جامعة الملك سعود
King Saud University

CSC 220: Computer Organization

Unit 2
Digital Circuit Design

Prepared by:
Md Saiful Islam, PhD

Updated by:
Isra Al-Turaiki, PhD

Department of Computer Science
College of Computer and Information Sciences

Overview

- Binary Logic and gates
  - Binary Logic
  - Logic Gates
- Boolean Algebra
- Standard Forms (SOP/POS)

Chapter-2

M. Morris Mano, Charles R. Kime and Tom Martin, **Logic and Computer Design Fundamentals**, Global (5th) Edition, Pearson Education Limited, 2016. ISBN: 9781292096124

Binary Logic

- Deals with binary variables and with operations that assume logical meaning.
- Variables are designated by letters of the alphabet, such as *A*, *B*, *C*, *X*, *Y*, and *Z*.
- Basic logical operations: AND, OR, NOT

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

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		

Binary Logic

- Basic *logical operators* are the logic functions:
  - AND denoted by a dot (·) Ex:Y= A.B or AB
  - OR is denoted by a plus (+) Ex:Y= A+B
  - NOT denoted by an overbar (‾), a single quote mark (') after, or (~) before the variable. Ex:Y=A’

- Should not be confused with binary arithmetic.

Arithmetic → 1 + 1 = 2 (read “one plus one equals two”)

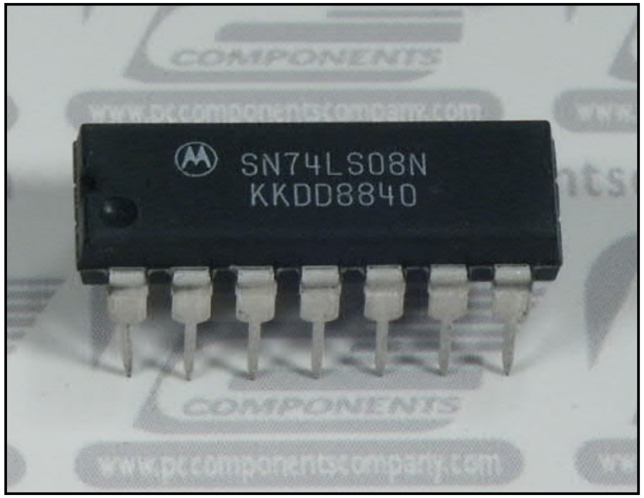
is not the same as

Logic → 1 + 1 = 1 (read “1 or 1 equals 1”).

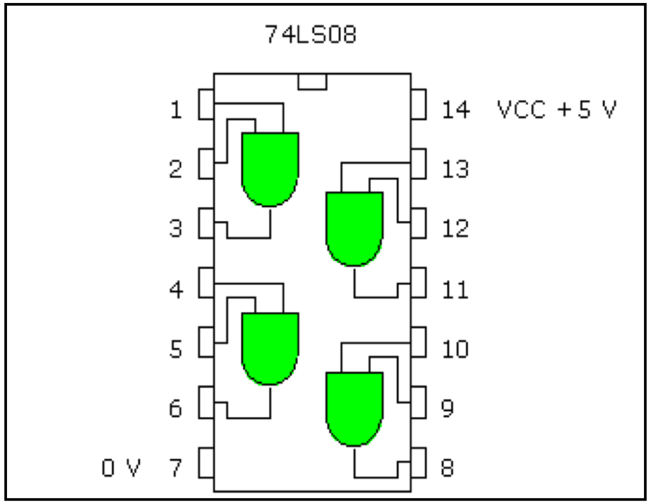
Digital Circuit

- Digital Circuit (hardware) manipulate binary information
  - **Input-output:** one or more binary values
  - Hardware consists of a few simple building blocks called *logic gates*
  - **Logic gate:** a electronic device the operates on one or more input signals and produce an output.
    - **Basic Logic gates:** AND, OR, NOT, …
    - **Additional gates:** NAND, NOR, XOR, XNOR…
- Logic gates are built using *transistors*
  - NOT gate can be implemented by a single transistor
  - AND-OR gate requires 3 transistors
- Transistors are the fundamental devices
  - Pentium consists of 3 million transistors
  - Compaq Alpha consists of 9 million transistors
  - Now we can build chips with more than 100 million transistors

