

CDA 3200 Digital Systems

Instructor: Dr. Janusz Zalewski

Developed by: Dr. Dahai Guo

Spring 2012

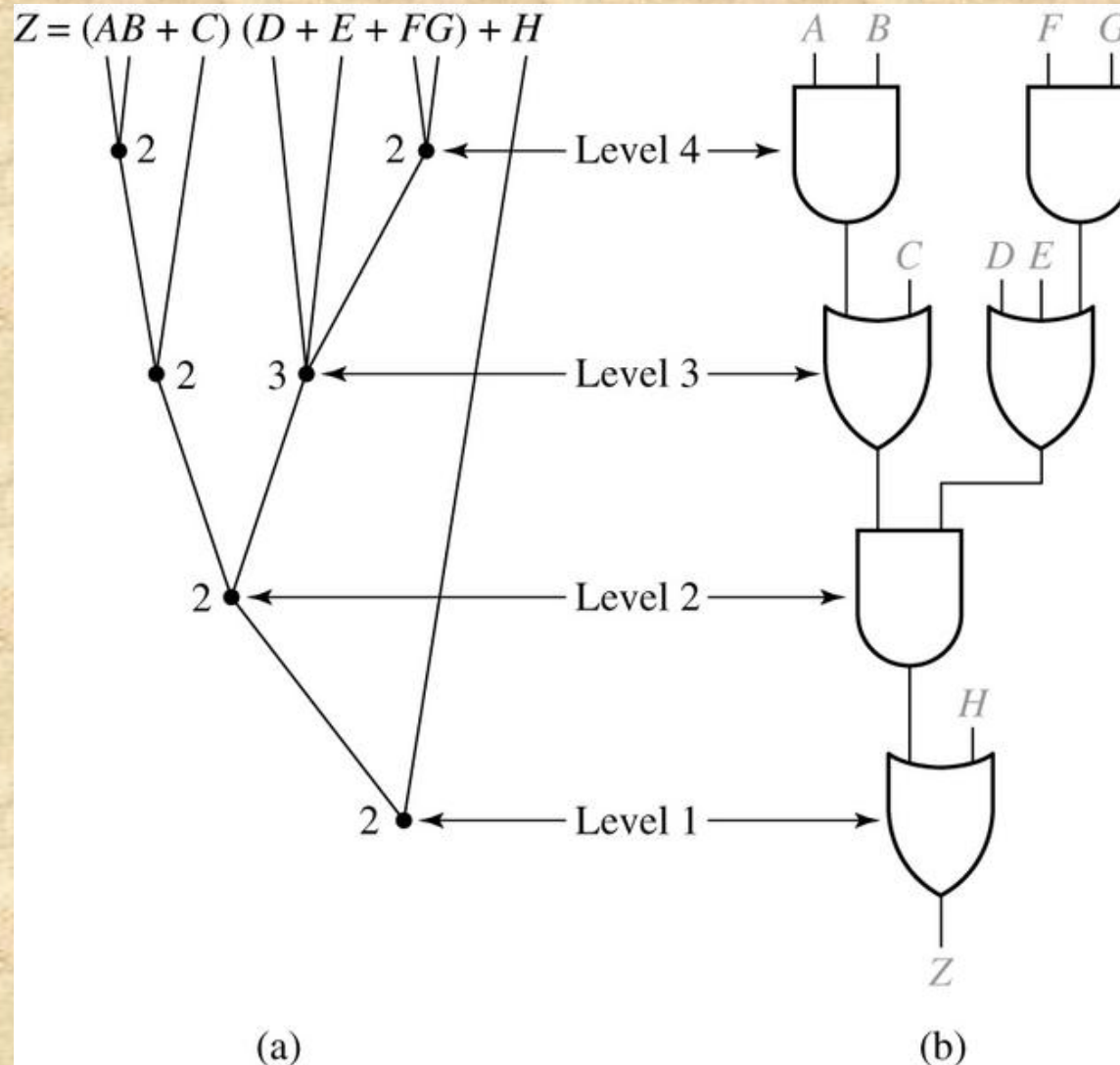
Outline

- Multi-Level Gate Circuits
- NAND and NOR Gates
- Design of Two-Level Circuits Using NAND and NOR Gates

Multi-Level Gate Circuits (1/3)

- The maximum number of gates cascaded in series between a circuit input and the output is referred to as the number of levels of gates.
 - sum-of-products: two levels
 - product-of-sums: two levels
- Inverters are not counted.

Multi-Level Gate Circuits (2/3)

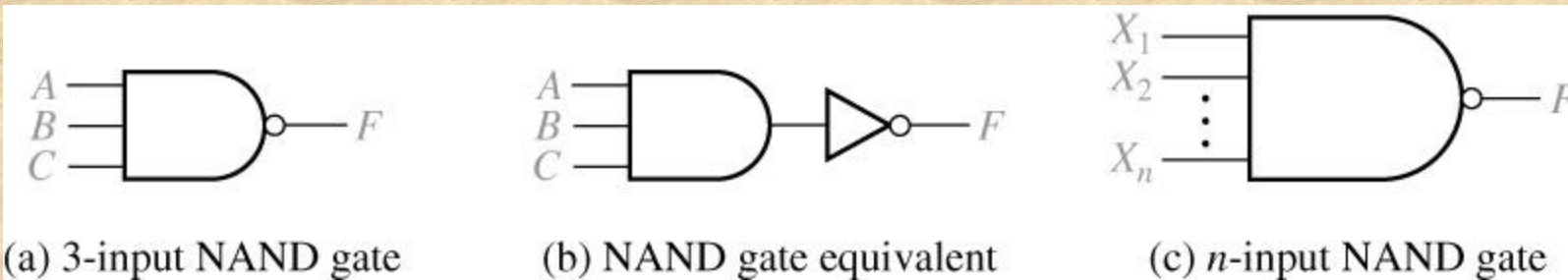


Multi-Level Gate Circuits (3/3)

- When the input of a gate is changed, there is a finite time before the output changes.
- The number of gates which can be cascaded is limited by gate delays.
- When several gates are cascaded, the gate delay may become excessive and slow down the operation of the digital system.

