DIGITAL LOGIC

Chapter 2
Boolean Algebra

2025 Fall

baiyh@sustech.edu.cn

Today's Agenda

- Recap
- Boolean Algebra
  - 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)
  - Other Logic Operations
- Reading: Textbook, Chapter 2

Recap

- Number Systems
  - Binary, Octal, Hexadecimal
  - Conversion: Use weighted sums (to Decimal) and division/multiplication (from Decimal).
- Binary Arithmetic
  - Addition/Subtraction: Column-wise with carry/borrow.
  - Complements: Simplify subtraction.
    - 1's Comp: Bitwise NOT.
    - 2's Comp: (1's Comp + 1); preferred method.
- Signed Numbers
  - MSB is Sign Bit: 0 = Positive, 1 = Negative.
  - 2's Complement is Standard: original cod for positive value, 2's complement of absolute for negative value.
- Binary Codes
  - BCD: 4 bits per decimal digit. Adding extra 6 when >9 or carry while doing addition
  - Gray Code: Adjacent values differ by 1 bit.
  - Parity Bit: Single-bit error detection.
- Binary Logic
  - Gates (AND, OR, NOT): Building blocks.
  - Logic Levels: HIGH (1) / LOW (0) voltage ranges.

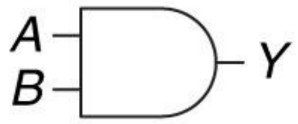
Outline

- **Boolean Algebra & Axioms and Theorems**
- 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’

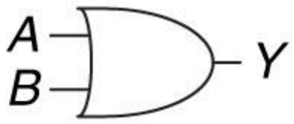
AND



$$Y = AB$$

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

OR

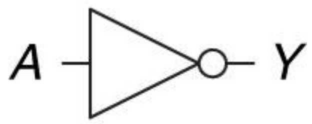


$$Y = A + B$$

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

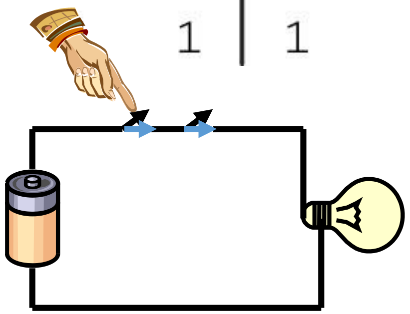
,

NOT

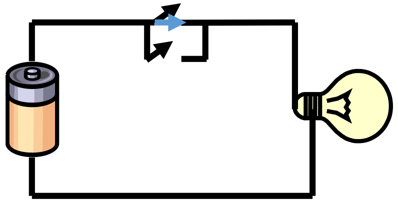


$$Y = \overline{A}$$

A	Y
0	1
1	0



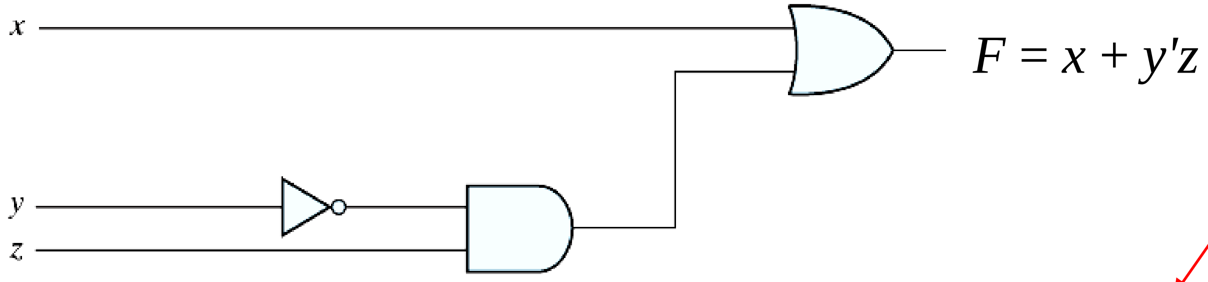
switches in series



switches in parallel

Boolean Equation and Truth Table

- Three different circuit representation method
- 1. Boolean Equation: F = x + y'z
- 2. Logic diagram:



- 3. Truth table (真值表)
  - The truth table of F has 2^n entries
  (n = num of inputs)

if x = y = 0, z = 1,
F = 0 + 1•1 = 1

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

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
- An important property in Axioms and theorems：Duality (对偶性)
  - Replace • with +, 0 with 1, the = relation is still valid

