Boolean Algebra and Logic Gates

Boolean Algebra and Logic Gates

ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Acknowledgement

This slide show is intended for use in class, and is not a complete document. Students need to refer to the book to read more lessons and exercises. Students have the right to download and store lecture slides for reference purposes; Do not redistribute or use for purposes outside of the course.

[1] Morris R. Mano (Author), Michael D. Ciletti, [2019] **Digital Design: With an Introduction to the Verilog HDL,** chapter 2 - Boolean algebra and Logic gates, 5th Edition.

🖂 **trantrungtin.tdtu.edu.vn**

Chapter Objectives

1. Boolean Algebra
2. Precedence of Operators
3. Truth Table
4. Duality
5. Basic Theorems
6. Complement of Functions
7. Standard Forms
8. Minterms and Maxterms
9. Canonical Forms

Syllabus

2.1 Introduction

2.2 Basic Definitions

2.3 Axiomatic Def. of Bool. Algebra

2.4 Basic Theorems and Properties of Boolean Algebra

2.5 Boolean Functions

2.6 Canonical and Standard Forms

2.7 Other Logic Operations

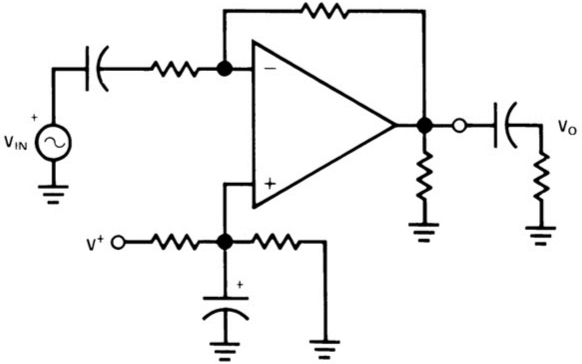
2.8 Digital Logic Gates

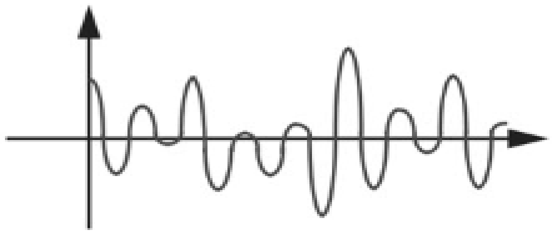
2.9 Integrated Circuits

2.10 Summary

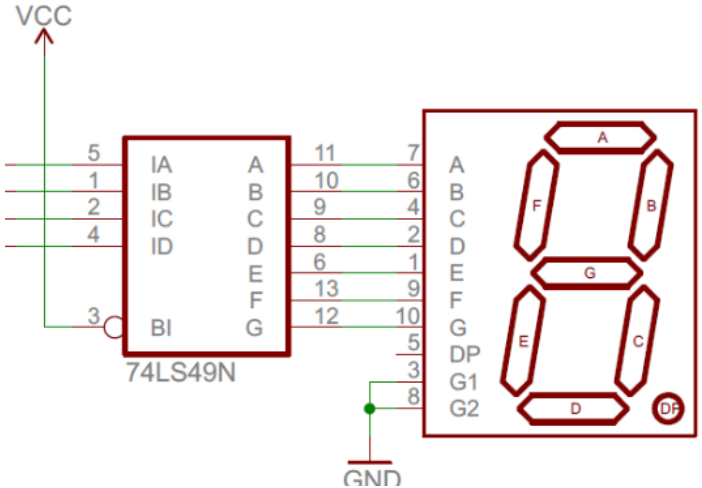
2.1 Introduction

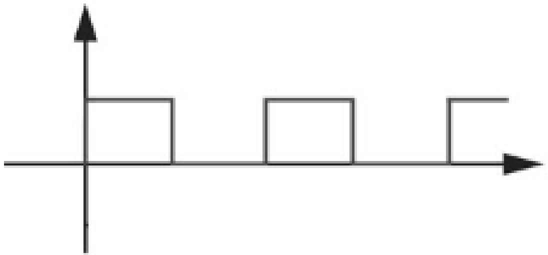
- **Analog circuit**





- **Digital circuit**





2.1 Digital Circuits

- Advantages of digital circuits over analog circuits
    - More reliable (simpler circuits, less noise-prone)
    - Specified accuracy (determinable)
    - Abstraction can be applied using simple mathematical model – **Boolean Algebra**
    - Ease design, analysis and simplification of digital circuit – **Digital Logic Design**

