Chapter3: Gate-Level Minimization

Lecture4- NAND and NOR Implementations, Other Two-level Forms

Engr. Arshad Nazir, Asst Prof
Dept of Electrical Engineering
SEECS

Fall 2021

Objectives

- Study NAND and NOR Implementations
- Wired Logic
- Non-degenerate

NAND and NOR Implementations

- Digital circuits are frequently constructed with NAND and NOR gates rather than with AND and OR gates due to following reasons:-
  - they are easier to fabricate with electronic components
  - they are used in all IC digital logic families
- NAND and NOR gates are said to be universal gates because they can perform all three basic functions namely AND, OR, and NOT operations and hence any logic function can be implemented with either all NAND or all NOR gates.
- Because of their use, rules have been developed that allow us to convert Boolean functions using AND, OR and NOT into the equivalent NAND and NOR logic diagrams.

NAND Circuits

- The NAND gate is a universal gate that can be used to construct any gate, therefore being able to replace all AND and OR gates.

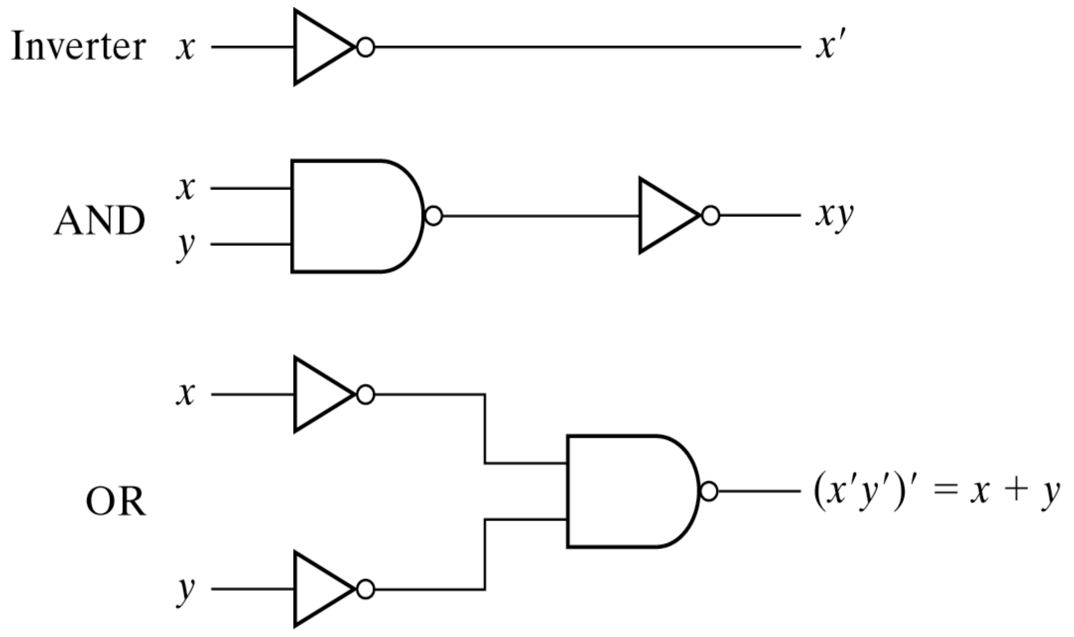
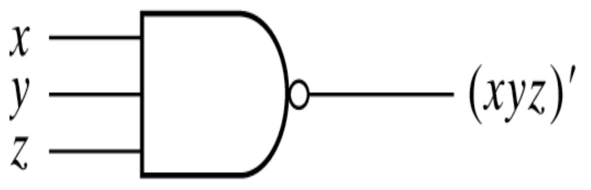


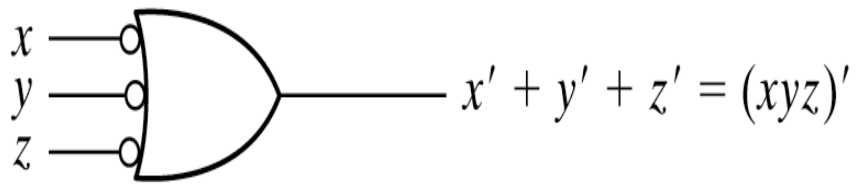
Fig. 3-18 Logic Operations with NAND Gates

NAND Notation

- A convenient method for creating a NAND circuit is to obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NAND logic.
- To facilitate the conversion to NAND logic we define equivalent alternative symbols as shown below for NAND gate



(a) AND–invert



(b) Invert–OR

Fig. 3-19 Two Graphic Symbols for NAND Gate

Two-Level Implementations

- The implementation of Boolean functions with NAND gates requires that the function be in sum of products form.
  - F = AB + CD
- All three diagrams are equivalent

