Chapter 7. Multi-Level Gate Circuits
NAND and NOR gates

Multi-Level Gate Networks

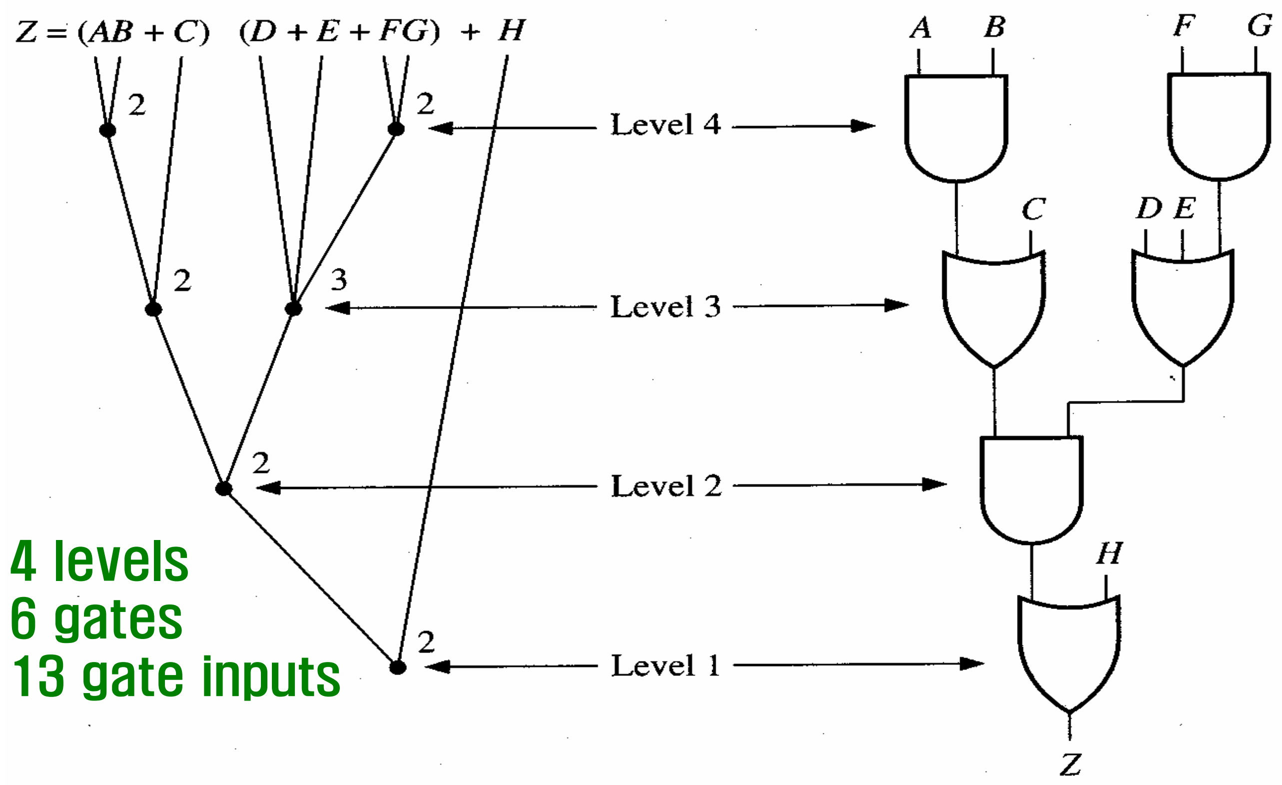
- The maximum number of gates cascaded in series between a network input and the output is referred to as the number of **levels** of gates

- A function written in SOP or POS form corresponds directly to a two-level gate network.

- We will assume that all variables and their complements are available as network inputs. (This is usually the case in digital networks where the gates are driven by flip-flop outputs.)

- Number of levels affects:
  - Number of gates and gate inputs (related to cost)
  - Gate propagation delays

◆Four-Level Realization of Z

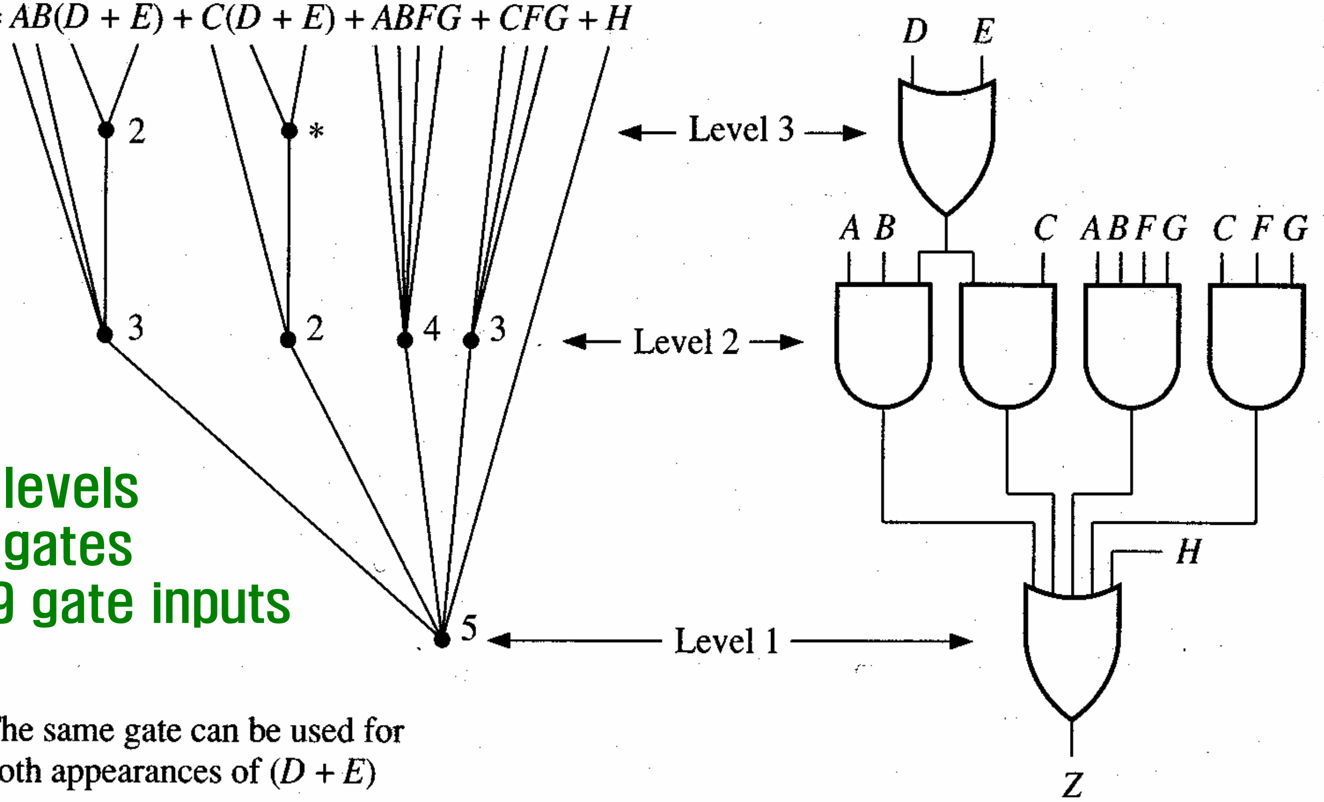


4 levels
6 gates
13 gate inputs

Two Realizations for Z (2)

