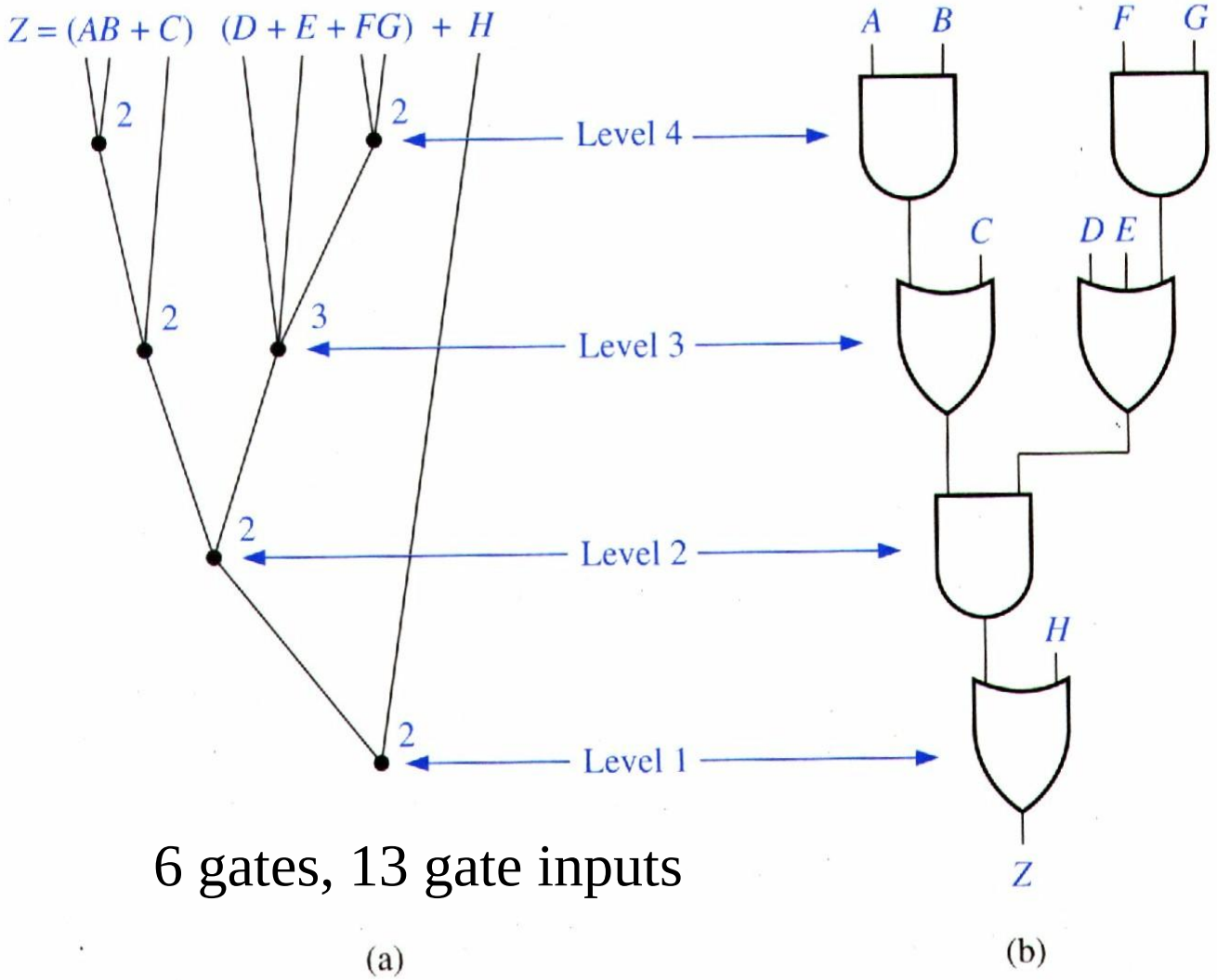
Lecture 7 Multi-Level Gate Networks

- AND-OR network: two-level
  - SOP expression. Using factoring to increase the number of levels.
- OR-AND network: two-level
  - POS expression: Using multiplying out to increase the number of levels.
  - Why?
    - It may reduce the number of gates or gate inputs.
    - The number of gates which can be cascaded is limited by the gate delays.

Example

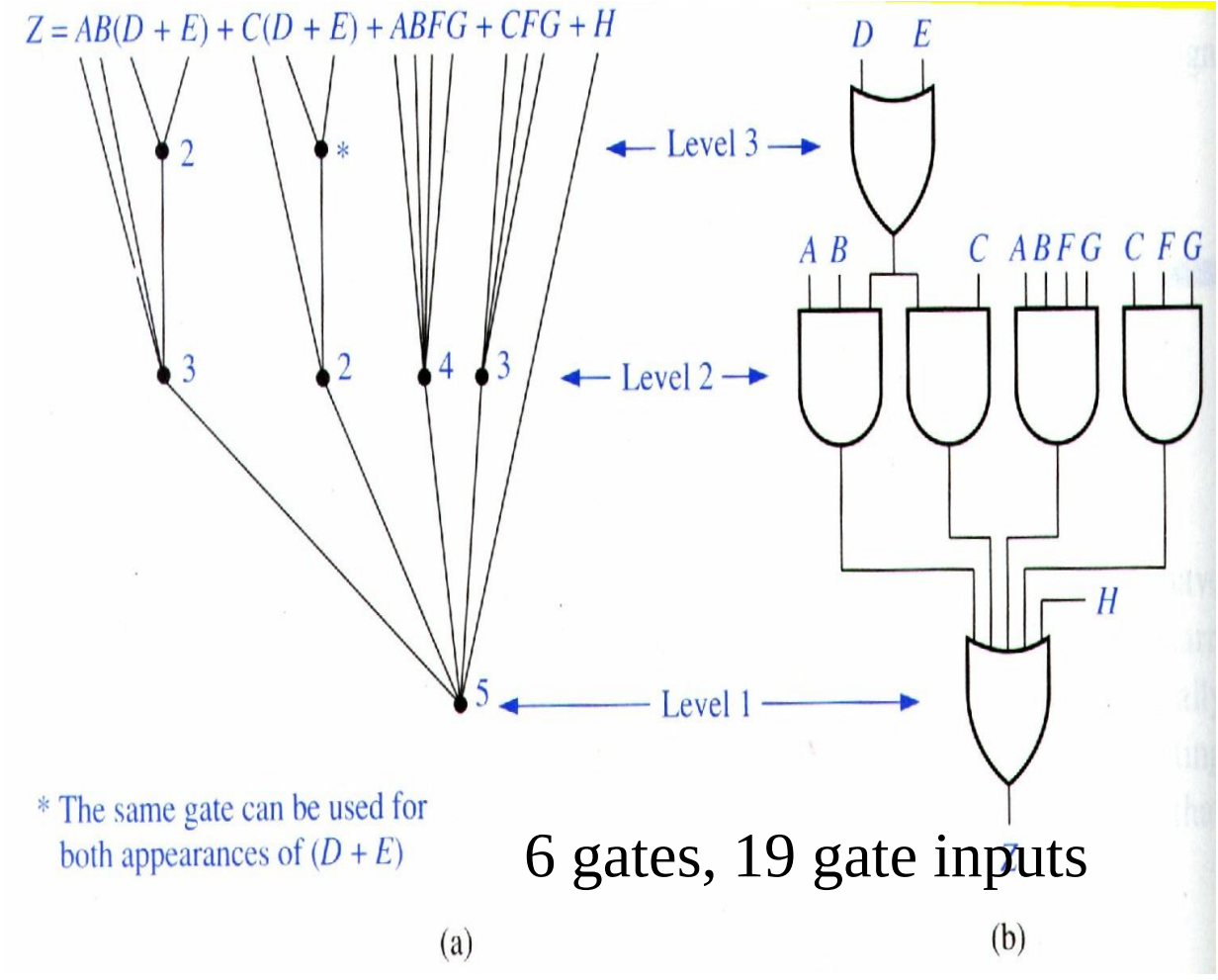
- A four-level realization of Z
  - Z = (AB + C)[(D+E) + FG] + H and
  - Z = AB(D+E) + C(D+E) +ABFG + CFG + H



6 gates, 13 gate inputs

Example (cont.)

