DIGITAL LOGIC

Chapter 3 part2:
Two Level Implementation

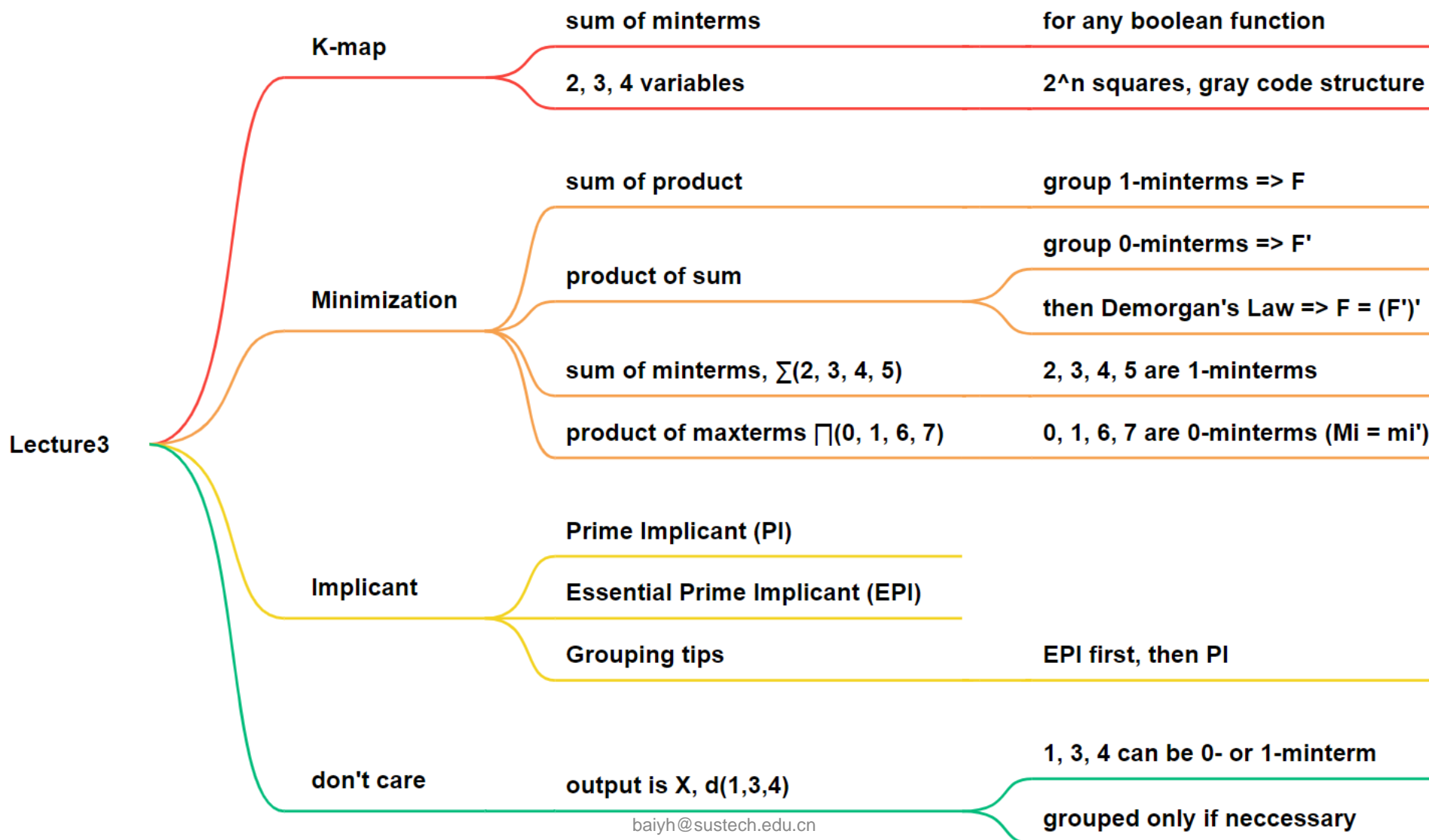2024 Fall

baiyh@sustech.edu.cn

Today's Agenda

- Recap
- Context
  - NAND and NOR Implementation
  - Other Two-Level Implementations
  - Exclusive-OR Function
- Reading: Textbook, Chapter 3.6-3.9

Recap

Lecture3

K-map
- sum of minterms — for any boolean function
- 2, 3, 4 variables — 2^n squares, gray code structure

Minimization
- sum of product — group 1-minterms => F
- product of sum
  - group 0-minterms => F'
  - then Demorgan's Law => F = (F')'
- sum of minterms, $\sum(2, 3, 4, 5)$ — 2, 3, 4, 5 are 1-minterms
- product of maxterms $\prod(0, 1, 6, 7)$ — 0, 1, 6, 7 are 0-minterms ($M_i = m_i'$)

Implicant
- Prime Implicant (PI)
- Essential Prime Implicant (EPI)
- Grouping tips — EPI first, then PI

don't care
- output is X, d(1,3,4)
  - 1, 3, 4 can be 0- or 1-minterm
  - grouped only if neccessary

baiyh@sustech.edu.cn

Recall: 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 $F = x$
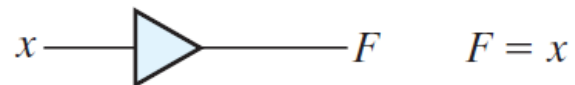
x	F
0	0
1	1

Recall: Logic Gates

NAND	x ——┐ y ——┘ ▷o —F	$F = (xy)'$	x	y	F
			0	0	1
			0	1	1
			1	0	1
			1	1	0
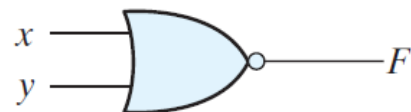
NOR	x —— y —— ▷o —F	$F = (x + y)'$	x	y	F
			0	0	1
			0	1	0
			1	0	0
			1	1	0

Exclusive-OR (XOR)	x —— y —— ▷ —F	$F = xy' + x'y$ $= x \oplus y$	x	y	F
			0	0	0
			0	1	1
			1	0	1
			1	1	0

Exclusive-NOR or equivalence	x —— y —— ▷o —F	$F = xy + x'y'$ $= (x \oplus y)'$	x	y	F
			0	0	1
			0	1	0
			1	0	0
			1	1	1
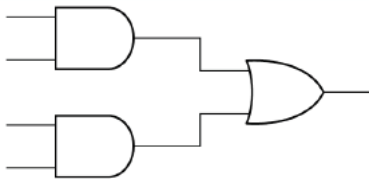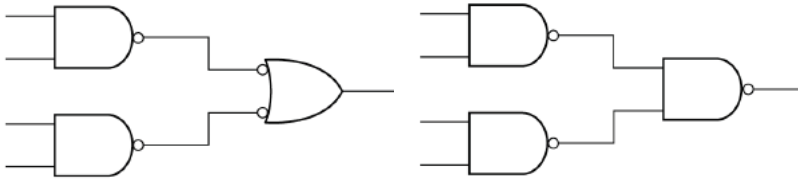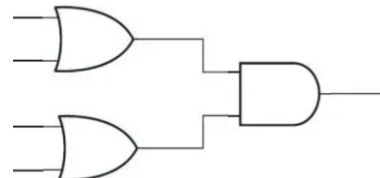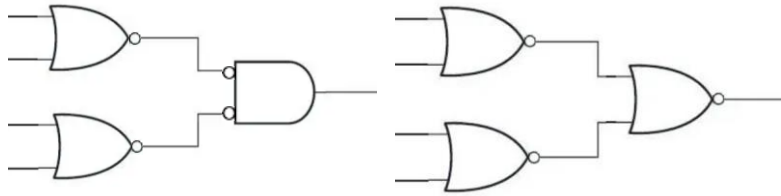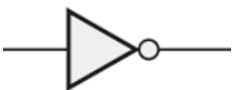
baiyh@sustech.edu.cn