◆Three-Level Realization of Z

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



3 levels
6 gates
19 gate inputs

* The same gate can be used for both appearances of $(D + E)$

Example of Multi-Level Design using AND and OR Gates

◆ Problem: Find a network of **AND** and **OR** gates to realize

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

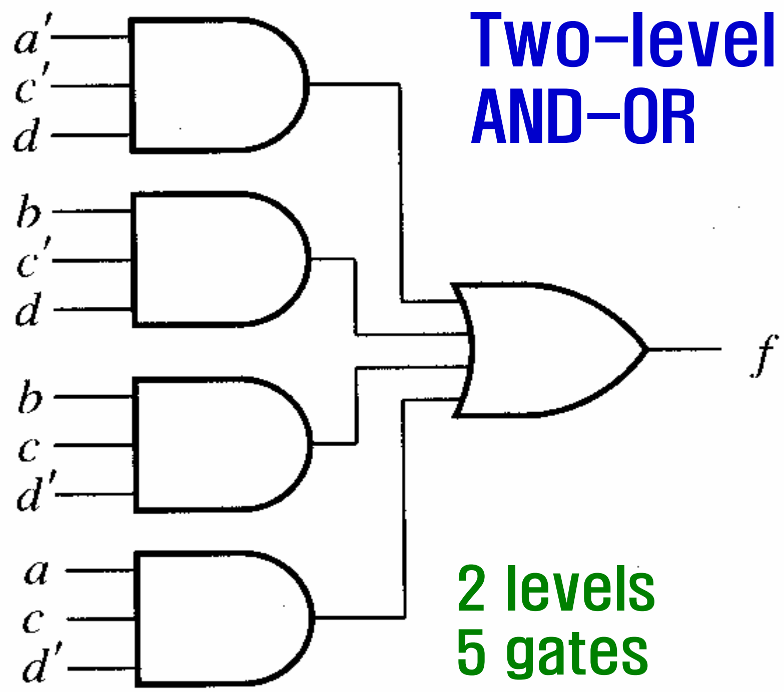
Consider solns. with 2 and **3** gate levels.Try to minimize the number of gates and the total number of gate inputs.

Example of Multi-Level Design using AND and OR Gates

Soln.: First simplify f using K-map

cd \ ab	00	01	11	10
00	0	0	0	0
01	1	1	1	0
11	0	0	0	0
10	0	1	1	1

$$f = a'c'd + bc'd + bcd' + acd' \quad (8.1)$$



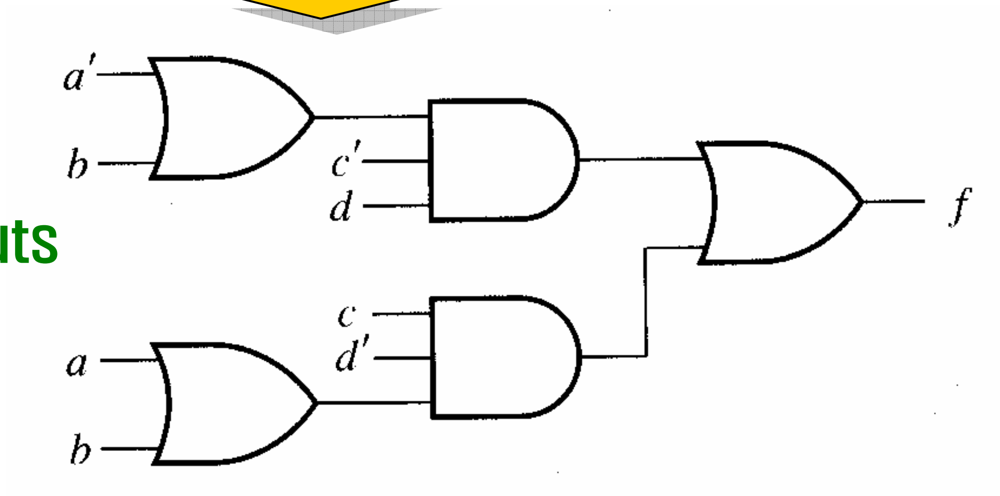
Two-level AND-OR

2 levels
5 gates
16 gate inputs

Example Continued…

Factoring Eqn. (8-1) yields:
$f = c'd(a'+b)+cd'(a+b)$ (8.2)

3 levels
5 gates
12 gate inputs



*Three-level OR-AND-OR

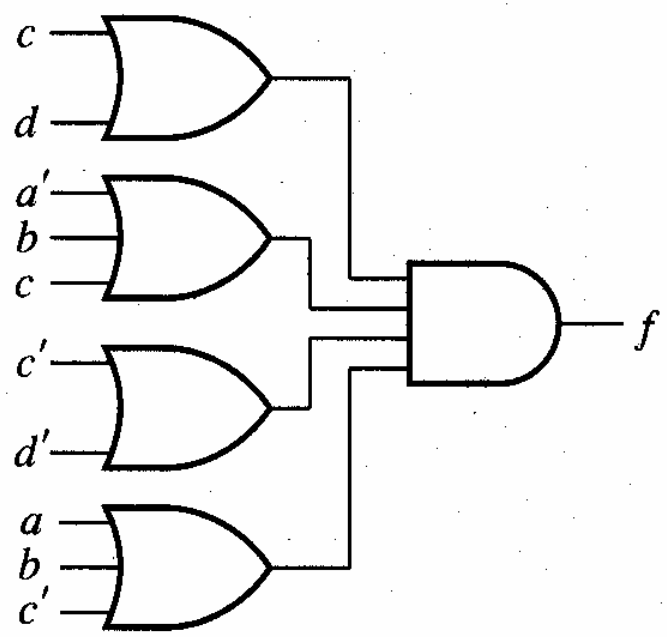
Example Continued...

Grouping 0's on the K-map yields:

$f' = c'd' + ab'c' + cd + a'b'c$ (8-3)

$f = (c+d)(a'+b+c)(c'+d')(a+b+c')$ (8-4)

2 levels
5 gates
14 gate inputs



*Two-level OR-AND

Example Continued…

◆ We can factor eqn. (8-3) to obtain a 3-level expression for f'

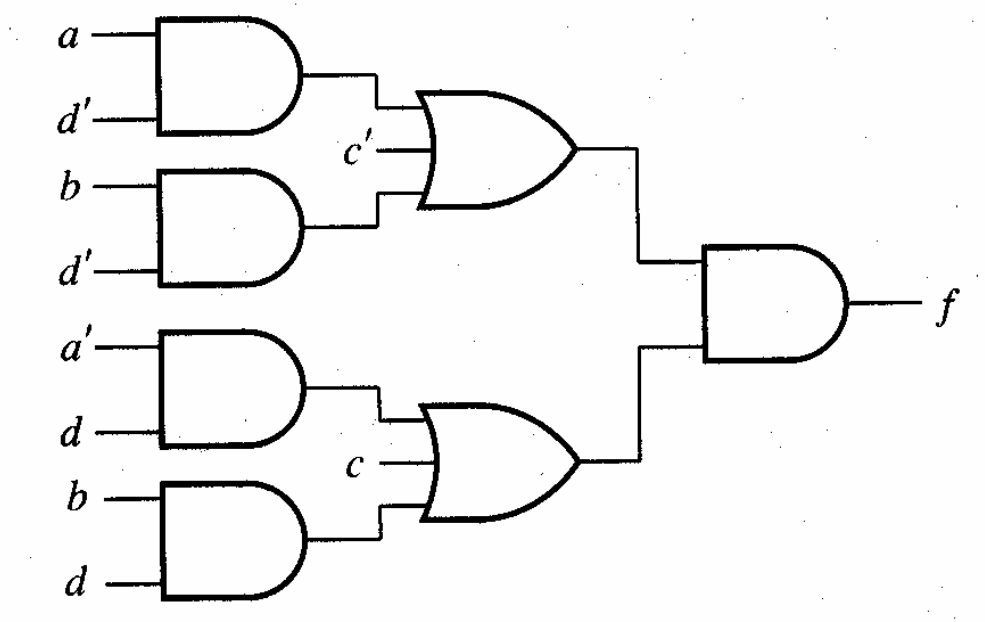
$$f' = c'd'+ab'c' + cd+a'b'c \qquad (8.3)$$
$$= c'(d'+ab') + c(d+a'b')$$
$$= c'(d'+a)(d'+b') + c(d+a')(d+b') \qquad (8.7)$$

