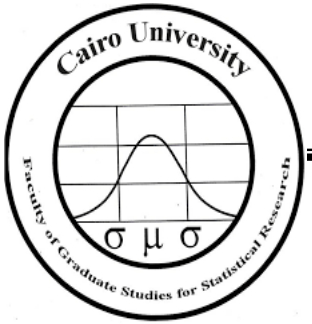


Computer Science Department

Cairo University

CS504

Digital Logic & Computer Organization

Lecture 5

Lecture Outline (Chapter 3)

- ★ **Don't Cares in K-Maps (Section 3.5)**
- ★ **Other Gate Types (NAND and NOR) (Section 3.6)**
  - **NAND Gate**
  - **NAND-NAND Implementation**
  - **Multi-Level NAND Circuits**
  - **NOR Gate**
  - **NOR-NOR Implementation**
  - **Multi-Level NOR Circuits**

Don't Cares in K-Maps

- **Sometimes a function table or K-map contains entries for which it is known:**
  - **The input values for the minterm will never occur, or**
  - **The output value for the minterm is not used**
- **In these cases, the output value need not be defined**
- **Instead, the output value is defined as a "don't care"**
- **By placing "don't cares" (an "x" entry) in the function table or map, the cost of the logic circuit may be lowered.**
- **Example: A logic function having the binary codes for the BCD digits as its inputs. Only the codes for 0 through 9 are used. The six codes, 1010 through 1111 never occur, so the output values for these codes are "x = don't cares."**

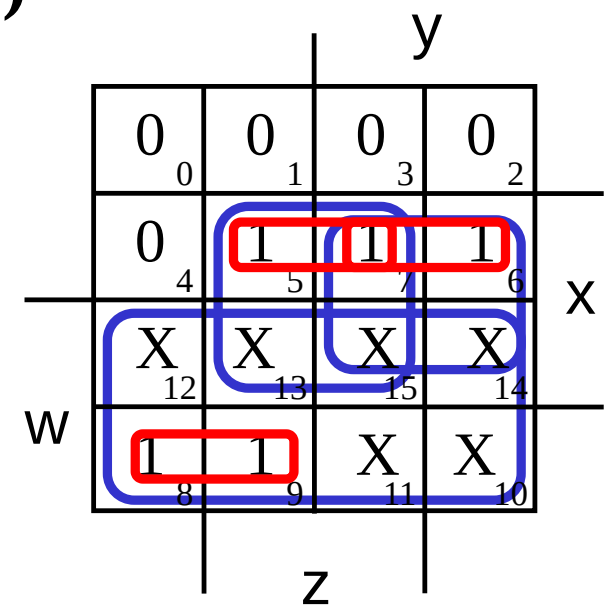
Example: BCD “5 or More”

- The map below gives a function F(w,x,y,z) which is defined as "5 or more" over BCD inputs. With the don't cares used for the 6 non-BCD combinations:
- If the don’t cares were treated as 0’s we get:

  $F1 = \overline{w}\,x\,z + \overline{w}\,x\,y + w\,\overline{x}\,\overline{y}$ (G = 12)
- If the don’t cares were 1’s we get:

  $F2 = w + x\,z + x\,y$ (G = 7 better)

The selection of don’t cares depends on which combination gives the simplest expression



Product-of-Sums Example

- **Find the optimum POS expression for :**

  $F(A, B, C, D) = \sum (3,9,11,12,13,14,15) + \sum_d (1,4,6)$

  Where $\sum_d$ indicates the don't care minterms

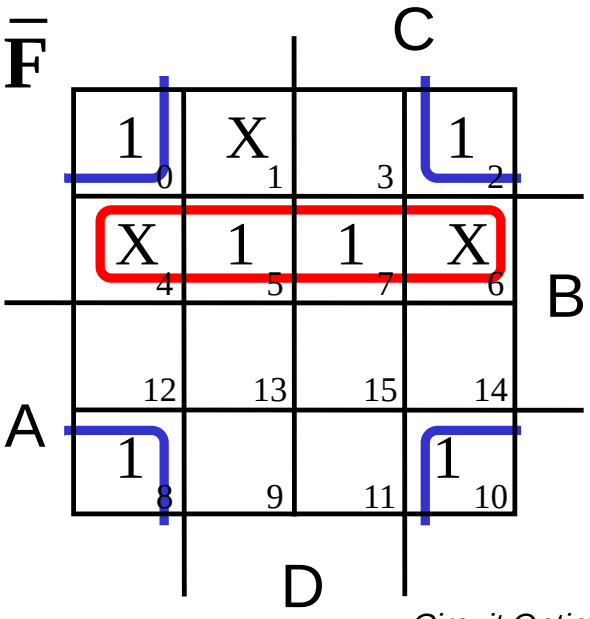
- **Solution: Find $\overline{F} = \sum (0, 2, 5, 7, 8, 10) + \sum_d (1,4,6)$**

- **$\overline{F} = \overline{B}\,\overline{D} + \overline{A}\,B$**

- **Optimum POS expression:**

  $F = (B + D)(A + \overline{B})$

  Gate input cost (G = 6)

