Boolean Algebra and Logic Gates

Aziz Qaroush

Basic Definitions

- Boolean Algebra defined with a set of elements, a set of operators and a number of axioms (البديهيات) or postulates (المسلمات).
- A set if a collection of objects having a common property

x, y	Elements
$x \in S$	*x* is an element of set *S*
$y \notin S$	*y* is not an element of set *S*
$a * b = c$ $a, b, c \in S$	* Binary operator and result *c* of operation on a and b is an element of set S

Basic Definitions

1. Closure: A set S is closed with respect to a binary operator if, for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S.
2. Associative law: $(x*y)*z=x*(y*z)$ for all $z, y, z \in S$.
3. Commutative law: $x*y=y*x$
4. Identity Element: $e*x=x*e=x$
5. Inverse: A set S having the identity element e with respect to binary operator * is said to have an inverse whenever, for every $x \in S$, there exists an element y such that $x*y=e$
6. Distributive law: If * and . are binary operators on S, * is said to be distributive over . whenever $x*(y.z)=(x*y).(x*z)$

Axiomatic Definition of Boolean Algebra

Boolean algebra is an algebraic structure defined by a set of elements, *B*, together with binary operators (+) and (.)

1. (a) The structure is closed with respect to +
   (b) The structure is closed with respect to .
2. (a) The element 0 is an identity element with respect to +
   (b) The element 1 is an identity element with respect to .
3. (a) The structure is commutative with respect to +
   (b) The structure is commutative with respect to .
4. (a) The operator . is distributive over +
   (b) The operator + is distributive over .
5. For every element $x \in B$, there exists an element $x' \in B$ called the complement of x such that

   (a) $x+x'=1$ and

   (b) $x.x'=0$
6. There exist at least two elements $x, y \in B$ such that $x \neq y$

Two-valued Boolean Algebra

Defined on a set of two elements $B = \{0,1\}$

x	y	$x.y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

Two-valued Boolean Algebra

x	y	z	y+z	x.(y+z)	x.y	x.z	(x.y)+(x.z)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Duality Principle

- The **dual** of a Boolean expression can be obtained by:
  - Interchanging AND ($\cdot$) and OR (+) operators
  - Interchanging 0's and 1's
- Example: the dual of $x(y + z')$ is $x + yz'$
  - The complement operator does not change
- The properties of Boolean algebra appear in **dual pairs**
  - If a property is proven to be true then its dual is also true

	Property	Dual Property
Identity	$x + 0 = x$	$x \cdot 1 = x$
Complement	$x + x' = 1$	$x \cdot x' = 0$
Distributive	$x\,(y + z) \;= xy + xz$	$x + yz \;= (x + y)(x + z)$

Basic 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)	$x(x + y) = x$

Theorem 1(a)

$$x + x = x$$

$$x + x = (x + x) \cdot 1$$

but

$$1 = x + x'$$

thus

$$x + x = (x + x)(x + x')$$

Distributing + over . gives in general

$$x + (xy) = x + xy = (x + x)(x + y)$$

$$x + x = (x + x)(x + x') = x + xx'$$

$$= x + 0$$

$$= x$$

Theorem 1(b)

$$x \cdot x = x$$

$$\begin{aligned} x \cdot x &= xx + 0 \\ &= xx + xx' \\ &= x(x + x') \\ &= x \cdot 1 \\ &= x \end{aligned}$$

It can be proved by duality of Theorem 1(a)

Theorem 2(a)

$$x + 1 = 1$$

$$\begin{aligned} x + 1 &= 1 \cdot (x + 1) \\ &= (x + x')(x + 1) \\ &= x + x' \cdot 1 \\ &= x + x' \\ &= 1 \end{aligned}$$

Theorem 2(b)

$$x \cdot 0 = 0$$

By duality of Theorem 2(a)

Theorem 3

$$(x')' = x$$

$x + x' = 1$ and $x \cdot x' = 0$

Both equations define the complement

The complement of x' is x and is also $(x')'$

Since the complement is unique $(x')' = x$

Theorem 6(a) Absorption

$$x + xy = x$$

$$\begin{aligned} x + xy &= x \cdot 1 + xy \\ &= x(1 + y) \\ &= x(y + 1) \\ &= x \cdot 1 \\ &= x \end{aligned}$$

Theorem 6(b) Absorption

$$x(x + y) = x$$

By duality of Theorem 6(a)

Theorem 6(a) Absorption

Proof by truth table

x	y
0	0
0	1
1	0
1	1

xy	$x+xy$
0	0
0	0
0	1
1	1

DeMorgan's Theorem

- $(x + y)' = x' \, y'$
- $(x \, y)' = x' + y'$

Can be verified
Using a Truth Table

x	y	x'	y'	x+y	(x+y)'	x'y'	x y	(x y)'	x'+ y'
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

Identical **Identical**

- Generalized DeMorgan's Theorem:
- $(x_1 + x_2 + \cdots + x_n)' = x_1' \cdot x_2' \cdot \; \cdots \; \cdot \; x_n'$
- $(x_1 \cdot x_2 \cdot \cdots \cdot x_n)' = x_1' + x_2' + \; \cdots + \; x_n'$

Boolean Functions

- Boolean functions are described by expressions that consist of:
  - Boolean variables, such as: x, y, etc.
  - Boolean constants: 0 and 1
  - Boolean operators: AND ($\cdot$), OR (+), NOT (')
  - Parentheses, which can be nested
- Example: $f = x(y + w'z)$
  - The dot operator is implicit and need not be written
- Operator precedence: to avoid ambiguity in expressions
  - Expressions within parentheses should be evaluated first
  - The NOT (') operator should be evaluated second
  - The AND ($\cdot$) operator should be evaluated third
  - The OR (+) operator should be evaluated last

Truth Table

- A truth table can represent a Boolean function
- List all possible combinations of 0's and 1's assigned to variables
- If *n* variables then 2^n rows

