Unit 7. Multi-Level Gate Circuits / NAND and NOR Gates

Spring 2015

School of Electrical Engineering

Prof. Jong-Myon Kim

Objectives – To Learn

Topics introduced in this chapter:

- Design a minimal two-level or multi-level circuit
- Design or analyze a two-level gate circuit
- Design or analyze a multi-level gate circuit
- Convert circuits by adding or deleting inversion bubbles
- Design a minimal two-level or multiple-output circuit using Karnaugh maps

Multi-Level Gate Networks

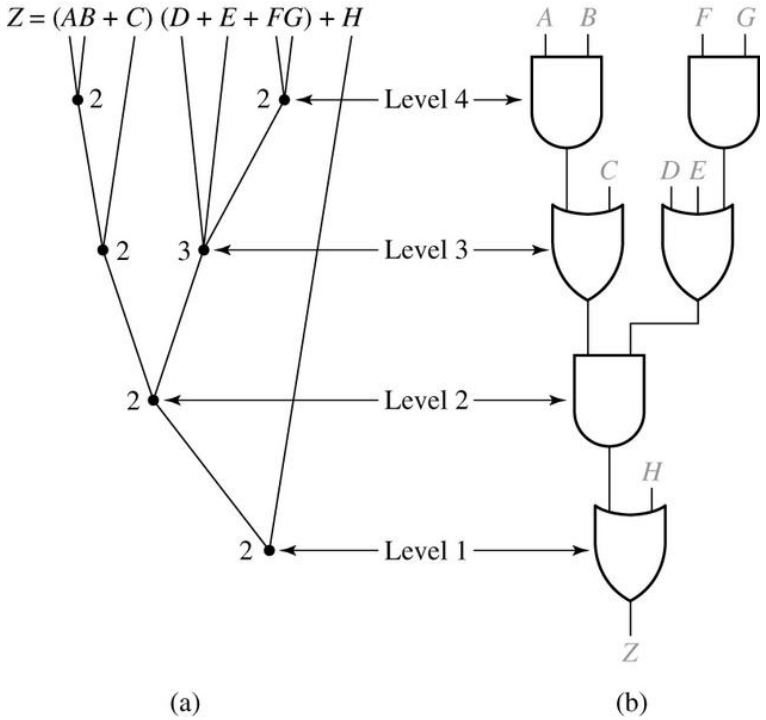
- Number of levels of gates : maximum number of gates cascaded in series
- SOP, POS : two-level networks (assume complement form is available, do not count inverters)
- Networks
  - AND-OR
  - OR-AND
  - OR-AND-OR
  - network of AND and OR gates : no particular ordering
- Trade-off
  - increase the number of levels → reduce # gates (reduce costs. But not always) may slow down the operation
- In many applications, the number of gates which can be cascaded is limited by gate delays

Multi–Level Gate Circuits

- **Terminology**

  AND-OR, OR-AND, OR-AND-OR, AND and OR

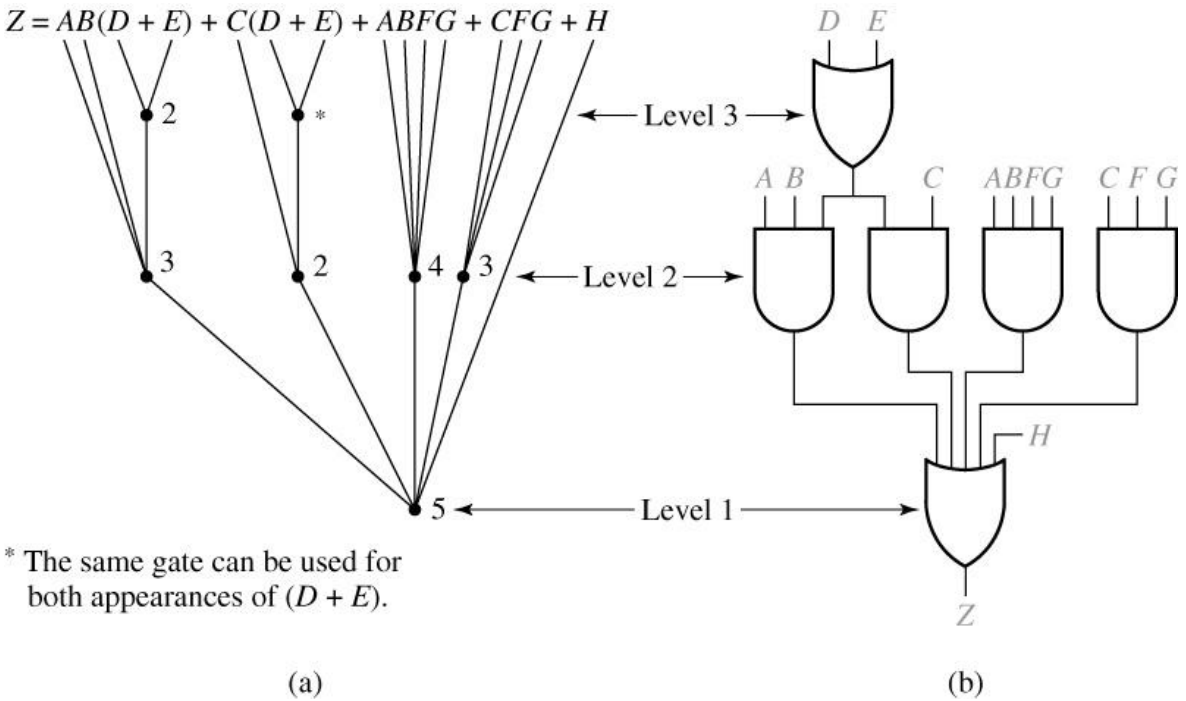
Four-Level Realization of Z



Multi–Level Gate Circuits

Three-Level Realization of Z

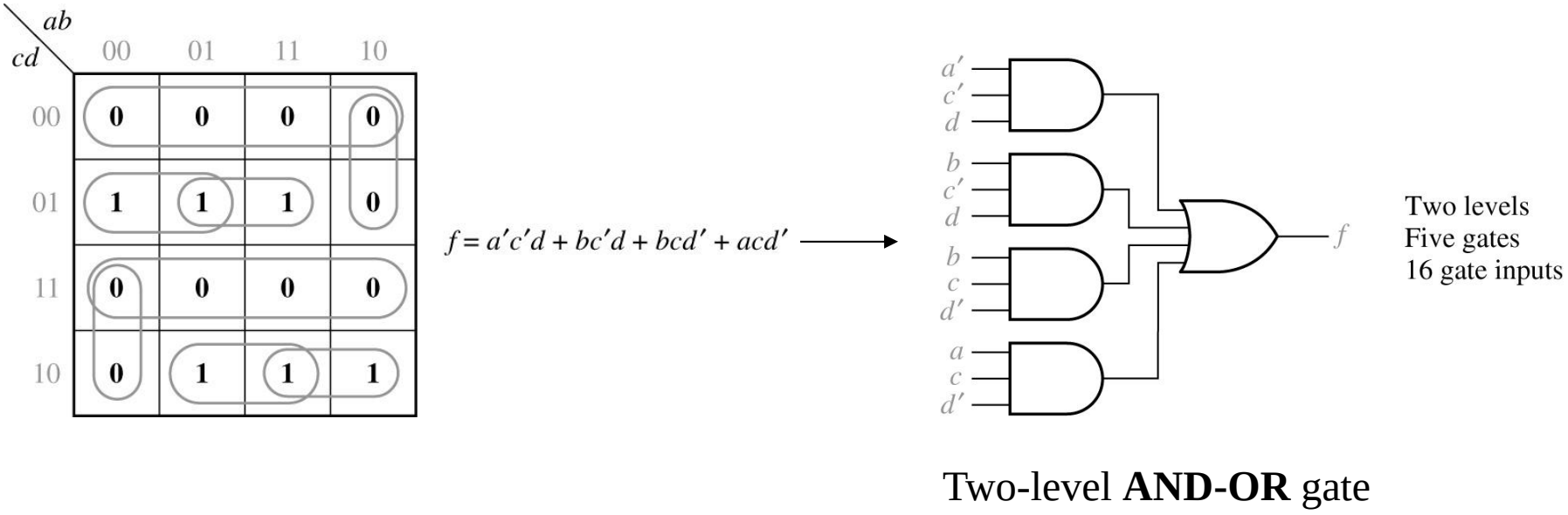
$$Z = (AB + C)[(D + E) + FG] + H$$
$$= AB(D + E) + C(D + E) + ABFG + CFG + H$$