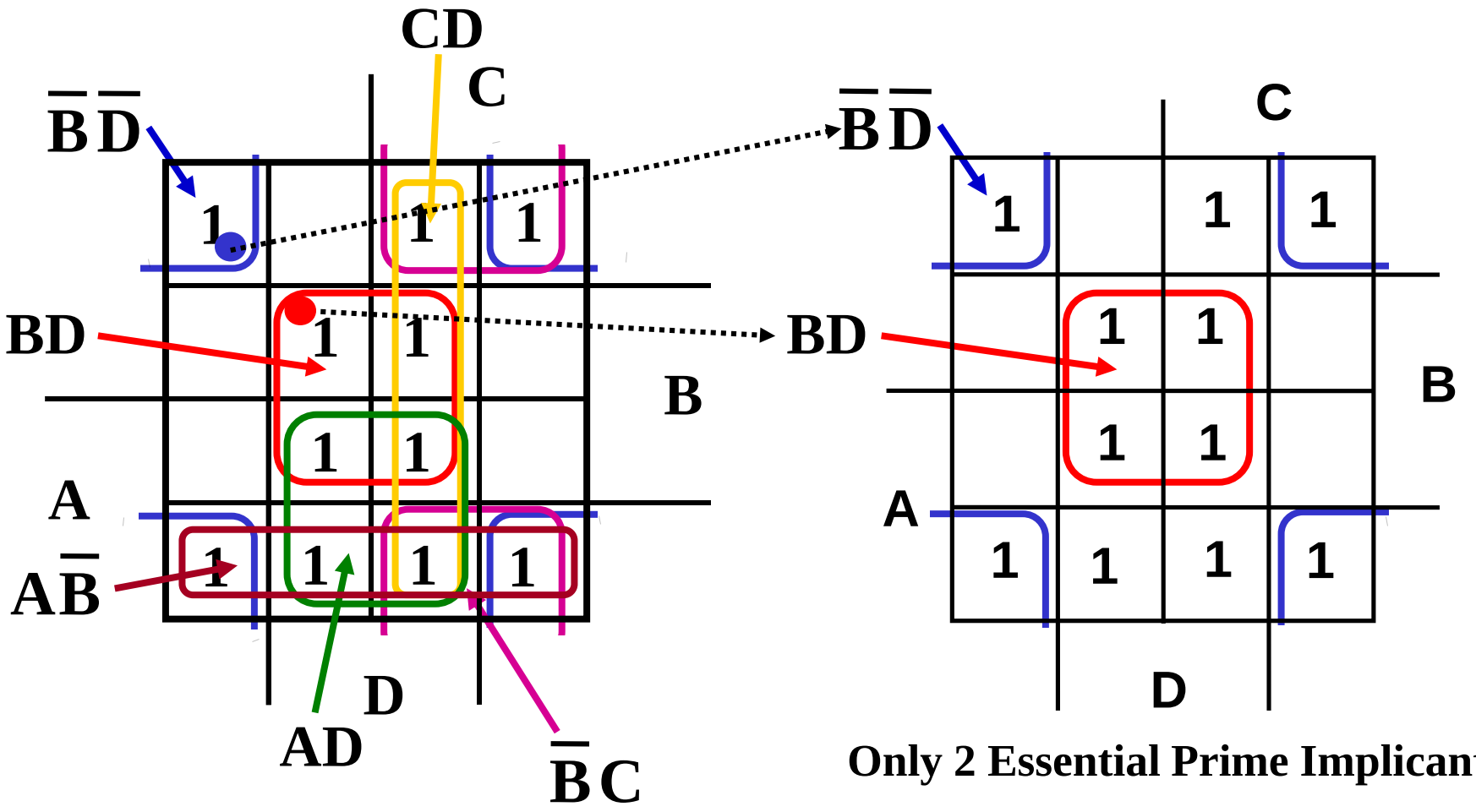


Systematic Simplification

- A Prime Implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map into a rectangle, with the number of squares equal to a power of 2.
- A prime implicant is called an Essential Prime Implicant if it is the only prime implicant that covers one or more minterms.
- Prime Implicants and Essential Prime Implicants can be determined by inspection of a K-Map.
- A set of prime implicants covers all minterms if, for each minterm of the function, at least one prime implicant in the set of prime implicants includes the minterm.

Example of Prime Implicants

- **Find ALL Prime Implicants**



Only 2 Essential Prime Implicants

Prime Implicant Practice

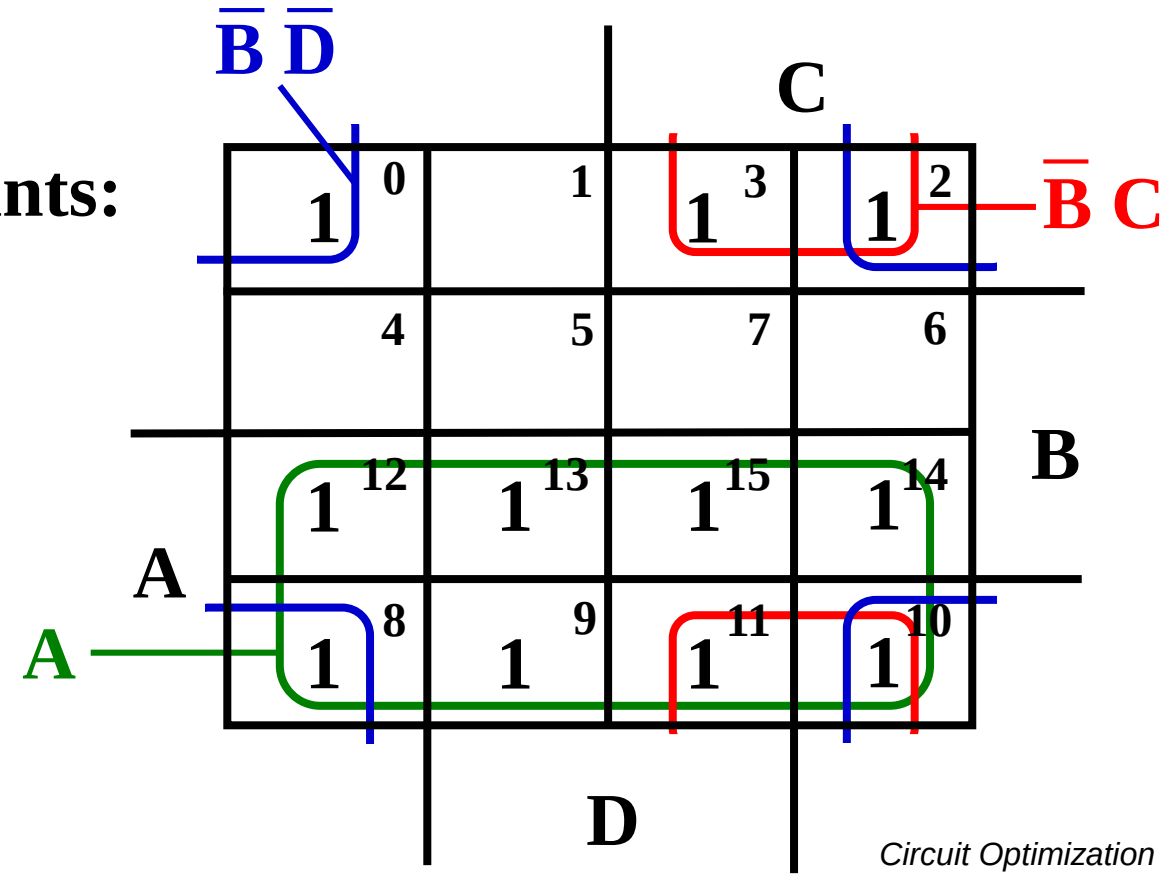
- **Find all prime implicants for**

$$F(A, B, C, D) = \sum (0,2,3,8,9,10,11,12,13,14,15)$$

3 prime implicants:

