DIGITAL LOGIC(H)

Lecture 2
Boolean Algebra

2024 Fall

Today's Agenda

- Recap
- Context
  - Boolean Algebra (布尔代数)
  - Axioms (公理) and Theorems(定理)
  - Boolean Functions (布尔方程)
  - Canonical (范式) and Standard form(标准式)
- Reading: Textbook, Chapter 2

Recap

- Chapter1
  - Number Systems
    - Decimal
    - Binary
    - Octal
    - Hexadecimal
  - Number based Conversion
    - Base-r to Decimal: $D = \sum_{i=-m}^{n-1} d_i r^i$
    - Decimal to Base-r: division for integer part, multiplication for fraction
    - Base-r to Base-r: group bits from radix point
  - Complements
    - r's complement
    - r−1's complement
    - Binary
      - 2's complement
      - 1's complement
    - subtraction
    - signed binary numbers
  - Representation of data: code
    - BCD
    - Gray code
    - ASCII
    - Parity

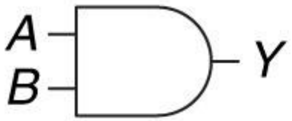
Outline

- **Axioms and Theorems of Boolean Algebra**
- Simplify Boolean Functions
- Canonical and Standard form
- Other Logic Operations

Binary Logic

- Deal with Variables like A, B… taking two values:
  - ‘0’, ‘1’; ‘L’, ‘H’; ‘T’, ‘F’

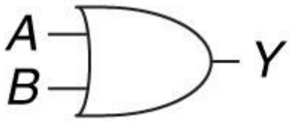
AND



$Y = AB$

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

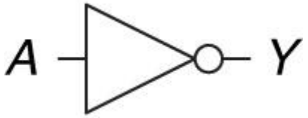
OR



$Y = A + B$

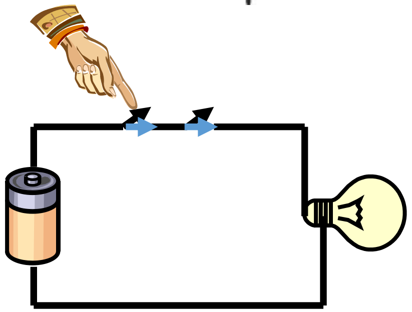
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

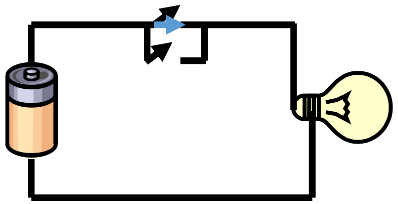
NOT



$Y = A'$

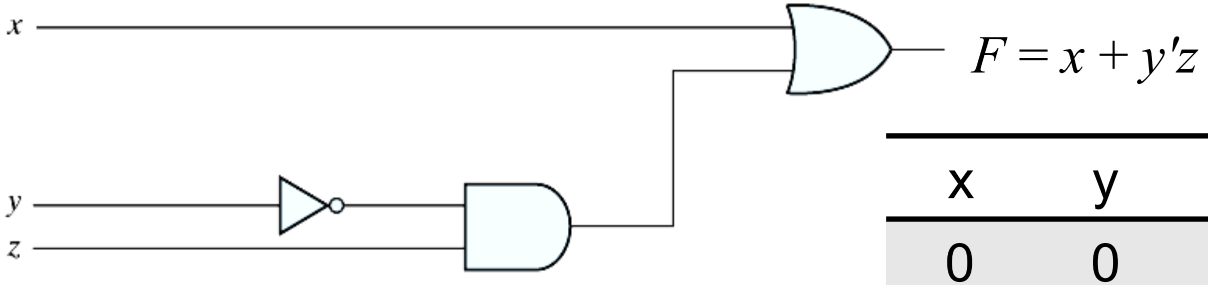
A	Y
0	1
1	0





Boolean Equation and Truth Table

- Boolean Equation: F = x + y'z
- Logic diagram:



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

- if x = y = 0, z = 1
  - F = 0 + 1•1 = 1
- Truth table (真值表)
  - The truth table of F has 2^n entries (n = num of inputs)

Boolean Algebra

- Boolean algebra(逻辑代数), a deductive mathematical system developed by George Boole in 1854, deals with the rules by which logical operations are carried out.
- Boolean algebra is an algebraic structure defined by
  - a set of elements S: binary variables;
  - a set of binary operators: AND(•), OR(+) and NOT(’);
  - and a number of Axioms/theorems.