◆ Taking the complement:

$$f = (c+a'd+bd)(c'+ad'+bd') \qquad (8.6)$$

Example Continued…

3 levels
7 gates
16 gate inputs

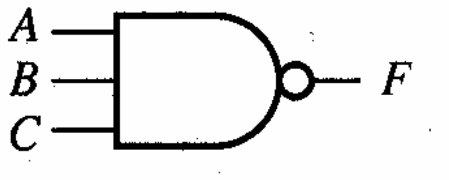


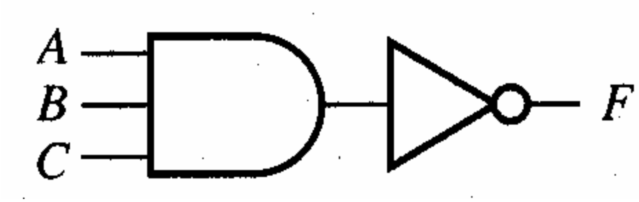
Three-level AND-OR-AND

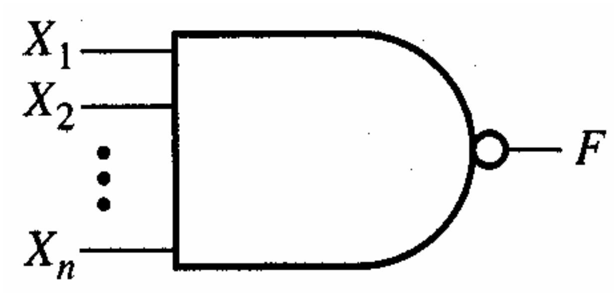
In general, if an expression for f′ has n levels, the complement of that expression is an n-level expression for f

Additional Logic Operations – NAND

NAND (NOT – AND) is the complement of the AND operation





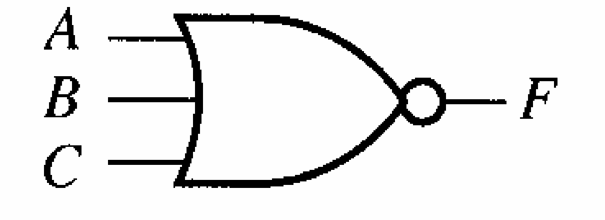


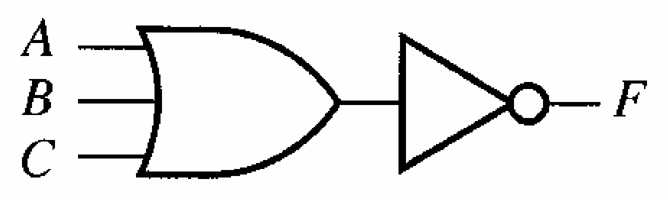
Output of this gate is 1 iff one or more inputs are 0.

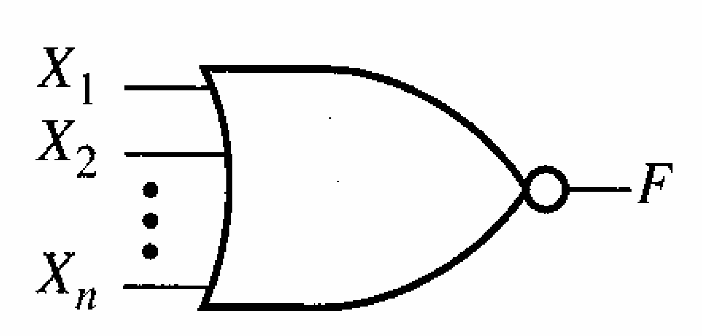
$$F = (X_1 X_2 \cdots X_n)' = X_1' + X_2' + \cdots + X_n'$$

Additional Logic Operations – NOR

NOR (NOT – OR) is the complement of the OR operation



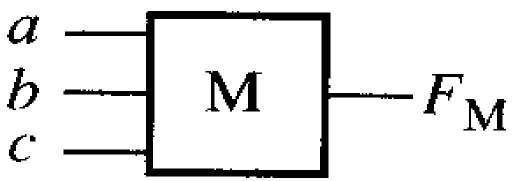




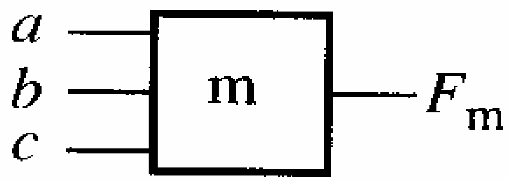
Output of this gate is 1 iff all inputs are 0.

$$F = (X_1 + X_2 + \cdots + X_n)' = X_1' X_2' \cdots X_n'$$

Additional Logic Operations



(a) Majority gate



(b) minority gate

Majority Gate:

- Has an odd number of inputs
- Output is 1 iff a majority of its inputs are 1

Minority Gate:

- Has an odd number of inputs
- Output is 1 iff a minority of its inputs are 1

Additional Logic Operations

a	b	c	F_M (Majority)	F_m (Minority)
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

(c) truth table

From the truth table the function realized by 3-input majority gate is:

$F_M = a'bc+ab'c+abc'+abc = bc+ac+ab$

By inspection of the table $F_m = F_M'$

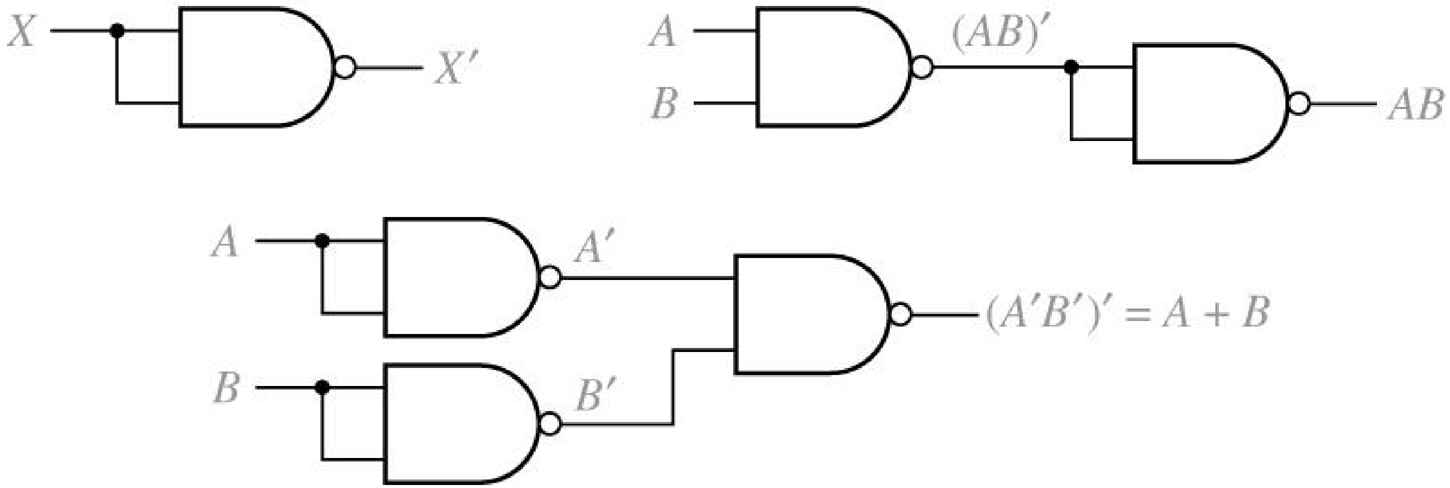
$F_m=(bc+ac+ab)' = (b'+c')(a'+c')(a'+b')$

Functionally Complete Sets of Logic Gates

- AND , OR, NOT are all that's needed to express any combinational logic function as a switching algebra expression
  - operators are all that were originally defined
  - Thus the set {AND, OR, NOT} is said to be functionally complete.
- Other functionally complete sets exist
  - {NAND} NAND by itself
  - {NOR} NOR by itself
- We can demonstrate how just NANDs or NORs (sometimes called "universal gates") can do AND, OR, NOT operations

