Boolean Algebra and Logic Gates

Hsi-Pin Ma 馬席彬

https://eeclass.nthu.edu.tw/course/3452

Department of Electrical Engineering

National Tsing Hua University

Outline

- Algebraic Properties
- Boolean Algebra
- Two-valued Boolean Algebra
- Basic Theorems and Properties of Boolean Algebra
- Boolean Functions
- Normal and Standard Forms
- Other Logic Operations

Algebraic Properties

Basic Definition

- A *set* is a collection of objects with a common property.
- A *binary operator* on a set *S* is a rule that assigns to, each pair of elements in *S*, another unique element in S.
- The *axioms* (*postulates*) of an algebra are the basic assumptions from which all theorems of the algebra can be proved.
- It is assumed that there is an *equivalent relation* (=), which satisfies that *principle of substitution*.
  - It is ***reflexive***, ***symmetric***, and ***transitive***.

Most Common Axioms Used to Formulate an Algebra Structure (1/2)

- Closure
  - A set S is closed with respective to a binary operator * if and only if $\forall x, y \in S, (x * y) \in S$
- Associativity
  - A binary operator * on S is associative if and only if $\forall x, y, z \in S, (x * y) * z = x * (y * z)$
- Commutativity
  - A binary operator * defined on S is commutative if and only if $\forall x, y \in S, x * y = y * x$

Most Common Axioms Used to Formulate an Algebra Structure (2/2)

- Identity element
  - A set *S* has an identity element with respective to * if and only if $\exists e \in S$ such that $\forall x \in S, e * x = x * e = x$
- Inverse element
  - A set *S* having the identity element e with respect to * has an inverse if and only if $\forall x \in S, \exists y \in S$ such that $x * y = e$
- Distributivity
  - If * and • are binary operators on *S*, * is distributive over • if and only if $\forall x, y, z \in S, x * (y \cdot z) = (x * y) \cdot (x * z)$

Example: A Field

- A field is a set of elements, together with two binary operators.
- The set of real numbers together with the binary operators + and •, forms the field of real numbers.
  - '+' defines addition.
  - The additive identity is 0.
  - The additive inverse defines the subtraction.
  - The binary operator • defines multiplication.
  - The multiplicative identity is 1.
  - For $a \neq 0$, 1/a (the multiplicative inverse of a) defines devision.
  - The only distributive law applicable is that of • over +

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

Boolean Algebra

Axiomatic Definition

- Boolean algebra
  - An algebraic system of logic introduced by George Boole in 1854
- Switching algebra
  - A 2-valued Boolean algebra introduced by Claude Shannon in 1938
- ★ Huntington postulates
  - A formal definition of Boolean Algebra in 1904
  - Defined on a set B with binary operators + and •, and the equivalence relation =.

Huntington Postulates (1/2)

- Defined by a set *B* with binary operators + and •
  - Closure with respect to + and • **(P1)**
    - $x, y \in B \Rightarrow x + y \in B, x \cdot y \in B$
  - An identity element with respect to + and • **(P2)**
    - $0 + x = x + 0 = x, 1 \cdot x = x \cdot 1 = x$
  - Commutative with respective to + and • **(P3)**
    - $x + y = y + x, x \cdot y = y \cdot x$

Huntington Postulates (2/2)

- Distributive over + and $\bullet$ **(P4)**
  - $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
  - $x + (y \cdot z) = (x + y) \cdot (x + z)$
- $\forall x \in B, \exists x' \in B$ (called the ***complement*** of x) such that $x + x' = 1, \; x \cdot x' = 0$ **(P5)**
- There are ***at least*** 2 distinct elements in *B* **(P6)**
  - *There exist at least two* $x, y \in B$, such that $x \neq y$

Notes (1/2)

- The axioms are *independent*, none can be proved from others.
- *Associativity* is not included, since it can be derived (both + and •) from the given axioms.
- In *ordinary algebra*, + is *not distributive* over •.
- No *additive* or *multiplicative* inverses; no subtraction or division operations.
- Complement is *not available* in ordinary algebra.
- *B* is as yet undefined. It it to be defined as the set {0,1} (*two-valued Boolean Algebra*). In ordinary algebra, the set *S* can contain an infinite set of elements.

Notes (2/2)

- Boolean algebra
  - Set *B* of at least 2 elements (not *variables*)
  - Rules of operation for the 2 binary operators (+ and •)
  - Huntington postulates satisfied by the elements of B and the operators.
- Two-valued Boolean algebra (switching algebra)
  - $B \equiv \{0, 1\}$
  - The binary operators are defined as the logical AND (•) and OR (+). For convenience, a unary operation NOT (complement) is also included for basic operations.
  - The Huntington postulates are still valid.
- Unless otherwise noted, we will use the term ***Boolean algebra*** for the ***2-valued Boolean algebra***.

Two-valued Boolean Algebra

Two-valued Boolean Algebra

- $B \equiv \{0, 1\}$ is the set.
- The binary operator for + and•, and the unary operator *complement*.

input	output
$x\ y$	$x \cdot y$
0 0	0
0 1	0
1 0	0
1 1	1

input	output
$x\ y$	$x+y$
0 0	0
0 1	1
1 0	1
1 1	1

input	output
x	x'
0	1
1	0

Huntington Postulates Test (1/3)

- Closure
  - {0,1} of the operator results still in *B*.
- Identity elements
  - $0+0=0, 0+1=1+0=1$ (0: identity of +)
  - $1 \cdot 1=1, 1 \cdot 0=0 \cdot 1=0$ (1: identity of •)
- Commutative
  - Obviously from the table

Huntington Postulates Test (2/3)

- Distributive
  - Holds for • over +

x	y	z	$y + z$	$x \cdot (y + z)$	$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
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

  - Can be shown to hold for + over • .

