

DIGITAL LOGIC

Chapter 3 part2: Two Level Implementation

2024 Fall

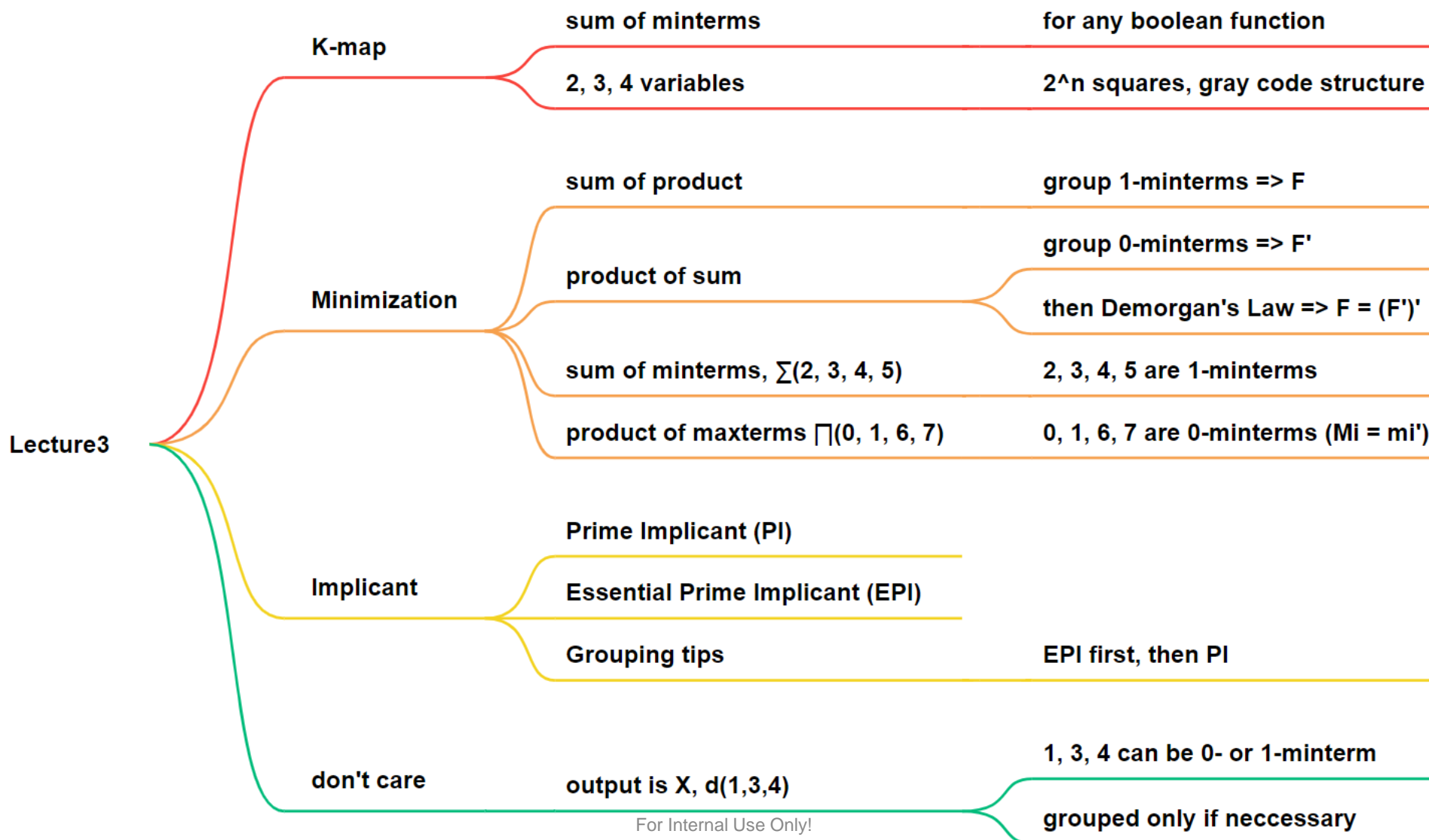
This PowerPoint is for internal use only at Southern University of Science and Technology.
Please do not repost it on other platforms without permission from the instructor.

Today's Agenda

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



Recap



Recall: Logic Gates

AND



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

OR



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

Inverter



x	F
0	1
1	0

Buffer



x	F
0	0
1	1

Recall: Logic Gates

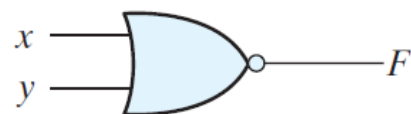
NAND



$$F = (xy)'$$

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



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

$$= (x \oplus y)'$$

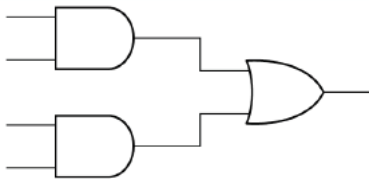
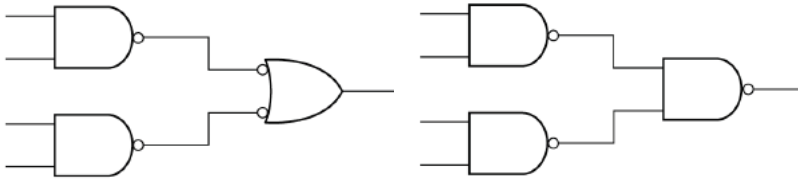
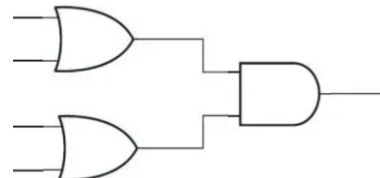
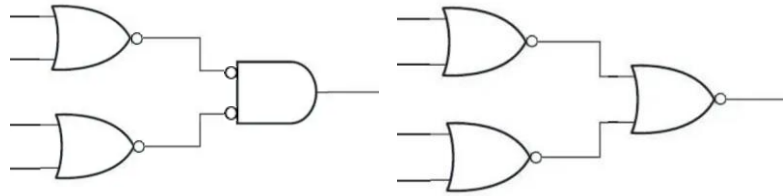
x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

Outline


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

Universal Gates

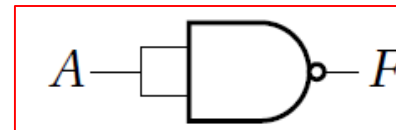
- NAND gates and NOR gates are called universal gates or universal building blocks.
 - Any type of gates or logic functions can be implemented by these gates.
 - NAND and NOR gates are easier to fabricate thus are frequently used.