NAND as a Functionally Complete Set

◆ NAND gate realization of NOT, AND, and OR



◆ AND realized by using OR and NOT

$$XY = (X' + Y')'$$

Conversion of a SOP to Two-Level Forms

◆DeMorgan's laws

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

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

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

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

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

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

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

Conversion of a SOP to Two-Level Forms

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

NOR-NOR-INVERT

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

OR-AND

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

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

NOR-NOR

$$= (A'B'C' + A'BC + A'CD')'$$

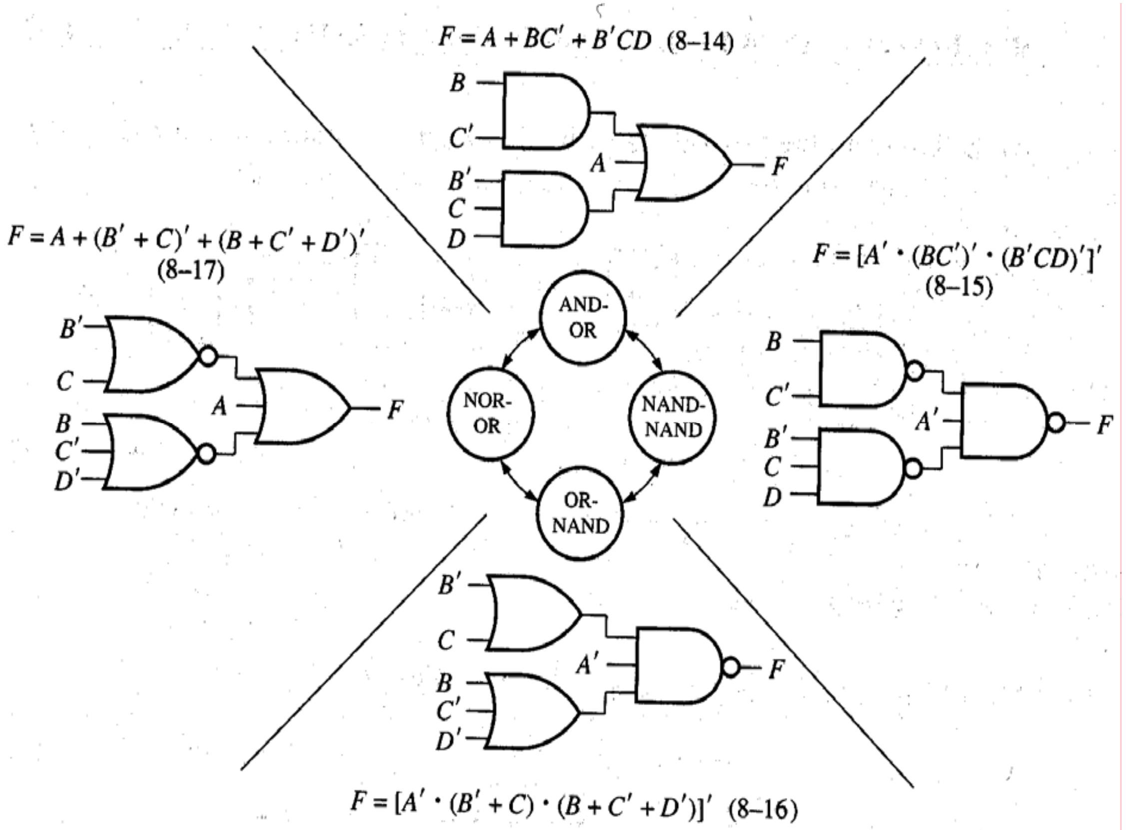
AND-NOR

$$= (A'B'C')' \cdot (A'BC)' \cdot (A'CD')'$$

NAND-AND

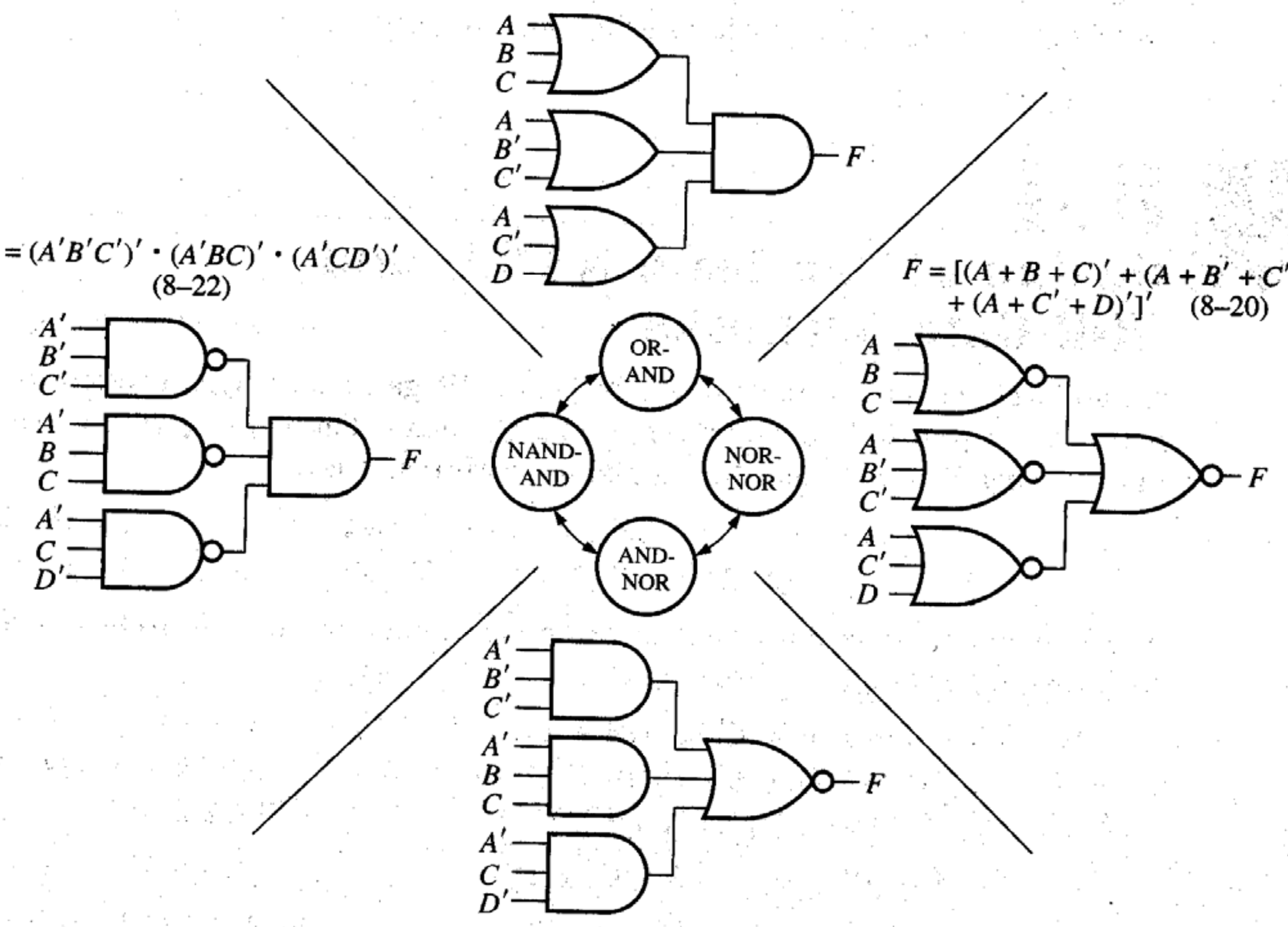
Eight Basic Forms for Two-Level Circuits

◈ A two-level circuit composed of AND and OR gates is easily converted to a circuit composed NAND gates or of NOR gates. This conversion carried out by using F=(F')' and then applying DeMorgan's law



Eight Basic Forms for Two-Level Circuits

$$F = (A + B + C)(A + B' + C')(A + C' + D) \quad (8\text{–}19)$$



$$F = (A'B'C')' \cdot (A'BC)' \cdot (A'CD')' \quad (8\text{–}22)$$

$$F = [(A + B + C)' + (A + B' + C')' + (A + C' + D)']' \quad (8\text{–}20)$$

$$F = (A'B'C' + A'BC + A'CD')' \quad (8\text{–}21)$$

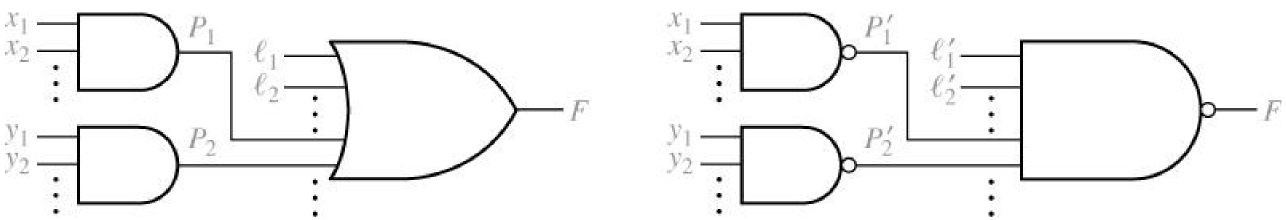
Two-Level NAND-NAND Circuits

- Procedure for designing a minimum two-level NAND-NAND network:

  1. Find a minimum SOP expression for F
  2. Draw the corresponding two-level AND-OR network.
  3. Replace all gates with NAND gates leaving
  4. The gate interconnections unchanged.
     If the output gate has any single literals as inputs, complement these literals.