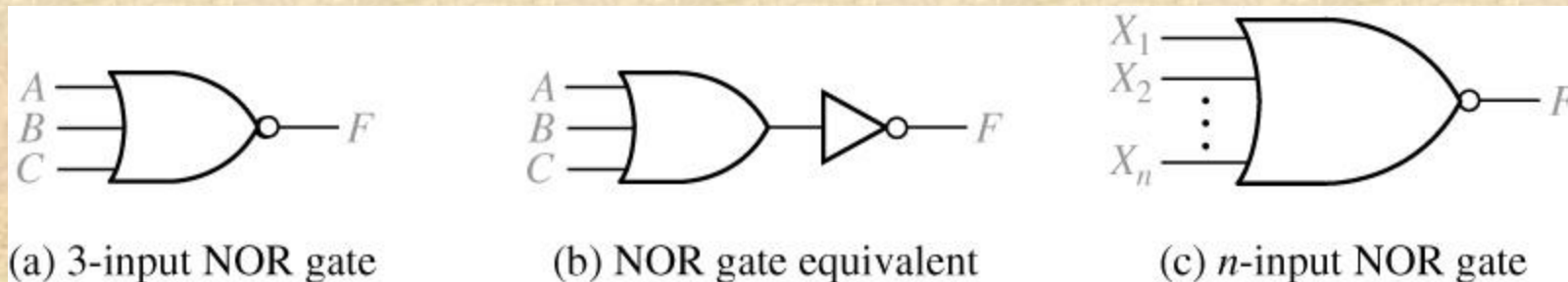
NAND and NOR Gates (1/5)

- NAND gate



– $F(ABC) = (ABC)' = A' + B' + C'$

- NOR gate



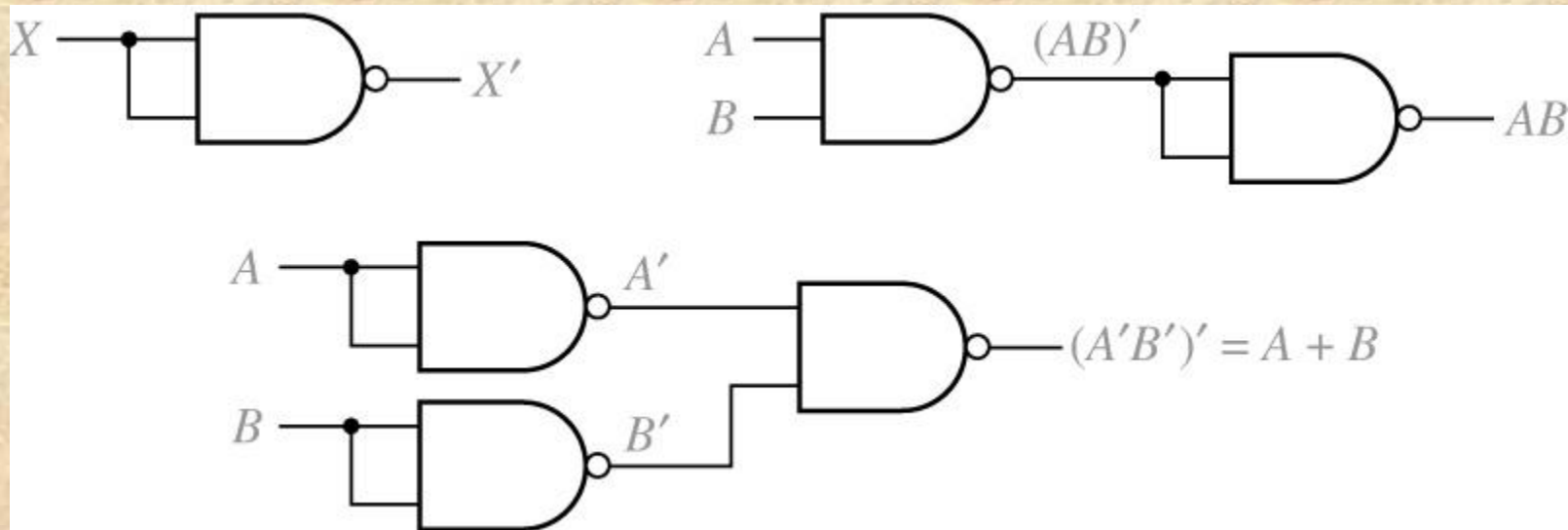
– $F(ABC) = (A+B+C)' = A'B'C'$

- Any logic function can be implemented using only NAND or NOR gates.

NAND and NOR Gates (2/5)

- If we can use NAND or NOR gates to implement AND, OR, and inverter, then we can prove that any logic function can be expressed using only NAND or NOR gates.
- A set of logic operations is said to be *functionally complete* if any Boolean function can be expressed in terms of this set of operations. The set AND, OR, and NOT is obviously functionally complete.