$$F_1 = x + y'z$$

$$F_2 = x'y'z + x'yz + xy'$$

Table 2.2
Truth Tables for F_1 and F_2

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

Boolean functions

Transform the algebraic equation of F_1 to a circuit diagram using logic gates

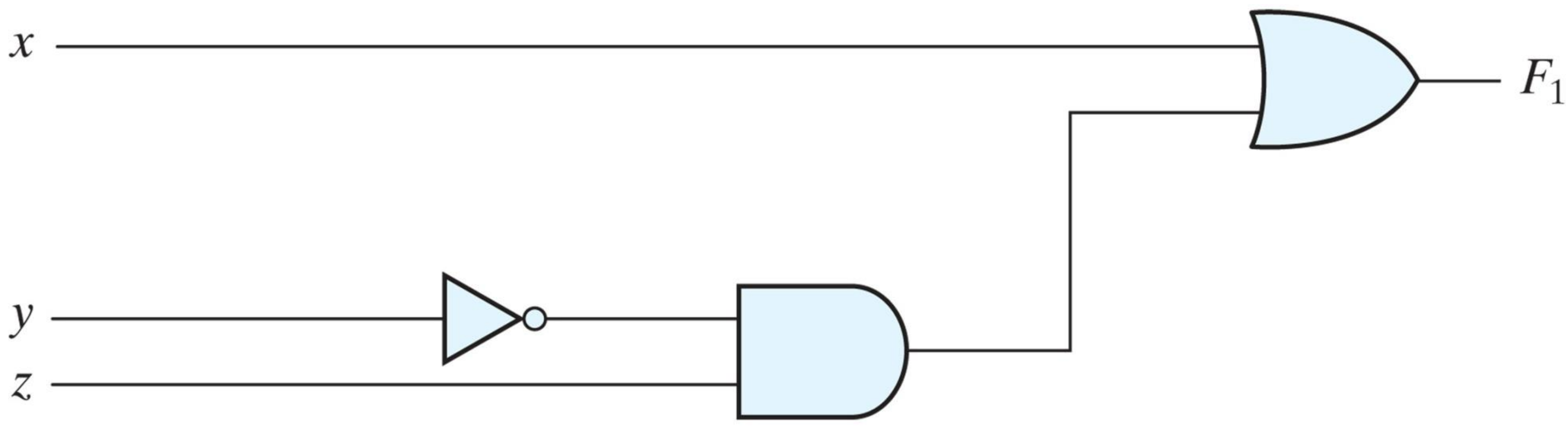
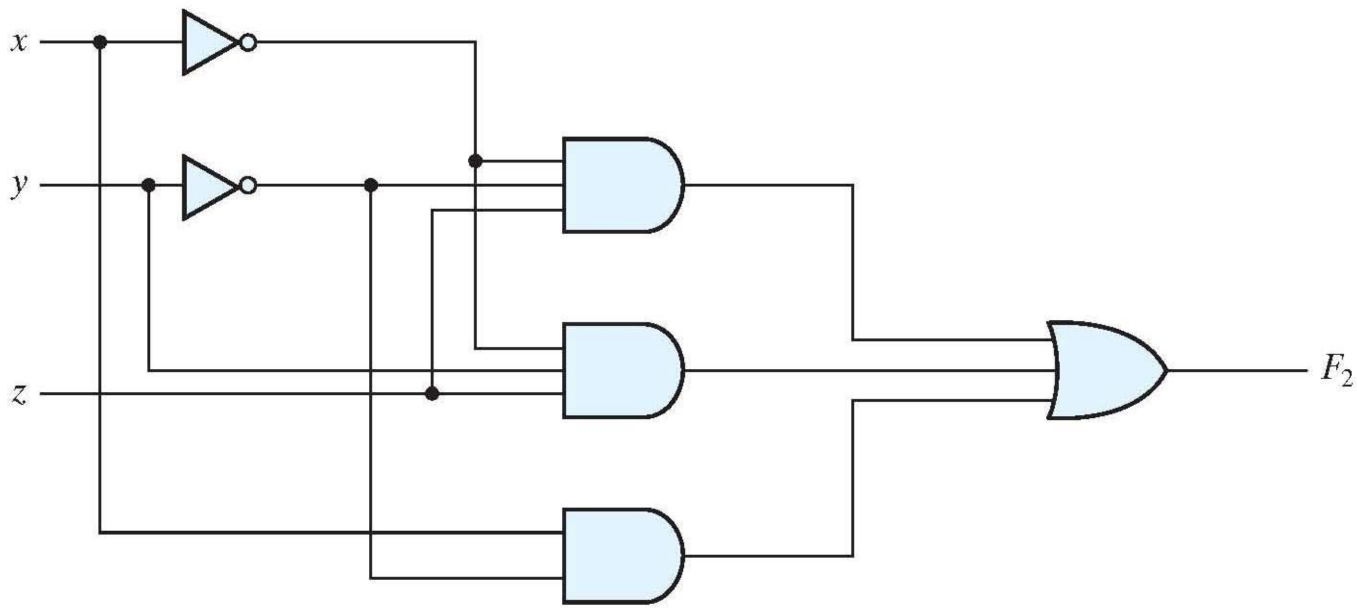


FIGURE 2.1 Gate implementation of $F_1 = x + y'z$

Boolean functions

Transform the algebraic equation of F_2 to a circuit diagram using logic gates



(a) $F_2 = x'y'z + x'yz + xy'$

Algebraic manipulation (معالجة)

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

Use Boolean Algebra to simplify Boolean functions to produce simpler circuits (minimum number of literals)

Example 1 $x(x' + y) = xx' + xy = 0 + xy = xy$

Example 2 $x + x'y = (x + x')(x + y) = 1(x + y)$

$= x + y$

Example 3 $(x + y)(x + y') = x + xy + xy' + yy'$

$= x(1 + y + y') + 0 = x$

Example 5 $(x + y)(x' + z)(y + z)$

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

By duality of Consensus Theorem

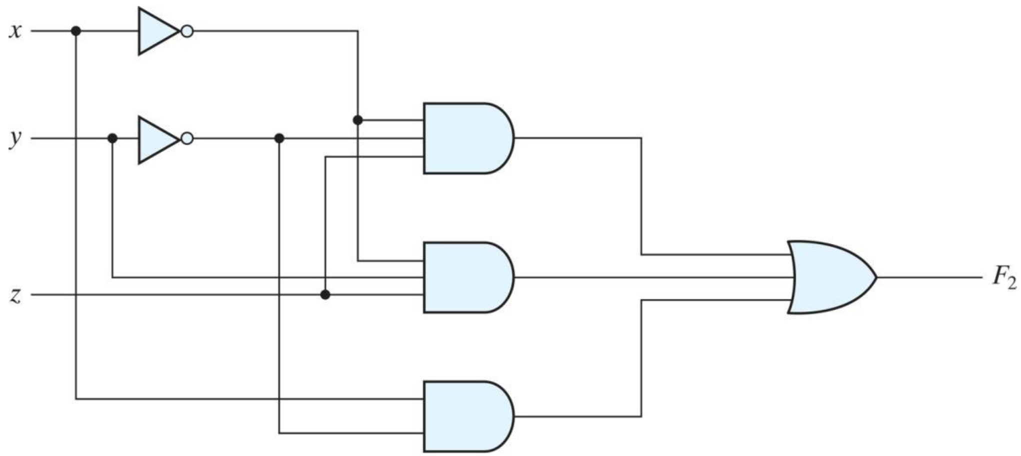
Consensus Theorem

- **Prove that:** $xy + x'z + yz = xy + x'z$ (consensus theorem)
- **Proof:** $xy + x'z + yz$

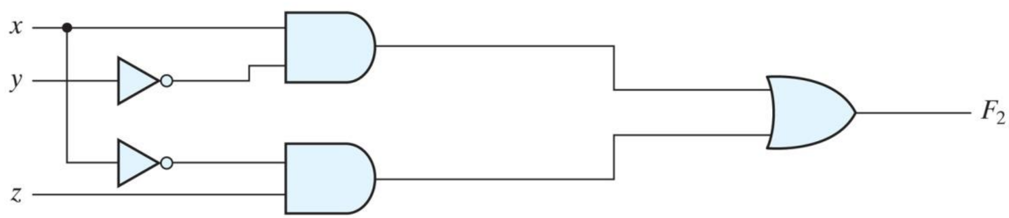
$= xy + x'z + 1 \cdot yz$	$yz = 1 \cdot yz$
$= xy + x'z + (x + x')yz$	$1 = (x + x')$
$= xy + x'z + xyz + x'yz$	Distributive $\cdot$ over +
$= xy + xyz + x'z + x'yz$	Associative commutative +
$= xy \cdot 1 + xyz + x'z \cdot 1 + x'zy$	$xy = xy \cdot 1, \;\; x'yz = x'zy$
$= xy(1 + z) + x'z(1 + y)$	Distributive $\cdot$ over +
$= xy \cdot 1 + x'z \cdot 1$	$1 + z = 1, \;\; 1 + y = 1$
$= xy + x'z$	$xy \cdot 1 = xy, \;\; x'z \cdot 1 = x'z$

Boolean functions

Compare the two implementations



(a) $F_2 = x'y'z + x'yz + xy'$



(b) $F_2 = xy' + x'z$

FIGURE 2.2 Implementation of Boolean function F_2 with gates

Complementing Boolean Functions

- What is the complement of $f = x'yz' + xy'z'$?
- Use DeMorgan's Theorem:
  - Complement each variable and constant
  - Interchange AND and OR operators
- So, what is the complement of $f = x'yz' + xy'z'$?

