

Multi-Level Gate Circuit NAND and NOR Gates

Contents

1. Multi-level Gate Circuits
2. NAND and NOR Gates
3. Design of Two-Level NAND- and NOR- Gate Circuits
4. Design of Multi-Level NAND- and NOR- Gate Circuits
5. Circuit Conversion Using Alternative Gate Symbols
6. Design of Two-Level, Multiple Output Circuits
7. Multiple Output NAND- and NOR- Gate Circuits

Objectives

- Design a minimal two-level or multi-level circuit of AND and OR gates to realize a given function. (Consider *both* circuits with an OR gate at the output and circuits with an AND gate at the output.)
- Design or analyze a two-level gate circuit using any one of the eight basic forms (AND-OR, NAND-NAND, OR-NAND, NOR-OR, OR-AND, NOR-NOR, AND-NOR, and NAND-AND).
- Design or analyze a multi-level NAND-gate or NOR-gate circuit.
- Convert circuits of AND and OR gates to circuits of NAND gates or NOR gates, and conversely, by adding or deleting inversion bubbles.
- Design a minimal two-level, multiple-output AND-OR, OR-AND, NAND-NAND, or NOR-NOR circuit using Karnaugh maps.

Multi-Level Gate Circuits



Terminology:

- 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 (not to be confused with voltage levels).
- **AND-OR circuit** means a two-level circuit composed of a level of AND gates followed by an OR gate at the output.
- **OR-AND circuit** means a two-level circuit composed of a level of OR gates followed by an AND gate at the output.
- **OR-AND-OR circuit** means a three-level circuit composed of a level of OR gates followed by a level of AND gates followed by an OR gate at the output.
- Circuit of AND and OR gates implies no particular ordering of the gates; the output gate may be either AND or OR.

Multi-Level Gate Circuits

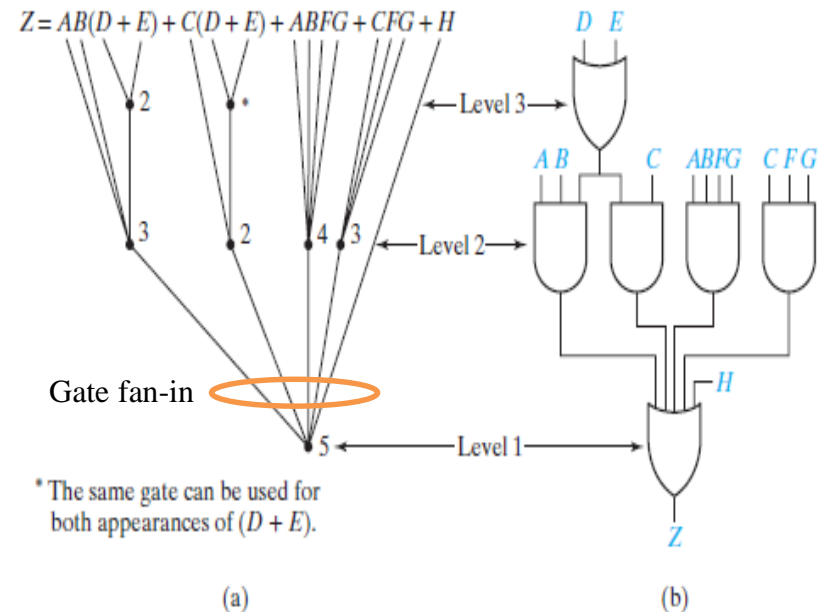
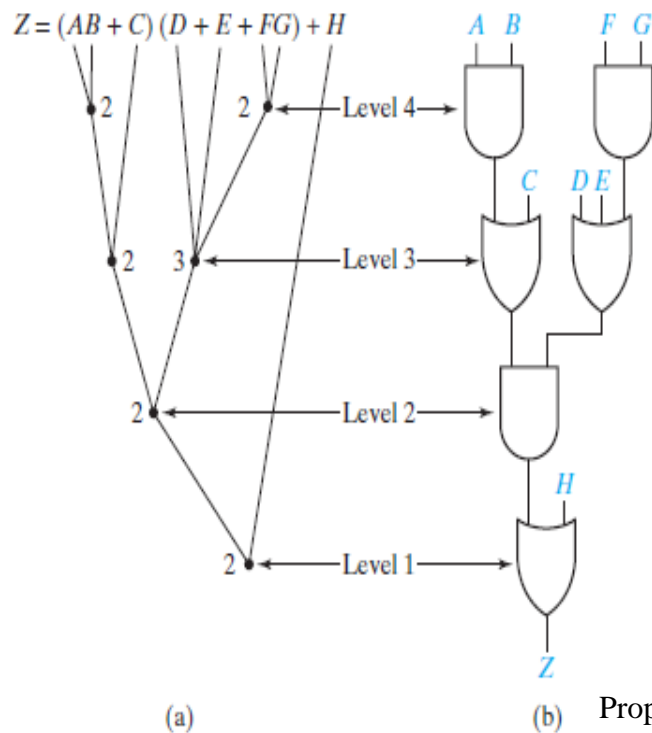
Different Realizations for the Same Expression:

- The number of gates, gate inputs, and levels in a circuit can be determined by inspection.
- Change the expression for Z to three levels by partially multiplying it out:

Factoring  $Z = (AB + C)(D + E + FG) + H$  Multiplying out

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

FIGURE 7-1
Four-Level
Realization of Z
© Cengage Learning 2014



Three levels, six gates, and 19 inputs

Four levels, six gates, and 13 inputs

Multi-Level Gate Circuits

Multi-Level Gate Circuits:

- The number of levels in an AND-OR circuit can be usually be increased by factoring the sum-of-products expression from which it was derived.
- The number of levels in an OR-AND circuit can be usually be increased by multiplying out some of the terms in the product-of-sums expression from which it was derived.
- Sometimes factoring or multiplying out to increase the number of levels of gates will reduce the required number of gates and gate inputs.
- In many applications, the number of gate which can be cascaded is limited by gate delays.
- Thus, the number of gates, gate inputs, and levels in a circuit can be determined by inspection.

