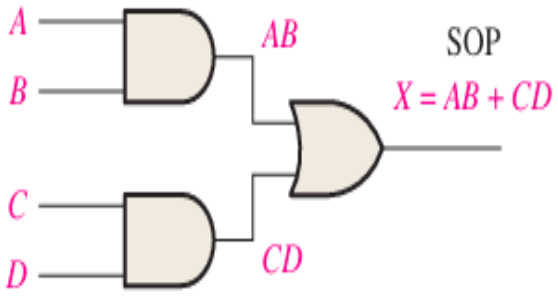
Combinational Logic Analysis

- When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is in the category of combinational logic.
- In combinational logic, the out put level is at all times dependent on the combination of input levels.

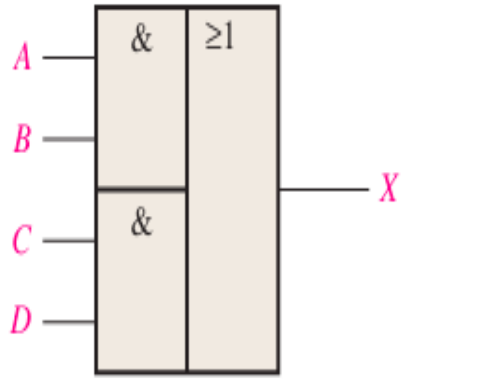
Basic Combinational Logic Circuits

AND-OR Logic

AND-OR logic produces an SOP expression.



(a) Logic diagram (ANSI standard distinctive shape symbols)



(b) ANSI standard rectangular outline symbol

For a 4-input AND-OR logic circuit, the output X is HIGH (1) if both input A and input B are HIGH (1) or both input C and input D are HIGH (1).

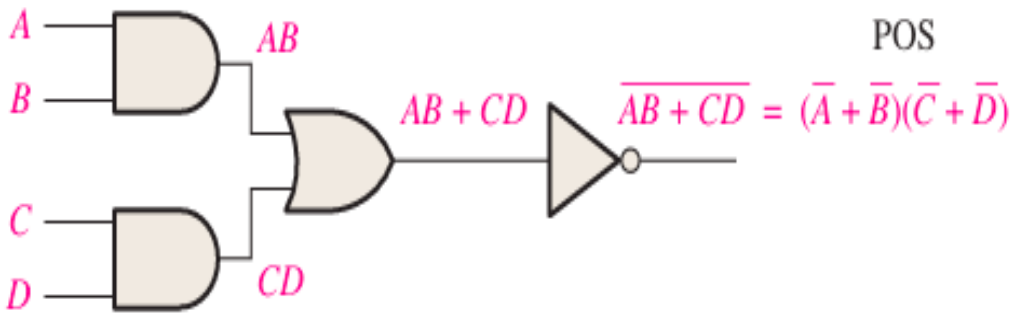
Truth table for the AND-OR logic

Inputs						Output
A	*B*	*C*	*D*	*AB*	*CD*	*X*
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

AND-OR-Invert Logic

- When the output of an AND-OR circuit is complemented (inverted), it results in an AND-OR Invert circuit.
- AND-OR logic directly implements SOP expressions. POS expressions can be implemented with AND-OR-Invert logic.

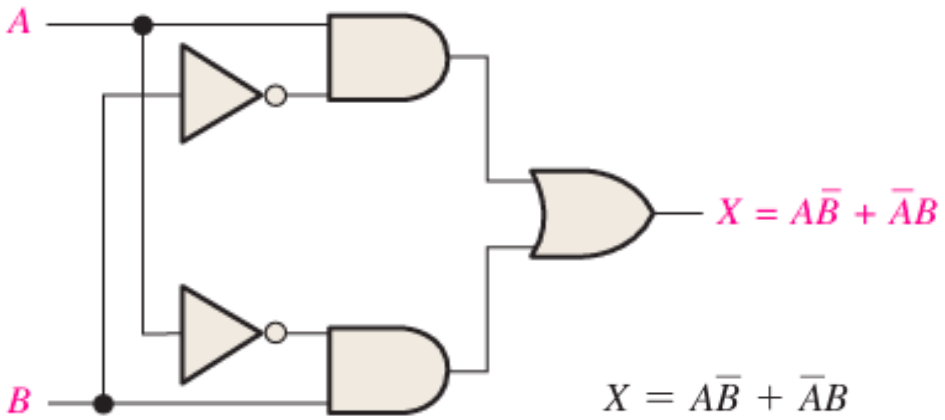
$$X = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) = (\overline{AB})(\overline{CD}) = \overline{\overline{(\overline{AB})(\overline{CD})}} = \overline{\overline{AB} + \overline{CD}} = \overline{AB + CD}$$