  Answer: $f' = (x + y' + z)(x' + y + z)$
- **Example 2:** Complement $g = (a' + bc)d' + e$
- **Answer:** $g' = (a(b' + c') + d)e'$

Complement of a function

Find the complement of the following functions

$$F_1' = (x'yz' + x'y'z)'$$
$$= (x'yz')'(x'y'z)'$$
$$= (x + y' + z)(x + y + z')$$

$$F_2' = [x(y'z' + yz)]'$$
$$= x' + (y'z' + yz)'$$
$$= x' + (y'z')'(yz)'$$
$$= x' + (y + z)(y' + z')$$
$$= \boldsymbol{x' + yz' + y'z}$$

Complement of a function

Find the complement of the following functions by taking their duals and complementing each literal

$$F_1 = x'yz' + x'y'z$$

The dual $(x' + y + z')(x' + y' + z)$

Complement each literal $(x + y' + z)(x + y + z') = F_1'$

$$F_2 = x(y'z' + yz)$$

The dual $x + (y' + z')(y + z)$

Complement each literal $x' + (y + z)(y' + z') = F_2'$

Simplification Example

- Example: Show that the following equality holds

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

- Simplification

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

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

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

Simplification Example

Example

Simplify the following function

i. $\mathbf{G} = \overline{\left(\overline{(A+\overline{B}+C)}.(\overline{AB}+\overline{C}\,\overline{D}) + \overline{ACD} \right)}$

Solution:

$$G = \overline{\left(\overline{(A+\overline{B}+C)}.(\overline{AB}+\overline{C}\,\overline{D}) + \overline{ACD} \right)}$$

$$= \left((A+\overline{B}+C) + \left(AB.(C+D)\right) \right).ACD$$

$$=(A+\overline{B}+C).ACD + \left(AB.(C+D)\right).ACD$$

$$= \ (ACD + ACD\overline{B}) + (ACDB + ACDB)$$

$$= \ ACD + ACDB$$

$$= ACD$$

Canonical Forms

- Minterms and Maxterms
- Sum-of-Minterm (SOM) Canonical Form
- Product-of-Maxterm (POM) Canonical Form
- Representation of Complements of Functions
- Conversions between Representations

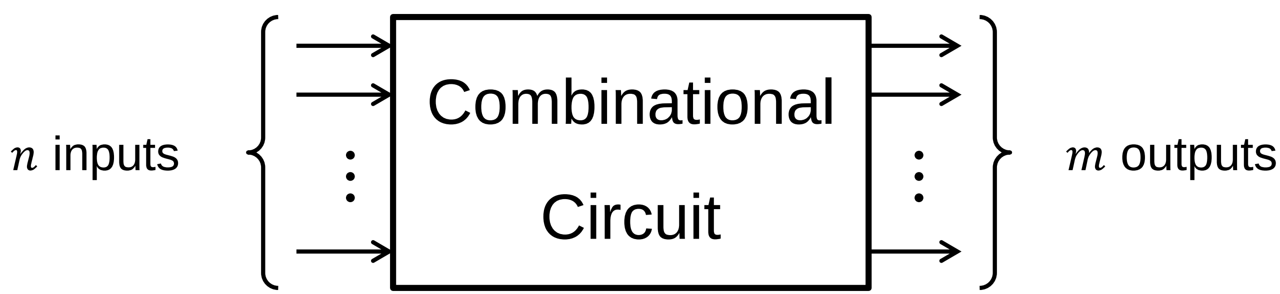
Combinational Circuit

- A combinational circuit is a block of logic gates having:

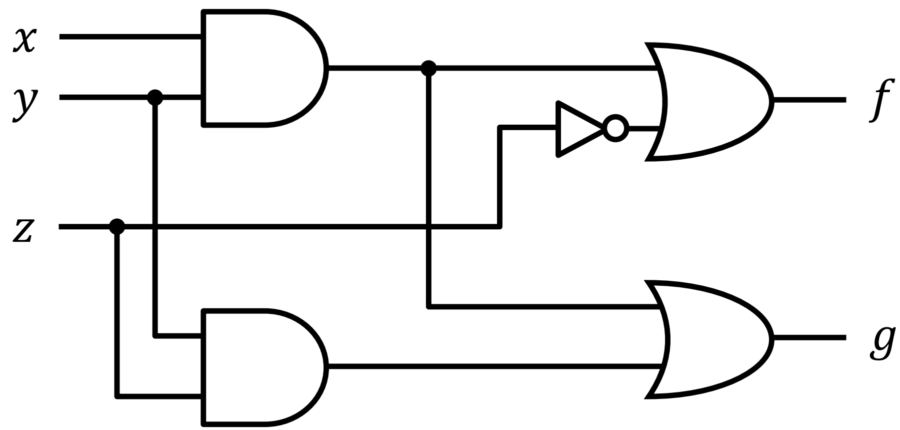
  n inputs: $x_1, x_2, \dots, x_n$

  m outputs: $f_1, f_2, \dots, f_m$

- Each output is a function of the input variables
- Each output is determined from present combination of inputs
- Combination circuit performs operation specified by logic gates



Example of a Simple Combinational Circuit



- The above circuit has:
  - Three inputs: x, y, and z
  - Two outputs: f and g
- What are the logic expressions of f and g ?
- **Answer:** $f = xy + z'$

  $g = xy + yz$

From Truth Tables to Gate Implementation

- Given the truth table of a Boolean function f, how do we implement the truth table using logic gates?

Truth Table

x y z	f
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	0
1 1 1	1

What is the logic expression of f?

What is the gate implementation of f?