NAND and NOR Gates (3/5)



$X' = X \text{ nand } X$

$AB = (A \text{ nand } B) \text{ nand } (A \text{ nand } B)$

$A + B = (A \text{ nand } A) \text{ nand } (B \text{ nand } B)$

NAND and NOR Gates (4/5)

- Actually, as long as we could show NAND can express OR and NOT (AND and NOT), we can show NAND is functionally complete.
 - $XY = (X' + Y)'$
 - $X + Y = (X'Y)'$

NAND and NOR Gates (5/5)

- Can you prove that NOR is functionally complete?

Design of Two-Level Circuits Using NAND and NOR Gates (1/7)

- The conversion from circuits composed of AND and OR gates to circuits composed of NAND or NOR gates is carried out by using $F=(F')'$ and then applying DeMorgan's laws:

$$- (X_1+X_2+\dots+X_n)'=X_1'X_2'\dots X_n'$$

$$- (X_1X_2\dots X_n)'= X_1'+X_2'+\dots+X_n'$$

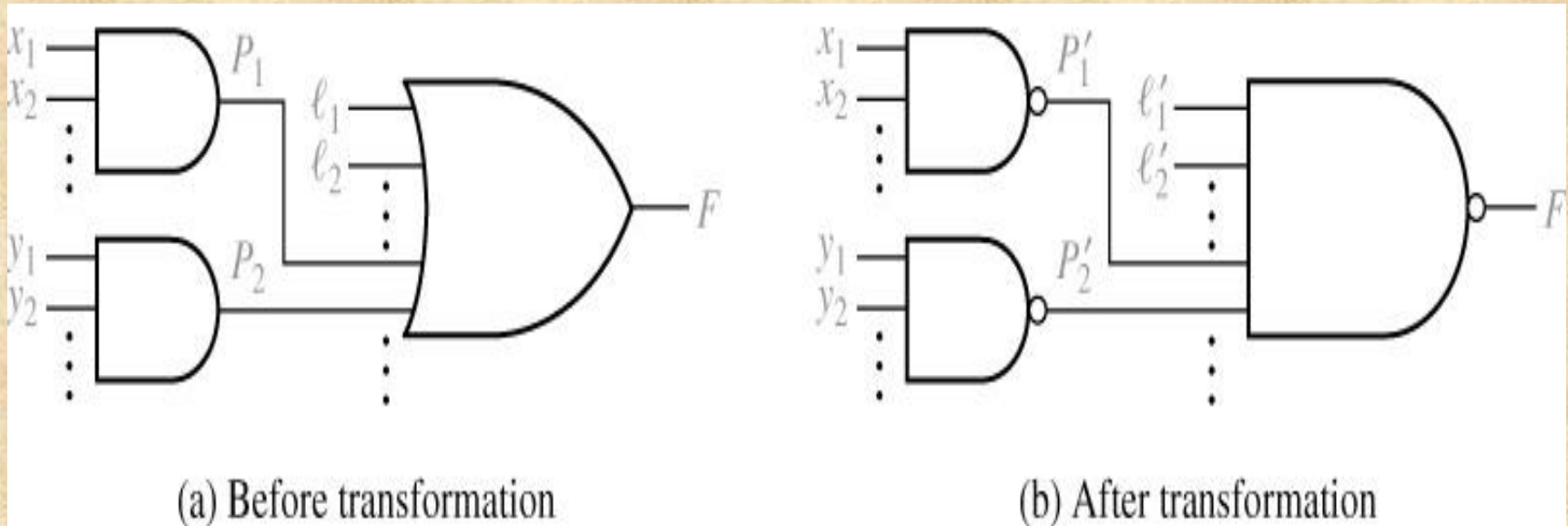
Design of Two-Level Circuits Using NAND and NOR Gates (2/7)

- $F = A + BC' + B'CD$
- $= [(A + BC' + B'CD)']'$ (F')'
- $= [A'(BC')'(B'CD)']'$ NAND-NAND
- $= [A'(B'+C)(B+C'+D')']'$ OR-NAND
- $= A + (B'+C)' + (B+C'+D')'$ NOR-OR
- We started with the minimum sum-of-products expression.

Design of Two-Level Circuits Using NAND and NOR Gates (3/7)

- Procedure for designing a minimum two-level NAND-NAND circuit
 - Find a minimum sum-of-products expression for F : Karnaugh maps, Quinne-McCluskey and Petrick methods
 - Draw the corresponding two-level AND-OR circuit
 - Replace all gates with NAND gates leaving the gate interconnections unchanged.
 - If the output gate has any single literals as inputs, complement these literals.