Outline

- **NAND Implementation**
- NOR Implementation
- Exclusive-OR Function
- Other Two-Level Implementations

Universal Gates

- NAND gates and NOR gates are called universal gates or universal building blocks.
  - Any type of gates or logic functions can be implemented by these gates.
  - NAND and NOR gates are easier to fabricate thus are frequently used.

	Standard form	Universal Gate implementation
Sum-of-products	AND-OR	NAND-NAND
Product-of-sums	OR-AND	NOR-NOR

NAND circuits

- Inverter

$F = A' = \text{(AA)'}$
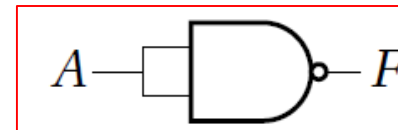
- AND

$F = AB = ((AB)')' = \text{((AB)' (AB)')}$

- OR

$F = A+B = ((A+B)')' = \text{(A'B')'}$

- XOR
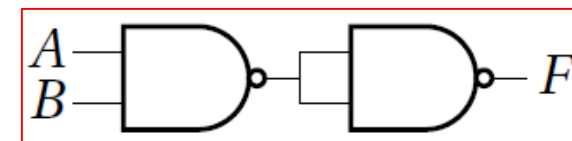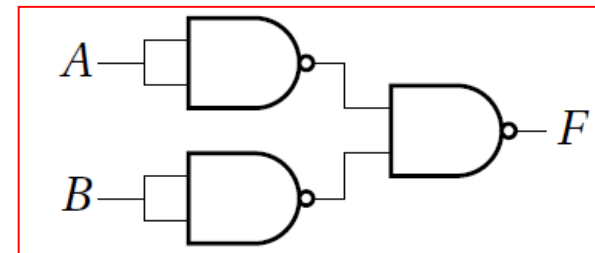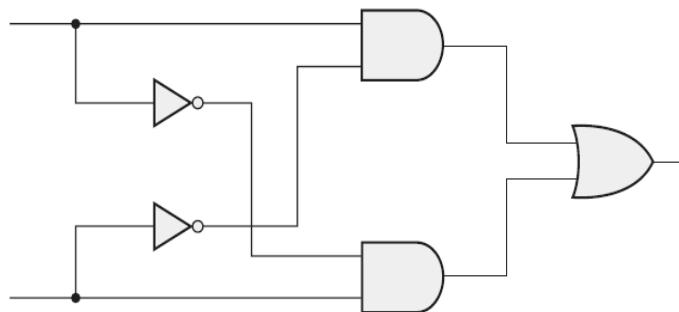
$F = AB'+A'B = AB'+A'B+AA'+BB'$
$\quad = (AB'+AA') + (A'B+BB') = A(A'+B') + B(A'+B')$
$\quad = ((A(AB)' + B(AB)')')' = \text{((A(AB)')'(B(AB)')')'}$
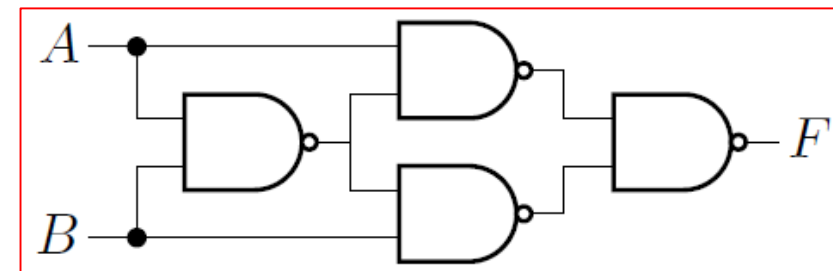
NAND circuits

- To facilitate the conversion to NAND logic, it is convenient to define an alternative graphic symbol for the gate.

- AND-invert and Invert-OR are both NAND gates

(a) AND-invert: inputs x, y, z produce output $(xyz)'$

(b) Invert-OR: inputs x, y, z produce output $x' + y' + z' = (xyz)'$
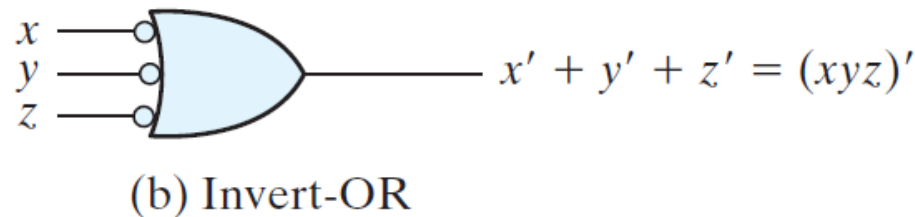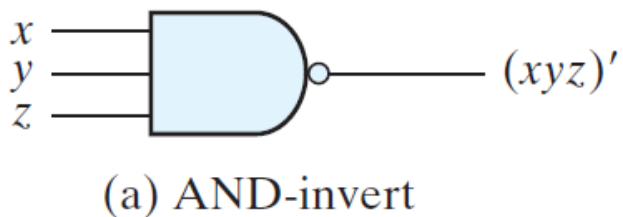
NAND-NAND Implementation