To answer these questions, we need to define Minterms and Maxterms

Minterms and Maxterms

- **Minterms** are AND terms with every variable present in either true or complement form
- **Maxterms** are OR terms with every variable present in either true or complement form

Minterms and Maxterms for 2 variables x and y

x	y	index	Minterm	Maxterm
0	0	0	$m_0 = x'y'$	$M_0 = x + y$
0	1	1	$m_1 = x'y$	$M_1 = x + y'$
1	0	2	$m_2 = xy'$	$M_2 = x' + y$
1	1	3	$m_3 = xy$	$M_3 = x' + y'$

- For *n* variables, there are 2^n Minterms and Maxterms

Minterms and Maxterms for 3 Variables

x	y	z	index	Minterm	Maxterm
0	0	0	0	$m_0 = x'y'z'$	$M_0 = x + y + z$
0	0	1	1	$m_1 = x'y'z$	$M_1 = x + y + z'$
0	1	0	2	$m_2 = x'yz'$	$M_2 = x + y' + z$
0	1	1	3	$m_3 = x'yz$	$M_3 = x + y' + z'$
1	0	0	4	$m_4 = xy'z'$	$M_4 = x' + y + z$
1	0	1	5	$m_5 = xy'z$	$M_5 = x' + y + z'$
1	1	0	6	$m_6 = xyz'$	$M_6 = x' + y' + z$
1	1	1	7	$m_7 = xyz$	$M_7 = x' + y' + z'$

Maxterm M_i is the **complement** of Minterm m_i

$M_i = m_i'$ and $m_i = M_i'$

Purpose of the Index

- Minterms and Maxterms are designated with an 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 or complemented form
- For Minterms:
  - '**1**' means the variable is **Not Complemented**
  - '**0**' means the variable is **Complemented**
- For Maxterms:
  - '**0**' means the variable is **Not Complemented**
  - '**1**' means the variable is **Complemented**

Sum-Of-Minterms (SOM) Canonical Form

Truth Table

x y z	f	Minterm
0 0 0	0	
0 0 1	0	
0 1 0	1	$m_2 = x'yz'$
0 1 1	1	$m_3 = x'yz$
1 0 0	0	
1 0 1	1	$m_5 = xy'z$
1 1 0	0	
1 1 1	1	$m_7 = xyz$

Sum of Minterm entries that evaluate to '**1**'

Focus on the '**1**' entries

$$f = m_2 + m_3 + m_5 + m_7$$

$$f = \sum(2, 3, 5, 7)$$

$$f = x'yz' + x'yz + xy'z + xyz$$

Examples of Sum-Of-Minterms

- $f(a,b,c,d) = \sum(2,3,6,10,11)$
- $f(a,b,c,d) = m_2 + m_3 + m_6 + m_{10} + m_{11}$
- $f(a,b,c,d) = a'b'cd' + a'b'cd + a'bcd' + ab'cd' + ab'cd$
- $g(a,b,c,d) = \sum(0,1,12,15)$
- $g(a,b,c,d) = m_0 + m_1 + m_{12} + m_{15}$
- $g(a,b,c,d) = a'b'c'd' + a'b'c'd + abc'd' + abcd$

Product-Of-Maxterms (POM) Canonical Form

Truth Table

x y z	f	Maxterm
0 0 0	0	$M_0 = x + y + z$
0 0 1	0	$M_1 = x + y + z'$
0 1 0	1	
0 1 1	1	
1 0 0	0	$M_4 = x' + y + z$
1 0 1	1	
1 1 0	0	$M_6 = x' + y' + z$
1 1 1	1	

Product of Maxterm entries that evaluate to '**0**'

Focus on the '**0**' entries

$$f = M_0 \cdot M_1 \cdot M_4 \cdot M_6$$

$$f = \prod (0, 1, 4, 6)$$

$$f = (x + y + z)(x + y + z')(x' + y + z)(x' + y' + z)$$

Examples of Product-Of-Maxterms

- $f(a,b,c,d) = \prod(1,3,11)$
- $f(a,b,c,d) = M_1 \cdot M_3 \cdot M_{11}$
- $f(a,b,c,d) = (a+b+c+d')(a+b+c'+d')\,(a'+b+c'+d')$
- $g(a,b,c,d) = \prod(0,5,13)$
- $g(a,b,c,d) = M_0 \cdot M_5 \cdot M_{13}$
- $f(a,b,c,d) = (a+b+c+d)(a+b'+c+d')\,(a'+b'+c+d')$

Conversions between Canonical Forms

- The same Boolean function f can be expressed in two ways:
  - Sum-of-Minterms $f = m_0 + m_2 + m_3 + m_5 + m_7 = \sum(0,2,3,5,7)$
  - Product-of-Maxterms $f = M_1 \cdot M_4 \cdot M_6 = \prod(1,4,6)$

Truth Table

x y z	f	Minterms	Maxterms
0 0 0	1	$m_0 = x'y'z'$	
0 0 1	0		$M_1 = x + y + z'$
0 1 0	1	$m_2 = x'yz'$	
0 1 1	1	$m_3 = x'yz$	
1 0 0	0		$M_4 = x' + y + z$
1 0 1	1	$m_5 = xy'z$	
1 1 0	0		$M_6 = x' + y' + z$
1 1 1	1	$m_7 = xyz$	

To convert from one canonical form to another, interchange the symbols $\sum$ and $\prod$ and list those numbers missing from the original form.

Function Complement

Truth Table

x y z	f	f'
0 0 0	1	0
0 0 1	0	1
0 1 0	1	0
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	0	1
1 1 1	1	0

Given a Boolean function f

$$f(x, y, z) = \sum(0, 2, 3, 5, 7) = \prod(1, 4, 6)$$

Then, the complement f' of function f

$$f'(x, y, z) = \prod(0, 2, 3, 5, 7) = \sum(1, 4, 6)$$

The complement of a function expressed by a Sum of Minterms is the Product of Maxterms with the same indices. Interchange the symbols $\sum$ and $\prod$, but keep the same list of indices.

Algebraic Conversion to Sum-of-Minterms

- Expand all terms first to explicitly list all minterms
- AND any term missing a variable v with $(v + \overline{v})$
- Example 1: $f = x + \overline{x}\,\overline{y}$ (2 variables)

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

  $f = x\,y + x\,\overline{y} + \overline{x}\,\overline{y}$

  $\mathrm{f} = \mathrm{m}_3 + \mathrm{m}_2 + \mathrm{m}_0 = \sum(0, 2, 3)$
- Example 2: $g = a + \overline{b}\,c$ (3 variables)

  $g = a\,(b + \overline{b})(c + \overline{c}) + (a + \overline{a})\,\overline{b}\,c$

  $g = a\,b\,c + a\,b\,\overline{c} + a\,\overline{b}\,c + a\,\overline{b}\,\overline{c} + a\,\overline{b}\,c + \overline{a}\,\overline{b}\,c$

  $g = \overline{a}\,\overline{b}\,c + a\,\overline{b}\,\overline{c} + a\,\overline{b}\,c + a\,b\,\overline{c} + a\,b\,c$

  $g = \mathrm{m}_1 + \mathrm{m}_4 + \mathrm{m}_5 + \mathrm{m}_6 + \mathrm{m}_7 = \sum(1, 4, 5, 6, 7)$

Algebraic Conversion to Product-of-Maxterms

- Expand all terms first to explicitly list all maxterms
- OR any term missing a variable v with $v \cdot \overline{v}$
- Example 1: $f = x + \overline{x}\,\overline{y}$ (2 variables)

  Apply 2nd distributive law:

  $f = (x + \overline{x})\,(x + \overline{y}) = 1 \cdot (x + \overline{y}) = (x + \overline{y}) = M_1$
- Example 2: $g = a\,\overline{c} + b\,c + \overline{a}\,\overline{b}$ (3 variables)

  $g = (a\,\overline{c} + b\,c + \overline{a})\,(a\,\overline{c} + b\,c + \overline{b})$ (distributive)