Huntington Postulates Test (3/3)

- Complement
  - $x + x' = 1$ since $0 + 0' = 0 + 1 = 1$ and $1 + 1' = 1 + 0 = 1$
  - $x \cdot x' = 0$ since $0 \cdot 0' = 0 \cdot 1 = 0$ and $1 \cdot 1' = 1 \cdot 0 = 0$
- The two-valued Boolean algebra has two distinct elements, 0 and 1, with $0 \neq 1$.

Basic Theorems and Properties of Boolean Algebra

Duality

- Every algebraic expression deducible from the postulates of Boolean algebra remains valid if the ***operators*** and ***identity elements*** are ***interchanged***.
  - Binary operators: AND <=> OR
  - Identity elements: 1 <=> 0

Postulates and Theorems of Boolean Algebra

	(a)	(b)
P2	$x+0 = x$	$x \cdot 1 = x$
p5	$x+x' = 1$	$x \cdot x' = 0$
T1	$x + x = x$	$x \cdot x = x$
T2	$x + 1 = 1$	$x \cdot 0 = 0$
T3, involution	$(x')' = x$	
p3, commutative	$x+y = y+x$	$x \cdot y = y \cdot x$
T4, associative	$x+(y+z)=(x+y)+z$	$x \cdot (y \cdot z) =(x \cdot y) \cdot z$
P4, distributive	$x \cdot (y+z)=x \cdot y+x \cdot z$	$x+y \cdot z=(x+y) \cdot (x+z)$
T5, DeMorgan	$(x+y)' =x' \cdot y'$	$(x \cdot y)'=x'+y'$
T6, absorption	$x+x \cdot y = x$	$x \cdot (x+y) =x$

Basic Theorems (1/5)

- Theorem 1 (Idempotency)
  - $(\mathrm{a})x + x = x, \ (\mathrm{b})x \cdot x = x$

	Statement	*Justification*
$x + x$	$= (x + x) \cdot 1$	postulate 2(b)
	$= (x + x)(x + x')$	5(a)
	$= x + xx'$	4(b)
	$= x + 0$	5(b)
	$= x$	2(a)

	Statement	*Justification*
$x \cdot x$	$= xx + 0$	postulate 2(a)
	$= xx + xx'$	5(b)
	$= x(x + x')$	4(a)
	$= x \cdot 1$	5(a)
	$= x$	2(b)

Basic Theorems (2/5)

- Theorem 2

$$(a) x + 1 = 1, \quad (b) x \cdot 0 = 0$$

	Statement	*Justification*
$x + 1$	$= 1 \cdot (x + 1)$	postulate 2(b)
	$= (x + x')(x + 1)$	5(a)
	$= x + x' \cdot 1$	4(b)
	$= x + x'$	2(b)
	$= 1$	5(a)

– (b) can be proved by duality

Basic Theorems (3/5)

- Theorem 3 (Involution)

$$(x')' = x$$

  - P5 defines the complement of x, and the complement of x′ is both x and (x′)′
- Theorem 4 (Associativity)

$$\textbf{(a)}\ x + (y + z) = (x + y) + z, \quad \textbf{(b)}\ x(yz) = (xy)z$$

  - Can be proved by truth table

Basic Theorems (4/5)

- Theorem 5 (DeMorgan's Theorem)
  - **(a)** $(x+y)' = x' \cdot y'$, **(b)** $(xy)' = x' + y'$
  - Duality principle

x	y	x+y	(x+y)′	x′	y′	x′y′
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Basic Theorems (5/5)

- Theorem 6 (Absorption)
  - **(a)** $x + xy = x$, **(b)** $x(x + y) = x$

	Statement	*Justification*
$x + xy$	$= x \cdot 1 + xy$	postulate 2(b)
	$= x(1 + y)$	4(a)
	$= x(y + 1)$	3(a)
	$= x \cdot 1$	2(a)
	$= x$	2(b)

Operator Priority

- Operator precedence
  - Parentheses
  - NOT
  - AND
  - OR
- Examples
  - $xy'+z$
  - $(xy+z)'$

Boolean Functions

Boolean Functions

- A Boolean function is an algebraic expression formed with
  - Binary variables
  - Logic operators AND, OR
  - Unary NOT
  - Parentheses
  - An equal sign
- Examples
  - $F_1 = x + y'z$
  - $F_2 = x'y'z + x'yz + xy'$