	Standard form	Universal Gate implementation
Sum-of-products	<p>AND-OR</p> 	<p>NAND-NAND</p> 
Product-of-sums	<p>OR-AND</p> 	<p>NOR-NOR</p> 

NAND circuits

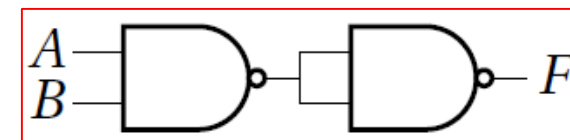
• Inverter 

$$F = A' = (AA)'$$



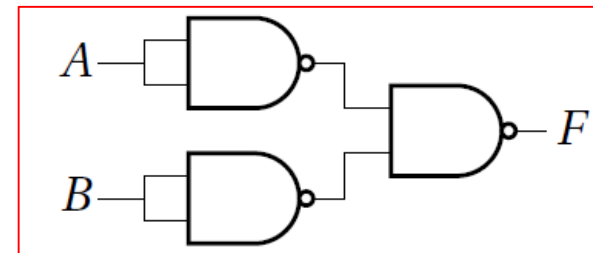
• AND 

$$F = AB = ((AB)')' = ((AB)' (AB)')$$

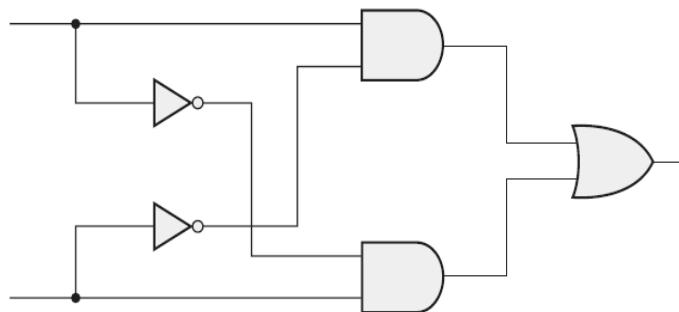


• OR 

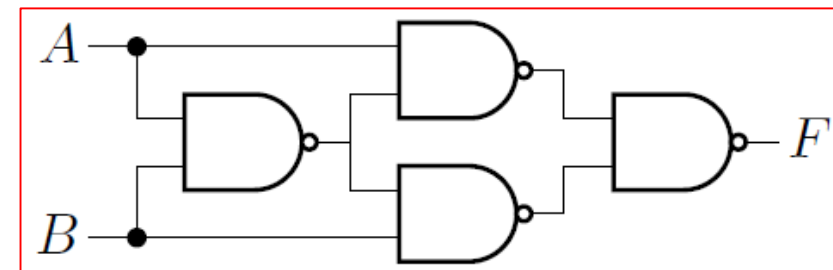
$$F = A + B = ((A + B)')' = (A'B')'$$



• XOR

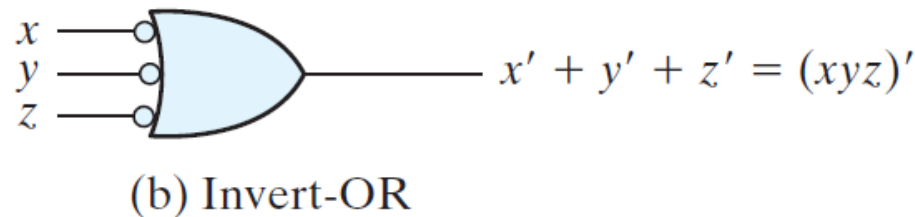
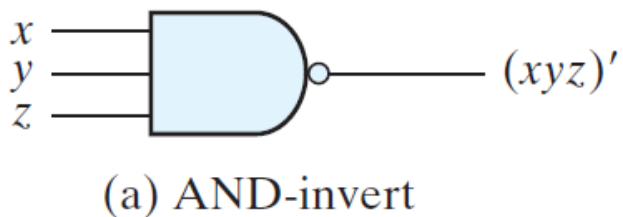


$$\begin{aligned} F &= AB' + A'B = AB' + A'B + AA' + BB' \\ &= (AB' + AA') + (A'B + BB') = A(A' + B') + B(A' + B') \\ &= ((A(AB)') + B(AB'))' = ((A(AB)')' (B(AB)'))' \end{aligned}$$



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



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.

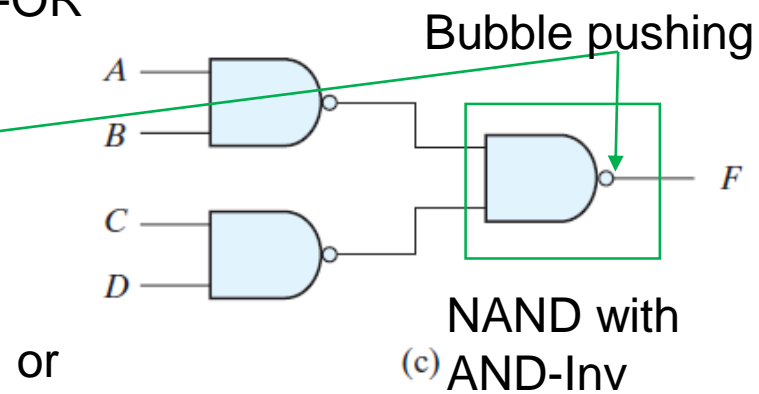
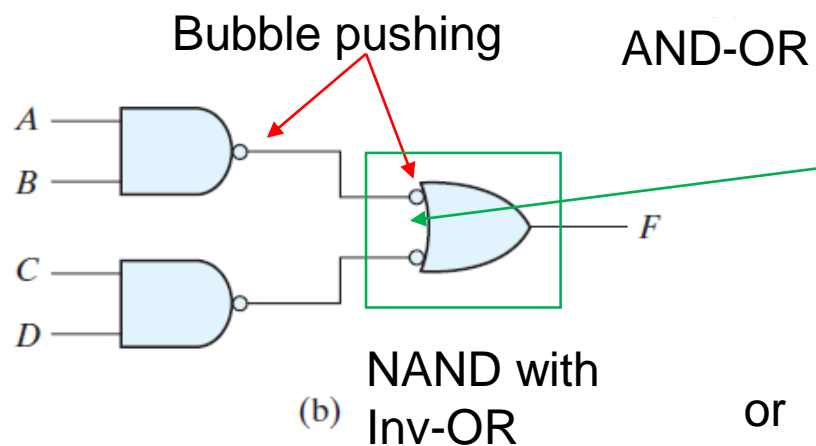
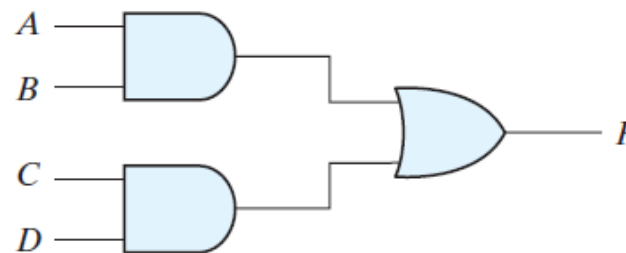
NAND-NAND Example1