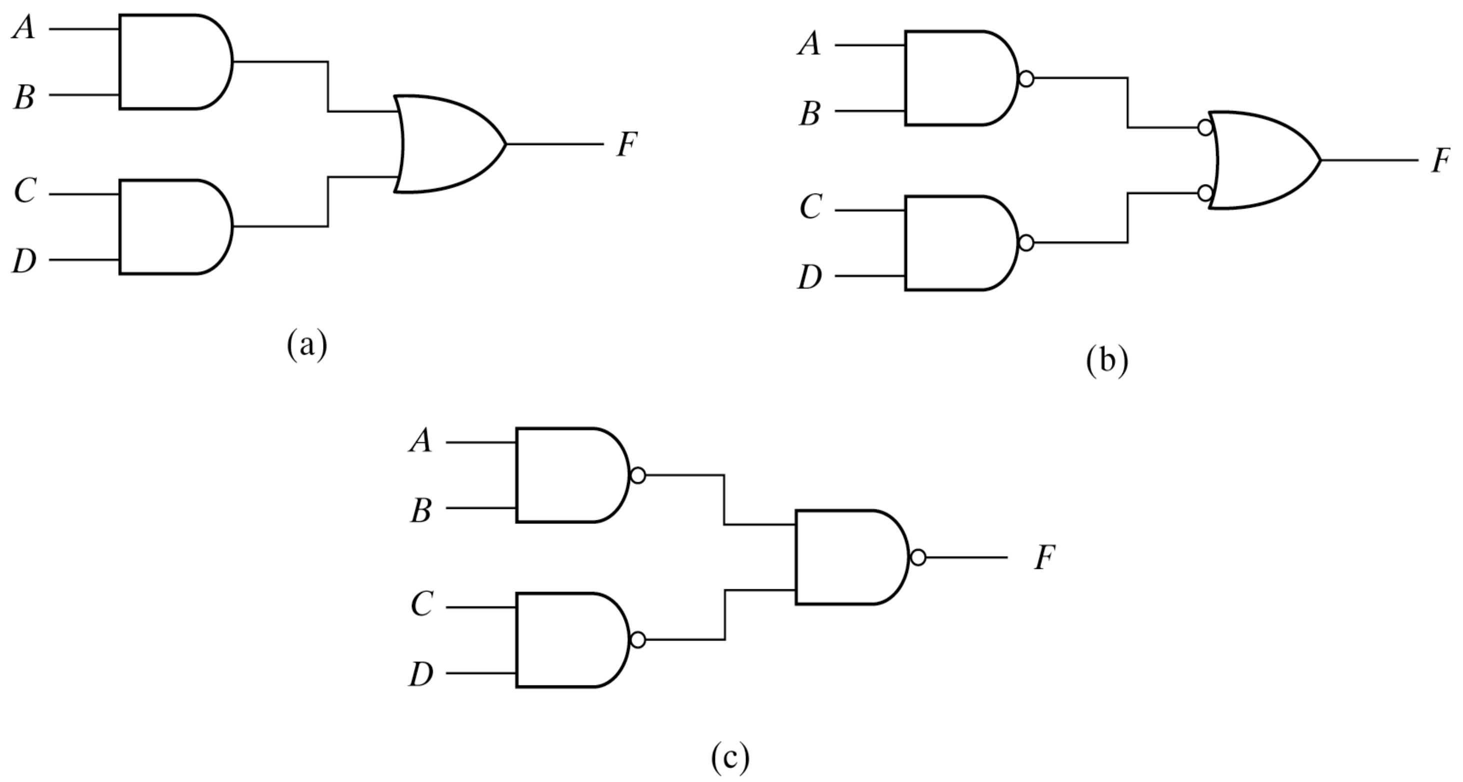
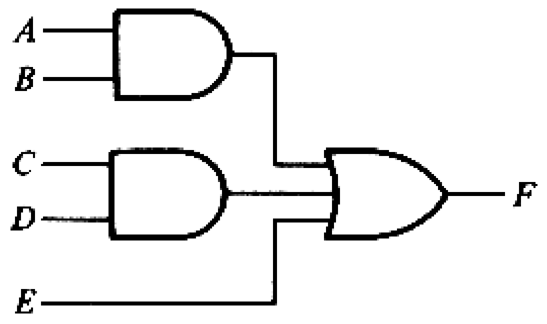


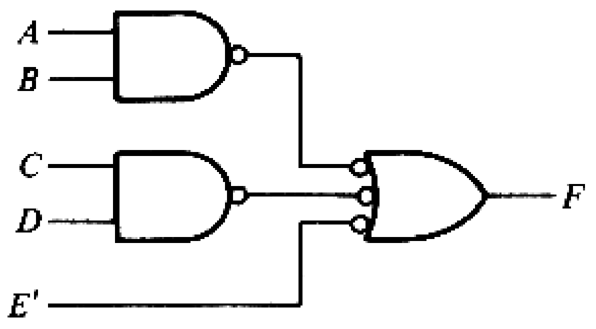
Fig. 3-20 Three Ways to Implement $F = AB + CD$

Two-Level Implementations

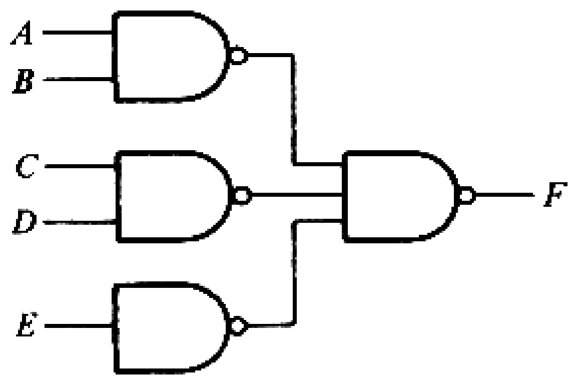
- F = AB+CD+E
- F = ((AB)' (CD)' E')' =AB+CD+E