Multi–Level Gate Circuits

Example : Multi-Level Design Using AND and OR Gates

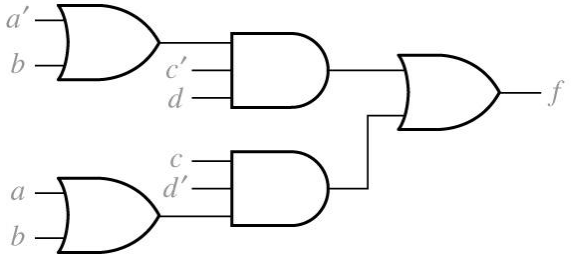
$$f(a,b,c,d) = \sum m(1,5,6,10,13,14)$$



Two-level **AND-OR** gate

Multi–Level Gate Circuits

$$f = a'c'd + bc'd + bcd' + acd'$$
$$= c'd(a' + b) + cd'(a + b)$$



Three levels
Five gates
12 gate Inputs

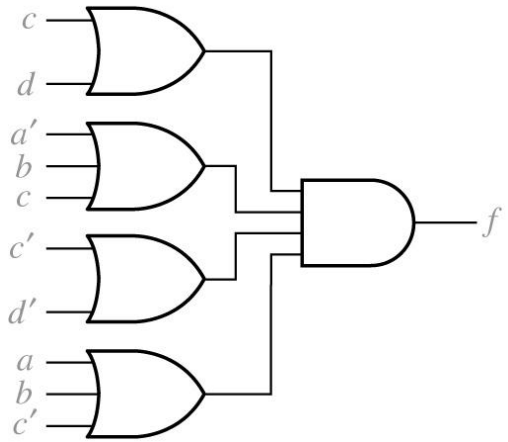
Three-level **OR-AND-OR** gate

From 0's on the Karnaugh map

$$f' = c'd' + ab'c' + cd + a'b'c$$

$$\downarrow f''$$

$$f = (c + d)(a' + b + c)(c' + d')(a + b + c')$$



Two levels
Five gates
14 gate inputs

Two-level **OR-AND** gate

Multi–Level Gate Circuits

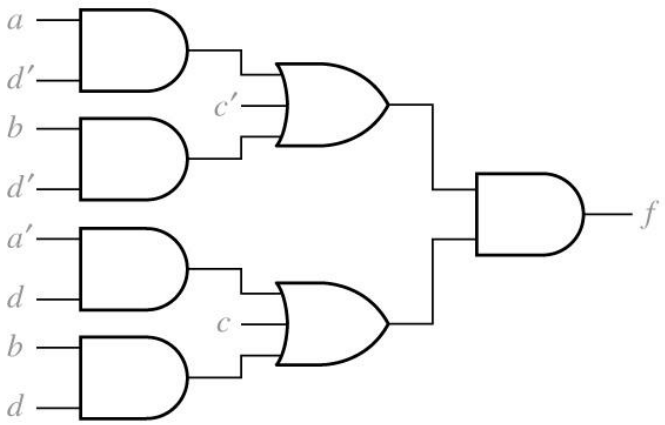
Using $X'Y + XZ = (X + Y)(X' + Z)$

$$f = [c + d(a' + b)][c' + d'(a + b)]$$

If we multiply out $d'(a + b)$ and $d(a' + b)$

$$f = (c + a'd + bd)(c' + ad' + bd')$$

$$f' = c'(d' + ab') + c(d + a'b')$$
$$= c'(d' + a)(d' + b') + c(d + a')(d + b')$$



Three levels
Seven gates
16 gate inputs

Three-level **AND-OR-AND** gate

NAND / NOR Gates

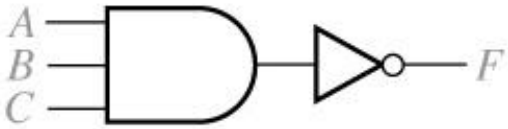
- Gates used up to this point
  - AND, OR, inverter, XOR, Equivalence gates
- NAND & NOR gates
  - can be constructed using transistor logic
  - commonly available in IC form
  - frequently used since they are generally faster and use fewer components than AND or OR gates
  - Any logic function can be implemented using only NAND gates or only NOR gates

NAND Gates

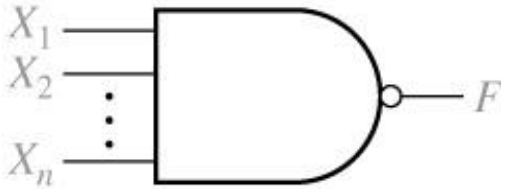
- NAND : AND followed by an inverter
- output of NAND is 1 iff one or more of its inputs are 0

- F = (ABC)' = A'+B'+C'
- F = $(X_1X_2X_3 \ldots X_n)' = X_1'+X_2'+X_3'+ \ldots + X_n'$

(a) 3-input NAND gate



(b) NAND gate equivalent

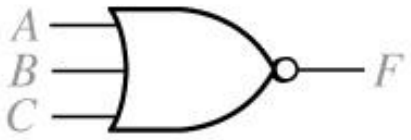


(c) *n*-input NAND gate

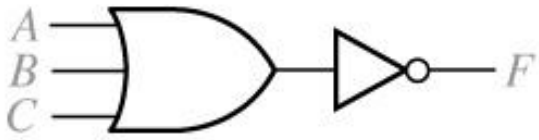
NOR Gates

- NOR : OR followed by an inverter
- output of NOR is 1 iff all inputs are 0

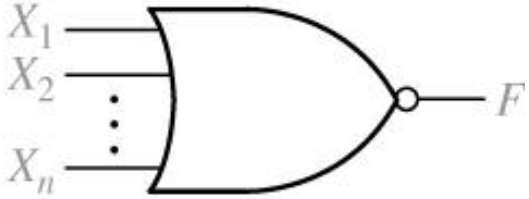
- F = (A+B+C)' = A'B'C'
- F = $(X_1+X_2+ \ldots + X_n)' = X_1'X_2' \ldots X_n'$