Integrated Circuits



Quadruple AND Chip

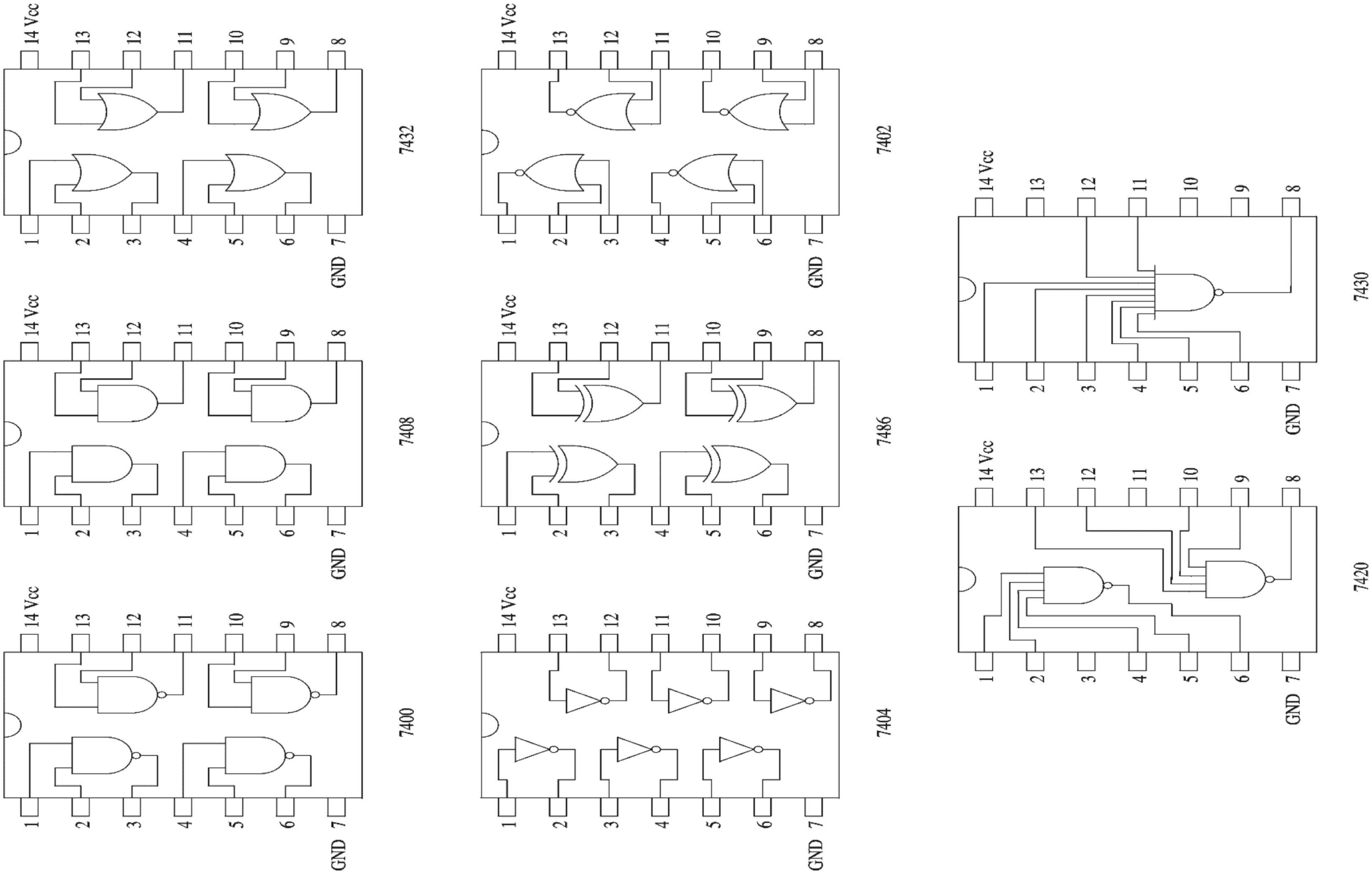


Logic Diagram of a Quadruple AND Chip

Levels of Integration

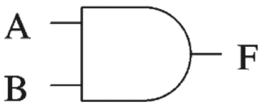
- Integration levels
  - SSI (small scale integration)
    - Introduced in late 1960s
    - 1-10 gates (previous examples)
  - MSI (medium scale integration)
    - Introduced in late 1960s
    - 10-100 gates
  - LSI (large scale integration)
    - Introduced in early 1970s
    - 100-10,000 gates
  - VLSI (very large scale integration)
    - Introduced in late 1970s
    - More than 10,000 gates

Typical SSI Circuits



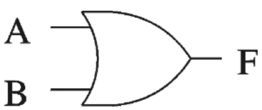
Logic Gates

- Basic gates
  - **AND**
  - **OR**
  - **NOT**
- Functionality can be expressed by a truth table
  - A truth table lists output for each possible input combination
- Precedence
  - NOT > AND > OR
  - $F = A\overline{B} + \overline{A}B$
    $= (A(\overline{B})) + ((\overline{A})B)$



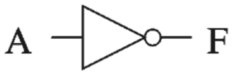
AND gate

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



OR gate

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



NOT gate

A	F
0	1
1	0

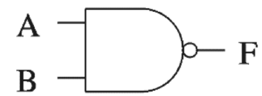
Logic symbol

Truth table

Additional Logic Gates

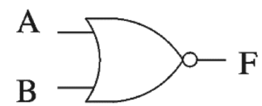
- NAND= = AND + NOT
- NOR = OR + NOT
- NAND and NOR gates require only 2 transistors
  - AND and OR need 3 transistors!