Multi-Level Gate Circuits

Example 1: Try to minimize the number of gates and the total number of gate inputs

Example of Multi-Level Design Using AND and OR Gates

Problem: Find a circuit of AND and OR gates to realize

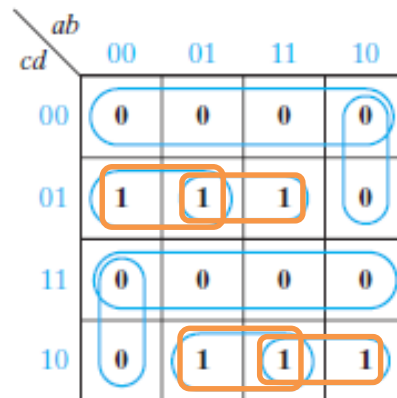
$$f(a, b, c, d) = \Sigma m(1, 5, 6, 10, 13, 14)$$

Consider solutions with two levels of gates and three levels of gates. Try to minimize the number of gates and the total number of gate inputs. Assume that all variables and their complements are available as inputs.

Solution: First, simplify f by using a Karnaugh map (Figure 7-3):

FIGURE 7-3

© Cengage Learning
2014



$$f = a'c'd + bc'd + bcd' + acd' \quad (7-1)$$

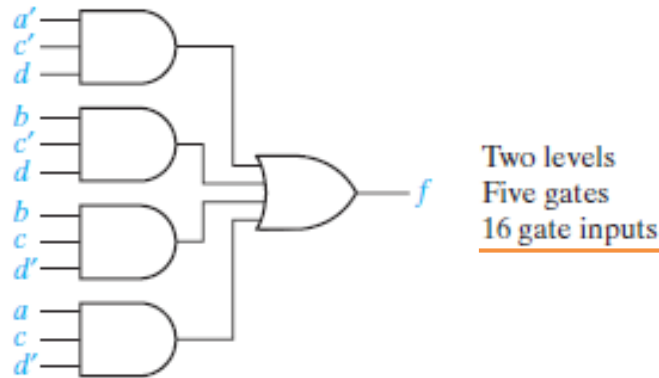
Multi-Level Gate Circuits

Example 1: Two-level AND-OR circuit (Sum-of-products expression)

This leads directly to a two-level AND-OR gate circuit (Figure 7-4):

FIGURE 7-4

© Cengage Learning
2014



Factoring Equation (7-1) yields

$$f = c'd(a' + b) + cd'(a + b) \quad (7-2)$$

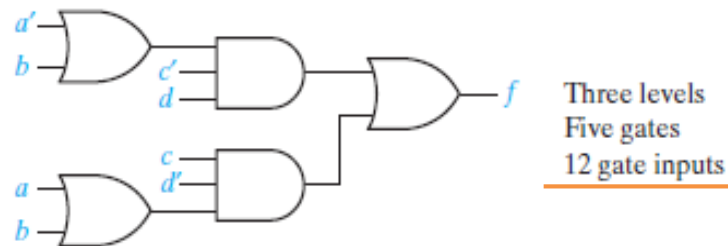
Multi-Level Gate Circuits

Example 1: Three-level OR-AND-OR circuit (By factoring)

which leads to the following three-level OR-AND-OR gate circuit (Figure 7-5):

FIGURE 7-5

© Cengage Learning
2014



By factoring

- Gate inputs: 16 \rightarrow 12 (cost)
- Maximum gate fan-in: 4 inputs \rightarrow 3 inputs (cost)
- 2-level \rightarrow 3-level (propagation delay)

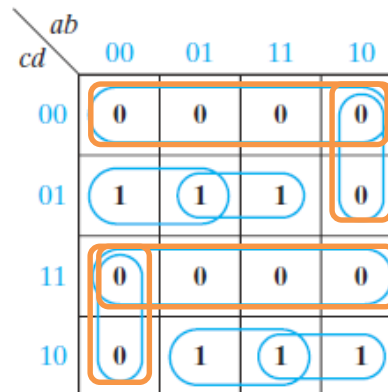
Both of these solutions have an OR gate at the output. A solution with an AND gate at the output might have fewer gates or gate inputs. A two-level OR-AND circuit corresponds to a product-of-sums expression for the function. This can be obtained from the 0's on the Karnaugh map as follows:

$$f' = c'd' + ab'c' + cd + a'b'c \quad (7-3)$$

$$f = (c + d)(a' + b + c)(c' + d')(a + b + c') \quad (7-4)$$

FIGURE 7-3

© Cengage Learning
2014

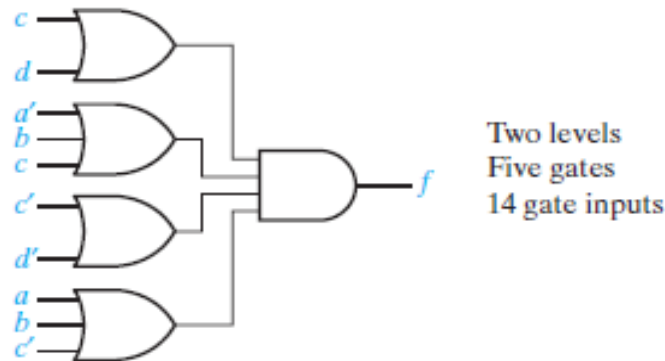


Multi-Level Gate Circuits

Example 1: Two-level OR-AND circuit (Product-of-sums expression)

Equation (7-4) leads directly to a two-level OR-AND circuit (Figure 7-6):

FIGURE 7-6
© Cengage Learning
2014



To get a three-level circuit with an AND-gate output, we partially multiply out Equation (7-4) using $(X + Y)(X + Z) = X + YZ$:

$$f = [c + d(a' + b)][c' + d'(a + b)] \quad (7-5)$$

Equation (7-5) would require four levels of gates to realize; however, if we multiply out $d'(a + b)$ and $d(a' + b)$, we get

$$f = (c + a'd + bd)(c' + ad' + bd') \quad (7-6)$$

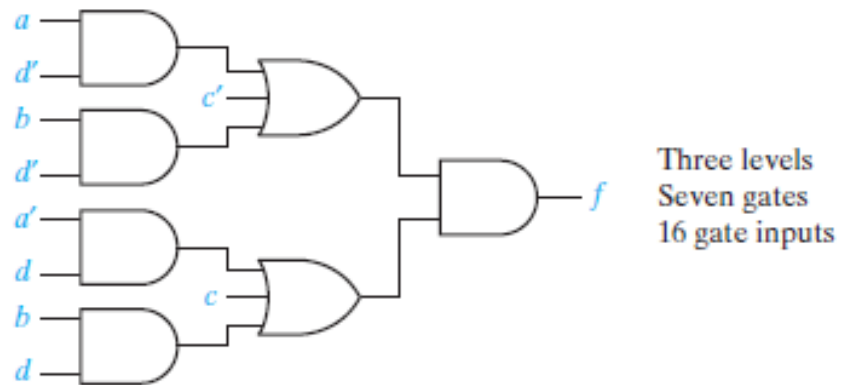
which leads directly to a three-level AND-OR-AND circuit (Figure 7-7):

Multi-Level Gate Circuits

Example 1: Three-level AND-OR-AND circuit (By multiplying out)

FIGURE 7-7

© Cengage Learning
2014



For this particular example, the best two-level solution had an AND gate at the output (Figure 7-6), and the best three-level solution had an OR gate at the output (Figure 7-5). In general, to be sure of obtaining a minimum solution, one must find both the circuit with the AND-gate output and the one with the OR-gate output.