(a) 3-input NOR gate



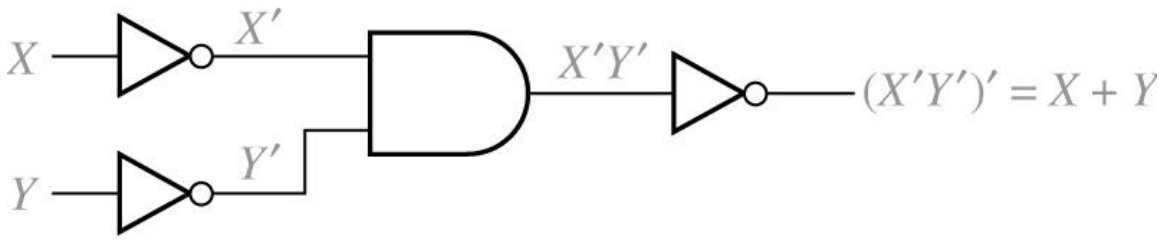
(b) NOR gate equivalent



(c) *n*-input NOR gate

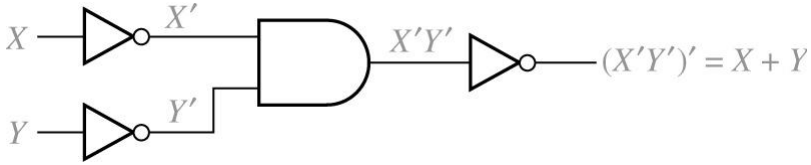
Functionally Complete Set

- 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
- AND, OR, NOT
- AND, NOT (OR can be made from AND, NOT)
- NAND
- OR, NOT

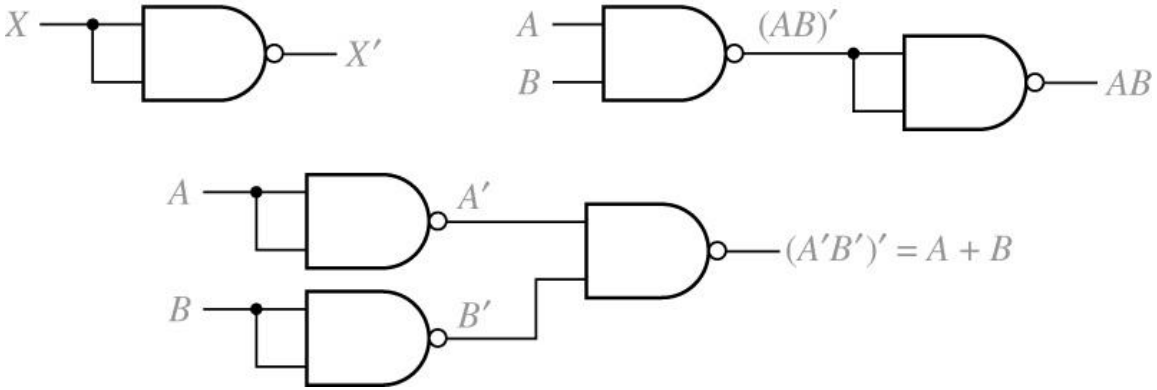


Functionally Complete Set

OR gate realization using NOT, AND



using NAND gates to realize NOT, AND, OR



AND realization using OR and NOT

$$XY = \left(X' + Y'\right)'$$

Design of Two Level Circuits using NAND and NOR Gates

DeMorgan's laws

$$\left(X_1 + X_2 + \ldots + X_n\right)' = X_1{}' X_2{}' \ldots X_n{}'$$

$$\left(X_1 X_2 \ldots X_n\right)' = X_1{}' + X_2{}' + \ldots + X_n{}'$$

Conversion of a sum-of-products to several other two-level forms

$$F = A + BC' + B'CD = \left[\left(A + BC' + B'CD\right)'\right]' \qquad \text{AND–OR}$$

$$= \left[A' \cdot \left(BC'\right)' \cdot \left(B'CD\right)'\right]' \qquad \text{NAND–NAND}$$

$$= \left[A' \cdot \left(B' + C\right) \cdot \left(B + C' + D'\right)\right]' \qquad \text{OR–NAND}$$

$$= A + \left(B' + C\right)' + \left(B + C' + D'\right)' \qquad \text{NOR–OR}$$

$$F = \left\{\left[A + \left(B' + C\right)' + \left(B + C' + D'\right)'\right]'\right\}' \qquad \text{NOR–NOR–INVERT}$$

Design of Two Level Circuits using NAND and NOR Gates

$$F = (A+B+C)(A+B'+C')(A+C'+D) \qquad \text{OR–AND}$$

$$= \left\{ \left[(A+B+C)(A+B'+C')(A+C'+D) \right]' \right\}'$$

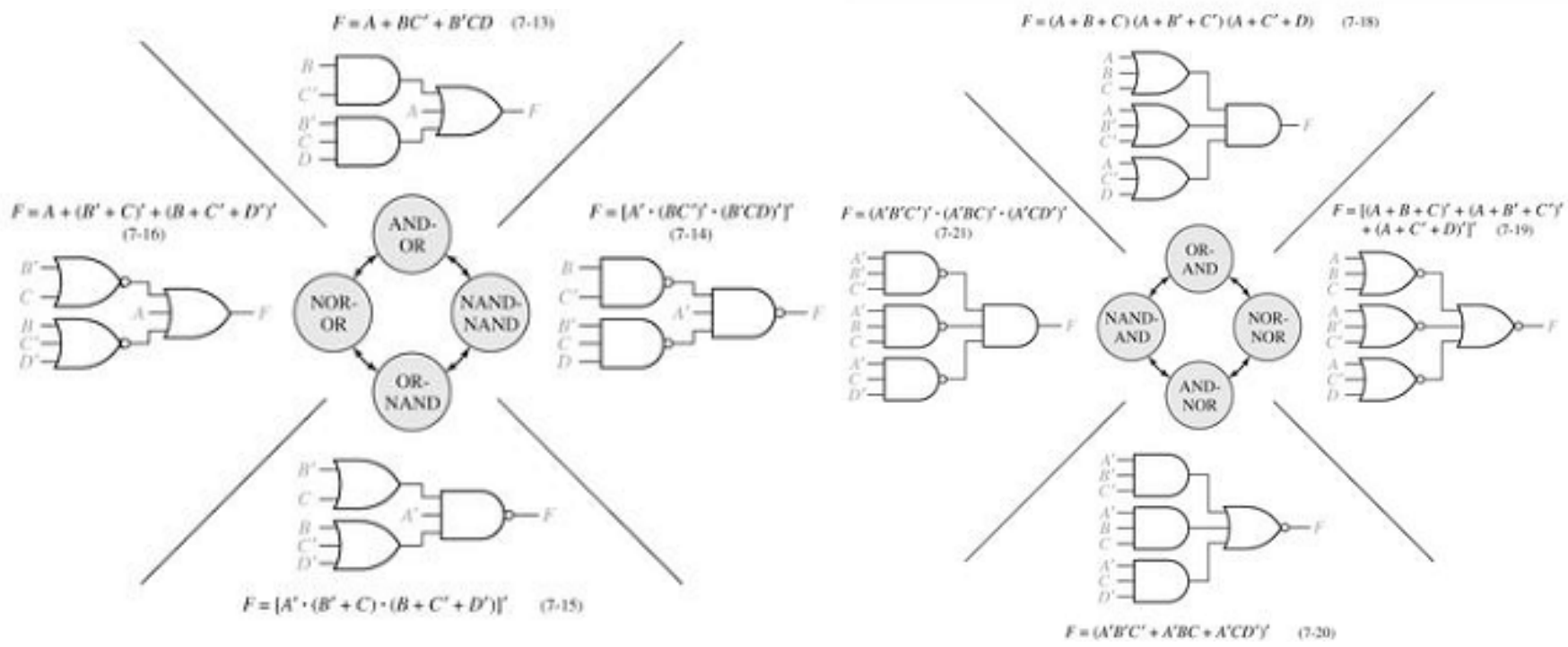
$$= \left[(A+B+C)' + (A+B'+C')' + (A+C'+D)' \right]' \qquad \text{NOR–NOR}$$