- XOR: exclusive-OR
- XNOR: complement of XOR



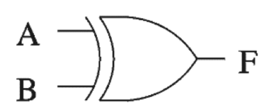
NAND gate

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



NOR gate

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



XOR gate

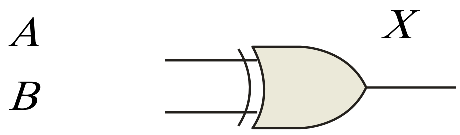
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

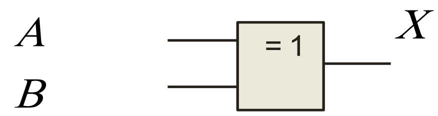
Logic symbol

Truth table

XOR Gate-1

The XOR Gate





The **XOR gate** produces a HIGH output only when the inputs are at opposite logic levels. The truth table is

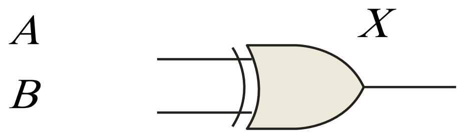
Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

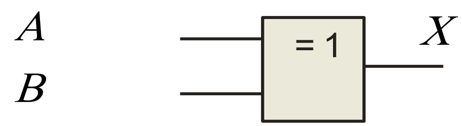
The **XOR** operation is written as $X = \overline{A}B + A\overline{B}$. Alternatively, it can be written with a circled plus sign between the variables as

$$X = A \oplus B.$$

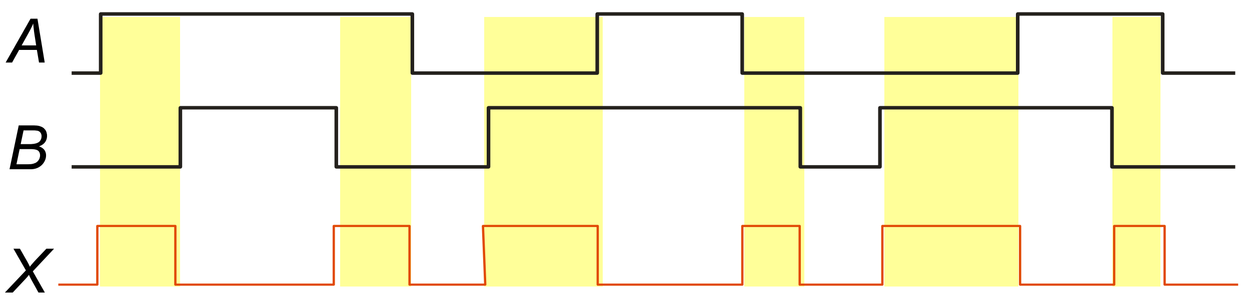
XOR Gate-2

The XOR Gate





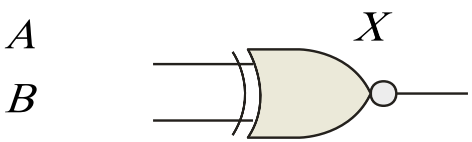
Example waveforms:

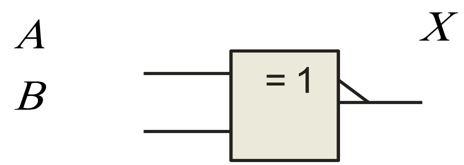


Notice that the XOR gate will produce a HIGH only when exactly one input is HIGH.

XNOR Gate-1

The XNOR Gate





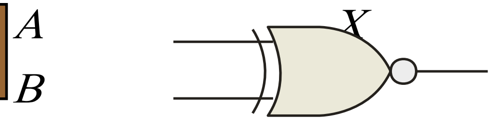
The **XNOR gate** produces a HIGH output only when the inputs are at the same logic level. The truth table is

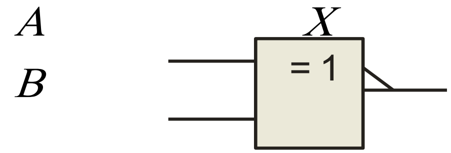
Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

The **XNOR** operation can be shown as $X = AB + \bar{A}\bar{B}$.

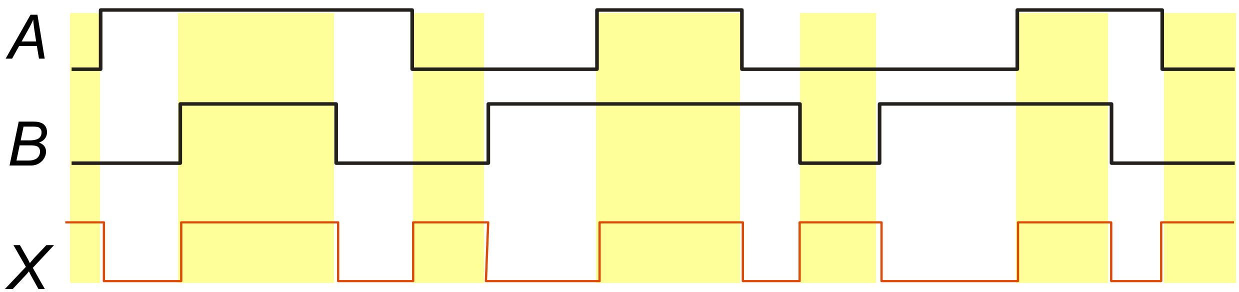
XNOR Gate-2

The XNOR Gate





Example waveforms:



Notice that the XNOR gate will produce a HIGH when both inputs are the same. This makes it useful for comparison functions.