Boolean Axioms and Theorems of One Variable

- Axioms and theorems to simplify Boolean equations
- Duality (对偶性) in Axioms and theorems:
  - Replace • with +, 0 with 1, the = relation remains

	Theorem	Dual	Name
1	$x + 0 = x$	$x \cdot 1 = x$	Identity
2	$x + 1 = 1$	$x \cdot 0 = 0$	Null Element
3	$x + x = x$	$x \cdot x = x$	Idempotency
4	$(x')' = x$		Involution
5	$x + x' = 1$	$x \cdot x' = 0$	Complements

- Operator precedence
  - Parentheses > NOT > AND > OR

Boolean Axioms and Theorems of Several Variables

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Dual: Replace • with +, 0 with 1, the = relation remains

	Theorem	Dual	Name
6	xy = yx	x + y = y + x	Commutativity
7	(xy)z = x(yz)	(x + y) + z = x + (y + z)	Associativity
8	x(y + z) = xy + xz	x + yz = (x + y)(x + z)	Distributivity
9	x + xy = x	x(x + y) = x	Absorption
10	xy + xy’ = x	(x + y)(x + y’) = x	Combining
11	(x+y’)y = xy	xy’ + y = x + y	Simplification
12	xy + x’z + yz = xy + x’z	(x + y)(x’ + z)(y + z) = (x + y)(x’ + z)	Consensus
13	(x + y)’ = x’y’	(xy)’ = x’ + y’	DeMorgan’s law

Note: 8’s Dual differs from traditional algebra: OR (+) distributes over AND (•)

Proofs (1)

Algebraic method

- **Absorption**
  - $x + xy = x$
  - pf: $x + xy = x \cdot 1 + x \cdot y = x(1+y) = x$
- **Combining**
  - $(x + y)(x + y') = x$
  - pf: $(x + y)(x + y') = x + yy' = x + 0 = x$
- **Simplification**
  - $xy' + y = x + y$
  - pf: $xy' + y = xy' + (x+x')y = xy' + xy + x'y$

    $= (xy' + xy) + (xy + x'y) = x(y'+y) + y(x+x') = x + y$
- **Consensu**
  - $xy + x'z + yz = xy + x'z$
  - pf: $xy + x'z + yz = xy + x'z + (x+x')yz$

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

    $= (xy + xyz) + (x'z + x'zy) = xy + x'z$

Proofs (2)

- **DeMorgan's Law**
  - $(x + y)' = x'y'$ $(xy)' = x' + y'$

Truth table method

pf:

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

- **Associativity**
  - $(xy)z = x(yz)$
  - $(x + y) + z = x + (y + z)$

x	y	z	(xy)z	x(yz)	(x+y)+z	x+(y+z)
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	0	1	1
0	1	1	0	0	1	1
1	0	0	0	0	1	1
1	0	1	0	0	1	1
1	1	0	0	0	1	1
1	1	1	1	1	1	1

Outline

- Axioms and Theorems of Boolean Algebra
- **Simplify Boolean Functions**
- 权威性的
- Canonical and Standard form
- Other Logic Operations

Boolean Functions

- A Boolean function from an algebraic expression can be realized to a logic diagram composed of logic gates.
  - Binary variables
  - operators OR, AND, NOT
  - Parentheses
- Terminology:
  - Literal: A variable or its complement
  - Product term: literals connected by •
  - Sum term: literals connected by +
- Example: 内部是3个积项，然后相加
  - A′B′C + A′BC +AB′ has 8 literals, 3 product term
  - (A+B’+C)(A’+C) has 5 literals, 2 sum term

