 &= \bar{a}(a+b) + \bar{b}(a+b) && [P5(b)] \\
 &= (\bar{a} + \bar{b})(a+b) && [P5(b)]
 \end{aligned}$$

➤ Some other useful relationships

- $a \oplus a = 0$ (2.25)
- $a \oplus \bar{a} = 1$ (2.26)
- $a \oplus 0 = a$ (2.27)
- $a \oplus 1 = \bar{a}$ (2.28)
- $\bar{a} \oplus \bar{b} = a \oplus b$ (2.29)
- $a \oplus b = b \oplus a$ (2.30)
- $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ (2.31)

2.3.2 Basic Functional Components (17)

- Output of XOR gate is asserted when the mathematical sum of inputs is *one*:

ab	$sum(a, b)$	$sum(a, b) = 1?$	$f(a, b) = a \oplus b$
00	0	False	0
01	1	True	1
10	1	True	1
11	2	False	0

- The output of XOR is the *modulo-2* sum of its inputs.

2.3.2 Basic Functional Components (18)

➤ Exclusive-NOR (XNOR)

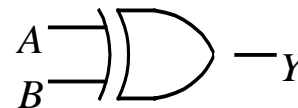
$$\blacksquare f_{\text{XNOR}}(a, b) = \overline{a \oplus b} = a \odot b \quad (2.32)$$

a	b	$f_{\text{XNOR}}(a, b) = a \odot b$
0	0	1
0	1	0
1	0	0
1	1	1

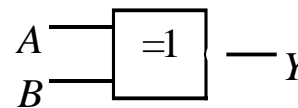
(a)

A	B	Y
L	L	H
L	H	L
H	L	L
H	H	H

(b)



(c)



(d)

- (a) XNOR logic function
- (b) Electronic XNOR gate
- (c) Standard symbol
- (d) IEEE block symbol

2.3.2 Basic Functional Components (19)

➤ SOP and POS of XNOR

$$\begin{aligned}
 a \odot b &= \overline{a \oplus b} \\
 &= \overline{\overline{ab} + a\overline{b}} && \text{[P2]} \\
 &= \overline{\overline{ab}} \cdot \overline{a\overline{b}} && \text{[T8(a)]} \\
 &= (a + \overline{b})(\overline{a} + b) && \text{[T8(b)]} \\
 &= a\overline{a} + ab + \overline{a}\overline{b} + \overline{b}b && \text{[P5(b)]} \\
 &= ab + \overline{a}\overline{b} && \text{[P6(b), P2(a)]}
 \end{aligned}$$

$$\text{➤ } a \oplus \overline{b} = a \odot b$$

2.4 Analysis of Combinational Circuits (1)

➤ Digital Circuit *Design*:

- Word description of a function
 - ⇒ a set of switching equations
 - ⇒ hardware realization (gates, programmable logic devices, etc.)

➤ Digital Circuit *Analysis*:

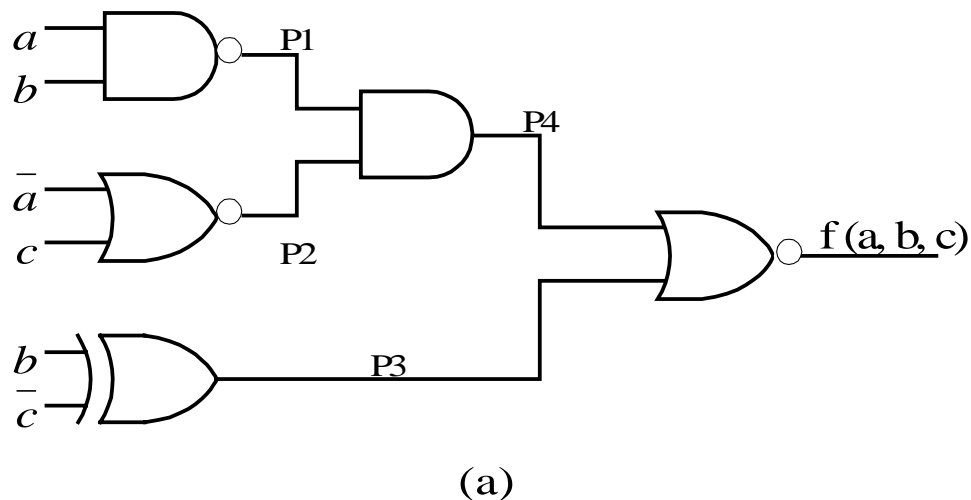
- Hardware realization
 - ⇒ switching expressions, truth tables, timing diagrams, etc.

➤ Analysis is used

- To determine the behavior of the circuit
- To verify the correctness of the circuit
- To assist in converting the circuit to a different form.

2.4 Analysis of Combinational Circuits (2)

- **Algebraic Method:** Use switching algebra to derive a desired form.
- **Example 2.33:** Find a simplified switching expressions and logic network for the following logic circuit (Fig. 2.21a).