Boolean Algebra

- A Boolean function consists of
  - Binary variables
  - Constants 0, 1
  - Logic operators: AND (.), OR (+), NOT(-), …
- A function with N input variables
  - With *N* logical variables, we can define

    2^N combination of inputs
- A Boolean function can be:
  - single-output function
  - multiple-output function

Boolean Algebra

- Designing a Logic Circuit
  - A truth table is used to represent a logic function
  - Logical expressions can be obtained from truth table
  - Logical expressions can be transfer to logic diagram of the circuit

- Example:
  - Majority function
    - Output is one whenever majority of inputs is 1
    - We use 3-input majority function

Boolean Algebra

Truth Table:

3-input majority function

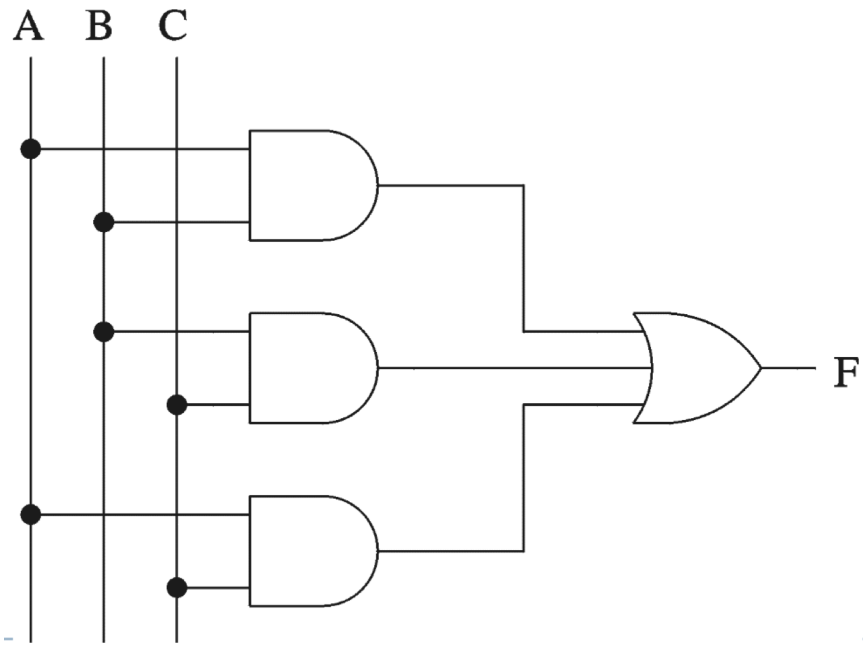
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Logical expression form

F = A’BC + AB’C + ABC’ + ABC

= AB + BC + AC (after simplification)

Logic diagram



Boolean Algebra

- An algebraic structure defined on a set of at least two elements, (X,Y) together with three binary operators (denoted +, · and $\overline{\ }$) 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

Boolean Algebra

- An algebraic structure defined on a set of at least two elements, (X,Y) 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.	$\overline{\overline{X}} = X$		

between a single variable X, its complement X, and the binary constants 0 and 1.

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

Boolean Algebra

- An algebraic structure defined on a set of at least two elements, (X,Y) together with three binary operators (denoted +, · and $\overline{\ }$) 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

have counterparts in ordinary algebra

Boolean Algebra

- The *dual* of an algebraic expression is obtained by interchanging OR and AND operations and replacing 1s by 0s and 0s by 1s.

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.	$X \cdot \overline{X} = 0$	do not apply in ordinary algebra
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

Boolean Algebra

$F1 = xy\bar{z}$

$F2 = x + \bar{y}z$

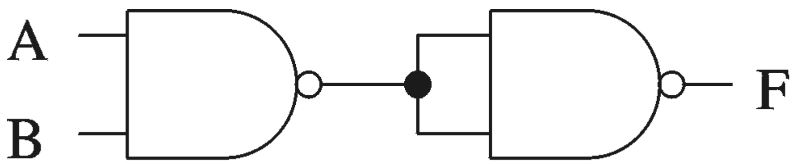
$F3 = \bar{x}\bar{y}\bar{z} + \bar{x}yz + x\bar{y}$

$F4 = x\bar{y} + \bar{x}z$

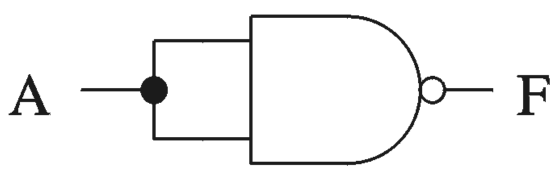
x	y	z	F1	F2	F3	F4
0	0	0	0	0	1	0
0	0	1	0	1	0	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