A, $\overline{B}$ C, $\overline{B}$ $\overline{D}$

All 3 prime implicants are essential



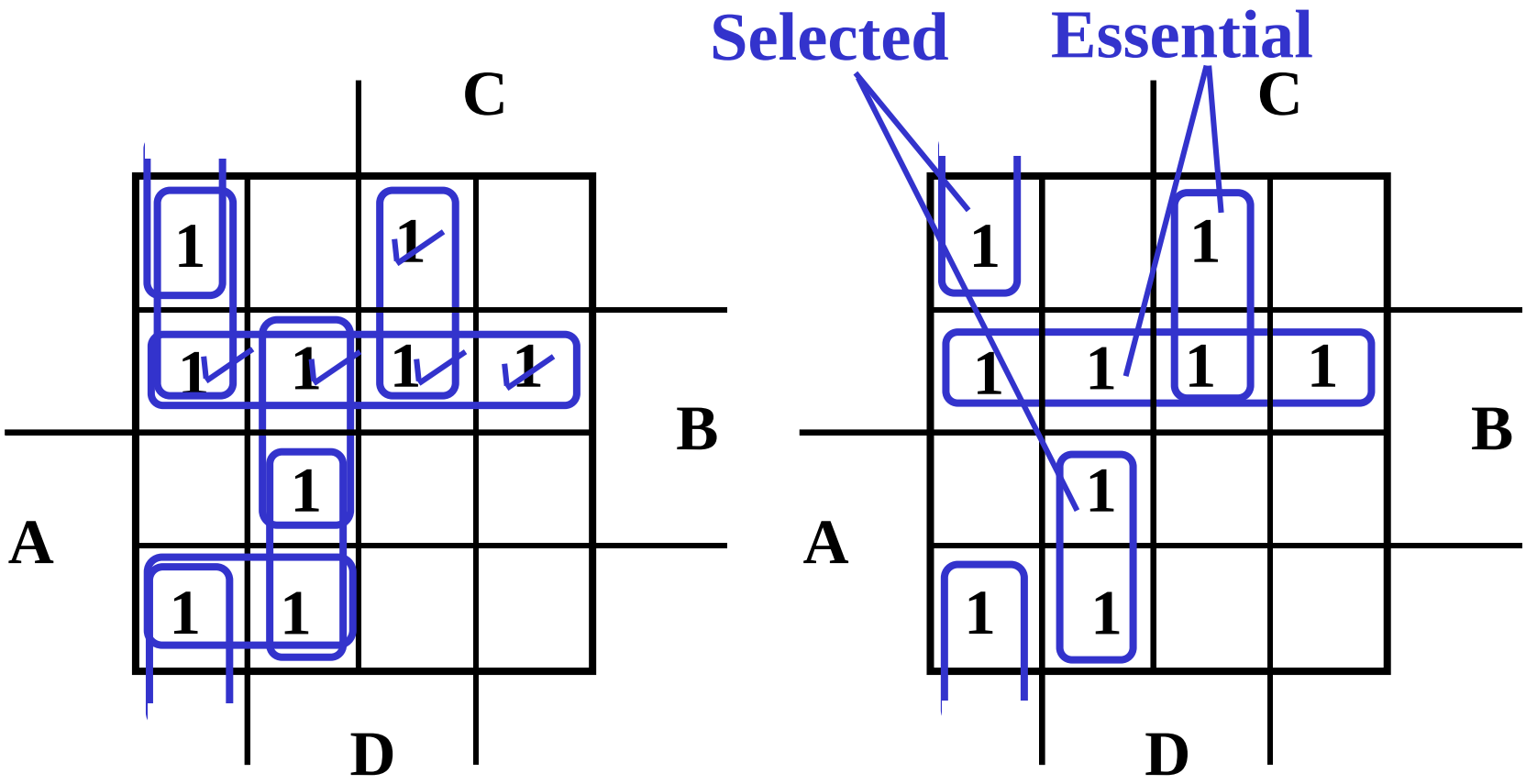
Optimization Algorithm

1. Find **all** prime implicants.
2. Include **all essential** prime implicants in the solution.
3. Select a minimum cost set of non-essential prime implicants to cover all minterms not yet covered.

- Prime implicant selection rule:
  A. Minimize the overlap among prime implicants.
  B. In particular, in the final solution, make sure that each prime implicant selected includes at least one minterm not included in any other prime implicant selected.

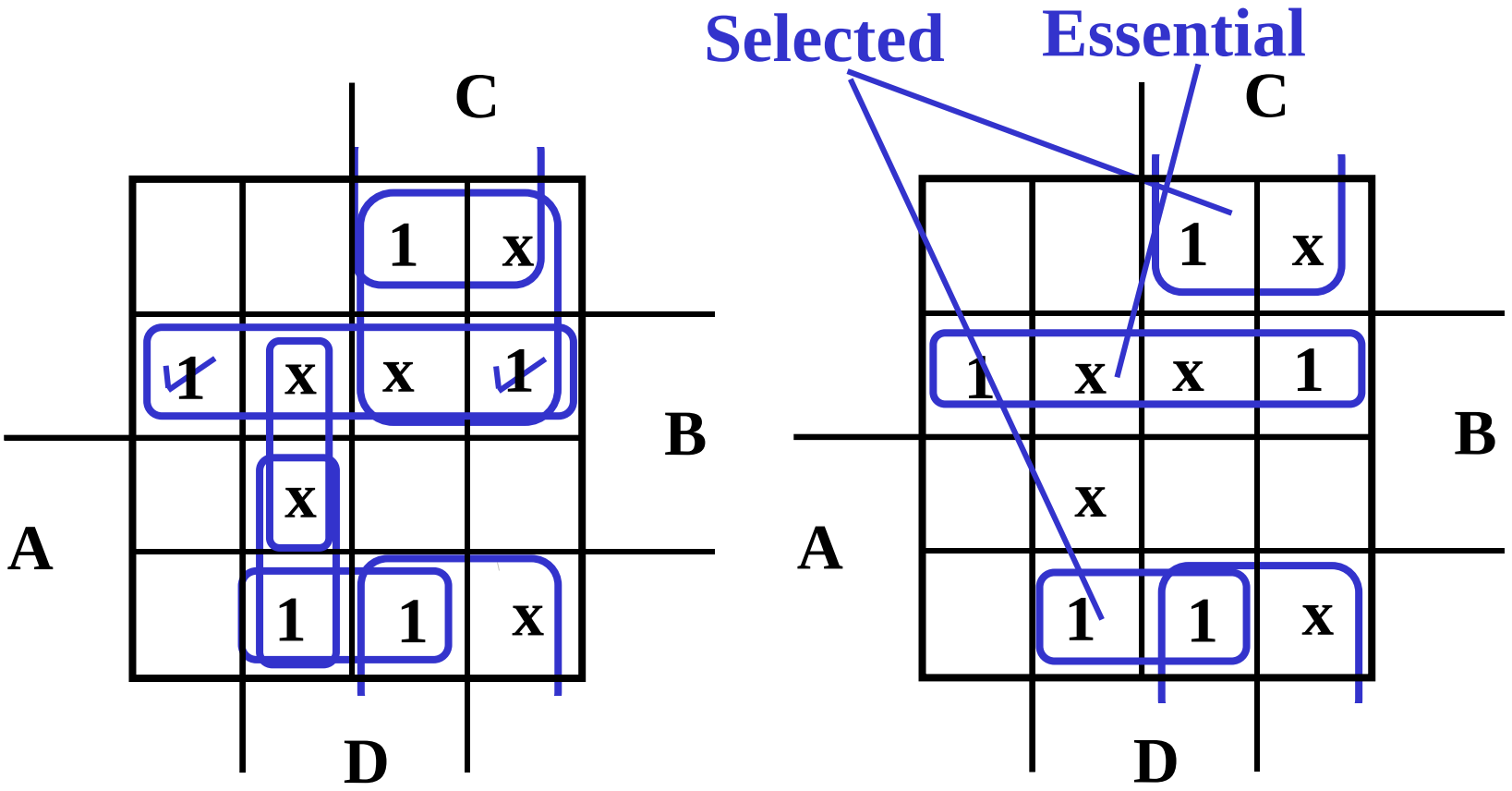
Selection Rule Example

- **Simplify F(A, B, C, D) given on the K-map**



Selection Rule Example with Don't Cares

- Simplify F(A, B, C, D) given on the K-map.