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

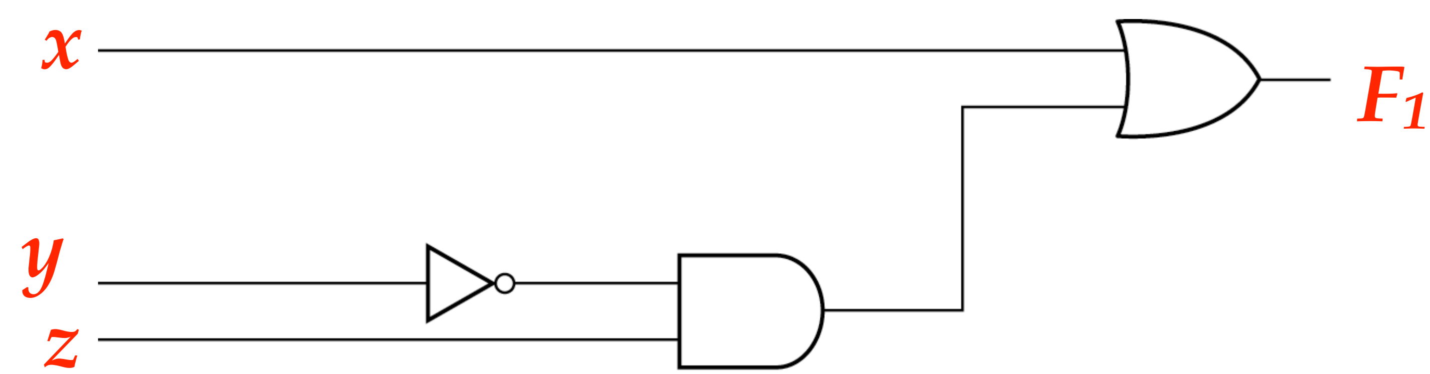
Boolean Functions

- Can be represented by a truth table, with 2^n rows in the table (n: # of variable in the function)
- There are infinitely many algebraic expressions that specify a given Boolean function. It's important to find the ***simplest*** one. (cost)
- Any Boolean function can be transformed in a straightforward manner from an algebraic expression into a ***logic diagram*** of only AND, OR, and NOT gates.