(a) AND-OR



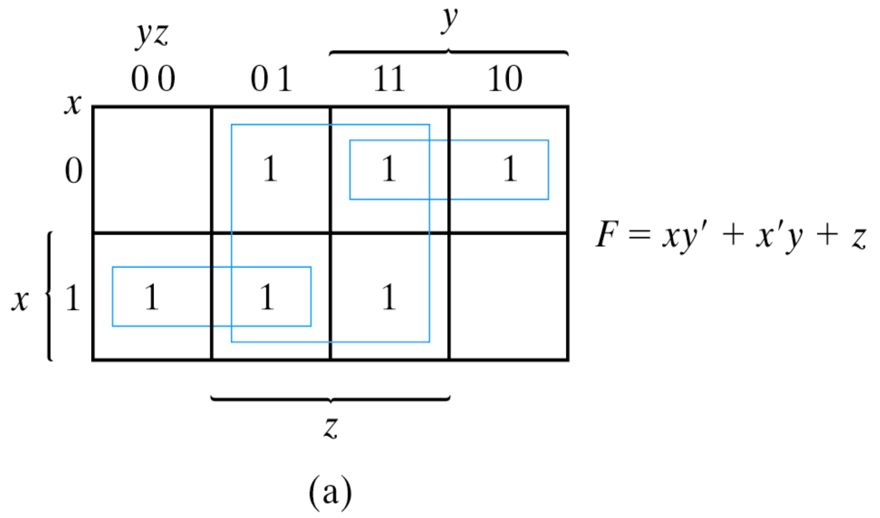
(b) NAND-NAND



(c) NAND-NAND

Example 3-10

- Implement F(x,y,z)= S(1,2,3,4,5,7) with NAND gates



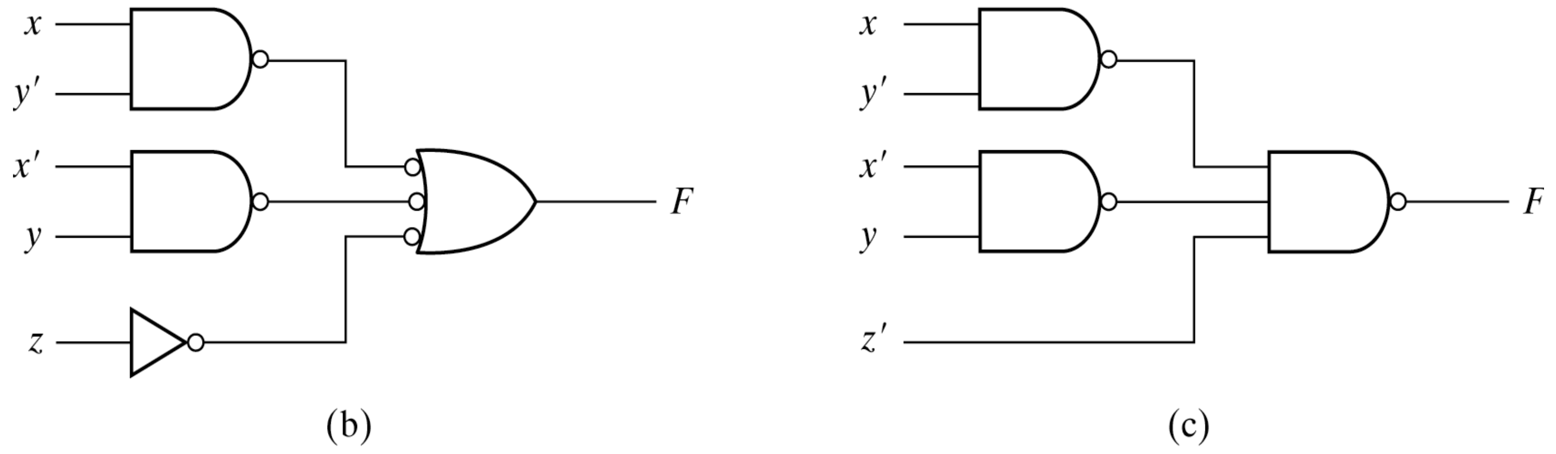


Fig. 3-21 Solution to Example 3-10

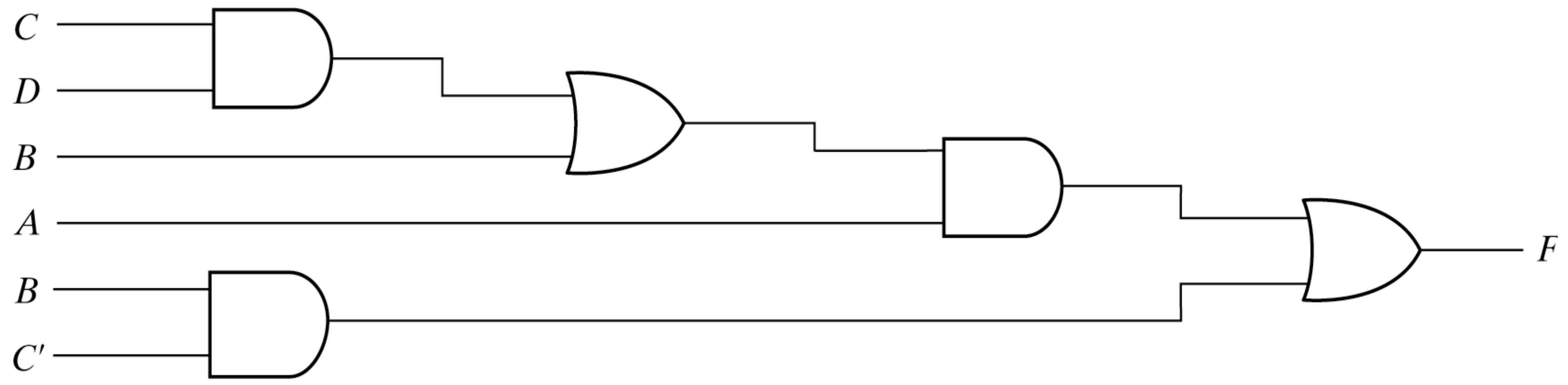
Two-Level NAND Rules

- Given a Boolean function, follow these rules to obtain the NAND logic diagram:
  - ➢ Simplify the function and express it in sum of products
  - ➢ Draw a NAND gate for each product term of the expression that has at least two literals. This is group of first level gates
  - ➢ Draw a single gate using the AND-invert or the invert-OR graphic symbol in the second level, with inputs coming from outputs of first level gates
  - ➢ A term with a single literal requires an inverter in the first level, unless the single literal is already complemented

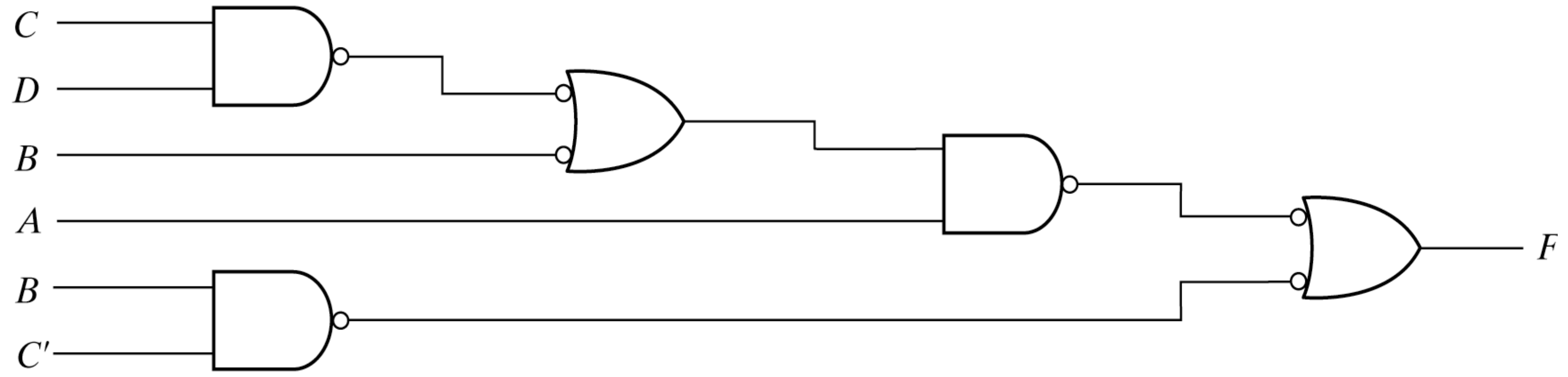
Multilevel NAND Circuits

- The general procedure for converting a multi-level AND-OR diagram into an all-NAND diagram is as follows:
  - Convert all AND gates to NAND gates with AND-invert graphic symbols
  - Convert all OR gates to NAND gates with invert-OR graphic symbols
  - Check all the bubbles in the diagram
    - Every bubble that is not compensated by another along the same line will require the insertion of an inverter or complement the input literal