2.2 Types Of Logic Blocks

- Combinational: no memory, output depends solely on the input
  - Gates
  - Decoders, multiplexers
  - Adders, multipliers
- Sequential: with memory, output depends on both input and current state
  - Counters, registers
  - Memories

Boolean Algebra

- Boolean values:
  - True (1)
  - False (0)
- Connectives
  - Conjunction (AND)
    - A · B; A ∧ B
  - Disjunction (OR)
    - A + B; A ∨ B
  - Negation (NOT)
    - $\overline{A}$; ¬A; A'

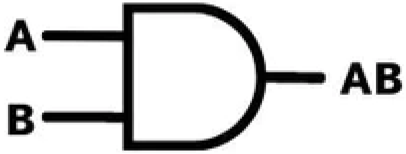
Truth tables

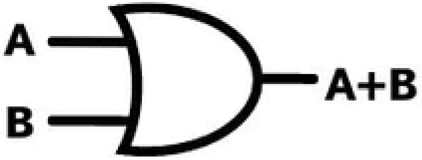
A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

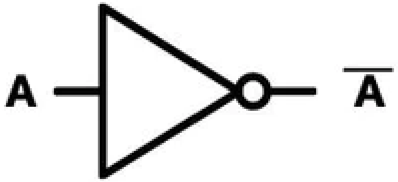
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

A	A'
0	1
1	0

Logic gates







And (·)

- Do write the AND operator · instead of omitting it.
  - Example: Write a·b instead of ab
  - Why? Writing ab could mean it is a 2-bit value.

Laws Of Boolean Algebra

- Identity laws
  - $A + 0 = 0 + A = A$; $A \cdot 1 = 1 \cdot A = A$
- Inverse/complement laws
  - $A + A' = 1$; $A \cdot A' = 0$
- Commutative laws
  - $A + B = B + A$; $A \cdot B = B \cdot A$
- Associative laws
  - $A + (B + C) = (A + B) + C$; $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
- Distributive laws
  - $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$; $A + (B \cdot C) = (A + B) \cdot (A + C)$

Precedence Of Operators

- Precedence from highest to lowest
  - Not
  - And
  - Or
- Examples:
  - $A \cdot B + C = (A \cdot B) + C$
  - $X + Y' = X + (Y')$
  - $P + Q' \cdot R = P + ((Q') \cdot R)$
- Use parenthesis to overwrite precedence. Examples:
  - $A \cdot (B + C)$
  - $(P + Q)' \cdot R$

Truth Table

- Provide a listing of every possible combination of inputs and its corresponding outputs.
  - Inputs are usually listed in binary sequence.
- Example
  - Truth table with **3 inputs** and **2 outputs**

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

Proof Using Truth Table

- Prove: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
  - Construct truth table for LHS and RHS

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

Check that column for LHS = column for RHS

Duality

- If the AND/OR operators and identity elements 0/1 in a Boolean equation are interchanged, it remains valid
- Example:
  - The dual equation of $a+(b\cdot c)=(a+b)\cdot(a+c)$ is $a\cdot(b+c)=(a\cdot b)+(a\cdot c)$
- Duality gives free theorems – “two for the price of one”. You prove one theorem and the other comes for free!
- Examples:
  - If $(x+y+z)' = x'\cdot y'\cdot z'$ is valid, then its dual is also valid:
    $(x\cdot y\cdot z)' = x'+y'+z'$
  - If $x+1 = 1$ is valid, then its dual is also valid:
    $x\cdot 0 = 0$

Basic Theorems (1/2)

- Idempotency
  - $X + X = X$; $X \cdot X = X$
- Zero and One elements
  - $X + 1 = 1$; $X \cdot 0 = 0$
- Involution
  - $(X')' = X$
- Absorption
  - $X + X \cdot Y = X$; $X \cdot (X + Y) = X$
- Absorption (variant)
  - $X + X' \cdot Y = X + Y$; $X \cdot (X' + Y) = X \cdot Y$

Basic Theorems (2/2)

- DeMorgan’s
  - $(X + Y)' = X' \cdot Y'$; $(X \cdot Y)' = X' + Y'$
- DeMorgan’s Theorem can be generalised to more than two variables, example:

$$(A + B + \ldots + Z)' = A' \cdot B' \cdot \ldots \cdot Z'$$

- Consensus
  - $X \cdot Y + X' \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$
  - $(X+Y) \cdot (X'+Z) \cdot (Y+Z) = (X+Y) \cdot (X'+Z)$