Design of Two-Level Circuits Using NAND and NOR Gates (4/7)



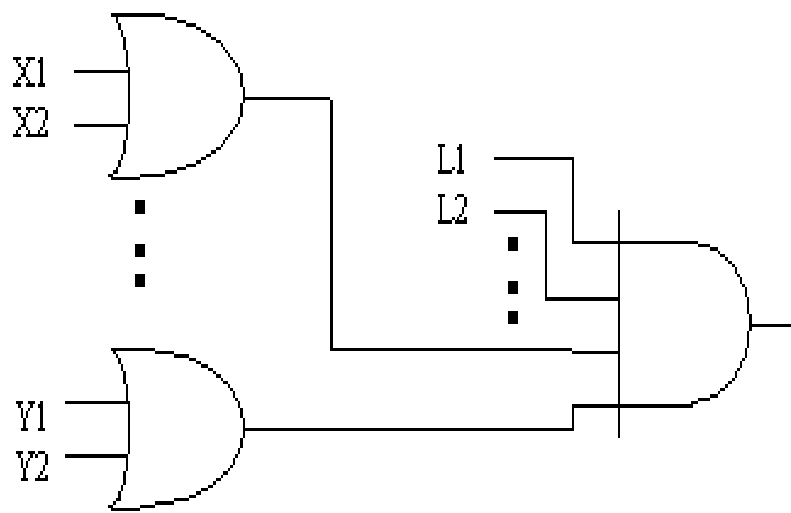
Design of Two-Level Circuits Using NAND and NOR Gates (5/7)

- $F=(A+B+C)(A+B'+C')(A+C'+D)$
- $=\{[(A+B+C)(A+B'+C')(A+C'+D)]'\}'$ (F')'
- $=[(A+B+C)'+(A+B'+C')'+(A+C'+D)']'$ **NOR-NOR**
- $=(A'B'C'+A'BC+A'CD)'$ AND-NOR
- $=(A'B'C')'(A'BC)'(A'CD)'$ NAND-AND
- We started with the minimum product-of-sums expression.

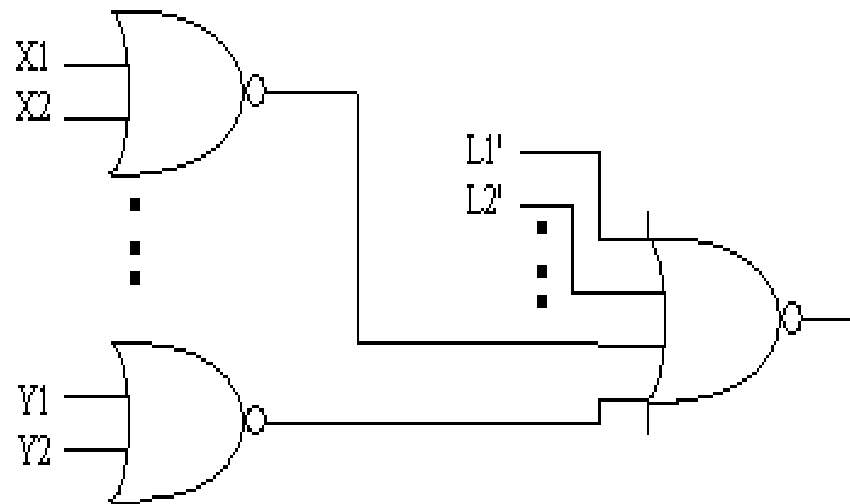
Design of Two-Level Circuits Using NAND and NOR Gates (6/7)

- Procedure for designing a minimum two-level NOR-NOR circuit
 - Find a minimum product-of-sums expression for F
 - Draw the corresponding two-level OR-AND for F
 - Replace all gates with NOR gates leaving the gate interconnections unchanged.
 - If the output gate has any single literals as inputs, complement these literals.

Design of Two-Level Circuits Using NAND and NOR Gates (7/7)



(a) Before Transformation

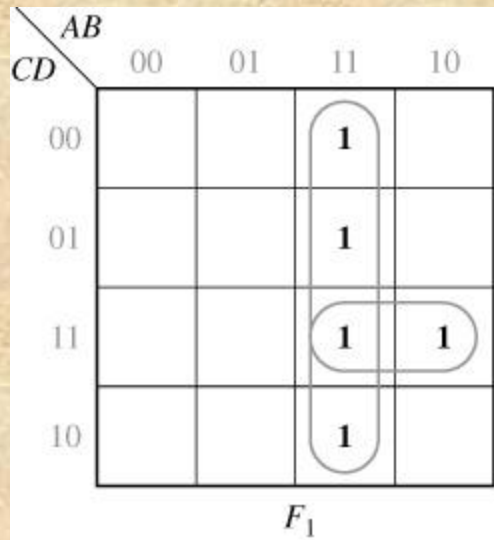


(b) After Transformation

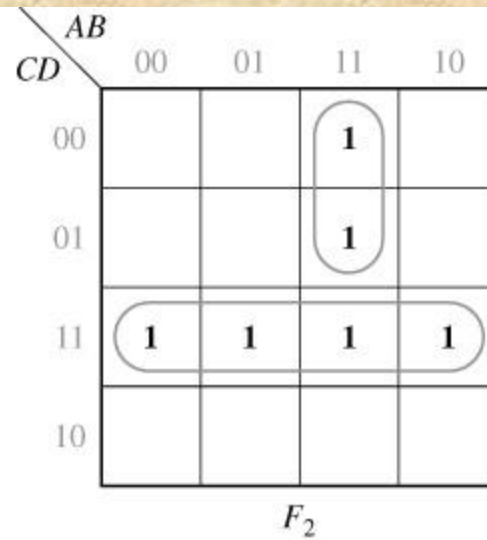
Design of Two-Level, Multiple-Output Circuits (1/10)

- Given a logic function, we can simplify it using some methods.
- But if we need to design a circuit to implement several functions, it may not be enough to simplify each function separately.
- Example:
 - $F_1(A,B,C,D)=\text{sum}[m(11,12,13,14,15)]$
 - $F_2(A,B,C,D)=\text{sum}[m(3,7,11,12,13,15)]$
 - $F_3(A,B,C,D)=\text{sum}[m(3,7,12,13,14,15)]$