Multilevel NAND Example



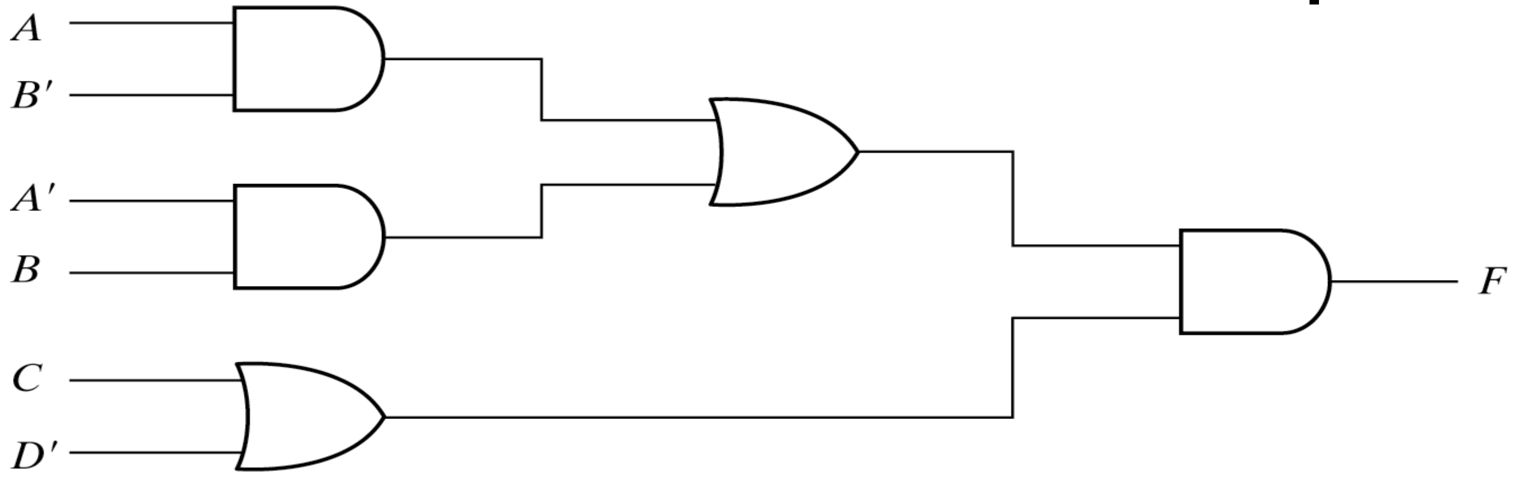
(a) AND-OR gates



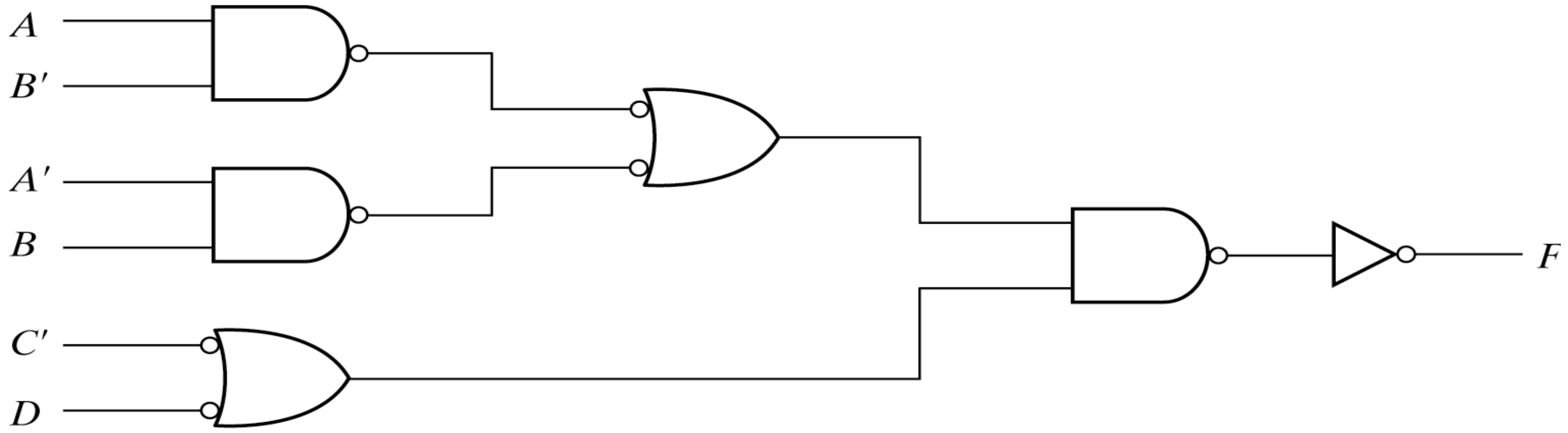
(a) NAND gates

Fig. 3-22 Implementing $F = A(CD + B) + BC$

Multilevel NAND Example



(a) AND-OR gates



(b) NAND gates

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

NOR Circuits

- The NOR operation is a dual of the NAND operation and therefore all procedures and rules for NOR logic are the dual of the corresponding procedures and rules for the NAND logic.