- $F(A,B,C,D) = AB + CD$
 - Starting point: sum of products form

- algebraic method (DeMorgan's)

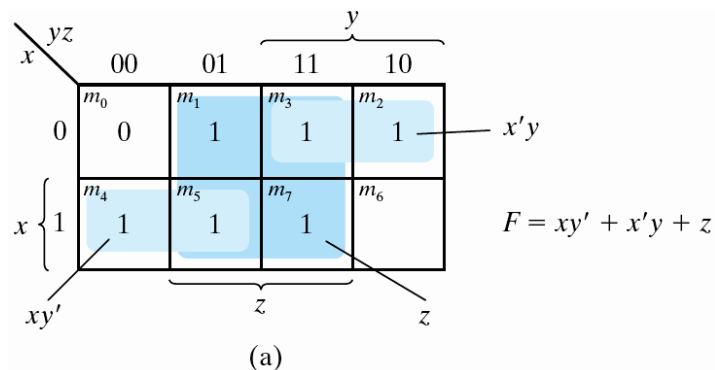
$$\begin{aligned}
 F &= AB + CD \\
 &= ((AB + CD)')' \\
 &= ((AB)' (CD)')'
 \end{aligned}$$

- graphic method (Bubble pushing)



NAND-NAND Example2

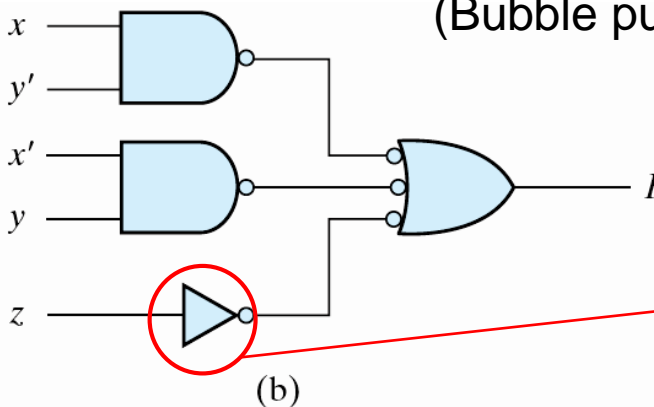
- Example: Implement the following Boolean function with NAND gates
 - $F(x,y,z) = \sum(1,2,3,4,5,7)$
 - Starting point: K-map simplification into sum of product form



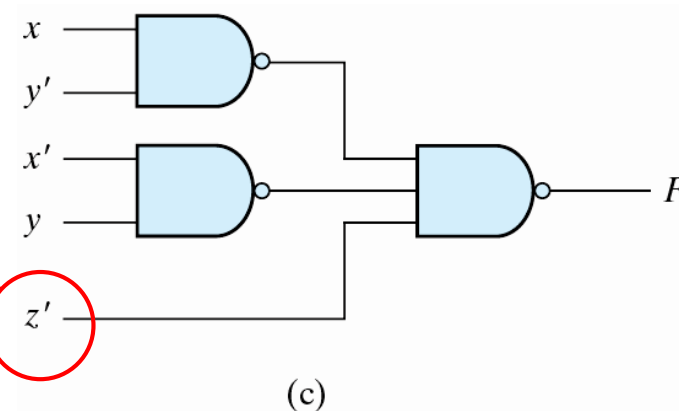
- algebraic method
(DeMorgan's)

$$\begin{aligned}
 F &= xy' + x'y + z \\
 &= ((xy' + x'y + z)')' \\
 &= ((xy')' (x'y)' z')'
 \end{aligned}$$

- graphic method
(Bubble pushing)



or



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

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

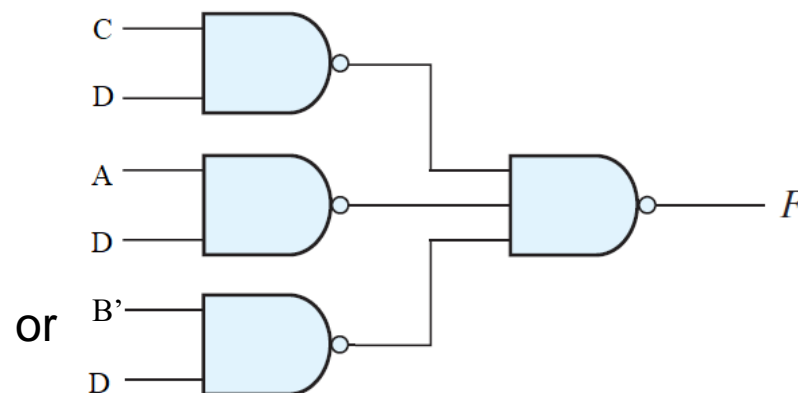
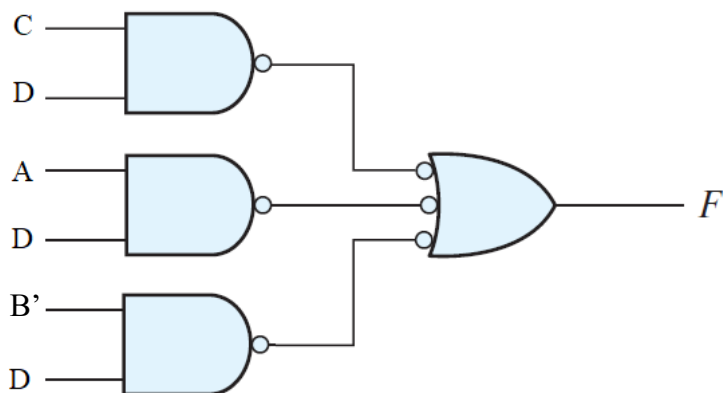
- Algebraic method