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

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

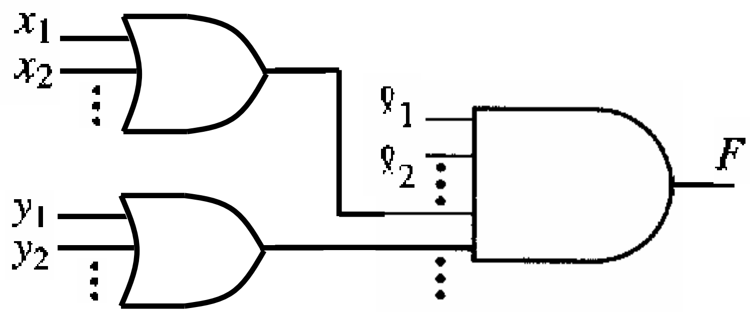


(a) Before transformation (b) After transformation

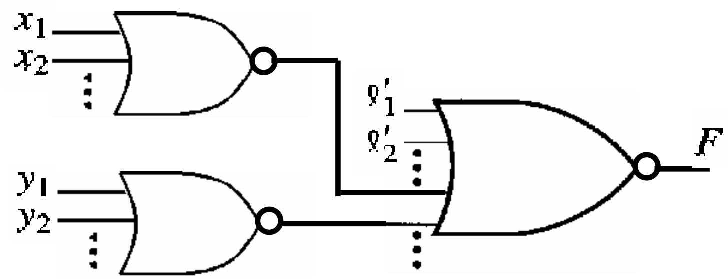
Two-Level NOR-NOR Circuits

◈ Procedure for designing a minimum two-level NOR-NOR network :

1. Find a minimum POS expression for F
2. Draw the corresponding two-level OR-AND network.
3. Replace all gates with NOR gates leaving
4. The gate interconnections unchanged.
   If the output gate has any single literals as inputs, complement these literals.



(a) Before transformation



(b) After transformation

Two Level Form Summary

- Any logic function in SOP form can be implemented in the two level gate forms of AND-OR, NAND-NAND.

- Any logic function in POS form can be implemented in the two level gate forms of OR-AND, NOR-NOR.

Multi-Level NAND Circuits

- Combinational circuits are more frequently constructed with NAND or NOR gates rather than AND and OR gates. NAND and NOR are more common from the hardware point of view, because they are readily available in IC form.

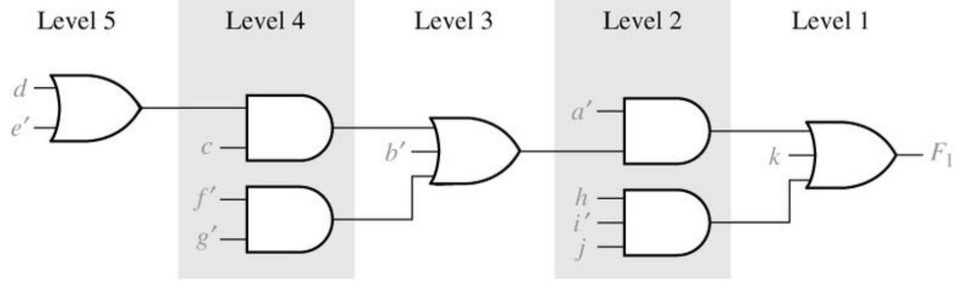
- The implementation of Boolean functions with NAND gates may be obtained by means of a simple block diagram manipulation technique.
  1. From the given algebraic expression, draw the logic diagram with AND, OR, and NOT gates. Assume that both the normal and complement inputs are available.
  2. Draw a second logic diagram with each gate replaced by its equivalent NAND logic.
  3. Remove any two cascaded inverters from the diagram. Remove inverters connected to single external inputs and complement the corresponding input variable.