- A three-level realization of Z
  - Z = (AB + C)[(D+E) + FG] + H and
  - Z = AB(D+E) + C(D+E) +ABFG + CFG + H



6 gates, 19 gate inputs

Multi-Level Design

- Compare among AND-OR, OR-AND-OR, OR-AND, and AND-OR-AND network

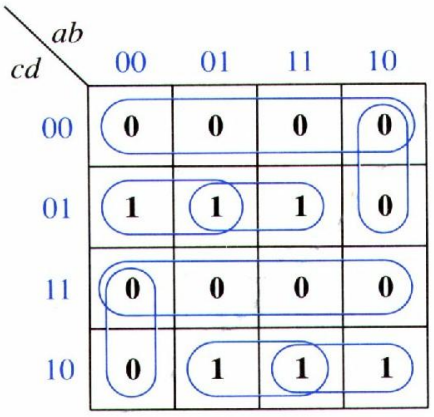
EXAMPLE OF MULTI-LEVEL DESIGN USING AND AND OR GATES

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

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

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



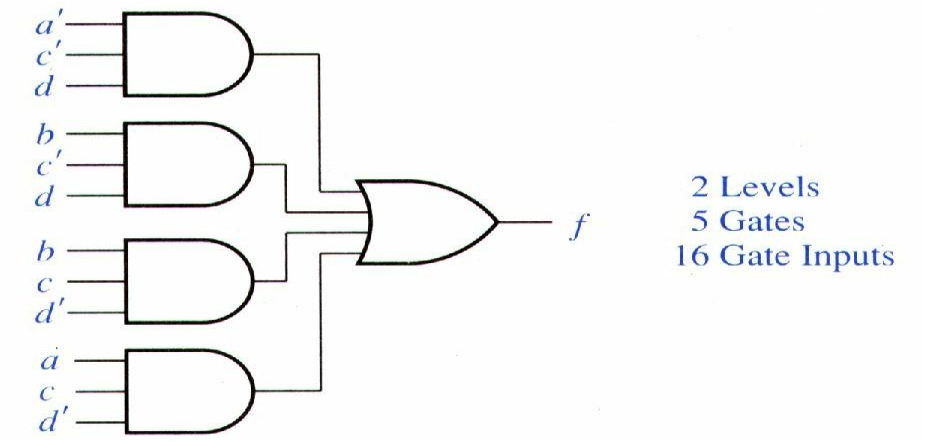
$f = a'c'd + bc'd + bcd' + acd'$

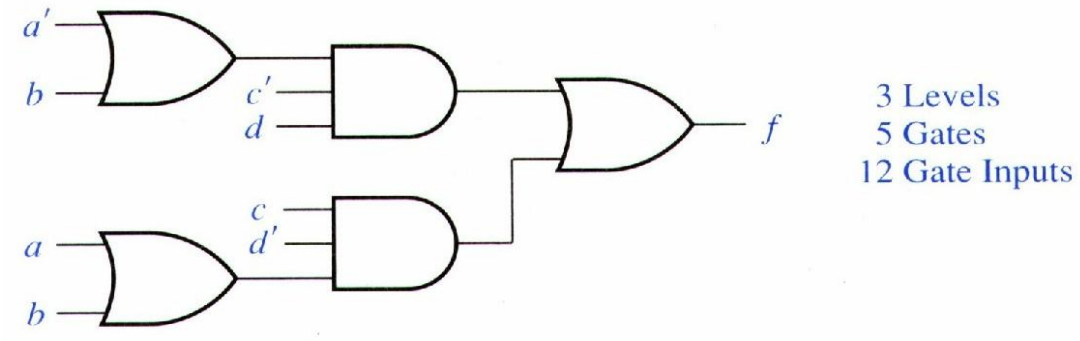
Multi-Level Design (cont.)

- OR output

  f = a'c'd + bc'd + bcd' +acd'

  f = c'd(a'+b) + cd'(a+b); factoring f





Multi-Level Design (cont.)

- AND output

  f’ = c’d’ + ab’c’ + cd + a’b’c’

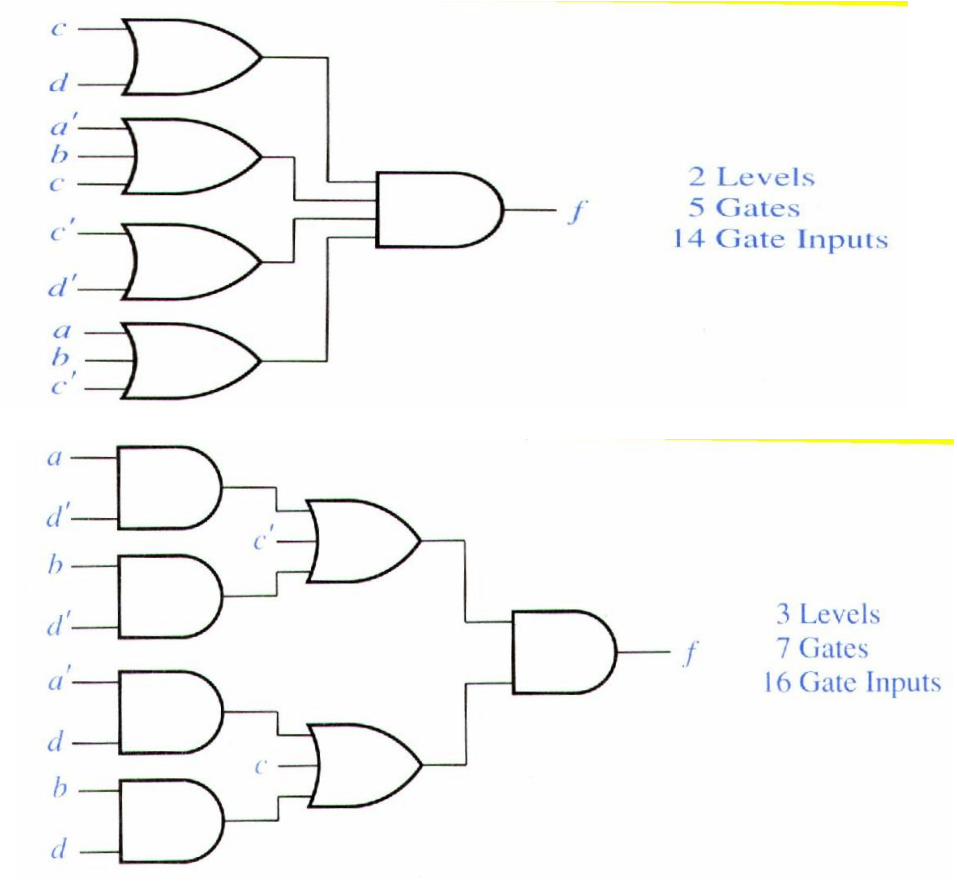
  f = (c+d)(a’+b+c)(c’+d’)(a+b+c’)

  Use (X+Y)(X+Z) = X +YZ

  f = [c + d(a’ + b)][c’ + d’(a+b)] (4-level)

  f = (c + a’d + bd)(c’ +ad’ + bd’) (3-level)

  Which one is better?



Rule of Thumb

- To find an OR output gate
  - Use SOP expression
    - f = a'c'd + bc'd + bcd' +acd'
      - To find an OR-AND-OR gate network
        - » Factor an SOP
        - » f = c'd(a'+b) + cd'(a+b)
- To find an AND output gate
  - Use POS expression
    - f' =c'd' + ab'c' + cd +a'b'c (K-map)
    - Then use DeMorgan Law to find f
    - f = (c+d)(a'+b+c)(c'+d')(a+b+c)
      - To find a three level network with AND output
        - » Partially multiply out the POS expression
        - » f = (c+a'd + bd)(c'+ad' + bd')

NAND and NOR Gate

- NAND gate

  F = (ABC)’ = A’ + B’ + C’.

  F = 1 iff one or more of its inputs are 0.

  F = 0 if all inputs are 1.

- NOR gate

  F = (A + B + C)’ = A’B’C’

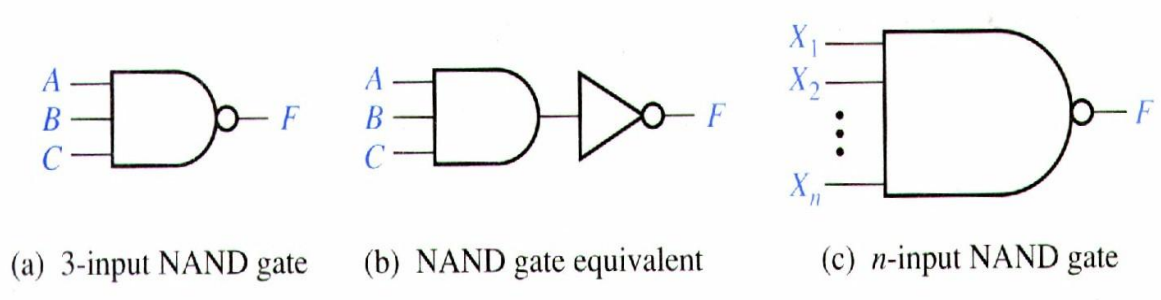
  F = 1 iff all inputs are 0.

  F = 0 if any input is 1.

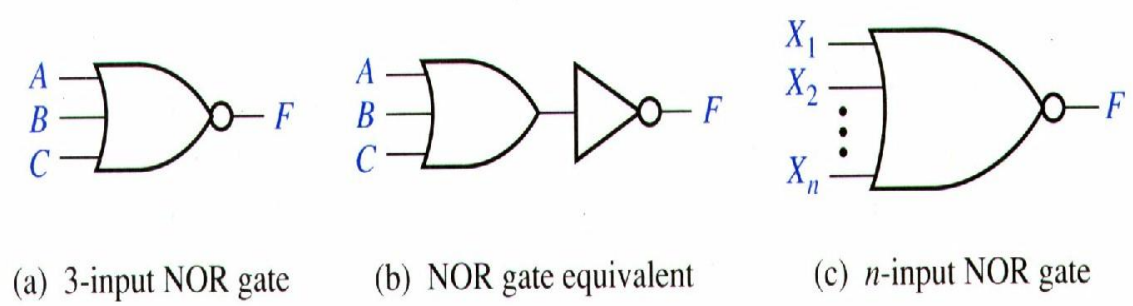
- NAND and NOR are duals.
  - If a given circuit realizes the NAND function for positive logic, it realizes the NOR function for negative logic.