$$F = CD + AD + B'D$$

$$= [(CD + AD + A'D)']' = [(CD)'(AD)'(B'D)']'$$

- Graphic method

		C			
		CD			
		00	01	11	10
A	B	00	1	1	0
	01	0	0	1	0
	11	0	1	1	0
	10	0	1	1	0

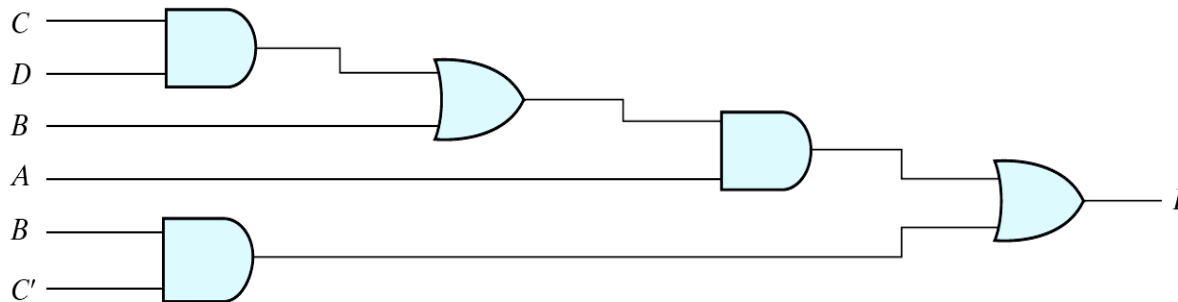


Multilevel NAND Implementation

- 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 \rightarrow NAND-NAND logic
 - For every bubble that is not compensated by another small circle along the same line, insert an inverter.

AND \rightarrow AND + inverter

OR: inverter + OR = NAND



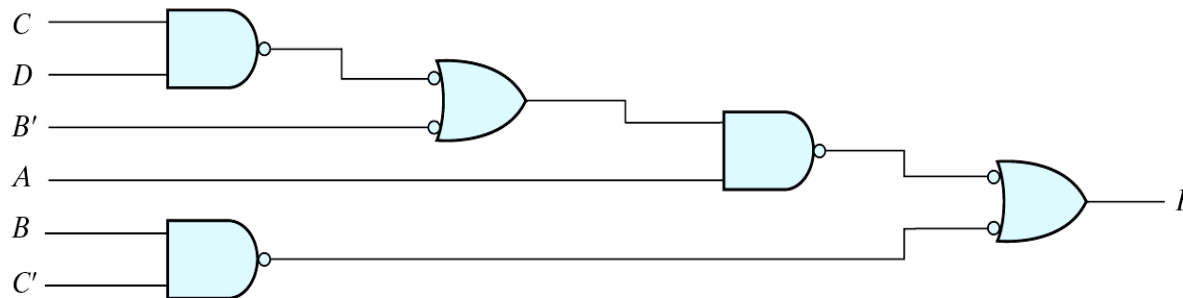
(a) AND-OR gates For Internal Use Only!

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 \rightarrow NAND-NAND logic
 - For every bubble that is not compensated by another small circle along the same line, insert an inverter.

AND \rightarrow AND + inverter

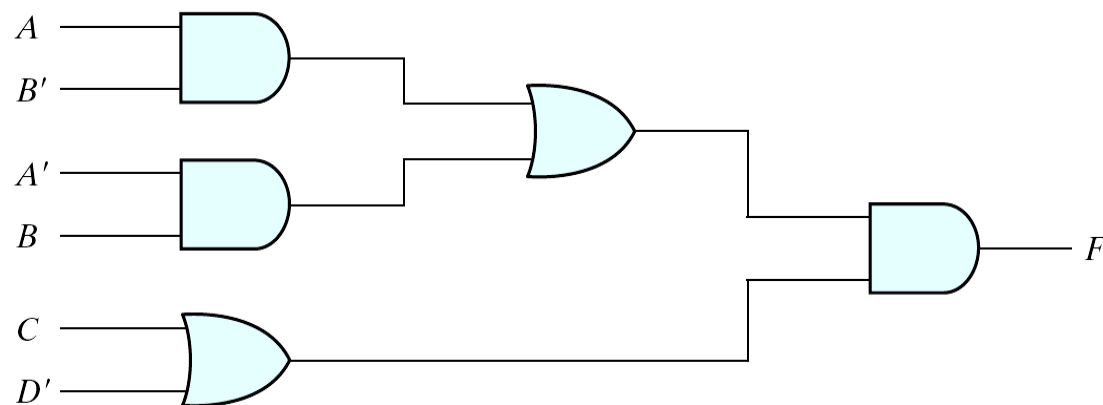
OR: inverter + OR = NAND



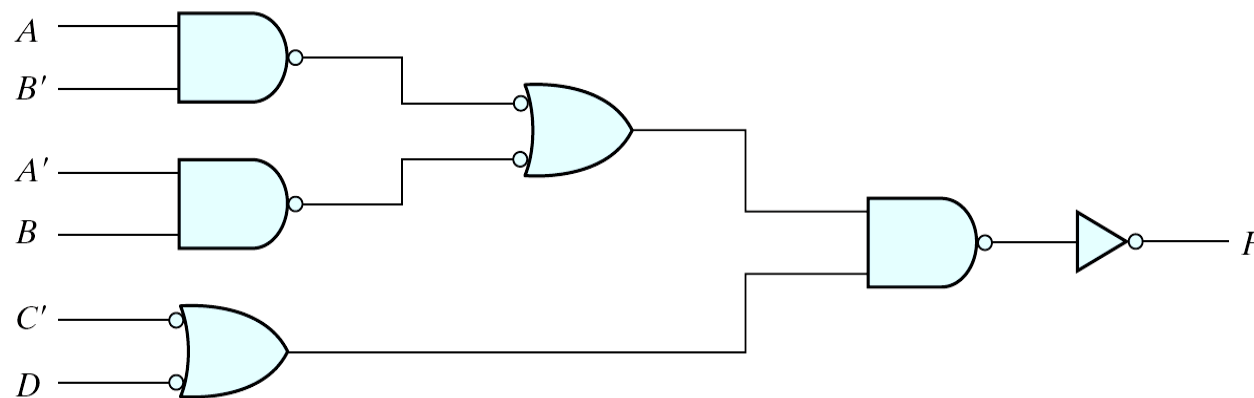
(b) NAND gates For Internal Use Only!