Multi-Level Gate Circuits

Example 1:

If an expression for f' has n levels, the complement of that expression is an n -level expression for f . Therefore, to realize f as an n -level circuit with an AND-gate output, one procedure is first to find an n -level expression for f' with an OR operation at the output level and then complement the expression for f' . In the preceding example, factoring Equation (7-3) gives a three-level expression for f' :

$$\begin{aligned} f' &= c'(d' + ab') + c(d + a'b') \\ &= c'(d' + a)(d' + b') + c(d + a')(d + b') \end{aligned} \quad (7-7)$$

Complementing Equation (7-7) gives Equation (7-6), which corresponds to the three-level AND-OR-AND circuit of Figure 7-7.

NAND and NOR Gates

NAND and NOR Gates:

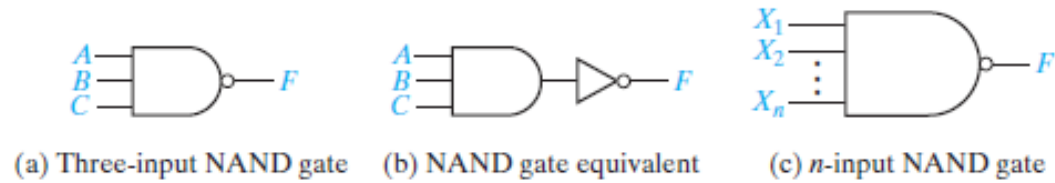
- Logic designer frequently use NAND and NOR gates because they are generally faster and use fewer components than AND or OR gates.
- Any logic function can be implemented using only NAND gates or NOR gates.
- The set AND, OR, and NOT is obviously *functionally complete* because any function can be expressed in sum-of-products form, a sum-of-products expression uses only the AND, OR, and NOT operations.
- Similarly, a set of logic gates is *functionally complete* if all switching functions can be realized using this set of gates.
- The NAND gate forms a *functionally complete set by itself*, and any switching function can be realized using only NAND gates.
- Similarly, the NOR gate forms a *functionally complete set by itself*, and any switching function can be realized using only NOR gates.

NAND and NOR Gates

NAND Gates:

FIGURE 7-8
NAND Gates

© Cengage Learning 2014



- A **NAND Gate** is equivalent to an AND gate followed by an inverter, or AND-NOT gate (as shown in Figure 7-8(b)). The n -input NAND gate's output is:

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

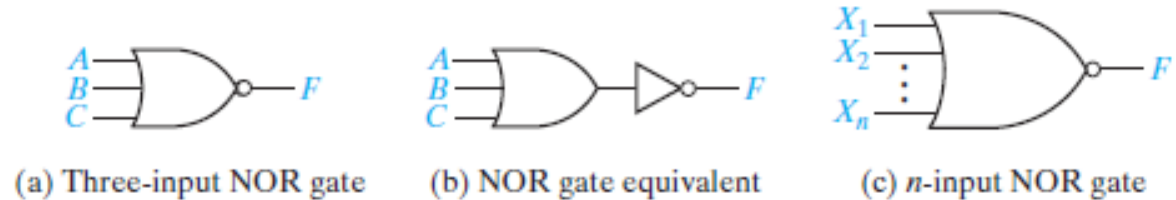
The output of this gate is 1 iff one or more of its inputs are 0.

NAND and NOR Gates

NOR Gates:

FIGURE 7-9
NOR Gates

© Cengage Learning 2014



A **NOR Gate** is equivalent to an OR gate followed by an inverter, or an OR-NOT gate. The n -inputs NOR gate's output is:

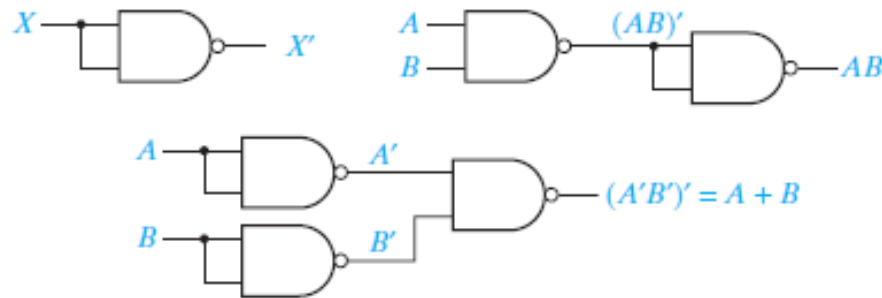
$$F = (X_1 + X_2 + \cdots + X_n)' = X_1' X_2' \cdots X_n'$$

NAND and NOR Gates

Functionally Complete Operations and 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.
- If a single gate forms a functionally complete set by itself, then any switching function can be realized using only gates of that type, i.e. NAND, NOR.

FIGURE 7-10
NAND Gate
Realization of
NOT, AND, and OR
© Cengage Learning 2014



NAND and NOR Gates

Procedure to Determine if Functionally Complete:

- Write out a minimum sum-of-products expression for the function realized by each gate.
 - ✓ If no complement appears in any of these expressions, then NOT cannot be realized, and the set is not functionally complete.
 - ✓ If a complement appears in one of the expressions, then NOT can generally be realized by an appropriate choice of inputs to the corresponding gate. (We will always assume that 0 and 1 are available as gate inputs).
- Next, attempt to realize AND or OR, keeping in mind that NOT is now available. Once AND or OR has been realized, the other one can always be realized using DeMorgan's laws if no more direct procedure is apparent. For example, if OR and NOT are available, AND can be realized by

$$XY = (X' + Y')' \quad (7-10)$$

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

Conversion of AND and OR Gate Circuits to NAND and NOR:

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

$$(X_1 + X_2 + \cdots + X_n)' = X_1'X_2' \cdots X_n' \quad (7-11)$$

$$(X_1X_2 \cdots X_n)' = X_1' + X_2' + \cdots + X_n' \quad (7-12)$$

Example 2: The following example illustrates conversion of a minimum sum-of-products form to several other two-level forms:

$$F = A + BC' + B'CD = [(A + BC' + B'CD)']' \quad (7-13)$$

$$= [A' \cdot (BC')' \cdot (B'CD)']' \quad (\text{by 7-11}) \quad (7-14)$$

$$= [A' \cdot (B' + C) \cdot (B + C' + D')]' \quad (\text{by 7-12}) \quad (7-15)$$

$$= A + (B' + C)' + (B + C' + D')' \quad (\text{by 7-12}) \quad (7-16)$$

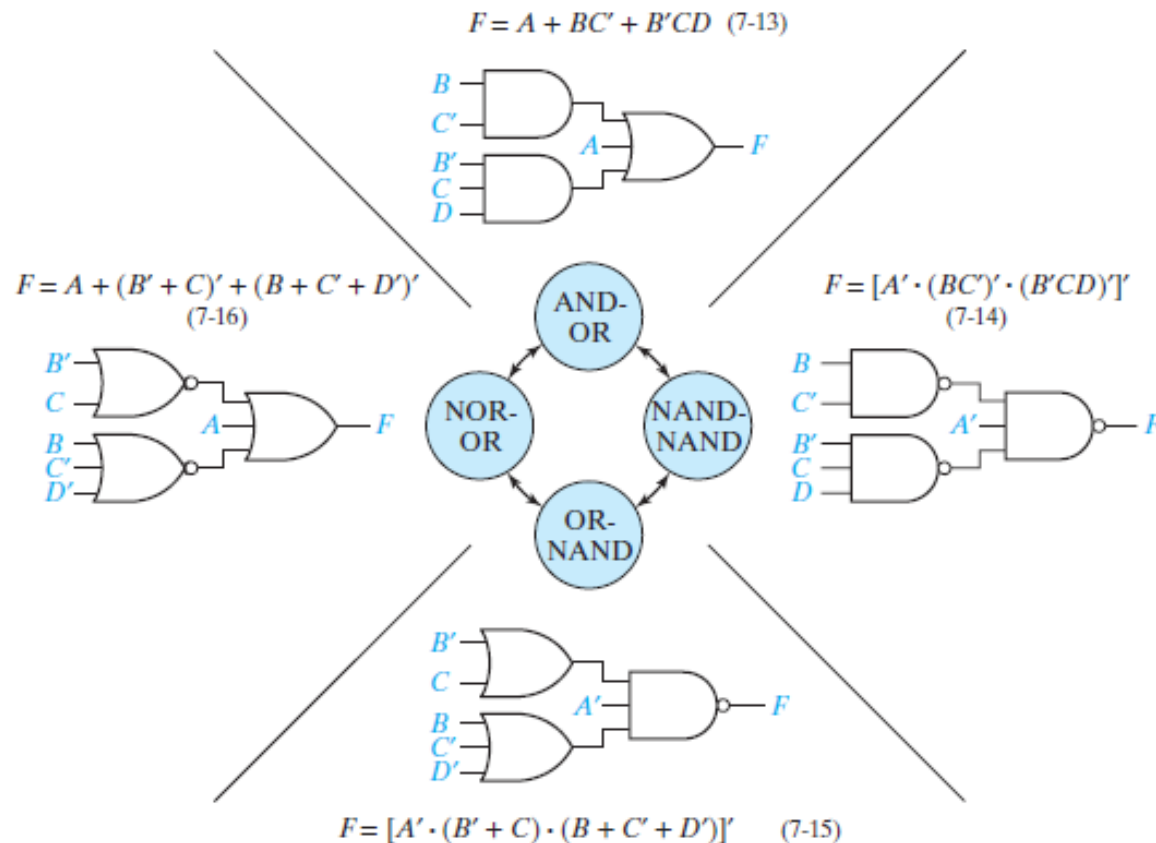
Equations (7-13), (7-14), (7-15), and (7-16) represent the AND-OR, NAND-NAND, OR-NAND, and NOR-OR forms, respectively, as shown in Figure 7-11.

Rewriting Equation (7-16) in the form

$$F = \{[A + (B' + C)' + (B + C' + D')']'\}' \quad (7-17)$$

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

Two-Level Circuits (Start with sum-of-products expression):



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

Example 2: leads to a three-level NOR-NOR-INVERT circuit. However, if we want a two-level circuit containing only NOR gates, we should start with the minimum product-of-sums form for F instead of the minimum sum of products. After obtaining the minimum product of sums from a Karnaugh map, F can be written in the following two-level forms:

$$F = (A + B + C)(A + B' + C')(A + C' + D) \quad (7-18)$$

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

$$= [(A + B + C)' + (A + B' + C')' + (A + C' + D)']' \quad (\text{by 7-12}) \quad (7-19)$$