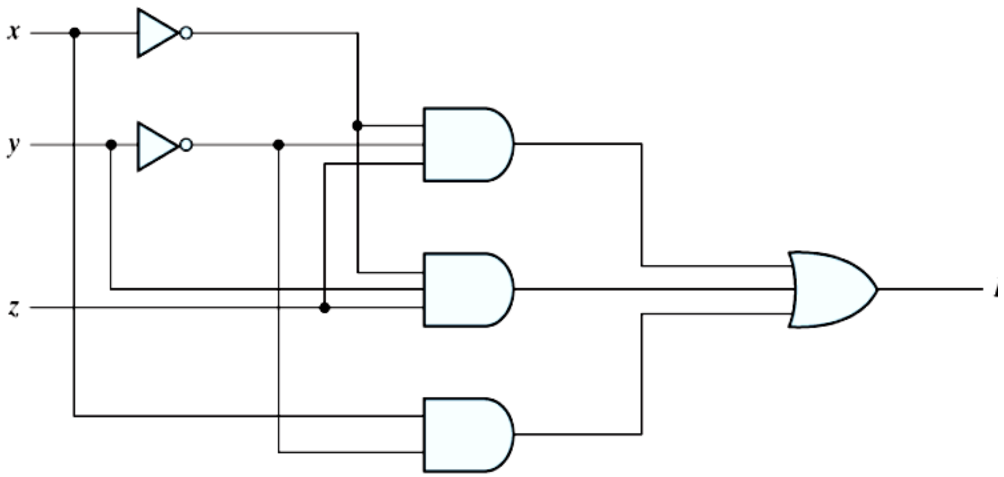
Boolean Functions

- Each Boolean function has
  - only one representation in truth table
  - but a variety of ways in algebraic form/gate implementation.
- Examples
  - $F_1 = x' y' z + x' y z + x y'$
  - $F_2 = x y' + x' z$
  - $F_1 = F_2$
    - Same truth table
    - Different algebraic expression

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	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

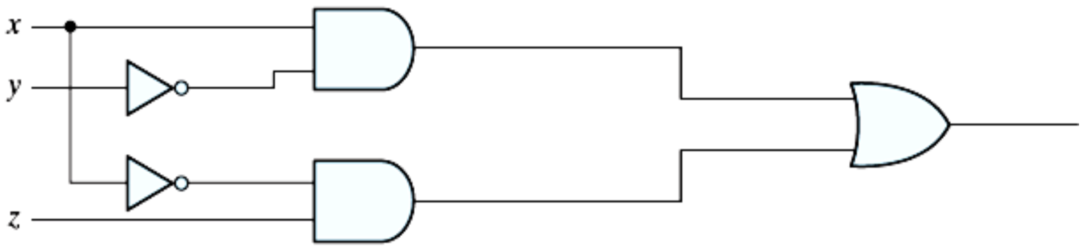
Gate Implementation

- $F_1 = x'y'z + x'yz + xy'$
  - 8 literals
  - 3 terms (implementation with a gate)



- $F_2 = x'z + xy'$
  - 4 literals
  - 2 terms
  - Simpler circuit, more economical

F_1	$= x'y'z + x'yz + xy'$	
	$= x'z(y' + y) + xy'$	Distributivity
	$= x'z + xy' = F_2$	Complements



Algebraic Simplification

- Minimize the number of literals and terms for a simpler circuits (less expensive)
- Algebraic simplification can minimize literals and terms. However, no specific rules to guarantee the optimal results
- Usually not possible by hand for complex functions, use computer minimization program
- More advanced techniques in the next lectures (K-Map)
- Useful rules
  - Distributivity
  - Idempotency
  - Complements
  - DeMorgan's
  - etc

Example

- Examples:

$$\begin{aligned} F &= A'BC + A' \\ &= A'(BC + 1) && \text{Distributivity} \\ &= A' && \text{Null Element} \end{aligned}$$

$$\begin{aligned} F &= XYZ + XY'Z + XYZ' \\ &= XYZ + XY'Z + XYZ + XYZ' && \text{Idempotency} \\ &= XZ(Y + Y') + XY(Z + Z') && \text{Distributivity} \\ &= XZ + XY && \text{Complements} \\ &= X(Y + Z) && \text{Distributivity} \end{aligned}$$

Exercise:

$$\begin{aligned} F &= A'B'C + A'BC + AB' \\ &= ?\ A'B'C + A'BC + AB'C + AB'C' \\ &= ?\ A'C + AB' \end{aligned}$$