NAND and NOR (cont.)

- Symbols



(a) 3-input NAND gate (b) NAND gate equivalent (c) *n*-input NAND gate

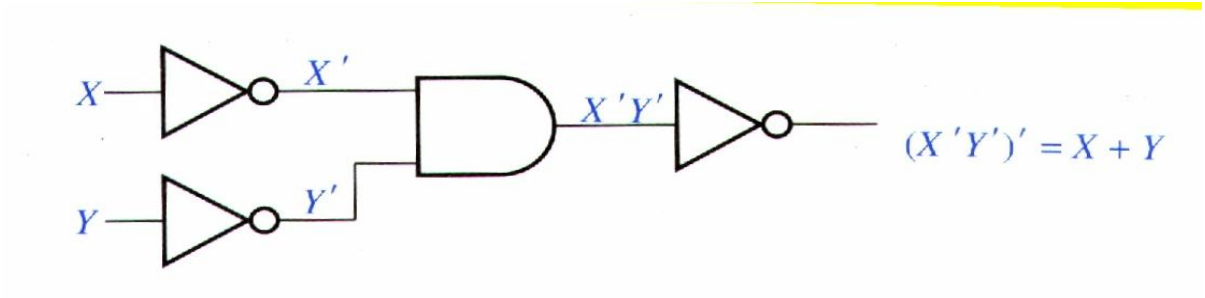


(a) 3-input NOR gate (b) NOR gate equivalent (c) *n*-input NOR gate

Functionally Complete

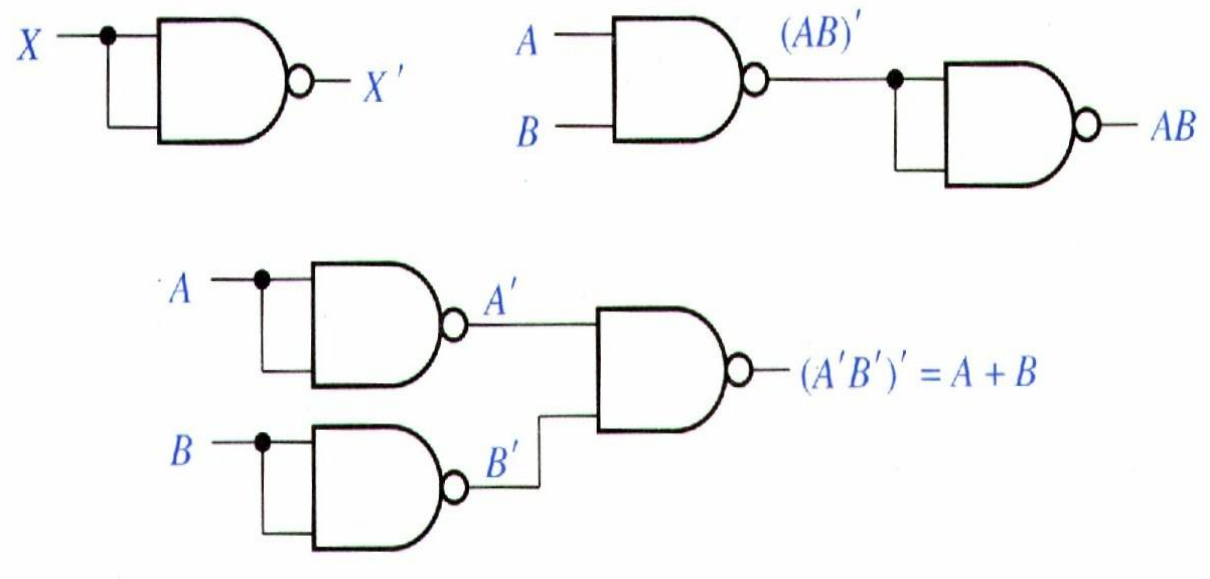
- Functionally complete: If any Boolean function can be expressed in terms of this set of operations.
  - {AND, OR, NOT) is functionally complete since any function can be expressed in an SOP form.
  - {AND and NOT} is a functionally complete set of gates since OR can be realized using AND and NOT since

    X + Y = (X'Y')'



Functionally Complete Gate

- NAND gate
  - Since AND and NOT are functionally complete set of gates.
  - Any function can be realized using only NAND gates.



NAND and NOR why uses them?

- In design using gate level IC, NAND and NOR are usually used because
  - They are faster than AND and OR gates.
  - NAND and NOR are functionally complete gates.
  - More fan-ins variety to choose for NAND and NOR gate ICs.

Convert SOP to Other Networks

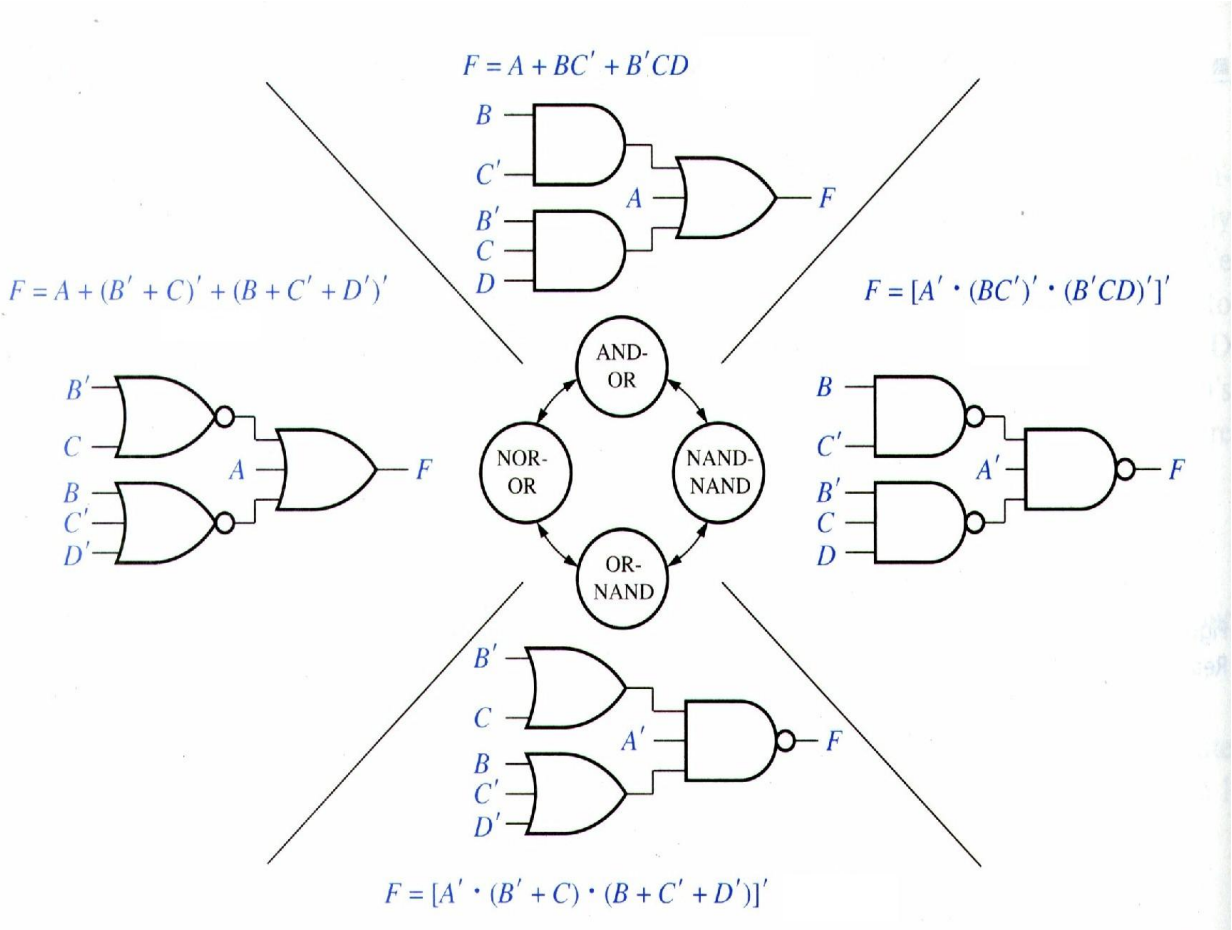
F = A + BC’ + B’CD AND-OR

= [(A+BC’+B’CD)’]’