2.4 Analysis of Combinational Circuits (3)

- Write switching expression for each gate output:

$$P_1 = \overline{ab}, \quad P_2 = \overline{a+c}, \quad P_3 = b \oplus \overline{c}, \quad P_4 = P_1 \cdot P_2 = \overline{ab} \cdot \overline{(a+c)}$$

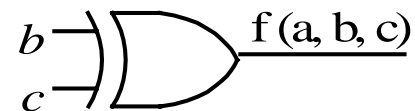
- The output is: $f(a,b,c) = \overline{P_3 + P_4} = \overline{(b \oplus \overline{c}) + \overline{ab} \cdot \overline{(a+c)}}$

- Simplify the output function using switching algebra:

$$\begin{aligned} \overline{f(a,b,c)} &= \overline{(b \oplus \overline{c}) + \overline{ab} \cdot \overline{a+c}} \\ &= bc + \overline{b}\overline{c} + \overline{ab} \cdot \overline{a+c} \quad [\text{Eq. 2.24}] \\ &= bc + \overline{b}\overline{c} + (\overline{a} + \overline{b})a\overline{c} \quad [\text{T8}] \\ &= bc + \overline{b}\overline{c} + a\overline{b}\overline{c} \quad [\text{T5(b)}] \\ &= bc + \overline{b}\overline{c} \quad [\text{T4(a)}] \end{aligned}$$

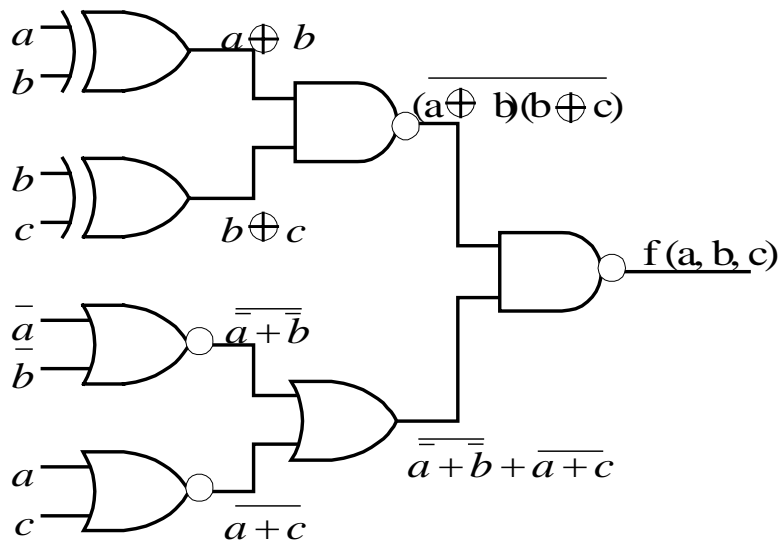
$$\overline{f(a,b,c)} = b \odot c \quad [\text{Eq. 2.32}]$$

$$\text{Therefore, } f(a,b,c) = (b \odot c)' = b \oplus c$$



2.4 Analysis of Combinational Circuits (4)

➤ **Example 2.34:** Find a simplified switching expressions and logic network for the following logic circuit (Fig. 2.22).

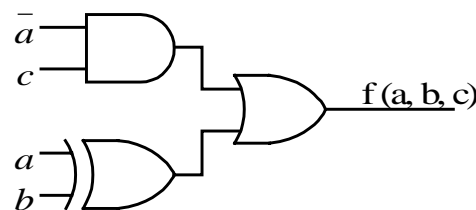


Given circuit

2.4 Analysis of Combinational Circuits (5)

➤ Derive the output expression:

$$\begin{aligned}
 f(a,b,c) &= \overline{(a \oplus b)(b \oplus c) \cdot (\bar{a} + \bar{b} + a + c)} \\
 &= \overline{(a \oplus b)(b \oplus c) + \bar{a} + \bar{b} + a + c} && \text{[T8(b)]} \\
 &= (a \oplus b)(b \oplus c) + (\bar{a} + \bar{b})(a + c) && \text{[T8(a)]} \\
 &= (a\bar{b} + \bar{a}b)(b\bar{c} + \bar{b}c) + (\bar{a} + \bar{b})(a + c) && \text{[Eq. 2.24]} \\
 &= a\bar{b}\bar{c} + a\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + \bar{a}a + \bar{a}c + a\bar{b} + \bar{b}c && \text{[P5(b)]} \\
 &= a\bar{b}\bar{c} + a\bar{b}c + \bar{a}c + a\bar{b} + \bar{b}c && \text{[P6(b), T4(a)]} \\
 &= a\bar{b}\bar{c} + \bar{a}c + a\bar{b} + \bar{b}c && \text{[T4(a)]} \\
 &= a\bar{b}\bar{c} + \bar{a}c + a\bar{b} && \text{[T9(a)]} \\
 &= \bar{a}b + \bar{a}c + a\bar{b} && \text{[T7(a)]} \\
 &= \bar{a}c + a \oplus b && \text{[Eq. 2.24]}
 \end{aligned}$$



Simplified circuit

2.4 Analysis of Combinational Circuits (6)

- **Truth Table Method:** Derive the truth table one gate at a time.
- The truth table for Example 2.34:

abc	$\bar{a}c$	$a \oplus b$	$f(a,b,c)$
000	0	0	0
001	1	0	1
010	0	1	1
011	1	1	1
100	0	1	1
101	0	1	1
110	0	0	0
111	0	0	0