Boolean Function complement

- The complement of any function F is F', which can be obtained by DeMorgan's Theorem
  - Take the dual of expression, and then complement each literal in F
- Example: $F_3 = x'y'z+x'yz+xy'$
  - Step1, Dual: Replace • with +, 0 with 1

$$x'y'z + x'yz + xy' \xrightarrow{\text{Dual}} (x'+y'+z)(x'+y+z)(x+y')$$

  - Step2, complement each literal in F

$$F_3' = (x'y'z + x'yz + xy')'$$
$$= (x+y+z')(x+y'+z')(x'+y) \qquad \text{DeMorgan}$$

二元性

Pay attention! The dual is not duality! 原表达式与其对偶不一定等价

$x'y'z + x'yz + xy' \neq (x'+y'+z)(x'+y+z)(x+y')$

Outline

- Axioms and Theorems of Boolean Algebra
- Simplify Boolean Functions
- **Canonical and Standard form**
- Other Logic Operations

Minterms and Maxterms

- Minterms and Maxterms
- A **minterm**(最小项): an AND term consists of all literals in their normal form or in their complement form.
  - For example, two binary variables x and y,
    - x'y', x'y, xy', xy (m_0~m_3)
  - n variables can be combined to form 2^n minterms
- A **maxterm**(最大项): an OR term
  - For example, two binary variables x and y,
    - x+y, x+y', x'+y, x'+y' (M_0~M_3)
  - 2^n maxterms
- Each maxterm is the complement of its corresponding minterm and vice versa. ($M_i = m_i'$)

互补

Minterms and Maxterms

- Canonical forms
  - sum-of-minterms (som)
  - product-of-maxterms (pom)

Example: Minterms and maxterms for three binary variables

			Minterms		Maxterms	
x	**y**	**z**	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Canonical Forms

- A Boolean function F = xy+x'z can be expressed by
- a truth table
- either of the 2 canonical forms
  - sum-of-minterms
    - F = x'y'z + x'yz + xyz' + xyz
      $= m_1 + m_3 + m_6 + m_7 = \sum(1,3,6,7)$
  - product-of-maxterms
    - F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')
      $= M_0 \cdot M_2 \cdot M_4 \cdot M_5 = \prod(0,2,4,5)$

Why $F = \sum(1,3,6,7) = \prod(0,2,4,5)$?

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

Conversion between som and pom

- To convert from one canonical(som: Sum of Minterms) to another(pom: Product of Maxterms), interchange ∑ and ∏, and list the numbers that were excluded from the original form

$F = \sum(1, 3, 6, 7) = m_1 + m_3 + m_6 + m_7$

$F' = \sum(0, 2, 4, 5) = m_0 + m_2 + m_4 + m_5$

$F = \sum(1, 3, 6, 7)$	(som)
$= (F')' = (m_0 + m_2 + m_4 + m_5)'$	(Convolution)
$= m'_0 m'_2 m'_4 m'_5$	(DeMorgan's)
$= M_0 M_2 M_4 M_5$	($M_i = m_i'$)
$= \prod(0, 2, 4, 5)$	(pom)

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

Represent a Function in Canonical Forms

- Example: Express $F = A+B'C$ as a sum of minterms.
  - by truth table
  - or by expanding the missing variables in each term, using 1=x+x', 0=xx'
- Hint: xy = xy(z+z') = xyz + xyz'

$$
\begin{aligned}
F &= A+B'C \\
&= A(B+B') + B'C \\
&= AB + AB' + B'C \\
&= \boxed{AB(C+C')} + AB'(C+C') + (A+A')B'C \\
&= ABC+ABC'+AB'C+AB'C'+A'B'C \\
&= m_1 + m_4 + m_5 + m_6 + m_7 \\
&= \sum(1, 4, 5, 6, 7)
\end{aligned}
$$

7 6 5 4 1

Truth Table for $F = A + B'C$

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

Represent a Function in Canonical Forms

- Example: Express $F = xy + x'z$ as a product of maxterms.
  - by truth table
  - First convert to product of sum form, then expand, using $1=x+x'$, $0=xx'$
- Hints: $x + y = (x + y + zz') = (x+y+z)(x+y+z')$