- A Boolean function can be implemented with two-levels of NAND gates
  1. Starting point → Simplify the function in the form of two-level **sum-of-products** (AND-OR circuit).
  2. Transfer it to two-level NAND-NAND expression.
     - algebraically (**DeMorgan's Law**)
     - or graphically (**Bubble pushing**)
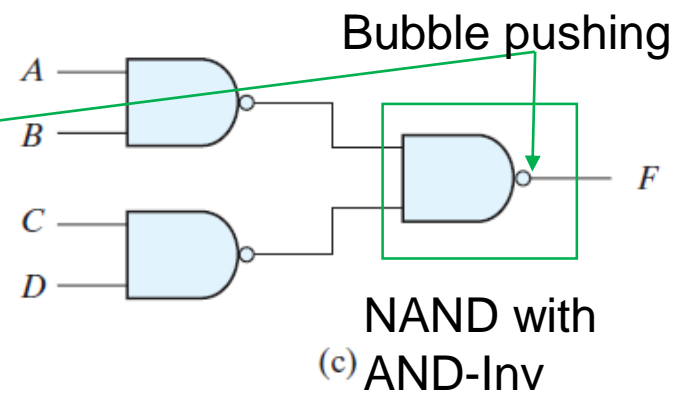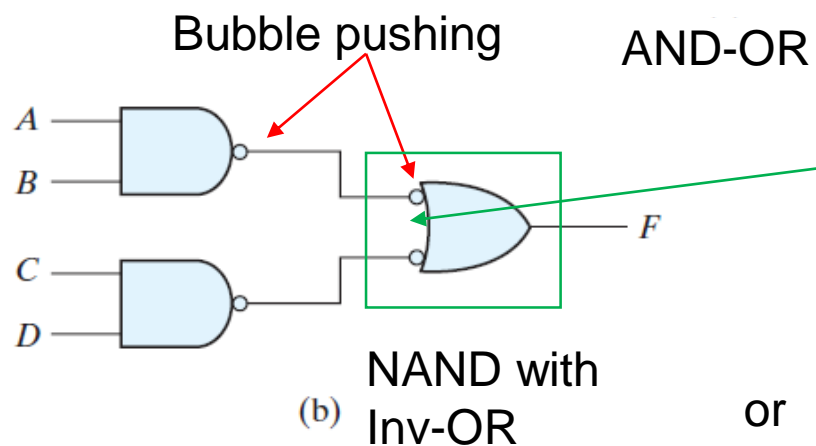 3. Draw the corresponding NAND gate implementation. A 1-input NAND gate can be replaced by an inverter.
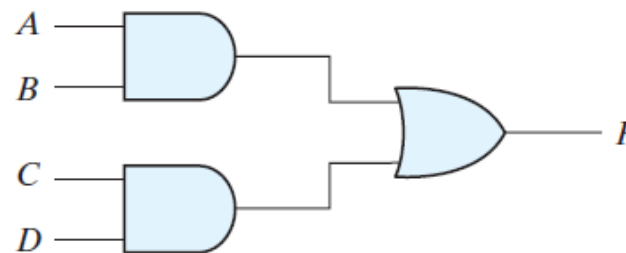
- F(A,B,C,D) = AB + CD
  - Starting point: sum of products form

- algebraic method (DeMorgan's)

F = AB+CD
  = ((AB+CD)')'
  = ((AB)' (CD)')'

- graphic method (Bubble pushing)

AND-OR

Bubble pushing

Bubble pushing

NAND with Inv-OR
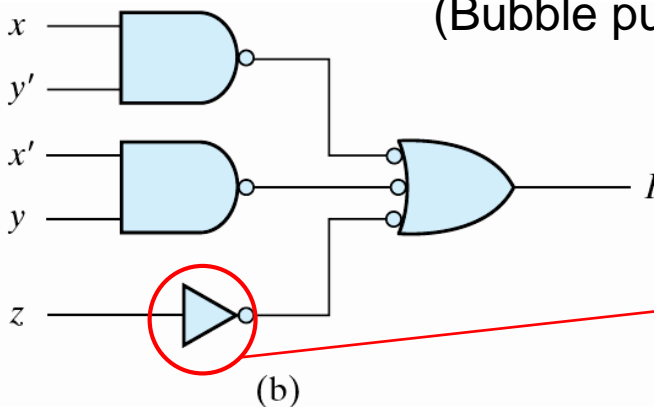
or

NAND with AND-Inv

NAND-NAND Example2

- Example: Implement the following Boolean function with NAND gates
  - $F(x,y,z) = \sum(1,2,3,4,5,7)$
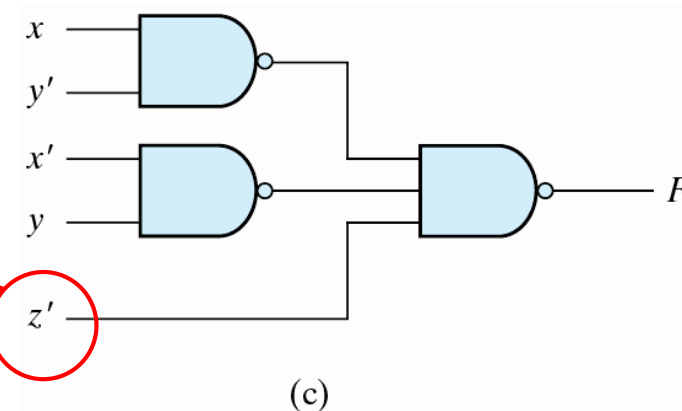 - Starting point: K-map simplification into sum of product form

(a)
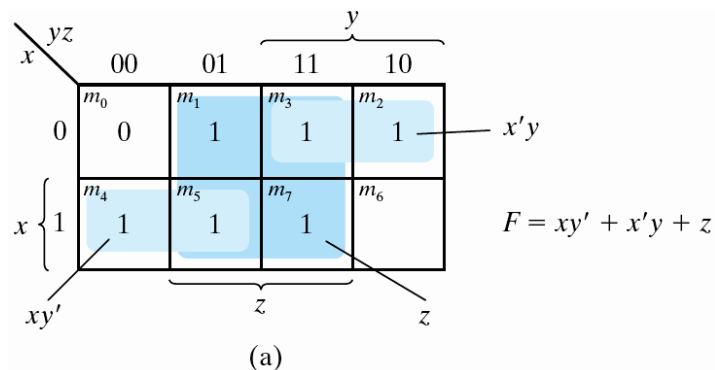
- algebraic method (DeMorgan's)

$F = xy' + x'y + z$
$\quad = ((xy' + x'y + z)')'$
$\quad = ((xy')' (x'y)' z')'$

- graphic method (Bubble pushing)

(b)

or

(c)

NAND-NAND Example3

- Exercise: Implement the following Boolean function with NAND gates
- F(A,B,C,D) = A'B'C'D+CD+AC'D

NAND-NAND Example3

- F(A,B,C,D) = A'B'C'D+CD+AC'D
  = A'B'C'D+(A+A')(B+B')CD+A(B+B')C'D
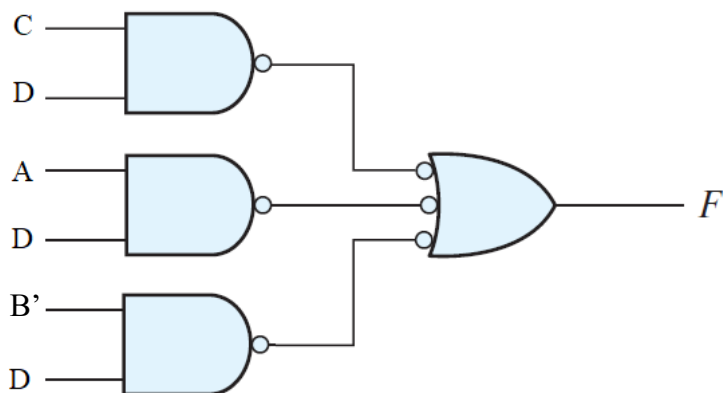 =A'B'C'D+ABCD+AB'CD+A'BCD+A'B'CD+ABC'D+AB'C'D
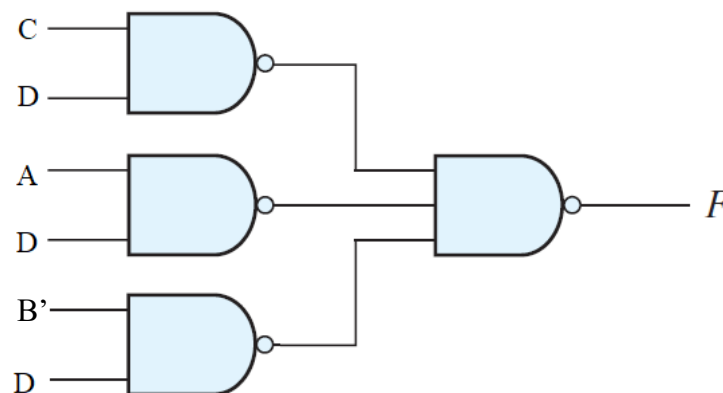 =∑(1,3,7,9,11,13,15)