Universal Gates-1

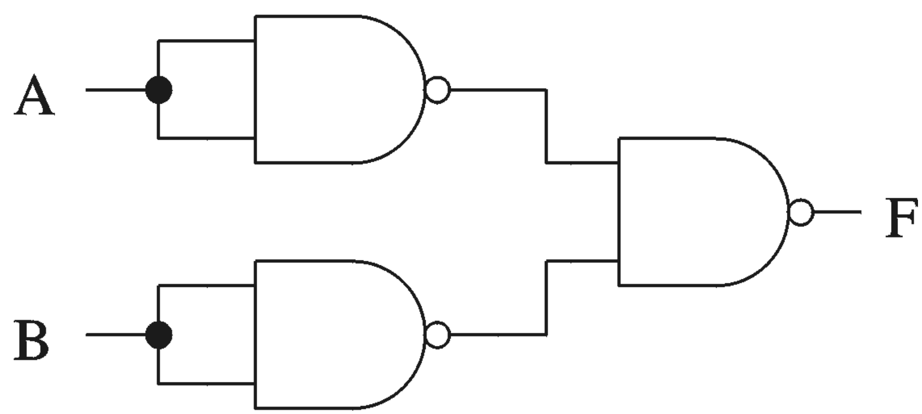
- NAND and NOR gates are called *universal gates*
- ***Proving NAND gate is universal***



AND gate



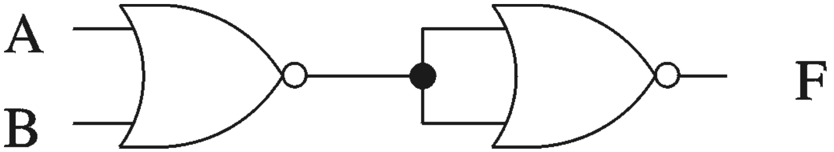
NOT gate



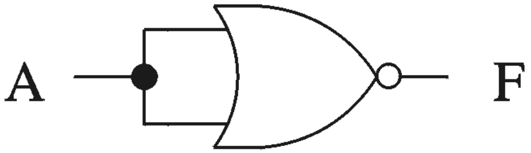
OR gate

Universal Gates-2

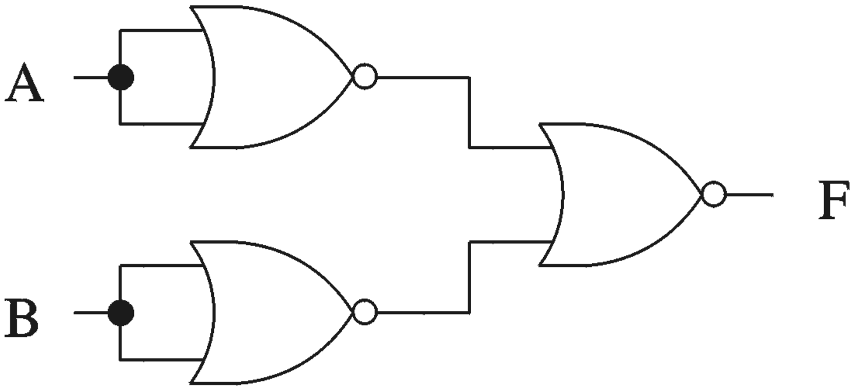
- *Proving NOR gate is universal*



OR gate



NOT gate



AND gate

Standard Forms

Standard Forms Boolean Expressions

- Sum-of-Products (SOP)
  - Derived from the Truth table for a function by considering those rows for which F = 1.
  - The logical sum (OR) of product (AND) terms.
  - Realized using an AND-OR circuit.
- Product-of-Sums (POS)
  - Derived from the Truth table for a function by considering those rows for which F = 0.
  - The logical product (AND) of sum (OR) terms.
  - Realized using an OR-AND circuit.