$$= (A'B'C' + A'BC + A'CD')' \qquad \text{AND–NOR}$$

$$= (A'B'C')' \cdot (A'BC)' \cdot (A'CD')' \qquad \text{NAND–AND}$$

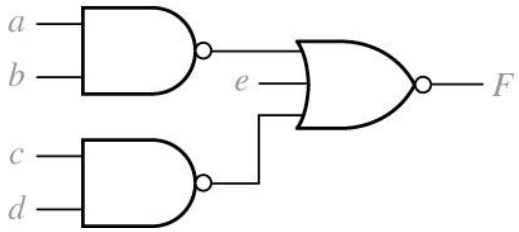
Design of Two Level Circuits using NAND and NOR Gates

Eight Basic Forms for Two-Level Circuits



Design of Two Level Circuits using NAND and NOR Gates

NAND-NOR

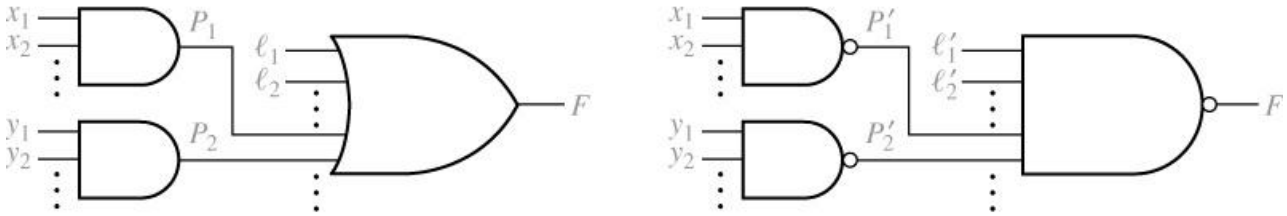


$$F = [(ab)' + (cd)' + e]' = abcde'$$

AND-OR to NAND-NAND Transformation

$(l_1, l_2 \ldots)$: literals $(P_1, P_2 \ldots)$: product terms

$$F = l_1 + l_2 + \ldots + P_1 + P_2 + \ldots = \left(l_1' l_2' \cdots P_1' P_2' \cdots \right)'$$



(a) Before transformation (b) After transformation

Design of Multi-Level NAND- and NOR-Gate Circuits

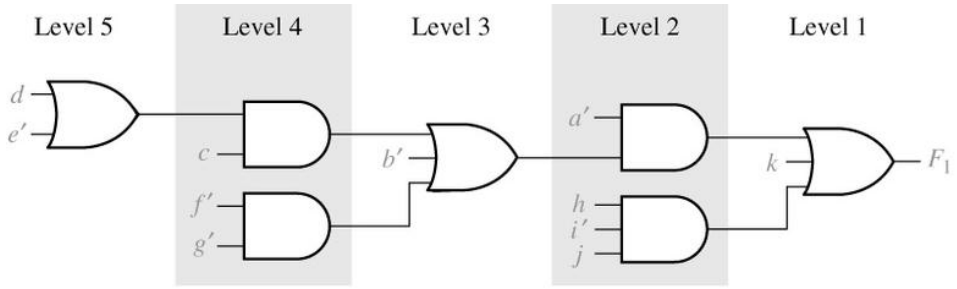
❖ Procedure : multi-level NAND-gate circuits

- Simplify the switching function
- Design a multi-level circuit of AND and OR gates
- Number the levels starting with the output gate as level 1
- Replace all gates with NAND gates, leaving all interconnections between gates unchanged
- Leave the inputs to levels 2,4,6,… unchanged

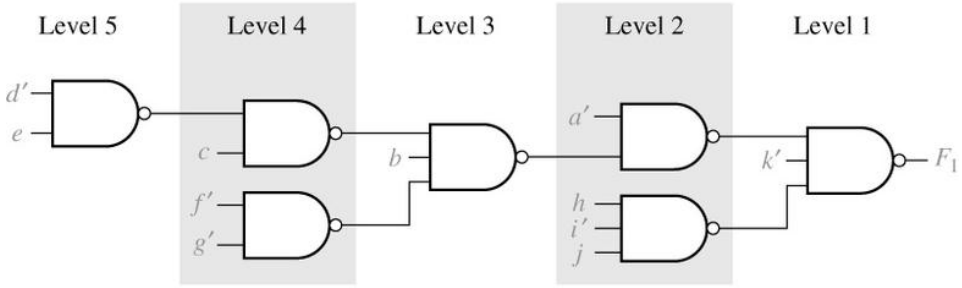
Design of Multi-Level NAND- and NOR-Gate Circuits