For a 4-input AND-OR-Invert logic circuit, the output X is LOW (0) if both input A and input B are HIGH (1) or both input C and input D are HIGH (1).

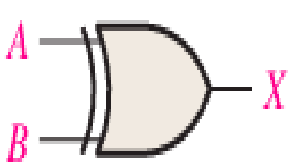
Exclusive-OR Logic

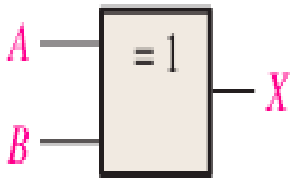
The XOR gate is actually a combination of other gates.



$$X = A\overline{B} + \overline{A}B$$

$$X = A \oplus B$$



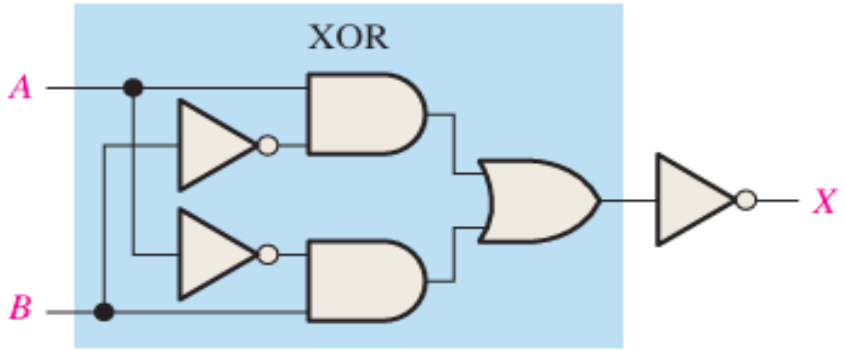


Truth table for an exclusive-OR.

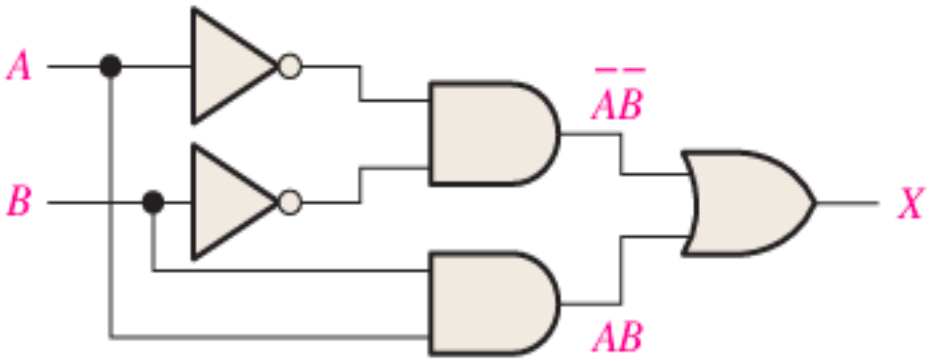
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-NOR Logic

$$X = \overline{A\overline{B} + \overline{A}B} = \overline{(A\overline{B})}\,\overline{(\overline{A}B)} = (\overline{A} + B)(A + \overline{B}) = \overline{A}\,\overline{B} + AB$$