Multilevel NAND Implementation

- Exercise: Implementing $F = (AB' + A'B)(C + D')$



(a) AND-OR gates

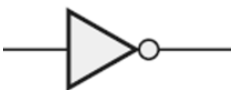


(b) NAND gates

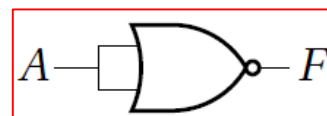
Outline


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

NOR circuits

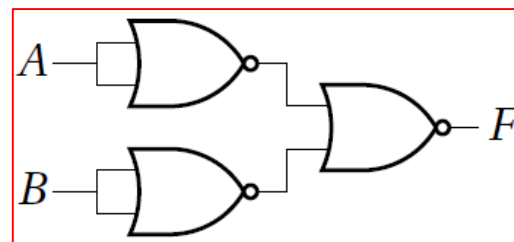
• Inverter 


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



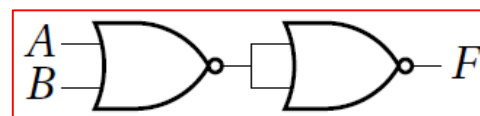
• AND 

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

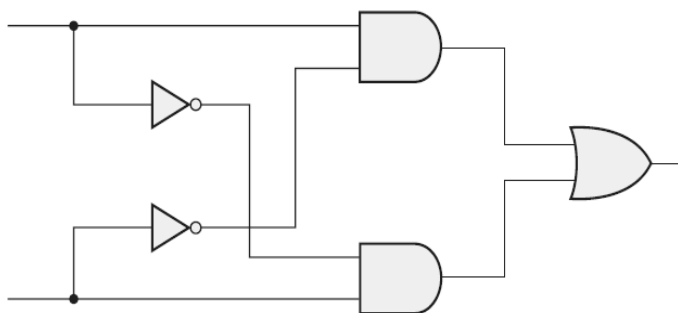


• OR 

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



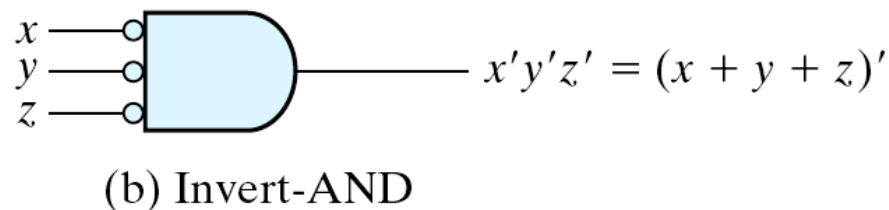
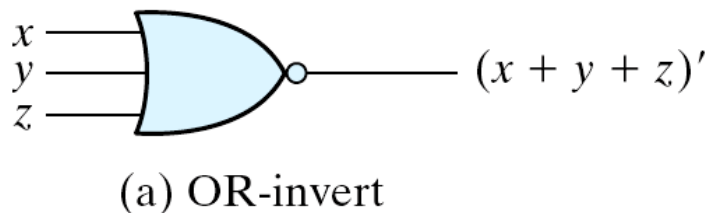
• XOR



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

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.



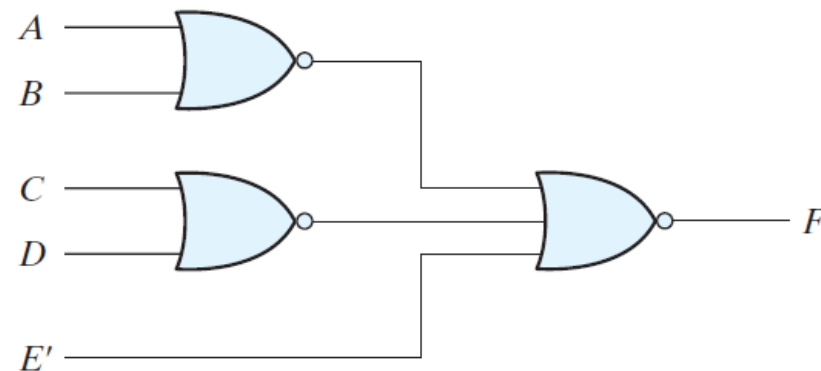
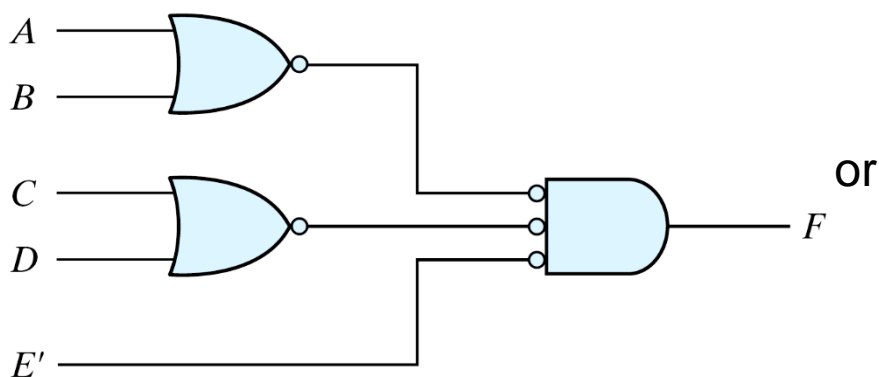
NOR-NOR Implementation

- 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) \Rightarrow NAND-NAND
product-of-sum (OR-AND) \Rightarrow 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 \rightarrow done
- Algebraic method
 - $F = (A+B)(C+D)E = ((A+B)(C+D)E)''$
 $= ((A+B)' + (C+D)' + E')' \rightarrow$ DeMorgan's
- Graphic method:



NOR-NOR Example2

- Example $F(x,y,z) = \sum(1,2,3,4,5,7)$

- Algebraic method

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

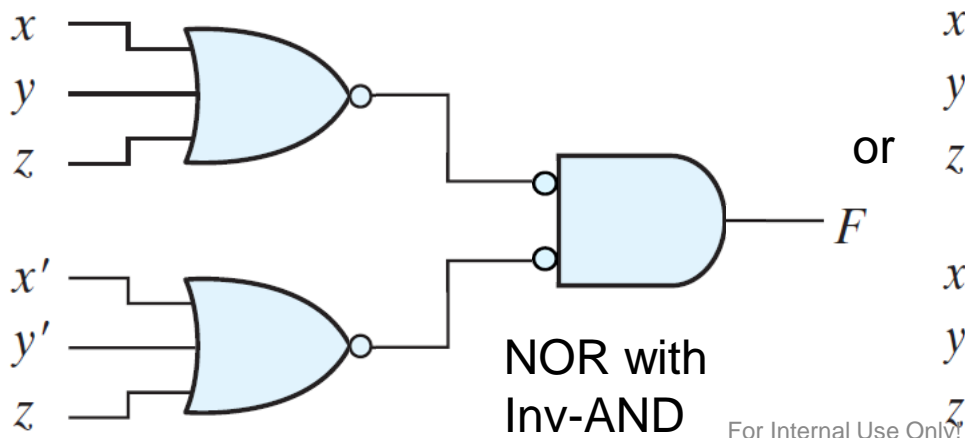
$$\begin{aligned} F &= (F')' = (x'y'z' + xyz')' = (((x'y'z')')' + ((xyz')')')' \\ &= ((x+y+z)' + (x'+y'+z)')' \end{aligned}$$

- Graphic method

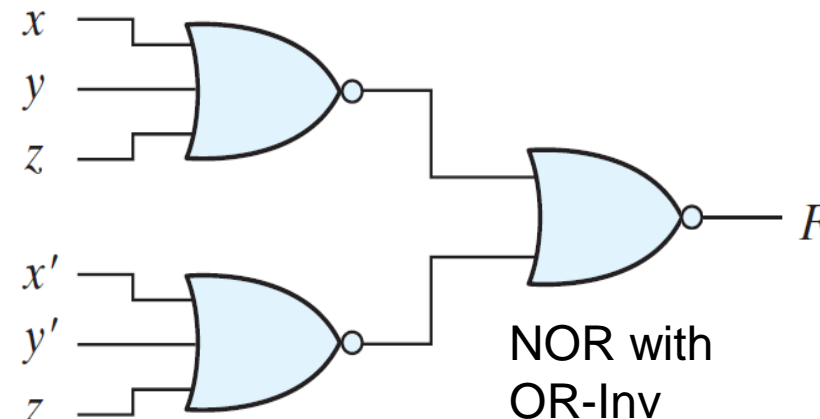
- $F' = x'y'z' + xyz'$

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

		y			
		00		01	11 10
x	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



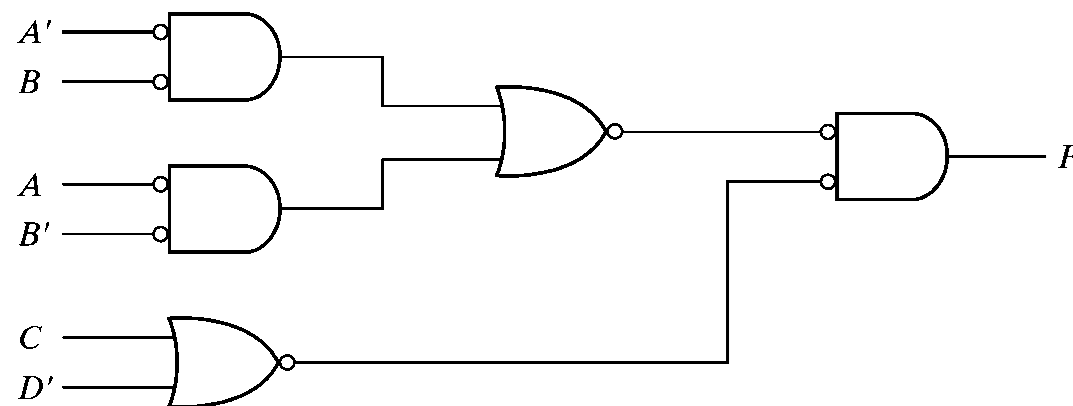
For Internal Use Only!



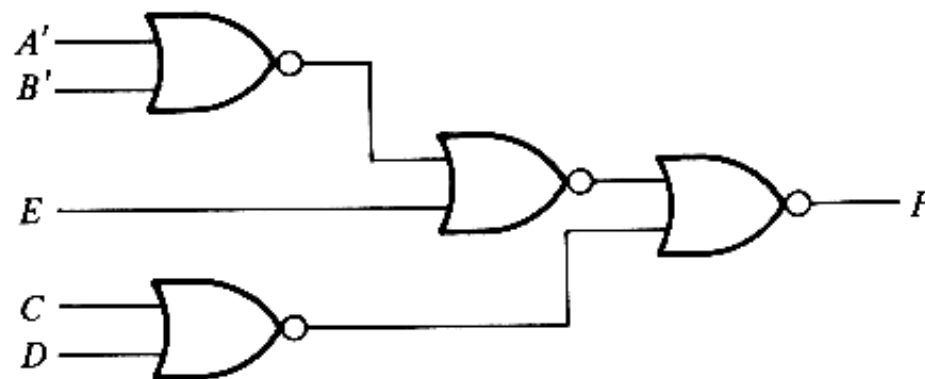
Multilevel NOR Implementation

- 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:

- $F = (AB' + A'B)(C + D')$



- $F = (AB + E)(C + D)$



For Internal Use Only!