- Algebraic method
  F = CD+AD+B'D
  =[(CD+AD+A'D)']' = [(CD)'(AD)'(B'D)']'
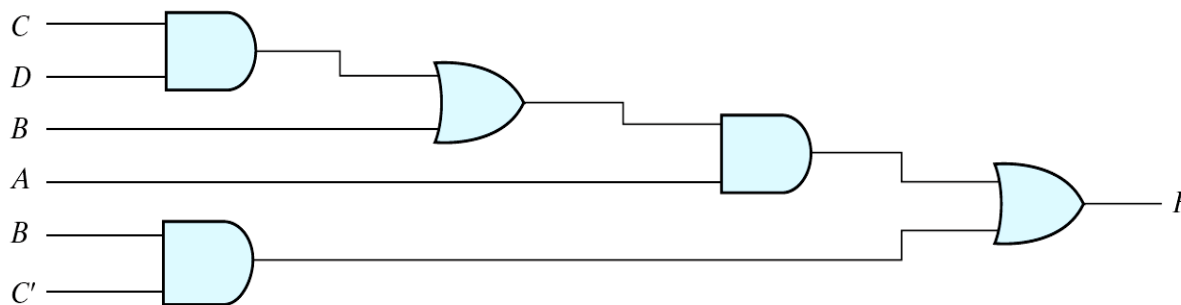

- Graphic method

or

Multilevel NAND Implementation

- Multilevel-NAND circuits conversion procedure
  - Convert all AND to NAND with AND-Invert graphic symbols
  - Convert all OR to NAND with Invert-OR graphic symbols
  - Check all the bubbles (inverter) in the diagram and insert possible inverter to keep the original function

- Example: F(A,B,C,D) = A(CD+B)+BC'
  - AND-OR logic → NAND-NAND logic
    - For every bubble that is not compensated by another small circle along the same line, insert an inverter.

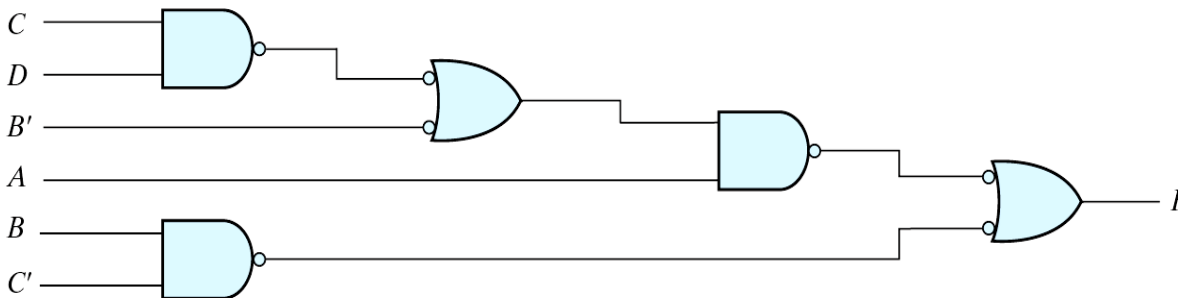AND → AND + inverter

OR: inverter + OR = NAND

(a) AND–OR gates

baiyh@sustech.edu.cn

Multilevel NAND Implementation

- Multilevel-NAND circuits conversion procedure
  - Convert all AND to NAND with AND-Invert graphic symbols
  - Convert all OR to NAND with Invert-OR graphic symbols
  - Check all the bubbles (inverter) in the diagram and insert possible inverter to keep the original function

- Example: F(A,B,C,D) = A(CD+B)+BC'
  - AND-OR logic → NAND-NAND logic
    - For every bubble that is not compensated by another small circle along the same line, insert an inverter.
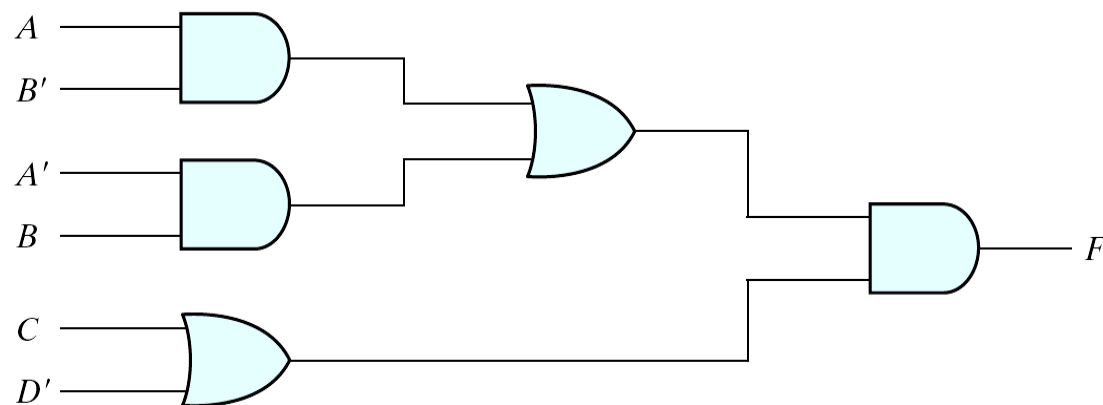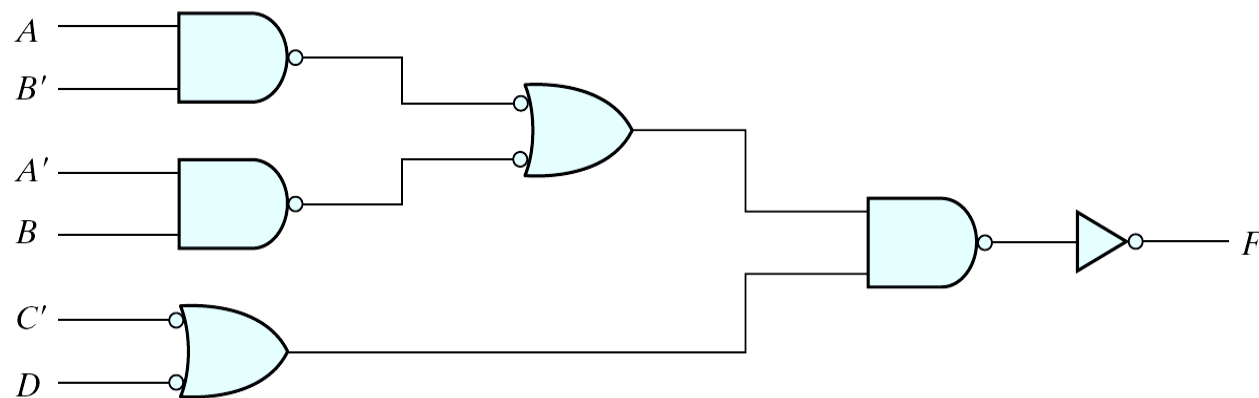
AND → AND + inverter