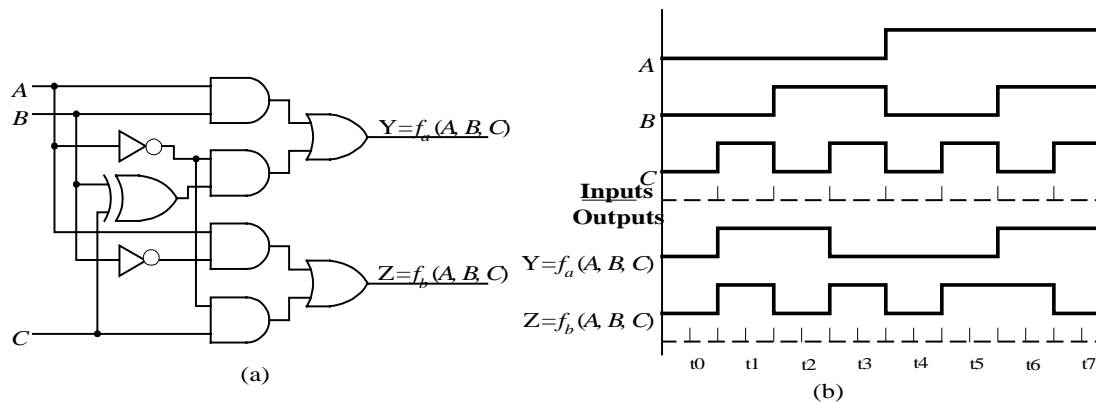
2.4 Analysis of Combinational Circuits (7)

➤ *Analysis of Timing Diagrams*

- *Timing diagram* is a graphical representation of input and output signal relationships over the time dimension.
- Timing diagrams may show intermediate signals and propagation delays.

2.4 Analysis of Combinational Circuits (8)

➤ **Example 2.35:** Derivation of truth table from a timing diagram



Time	Inputs	Outputs	
	ABC	$f_a(A, B, C)$	$f_b(A, B, C)$
t0	0 0 0	0	0
t1	0 0 1	1	1
t2	0 1 0	1	0
t3	0 1 1	0	1
t4	1 0 0	0	0
t5	1 0 1	0	1
t6	1 1 0	1	1
t7	1 1 1	1	0

(c)

2.4 Analysis of Combinational Circuits (9)

➤ Propagation Delay

- Physical characteristics of a logic circuit to be considered:
 - Propagation delays
 - Gate fan-in and fan-out restrictions
 - Power consumption
 - Size and weight

- *Propagation delay*: The delay between the time of an input change and the corresponding output change.

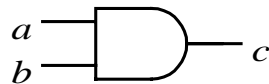
- Typical two propagation delay parameters:
 - t_{PLH} = propagation delay time, low-to-high-level output
 - t_{PHL} = propagation delay time, high-to-low-level output

- Approximation:

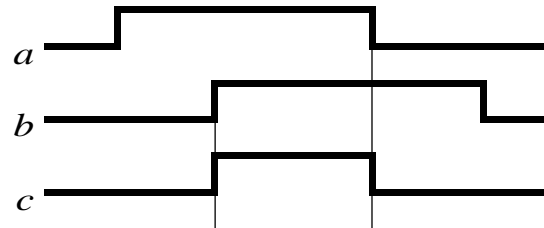
- $$t_{PD} = \frac{t_{PLH} + t_{PHL}}{2}$$

2.4 Analysis of Combinational Circuits(10)

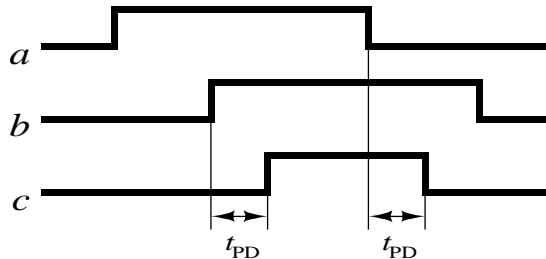
➤ Propagation delay through a logic gate



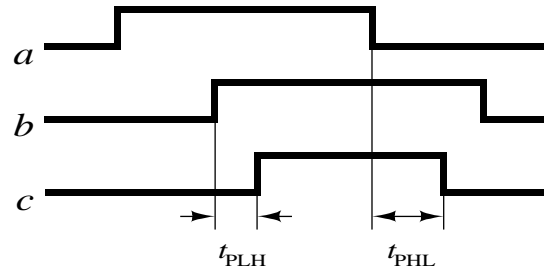
(a) Two-input AND gate



(b) Ideal (zero) delay



(c) $t_{PD} = t_{PLH} = t_{PHL}$



(d) $t_{PLH} < t_{PHL}$

2.4 Analysis of Combinational Circuits(11)

- Power dissipation and propagation delays for several logic families (Table 2.7)