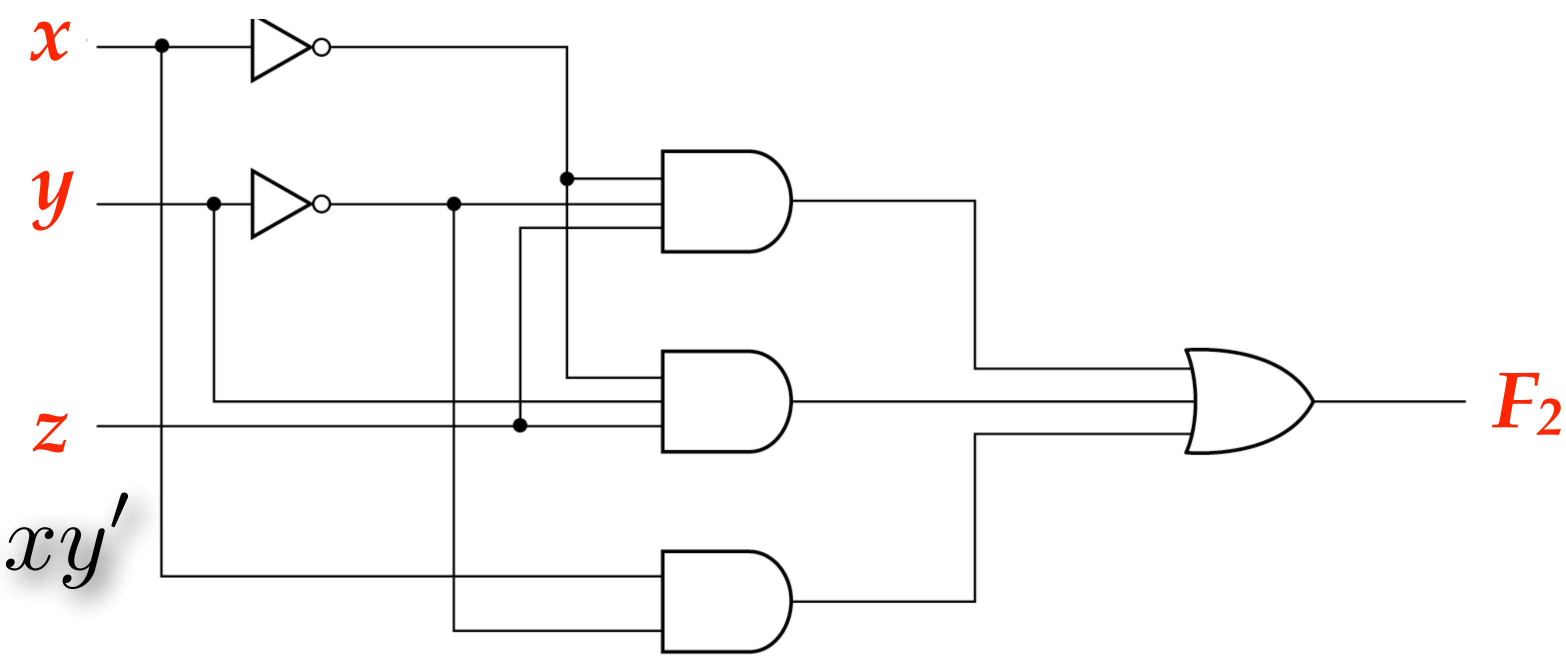
Gate Implementation

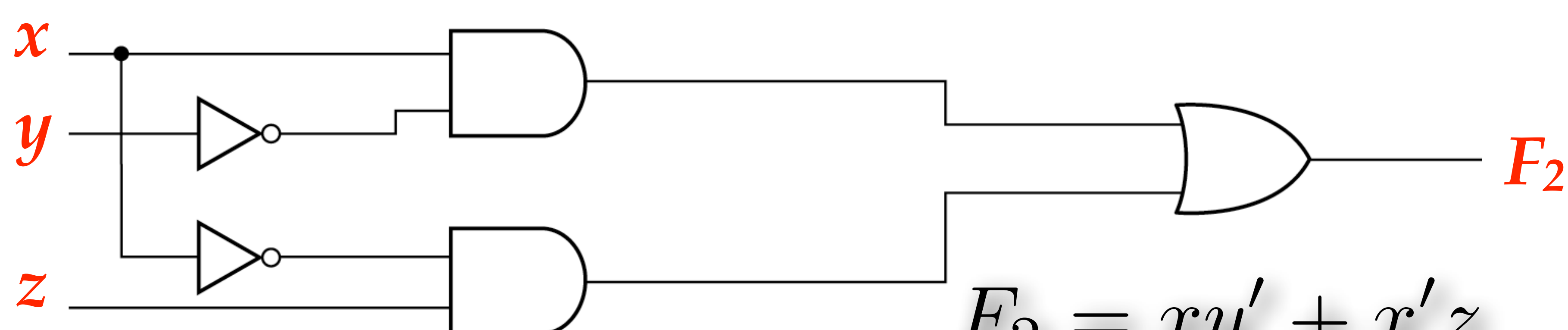
- Logic diagrams

$$F_1 = x + y'z$$



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





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

Boolean Functions

- A ***literal*** is a *variable* or its complement in a Boolean expression
  - $F_2 = x'y'z + x'yz + xy'$
    - 8 literals,
    - 1 OR term (sum term) and 3 AND terms (product terms).
    - literal: a input to a gate, term: implementation with a gate
- The complement of any function *F* is *F'*, which can be obtained by DeMorgan's Theorem.
  - Take the dual of *F*, and then complement each literal in *F*.
  - $F_2' = (x'y'z + x'yz + xy')' = (x+y+z')(x+y'+z')(x'+y)$

Algebraic Manipulation (1/2)

- Minimize the number of literals and terms for a simpler circuits (less expensive)
- Algebraic manipulation can minimize literals and terms. However, no specific rules to guarantee the optimal results.
- CAD tools for logic minimization are commonly used today.

Algebraic Manipulation (2/2)

- Some useful rules
  - $x(x' + y) = xy$
  - $x + x'y = x + y$
  - $xy + yz + x'z = xy + x'z$ (the Consensus Theorem I)
  - $(x + y)(y + z)(x' + z) = (x + y)(x' + z)$ (the Consensus Theorem II, duality from Consensus Theorem I)

Canonical and Standard Forms

Minterms and Maxterms

- Minterm (m_i) (or *standard product term*)
  - An AND (product) term consists of all literals (each appears exactly **once**) in their normal form or in their complement form, but not in both
    - eg. two binary variable x and y, the minterms are xy, xy′, x′y, x′y′
  - *n* variable can be combined to form 2^n minterms
- Maxterms (M_i) (or *standard sum term*)
  - An OR (sum) term consists of all literals (each appears exactly **once**) in their normal form or in their complement form, but not in both
    - eg. two binary variable x and y, the maxterms are x+y, x+y′, x′+y, x′+y′
- 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)

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

Example

- A Boolean function can be expressed by
  - a truth table
  - sum-of-minterms
    - $f_1 = x'y'z + xy'z' + xyz$
      $= m_1 + m_4 + m_7 = \sum(1,4,7)$
    - $f_2 = x'yz + xy'z + xyz' + xyz$
      $= m_3 + m_5 + m_6 + m_7 = \sum(3,5,6,7)$
  - product-of-maxterms
    - $f_1 = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$
      $= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 = \Pi(0,2,3,5,6)$
    - $f_2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$
      $= M_0 \cdot M_1 \cdot M_2 \cdot M_4 = \Pi(0,1,2,4)$

x	y	z	f_1	f_2	f_1'	f_2'
0	0	0	0	0	1	1
0	0	1	1	0	0	1
0	1	0	0	0	1	1
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	1	0
1	1	0	0	1	1	0
1	1	1	1	1	0	0

Canonical Forms

- Any function can be represented by either of the 2 canonical forms
  - To convert from one canonical from to another, interchange $\sum$ and Π , and list the numbers that were excluded from the original form.
  - $f_1 = \sum(1,4,7)$ is the sum of 1-minterms for f_1.
  - $f_1' = \sum(0,2,3,5,6)$ is the sum of 0-minterms for f_1.
- How to convert f=x+yz into canonical form?
  - by truth table
  - by expanding the missing variables in each term, using 1=x+x', 0=xx'

Standard Forms

- Canonical forms are seldom used.
- Standard forms
  - sum-of-products (sop)
    - Product terms (implicants) are the AND terms, which can have fewer literals than the minterms.
  - product-of-sums (pos)
    - Sum terms are the OR terms, which can have fewer literals than maxterms.
- Standard forms are not unique!