= [A’.(BC’)’.(B’CD)’]’ NAND

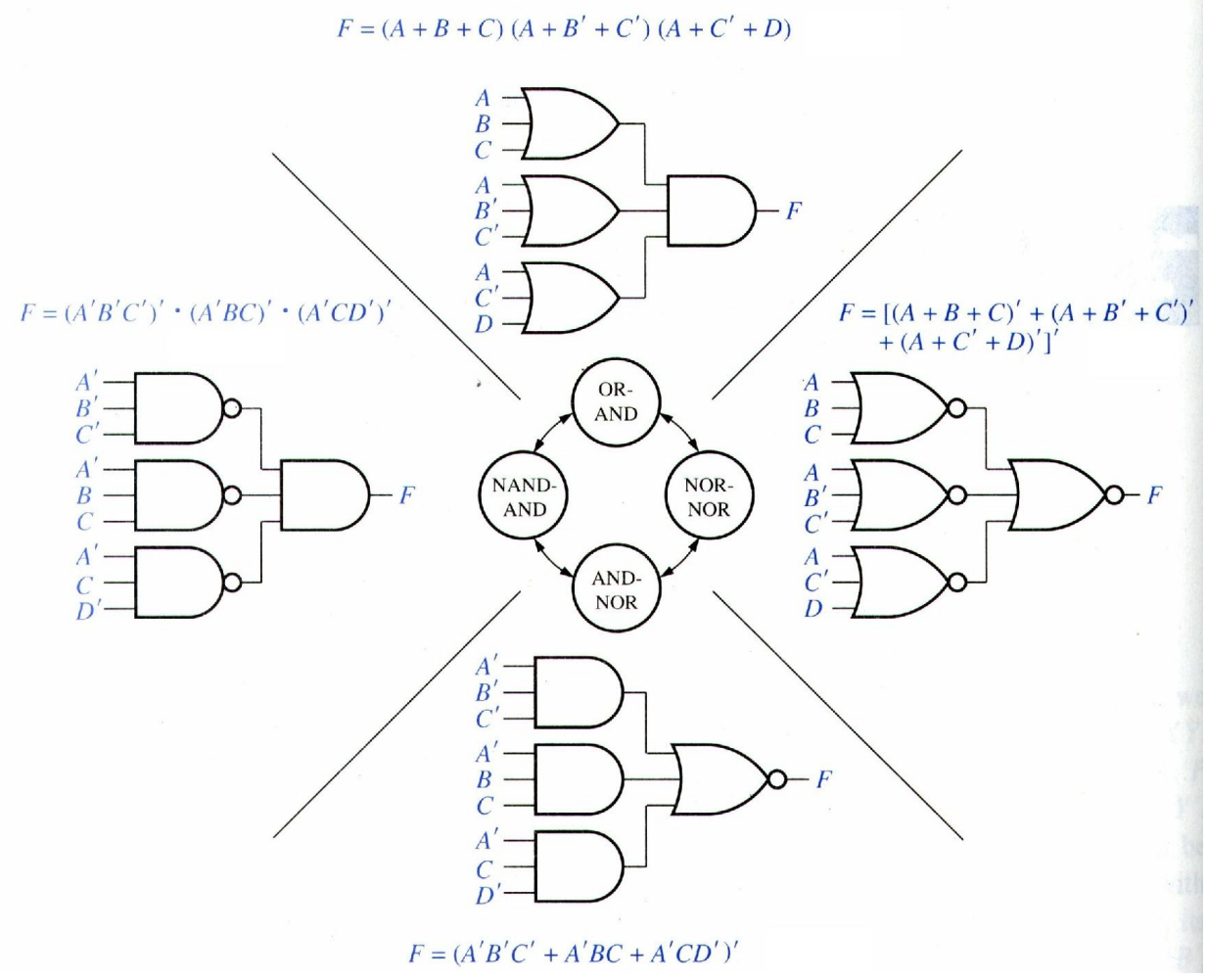
= [A’.(B’+C).(B+C’+D’)]’ OR-NAND

= A + (B’+C)’ + (B+C’+D’)’ NOR-OR



Convert POS to Other Forms

- F = (A+B+C)(A+B'+C')(A+C'+D) OR-AND
  = [{(A+B+C)(A+B'+C')(A+C'+D)]'}'
  = [(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

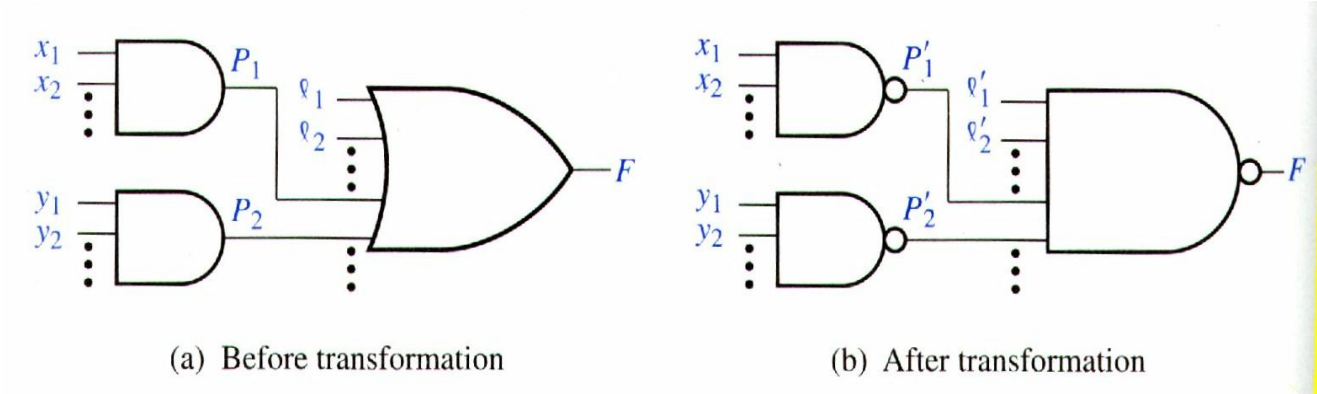


Design a NAND-NAND Network

- F = Minimum SOP
  - Assume that all variables and their complements are available as inputs.
  - Draw AND-OR network for F
  - Replaces all gates with NAND gates and complement single literals of the inputs for the output gates. For example,

$F = l_1 + l_2 + \ldots P_1 + P_2$

$= (l_1'l_2'\ldots P_1'.P_2'\ldots)'$



(a) Before transformation

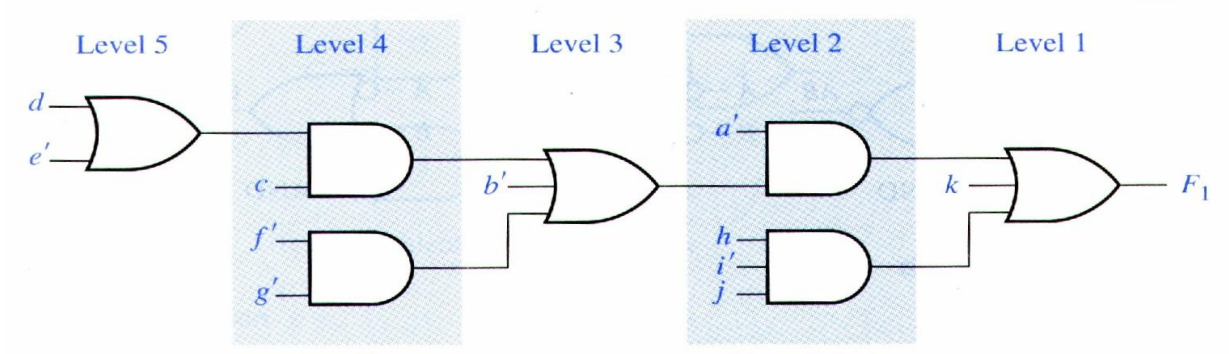
(b) After transformation

Design a NOR-NOR Network

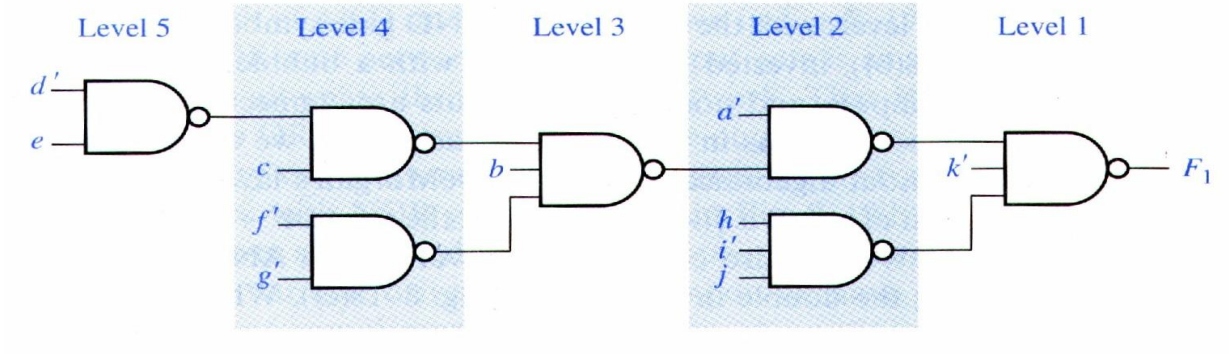
- F = Minimum POS
  - Assume that all variables and their complements are available as inputs.
  - Draw OR-AND network for F
  - Replaces all gates with NOR gates and complement single literals of the inputs for the output gates.