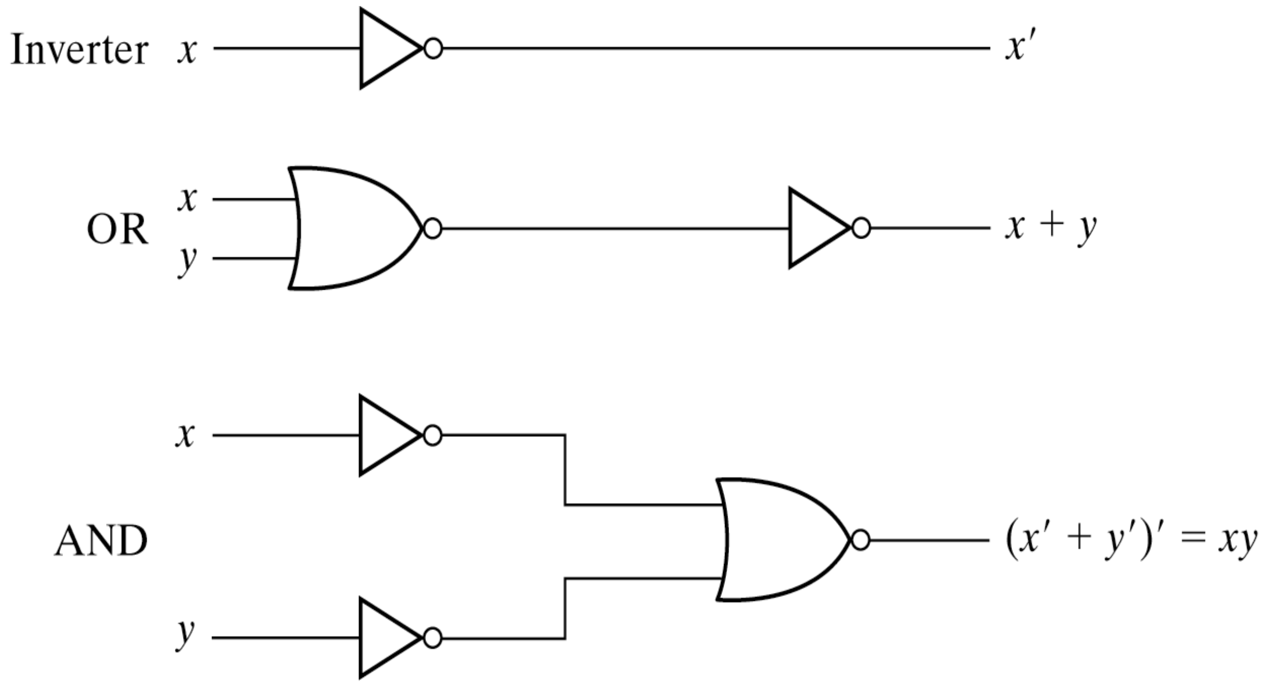
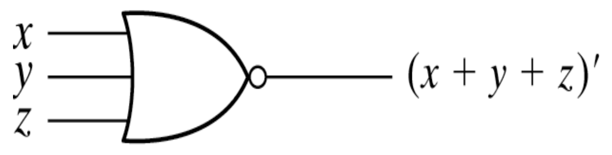


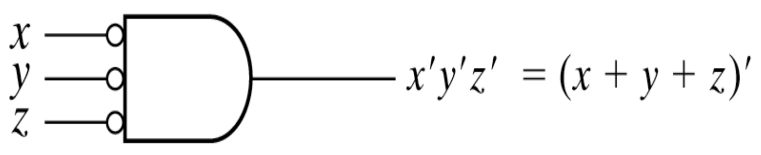
Fig. 3-24 Logic Operations with NOR Gates

NOR Notation

- A convenient method for creating a NOR circuit is to obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NOR logic.



(a) OR–invert



(a) Invert–AND

Fig. 3-25 Two Graphic Symbols for NOR Gate

Two-Level NOR Implementations

- The implementation of Boolean functions with NOR gates requires that the function be in product of sums form.