✓ Minterms covered by essential prime implicants

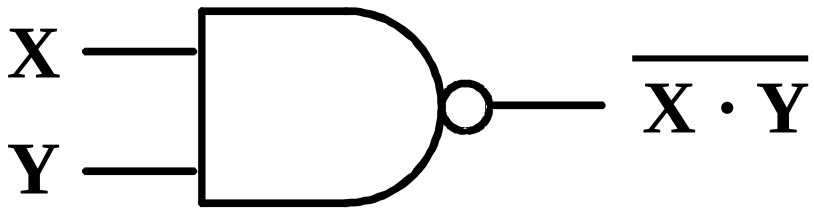
Other Gate Types (NAND and NOR)

- Why?
  - Ease of fabrication
  - Low cost implementation
  - Digital circuits are made of NAND or NOR, rather than AND and OR gates

- We need rules to convert from AND/OR/NOT to NAND/NOR circuits

NAND Gate

- The basic NAND gate has the following symbol and truth table:
  - AND-Invert (NAND) Symbol:

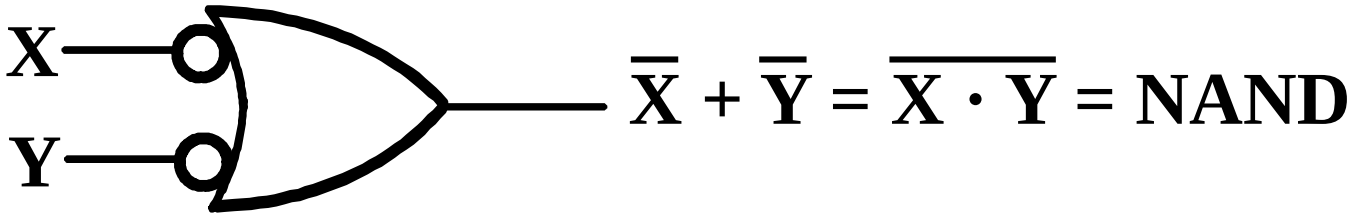


X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

- NAND represents NOT AND. The small "bubble" circle represents the invert function
- The NAND gate is implemented efficiently in CMOS technology in terms of chip area and speed

NAND Gate: Invert-OR Symbol

- Applying DeMorgan's Law: Invert-OR = NAND

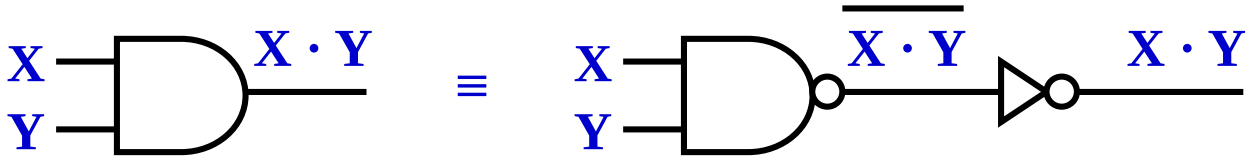


- This NAND symbol is called Invert-OR
  - Since inputs are inverted and then ORed together
- AND-Invert & Invert-OR both represent NAND gate
  - Having both makes visualization of circuit function easier
- Unlike AND, the NAND operation is NOT associative

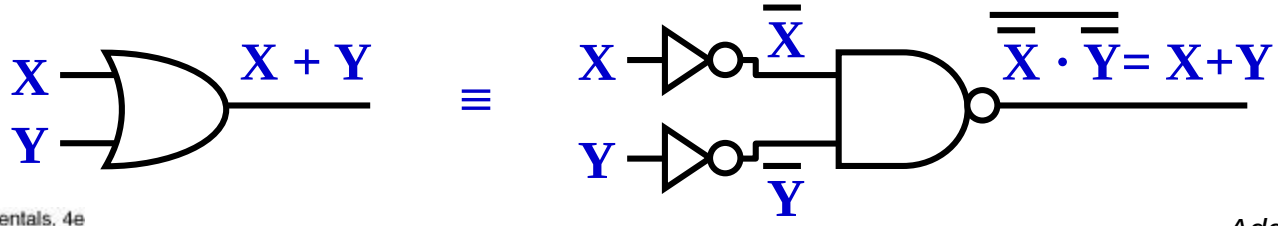
(X NAND Y) NAND Z ≠ X NAND (Y NAND Z)

The NAND Gate is Universal

- NAND gates can implement any Boolean function
- NAND gates can be used as inverters, or to implement AND / OR operations
- A NAND gate with one input is an inverter
- AND is equivalent to NAND with inverted output