Logic Family	Propagation Delay $t_{PD}(ns)$	Power Dissipation Per Gate (mW)	Technology
7400	10	10	Standard TTL
74H00	6	22	High-speed TTL
74L00	33	1	Low-power TTL
74LS00	9.5	2	Low-power Schottky TTL
74S00	3	19	Schottky TTL
74ALS00	3.5	1.3	Advanced low-power Schottky TTL
74AS00	3	8	Advanced Schottky TTL
74HC00	8	0.17	High-speed CMOS

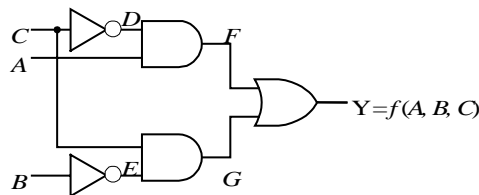
2.4 Analysis of Combinational Circuits(12)

- Propagation delays of primitive 74LS series gates (Table 2.8)

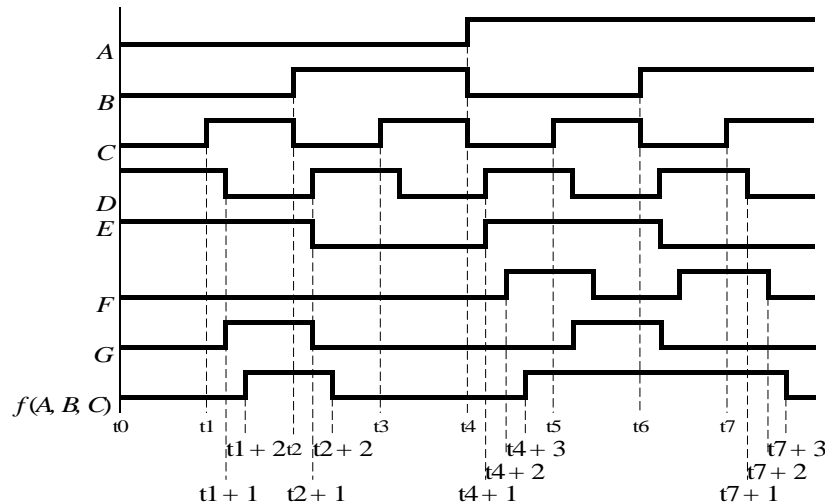
Chip	Function	t_{PLH}		t_{PHL}	
		Typical	Maximum	Typical	Maximum
74LS04	NOT	9	15	10	15
74LS00	NAND	9	15	10	15
74LS02	NOR	10	15	10	15
74LS08	AND	8	15	10	20
74LS32	OR	14	22	14	22

2.4 Analysis of Combinational Circuits(13)

➤ **Example 2.36:** Given a circuit diagram and the timing diagram, find the truth table and minimum switching expression.



ABC	f(A, B, C)
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	0



$$f(A, B, C)$$

$$= \sum m(1, 4, 5, 6)$$

$$= \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + ABC$$

$$= A\bar{C} + \bar{B}C$$

Synthesis of Combinational Logic Circuits (1)-Mano

- We Implement a switching function in two level
- Implementation with four type gates: AND, OR, NAND, NOR
- Total possible forms are $4 \times 4 = 16$ states

- | | |
|-------------|------------|
| ▪ AND-AND | ▪ OR-AND |
| ▪ AND-OR | ▪ OR-OR |
| ▪ AND_NAND | ▪ OR_NAND |
| ▪ AND-NOR | ▪ OR-NOR |
| ▪ NAND-AND | ▪ NOR-AND |
| ▪ NAND-OR | ▪ NOR-OR |
| ▪ NAND_NAND | ▪ NOR_NAND |
| ▪ NAND-NOR | ▪ NOR-NOR |

Synthesis of Combinational Logic Circuits (2)-Mano

➤ Non Degenerate forms: (Desired)

1. (a) AND-OR (b) NAND- NAND

2. (a) OR-AND (b) NOR- NOR

3. AND-OR-Invert

(a) AND- NOR (b) NAND-AND

4. OR-AND-Invert

(a) OR_NAND (b) NOR-OR

Synthesis of Combinational Logic Circuits (3)-Mano

➤ Example : Implement $f(A,B,C) = \sum m(1,2,4)$

In eight non degenerate forms

A	B	C	f	\bar{f}
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	0	1	0	1

Synthesis of Combinational Logic Circuits (4)-Mano

1. f in SOP form

AND-OR $f(A, B, C) = m_1 + m_2 + m_4 = \overline{A}BC + \overline{A}B\overline{C} + \overline{A}BC$

NAND-NAND $f(A, B, C) = \overline{\overline{\overline{A}BC} + \overline{\overline{A}B\overline{C}} + \overline{\overline{A}BC}}$
 $= \overline{\overline{A}BC} \cdot \overline{\overline{A}B\overline{C}} \cdot \overline{\overline{A}BC}$

2. f in POS form

OR-AND $f(A, B, C) = M_0 \cdot M_3 \cdot M_5 \cdot M_6 \cdot M_7$
 $= (A + B + C)(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + \overline{C})$