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

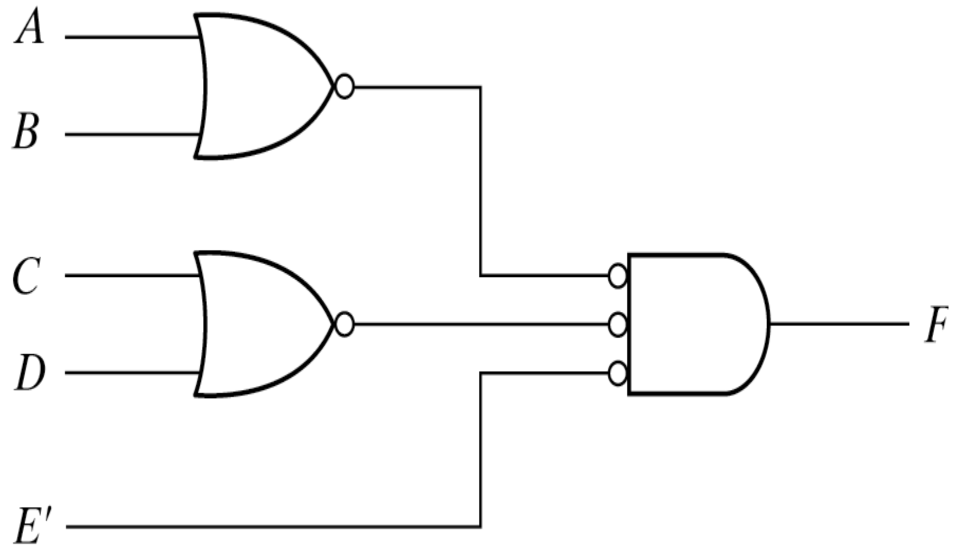


Fig. 3-26 Implementing $F = (A + B)(C + D)E$

Multilevel NOR Implementations

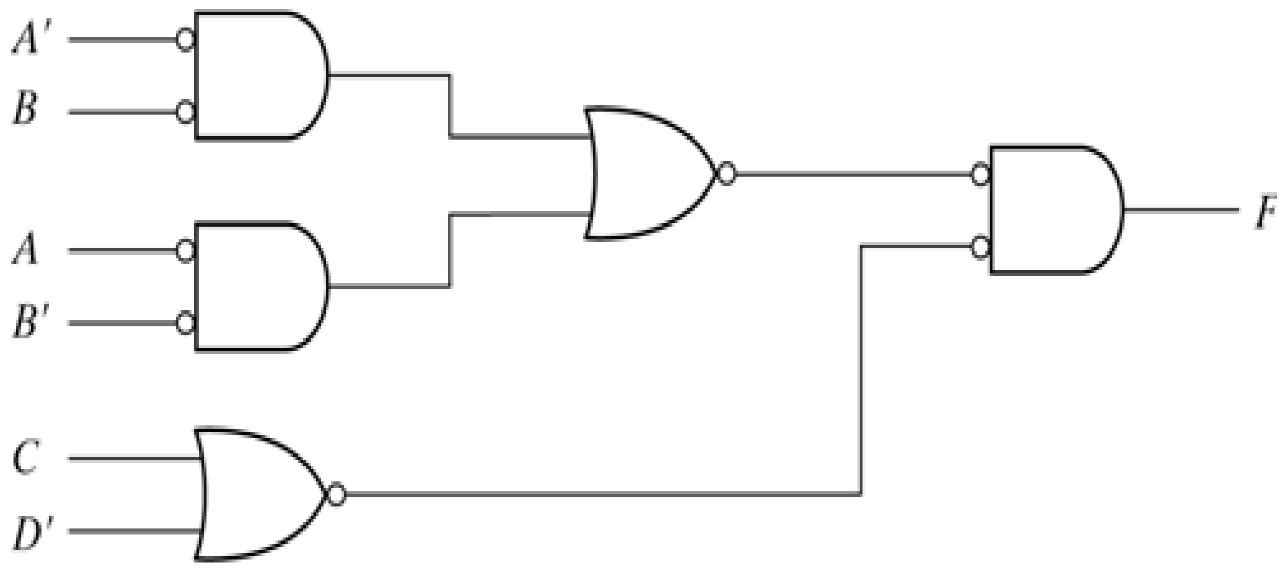
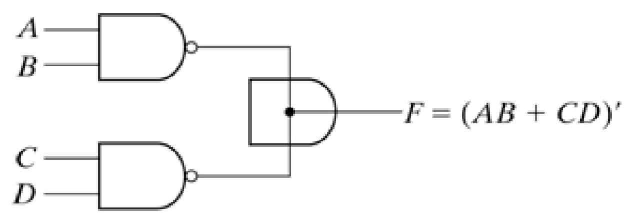


Fig. 3-27 Implementing $F = (AB' + A'B)(C + D')$ with NOR Gates

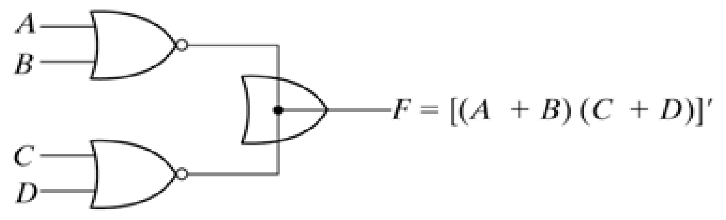
Other Two-Level Implementations

- The types of gates most often used in integrated circuits are NAND and NOR gates. For this reason, NAND and NOR logic implementations are the most important from a practical point of view.
- Some(but not all) NAND or NOR gates allow the possibility of a wired connection between the outputs of two gates to provide a specific logic function.
- For example, open-collector TTL NAND gates, when tied together, perform wired-AND logic. Similarly, the NOR outputs of ECL gates can be tied together to perform a wired-OR function. Both are depicted below



(a) Wired-AND in open-collector TTL NAND gates.

(AND-OR-INVERT)



(b) Wired-OR in ECL gates

(OR-AND-INVERT)

Fig. 3-28 Wired Logic

Non-degenerate Forms

- Consider that we have four types of gates:

  AND, OR, NAND, and NOR.

- In a two-level circuit, we can have as many as 16 combinations of two-level forms:
  - Eight of these combinations are called degenerate forms because they degenerate to a single operation
    - For example, an AND-AND circuit is simple an AND of all inputs
- The eight non-degenerate forms are:

Combine ‘1s’ on map	Combine ‘0s” on map
AND-OR	**OR-AND**
NAND-NAND	**NOR-NOR**
NOR-OR	**NAND-AND**
OR-NAND	**AND-NOR**

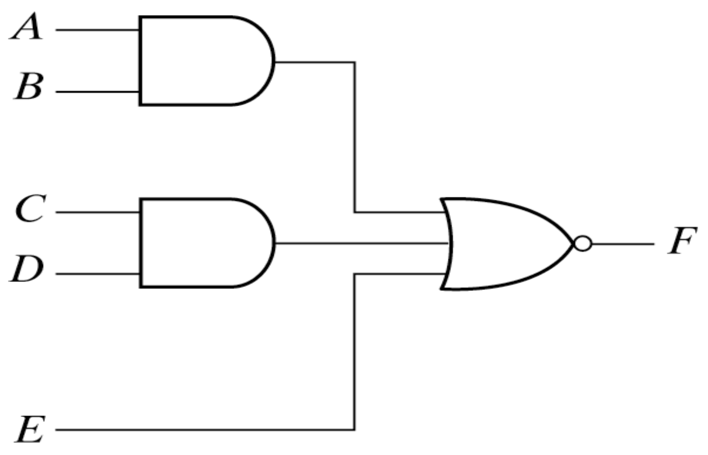
AND-OR-INVERT Implementation

- The two forms NAND-AND and AND-NOR are equivalent and can be treated together.
- Both perform the AND-OR-INVERT function
- AND-NOR resembles the AND-OR except for the inversion done by the bubble in the output of NOR gate.
- Therefore, if the complement of the function is simplified into sum-of-products form (by combining 0’s on the map, it will be possible to implement F’ with the AND-OR part of the function.
- In the figure (next slide) the function implemented is