Example : Multi-Level Circuit Conversion to NAND Gates

$$F_1 = a'\left[b' + c\left(d + e'\right) + f'g'\right] + hi'j + k$$



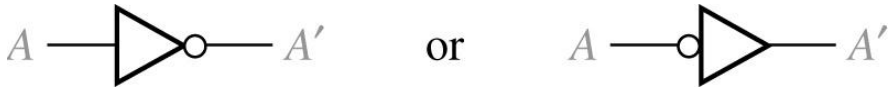
(a) AND-OR network



(b) NAND network

Circuit Conversion Using Alternative Gate Symbols

Inverter



Alternative Gate Symbols

$AB = (A' + B')'$ (a) AND

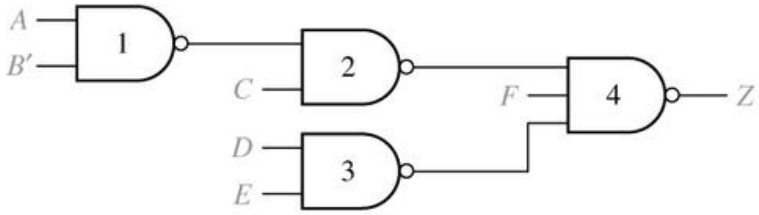
$A + B = (A'B')'$ (b) OR

$(AB)' = A' + B'$ (c) NAND

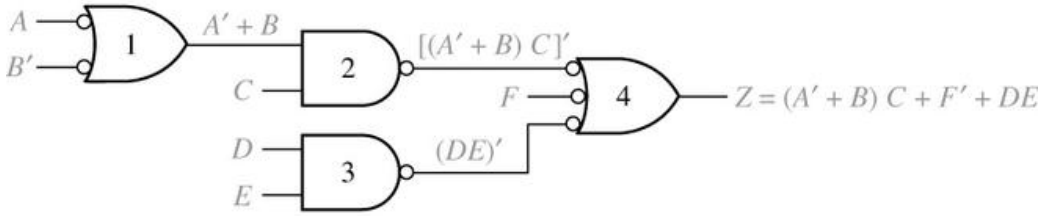
$(A + B)' = A'B'$ (d) NOR

Circuit Conversion Using Alternative Gate Symbols

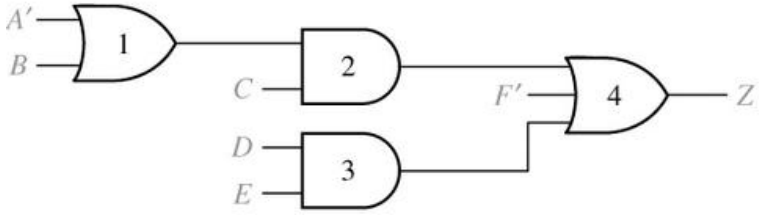
NAND Gate Circuit Conversion



(a) NAND gate network



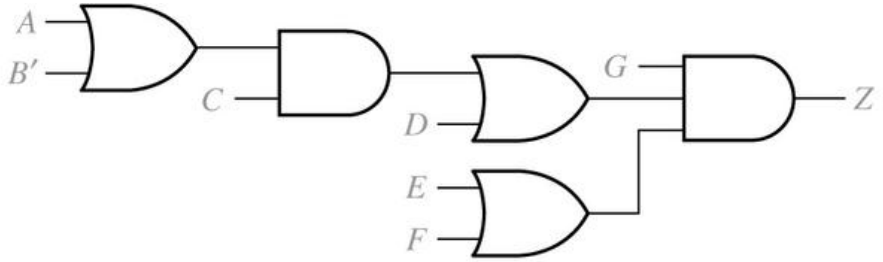
(b) Alternate form for NAND gate network



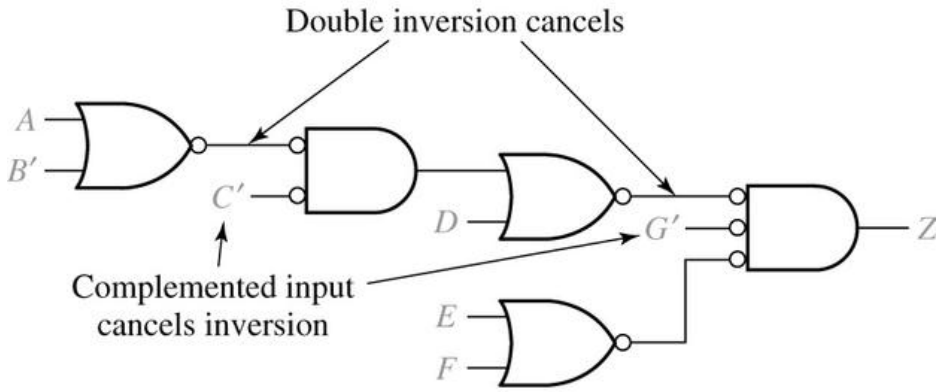
(c) Equivalent AND-OR network

Circuit Conversion Using Alternative Gate Symbols

Conversion to NOR Gates



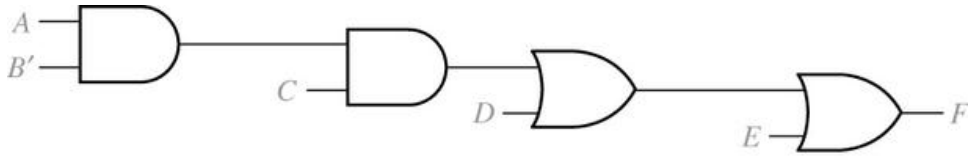
(a) Circuit with OR and AND gates



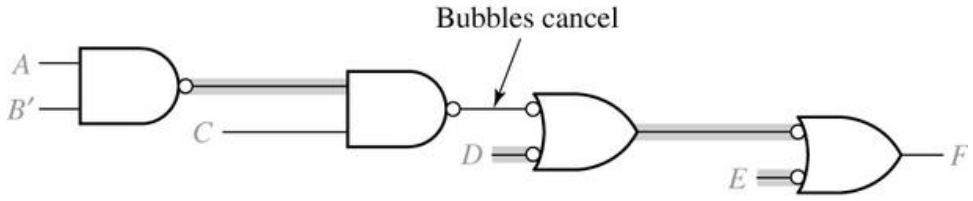
(b) Equivalent circuit with NOR gates

Circuit Conversion Using Alternative Gate Symbols

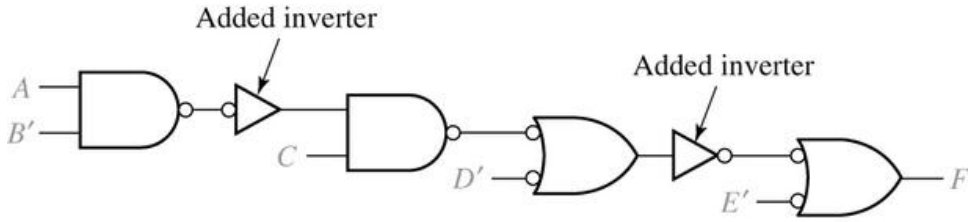
Conversion of AND-OR Circuits to NAND Gates