NOR-NOR

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

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

Synthesis of Combinational Logic Circuits (5)-Mano

3. \bar{f} in SOP form

AND-NOR

$$\begin{aligned} \overline{f(A,B,C)} &= m_0 + m_3 + m_5 + m_6 + m_7 \\ &= \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} \end{aligned}$$

$$\overline{f(A,B,C)} = \overline{\overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC}}$$

NAND-AND

$$\overline{f(A,B,C)} = \overline{\overline{ABC} \cdot \overline{ABC} \cdot \overline{ABC} \cdot \overline{ABC} \cdot \overline{ABC}}$$

4. \bar{f} in POS form

OR-NAND

$$\begin{aligned} \overline{f(A,B,C)} &= M_1 \cdot M_2 \cdot M_4 \\ &= (A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + B + C) \end{aligned}$$

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

NOR-OR

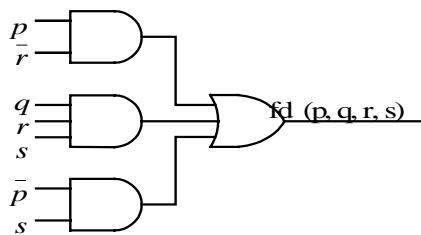
$$\overline{f(A,B,C)} = \overline{(A + B + \overline{C}) + (A + \overline{B} + C) + (\overline{A} + B + C)}$$

2.5 Synthesis of Combinational Logic Circuits (1)

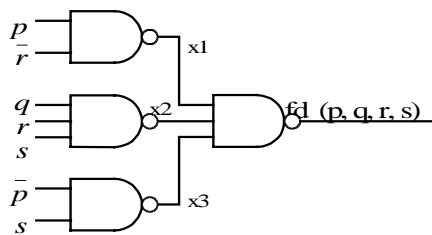
➤ AND-OR and NAND Networks

- Switching expression must be in SOP form.

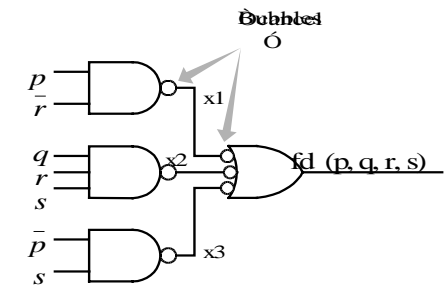
- Example: $f_{\delta}(p, q, r, s) = p\bar{r} + qrs + \bar{p}s$



(a) AND-OR network



(b) NAND network



(c) NAND network (preferred form)

- $$f_{\delta}(p, q, r, s) = \overline{\overline{p\bar{r} + qrs + \bar{p}s}} \quad [\text{T3}]$$

$$= \overline{\overline{p\bar{r}} \cdot \overline{qrs} \cdot \overline{\bar{p}s}} \quad [\text{T8(a)}]$$

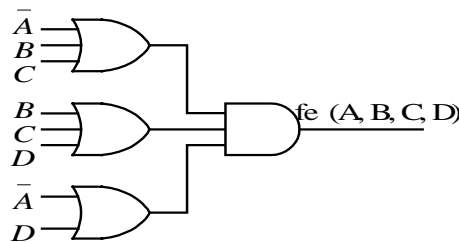
$$= x_1 \cdot x_2 \cdot x_3$$

where $x_1 = \overline{p\bar{r}}$, $x_2 = \overline{qrs}$, and $x_3 = \overline{\bar{p}s}$

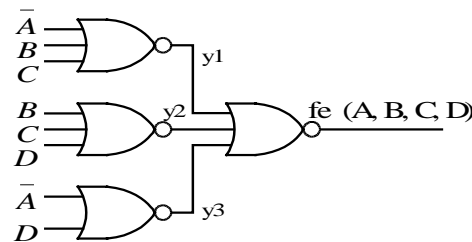
2.5 Synthesis of Combinational Logic Circuits (2)

➤ OR-AND and NOR Networks

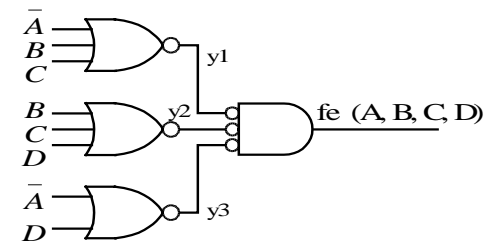
- Switching expression must be in POS form.
- Example: $f_{\mathcal{E}}(A, B, C, D) = (\bar{A} + B + C)(B + C + D)(\bar{A} + D)$



(a) OR-AND network



(b) NOR network



(c) NOR network (preferred for)