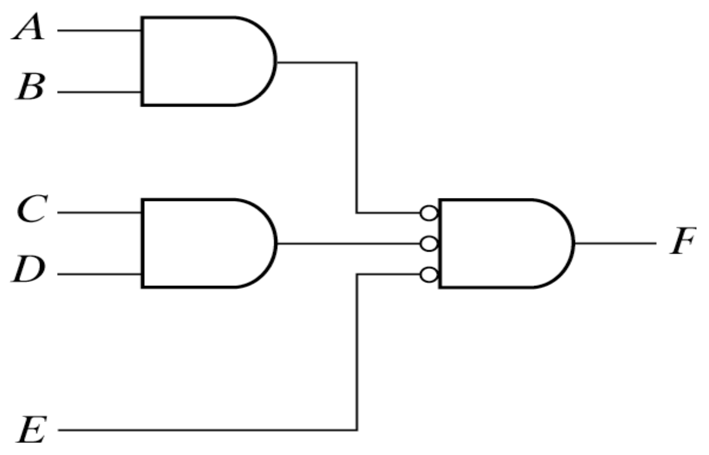
  $F = (AB+CD+E)'$

  $F' = AB+CD+E$ (sum of products for F’)

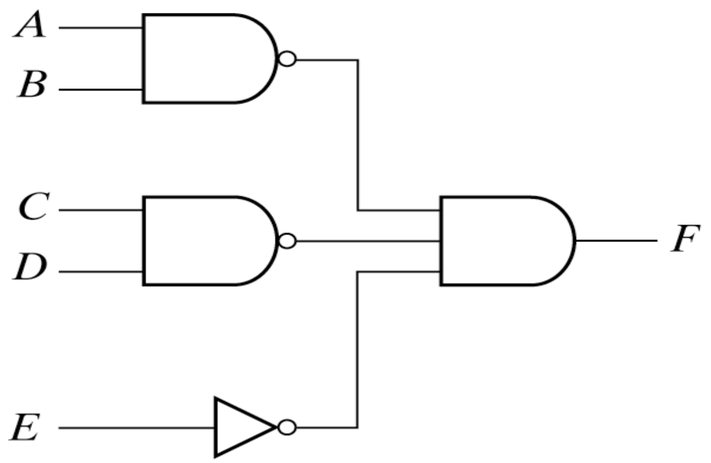
AND-OR-INVERT Implementation



(a) AND-NOR



(b) AND-NOR



(c) NAND-AND

Fig. 3-29 AND-OR-INVERT Circuits; $F = (AB + CD + E)'$

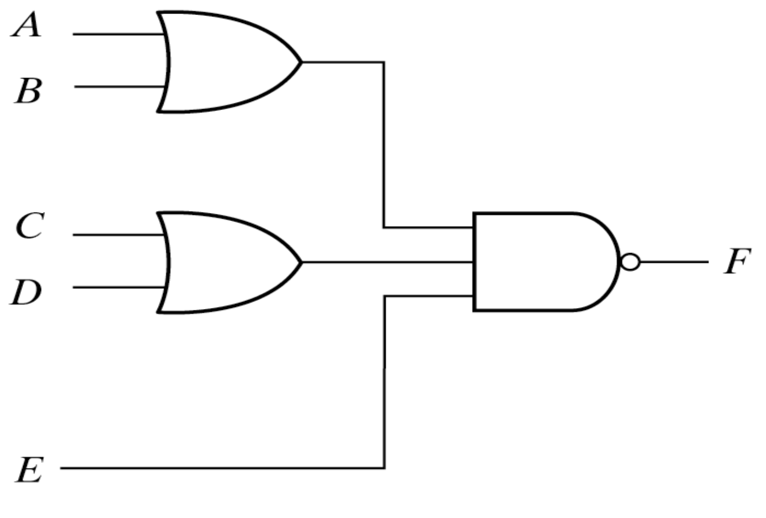
OR-AND-INVERT Implementation

- The OR-NAND and NOR-OR are equivalent and can be treated together.
- Both forms perform the OR-AND-INVERT function
- OR-NAND resembles the OR-AND except for the inversion done by the bubble in the output of NAND gate.
- Therefore, if the complement of the function is simplified into product-of-sums form (by combining 1's on map), it will be possible to implement F' with the OR-AND part of the function.
- In the figure (next slide) the function implemented is