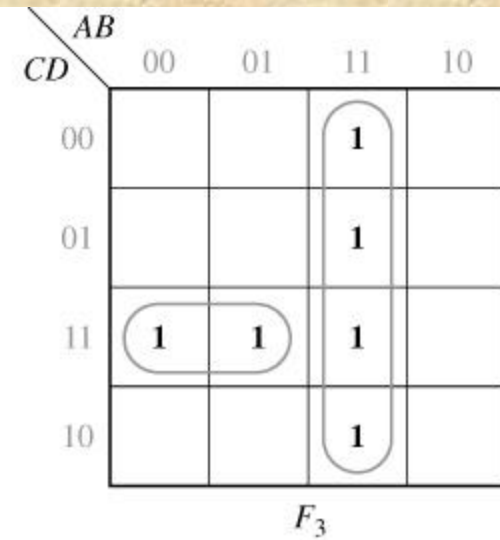
Design of Two-Level, Multiple-Output Circuits (2/10)



$$F_1 = AB + ACD$$

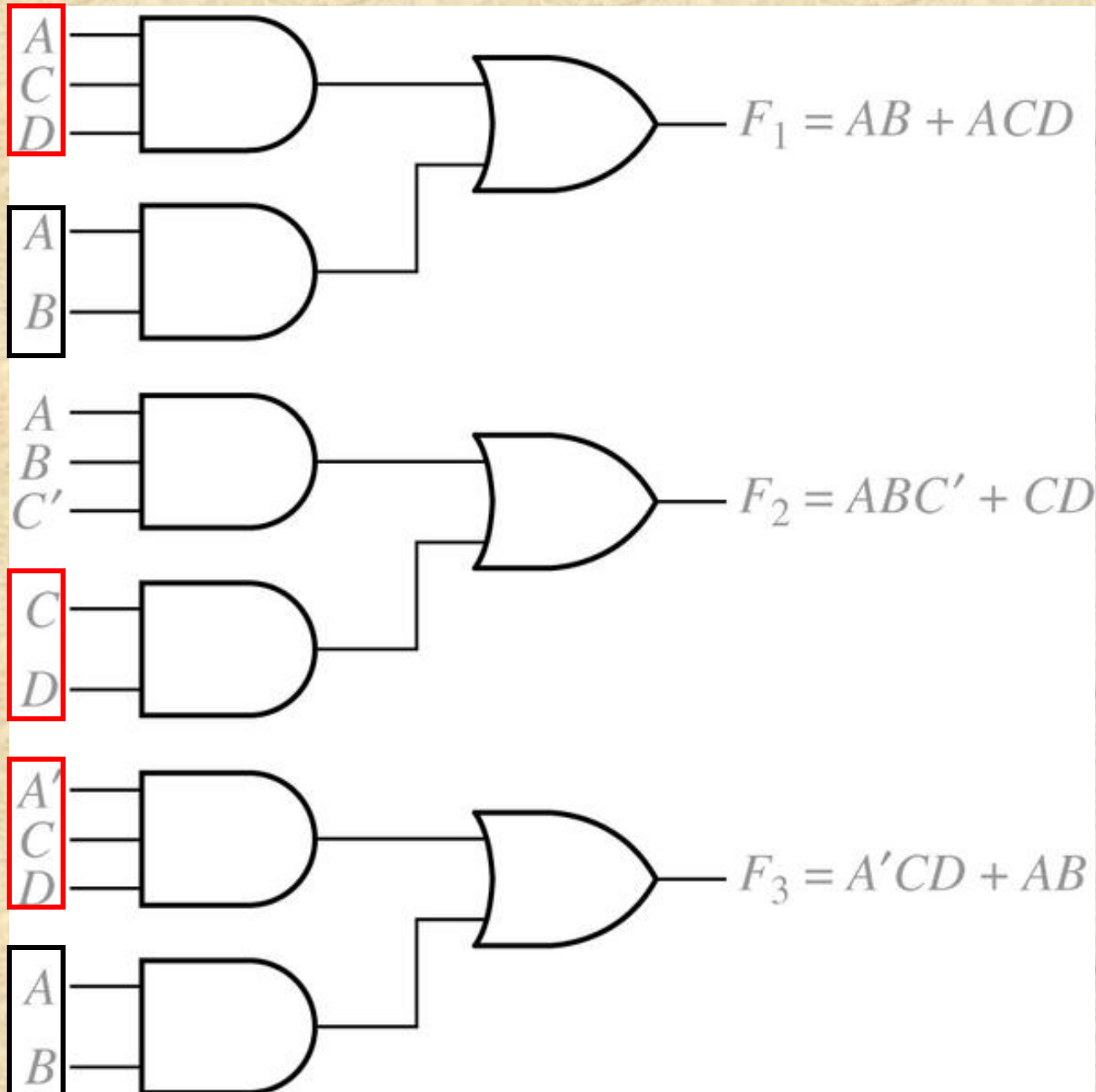


$$F_2 = ABC' + CD$$