Proving A Theorem

- Theorems can be proved using truth table, or by algebraic manipulation using other theorems/laws.
- Example: Prove absorption theorem $X + X \cdot Y = X$

$X + X \cdot Y = X \cdot 1 + X \cdot Y$ (by identity)

$= X \cdot (1+Y)$ (by distributivity

$= X \cdot (Y+1)$ (by commutativity)

$= X \cdot 1$ (by one element)

$= X$ (by identity)

- By duality, we have also proved $X \cdot (X+Y) = X$

Boolean Functions

- Examples of Boolean functions (logic equations):

F1(x,y,z) = x·y·z'

F2(x,y,z) = x + y'·z

F3(x,y,z) = x'·y'·z + x'·y·z + x·y'

F4(x,y,z) = x·y' + x'·z

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

Complement

- Given a Boolean function F, the complement of F, denoted as F', is obtained by interchanging 1 with 0 in the function's output values.
- Example: F1 = x·y·z'
- What is F1' ?

x	y	z	F1	F1'
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	1	
1	1	1	0	

Canonical and Standard Forms (1/2)

- Certain types of Boolean expressions lead to circuits that are desirable from implementation viewpoint.
- Two standard forms:
  - Sum-of-Products
  - Product-of-Sums
- Literals
  - A Boolean variable on its own or in its complemented form
  - Examples: x, x', y, y'
- Product term
  - A single literal or a logical product (AND) of several literals
  - Examples: x, x·y·z', A'·B, A·B, d·g'·v·w

Canonical and Standard Forms (2/2)

- Sum term
  - A single literal or a logical sum (OR) of several literals
  - Examples: x, $x+y+z'$, $A'+B$, $A+B$, $c+d+h'+j$
- Sum-of-Products (SOP) expression
  - A product term or a logical sum (OR) of several product terms
  - Examples: x, $x + y \cdot z'$, $x \cdot y' + x' \cdot y \cdot z$, $A \cdot B + A' \cdot B'$,
    $A + B' \cdot C + A \cdot C' + C \cdot D$
- Product-of-Sums (POS) expression
  - A sum term or a logical product (AND) of several sum terms
  - Examples: x, $x \cdot (y+z')$, $(x+y') \cdot (x'+y+z)$,
    $(A+B) \cdot (A'+B')$, $(A+B+C) \cdot D' \cdot (B'+D+E')$
- Every Boolean expression can be expressed in SOP or POS.

Do It Yourself

- Put the right ticks in the following table.

Expression	***SOP?***	***POS?***
$X' \cdot Y + X \cdot Y' + X \cdot Y \cdot Z$		
$(X+Y') \cdot (X'+Y) \cdot (X'+Z')$		
$X' + Y + Z$		
$X \cdot (W' + Y \cdot Z)$		
$X \cdot Y \cdot Z'$		
$W \cdot X' \cdot Y + V \cdot (X \cdot Z + W')$		

Minterms & Maxterms (1/2)

- A minterm of **n** variables is a product term that contains n literals from all the variables.
  - Example: On 2 variables x and y, the minterms are:
  - $x' \cdot y'$, $x' \cdot y$, $x \cdot y'$ and $x \cdot y$
- A maxterm of n variables is a sum term that contains n literals from all the variables.
  - Example: On 2 variables x and y, the maxterms are:
  - $x'+y'$, $x'+y$, $x+y'$ and $x+y$
- In general, with **n** variables we have $\mathbf{2^n}$ minterms and $\mathbf{2^n}$ maxterms.

Minterms & Maxterms (2/2)

- The minterms and maxterms on 2 variables are denoted by m0 to m3 and M0 to M3 respectively.

x	y	Minterms		Maxterms	
		Term	Notation	Term	Notation
0	0	$x' \cdot y'$	m0	$x+y$	M0
0	1	$x' \cdot y$	m1	$x+y'$	M1
1	0	$x \cdot y'$	m2	$x'+y$	M2
1	1	$x \cdot y$	m3	$x'+y'$	M3

- Each minterm is the complement of the corresponding maxterm
  - Example: $m2 = x \cdot y'$

    $m2' = (x \cdot y')' = x' + (y')' = x' + y = M2$

Canonical Forms

- Canonical/normal form: a unique form of representation.
    - Sum-of-minterms = Canonical sum-of-products
    - Product-of-maxterms = Canonical product-of-sums

Sum-Of-Minterms

- Given a truth table, example:
- Obtain sum-of-minterms expression by gathering the minterms of the function (where output is 1).