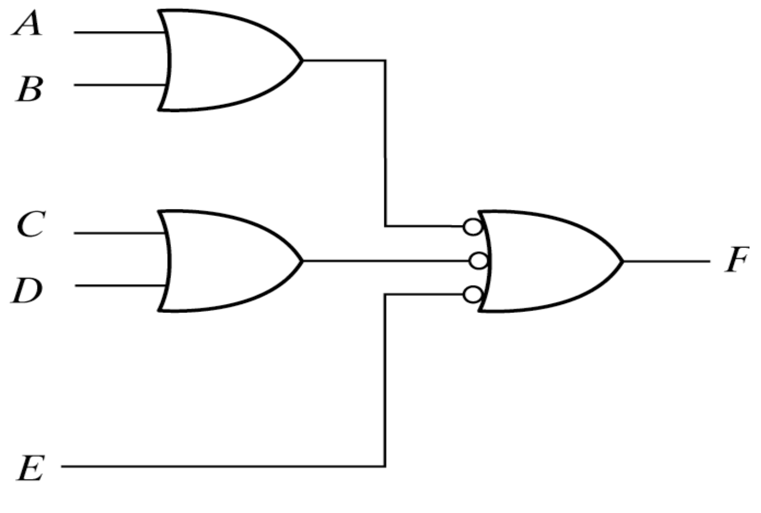
$$F = [\,(A+B)(C+D)E\,]'$$

$$F' = (A+B)(C+D)E \qquad \text{(product of sums for } F'\text{)}$$

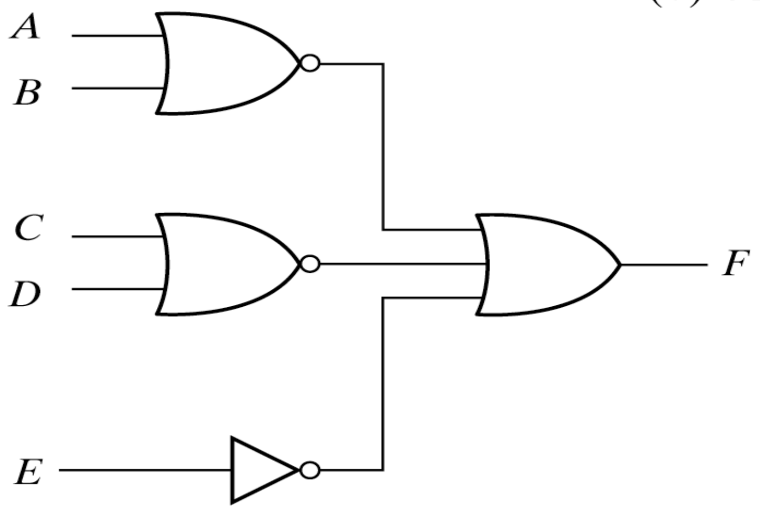
OR-AND-INVERT Implementation



(a) OR-NAND



(b) OR-NAND



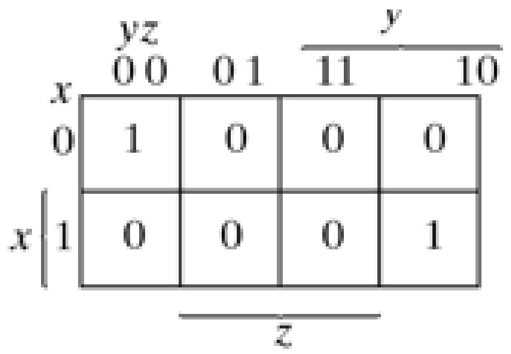
(c) NOR-OR

Fig. 3-30 OR-AND-INVERT Circuits; $F = [(A + B)(C + D)E]'$

Non-degenerate Forms

Example: Implement the following Boolean function F, using the Two-Level form of logic (a) AND-NOR (b) NAND-AND (c) OR-NAND and NOR-OR

F(x,y,z)=∑(1,6)



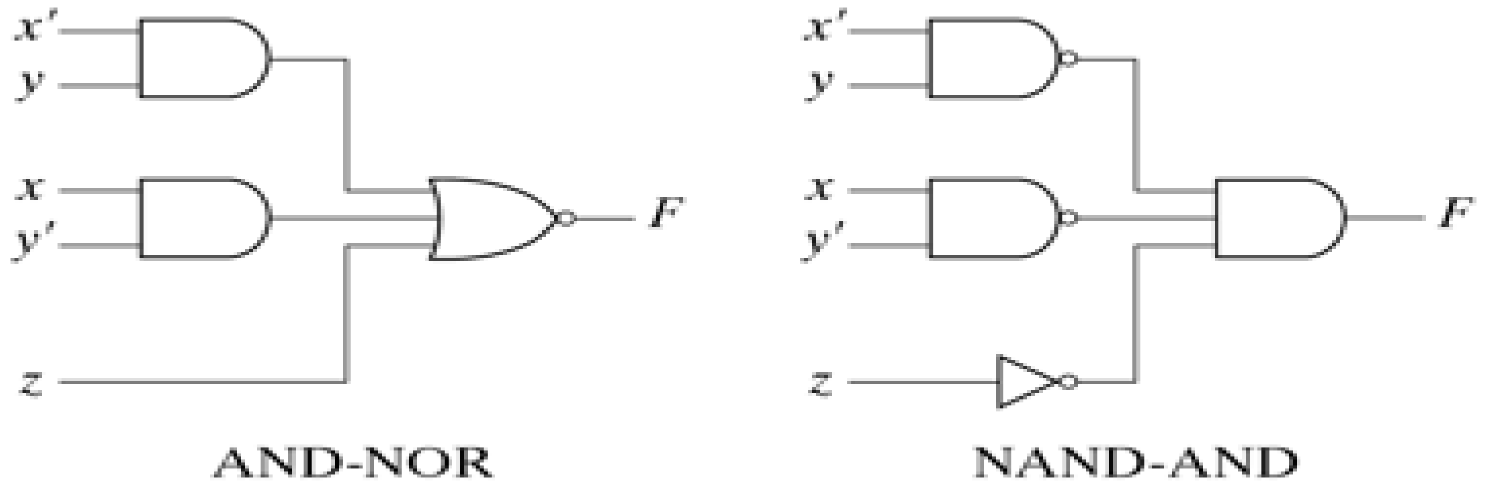
x \ yz	00	01	11	10
0	1	0	0	0
1	0	0	0	1

$F = x'y'z' + xyz'$

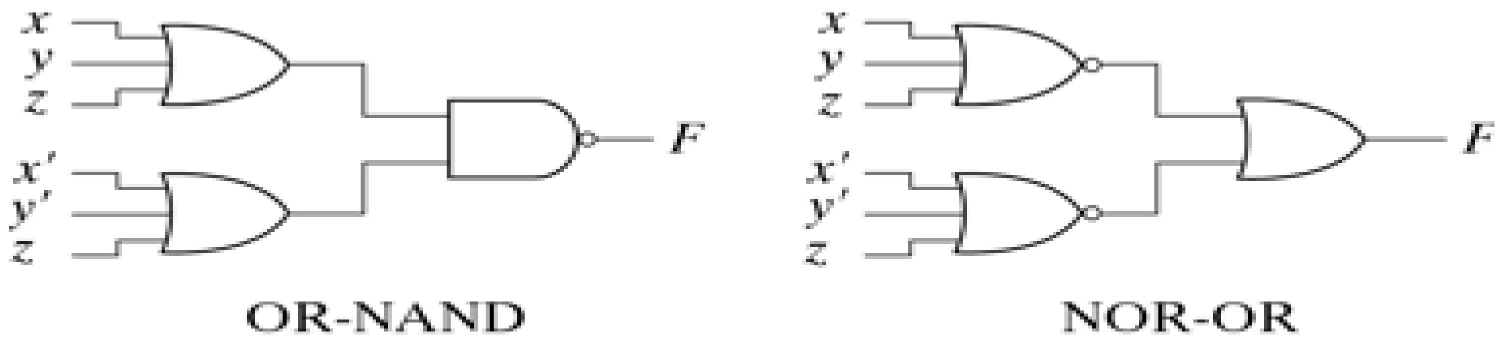
$F' = x'y + xy' + z$

(a) Map simplification in sum of products.

Non-degenerate Forms



(b) $F = (x'y + xy' + z)'$



(c) $F = [(x + y + z)(x' + y' + z)]'$

Fig. 3-31 Other Two-level Implementations

The End