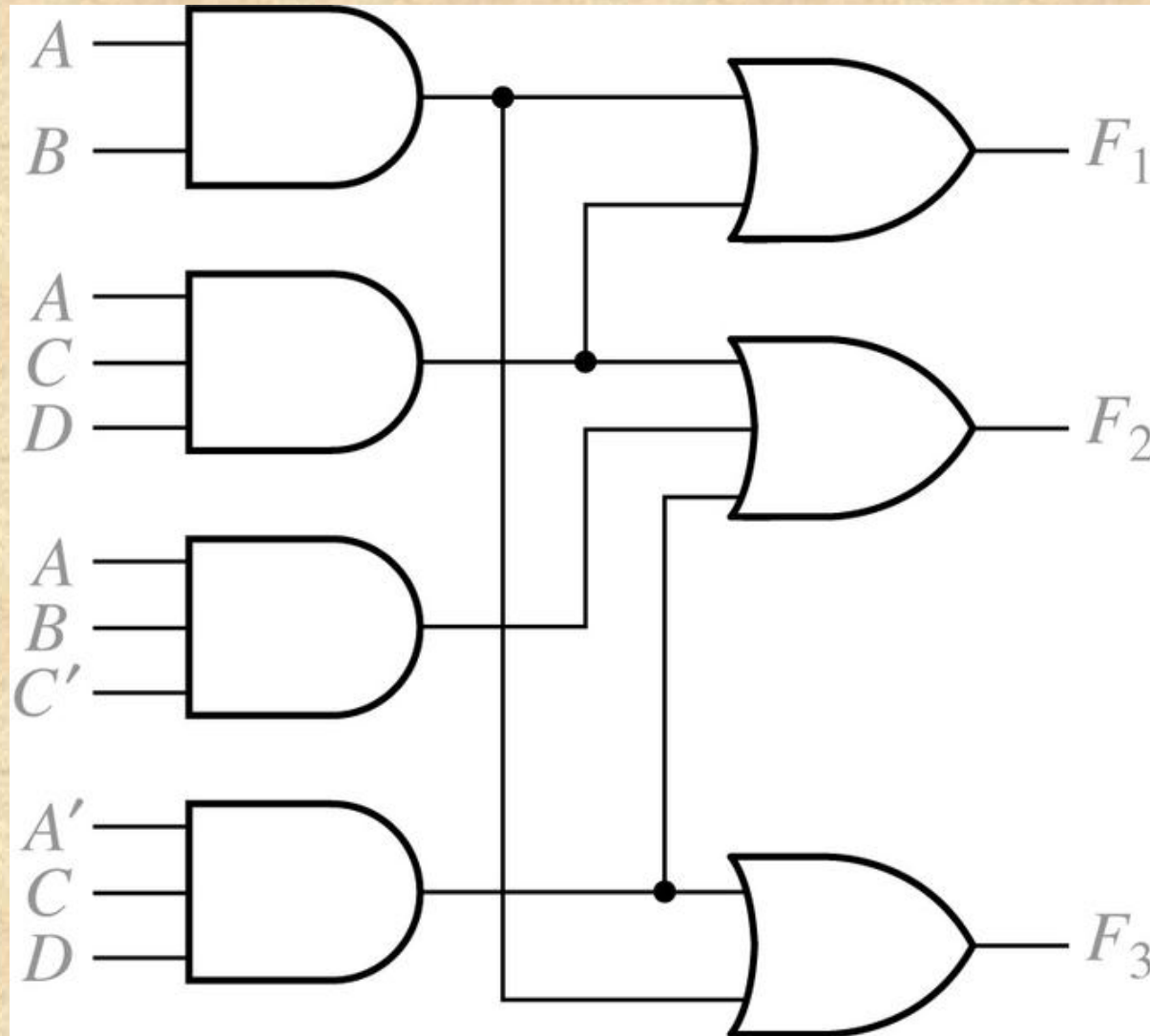


$$F_3 = A'CD + AB$$

Design of Two-Level, Multiple-Output Circuits (3/10)



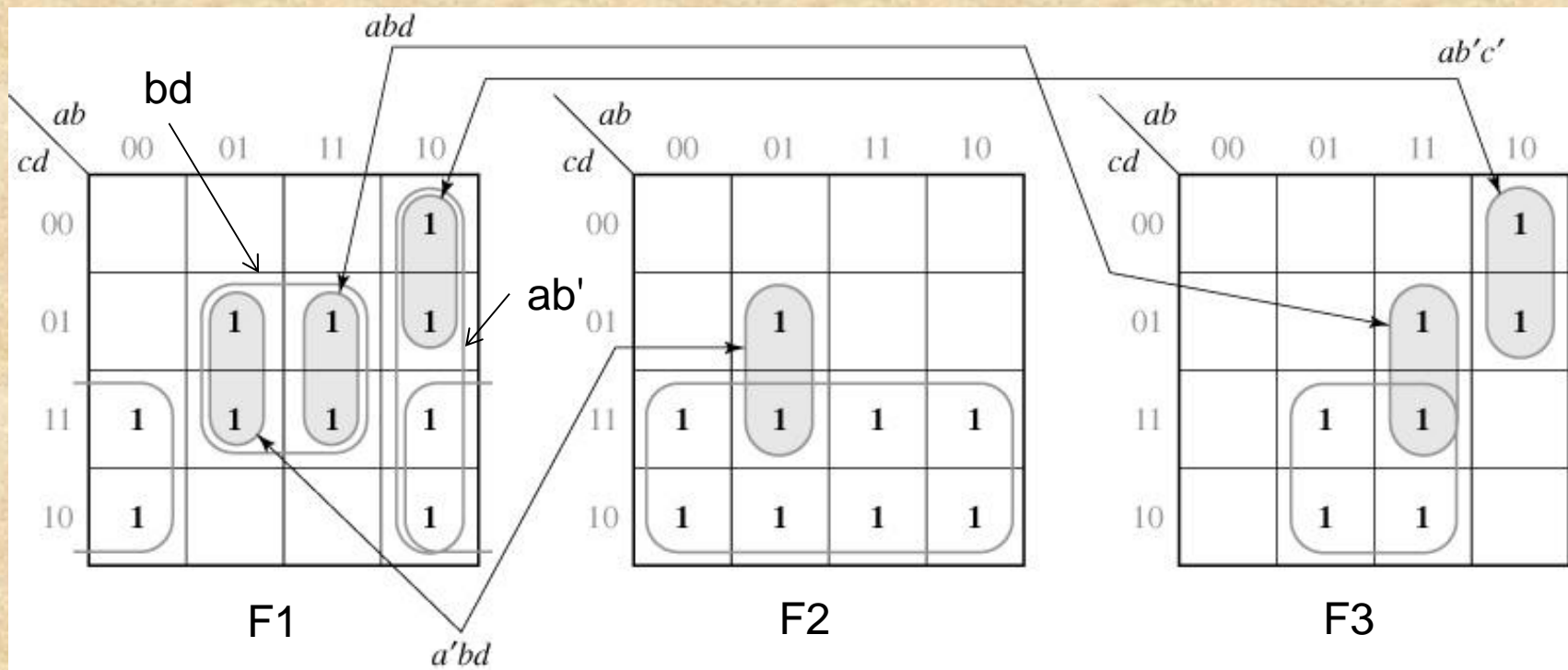
Design of Two-Level, Multiple-Output Circuits (4/10)