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

$F1 = x \cdot y \cdot z' = m6$

$F2 =$

$F3 =$

Product-Of-Maxterms

- Given a truth table, example:
- Obtain product-of-maxterms expression by gathering the maxterms of the function (where output is 0).

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

F1=

F2 = (x+y+z) · (x+y'+z) · (x+y'+z')

= M0 · M2 · M3 = ΠM(0,2,3)

F3 =

Conversion

- We can convert between sum-of-minterms and product-of-maxterms easily
- Example: **F2 = Σm(1,4,5,6,7) = ΠM(0,2,3)**
- Why? See F2' in truth table.

- F2' = m0 + m2 + m3

  Therefore,
  F2 = (m0 + m2 + m3)'
  = m0' · m2' · m3' (by DeMorgan’s)
  = M0 · M2 · M3 (m*x*' =M*x*)

x	y	z	F2	F2'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

2.7 Other Logic Operations

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

Table 2.7
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

Table 2.8
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

2.8 Digital Logic Gates

- Boolean expression: AND, OR and NOT operations
- Constructing gates of other logic operations
  - The feasibility and economy;
  - The possibility of extending gate's inputs;
  - The basic properties of the binary operations (commutative and associative);
  - The ability of the gate to implement Boolean functions.

Standard Gates

- Consider the 16 functions in Table 2.8 (slide 33)
  - 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: complement (F_{12}), transfer (F_3), AND (F_1), OR (F_7), NAND (F_{14}), NOR (F_8), XOR (F_6), and equivalence (XNOR) (F_9) are used as standard gates.
  - Complement: inverter.
  - Transfer: buffer (increasing drive strength).
  - Equivalence: XNOR.

Summary of Logic Gates

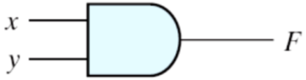
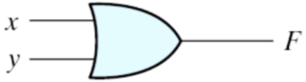
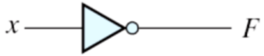
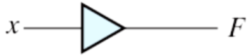
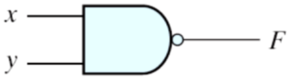
Name	Graphic symbol	Algebraic function	Truth table
AND		$F = xy$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0
Buffer		$F = x$	x | F 0 | 0 1 | 1

Figure 2.5 Digital logic gates

Summary of Logic Gates

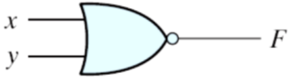
NAND



$F = (xy)'$

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

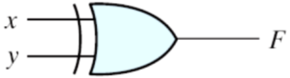
NOR



$F = (x + y)'$

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

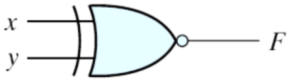
Exclusive-OR (XOR)



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

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

Figure 2.5 Digital logic gates

Multiple Inputs

- Extension to multiple inputs
  - A gate can be extended to multiple inputs.
    - If its binary operation is commutative and associative.
  - AND and OR are commutative and associative.
    - OR
      - $x+y = y+x$
      - $(x+y)+z = x+(y+z) = x+y+z$
    - AND
      - $xy = yx$
      - $(x\ y)z = x(y\ z) = x\ y\ z$

Multiple Inputs

- NAND and NOR are commutative but not associative → they are not extendable.