Standard Forms

- Standard form examples
  - $f_1=xy+xy'z+x'yz$ (sop form)
  - $f_1'=(x'+y')(x'+y+z')(x+y'+z')$ (pos form)
- Nonstandard forms can have fewer literals than standard forms
  - $xy+xy'z+xy'w=x(y+y'z+y'w)=x(y+y'(z+w))$
  - $xy+yz+zx=xy+(x+y)z=x(y+z)+yz=xz+y(x+z)$

Other Gate Types

Other Logic Operations

- For n binary variables
  - 2^n rows in the truth table
  - 2^{2^n} functions
  - 16 different Boolean functions if n=2
- All the new symbols except for the XOR are not in common use by digital designers

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

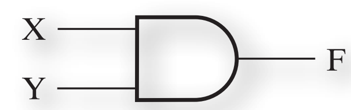
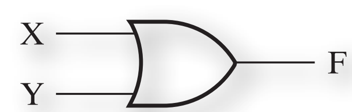
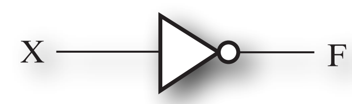
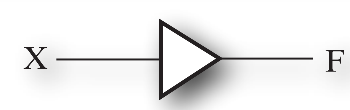
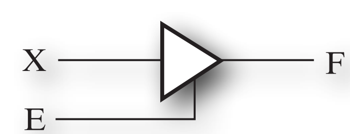
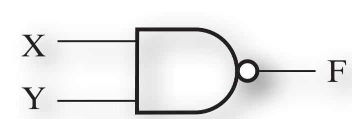
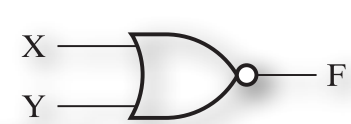
Boolean Expressions for the 16 Functions of Two Variables

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

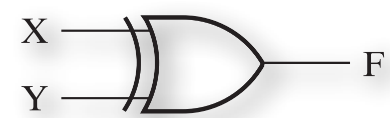
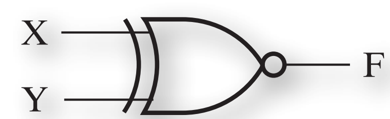
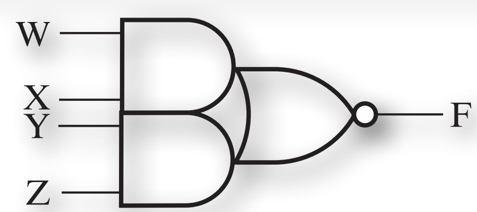
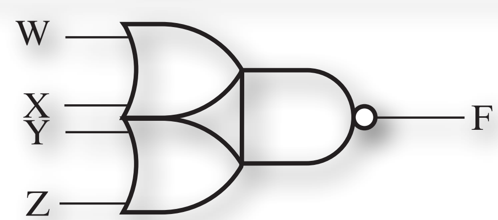
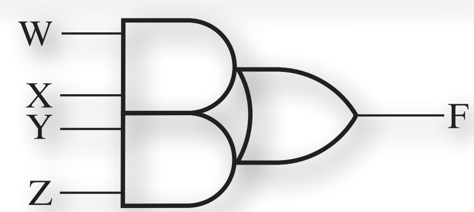
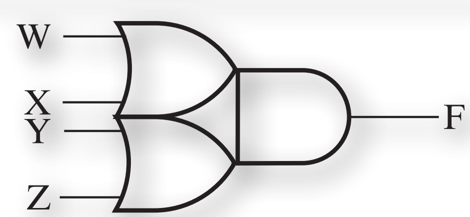
Digital Logic Gates

- Consider 16 functions
  - Two functions generate constants
    - Null/Zero, Identity/One
  - Four one-variable functions
    - Complement (inverter) , Transfer (buffer)
  - 10 functions that define 8 specific binary functions
    - AND, Inhibition, XOR, OR, NOR, Equivalence (XOR), Implication, NAND
    - Inhibition and Implication are neither commutative nor associative
    - NAND and NOR are commutative but not associative

Primitive Digital Logic Gates

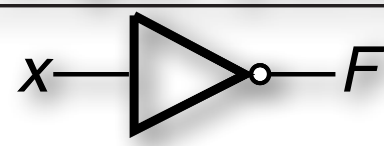
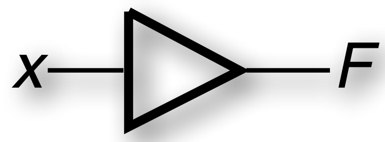
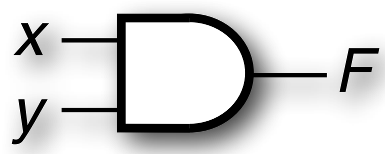
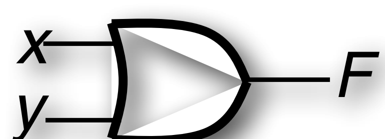
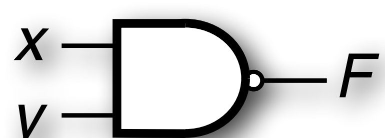
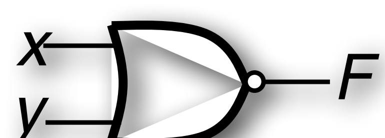
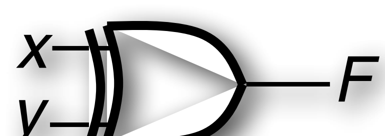
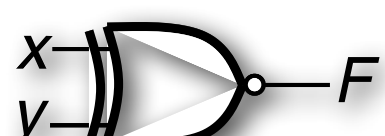
Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
AND		F = XY	X Y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		F = X + Y	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT (inverter)		$F = \overline{X}$	X | F 0 | 1 1 | 0
Buffer		F = X	X | F 0 | 0 1 | 1
3-State Buffer			E X | F 0 0 | Hi-Z 0 1 | Hi-Z 1 0 | 0 1 1 | 1
NAND		$F = \overline{X \cdot Y}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