Outline

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

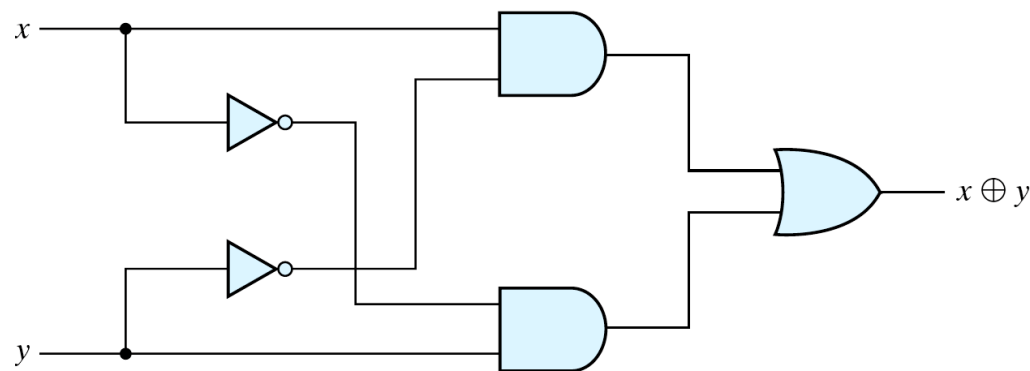
Exclusive-OR Function

- 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
 - $A \oplus B = B \oplus A$
 - $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

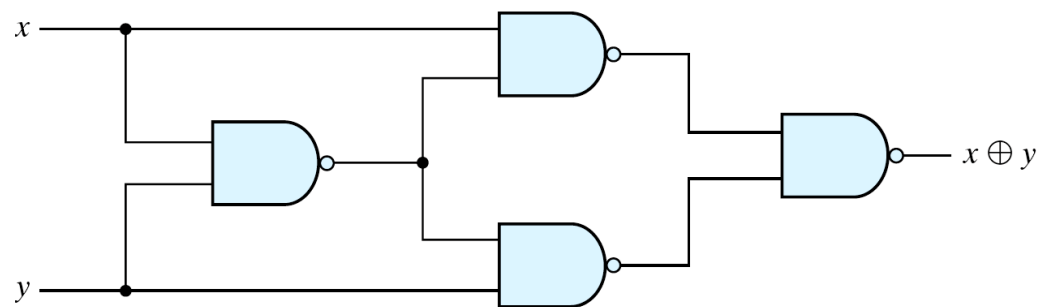
Exclusive-OR Implementations

- Implementations

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



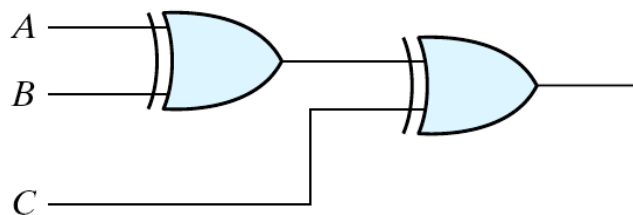
(a) With AND-OR-NOT gates



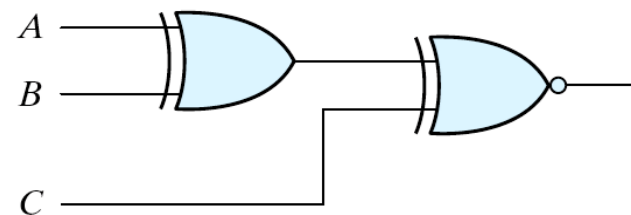
(b) With NAND gates

Odd function

- The XOR operation with three or more variables can be converted into an ordinary Boolean function by replacing the \oplus 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) \rightarrow if **odd** number of variables are equal to 1, then $F = 1$.
 - Even function (XNOR) \rightarrow 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 eighth bit 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.

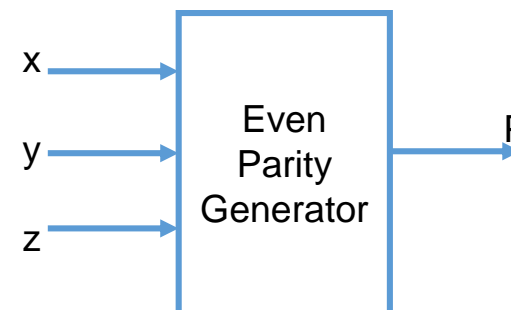
- Example:

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

Even-Parity-Generator Truth Table

Three-Bit Message			Parity Bit
<i>x</i>	<i>y</i>	<i>z</i>	<i>P</i>
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

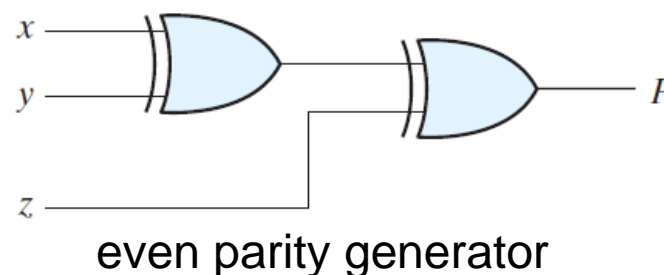


Parity Generation and Checking

- Parity Generation circuit and Checking circuit
- Generator produces an even parity bit with: $P = x \oplus y \oplus z$