Design Multi-Level NAND Gate Networks

- Repeat two-level AND-OR to two-level NAND-NAND procedure. Invert any literals which appear as inputs to level 1,3,5..
- F1 = a'[b' + c(d + e') + f'g'] + hi'j + k



(a) AND-OR network

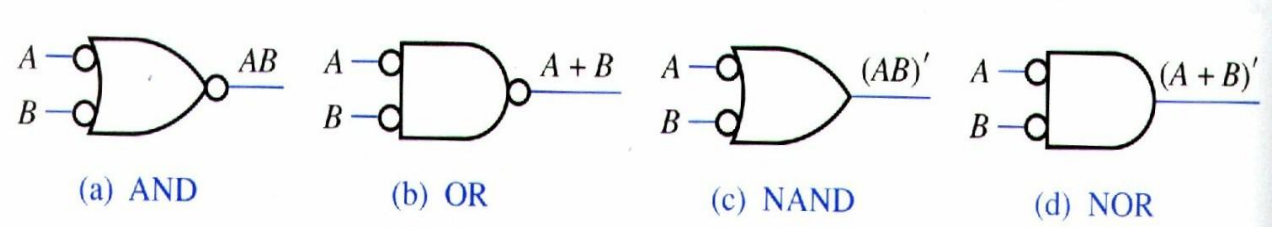


(b) NAND network

Alternative Gate Symbols

- Help in analysis of NAND and NOR gate network.

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



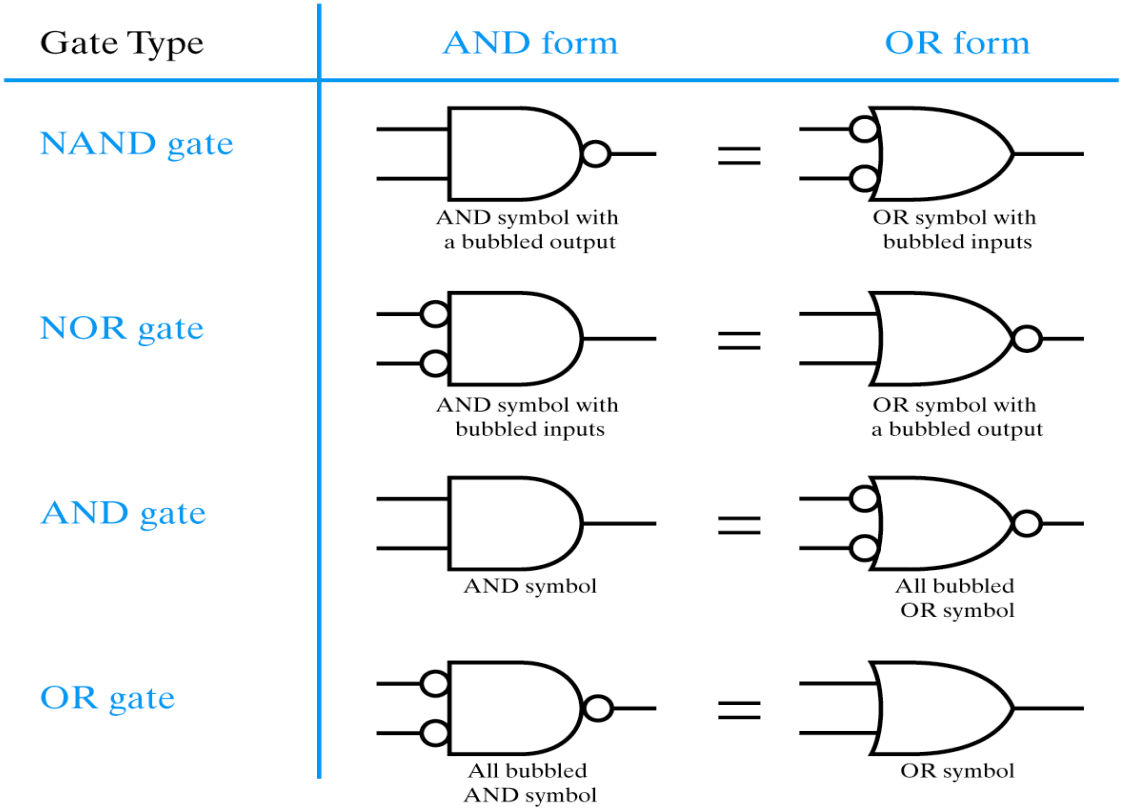
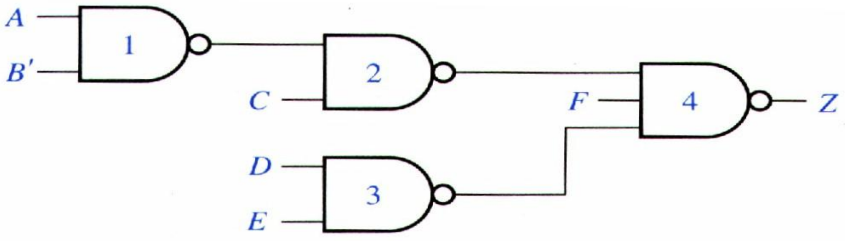


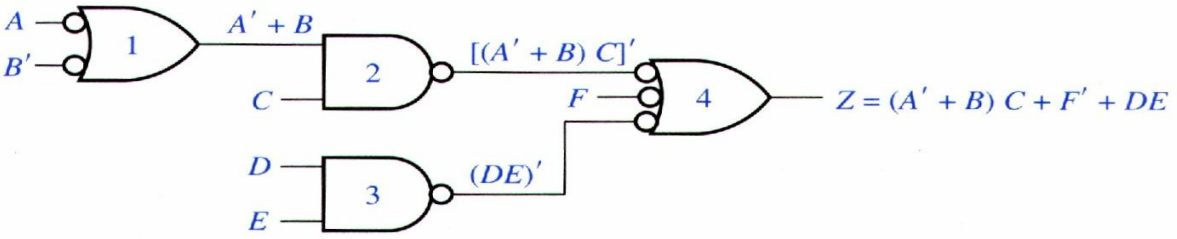
Figure 3.2.11 Summary of gate types and their DeMorgan equivalent symbols.

Alternative Network

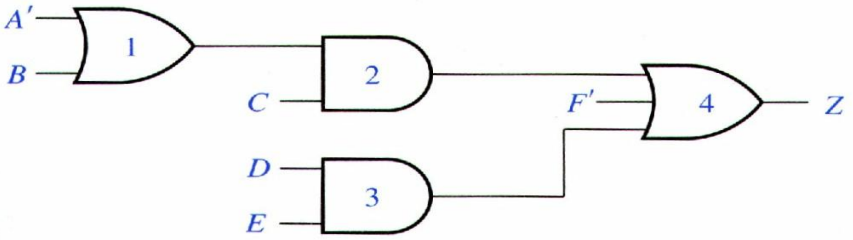
- Replace first and third NAND gates with alternative NAND gate symbol.



(a) NAND-gate network



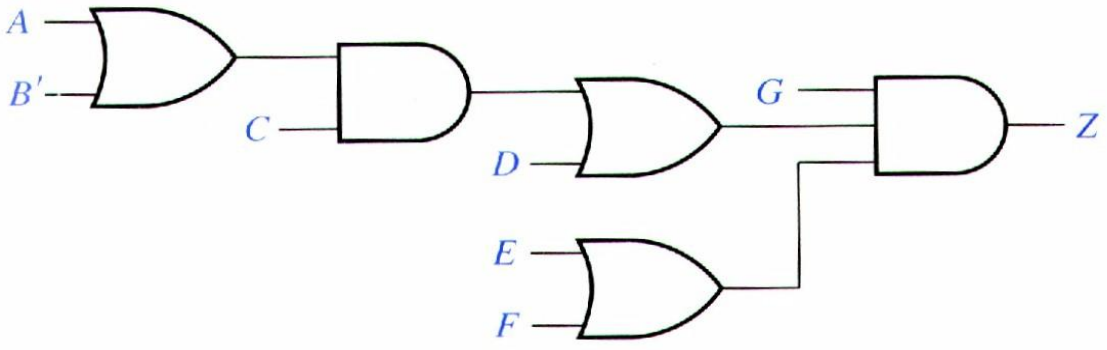
(b) Alternate form for NAND-gate network



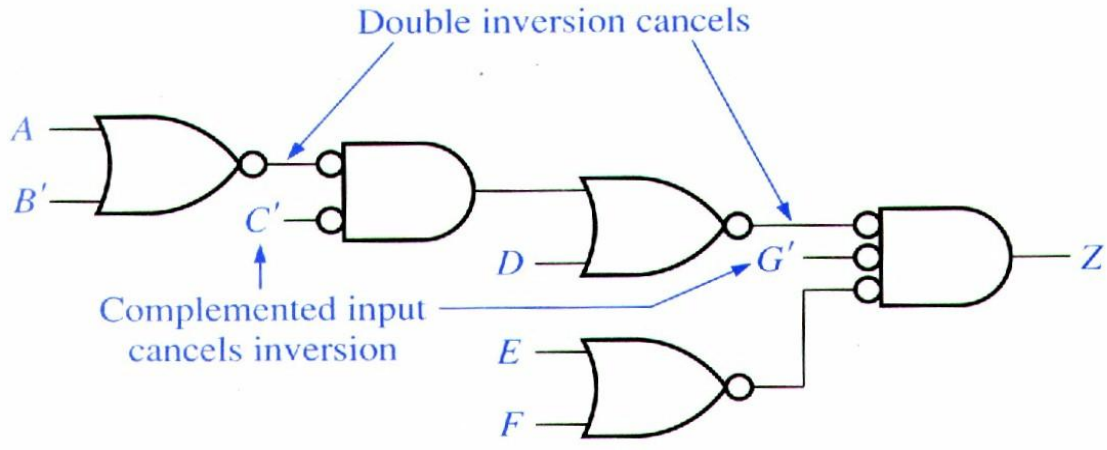
(c) Equivalent AND-OR network

More Example

- Multi-level POS network can be converted to NOR gate network.