Design of Two-Level, Multiple-Output Circuits (5/10)

- Determination of Essential Prime Implicants for Multiple-Output Realization
 - Some of the prime implicants essential to an individual function may not be essential to the multiple-output realization.

Design of Two-Level, Multiple-Output Circuits (6/10)

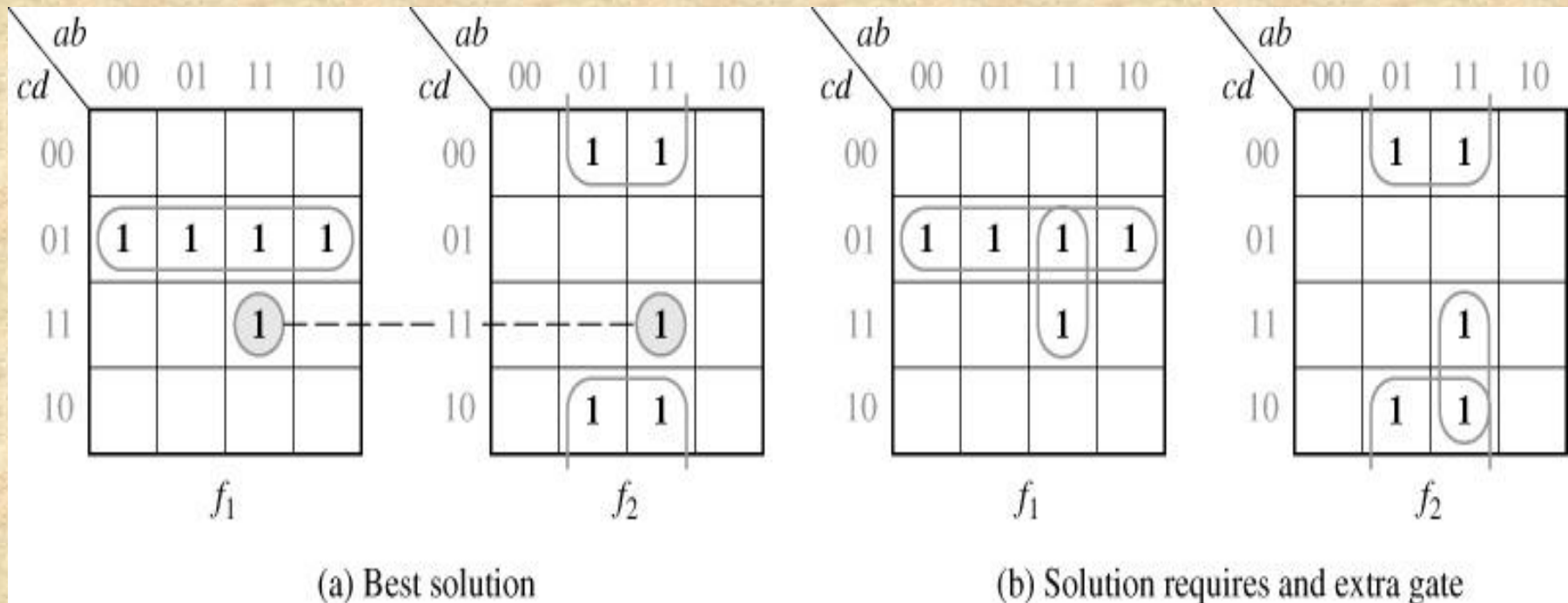


ab' and bd are essential to F1, but not to the multiple-output system.