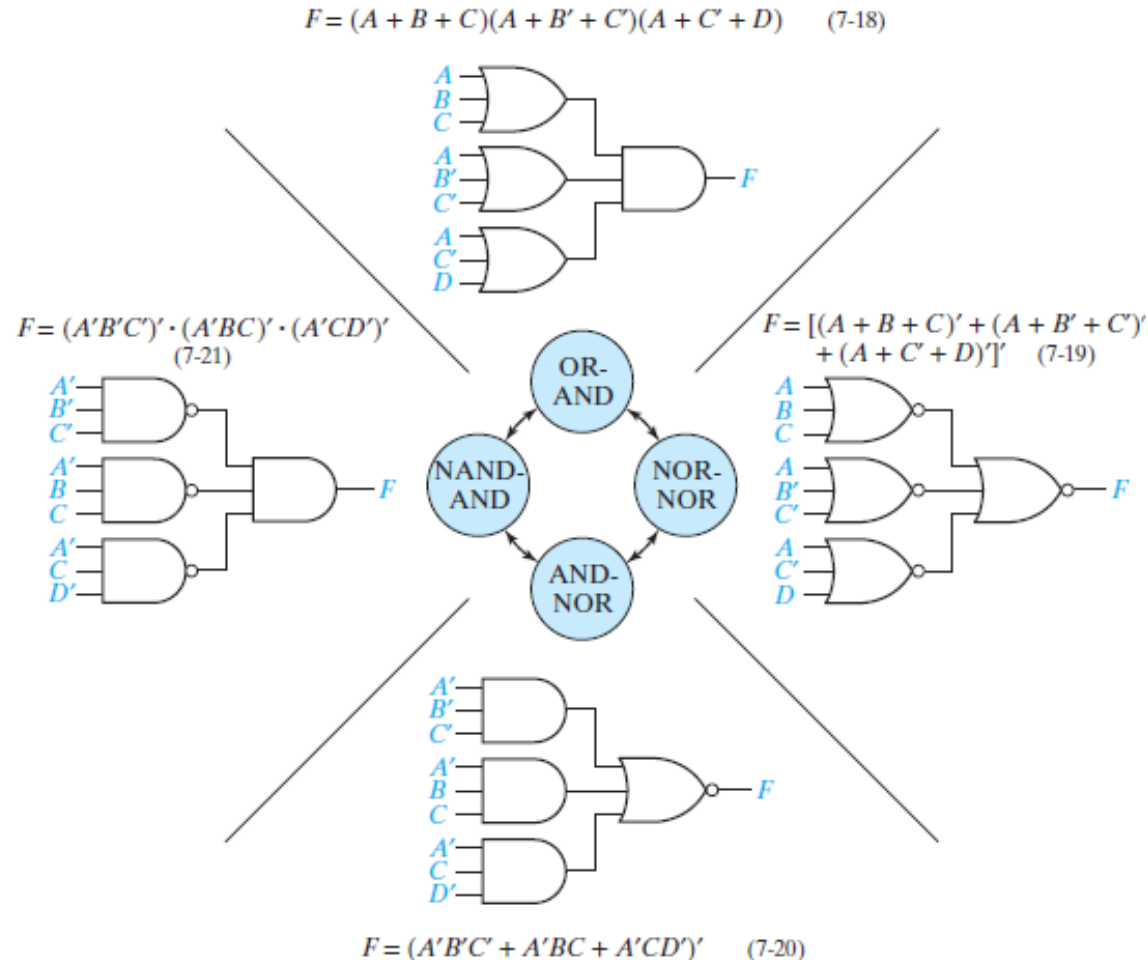
$$= (A'B'C' + A'BC + A'CD')' \quad (\text{by 7-11}) \quad (7-20)$$

$$= (A'B'C')' \cdot (A'BC)' \cdot (A'CD')' \quad (\text{by 7-11}) \quad (7-21)$$

Equations (7-18), (7-19), (7-20), and (7-21) represent the OR-AND, NOR-NOR, AND-NOR, and NAND-AND forms, respectively, as shown in Figure 7-11. Two-level AND-NOR (AND-OR-INVERT) circuits are available in integrated-circuit form. Some types of NAND gates can also realize AND-NOR circuits when the so-called *wired OR* connection is used.

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

Two-Level Circuits (Start with product-of-sums expression):



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

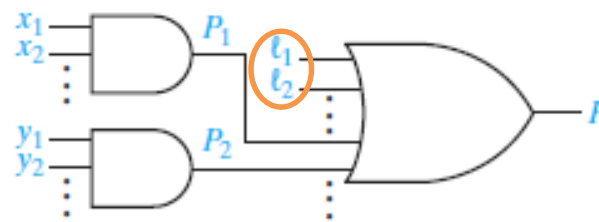
Procedure for Designing a Minimum Two-Level NAND-NAND circuit:

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

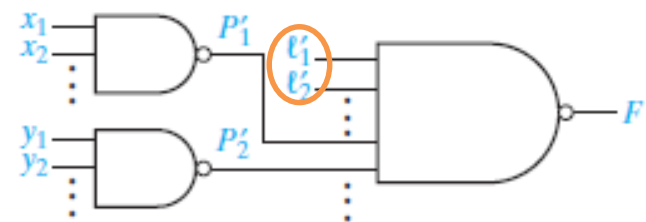
FIGURE 7-12

AND-OR to
NAND-NAND
Transformation

© Cengage Learning 2014



(a) Before transformation



(b) After transformation

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

Procedure for Designing a Minimum Two-Level NOR-NOR circuit:

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

This procedure is similar to that used for designing NAND-NAND circuits. Note, however, that for the NOR-NOR circuit, the starting point is a minimum *product of sums* rather than a sum of products.

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

Procedure to Design Multi-Level NAND-Gate Circuits:

1. Simplify the switching function to be realized.
2. Design a multi-level circuit of AND and OR gates. The output gate must be OR. AND-gate outputs cannot be used as AND-gate inputs; OR-gate outputs cannot be used as OR-gate inputs.
3. 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. Invert any literals which appear as inputs to levels 1, 3, 5, . . .

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

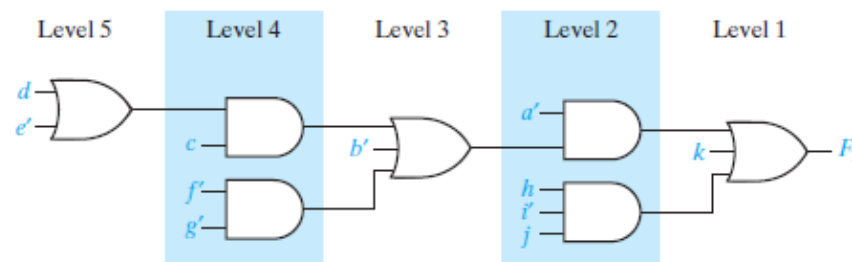
Example 3: Multi-Level Circuit Conversion to NAND Gates

Example

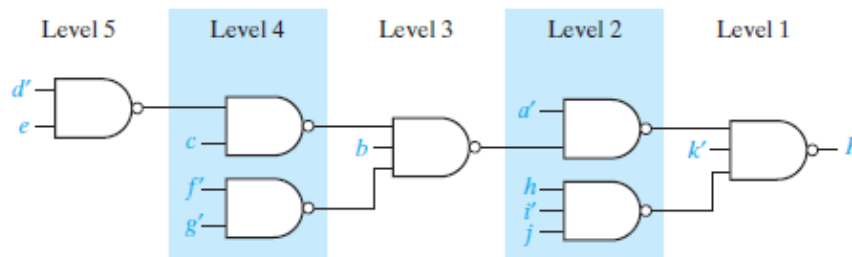
$$F_1 = a'[b' + c(d + e') + f'g'] + hi'j + k$$

Figure 7-13 shows how the AND-OR circuit for F_1 is converted to the corresponding NAND circuit.

FIGURE 7-13
Multi-Level Circuit
Conversion to
NAND Gates
© Cengage Learning
2014



(a) AND-OR network

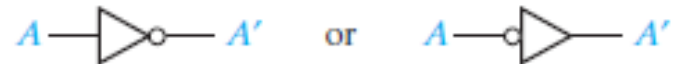


(b) NAND network

Circuit Conversion Using Alternative Gate Symbols

Alternative Gate Symbols:

- Alternate symbol for inverters:



- The following gate symbols have been derived using DeMorgan's Laws.