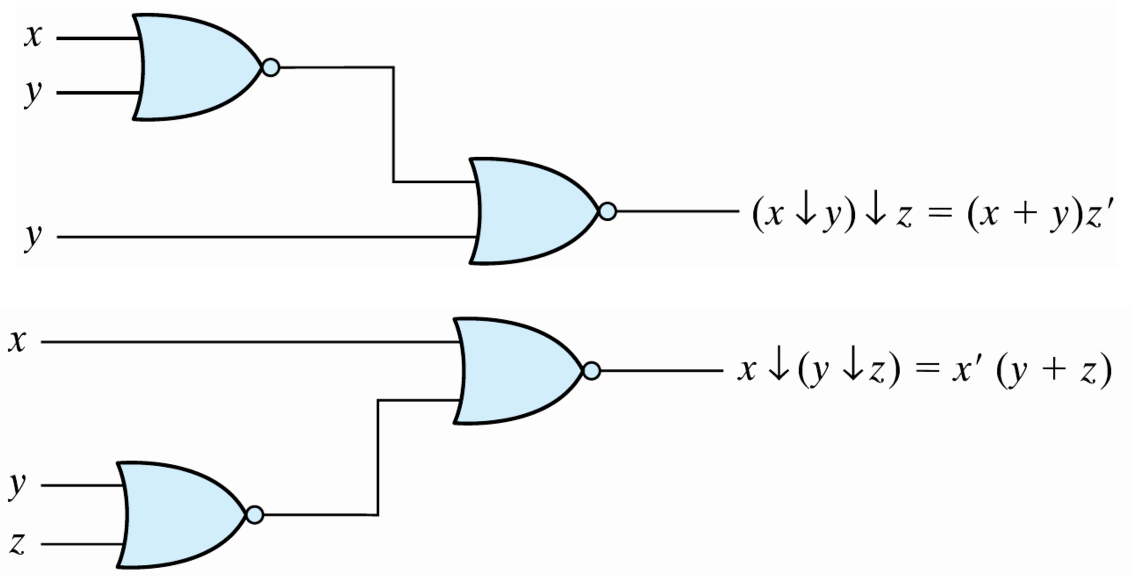


Figure 2.6 Demonstrating the nonassociativity of the NOR operator;
$(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

Multiple Inputs

- Multiple NOR = a complement of OR gate, Multiple NAND = a complement of AND.
- The cascaded NAND operations = sum of products.
- The cascaded NOR operations = product of sums.

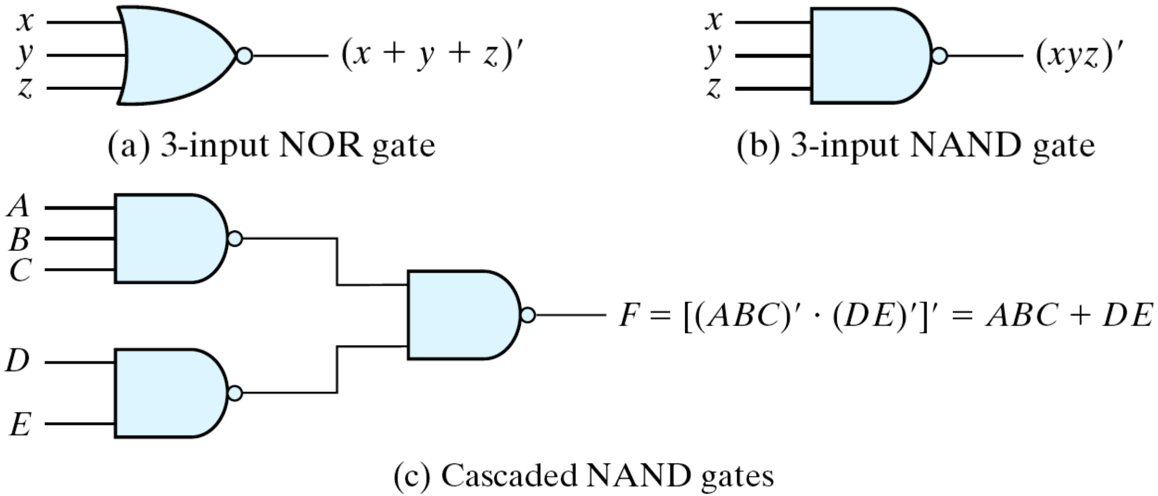
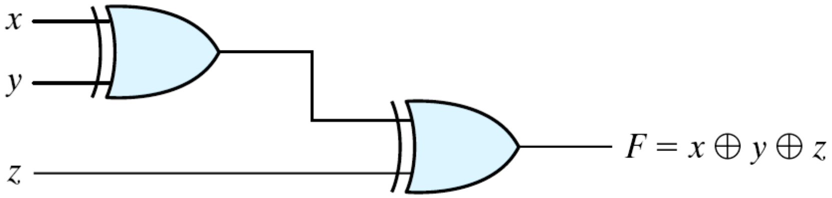


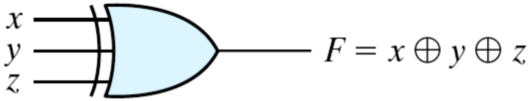
Figure 2.7 Multiple-input and cascated NOR and NAND gates

Multiple Inputs

- The XOR and XNOR gates are commutative and associative.
- Multiple-input XOR gates are uncommon?
- XOR is an odd function: it is equal to 1 if the inputs variables have an odd number of 1's.