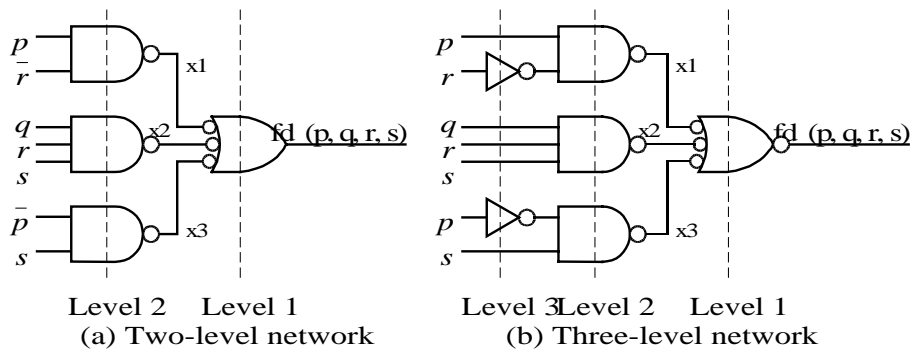
$$\begin{aligned}
 f_{\mathcal{E}}(A, B, C, D) &= \overline{\overline{(\bar{A} + B + C)(B + C + D)(\bar{A} + D)}} \\
 &= \overline{\overline{\bar{A} + B + C} + \overline{B + C + D} + \overline{\bar{A} + D}} \\
 &= \overline{y_1 + y_2 + y_3}
 \end{aligned}$$

where $y_1 = \overline{\bar{A} + B + C}$, $y_2 = \overline{B + C + D}$, and $y_3 = \overline{\bar{A} + D}$

2.5 Synthesis of Combinational Logic Circuits (3)

➤ Two-level Circuits

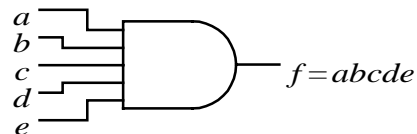
- Input signals pass through two levels of gates before reaching the output.



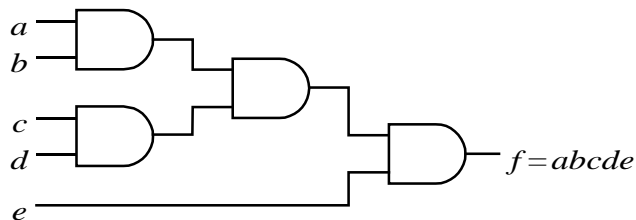
- Implementation procedure for NAND (NOR) logic:
 - Step 1. Express the function in minterm (maxterm) list form.
 - Step 2. Write out the minterms (maxterms) in algebraic form.
 - Step 3. Simplify the function in SOP (POS) form.
 - Step 4. Transform the expression into the NAND (NOR) form.
 - Step 5. Draw the NAND (NOR) logic diagram.

2.5 Synthesis of Combinational Logic Circuits (4)

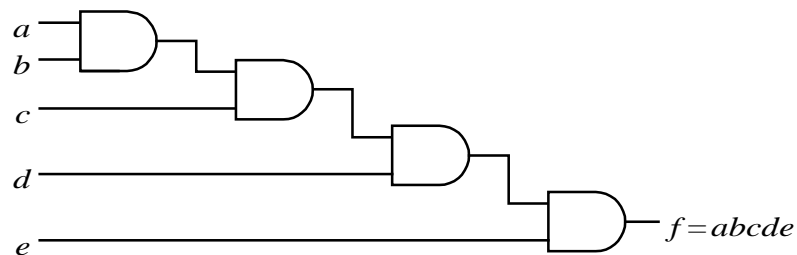
- Circuits with more than two levels are often needed due to fan-in constraints.



(a) A single five-input AND gate



(b) Three-level network of two-input gates



(c) Four-level network of two-input gates.

2.5 Synthesis of Combinational Logic Circuits (5)

➤ **Example 2.37:** NAND implementation of $f_{\phi}(X, Y, Z) = \Sigma m(0, 3, 4, 5, 7)$

$$1. f_{\phi}(X, Y, Z) = \Sigma m(0, 3, 4, 5, 7)$$

$$2. f_{\phi}(X, Y, Z) = m_0 + m_3 + m_4 + m_5 + m_7$$

$$= \overline{X}Y\overline{Z} + \overline{X}YZ + X\overline{Y}\overline{Z} + X\overline{Y}Z + XYZ$$

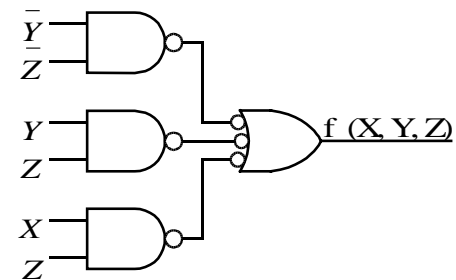
$$3. f_{\phi}(X, Y, Z) = \overline{Y}Z + YZ + XZ \quad [\text{T6(a)}]$$

$$4a. f_{\phi}(X, Y, Z) = \overline{\overline{\overline{Y}Z}} + \overline{\overline{\overline{YZ}}} + \overline{\overline{\overline{XZ}}} \quad [\text{T4}]$$

or

$$4b. f_{\phi}(X, Y, Z) = \overline{\overline{\overline{\overline{Y}Z}}} + \overline{\overline{\overline{\overline{YZ}}}} + \overline{\overline{\overline{\overline{XZ}}}} \quad [\text{T3}]$$

$$= \overline{\overline{\overline{Y}Z}} \cdot \overline{\overline{\overline{YZ}}} \cdot \overline{\overline{\overline{XZ}}} \quad [\text{T8(a)}]$$