- OR is equivalent to NAND with inverted inputs

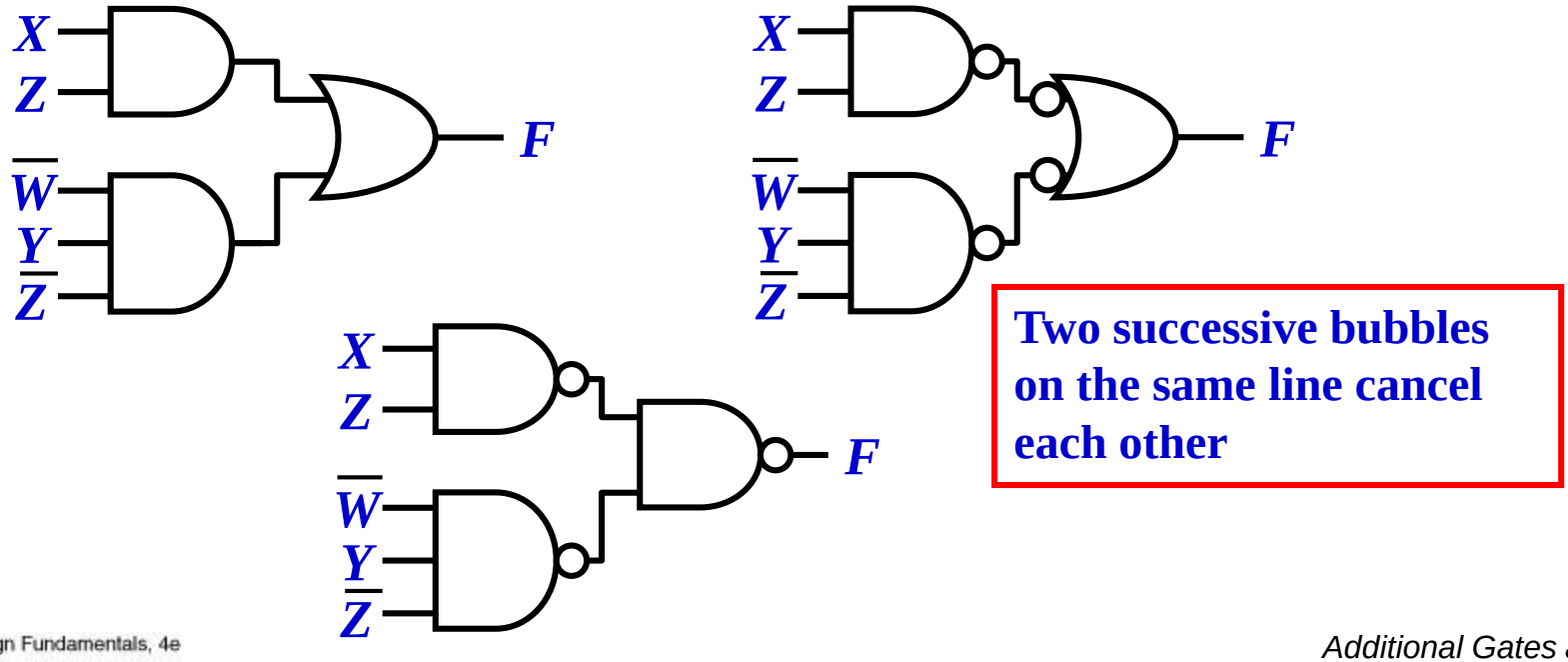


NAND–NAND Implementation

- Consider the Following SOP Expression:

$$F = XZ + \overline{W}Y\overline{Z}$$

- A 2-level AND-OR circuit can be converted easily to a NAND-NAND implementation



Two successive bubbles on the same line cancel each other

NAND–NAND Implementation (2)

- **Requires function to be in SOP form**
- **Example: F = AB+CD = ((AB)' . (CD)')'**
- **Three ways to implement F**

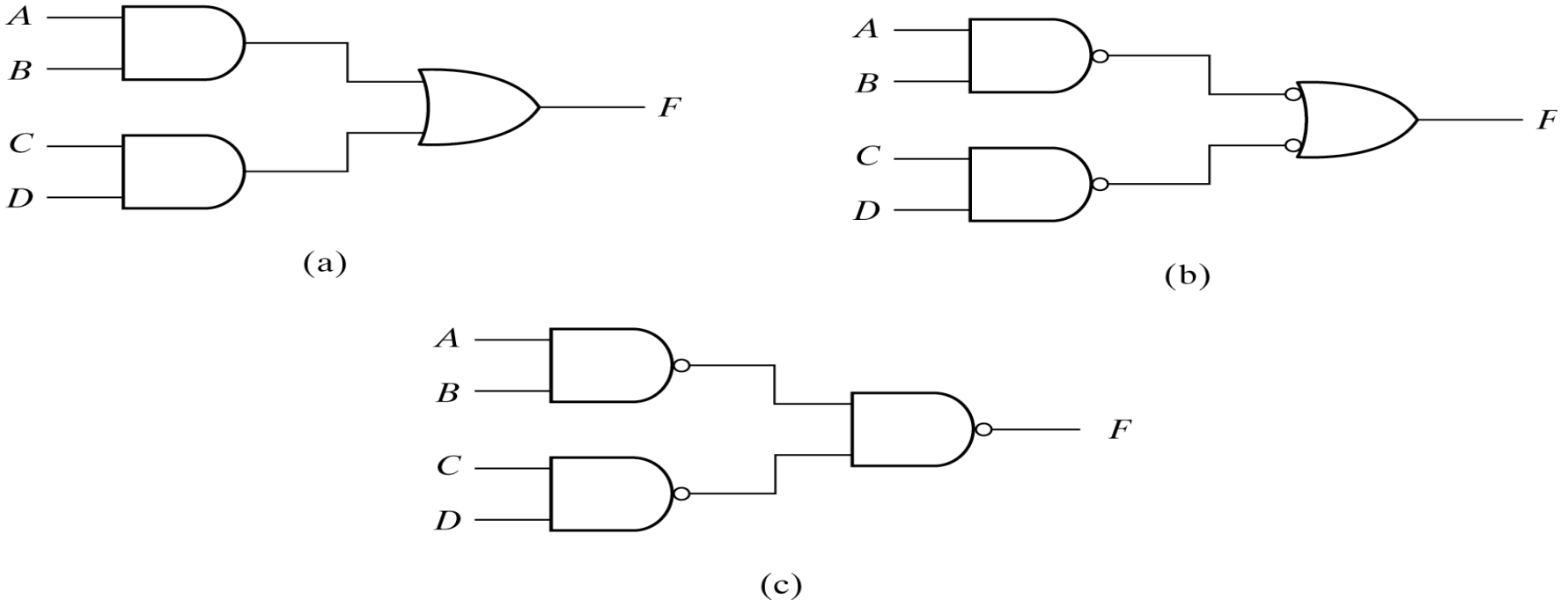


Fig. 3-20 Three Ways to Implement $F = AB + CD$

Multilevel NAND Circuits

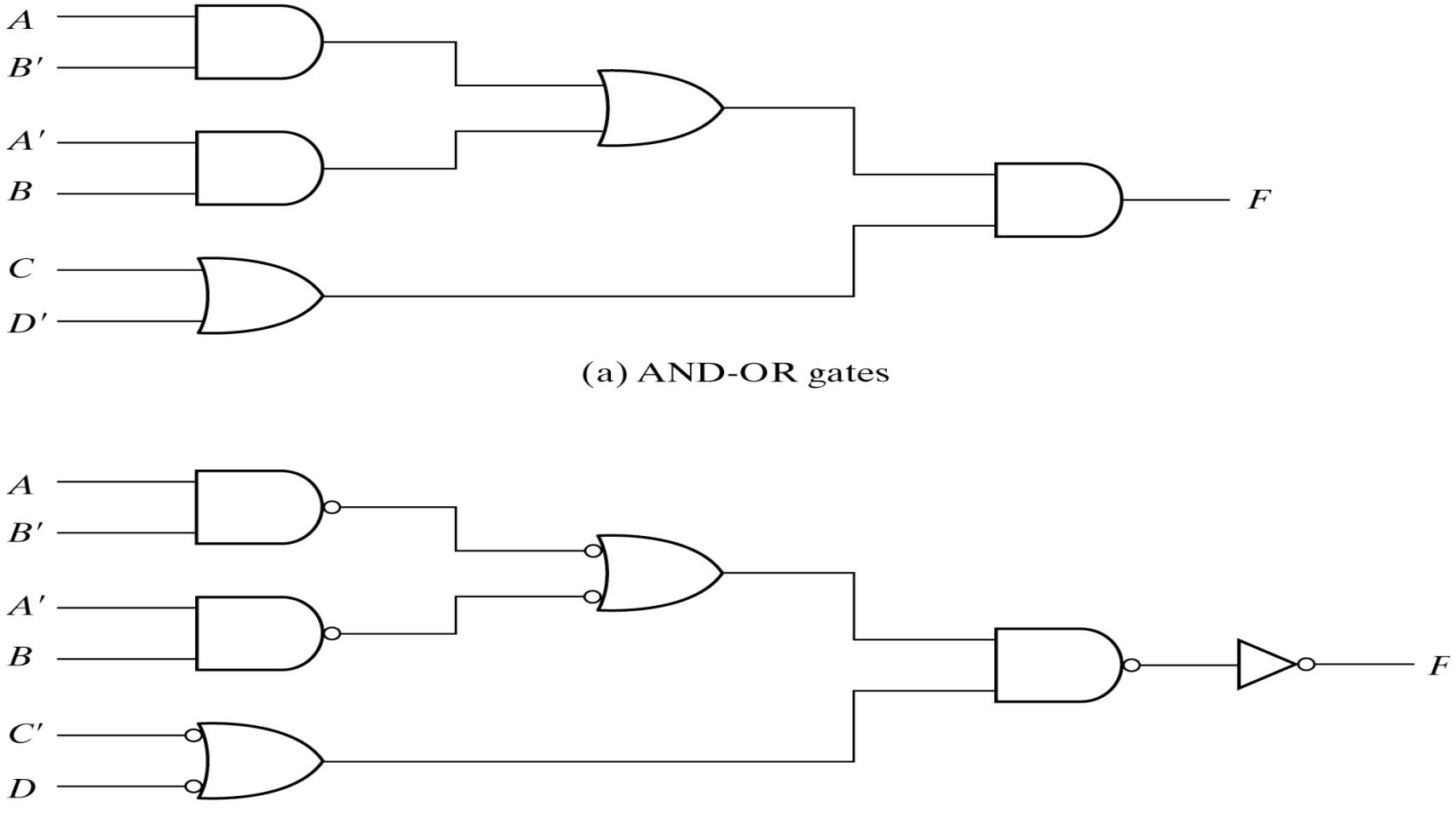
- SOP results in two-level designs
- Not all designs are two-level, e.g., **F=A(CD+B)+BC'**
- How do we convert multilevel circuits to NAND circuits?
- Rules