$x + yz = (x + y)(x + z)$ Distributivity

Tips: You can also use DeMorgan's Law (Involution first)

$$
\begin{aligned}
F &= xy + x'z \\
&= (xy + x')(xy + z) \\
&= (x+x')(y+x')(x+z)(y+z) \\
&= (x'+y)(x+z)(y+z) \\
&= \boxed{(x'+y+zz')}(x+z+yy')(y+z+xx') \\
&= (x'+y+z)(x'+y+z')(x+z+y)(x+z+y')\cancel{(y+z+x)}\cancel{(y+z+x')} \\
&= \underset{0}{(x+y+z)}\underset{2}{(x+y'+z)}\underset{4}{(x'+y+z)}\underset{5}{(x'+y+z')} \\
&= M_0 M_2 M_4 M_5 \\
&= \prod(0, 2, 4, 5)
\end{aligned}
$$

重复消去

Exercise

- How to convert f=x+y'z into canonical form?

$$f = x+y'z$$
$$= ?(y'+x)(z+x)$$
$$= (y'+x+zz')(z+x+yy')$$
$$= (x+y'+z)(x+y'+z')(x+y+z)$$
$$= \Pi(0,2,3)$$
$$= \Sigma(1,4,5,6,7)$$

Standard Forms

→ 不一定literal最少

- Canonical forms are very seldom the ones with the least number of literals.
- Standard forms: the terms that form the function may have fewer literals than the minterms.
  - Sum of products(sop): $F_1 = y' + xy + x'yz'$
  - Product of sums(pos): $F_2 = x(y'+z)(x'+y+z')$
  - $F_3 = A'B'CD+ABC'D'$
- Standard forms are not unique!

Outline

- Axioms and Theorems of Boolean Algebra
- Simplify Boolean Functions
- Canonical and Standard form
- **Other Logic Operations**

Other Logic Operations

- 2^n rows in the truth table of n binary variables.
- $\cancel{2^{2n}}$ functions for n binary variables. 2^{2^n}
- 16 functions of two binary variables.

Truth Tables for the 16 Functions of Two Binary Variables

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

- All the new symbols except for the exclusive-OR symbol are not in common use by digital designers.

Boolean Expressions

- When the three operators AND, OR, and NOT are applied on two variables A and B, they form 16 Boolean functions:

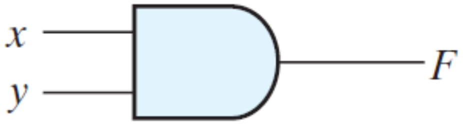
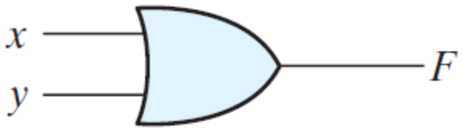
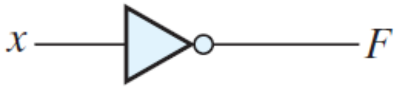
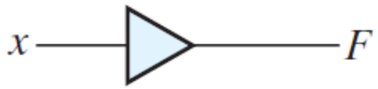
Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Digital Logic Gates

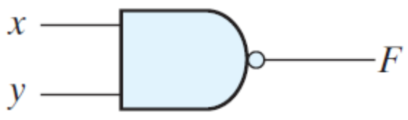
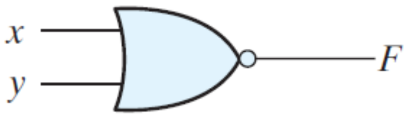
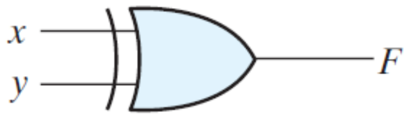
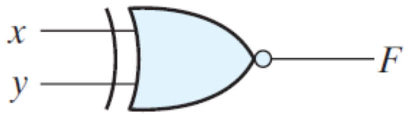
- Consider the 16 functions in previous Table
  - Two are equal to a constant (F_0 and F_{15}).
  - Four are repeated twice (F_4, F_5, F_{10} and F_{11}).

    不可交换/结合
  - Inhibition (F_2) and implication (F_{13}) are not commutative or associative.
  - The other eight are used as standard gates:
    - complement (F_{12})
    - transfer (F_3)
    - AND (F_1)
    - OR (F_7)
    - NAND (F_{14})
    - NOR (F_8)
    - XOR (F_6)
    - equivalence (XNOR) (F_9)
  - Complement: inverter.
  - Transfer: buffer (increasing drive strength).
  - Equivalence: XNOR.