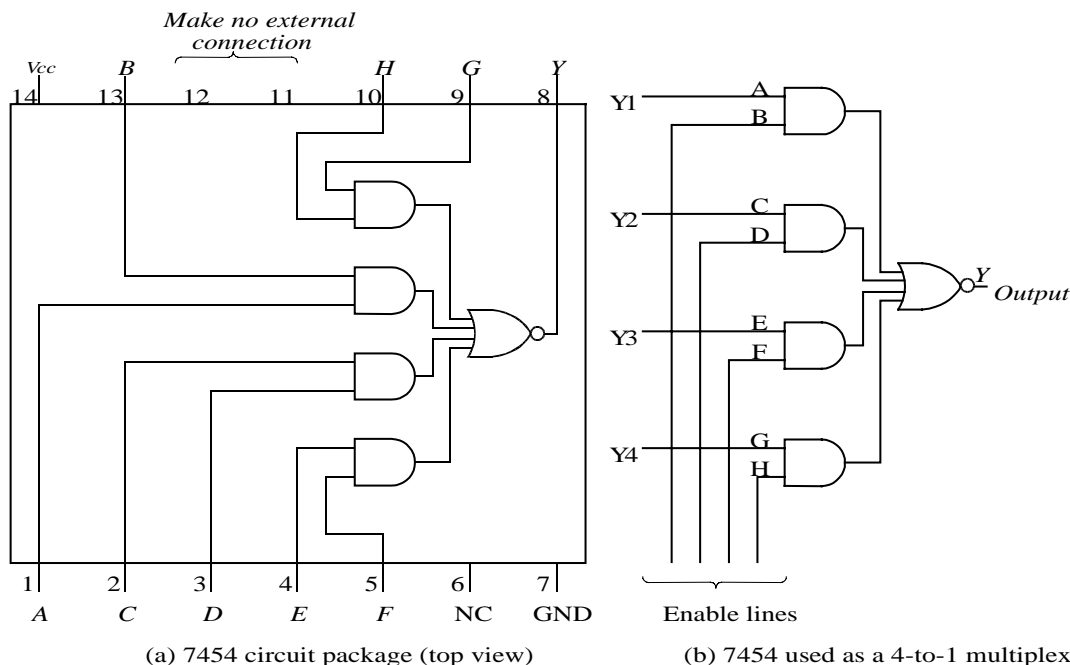
(a) NAND implementation

2.5 Synthesis of Combinational Logic Circuits (6)

➤ *AND-OR-invert Circuits*

- A set of AND gates followed by a NOR gate.
- Used to readily realize two-level SOP circuits.
- 7454 circuit:

$$F = \overline{AB + CD + EF + GH}$$



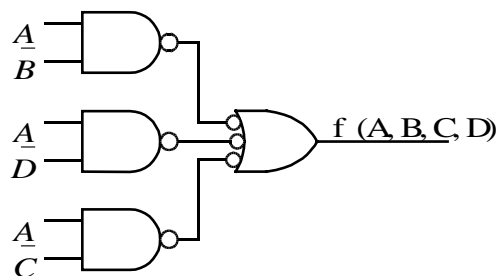
2.5 Synthesis of Combinational Logic Circuits (7)

➤ *Factoring*

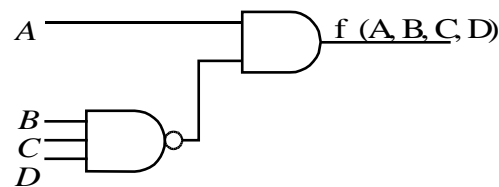
- A technique to obtain higher-level forms of switching functions.
- Higher-level forms:
 - May need less hardware
 - May be used when there are fan-in constraints
 - More difficult to design
 - Slower

➤ *Example 2.39:*

$$f(A, B, C, D) = A\bar{B} + A\bar{D} + A\bar{C} = A(\bar{B} + \bar{D} + \bar{C}) = A(\overline{BCD})$$



(a) Original form



(b) After factoring

2.5 Synthesis of Combinational Logic Circuits (8)

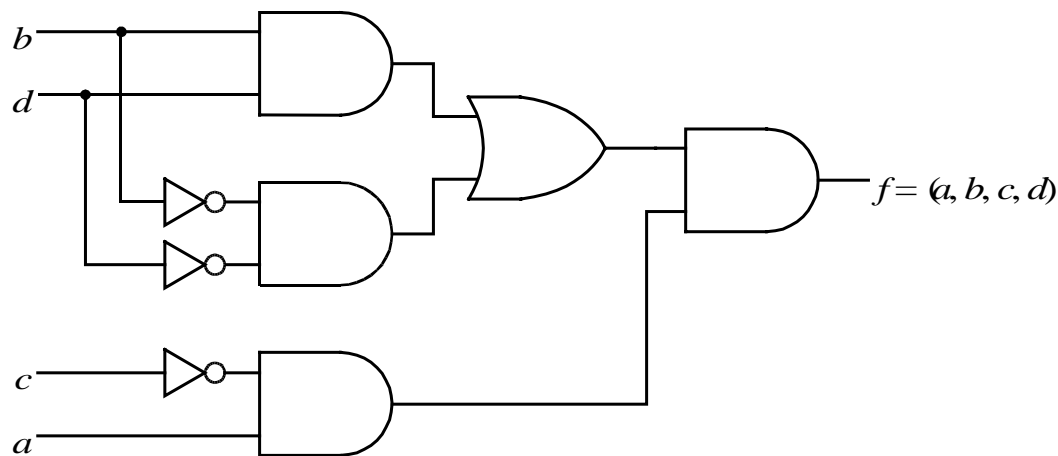
➤ **Example 2.40:** $f(a,b,c,d) = \Sigma m(8,13)$ with only two-input AND and OR gates.