  1-Convert all ANDs to NAND gates with AND-invert symbol

  2-Convert all ORs to NAND gates with invert-OR symbols

  3-Check the bubbles, insert bubble if not compensated

Multilevel NAND Circuits (2)



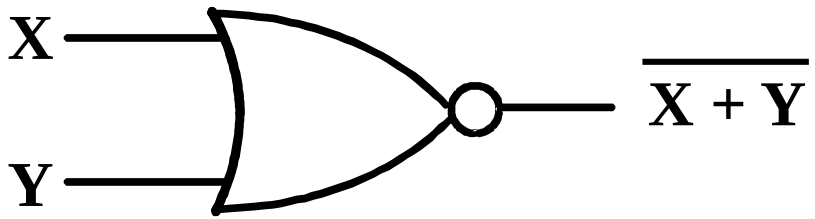
(a) AND-OR gates

(b) NAND gates

Fig. 3-23 Implementing $F = (AB' + A'B)(C + D')$

NOR Gate

- The basic NOR gate has the following symbol and truth table:
  - OR-Invert (NOR) Symbol:

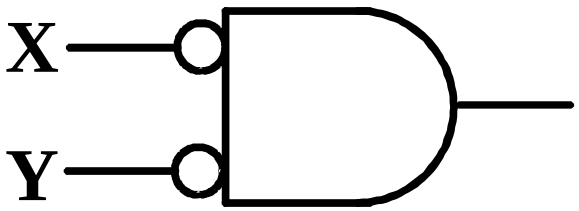


X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

- NOR represents NOT OR. The small "bubble" circle represents the invert function.
- The NOR gate is also implemented efficiently in CMOS technology in terms of chip area and speed

NOR Gate: Invert-AND Symbol

- The Invert-AND symbol is also used for NOR



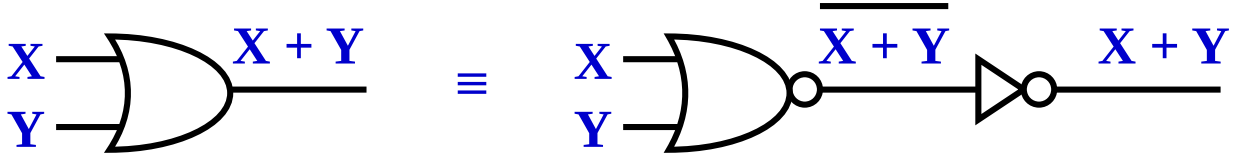
$$\overline{X} \cdot \overline{Y} = \overline{X + Y} = \text{NOR}$$

- This NOR symbol is called Invert-AND, since inputs are inverted and then ANDed together
- OR-Invert & Invert-AND both represent NOR gate
  - Having both makes visualization of circuit function easier
- Unlike OR, the NOR operation is NOT associative

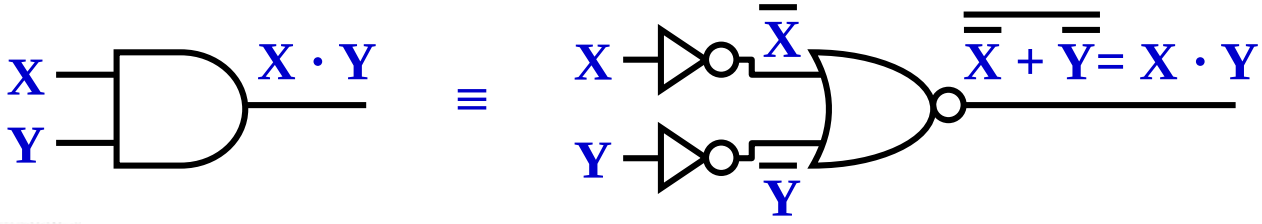
(X NOR Y) NOR Z ≠ X NOR (Y NOR Z)

The NOR Gate is also Universal

- NOR gates can implement any Boolean function
- NOR gates can be used as inverters, or to implement AND / OR operations
- A NOR gate with one input is an inverter
- OR is equivalent to NOR with inverted output