FIGURE 7-14
Alternative Gate Symbols

© Cengage Learning
2014



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

(a) AND



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

(b) OR



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

(c) NAND



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

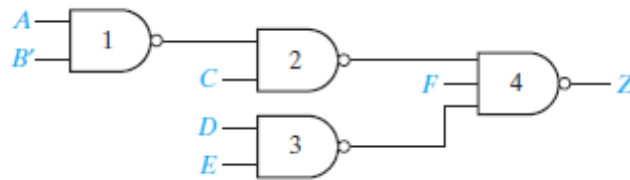
(d) NOR

Circuit Conversion Using Alternative Gate Symbols

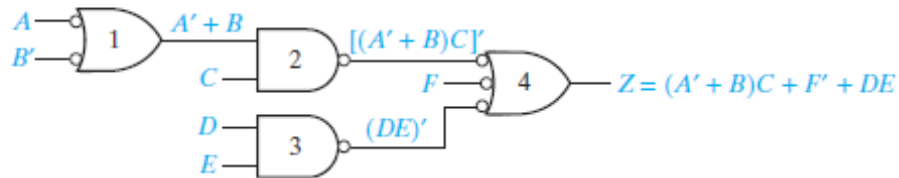
NAND Gate Circuit Conversion:

- In Figure 7-15(b), inverted outputs are always connected to inverted inputs, and noninverted outputs are connected to noninverted inputs.
- Convert the circuit to an AND-OR circuit by removing the double inversion.
- Complement the variable when we remove the inversion from the gate input.

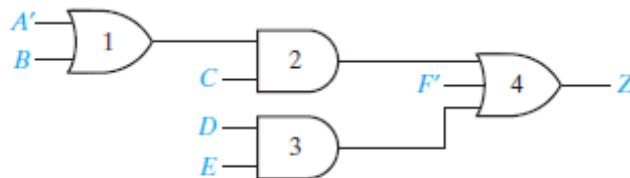
FIGURE 7-15
NAND Gate Circuit
Conversion
© Cengage Learning 2014



(a) NAND gate network



(b) Alternate form for NAND gate network



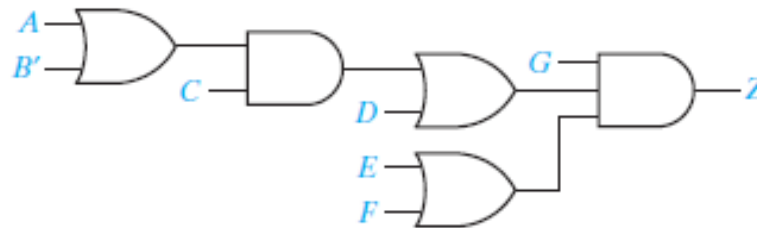
(c) Equivalent AND-OR network

Circuit Conversion Using Alternative Gate Symbols

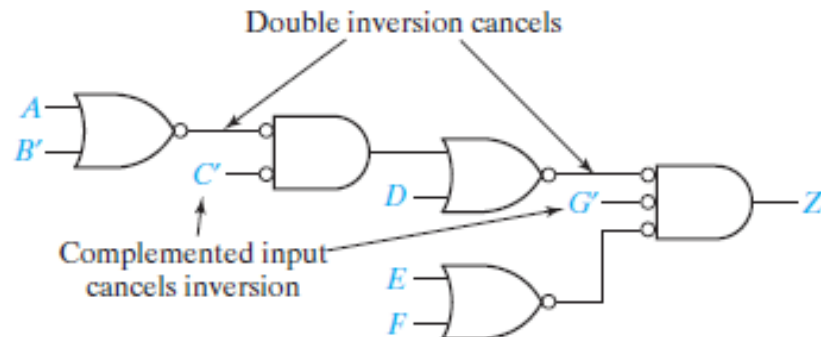
NOR Gate Circuit Conversion:

- Output gate is an AND-gate, and AND and OR gates alternate.
- Replace all of the OR and AND gates with NOR gates, as shown in Figure 7-16(b).
- When an input variable drives an inverted input, we must complement the variable to compensate. Therefore, we have complemented C and G .

FIGURE 7-16
Conversion to
NOR Gates
© Cengage Learning 2014



(a) Circuit with OR and AND gates



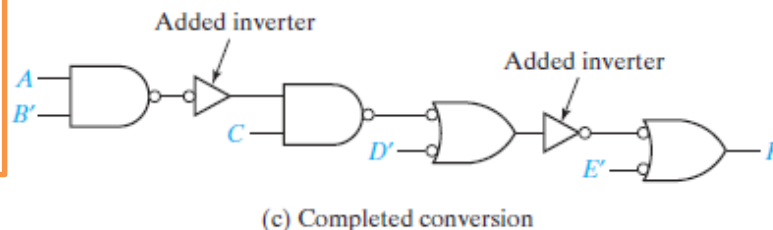
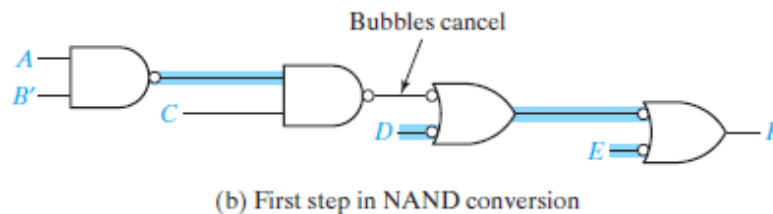
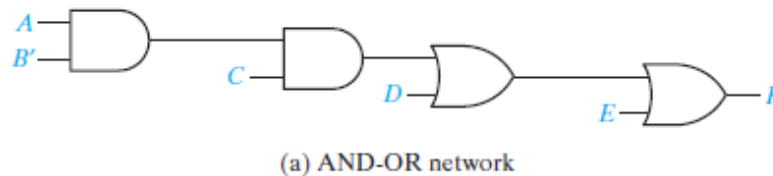
(b) Equivalent circuit with NOR gates

Circuit Conversion Using Alternative Gate Symbols

Conversion of AND-OR Circuit to NAND Gates:

- AND and OR gates do not alternate in this case.
- 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 bubble at the inputs.
- Whenever a noninverted gate output drives an inverted gate input or vice versa, insert an inverter.
- Whenever a variable drives an inverted input, complement the variable.