	Theorem	Dual form	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

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

	Theorem	Dual form	Name
6	$xy = yx$	$x + y = y + x$	Commutativity
7	$(xy)z = x(yz)$	$(x + y) + z = x + (y + z)$	Associativity
8	$x(y + z) = xy + xz$	$\boxed{x + yz = (x + y)(x + z)}$	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)

We can use algebraic method to prove

- **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: $xy' + y = xy' + (x+x')y = xy' + xy + x'y$
  $= (xy' + xy) + (xy + x'y) = x(y'+y) + y(x+x') = x + y$

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

We can also use truth table method to prove

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

- Boolean Algebra & Axioms and Theorems
- **Boolean Functions**
- 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:
  - $A'B'C + A'BC + AB'$ has 8 literals, 3 product terms
  - $(A+B'+C)(A'+C)$ has 5 literals, 2 sum terms

Boolean Functions

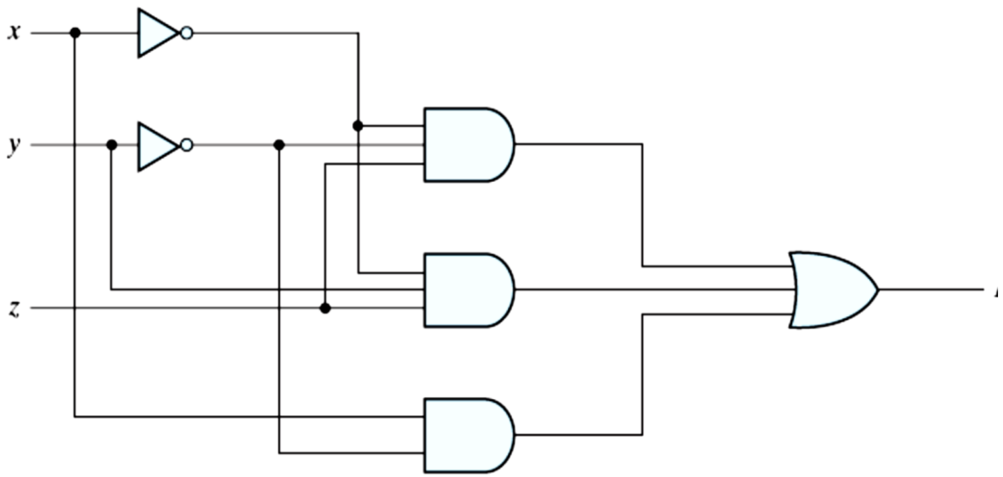
- Each Boolean function has
  - only one representation in truth table
  - but a variety of ways in algebraic form(Boolean equation) /gate implementation.
- Examples
  - $F_1 = x' y' z + x' y z + x y'$
  - $F_2 = x y' + x' z$
  - $F_1 = F_2$
    - Same truth table
    - Different algebraic expression

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

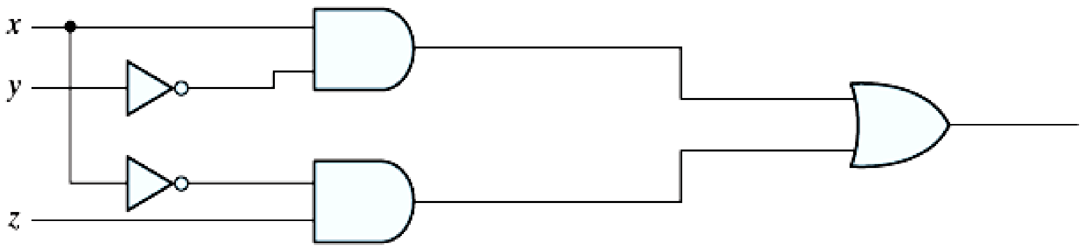
Gate Implementation

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

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



$$
\begin{aligned}
F_1 &= x'y'z + x'yz + xy' && \\
&= x'z(y' + y) + xy' && \text{Distributivity} \\
&= x'z + xy' = F_2 && \text{Complements}
\end{aligned}
$$



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:

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

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, find the dual form by replacing • with +, 0 with 1

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

Pay attention! The dual is not duality! the conversion is not finished yet

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

  - Step2, complement each literal in F

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

Exercise: simplify the function

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

baiyh@sustech.edu.cn

Outline

- Boolean Algebra & Axioms and Theorems
- 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. ($M_i = m_i’$)

Minterms and Maxterms

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

e.g. $M_0 = m_0' = (x+y+z)' = x'y'x'$ (DeMorgan's law)