A similar method can be used for NOR networks

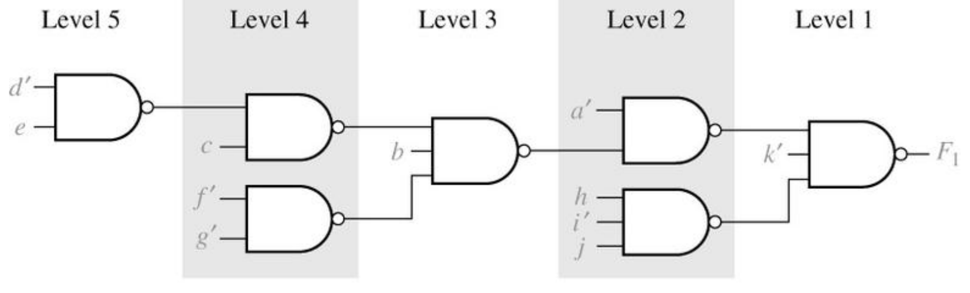
Example

◆ Multi-Level Circuit Conversion to NAND Gates

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



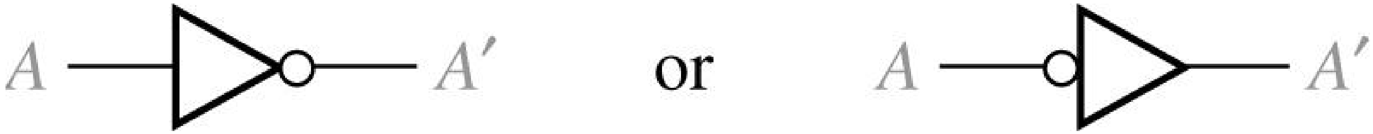
(a) AND-OR network



(b) NAND network

Circuit Conversion Using Alternative Gate Symbols

◈Inverter



◈Alternative Gate Symbols

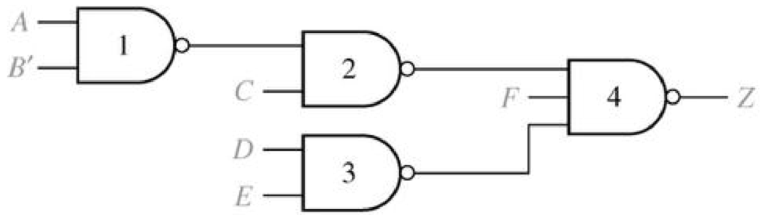
$AB = (A' + B')'$ (a) AND

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

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

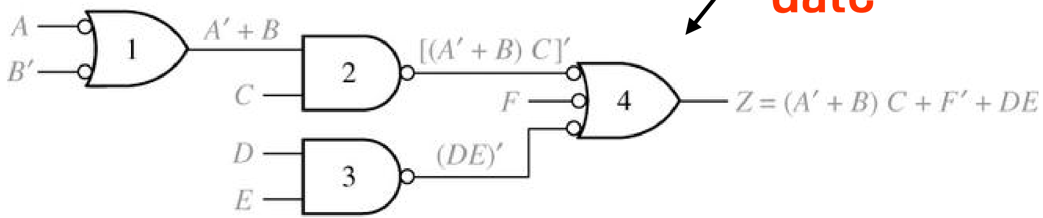
$(A + B)' = A'B'$ (d) NOR

◆NAND Gate Circuit Conversion

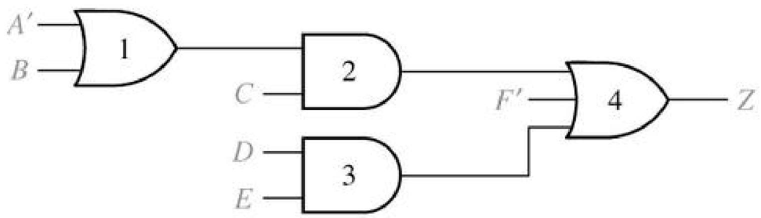


(a) NAND gate network

This is a NAND Gate

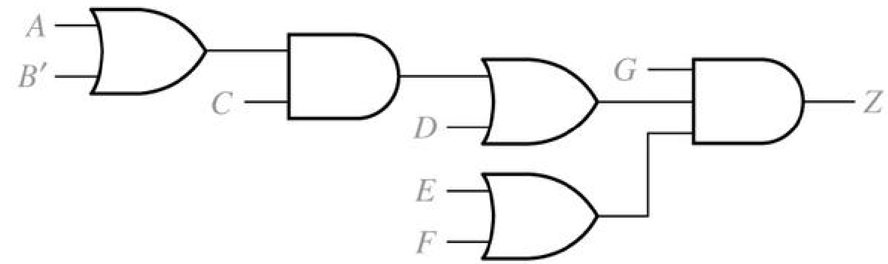


(b) Alternate form for NAND gate network

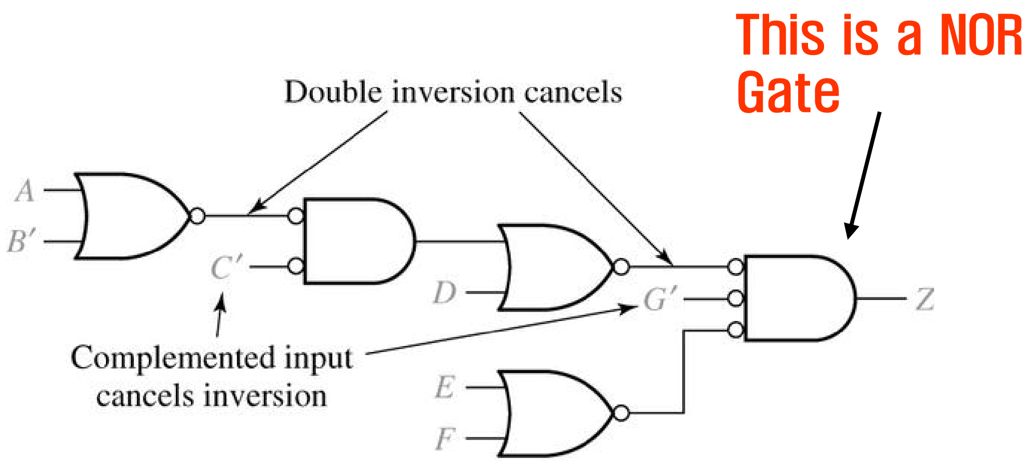


(c) Equivalent AND-OR network

◈Conversion to NOR Gates

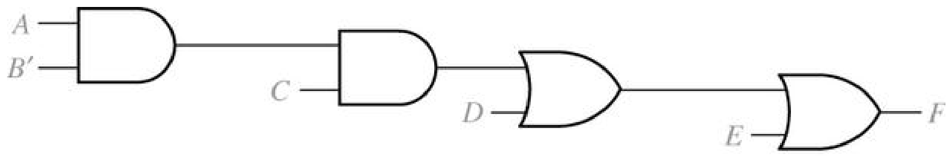


(a) Circuit with OR and AND gates

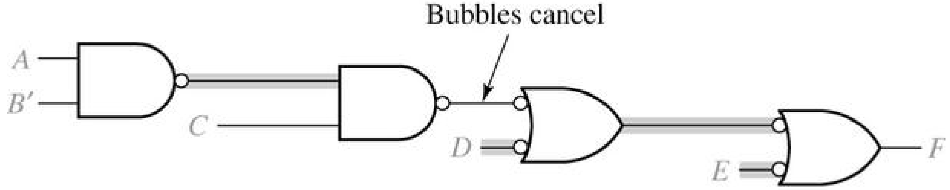


(b) Equivalent circuit with NOR gates

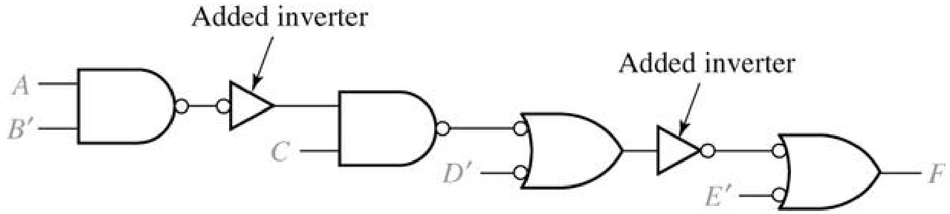
◆Conversion of AND-OR Circuits to NAND Gates



(a) AND-OR network



(b) First step in NAND conversion



(c) Completed conversion

Conversion of AND-OR Circuits to NAND Gates

1. Convert all AND gates to NAND gates by adding an inversion bubble at the output. Convert all OR gates to NAND gates by adding inversion bubbles at the inputs. (To convert NOR, add inversion bubbles at all OR gate outputs and all AND gate inputs)

2. Whenever an inverted output drives an inverted input, no further action is needed since the two inversion cancel

3. Whenever a noninverted gate output drives an inverted gate input or vice versa, insert an inverter so that the bubbles will cancel. (Choose an inverter with the bubble at the input or output as required.)

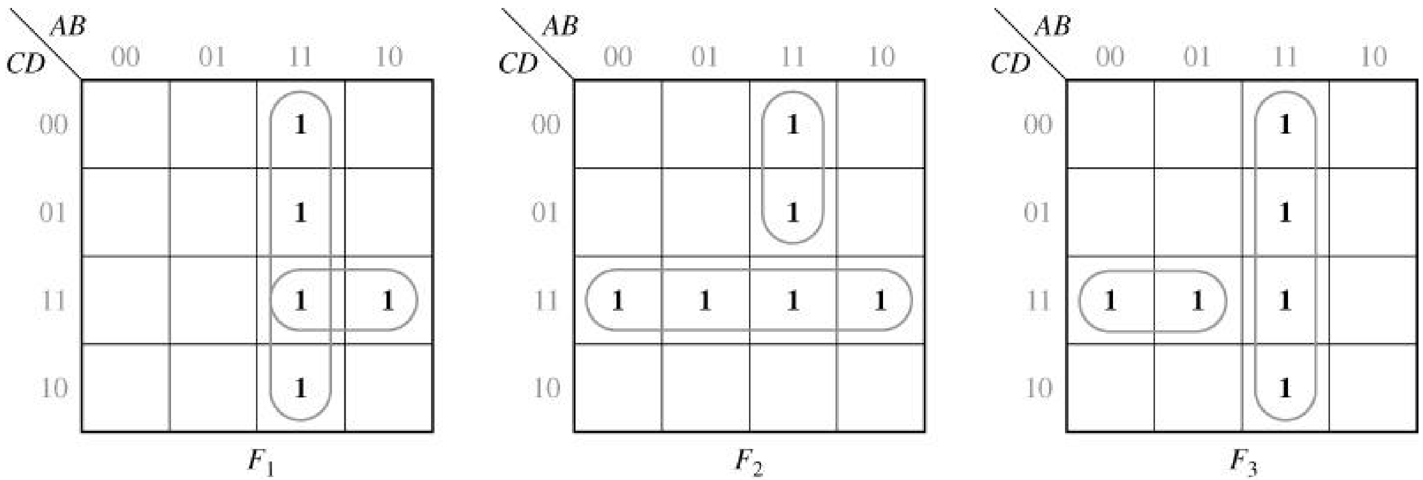
4. Whenever a variable drives an inverted input, complement the variable (or add an inverter) so the complementation cancels the inversion at the input.