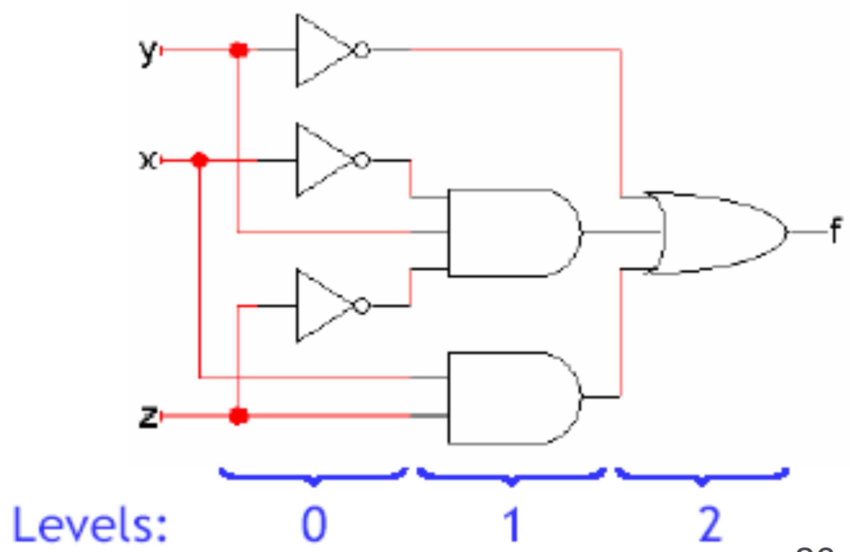
Sum-of-Products

Sum of products expressions

- There are many equivalent ways to write a function, but some forms turn out to be more useful than others.
- A sum of products or SOP expression consists of:
  - One or more terms *summed* (OR'ed) together.
  - Each of those terms is a *product of literals*.

$$f(x, y, z) = y' + x'yz' + xz$$

- Sum of products expressions can be implemented with two-level circuits.



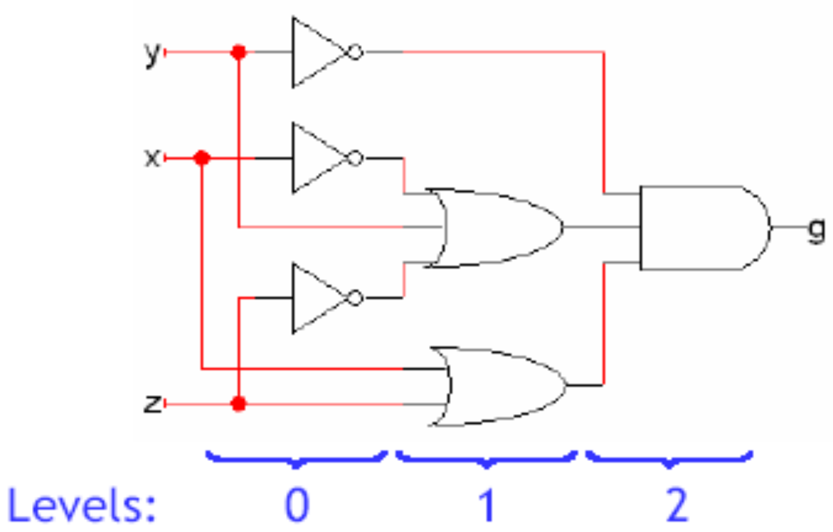
Product-of-Sums

Product of sums expressions

- As you might expect, we can work with the duals of these ideas too.
- A product of sums or POS consists of:
  - One or more terms *multiplied* (AND'ed) together.
  - Each of those terms is a *sum of literals*.

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

- Products of sums can also be implemented with two-level circuits.



Minterms

- A minterm is a special product of literals, in which each input variable appears exactly once.
- A function with n input variables has 2^n possible minterms.
- For instance, a three-variable function f(x,y,z) has 8 possible minterms:

x'y'z'	x'y'z	x'y z'	x'y z
x y'z'	x y'z	x y z'	x y z

- Each minterm is true for exactly one combination of inputs.

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \overline{x}_1\overline{x}_2\overline{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \overline{x}_1\overline{x}_2 x_3$	$M_1 = x_1 + x_2 + \overline{x}_3$
2	0	1	0	$m_2 = \overline{x}_1 x_2\overline{x}_3$	$M_2 = x_1 + \overline{x}_2 + x_3$
3	0	1	1	$m_3 = \overline{x}_1 x_2 x_3$	$M_3 = x_1 + \overline{x}_2 + \overline{x}_3$
4	1	0	0	$m_4 = x_1\overline{x}_2\overline{x}_3$	$M_4 = \overline{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\overline{x}_2 x_3$	$M_5 = \overline{x}_1 + x_2 + \overline{x}_3$
6	1	1	0	$m_6 = x_1 x_2\overline{x}_3$	$M_6 = \overline{x}_1 + \overline{x}_2 + x_3$
7	1	1	1	$m_7 = x_1 x_2 x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

Minterms

TABLE 2-9
Minterms for Three 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	$\overline{X}\,\overline{Y}\,\overline{Z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\overline{X}\,\overline{Y}Z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\overline{X}Y\overline{Z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\overline{X}YZ$	m_3	0	0	0	1	0	0	0	0
1	0	0	$X\overline{Y}\,\overline{Z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$X\overline{Y}Z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$XY\overline{Z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	XYZ	m_7	0	0	0	0	0	0	0	1

Sum-of-Minterms

Sum of minterms expressions

- A sum of minterms is a special kind of sum of products.
- Every function can be written as a *unique* sum of minterms expression.
- A truth table for a function can be rewritten as a sum of minterms just by finding the table rows where the function output is 1.

x	y	z	C(x,y,z)	C'(x,y,z)
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$$C = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7 = \Sigma m(3,5,6,7)$$

$$C' = x'y'z' + x'y'z + x'yz' + xy'z' = m_0 + m_1 + m_2 + m_4 = \Sigma m(0,1,2,4)$$

C' contains all the minterms *not* in C, and vice versa.