(a) AND-OR network



(b) First step in NAND conversion



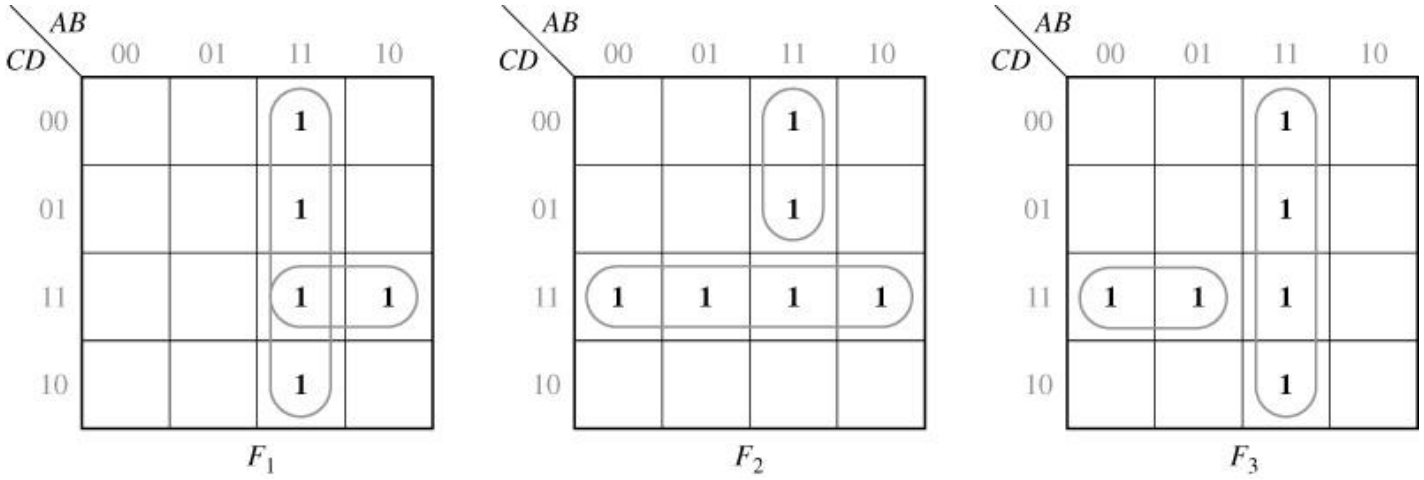
(c) Completed conversion

Design Two-Level, Multiple-Output Circuits

Example : Design a circuit with four inputs and three outputs

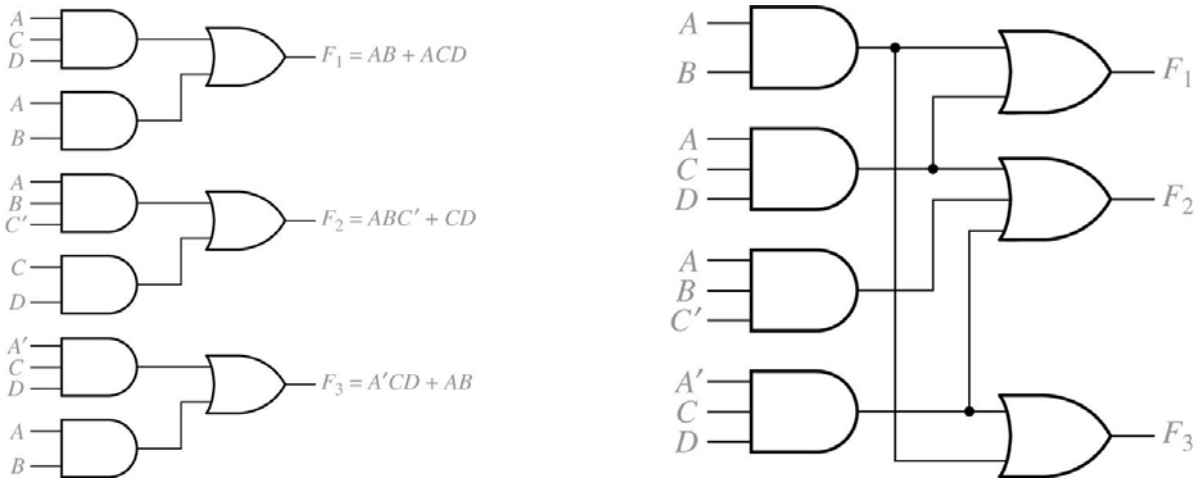
$$F_1(A,B,C,D) = \sum m(11,12,13,14,15)$$
$$F_2(A,B,C,D) = \sum m(3,7,11,12,13,15)$$
$$F_3(A,B,C,D) = \sum m(3,7,12,13,14,15)$$

Karnaugh Maps for Equations



Design Two-Level, Multiple-Output Circuits

- 9 gates, 21 gate inputs
- Further simplification?
  - AB
  - ACD + A'CD → CD
- Use of some gates in common between two or more functions sometimes leads to a more economical realization



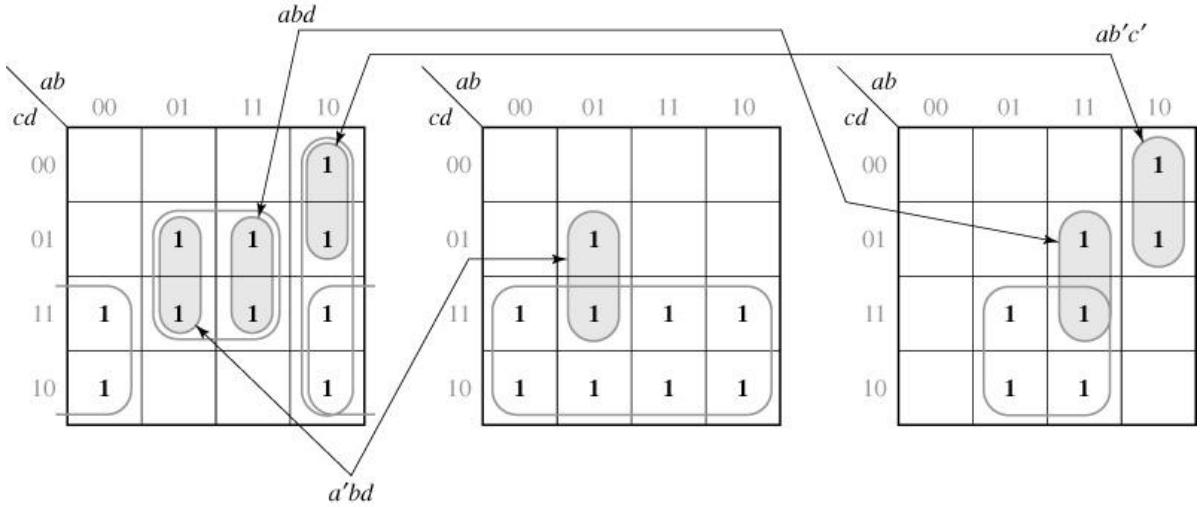
- 7 gates, 18 gate inputs vs 9 gates, 21 gate inputs
- In realizing multiple-output networks, use of minimum sum of PI for each function does not necessarily lead to a minimum cost solution
- minimize total # gates, min # gate inputs

Design Two-Level, Multiple-Output Circuits

Example : Design a multiple-output circuit with 4-inputs and 3-outputs

$$f_1 = \sum m(2,3,5,7,8,9,10,11,13,15)$$
$$f_2 = \sum m(2,3,5,6,7,10,11,14,15)$$
$$f_3 = \sum m(6,7,8,9,13,14,15)$$

Karnaugh Maps for Equations



Design Two-Level, Multiple-Output Circuits

Minimized equations if each function is minimized separately

$$f_1 = bd + b'c + ab'$$

$$f_2 = c + a'bd$$

$$f_3 = bc + ab'c' + \left\{ \begin{array}{c} abd \\ or \\ ac'd \end{array} \right\}$$