OR: inverter + OR = NAND

(b) NAND gates

Multilevel NAND Implementation

• Exercise: Implementing F = (AB′ +A′B)(C+ D′)
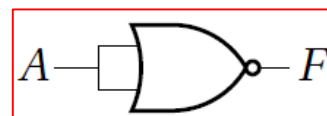

(a) AND–OR gates

(b) NAND gates

Outline

- NAND Implementation
- **NOR Implementation**
- Exclusive-OR Function
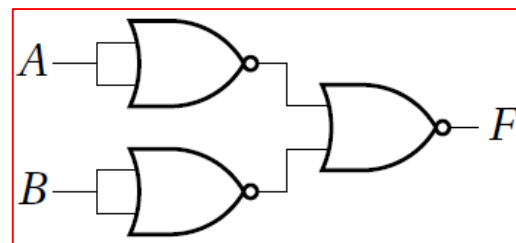- Other Two-Level Implementations
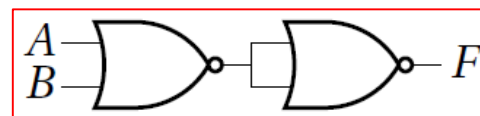
NOR circuits

- Inverter

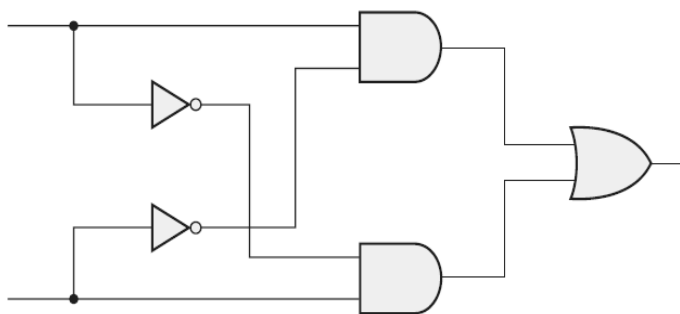 $F = A' = (A+A)'$

- AND

  $F = AB = ((AB)')' = (A'+B')'$

- OR

  $F=A+B=((A+B)')'$

- XOR

  $F = AB'+A'B = \dots$

NOR circuits
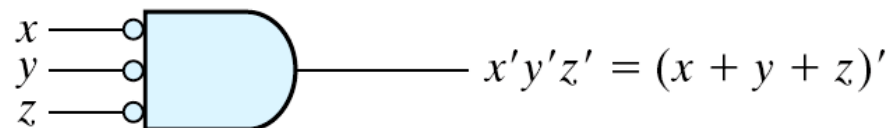
- To facilitate the conversion to NOR logic, it is convenient to define an alternative graphic symbol for the gate.
- NOR-NOR is the dual of the NAND-NAND implementation
  - All procedures and rules for NOR logic are the duals of the corresponding which developed for NAND logic.

$$x, y, z \rightarrow (x + y + z)'$$

(a) OR-invert

$$x, y, z \rightarrow x'y'z' = (x + y + z)'$$
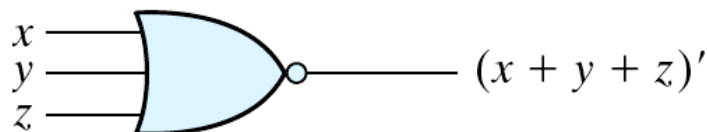
(b) Invert-AND

NOR-NOR Implementation

- Procedure of NOR-NOR implementation
  - Starting point → Simplify the function in the form of **product-of-sum** (OR-AND circuit).
  - Transfer it to 2-level NOR-NOR expression.
    - algebraically (**DeMorgan's Law**)
    - or, graphically (**Bubble pushing**)
  - Draw the corresponding NOR gate implementation. A 1-input NOR gate can be replaced by an inverter.

> sum-of-product (AND-OR) => NAND-NAND
> product-of-sum (OR-AND) => NOR-NOR

NOR-NOR Example1

- Example: Implement the following Boolean function with NOR gates
- F = (A + B)(C + D)E
  - Starting point: product of sums form → done
- Algebraic method
  - F = (A+B)(C+D)E = ((A+B)(C+D)E)')'
    = ((A+B)'+(C+D)'+E')'→DeMorgan's
- Graphic method:
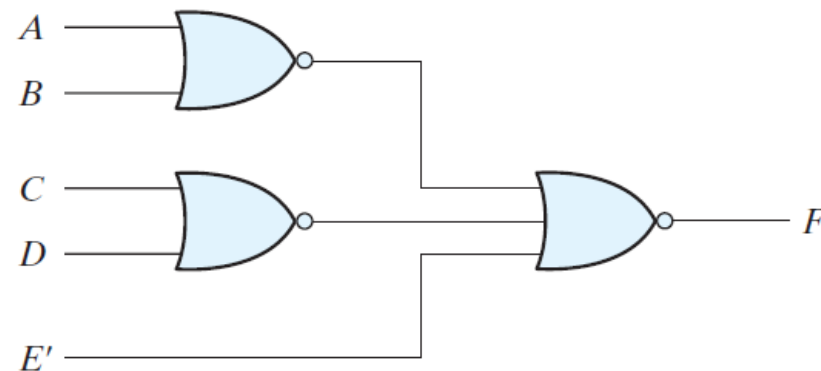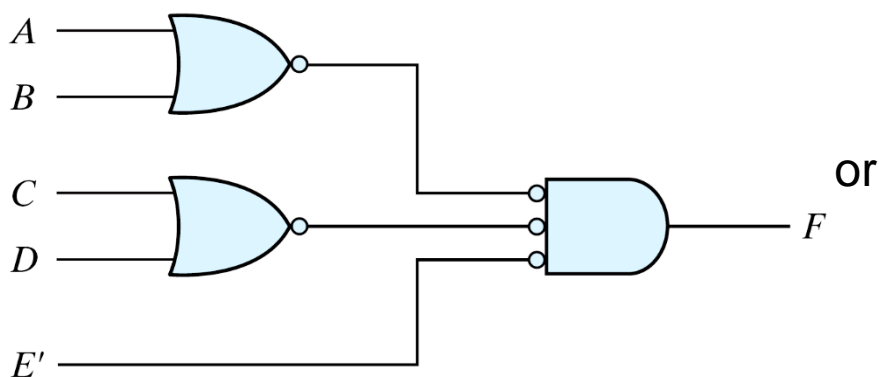
NOR-NOR Example2

- Example F(x,y,z) = ∑(1,2,3,4,5,7)

- Algebraic method

  F'=x'y'z' + xyz'

  F = (F')' = (x'y'z' + xyz')' = (((x'y'z)')' + ((xyz')')')'

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

- Graphic method

  - F'=x'y'z' + xyz'