Sum-of-Minterms

- Any function F can be represented by a sum of minterms, where each minterm is ANDed with the corresponding value of the output for F.
  - $F = \Sigma (m_i . f_i)$
    - where m_i is a minterm
    - and f_i is the corresponding functional output

Denotes the logical sum operation

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

$$\begin{aligned} f &= m_0 \cdot 1 + m_1 \cdot 1 + m_2 \cdot 0 + m_3 \cdot 1 \\ &= m_0 + m_1 + m_3 \\ &= \bar{x}_1\bar{x}_2 + \bar{x}_1 x_2 + x_1 x_2 \end{aligned}$$

  - Only the minterms for which $f_i = 1$ appear in the expression for function F.
  - $F = \Sigma (m_i) = \Sigma\, m(i)$ ← shorthand notation

Sum-of-Minterms

- Sum of minterms are a.k.a. Canonical Sum-of-Products
- Synthesis process
  - Determine the Canonical Sum-of-Products
  - Use Boolean Algebra (and K-maps) to find an optimal, functionally equivalent, expression.

Maxterms

- A maxterm is a *sum* of literals where each input variable appears once.
- A function with *n* input variables has 2^n possible maxterms.
- For instance, a function with three variables x, y and z has 8 possible maxterms:

x + y + z	x + y + z’	x + y’+ z	x + y’+ z’
x’+ y + z	x’+ y + z’	x’+ y’+ z	x’+ y’+ z’

- Each maxterm is *false* for exactly one combination of inputs.

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \overline{x}_1\overline{x}_2\overline{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \overline{x}_1\overline{x}_2 x_3$	$M_1 = x_1 + x_2 + \overline{x}_3$
2	0	1	0	$m_2 = \overline{x}_1 x_2\overline{x}_3$	$M_2 = x_1 + \overline{x}_2 + x_3$
3	0	1	1	$m_3 = \overline{x}_1 x_2 x_3$	$M_3 = x_1 + \overline{x}_2 + \overline{x}_3$
4	1	0	0	$m_4 = x_1\overline{x}_2\overline{x}_3$	$M_4 = \overline{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\overline{x}_2 x_3$	$M_5 = \overline{x}_1 + x_2 + \overline{x}_3$
6	1	1	0	$m_6 = x_1 x_2\overline{x}_3$	$M_6 = \overline{x}_1 + \overline{x}_2 + x_3$
7	1	1	1	$m_7 = x_1 x_2 x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

Maxterms

TABLE 2-10
Maxterms for Three 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

Product-of-Maxterms

Product of maxterms expressions

- Every function can also be written as a unique product of maxterms.
- A truth table for a function can be rewritten as a product of maxterms just by finding the table rows where the function output is 0.

x	y	z	C(x,y,z)	C'(x,y,z)
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$$C = (x + y + z)(x + y + z')(x + y' + z)(x' + y + z)$$
$$= M_0 M_1 M_2 M_4$$
$$= \prod M(0,1,2,4)$$ **When the o/p is Zero**
$$= \Sigma m(3,5,6,7)$$ **When the o/p is 1**

$$C' = (x + y' + z')(x' + y + z')(x' + y' + z)(x' + y' + z')$$
$$= M_3 M_5 M_6 M_7$$
$$= \prod M(3,5,6,7)$$

C' contains all the maxterms *not* in C, and vice versa.

Product-of-Maxterms

- Any function F can be represented by a product of Maxterms, where each Maxterm is ANDed with the *complement* of the corresponding value of the output for F.
  - $F = \prod (M_i \cdot f'_i)$
    - where M_i is a Maxterm
    - and f'_i is the complement of the corresponding functional output

Denotes the logical product operation

  - Only the Maxterms for which $f_i = 0$ appear in the expression for function F.
  - $F = \prod (M_i) = \prod M(i)$ ← shorthand notation

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

$f = M_2 = X_1 + \overline{X_2}$

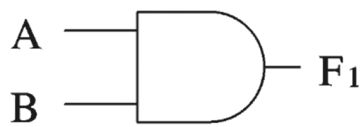
Product-of-Maxterms

- The Canonical Product-of-Sums for function F is the Product-of-Sums expression in which each sum term is a Maxterm.
- Synthesis process
  - Determine the Canonical Product-of-Sums
  - Use Boolean Algebra (and K-maps) to find an optimal, functionally equivalent, expression.