Complex Digital Logic Gates

Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
Exclusive-OR (XOR)		$F = X\overline{Y} + \overline{X}Y$ $= X \oplus Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR (XNOR)		$F = XY + \overline{X}\,\overline{Y}$ $= \overline{X \oplus Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1
AND-OR-INVERT (AOI)		$F = \overline{WX + YZ}$	
OR-AND -INVERT (OAI)		$F = \overline{(W + X)(Y + Z)}$	
AND-OR (AO)		$F = WX + YZ$	
OR-AND (OA)		$F = (W + X)(Y + Z)$	

Eight Basic Digital Logic Gates

typical CMOS implementation

Name	Graphic symbol	Function	No. transistors (cost)	Gate delay (ns) (performance)
Inverter		$F = x'$	2	1
Driver		$F = x$	4	2
AND		$F = xy$	6	2.4
OR		$F = x + y$	6	2.4
NAND		$F = (xy)'$	4	1.4
NOR		$F = (x + y)'$	4	1.4
XOR		$F = x \oplus y$	14	4.2
XNOR		$F = x \odot y$	12	3.2

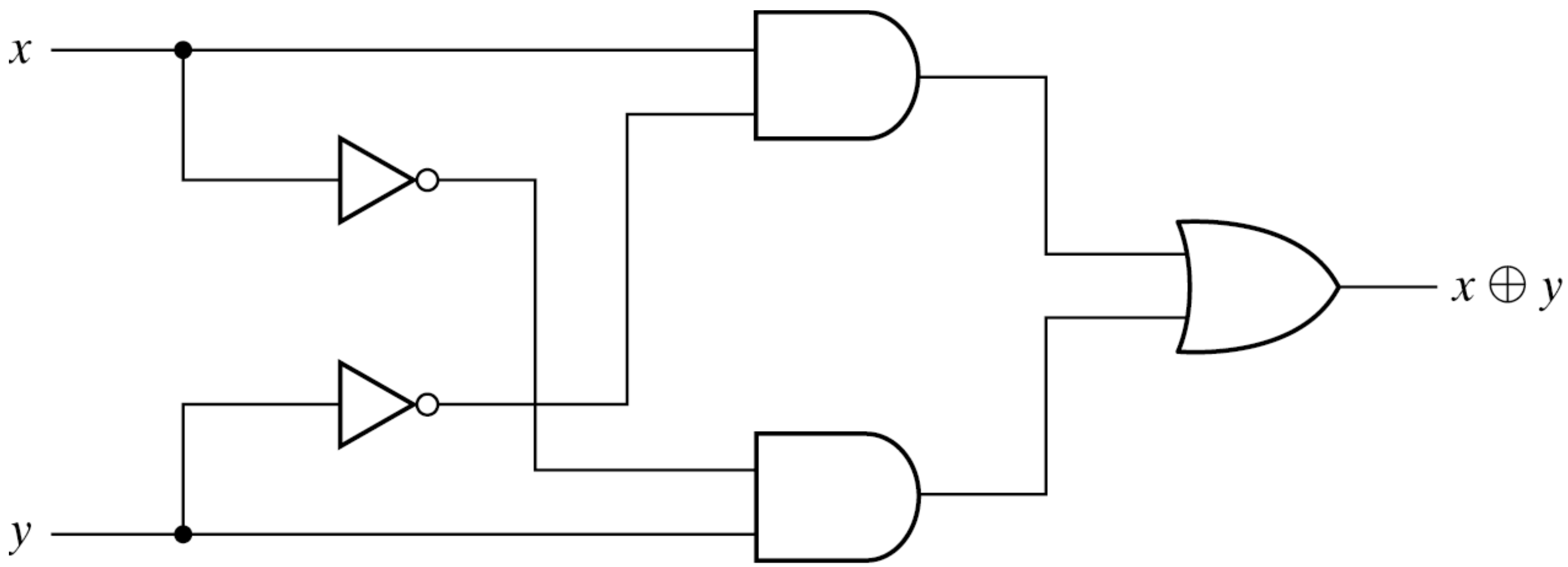
[Gajski]