FIGURE 7-17
Conversion of
AND-OR Circuit
to NAND Gates
© Cengage Learning 2014



- Note that when an inverter is added between two gates during the conversion procedure, the number of levels in the circuit is increased by 1.
- This is avoided if each path through the circuit alternately passes through AND and OR gates.

Circuit Conversion Using Alternative Gate Symbols

Procedure to Convert to a NAND (or NOR) Circuit:

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 to 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 because the two inversions 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.

Circuit Conversion Using Alternative Gate Symbols

Example 4: Multi-level circuits reduce gate fan-in

$$F = D'E + BCE + AB' + AC' \quad (7-22)$$

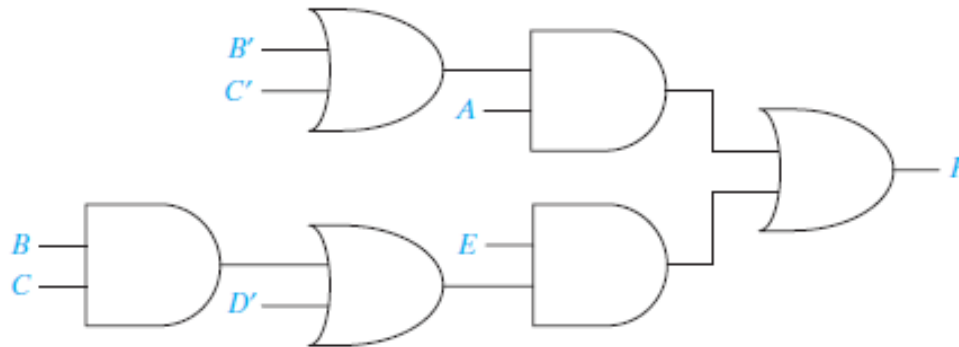
A two-level AND-OR circuit implementing F requires one 4-input OR, one 3-input AND, and three 2-input ANDs. To reduce the fan-in, F can be factored.

$$F = A(B' + C') + E(D' + BC) \quad (7-23)$$

The resulting four-level circuit using AND and OR gates is shown in Figure 7-18.

FIGURE 7-18
Limited Fan-In
Circuit

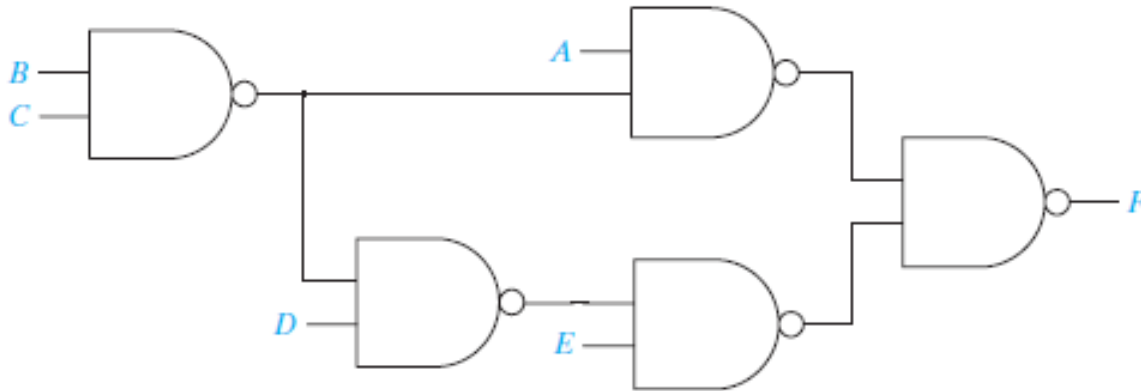
© Cengage Learning 2014



Circuit Conversion Using Alternative Gate Symbols

Example 4: Convert to NAND gates

Since the output gate is an OR, the circuit can be converted to NAND gates without increasing the number of levels; Figure 7-19 is the result. Note that the three-level OR with inputs B' and C' and the four-level AND with inputs B and C both become a NAND with inputs B and C ; hence, both can be replaced by the same gate.



Reducing the fan-in for some functions requires inserting inverters. The fan-in for $F = ABC + D$ can be reduced to 2 by factoring F as $F = (AB)C + D$. If this is implemented using two-input NAND gates, an inverter is required and the resulting circuit has four levels.

Design of Two-Level, Multiple-Output Circuits

Example 5: Multiple output realization

Design a circuit with four inputs and three outputs which realizes the functions

$$F_1(A, B, C, D) = \Sigma m(11, 12, 13, 14, 15)$$

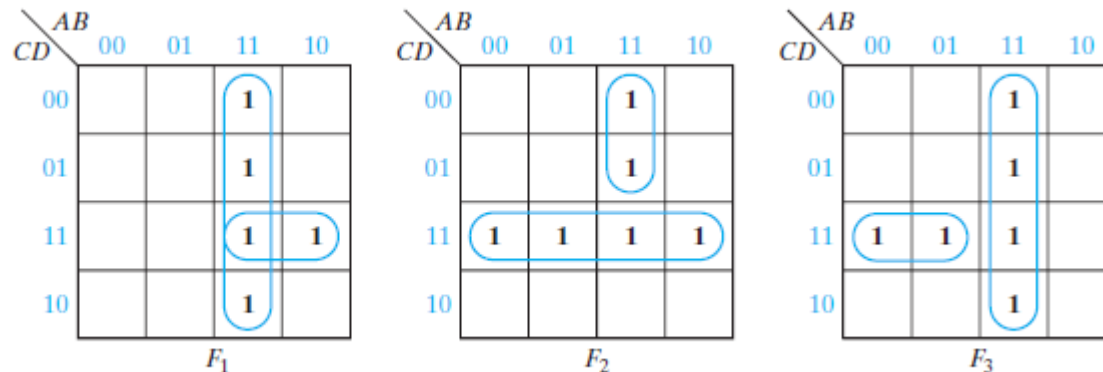
$$F_2(A, B, C, D) = \Sigma m(3, 7, 11, 12, 13, 15)$$

$$F_3(A, B, C, D) = \Sigma m(3, 7, 12, 13, 14, 15) \quad (7-24)$$

First, each function will be realized individually. The Karnaugh maps, functions, and resulting circuit are given in Figures 7-20 and 7-21. The cost of this circuit is 9 gates and 21 gate inputs.