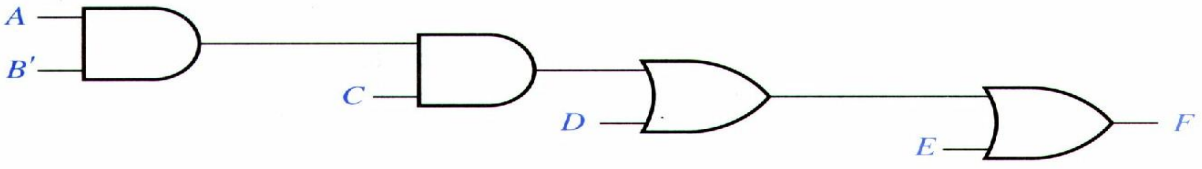
(a) AND-OR network



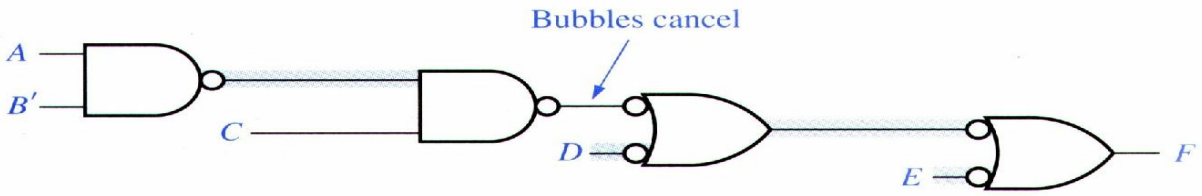
(b) Equivalent NOR-gate network

Non-Alternate AND-OR Network

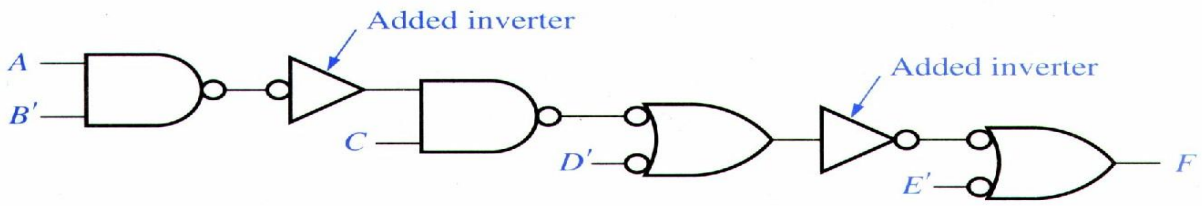
- Convert to NAND gates.
  - Convert all ANDs to NANDs.
  - Convert all ORs to alternative NANDs
  - Adjust by inserting inverter and complement single inverted input variables.



(a) AND-OR network



(b) First step in NAND conversion



(c) Completed conversion

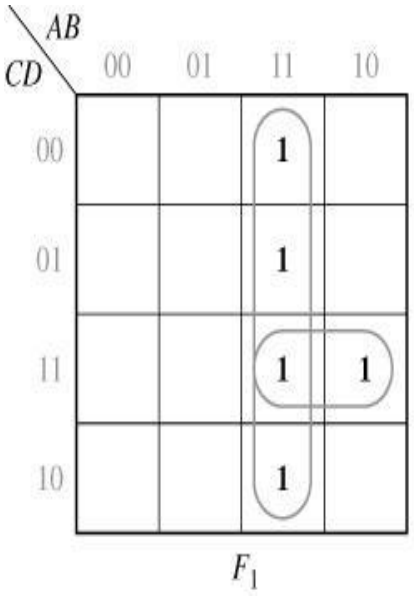
Multiple-Output Networks

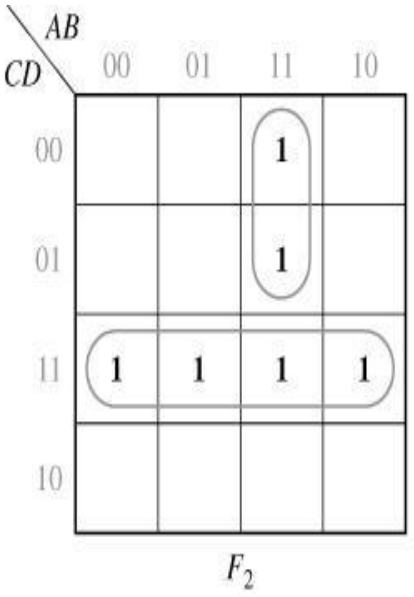
F1 = (A,B,C,D) = Σm(11,12,13,14,15)

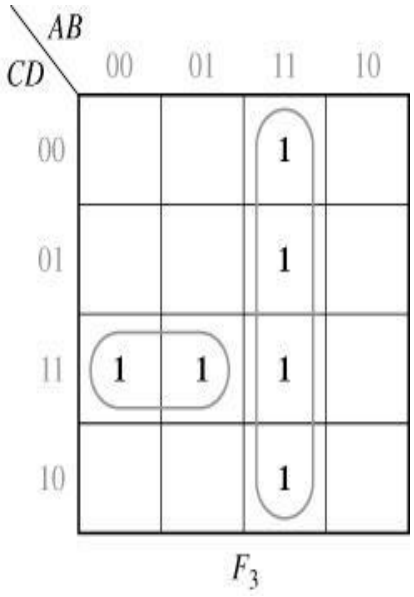
F2 = (A,B,C,D) = Σm(3,7,11,12,13,15)

F3 = (A,B,C,D) = Σm(3,7,12,13,14,15)