Example 1

◆Design a circuit with four inputs and three outputs

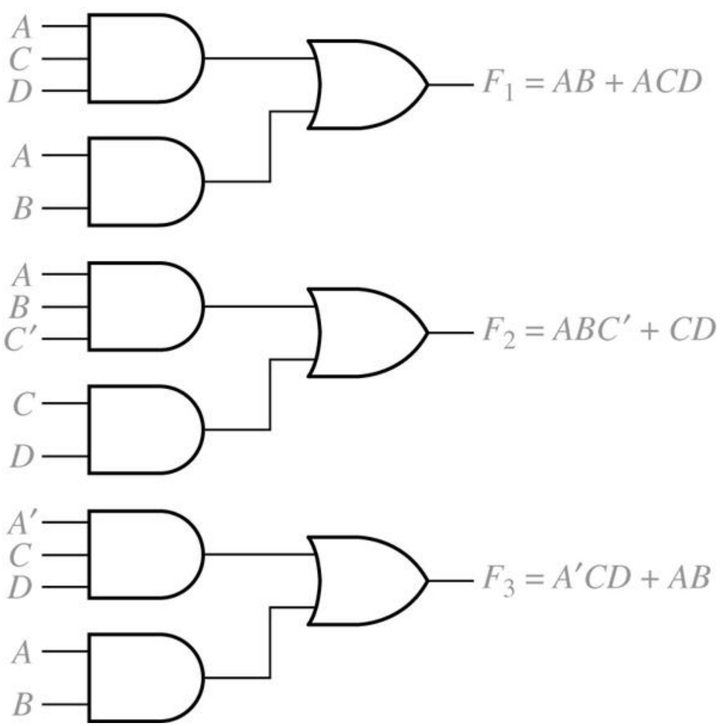
$$F_1(A,B,C,D)=\sum m(11,12,13,14,15)$$
$$F_2(A,B,C,D)=\sum m(3,7,11,12,13,15)$$
$$F_3(A,B,C,D)=\sum m(3,7,12,13,14,15)$$

Karnaugh Maps for Equations

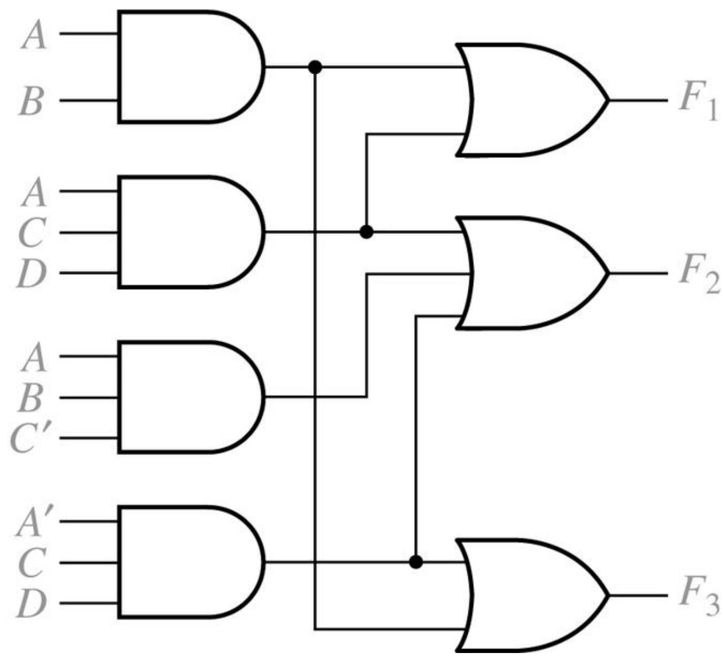


Example 1 (Cont.)

Realization of Equations



Multiple-Output Realization of Equations



Example 2

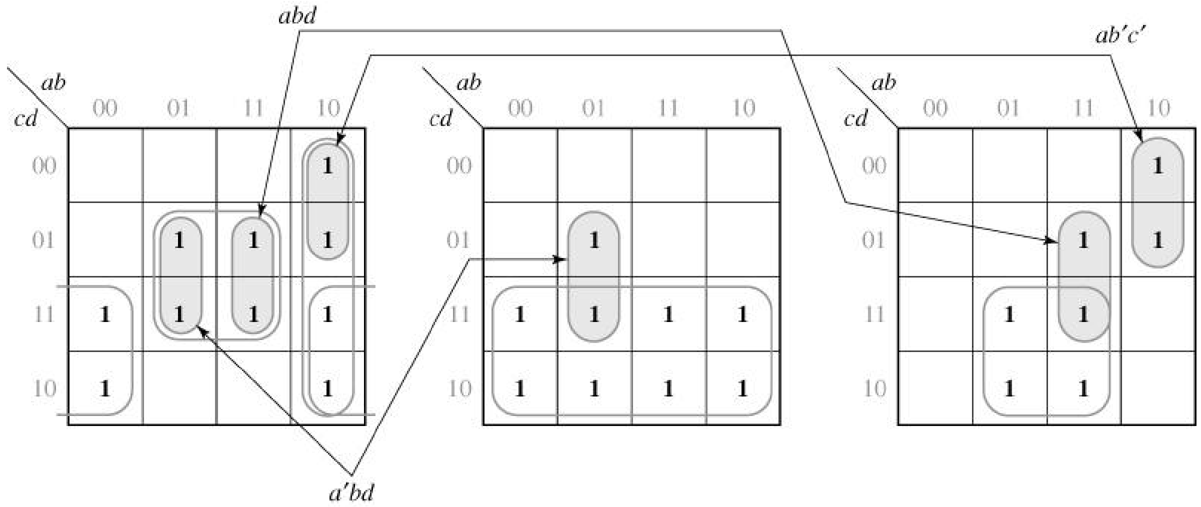
◈ Design a multiple-output circuit with 4-inputs and 3-outputs

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

$$f_2 = \sum m(2,3,5,6,7,10,11,14,15)$$

$$f_3 = \sum m(6,7,8,9,13,14,15)$$

Karnaugh Maps for Equations



Example 2 (Cont.)

Minimized equations

$$f_1 = bd + b'c + ab'$$
$$f_2 = c + a'bd$$
$$f_3 = bc + ab'c' + \begin{Bmatrix} abd \\ or \\ ac'd \end{Bmatrix}$$

10gates,

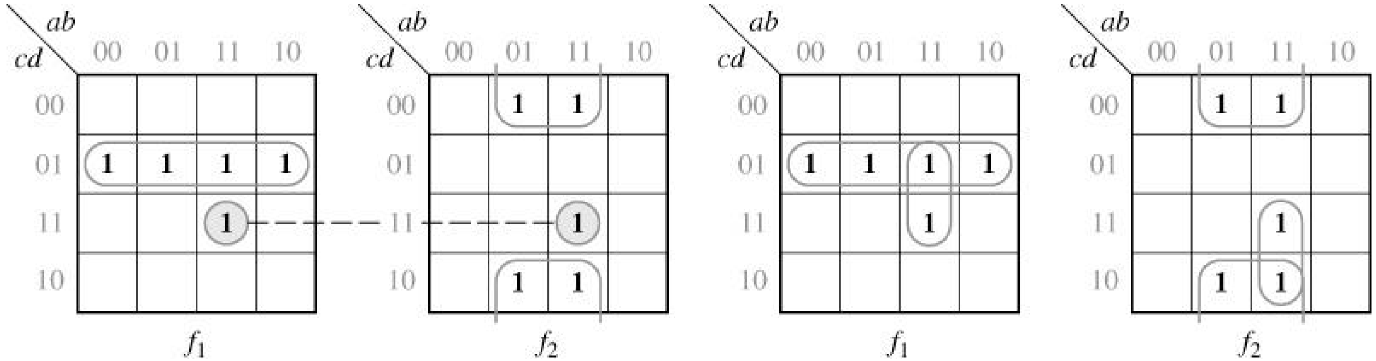
25gate input

The minimal solution

$$f_1 = \underline{a'bd} + \underline{abd} + \underline{ab'c'} + b'c$$
$$f_2 = c + \underline{a'bd}$$
$$f_3 = bc + \underline{ab'c'} + \underline{abd}$$