  - F = (x+y+z)(x'+y'+z) (starting point for bubble pushing)

	y		
yz			

	00	01	11	10
0	m_0 0	m_1 1	m_3 1	m_2 1
1	m_4 1	m_5 1	m_7 1	m_6 0

or

NOR with Inv-AND

NOR with OR-Inv

baiyh@sustech.edu.cn
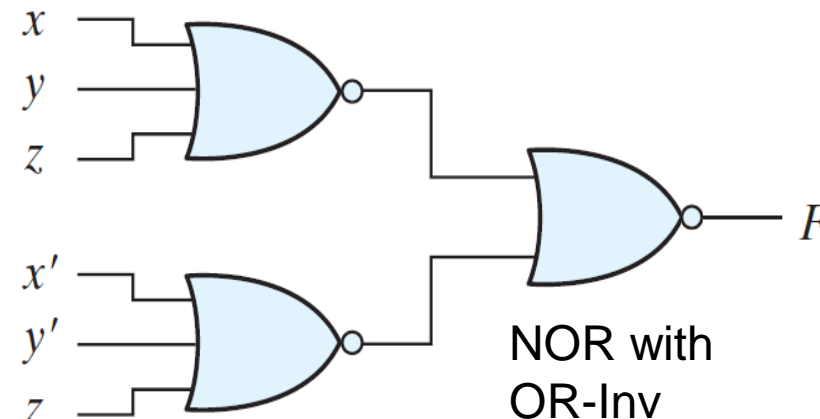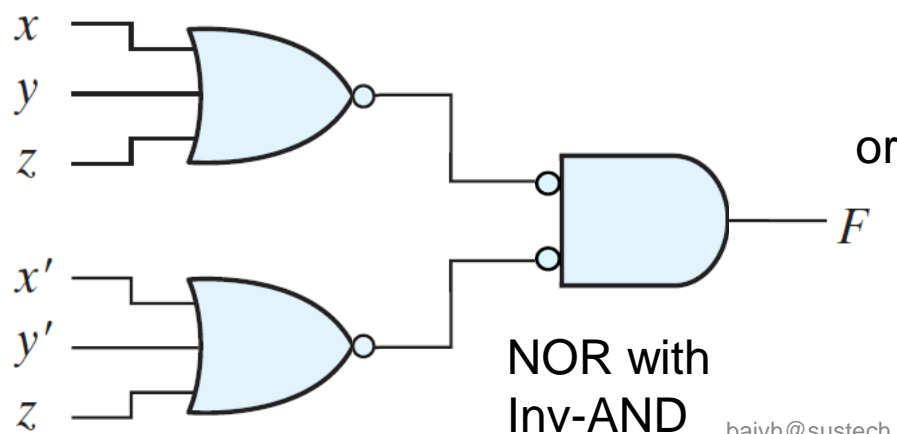
Multilevel NOR Implementation

- Change the OR gates to NOR gates with OR-invert graphic symbols and the AND gate to a NOR gate with an invert-AND graphic symbol.
- Example:
- F = (AB' +A'B)(C + D')
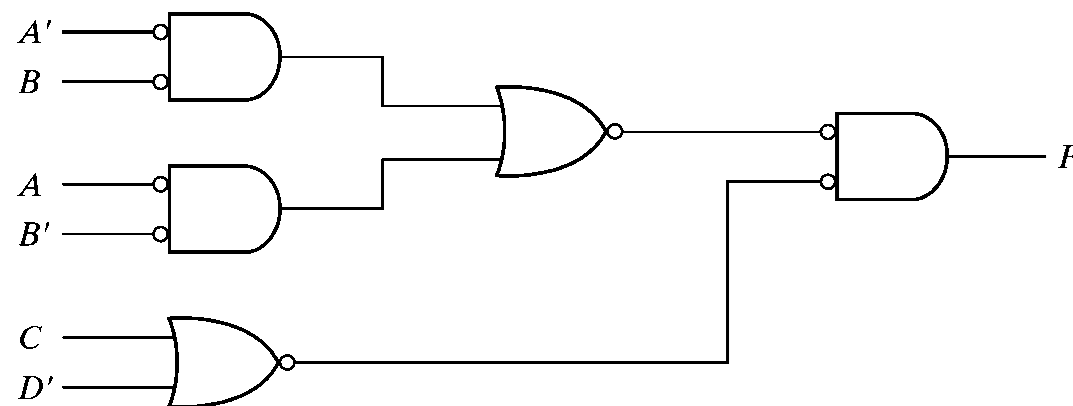
- F =(AB+E)(C+D)

Outline

- NAND Implementation
- NOR Implementation
- **Exclusive-OR Function**
- Other Two-Level Implementations

Exclusive-OR Function

- Exclusive-OR (XOR)
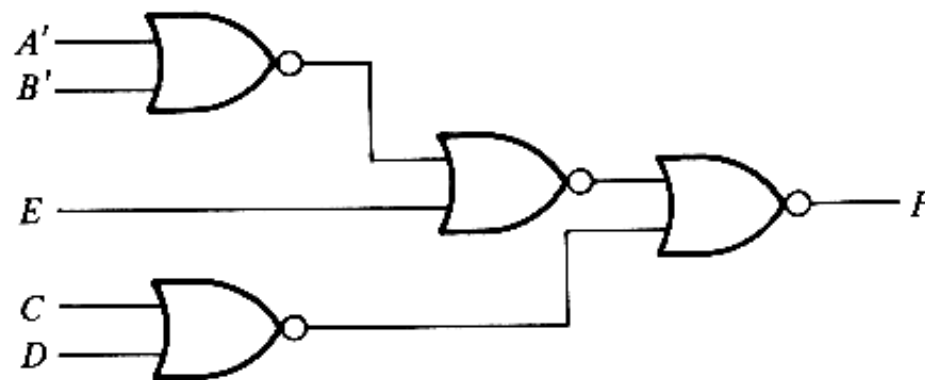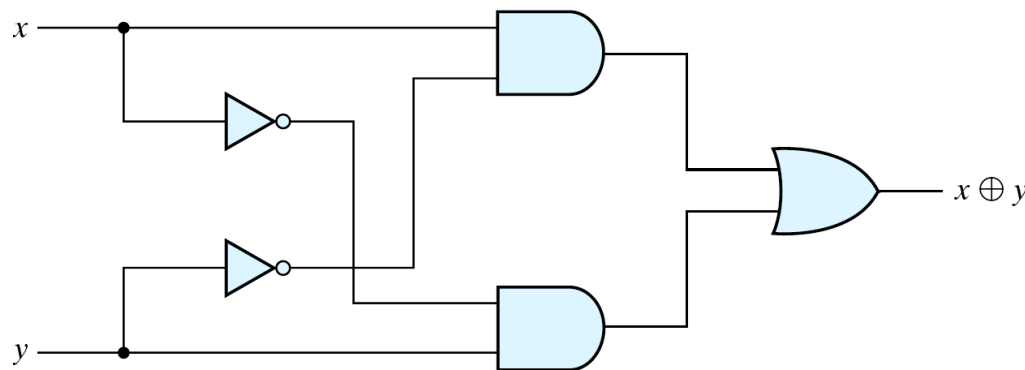 - $x \oplus y = xy' + x'y$
- Exclusive-NOR (XNOR or equivalency)
  - $(x \oplus y)' = xy + x'y'$
- Some identities
  - $x \oplus 0 = x$
  - $x \oplus 1 = x'$
  - $x \oplus x = 0$
  - $x \oplus x' = 1$
  - $x \oplus y' = x' \oplus y = (x \oplus y)'$
- Commutative and associative
  - $A \oplus B = B \oplus A$
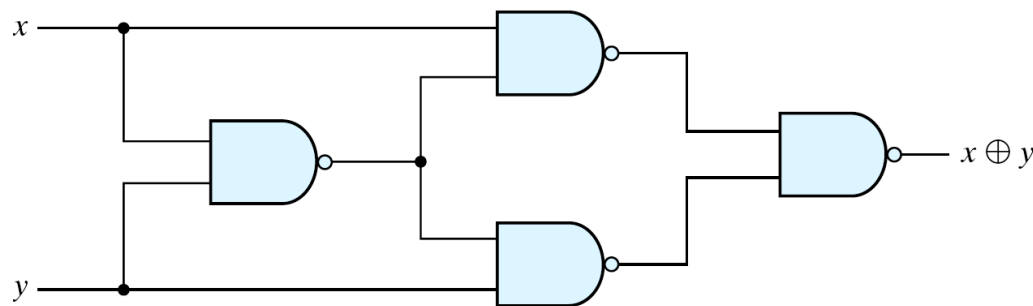 - $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