(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

Figure 2.8 3-input XOR gate

Positive and Negative Logic

- Positive and Negative Logic
  - Two signal values <=> two logic values
  - Positive logic: H=1; L=0
  - Negative logic: H=0; L=1
- Consider a TTL gate
  - A positive logic AND gate
  - A negative logic OR gate
  - The positive logic is used in this book

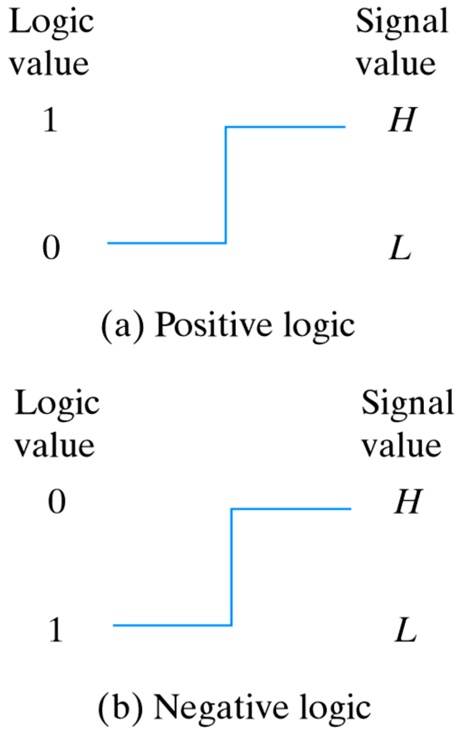
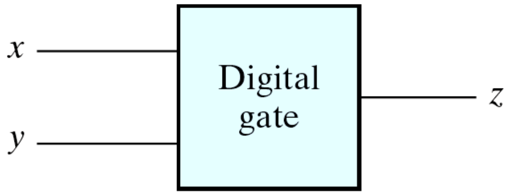


Figure 2.9 Signal assignment and logic polarity

Positive and Negative Logic

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

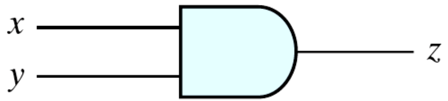
(a) Truth table with H and L



(b) Gate block diagram

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

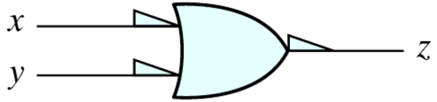
(c) Truth table for positive logic



(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

Figure 2.10 Demonstration of positive and negative logic

2.9 Integrated Circuits

- Level of Integration
- An IC (a chip)
- Examples:
  - Small-scale Integration (SSI): < 10 gates
  - Medium-scale Integration (MSI): 10 ~ 100 gates
  - Large-scale Integration (LSI): 100 ~ xk gates
  - Very Large-scale Integration (VLSI): > xk gates
- VLSI
  - Small size (compact size)
  - Low cost
  - Low power consumption
  - High reliability
  - High speed

Digital Logic Families

- Digital logic families: circuit technology
    - TTL: transistor-transistor logic (dying?)
    - ECL: emitter-coupled logic (high speed, high power consumption)
    - MOS: metal-oxide semiconductor (NMOS, high density)
    - CMOS: complementary MOS (low power)
    - BiCMOS: high speed, high density

Digital Logic Families

- The characteristics of digital logic families
    - Fan-out: the number of standard loads that the output of a typical gate can drive.
    - Power dissipation.
    - Propagation delay: the average transition delay time for the signal to propagate from input to output.
    - Noise margin: the minimum of external noise voltage that caused an undesirable change in the circuit output.

CAD

- CAD – Computer-Aided Design
    - Millions of transistors
    - Computer-based representation and aid
    - Automatic the design process
    - Design entry
        - Schematic capture
        - HDL – Hardware Description Language
            - Verilog, VHDL
    - Simulation
    - Physical realization
        - ASIC, FPGA, PLD

Chip Design

- Why is it better to have more gates on a single chip?
    - Easier to build systems
    - Lower power consumption
    - Higher clock frequencies

- What are the drawbacks of large circuits?
    - Complex to design
    - Chips have design constraints
    - Hard to test

- Need tools to help develop integrated circuits
    - Computer Aided Design (CAD) tools
    - Automate tedious steps of design process
    - Hardware description language (HDL) describe circuits
    - VHDL (see the lab) is one such system