10gates,
25gate input

The minimal solution

$$f_1 = \underline{a'bd} + \underline{abd} + \underline{ab'c'} + b'c$$

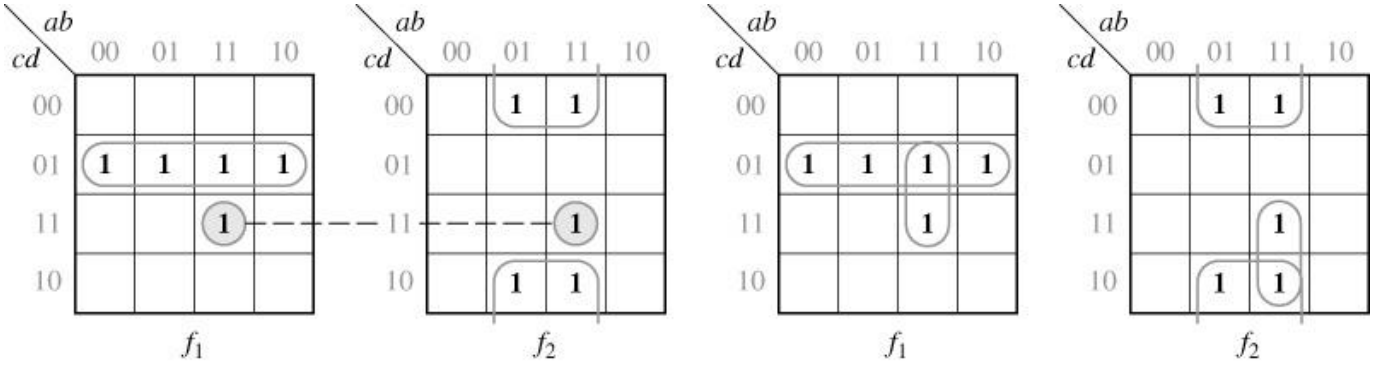
$$f_2 = c + \underline{a'bd}$$

$$f_3 = bc + \underline{ab'c'} + \underline{abd}$$

8 gates
22 gate inputs

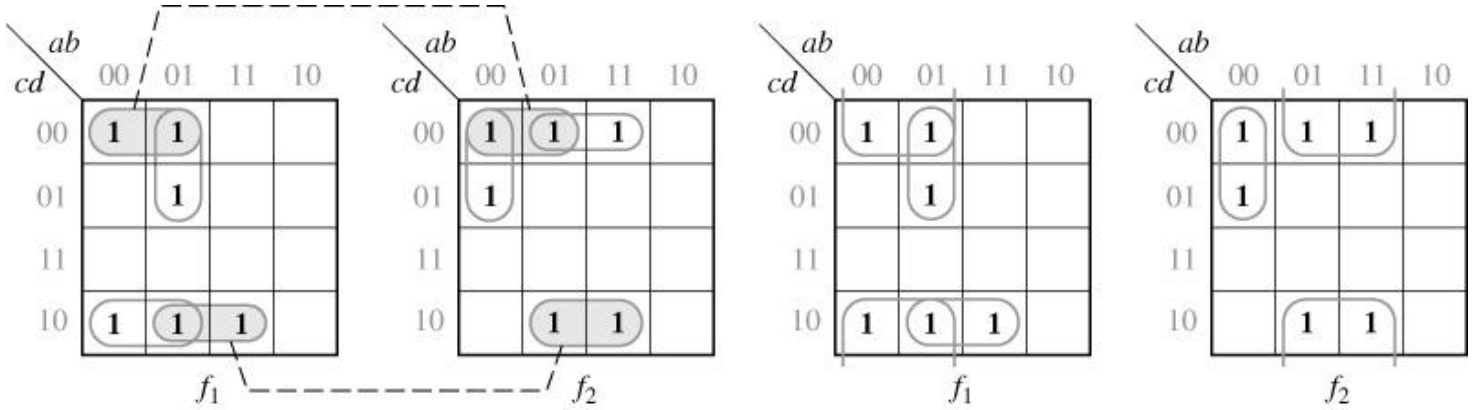
Design Two-Level, Multiple-Output Circuits

Determination of Essential Prime Implicants for Multiple-Output Realization



(a) Best solution

(b) Solution requires and extra gate



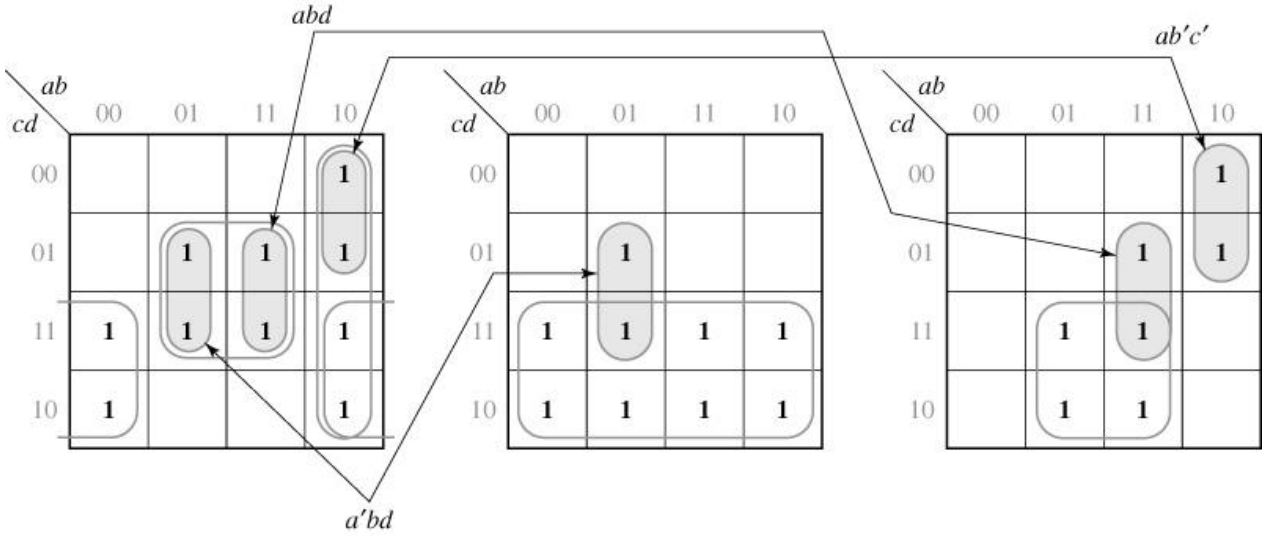
(a) Solution with maximum number of common terms requires 8 gates, 26 inputs

(b) Best solution requires 7 gates, 18 inputs and has no common terms

Design Two-Level, Multiple-Output Circuits

Determination of Essential Prime Implicants for Multiple-Output Realization

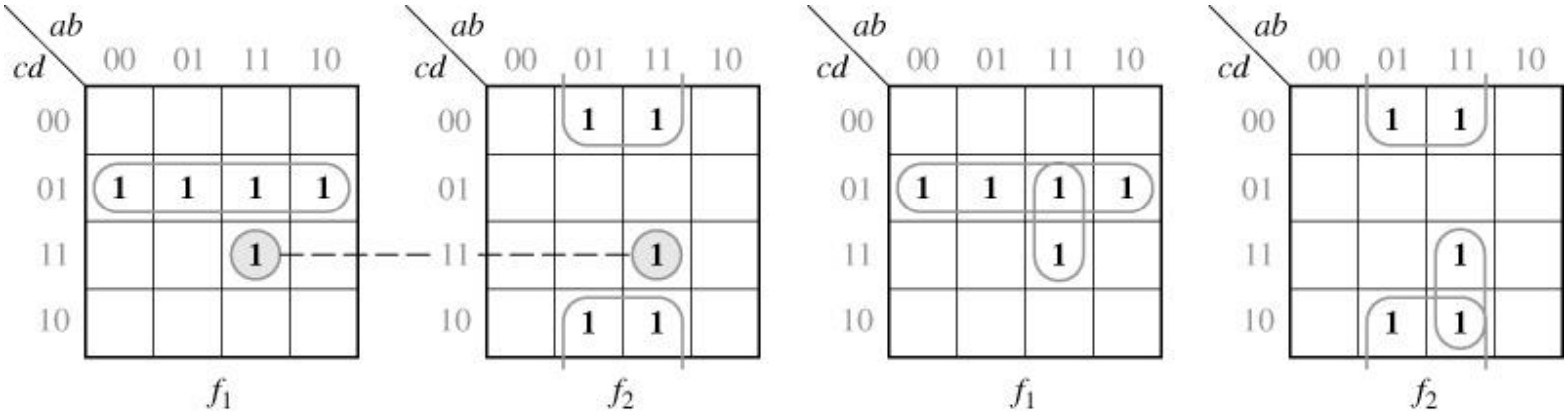
- Some PI essential to an individual function may not be essential to the multiple-output realization
- Example:
  - bd : EPI of f_1 (only PI which covers m_5) but not essential to multiple-output realization since m_5 also appears on the f_2 map (and hence might be covered by a term which is shared by f_1 and f_2)