- AND is equivalent to NOR with inverted inputs

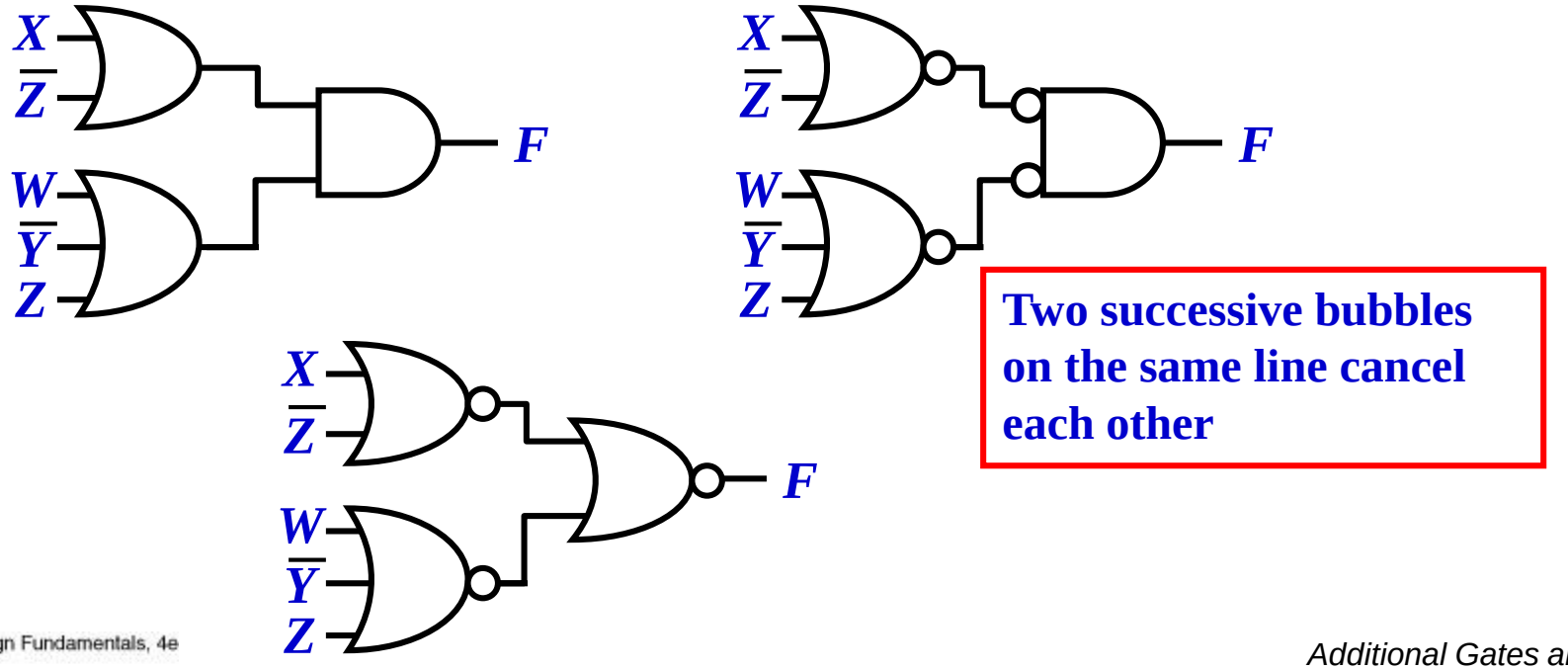


NOR–NOR Implementation

- Consider the Following POS Expression:

$$F = (X + \overline{Z})(W + \overline{Y} + Z)$$

- A 2-level OR-AND circuit can be converted easily to a NOR-NOR implementation



Two successive bubbles on the same line cancel each other

NOR–NOR Implementation (2)

- **Requires function to be in POS form**
- **Example: F = (A+B) . (C+D) . E**
- **F = ((A+B)' + (C+D)' + E')'**

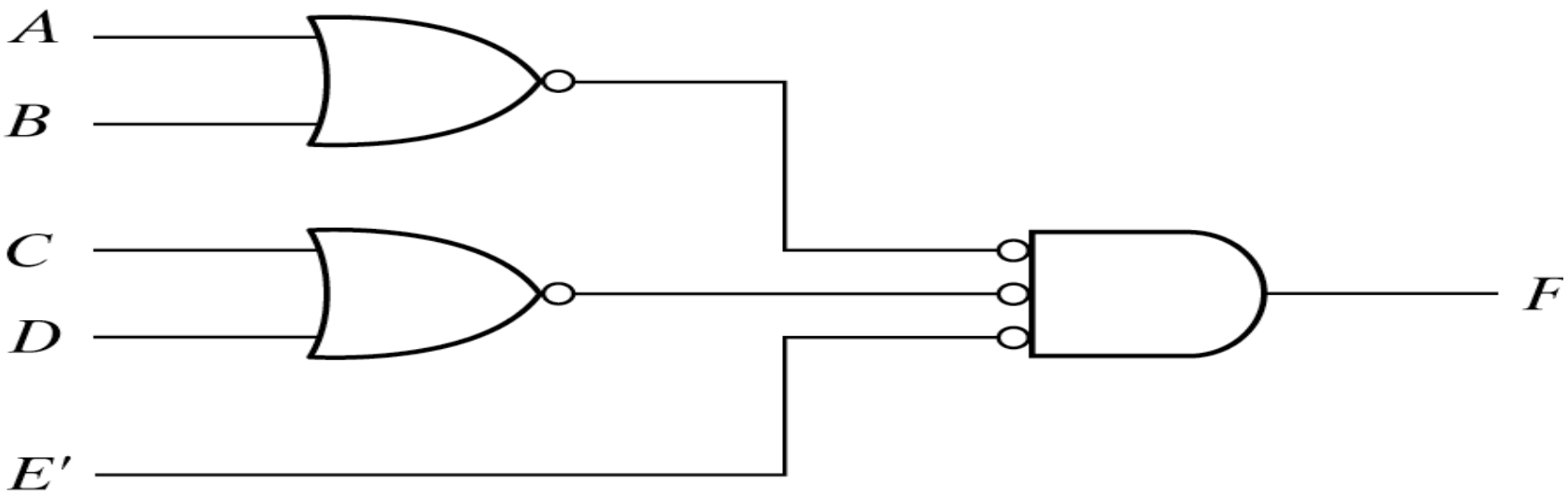


Fig. 3-26 Implementing $F = (A + B)(C + D)E$

Multilevel NOR Circuits

- POS results in two-level designs
- Not all designs are two-level, e.g., **F=(AB'+A'B)(C+D')**
- How do we convert multilevel circuits to NOR circuits?
- Rules

  1-Convert all ORs to NOR gates with OR-invert symbol

  2-Convert all ANDs to NOR gates with invert-AND symbols

  3-Check the bubbles, insert bubble if not compensated

Multilevel NOR Circuits (2)