  $g = (\overline{c} + b\,c + \overline{a})\,(a\,\overline{c} + c + \overline{b})$ $(x + \overline{x}\,y = x + y)$

  $g = (\overline{c} + b + \overline{a})\,(a + c + \overline{b})$ $(x + \overline{x}\,y = x + y)$

  $g = (\overline{a} + b + \overline{c})\,(a + \overline{b} + c) = M_5 \,.\, M_2 = \prod(2, 5)$

Summary of Minterms and Maxterms

- There are 2^n Minterms and Maxterms for Boolean functions with n variables, indexed from 0 to $2^n - 1$
- Minterms correspond to the **1-entries** of the function
- Maxterms correspond to the **0-entries** of the function
- Any Boolean function can be expressed as a Sum-of-Minterms and as a Product-of-Maxterms
- For a Boolean function, given the list of Minterm indices one can determine the list of Maxterms indices (and vice versa)
- The complement of a Sum-of-Minterms is a Product-of-Maxterms with the same indices (and vice versa)

Sum-of-Products and Products-of-Sums

- Canonical forms contain a larger number of literals
  - Because the Minterms (and Maxterms) must contain, by definition, all the variables either complemented or not
- Another way to express Boolean functions is in **standard** form
- Two standard forms: **Sum-of-Products** and **Product-of -Sums**
- Sum of Products (**SOP**)
  - Boolean expression is the **ORing** (sum) of **AND terms** (products)
  - Examples: $f_1 = xy' + xz$ $\qquad f_2 = y + xy'z$
- Products of Sums (**POS**)
  - Boolean expression is the **ANDing** (product) of **OR terms** (sums)
  - Examples: $f_3 = (x + z)(x' + y')$ $\qquad f_4 = x(x' + y' + z)$

Standard Forms

- **Sum of Products (SOP)**

$$F = \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + A\overline{B}C + ABC$$

$$A\overline{B}(\overline{C} + C)$$
$$= A\overline{B}(1)$$
$$= A\overline{B}$$

$$AC(\overline{B} + B)$$
$$= AC$$

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

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

$$\boxed{F = \overline{B}C + A\overline{B} + AC}$$

Standard Forms

❖ **Product of Sums (POS)**

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

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

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

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

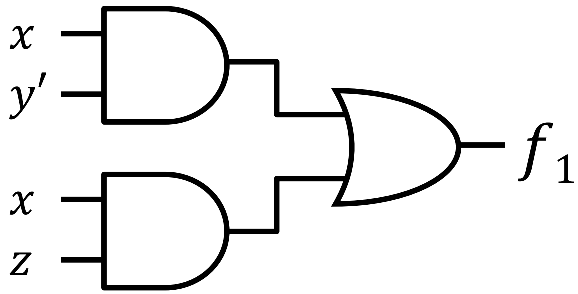
$$\bar{F} = \bar{A}\bar{C}\cancel{(\bar{B}+B)} + \bar{A}B\cancel{(\bar{C}+C)} + B\bar{C}\cancel{(\bar{A}+A)}$$

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

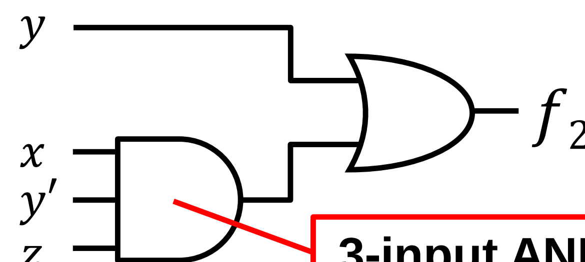
$$F = (A+C)(A+\bar{B})(\bar{B}+C)$$

Two-Level Gate Implementation

$$f_1 = xy' + xz$$



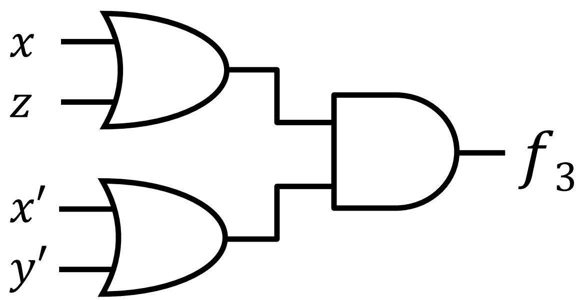
$$f_2 = y + xy'z$$



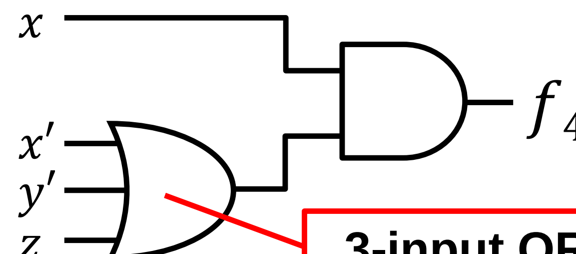
3-input AND gate

AND-OR implementations

$$f_3 = (x + z)(x' + y')$$



$$f_4 = x(x' + y' + z)$$



3-input OR gate

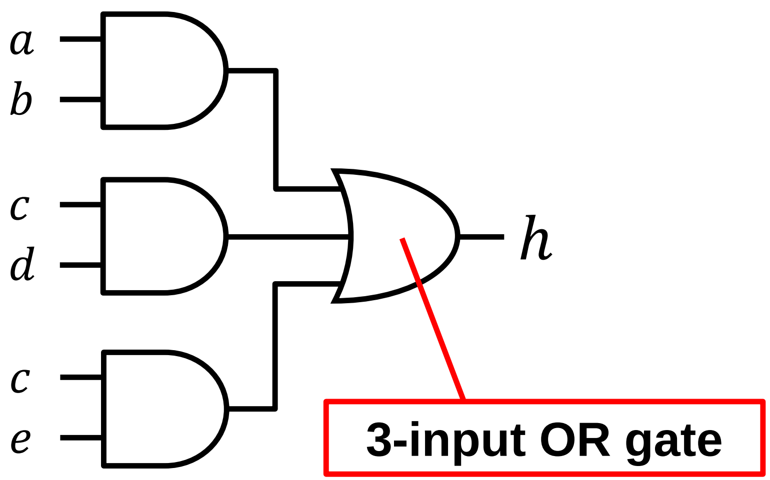
OR-AND implementations

Two-Level vs. Three-Level Implementation

- $h = ab + cd + ce$ (6 literals) is a sum-of-products
- h may also be written as: $h = ab + c(d + e)$ (5 literals)
- However, $h = ab + c(d + e)$ is a non-standard form
  - $h = ab + c(d + e)$ is not a sum-of-products nor a product-of-sums