Design Two-Level, Multiple-Output Circuits

Determination of Essential Prime Implicants for Multiple-Output Realization

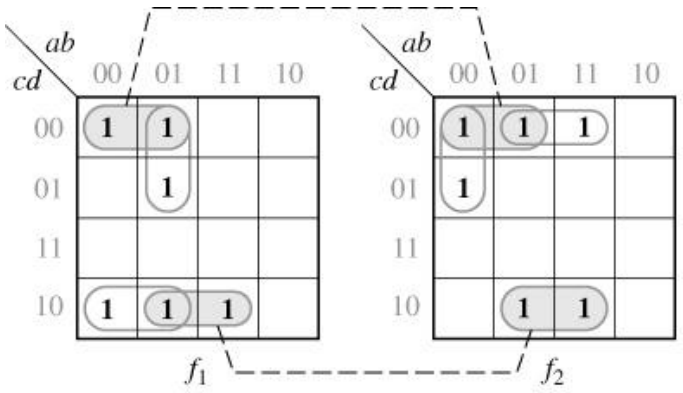
- How can we find PI's which are essential to one of the function and to the multi-output realization?
  - When we check each 1 on the map to see if it is covered by only one PI, we will only check for adjacencies those 1's which do not appear on the other function maps
  - c'd : essential
  - abd : not essential (because m_{15} also appears in f_2)
  - Left : 3 AND, 2 OR Right : 4 AND 2 OR



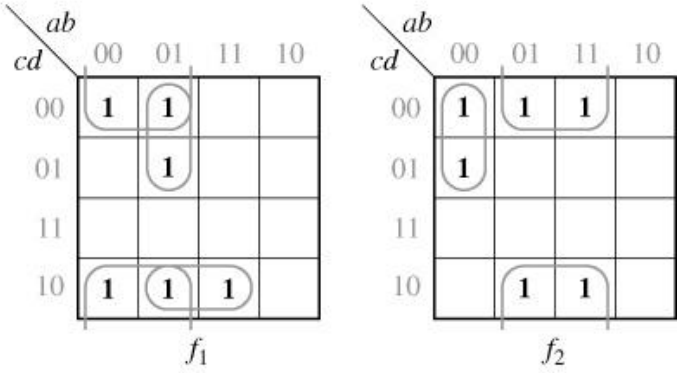
Design Two-Level, Multiple-Output Circuits

Determination of Essential Prime Implicants for Multiple-Output Realization

- How can we find PI's which are essential to one of the function and to the multi-output realization?
  - f_1의 minterm 중 f_2에 나타나지 않는 것은 m_2와 m_5뿐.
  - m_2를 둘러싸는 유일한 PI는 a'd'이므로 이것은 다중출력 구현에서 f_1에 essential
  - m_5를 둘러싸는 유일한 PI는 a'bc'이므로 이것도 essential
  - f_2의 map에서 bd'은 essential. why?



(a) Solution with maximum number of common terms requires 8 gates, 26 inputs



(b) Best solution requires 7 gates, 18 inputs and has no common terms

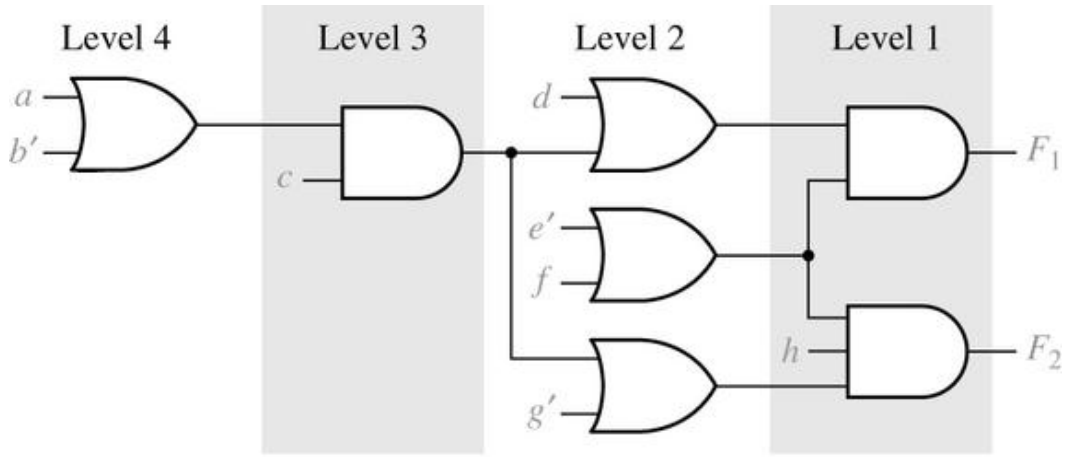
Multiple-Output NAND and NOR Circuits

- If all of the output gates are OR, direct conversion to NAND-gate circuit is possible
- If all of the output gates are AND, direct conversion to NOR-gate circuit is possible

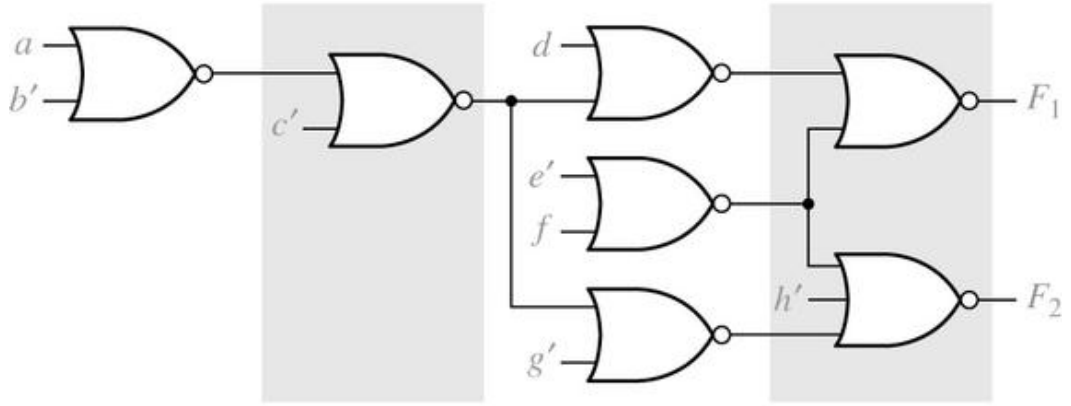
Multi-level Circuits Conversion to NOR Gates

$$F_1 = [(a + b')c + d](e' + f) \qquad F_2 = [(a + b')c + g'](e' + f)h$$

Multiple-Output NAND and NOR Circuits



(a) Network of AND and OR gates



(b) NOR network