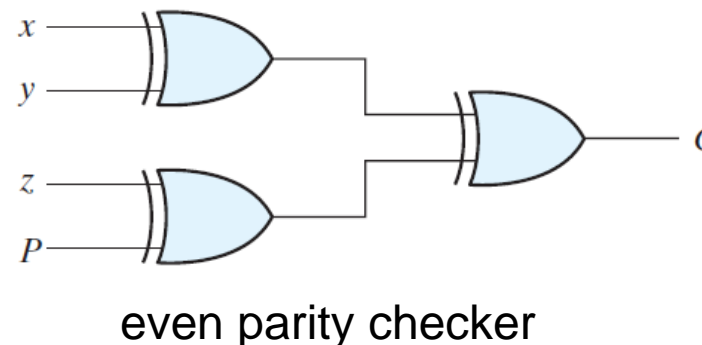
- $$P = xy'z' + x'yz' + xyz + x'y'z$$

$$= \sum(1, 2, 4, 7) - \text{odd function}$$



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



Outline

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

Two-Level Forms

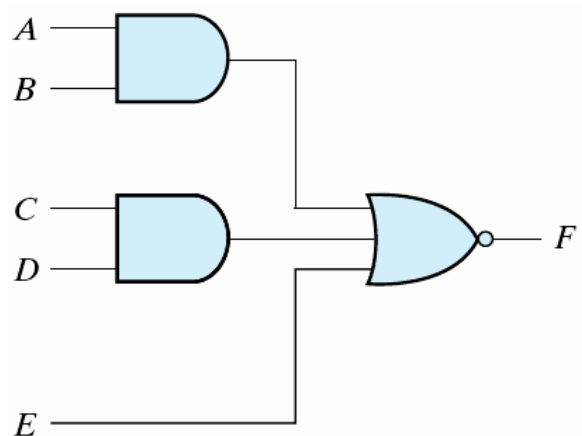
- AND/NAND/OR/NOR have 16 possible combinations of two-level forms
- Eight of them **degenerate** to a single operation
 - AND-AND \Rightarrow AND
 - OR-OR \Rightarrow OR
 - AND-NAND \Rightarrow NAND
 - OR-NOR \Rightarrow NOR
 - NAND-NOR \Rightarrow AND
 - NOR-NAND \Rightarrow OR
 - NAND-OR \Rightarrow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

Two-Level Forms

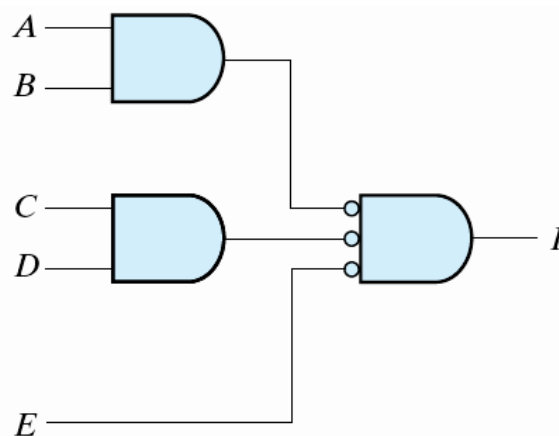
- Eight are **non-degenerate** forms
- AND-OR \Rightarrow standard sum-of-products
- NAND-NAND \Rightarrow standard sum-of-products
- OR-AND \Rightarrow standard product-of-sums
- NOR-NOR \Rightarrow standard product-of-sums
- NAND-AND/AND-NOR \Rightarrow AND-OR-INVERT (AOI)
 - **complement** of sum-of-products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

AND-OR-Invert Implementation

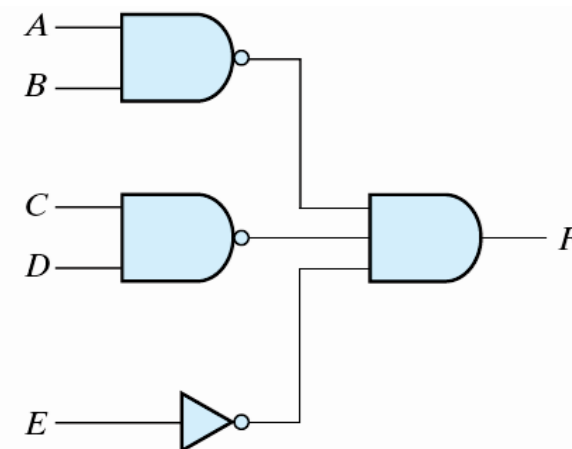
- NAND-AND = AND-NOR = AOI
 - $F(A,B,C,D,E) = (AB + CD + E)'$
 - $F'(A,B,C,D,E) = AB + CD + E$ (sum of products)
 - An AND-OR implementation requires an expression in sum-of-products form.
 - The AND-OR-INVERT implementation is similar, except for the inversion.



(a) AND-NOR



(b) AND-NOR

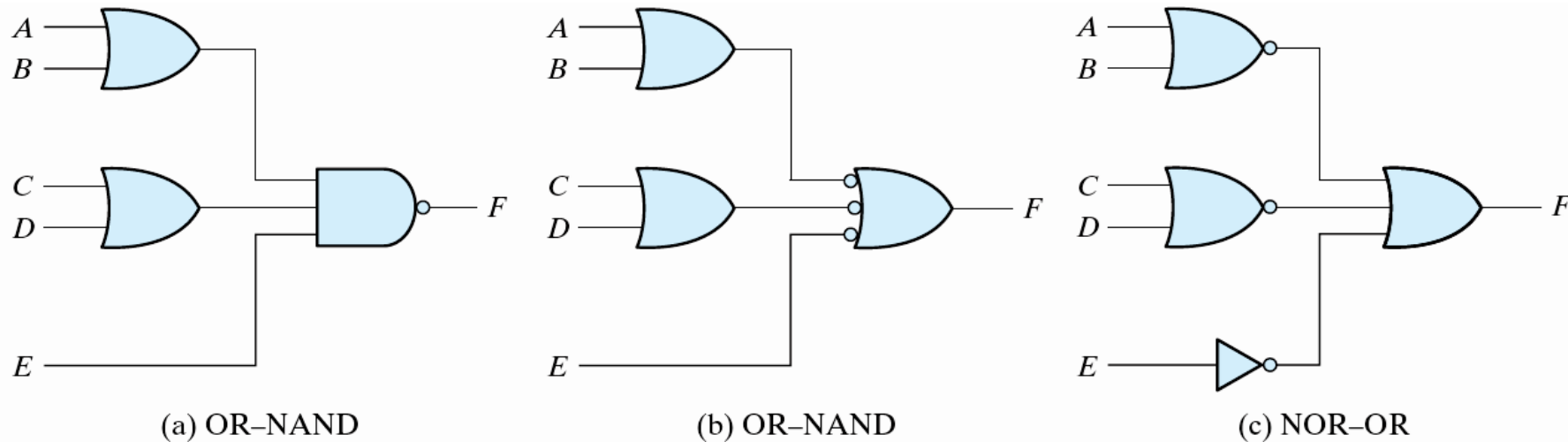


(c) NAND-AND

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.



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

AOI & OAI Example

- Example

		y z			
		00	01	11	10
x x'	0	m_0 1	m_1 0	m_3 0	m_2 0
	1	m_4 0	m_5 0	m_7 0	m_6 1

$x'y'z'$ points to m_0
 xyz' points to m_6

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

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

- 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.