Exclusive-OR Implementations

- Implementations
 - (x'+y')x + (x'+y')y = xy'+x'y = x⊕y

(a) With AND–OR–NOT gates

(b) With NAND gates

Odd function

- The XOR operation with three or more variables can be converted into an ordinary Boolean function by replacing the ⊕ with its equivalent Boolean expression.

- $A \oplus B \oplus C$ = (AB'+A'B)C'+(AB+A'B')C

    =AB'C'+A'BC'+ABC+A'B'C

    = $\sum$(1, 2, 4, 7)

  - Odd function (XOR) → if **odd** number of variables are equal to 1, then F = 1.
  - Even function (XNOR) → if **even** number of variables are equal to 1, then F = 1.
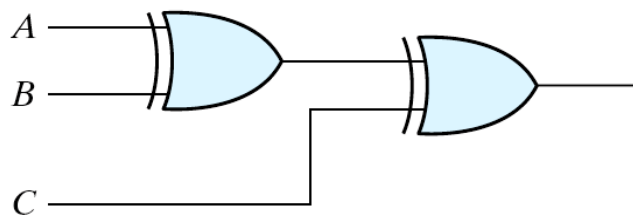
(a) 3-input odd function
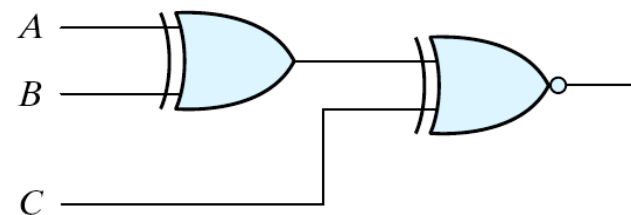
(b) 3-input even function

Recall: Error-Detecting Code

- Error-Detecting Code
  - An <u>eighth bit</u> is sometimes added to the ASCII character to indicate its parity.
 - A parity bit (校验位) is an extra bit included with a message to make the total number of 1's either even or odd.

- Example:

	With even parity	With odd parity
ASCII A = 1000001	01000001	11000001

Even-Parity-Generator Truth Table

Three-Bit Message			Parity Bit
x	y	z	P
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

XOR functions can be used for parity generator and parity checker

x →
y → Even Parity Generator → P
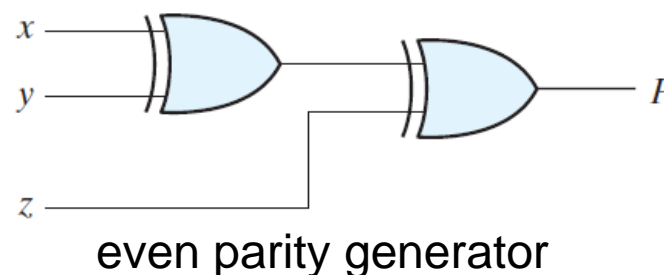z →

baiyh@sustech.edu.cn
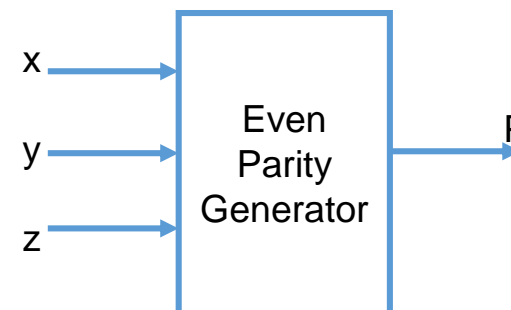
Parity Generation and Checking

- Parity Generation circuit and Checking circuit
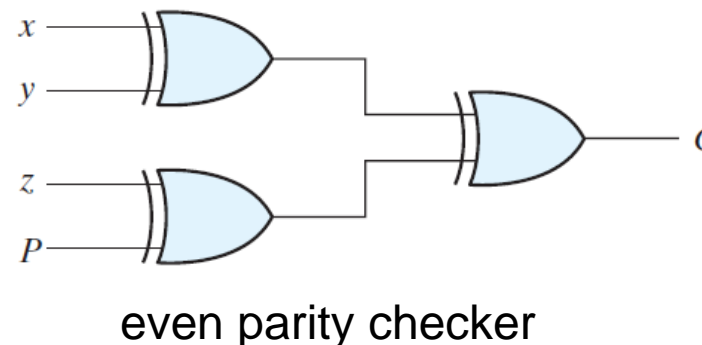- Generator produces an even parity bit with: P = $x \oplus y \oplus z$
  - P = xy'z'+x'yz'+xyz+x'y'z
    = $\sum$(1, 2, 4, 7) – odd function

even parity generator

x	y	z	Parity bit p
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

- Checker does even parity check with: C = $x \oplus y \oplus z \oplus P$
  - C=1: one bit error or an odd number of data bit error
  - C=0: correct or an even # of data bit error

even parity checker

Outline

- NAND Implementation
- NOR Implementation
- Exclusive-OR Function
- **Other Two-Level Implementations**

Two-Level Forms

- AND/NAND/OR/NOR have 16 possible combinations of two-level forms
- Eight of them **degenerate** to a single operation
  - AND-AND => AND
  - OR-OR => OR
  - AND-NAND => NAND
  - OR-NOR => NOR
  - NAND-NOR =>AND
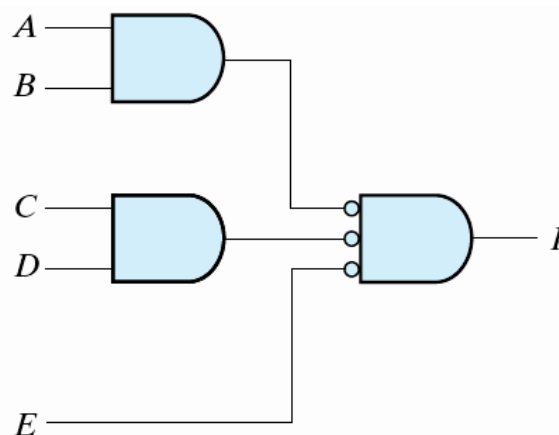 - NOR-NAND => OR
  - NAND-OR => NAND
  - NOR-AND => NOR