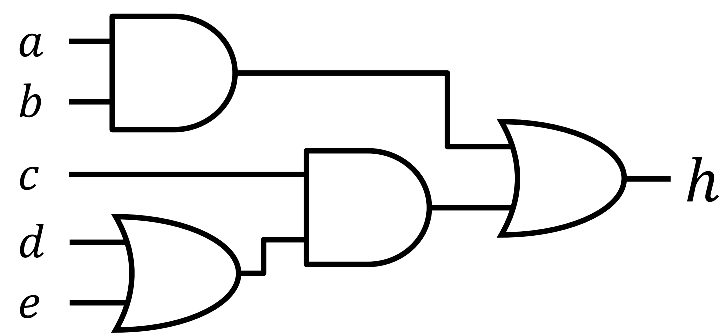
2-level implementation

$$h = ab + cd + ce$$



3-level implementation

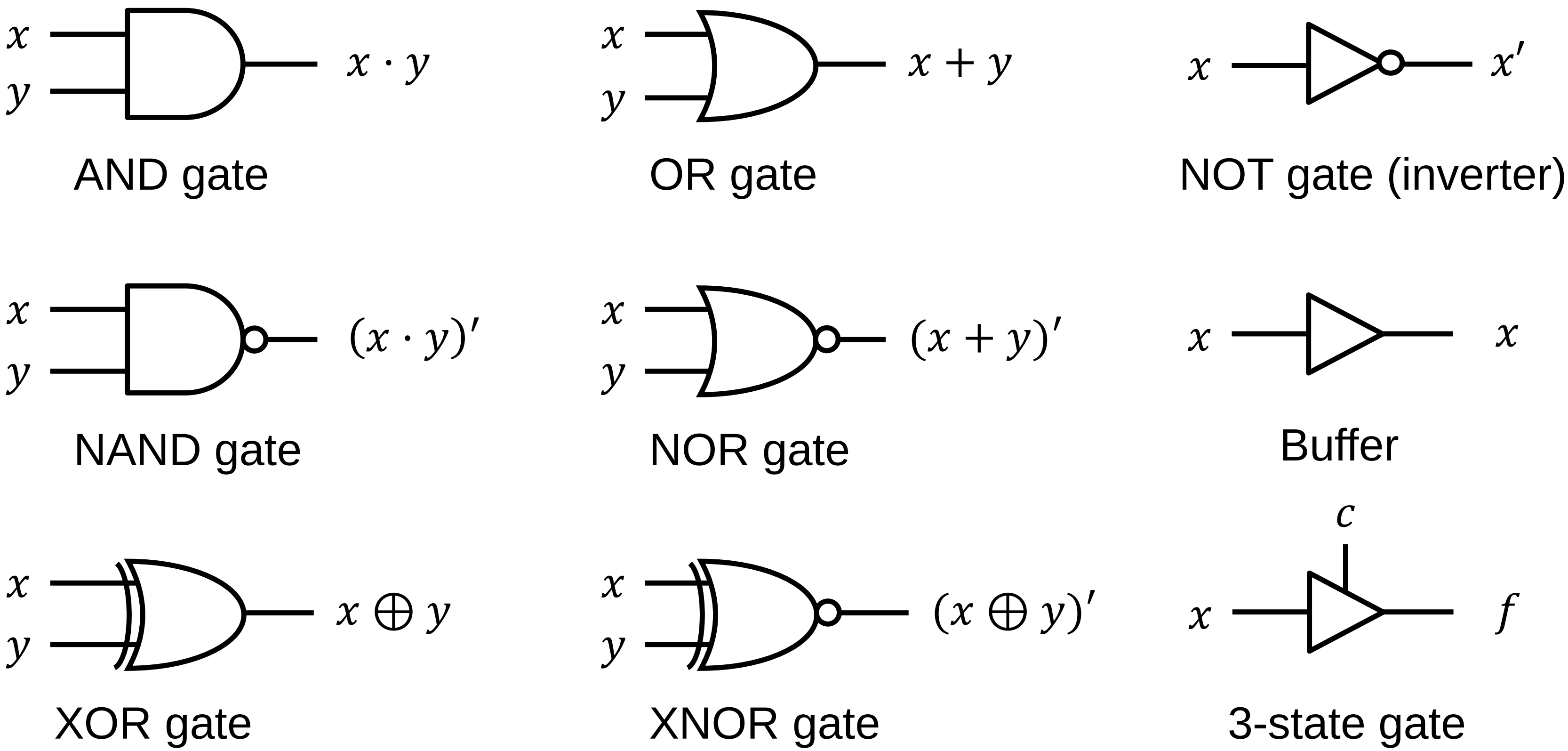
$$h = ab + c(d + e)$$



Additional Logic Gates and Symbols

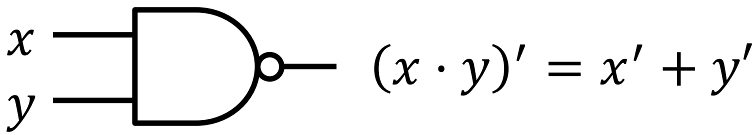
- Why?
  - Low cost implementation
  - Useful in implementing Boolean functions
- Factors to be weighed in considering the construction of other types of logic gates are
  - The feasibility and economy of producing the gate with physical components,
  - The possibility of extending the gate to more than two inputs,
  - The basic properties of the binary operator, such as commutativity and associativity,
  - The ability of the gate to implement Boolean functions alone or in conjunction with other gates.

Additional Logic Gates and Symbols

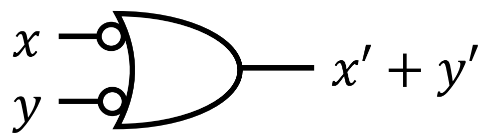


NAND Gate

- The NAND gate has the following symbol and truth table
- NAND represents **N**OT **AND**
- The small bubble circle represents the invert function



NAND gate



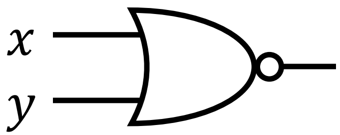
Another symbol for NAND

x	y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

- NAND gate is implemented efficiently in CMOS technology
  - In terms of chip area and speed

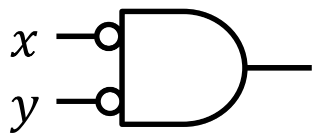
NOR Gate

- The NOR gate has the following symbol and truth table
- NOR represents **N**OT **OR**
- The small bubble circle represents the invert function



$(x+y)' = x' \cdot y'$

NOR gate



$x' \cdot y'$

Another symbol for NOR

x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

- NOR gate is implemented efficiently in CMOS technology
  - In terms of chip area and speed

Non-Associative NAND / NOR Operations

- Unlike AND, NAND operation is NOT associative

  (x NAND y) NAND z ≠ x NAND (y NAND z)