Exclusive-OR (XOR) Function

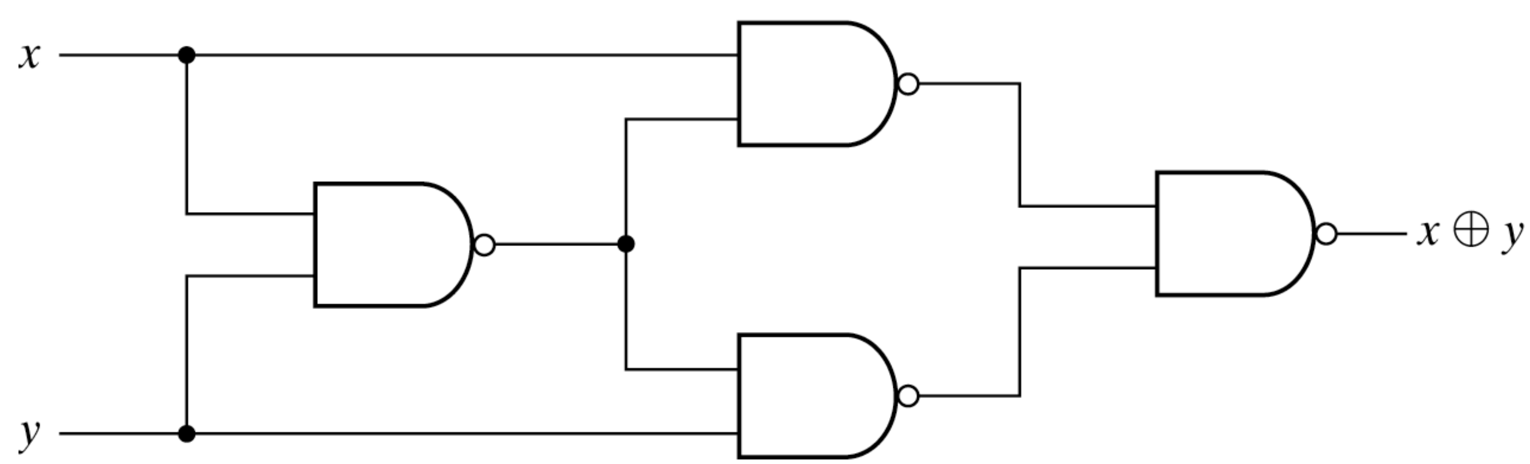
- XOR $x \oplus y = xy' + x'y$
- XNOR $(x \oplus y)' = xy + x'y'$
- Identity properties
  - $x \oplus 0 = x; x \oplus 1 = x'$
  - $x \oplus x = 0; x \oplus x' = 1$
  - $x \oplus y' = (x \oplus y)'; x' \oplus y = (x \oplus y)'$
- Commutative and associative
  - $A \oplus B = B \oplus A$
  - $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

XOR Implementation

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



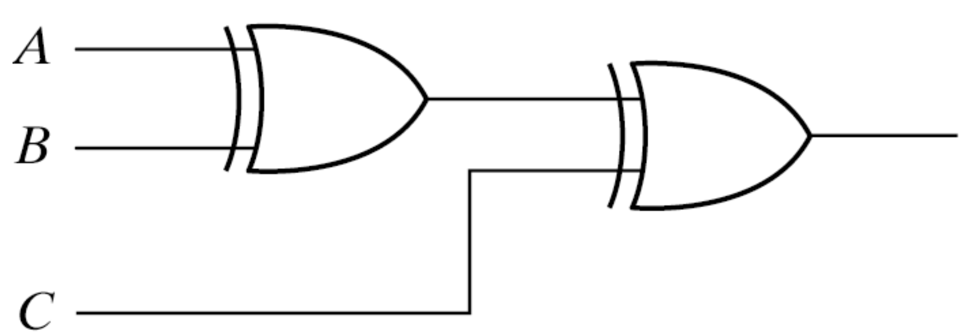
(a) With AND-OR-NOT gates



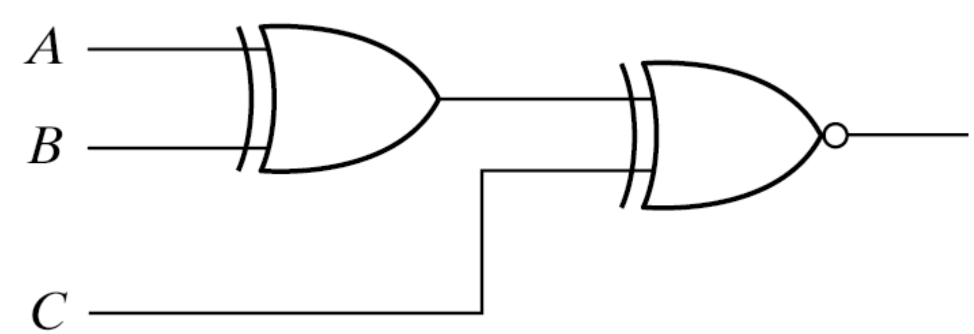
(b) With NAND gates

Odd and Even Function

$$
\begin{aligned}
A \oplus B \oplus C &= (AB' + A'B)C' + (AB + A'B')C \\
&= AB'C' + A'BC' + ABC + A'B'C \\
&= \sum(1, 2, 4, 7)
\end{aligned}
$$