- Write the canonical SOP form:

$$f(a,b,c,d) = \Sigma m(8,13) = \overline{a}\overline{b}\overline{c}\overline{d} + ab\overline{c}d \quad (2.34)$$

Two four-input AND gates and one two-input OR gate are needed.

- Apply factoring: $f(a,b,c,d) = \overline{a}\overline{b}\overline{c}\overline{d} + ab\overline{c}d = (\overline{a}\overline{c})(\overline{b}\overline{d} + bd)$ (2.35)



2.5 Synthesis of Combinational Logic Circuits (9)

➤ **Example 2.41:** A burglar alarm with four control switches, each of which produces logic 1 when:

Switch *A*: Secret switch is closed

Switch *B*: Safe is in its normal position in the closet

Switch *C*: Clock is between 1000 and 1400 hours

Switch *D*: Closet door is closed.

Write the equations of the control logic that produces logic 1 when
the safe is moved AND the secret switch is closed,
OR
the closet is opened after banking hours,
OR
the closet is opened with the control switch open.

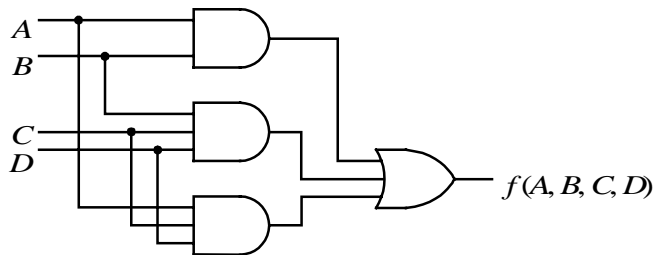
$$f(A, B, C, D) = AB + \overline{CD} + \overline{AD}$$

2.5 Synthesis of Combinational Logic Circuits (10)

➤ **Example 2.42:** The Doe family voter:

- Vote for either *hamburgers* (0) or *chicken* (1).
- Majority wins.
- If Mom and Dad agree, they win.
- John (Dad): A , Jane (Mom): B , Joe: C , Sue: D .
- The logic function is:

$$\begin{aligned}
 f(A, B, C, D) &= \bar{A}BCD + A\bar{B}CD + ABC\bar{D} + ABC\bar{D} + ABC\bar{D} + ABCD \\
 &= \bar{A}BCD + A\bar{B}CD + AB \\
 &= AB + ACD + \bar{A}BCD \\
 &= AB + ACD + BCD
 \end{aligned}$$



2.5 Synthesis of Combinational Logic Circuits (11)

➤ **Example 2.43:** Logic equations for a circuit that adds two 2-bit binary numbers $(A_1A_0)_2$ and $(B_1B_0)_2$, and produces sum bits $(S_1S_0)_2$ and carry bit C_1 ;

$$\begin{array}{r} A_1A_0 \\ + B_1B_0 \\ \hline C_1S_1S_0 \end{array}$$

2.5 Synthesis of Combinational Logic Circuits (12)

➤ Truth Table:

A1	A0	B1	B0	C1	S1	S0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	0	0

➤ Logic equations:

$$S_0 = \bar{A}_1 \bar{A}_0 \bar{B}_1 B_0 + \bar{A}_1 \bar{A}_0 B_1 B_0 + \bar{A}_1 A_0 \bar{B}_1 \bar{B}_0 \\ + \bar{A}_1 A_0 B_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 B_0 + A_1 \bar{A}_0 B_1 B_0 \\ + A_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 B_1 \bar{B}_0$$

$$S_1 = \bar{A}_1 \bar{A}_0 B_1 \bar{B}_0 + \bar{A}_1 \bar{A}_0 B_1 B_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 \\ + \bar{A}_1 A_0 B_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 B_0 \\ + A_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 B_1 B_0$$

$$C_1 = \bar{A}_1 A_0 B_1 B_0 + A_1 \bar{A}_0 B_1 \bar{B}_0 + A_1 \bar{A}_0 B_1 B_0 \\ + A_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 \bar{B}_0 + A_1 A_0 B_1 B_0$$

2.5 Synthesis of Combinational Logic Circuits (13)

➤ Reduced equations:

$$S_0 = A_0 \bar{B}_0 + \bar{A}_0 + B_0$$

$$S_1 = \bar{A}_1 \bar{A}_0 B_1 + \bar{A}_1 B_1 \bar{B}_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 \\ + A_1 A_0 B_1 B_0 + A_1 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1$$

$$C_1 = A_0 B_1 B_0 + A_1 A_0 B_0 + A_1 B_1$$