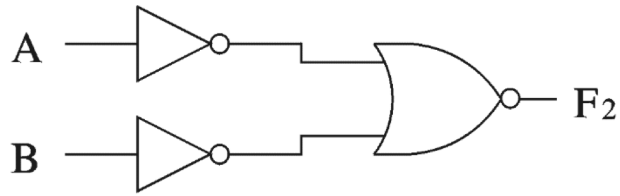
Logical Equivalence

- When two circuits implement same logic function

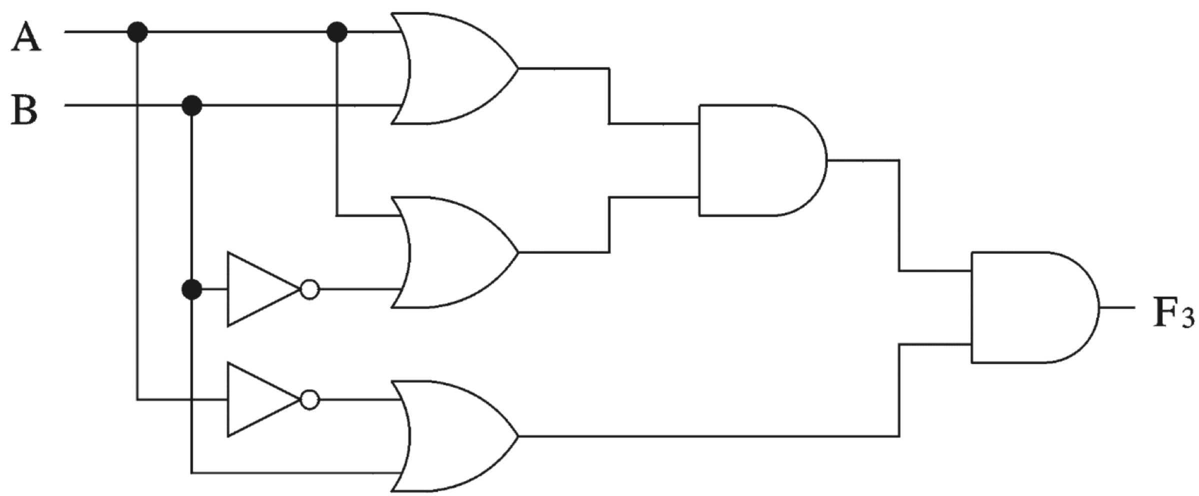
Example: All three circuits implement F = A B function



(a)



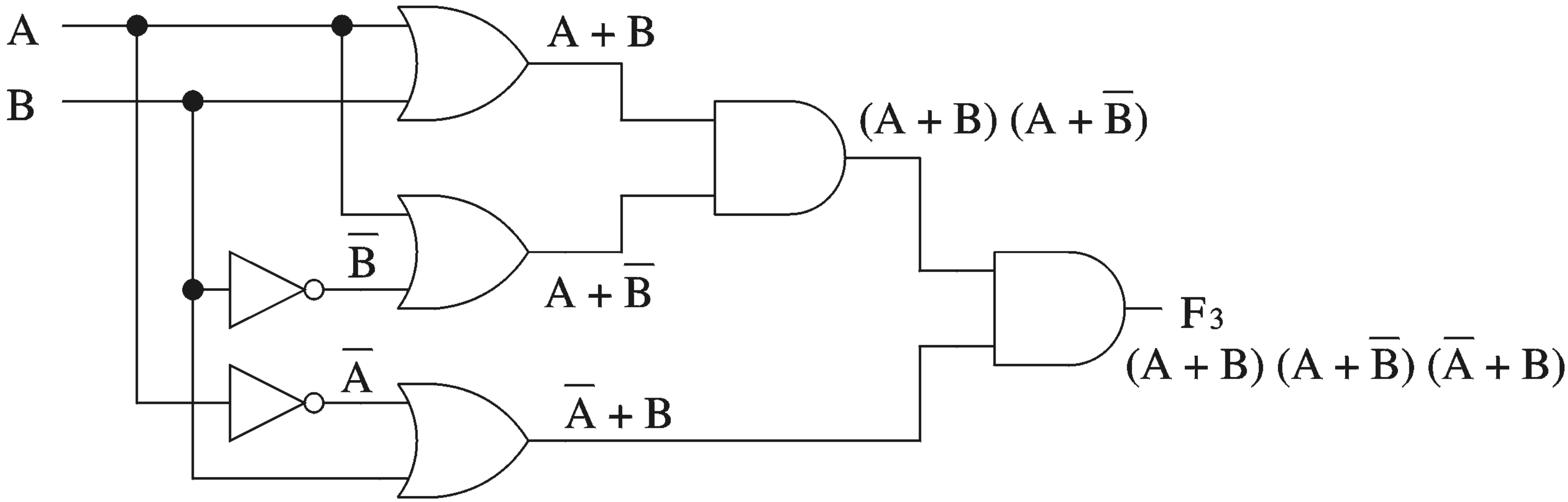
(b)



(c)

Logical Equivalence …

- Proving logical equivalence:
- Derivation of logical expression from a circuit
  - Trace from the input to output
  - Write down intermediate logical expressions along the path



Logical Equivalence …

- Build the truth table relating inputs to the output for each circuit
- If each function give the same output, they are logically equivalent

A	B	F1 = A B	F3 = $(A + B)(\overline{A} + B)(A + \overline{B})$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

- Exercise:
  - Show that $X \oplus Y$ is logically equivalent to X'Y+XY'