FIGURE 7-20
Karnaugh Maps for
Equations (7-24)

© Cengage Learning 2014



Design of Two-Level, Multiple-Output Circuits

Example 5: Multiple output realization

FIGURE 7-21
Realization of
Equations (7-24)

© Cengage Learning 2014

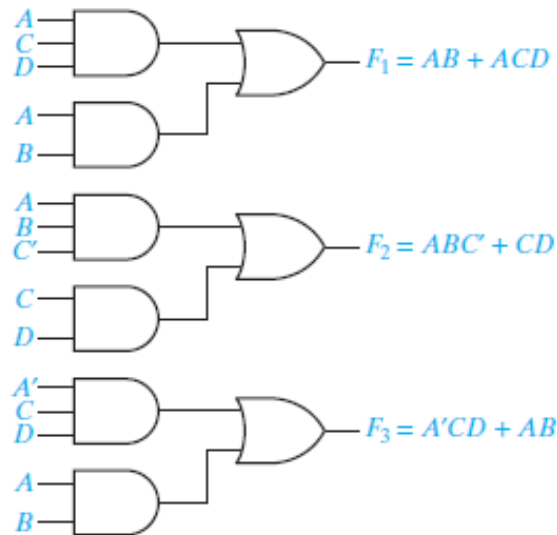
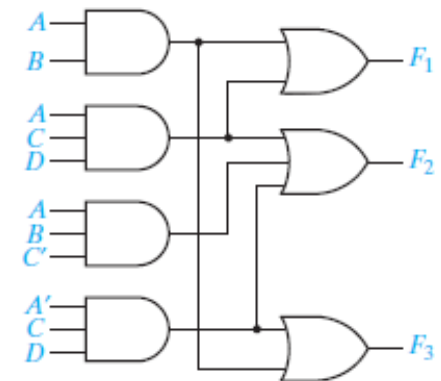


FIGURE 7-22
Multiple-Output
Realization of
Equations (7-24)

© Cengage Learning 2014



Design of Two-Level, Multiple-Output Circuits

Determination of Essential Prime Implicants for Multiple-Output Realization:

- We can find prime implicants, which are essential to one of the functions *and* to the multiple-output realization by a modification of the procedure used for the single-output case.
- In particular, when we check each 1 on the map to see if it is covered by only one prime implicant, we will only check those 1's which do not appear on the other function maps.

FIGURE 7-24
© Cengage Learning 2014

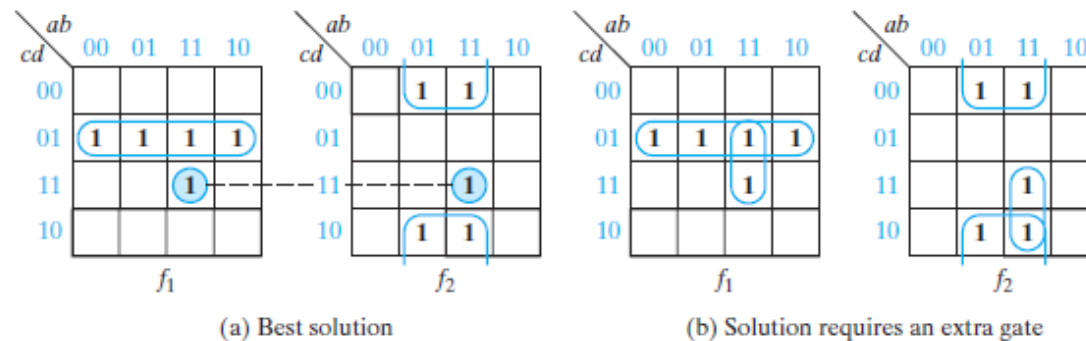
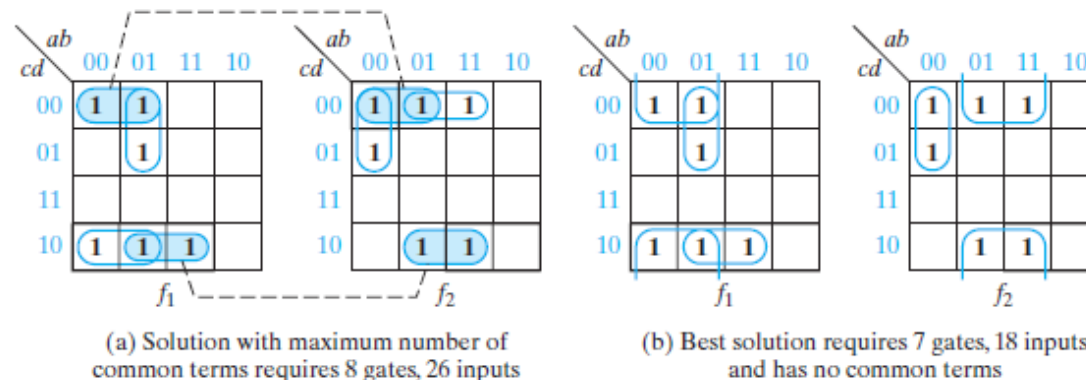


FIGURE 7-25
© Cengage Learning 2014



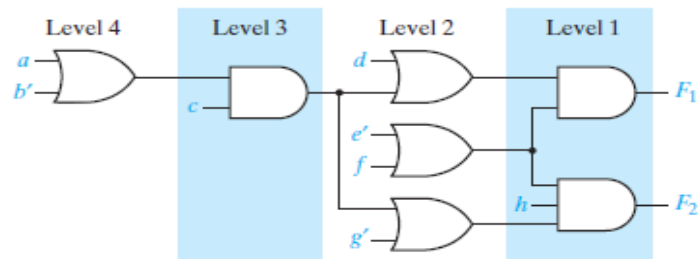
Multiple-Output NAND- and NOR-Gate Circuits

Converting a Two-Output Circuit to NOR Gates:

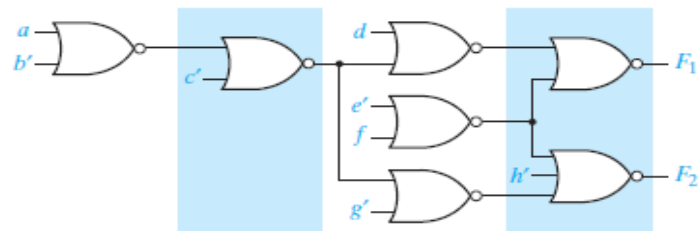
- The procedure given in Section 7.4 for design of single-output, multi-level NAND-and NOR-gate circuits also applies to multiple-output circuits.
- If all the output gates are AND, direct conversion to a NOR-gate circuit is possible.
- Note that the inputs to the first and third levels of NOR gates are inverted.

$$F_1 = [(a + b')c + d](e' + f)$$

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



(a) Network of AND and OR gates



(b) NOR network