  (x NAND y) NAND z = $((xy)'z)' = ((x'+y')z)' = xy + z'$

  x NAND (y NAND z) = $(x(yz)')' = (x(y'+z'))' = x' + yz$

- Unlike OR, NOR operation is NOT associative

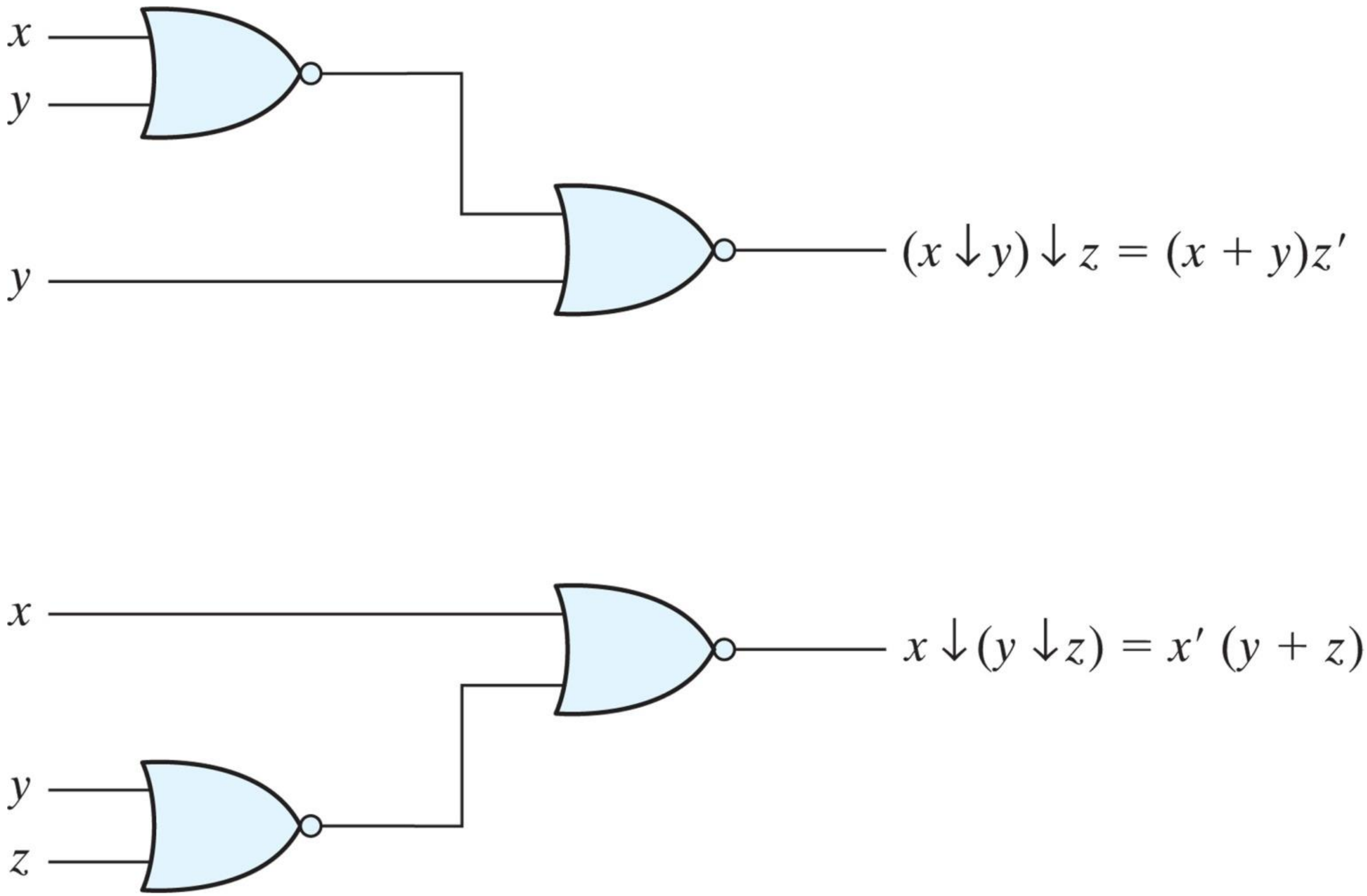
  (x NOR y) NOR z ≠ x NOR (y NOR z)

  (x NOR y) NOR z = $\left((x+y)'+z\right)' = \left((x'y')+z\right)' = (x+y)z'$

  x NOR (y NOR z) = $(x+(y+z)')' = \left(x+(y'z')\right)' = x'(y+z)$

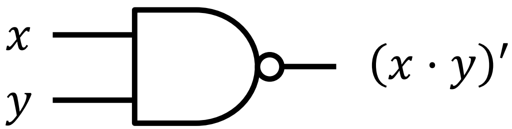
Extension to multiple inputs

Demonstrating the nonassociativity of the NOR operator: $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

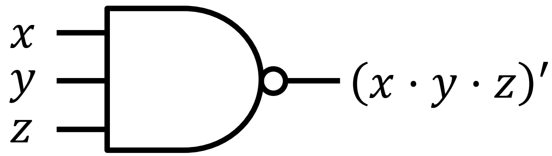


Multiple-Input NAND / NOR Gates

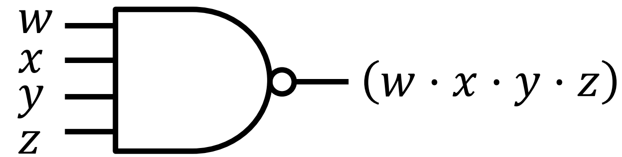
NAND/NOR gates can have multiple inputs, similar to AND/OR gates



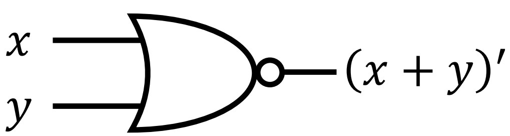
2-input NAND gate



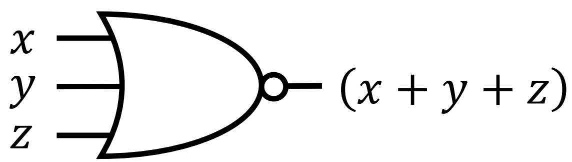
3-input NAND gate



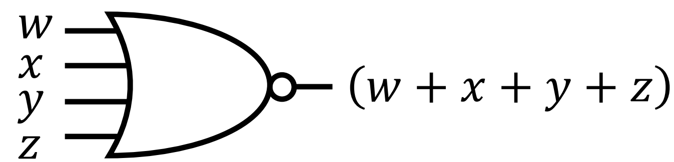
4-input NAND gate



2-input NOR gate



3-input NOR gate

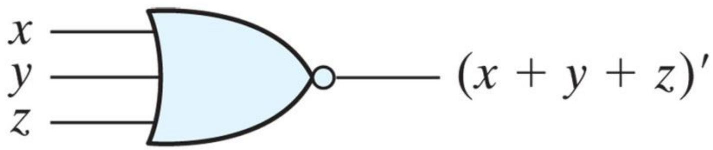


4-input NOR gate

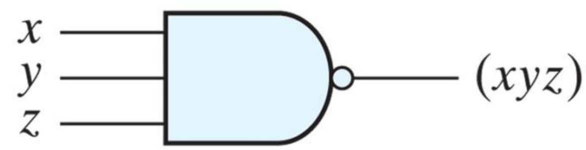
Note: a 3-input NAND is a single gate, NOT a combination of two 2-input gates.
The same can be said about other multiple-input NAND/NOR gates.