- Realized individually

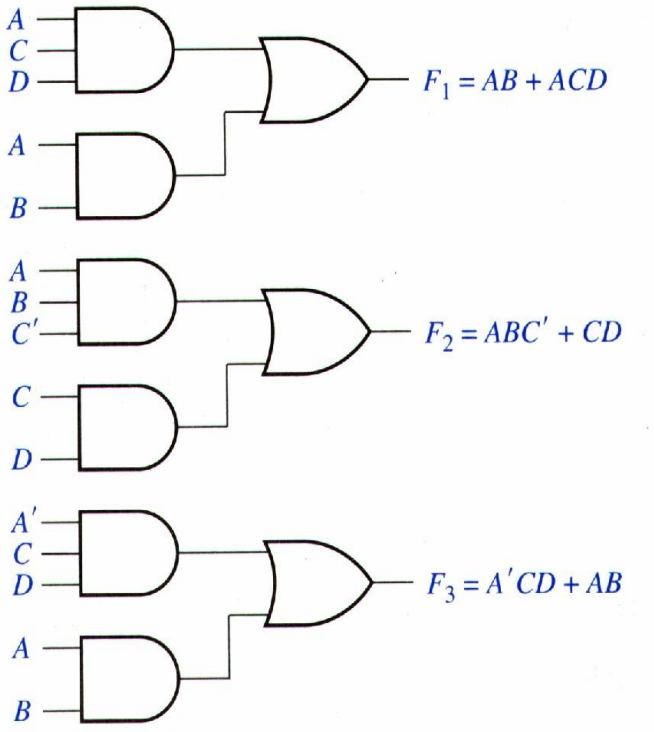


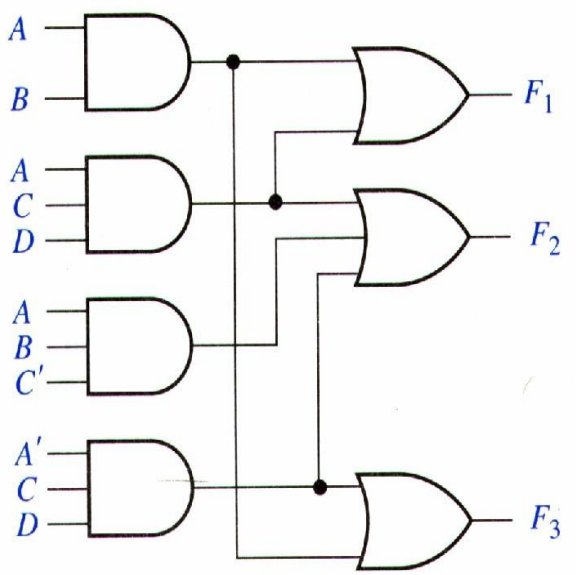




Realization Comparison

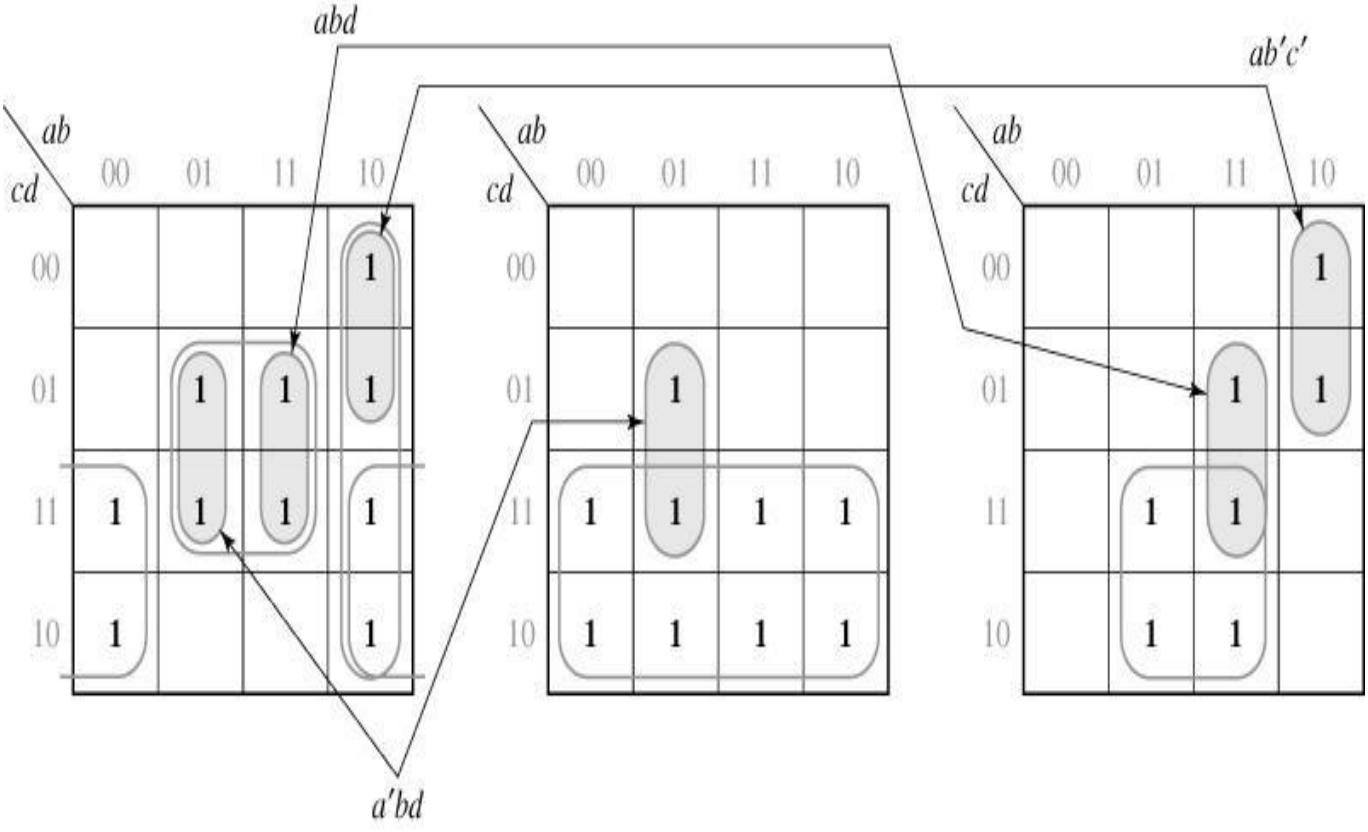
- Simplification
  - **(A,B) is used both in F1 and F3.**
  - **ACD for F1 and A'CD for F3**
  - **Replace CD in F2 by A'CD + ACD**
  - **Now F2 is realized by ABC' +A'CD + ACD.**





4 Inputs and 3 Outputs

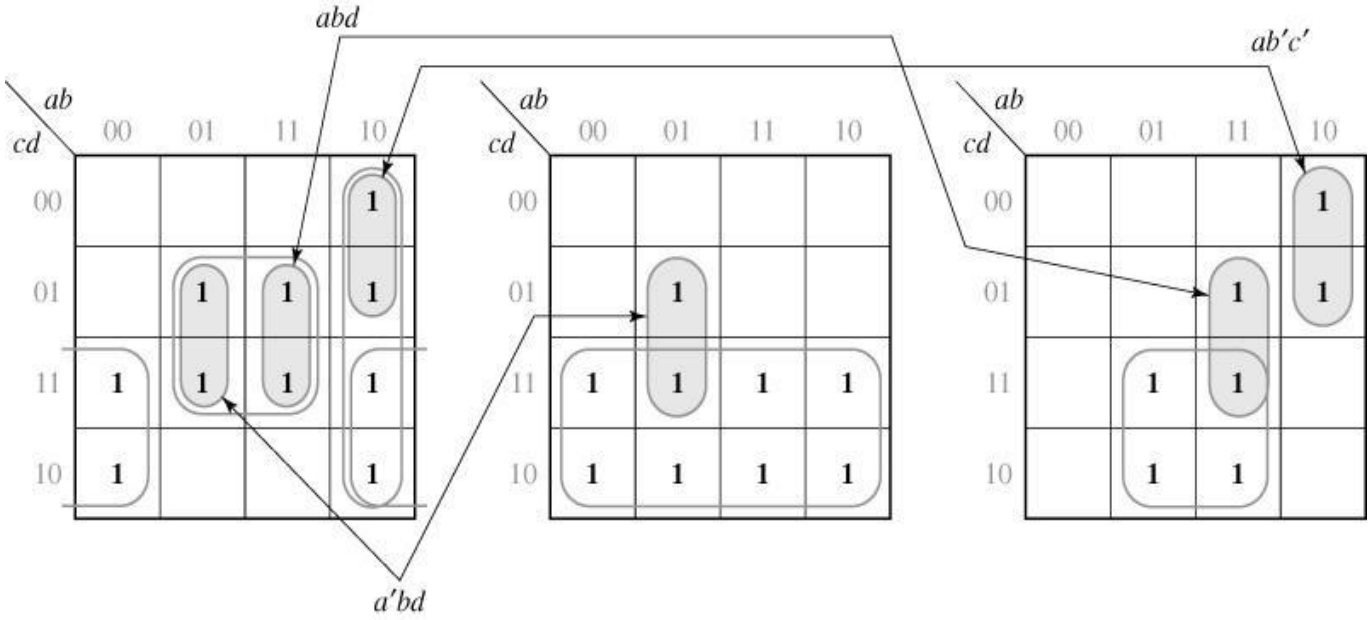
- $f_1 = \Sigma m(2,3,5,7,8,9,10,11,13,15)$
  - $f_1 = bd + b'c + ab'$
- $f_2 = \Sigma m(2,3,5,6,7,10,11,14,15)$
  - $f_2 = c + a'bd$
- $f_3 = \Sigma m(6,7,8,9,13,14,15)$
  - $f_3 = bc + ab'c' + abd$ (or $ac'd$)
  - (10 gates and 25 gate inputs, each function is minimized separately)



4 Inputs and 3 Outputs

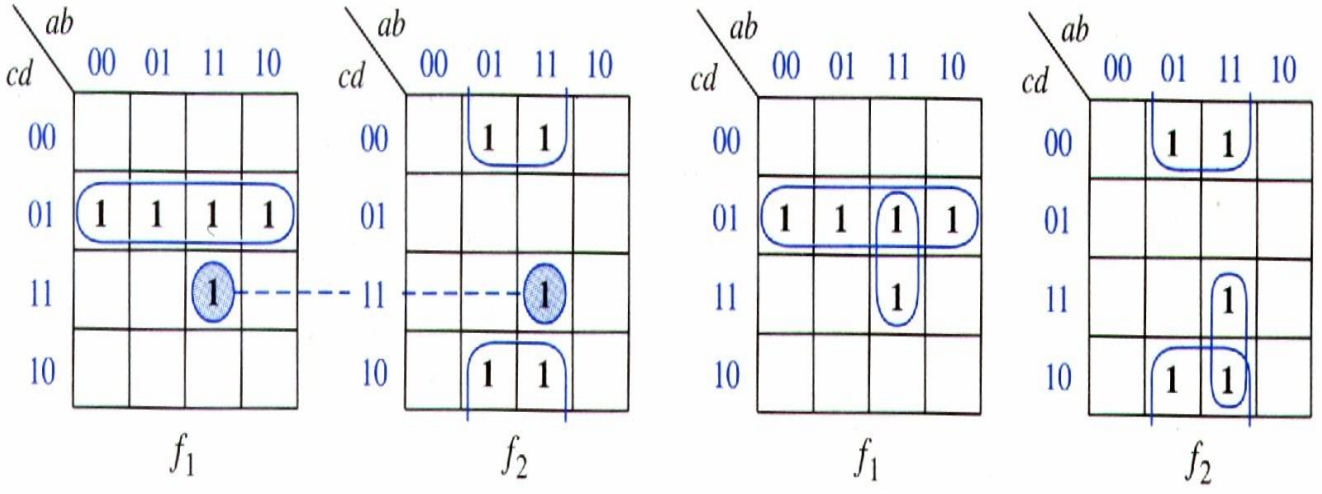
(cont.)