Two-Level Forms

- Eight are **non-degenerate** forms
- AND-OR => standard sum-of-products
- NAND-NAND => standard sum-of-products
- OR-AND => standard product-of-sums
- NOR-NOR => standard product-of-sums
- NAND-AND/AND-NOR => AND-OR-INVERT (AOI)
  - **complement** of sum-of-products
- OR-NAND/NOR-OR => OR-AND-INVERT (OAI)
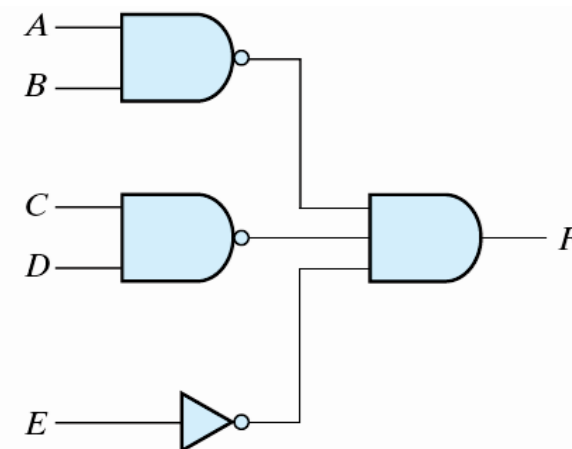 - **complement** of product-of-sums

AND-OR-Invert Implementation

- NAND-AND = AND-NOR = AOI
  - F(A,B,C,D,E)=(AB+CD+E)'
  - F'(A,B,C,D,E)=AB+CD+E (sum of products)
  - An AND–OR implementation requires an expression in sum-of-products form.
  - The AND–OR–INVERT implementation is similar, except for the inversion.
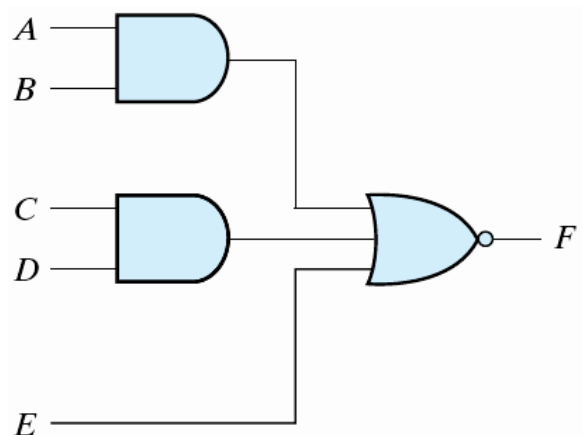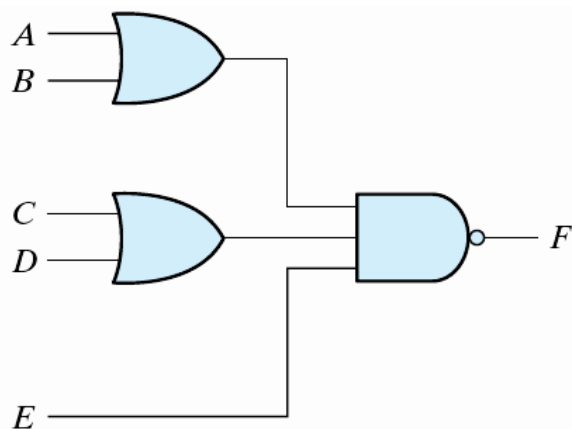

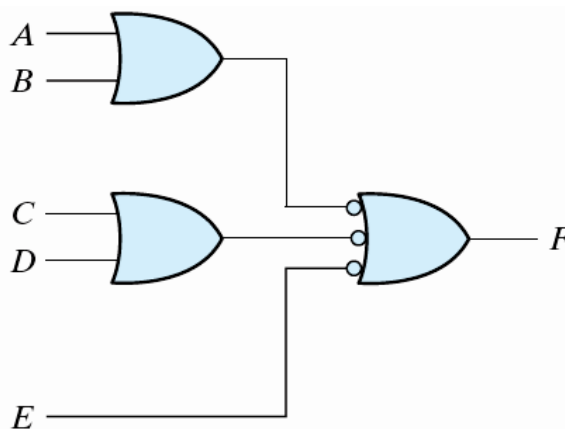(a) AND–NOR   (b) AND–NOR   (c) NAND–AND

Combine 0's in K-map to simplify F' in product-
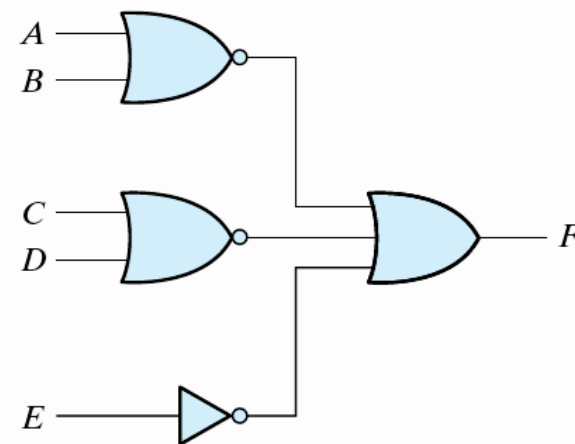of-sums and then invert the results

OR-AND-Invert Implementation

- OR-NAND = NOR-OR = OAI
  - F(A,B,C,D,E)=((A+B)(C+D)E)'
  - F' = (A+B)(C+D)E   (product of sums)
  - The AND–OR–INVERT implementation requires an expression in product-of-sums form.

(a) OR–NAND   (b) OR–NAND   (c) NOR–OR

Combine 1's in K-map to simplify F' in product-
of-sums and then invert the results

AOI & OAI Example

- Example

$F = x'y'z' + xyz'$
$F' = x'y + xy' + z$

- AND-OR
  - $F = x'y'z' + xyz'$
- NAND-NAND
  - $F = ((x'y'z')'(xyz')')$
- OR-AND
  - $F' = x'y + xy' + z \rightarrow F = z'(x'+y)(x+y')$
- NOR-NOR
  - $F' = x'y + xy' + z \rightarrow F = (z + (x'+y)' + (x+y')')'$
- AOI
  - $F' = x'y + xy' + z \rightarrow F = (x'y + xy' + z)'$
- OAI
  - $F = x'y'z' + xyz' \rightarrow F' = (x+y+z)(x'+y'+z) \rightarrow F = ((x+y+z)(x'+y'+z))'$

Summary

- Two and multi-level implementations:
  - Universal gates: NAND and NOR gates.
  - Procedure of two-level implementations using NAND or NOR gates.
  - Procedure of multi-level implementations using NAND or NOR gates.
- Exclusive-OR gate for error detection circuits.