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 (unique)
  - Either of the 2 canonical forms (unique som and pom form)
    - sum-of-minterms (sum of all the 1 terms)
      - $F = x'y'z + x'yz + xyz' + xyz$
        $= m_1 + m_3 + m_6 + m_7$
        $= \sum(1,3,6,7)$
    - product-of-maxterms (product of all the 0 terms)
      - $F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')$
        $= M_0 \cdot M_2 \cdot M_4 \cdot M_5$
        $= \prod(0,2,4,5)$

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, 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

Expand 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'

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

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

Expand 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

$F = xy + x'z$

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

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

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

$= \boxed{(x'+y+zz')}(x+z+yy')(y+z+xx')$

$= (x'+y+z)(x'+y+z')(x+z+y)(x+z+y')(\cancel{y+z+x})(\cancel{y+z+x'})$

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

$= M_0M_2M_4M_5$

$= \prod(0, 2, 4, 5)$

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

Exercise

- How to convert $f=x+y'z$ into canonical form (using $\sum$ or $\prod$)

$$
\begin{aligned}
f &= x+y'z \\
&= ?
\end{aligned}
$$

Standard Forms

- Standard forms: the terms that form the function may have fewer literals than the minterms/masterms.
  - Sum of products(sop): $F_1(x,y,z) = y' + xy + x'yz'$
  - Product of sums(pos): $F_2(x,y,z) = x(y'+z)(x'+y+z')$

  - Recall: canonical form examples
    - som: $F(a,b,c) = abc' + a'bc + abc$
    - pom: $F(a,b,c) = (a+b'+c)(a'+b+c)$
    - Comparing to canonical form, standard can have fewer literals

  - How about $F = A'B'C + ABC'$? (no enough information since we don’t know number of total inputs)
- Standard forms are not unique!

Canonical vs Standard Form

- We learnt simplification to reduce lengthy canonical forms into simpler, lower-cost standard forms, thereby using fewer logic gates to implement the circuit.

Property	Canonical Form	Standard Form
Uniqueness	Yes. A function has only one canonical SOP and one canonical POS form.	No. A function can have multiple standard forms.
Variable Requirement	Every term must include all input variables.	Terms may not include all input variables.
Expression Length	Usually long, with many terms and literals.	Usually shorter, simplified, with fewer terms and literals.
Relation to Truth Table	Directly corresponds; can be read directly from the truth table.	Does not directly correspond; it is a simplified result.
Purpose	Provides a unique and standardized description.	Provides a simpler and more efficient circuit implementation.

Outline

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

Other Logic Operations

- 2^n rows in the truth table of n binary variables.
- 2^{2n} functions for n binary variables.
- 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:

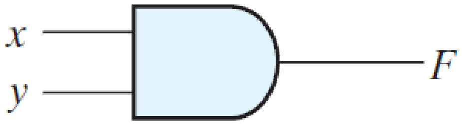
Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not 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}).
  - Inhibition (F_2) and implication (F_{13}) are not commutative or associative.
  - The other eight are used as standard gates:
    - complement (inverter) $F_{12} = x'$
    - transfer (buffer) $F_3 = x$
    - AND $F_1 = xy$
    - OR $F_7 = x+y$
    - NAND $F_{14} = (xy)'$
    - NOR $F_8 = (x+y)'$
    - XOR $F_6 = x \oplus y$
    - equivalence (XNOR) $F_9 = (x \oplus y)'$

Summary of Logic Gates

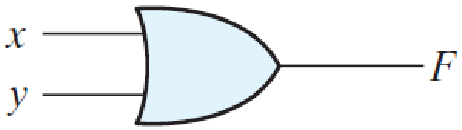
AND



$F = x \cdot y$

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

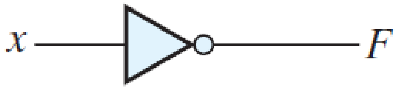
OR



$F = x + y$

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

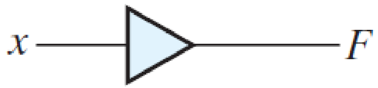
Inverter



$F = x'$

x	F
0	1
1	0

Buffer

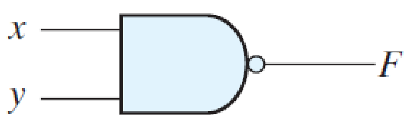


$F = x$

x	F
0	0
1	1

Summary of Logic Gates

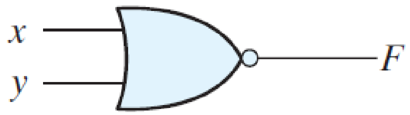
NAND



$F = (xy)'$

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

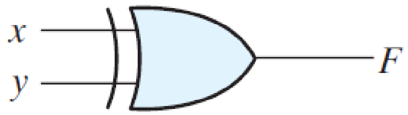
NOR



$F = (x + y)'$

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

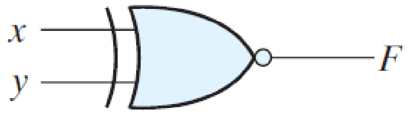
Exclusive-OR
(XOR)