Summary of Logic Gates

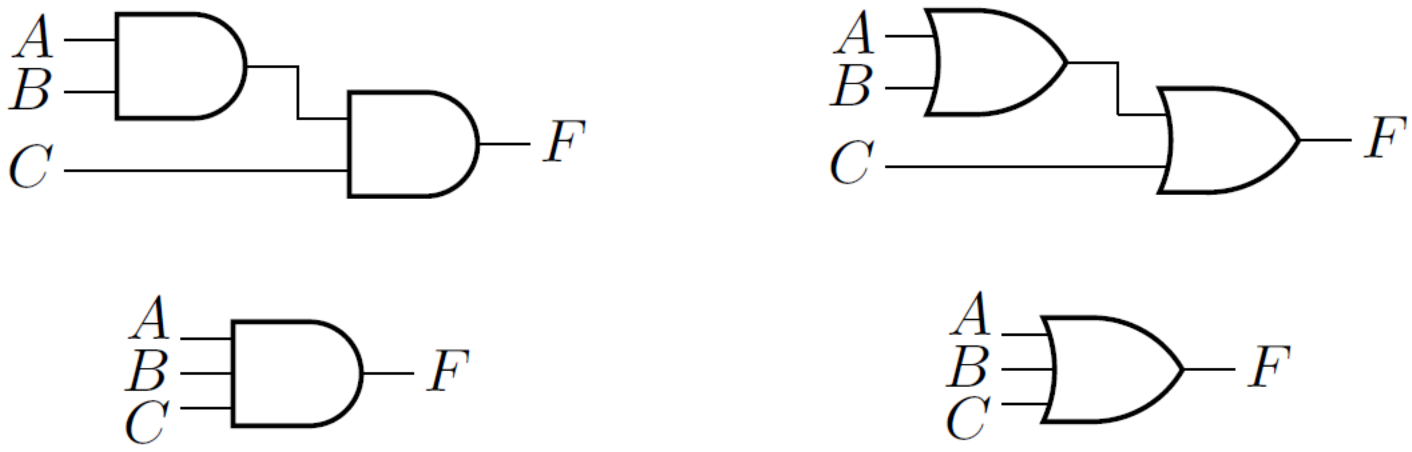
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0
Buffer		$F = x$	x | F 0 | 0 1 | 1

Summary of Logic Gates

Name	Graphic symbol	Algebraic function	Truth table
NAND		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x + y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

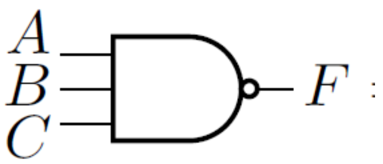
Multiple Inputs

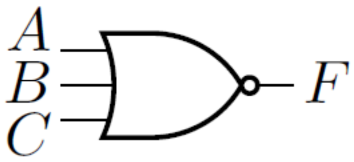
- Extension to multiple inputs
  - A gate can be extended to multiple inputs.
  - AND and OR are commutative and associative.
    - F = ABC = (AB)C
    - F = A + B + C = (A + B) + C



Multiple Inputs

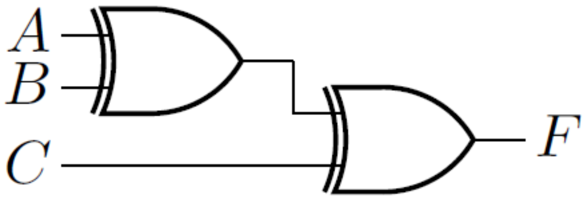
- NAND and NOR are commutative but not associative
  - ((AB)′C)′ ≠ (A(BC)′)′: does not follow associativity.
  - ((A + B)′ + C)′ ≠ (A + (B + C)′)′: does not follow associativity.

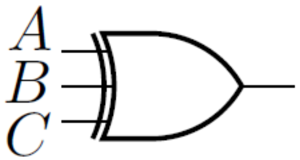




Multiple Inputs

- The XOR gates and equivalence gates both possess commutative and associative properties.
  - Gate output is low when even numbers of 1's are applied to the inputs, and when the number of 1's is odd the output is logic 0.
  - Multiple-input exclusive-OR and equivalence gates are uncommon in practice.





$F = A \oplus B \oplus C$