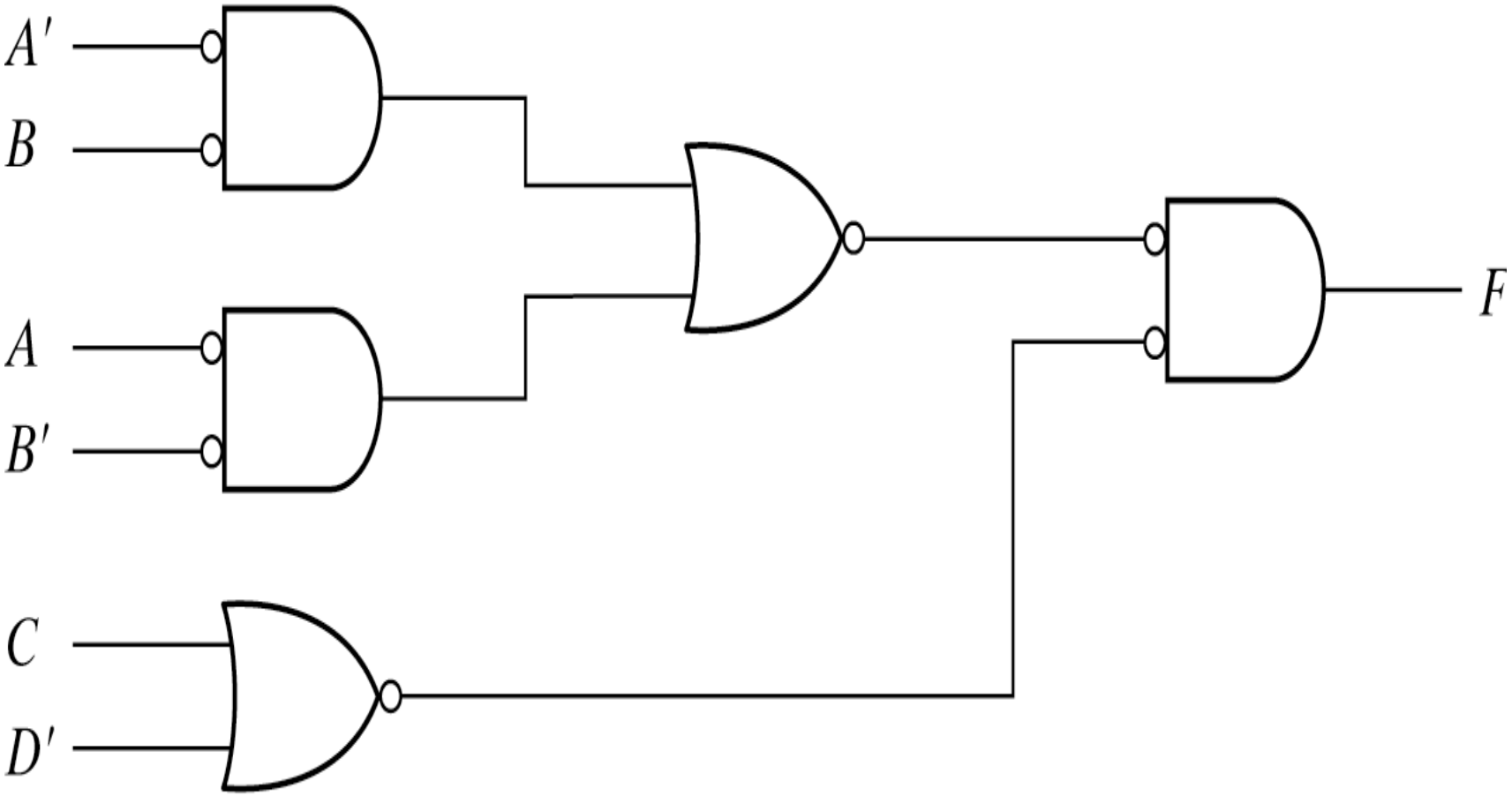
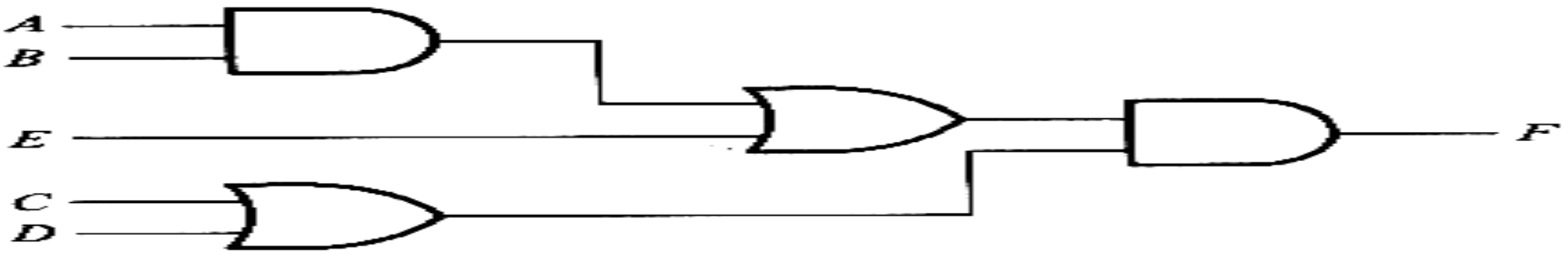
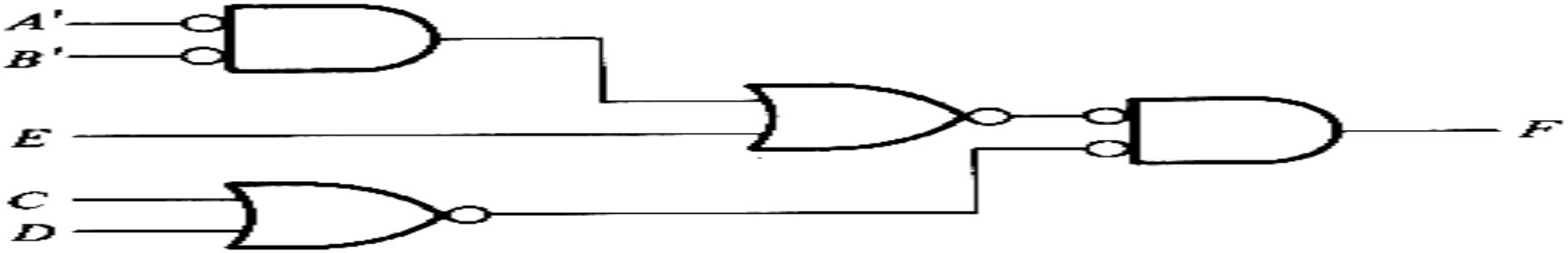


Fig. 3-27 Implementing $F = (AB' + A'B)(C + D')$ with NOR Gates

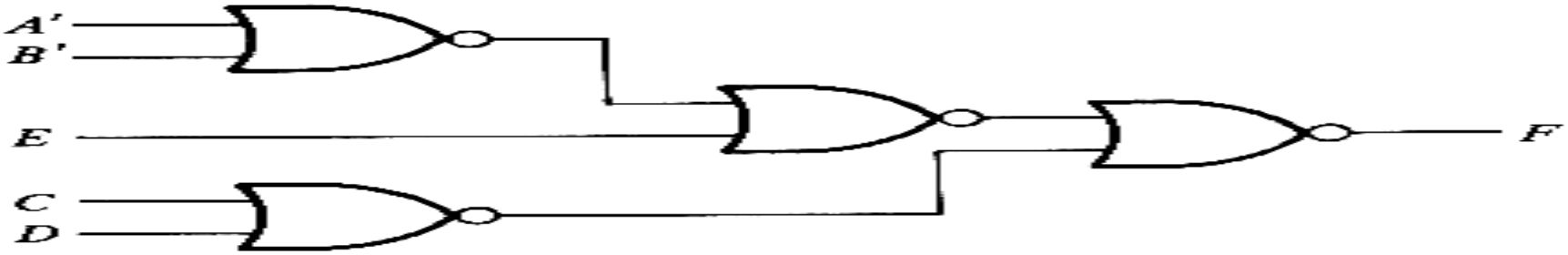
Multilevel NOR Circuits (3)



(a) AND-OR diagram



(b) NOR diagram



(c) Alternate NOR diagram

The End

Questions?