Extension to multiple inputs

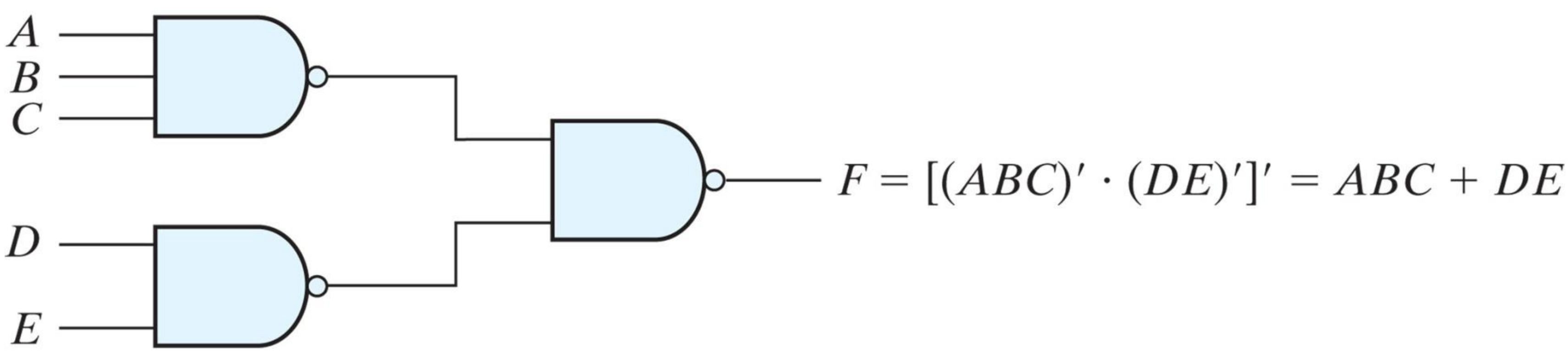
Multiple-input and cascaded NOR and NAND gates



(a) 3-input NOR gate



(b) 3-input NAND gate



(c) Cascaded NAND gates

Exclusive OR / Exclusive NOR

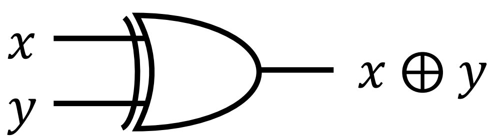
❖ Exclusive OR (XOR) is an important Boolean operation used extensively in logic circuits

❖ Exclusive NOR (XNOR) is the complement of XOR

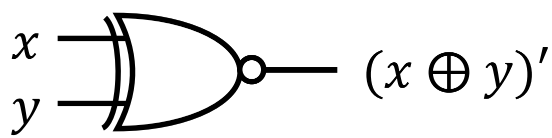
x y	XOR
0 0	0
0 1	1
1 0	1
1 1	0

x y	XNOR
0 0	1
0 1	0
1 0	0
1 1	1

XNOR is also known as **equivalence**



XOR gate



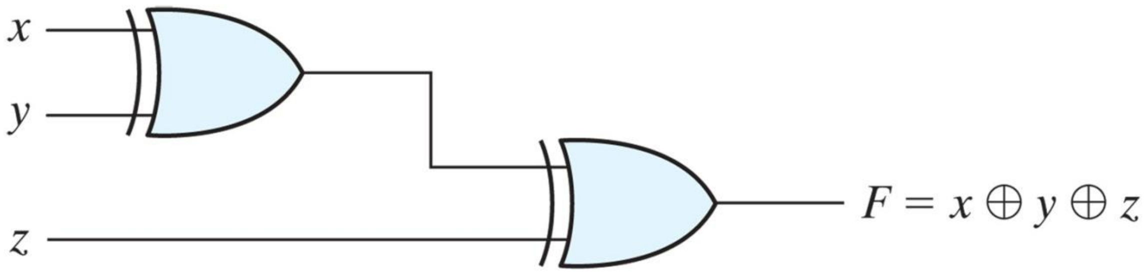
XNOR gate

XOR / XNOR Functions

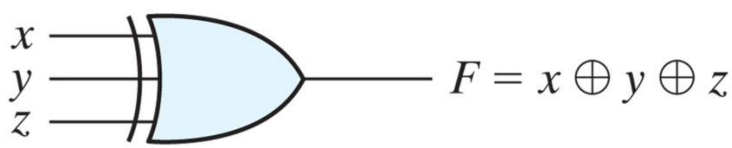
- The XOR function is: $x \oplus y = xy' + x'y$
- The XNOR function is: $(x \oplus y)' = xy + x'y'$
- XOR and XNOR gates are complex
  - Can be implemented as a true gate, or by
  - Interconnecting other gate types
- XOR and XNOR gates do not exist for more than two inputs
  - For 3 inputs, use two XOR gates
  - The cost of a 3-input XOR gate is greater than the cost of two XOR gates
- Uses for XOR and XNOR gates include:
  - Adders, subtractors, multipliers, counters, incrementers, decrementers
  - Parity generators and checkers

Extension to multiple inputs

Three-input exclusive-OR gate



(a) Using 2-input gates



(b) 3-input gate

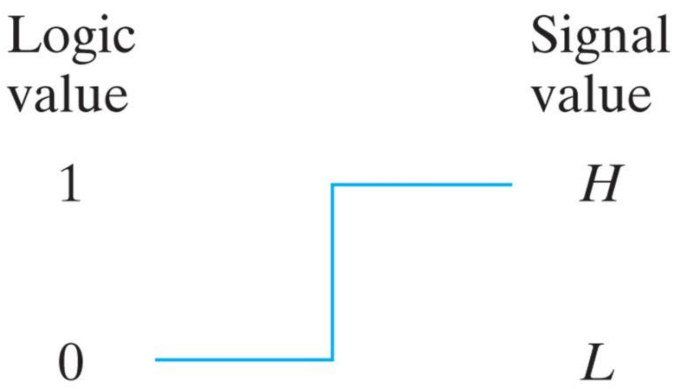
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(c) Truth table

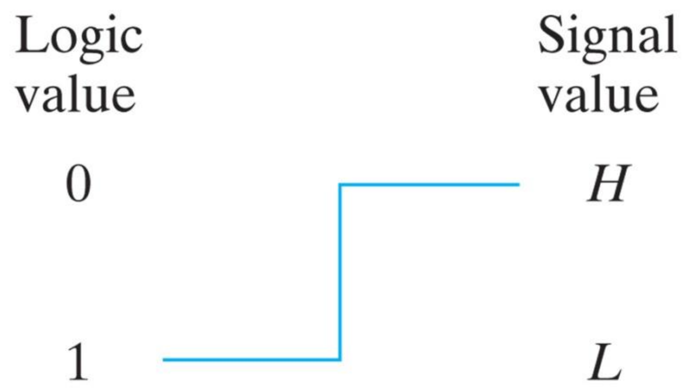
Positive and Negative Logic

- Choosing the high-level *H* to represent logic 1 defines a positive logic system
- Choosing the low-level *L* to represent logic 1 defines a negative logic system.
- It is up to the user to decide on a positive or negative logic polarity.

Signal assignment and logic polarity



(a) Positive logic



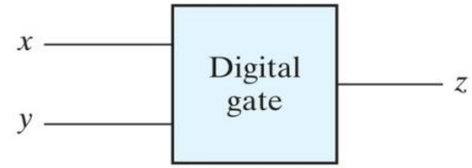
(b) Negative logic

Positive and Negative Logic

- The conversion from positive logic to negative logic and vice versa is essentially an operation that changes 1's to 0's and 0's to 1's in both the inputs and the output of a gate.
- Since this operation produces the dual of a function, the change of all terminals from one polarity to the other results in taking the dual of the function.

x	y	z
L	L	L
L	H	L
H	L	L
H	H	H

(a) Truth table with H and L



(b) Gate block diagram

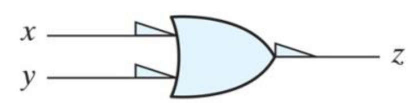
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

(c) Truth table for positive logic

(d) Positive logic AND gate

x	y	z
1	1	1
1	0	1
0	1	1
0	0	0

(e) Truth table for negative logic



(f) Negative logic OR gate