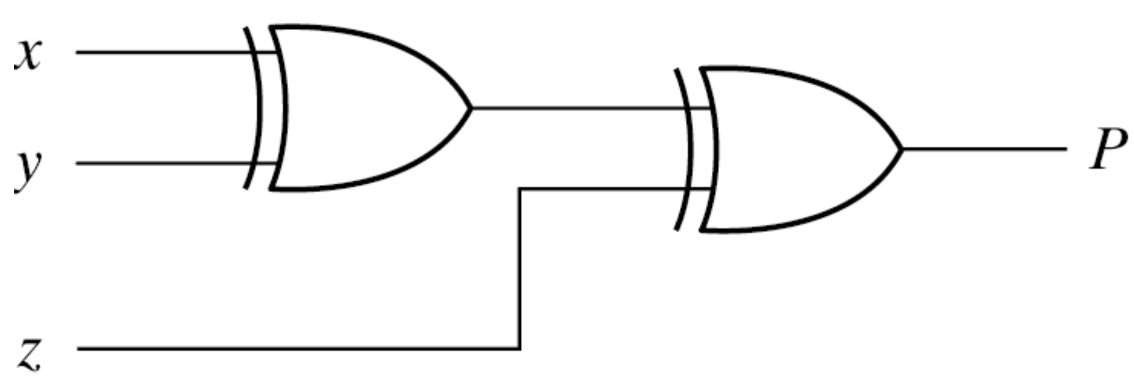
(a) 3-input odd function



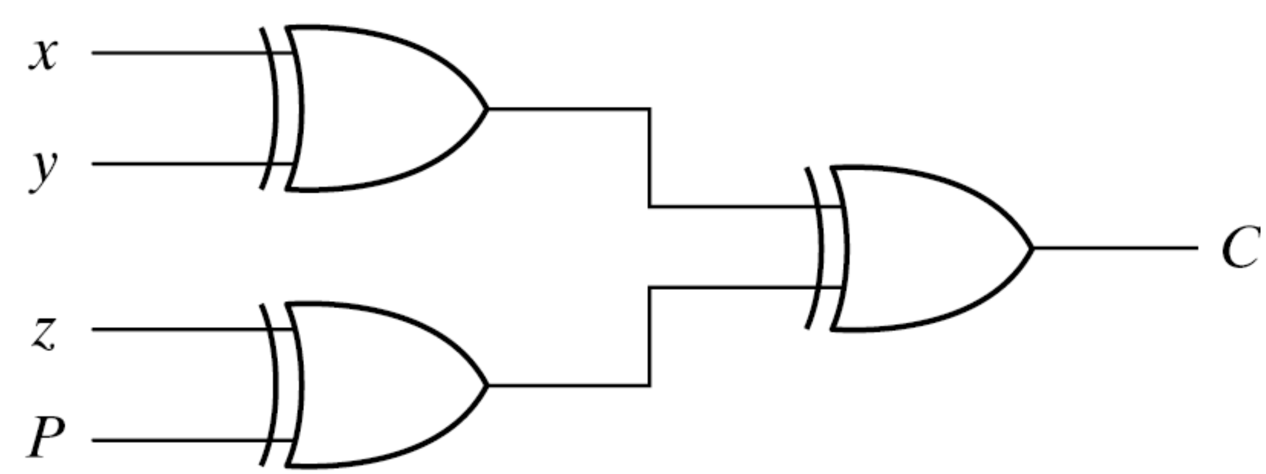
(b) 3-input even function

Parity Generation and Checking

- Parity generation
  - $P = x \oplus y \oplus z$
- Parity check
  - $C = x \oplus y \oplus z \oplus P$
  - C=1: an odd number of data bit error
  - C=0: correct or and even # of data bit error

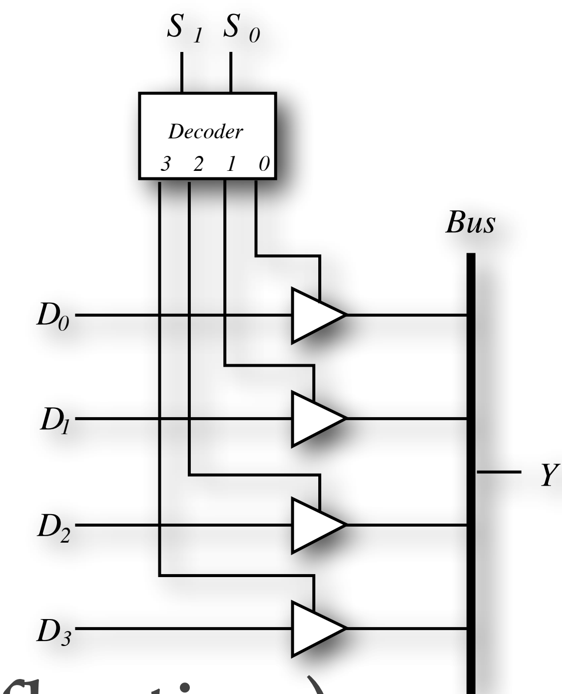


(a) 3-bit even parity generator

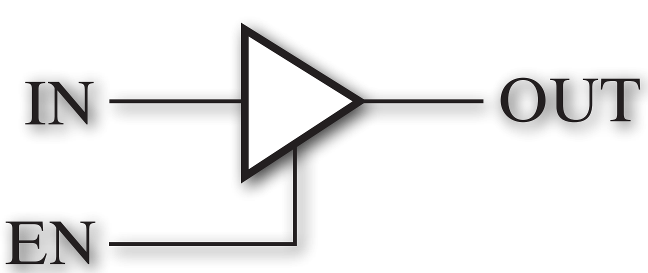


(a) 4-bit even parity checker

High-Impedance Outputs



- Three-state buffer
  - Three state: 1, 0, Hi-Z
  - Output: Hi-Z, Z, z (behaves as an open circuit, floating)
- Two useful properties
  - Hi-Z outputs can be connected together if no two or more gates drive the line at the same time to opposite 1 and 0 values.
  - Bidirectional input/output



(a) Logic symbol

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

(b) Truth table