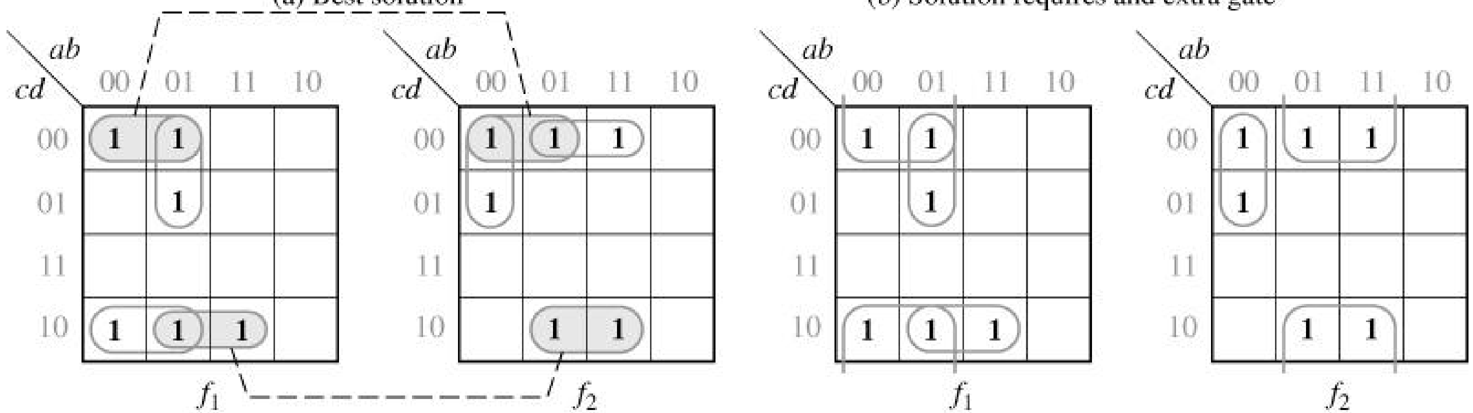
8 gates
22 gate inputs

◆ Determination of Essential Prime Implicants for Multiple-Output Realization



(a) Best solution

(b) Solution requires and extra gate

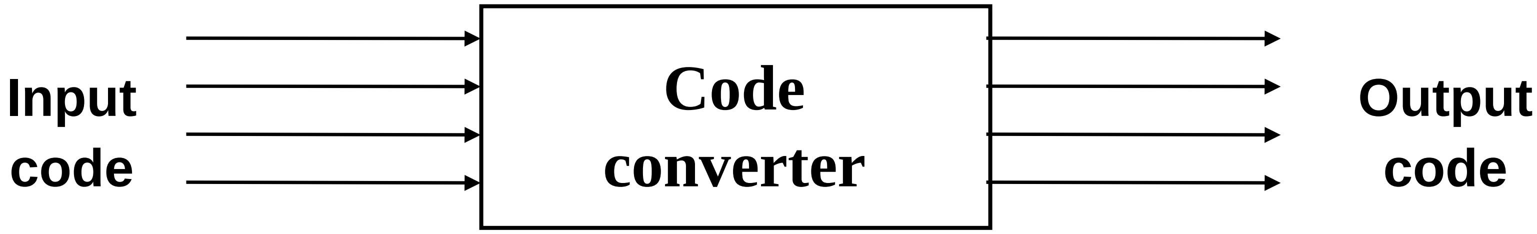


(a) Solution with maximum number of common terms requires 8 gates, 26 inputs

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

Example : Code Converter

◆ Code converters – take an input code, translate to its equivalent output code.



◆ Example: BCD to Excess-3 Code Converter.

Input: BCD digit

Output: Excess-3 digit

BCD-to-Excess-3 Code Converter

◈Truth table:

	BCD				Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

$$w = \Sigma\, m(5,6,7,8,9)$$

$$x = \Sigma\, m(1,2,3,4,9)$$

$$y = \Sigma\, m(0,3,4,7,8)$$

$$z = \Sigma\, m(0,2,4,6,8)$$

BCD-to-Excess-3 Code Converter

◈ K-Maps

$W = \Sigma m(5,6,7,8,9) + \Sigma d(10,11,12,13,14,15) = a + bc + bd = a + b\underline{(c+d)}$

CD \ AB	00	01	11	10
00	(0)	(4)	x (12)	1 (8)
01	(1)	1 (5)	x (13)	1 (9)
11	(3)	1 (7)	x (15)	x (11)
10	(2)	1 (6)	x (14)	x (10)

$x = \Sigma m(1,2,3,4,9) + \Sigma d(10,11,12,13,14,15)$

$bc'd' + b'd + b'c = bc'd' + b'\underline{(c+d)}$

CD \ AB	00	01	11	10
00	(0)	1 (4)	x (12)	0 (8)
01	1 (1)	(5)	x (13)	1 (9)
11	1 (3)	(7)	x (15)	x (11)
10	1 (2)	(6)	x (14)	x (10)

Underlined terms are common

BCD-to-Excess-3 Code Converter

◆K-Maps

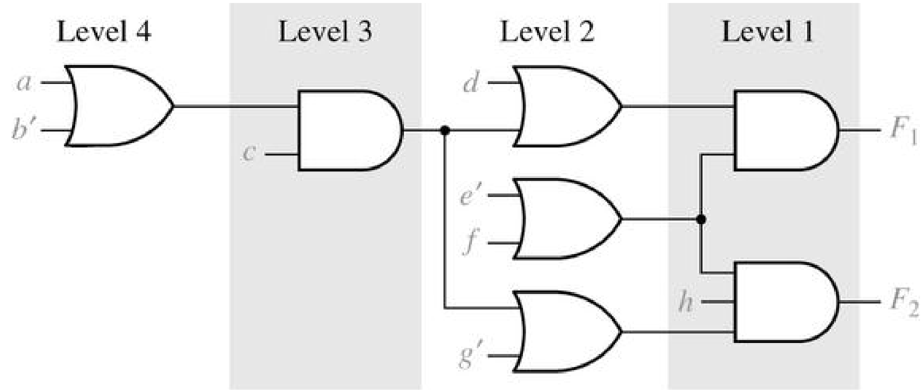
y = Σm(0,3,4,7,8)+
Σd(10,11,12,13,14,15)
= c'd'+cd

CD \ AB	00	01	11	10
00	1_{0}	1_{4}	x_{12}	1_{8}
01	1	5	x_{13}	9
11	1_{3}	1_{7}	x_{15}	x_{11}
10	2	6	x_{14}	x_{10}

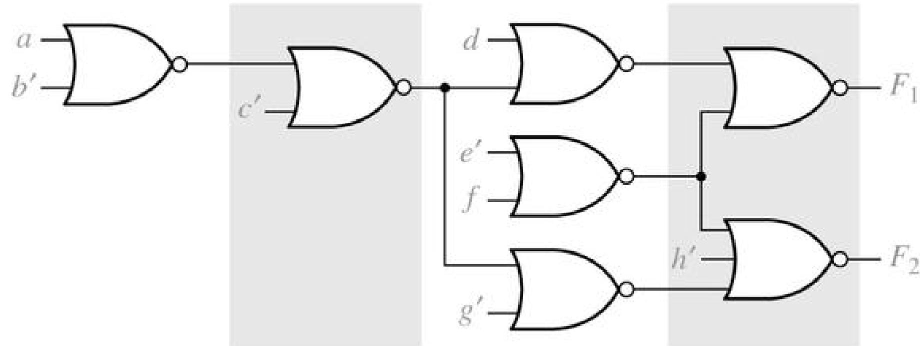
z = Σm(0,2,4,6,8)+
Σd(10,11,12,13,14,15)
= d'

CD \ AB	00	01	11	10
00	1_{0}	1_{4}	x_{12}	1_{8}
01	1	5	x_{13}	9
11	3	7	x_{15}	x_{11}
10	1_{2}	1_{6}	x_{14}	x_{10}

◈Multi-level Circuits Conversion to NOR Gates



(a) Network of AND and OR gates



(b) NOR network