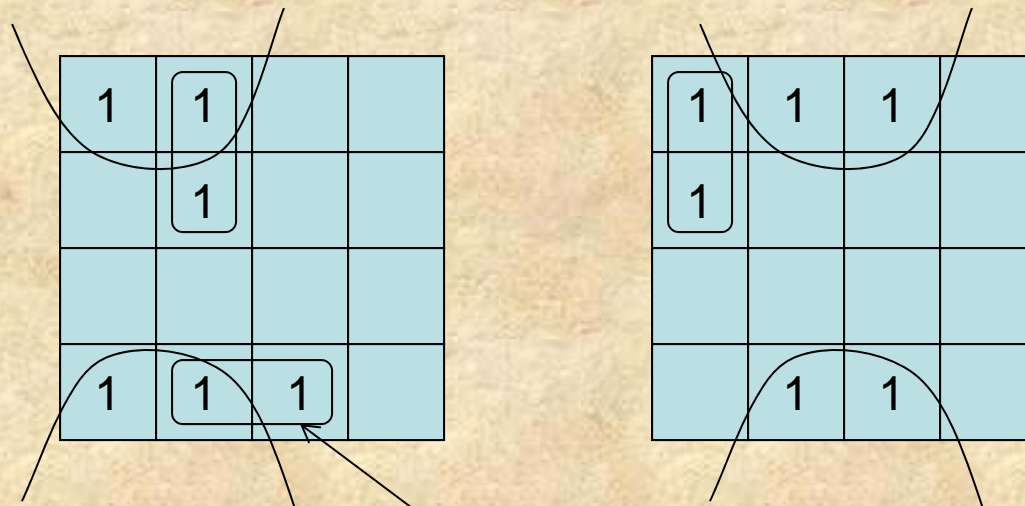
Design of Two-Level, Multiple-Output Circuits (7/10)

- Determination of Essential Prime Implicants for Multiple-Output Realization
 - Some of the prime implicants essential to an individual function may not be essential to the multiple-output realization.
 - When checking 1's for essential prime implicants, we only check those 1's which do not appear on the other function maps.

Design of Two-Level, Multiple-Output Circuits (8/10)

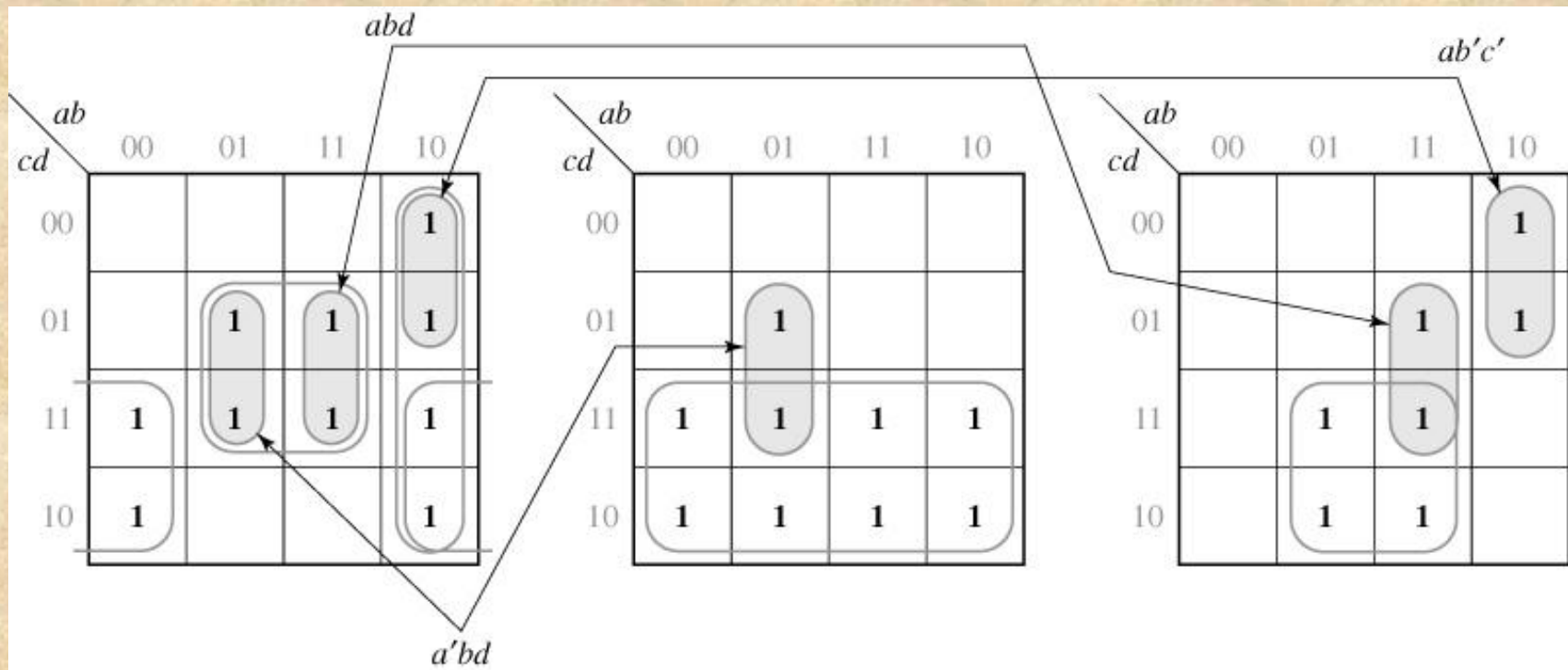


Design of Two-Level, Multiple-Output Circuits (9/10)



When checking 1's for essential prime implicants, we only check those 1's which do not appear on the other function maps.

Design of Two-Level, Multiple-Output Circuits (10/10)



- In this picture, no minterm only appears in one Karnaugh map.
- How to find essential prime implicants for a multi-output system is not discussed in the textbook.