$F = xy' + x'y$
$= x \oplus y$

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

Exclusive-NOR
or
equivalence

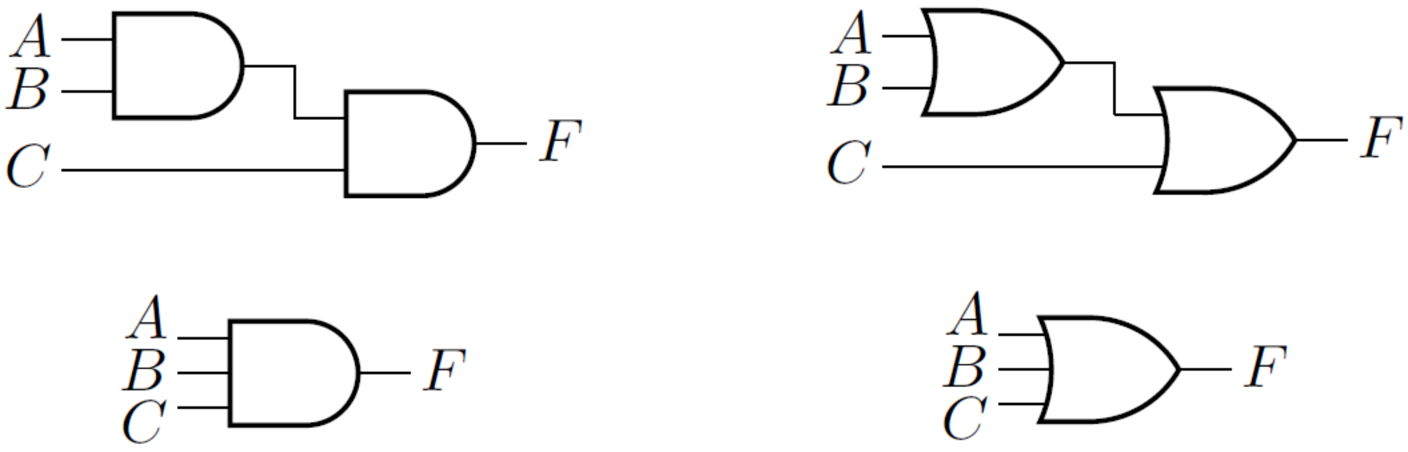


$F = xy + x'y'$
$= (x \oplus y)'$

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

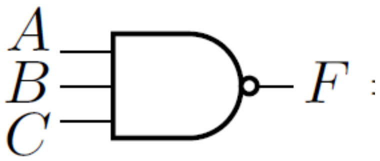
Multiple Inputs

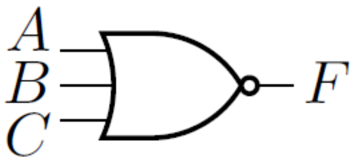
- Extension to multiple inputs
  - A gate can be extended to multiple inputs.
  - AND and OR are commutative and associative.
    - F = ABC = (AB)C
    - F = A + B + C = (A + B) + C



Multiple Inputs

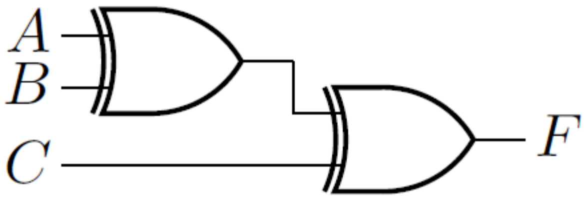
- NAND and NOR are commutative but not associative
  - $((AB)'C)' \neq (A(BC)')'$: does not follow associativity.
  - $((A + B)' + C)' \neq (A + (B + C)')'$: does not follow associativity.

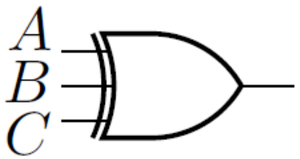




Multiple Inputs

- 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.





$F = A \oplus B \oplus C$

Note: The 3 inputs xor gate symbol is rarely used, we often combine two 2 inputs xor for logic diagram