- $f_1 = bd + b'c + ab'$
  - $f_1 = a'bd + abd + ab'c' + b'c$
  - bd can be replaced with a'bd + abd.
- $f_2 = c + a'bd$
  - $f_2 = c + a'bd$
- $f_3 = bc + ab'c' + abd$ (or ac'd)
  - $f_3 = bc + ab'c' + abd$
- (8 gates and 22 gate inputs)
- **Use common terms to save gates.**



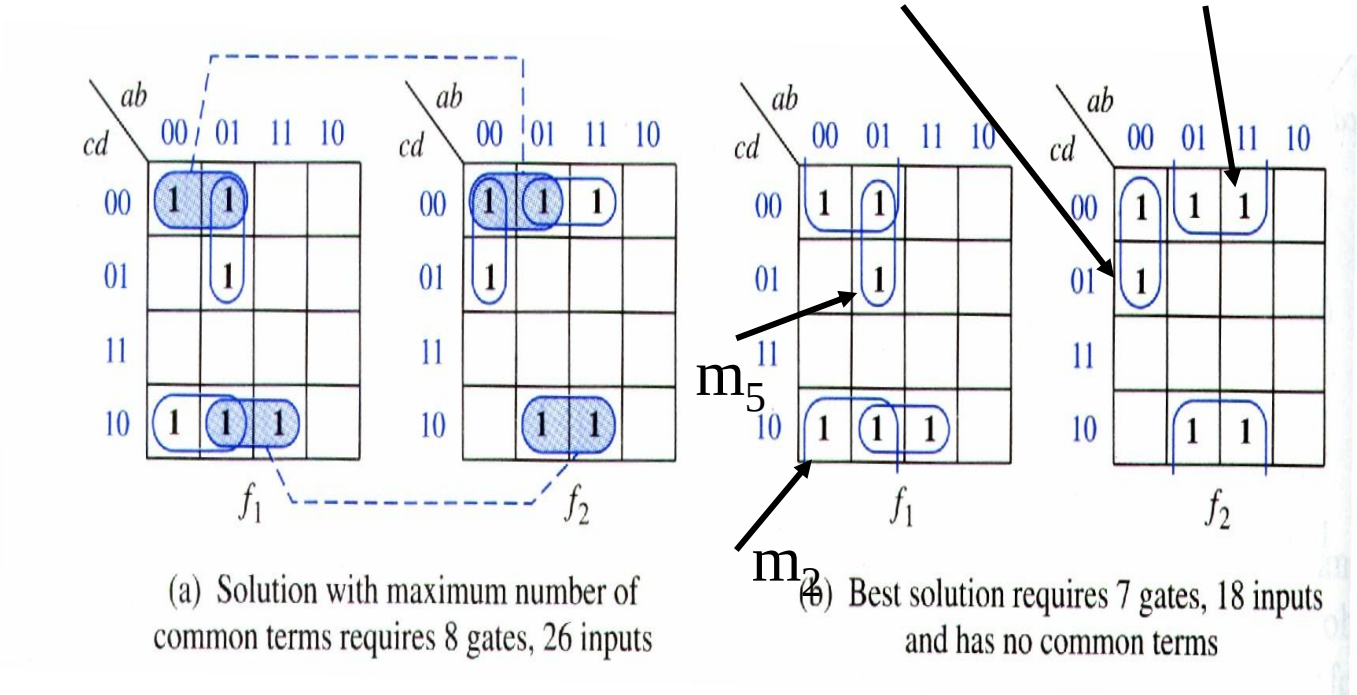
Multiple -Outputs

- (a) abcd in f_1 and f_2 uses the same gate.
- (b) Use more gate
- Why?
  - abd is not essential (because m_{15} also appears on f_2)
  - c'd is essential to f_1 and to the multiple-output realization.



Multiple -Outputs (cont.)

- (a) uses 8 gates (use as many as common terms)
- (b) uses 7 gates. Why?
  - In f_1, m_2 and m_5 does not appear on f_2. So m_2 (in f_1) to both f_1 and f_2 is unique, so a'd' is essential to f_1 in multiple-output. Similarly for m_5. On f_2, 1100 is unique. (Essential prime implicant first)



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

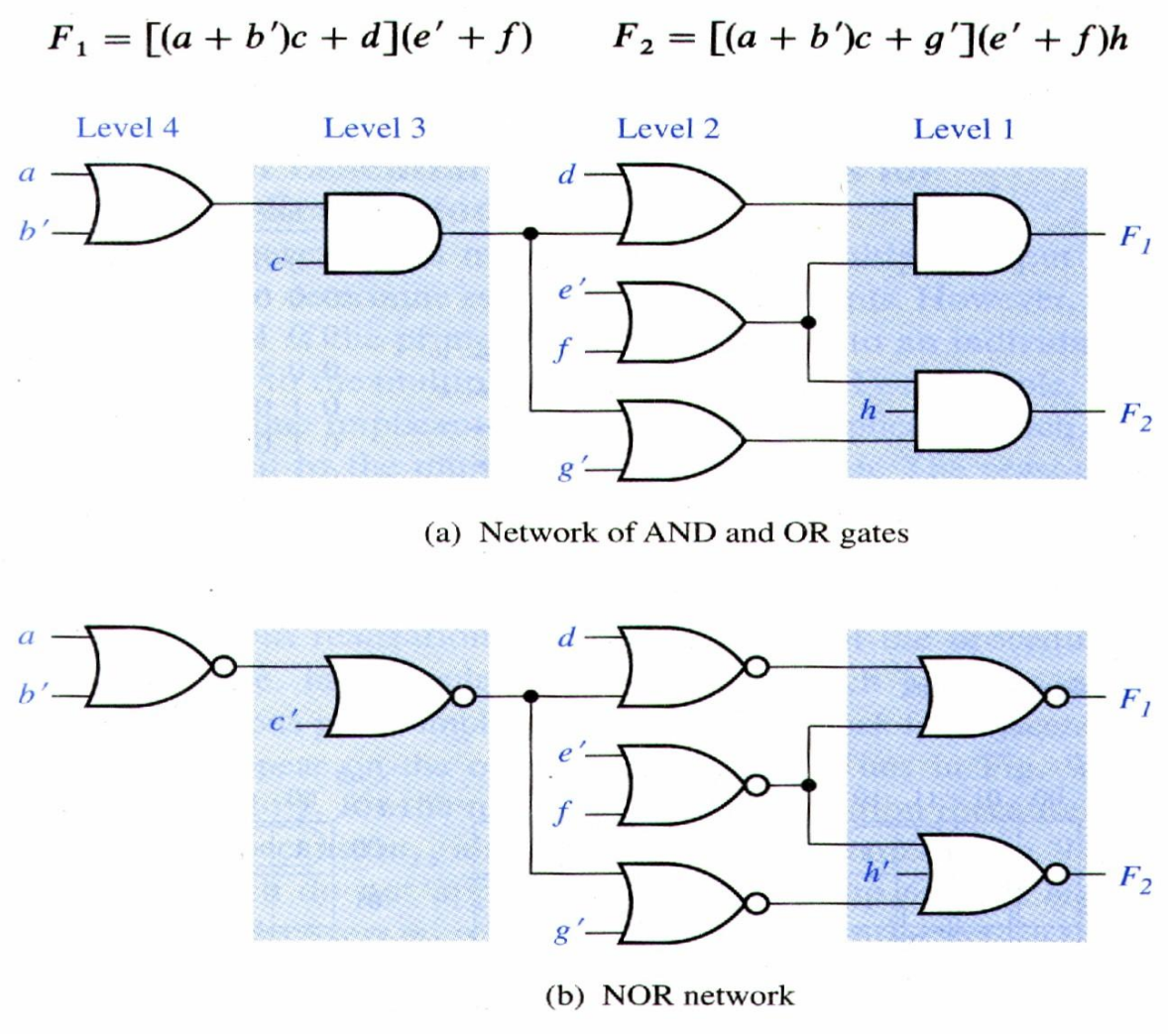
(b) Best solution requires 7 gates, 18 inputs and has no common terms

Multiple-Output Optimization

- Find the essential prime implicants for the multiple output.
- Checking for common terms to see if using them will help.

Multi-Output NAND and NOR Network

- OR-AND to NOR-NOR



Terminology

- Terms:
  - Fan-out: the maximum number of inputs to which an IC output can be connected without electrically loading down the output.
  - Fan in: the number of inputs to a gate.
  - PCB: printed circuit board.
  - HDL: hardware description language.
  - Synthesis: when a logic circuit is obtained from a functional description (equation or truth table). This process is called synthesis.