(a) $X = \overline{A\overline{B} + \overline{A}B}$

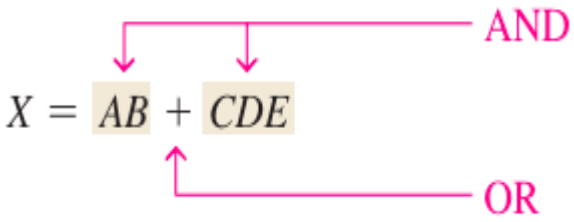


(b) $X = \overline{A}\,\overline{B} + AB$

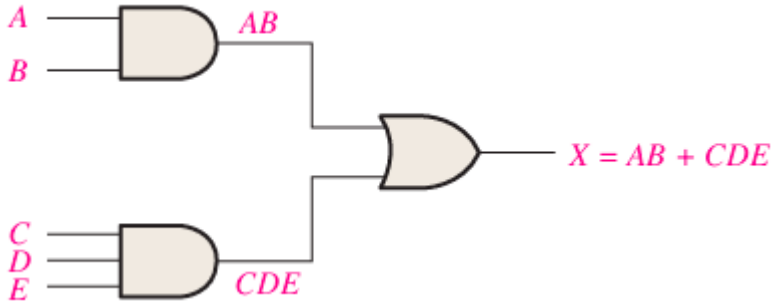
Implementing Combinational Logic

For every Boolean expression there is a logic circuit, and for every logic circuit there is a Boolean expression.

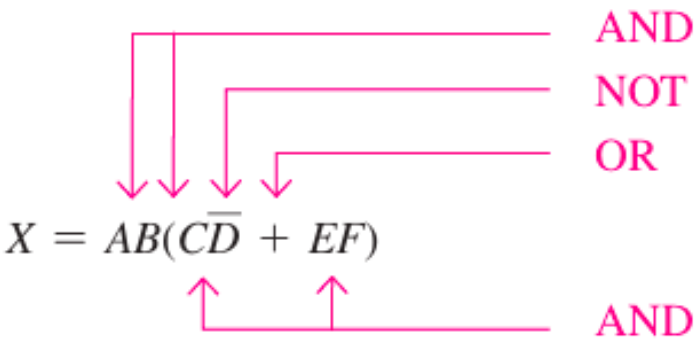
From a Boolean Expression to a Logic Circuit



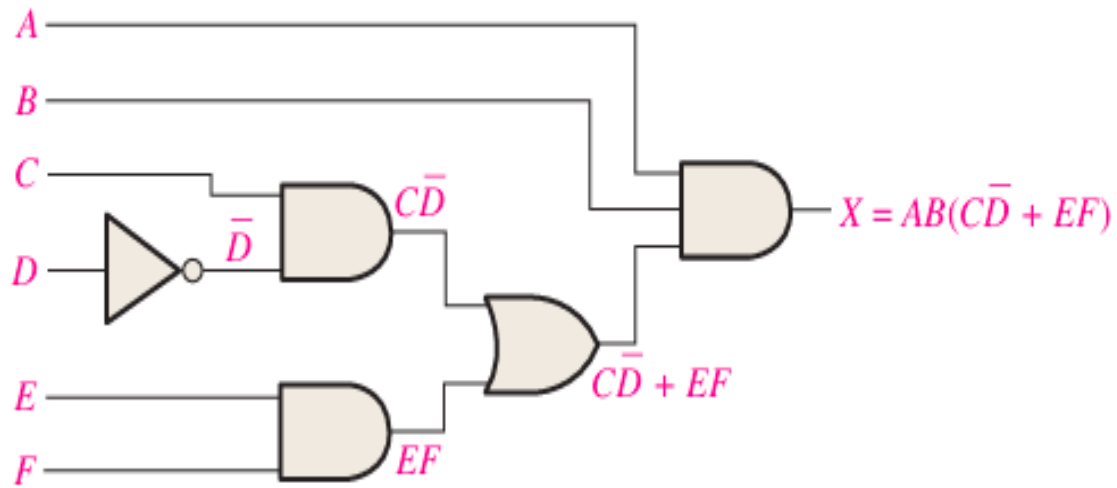
X = AB + CDE



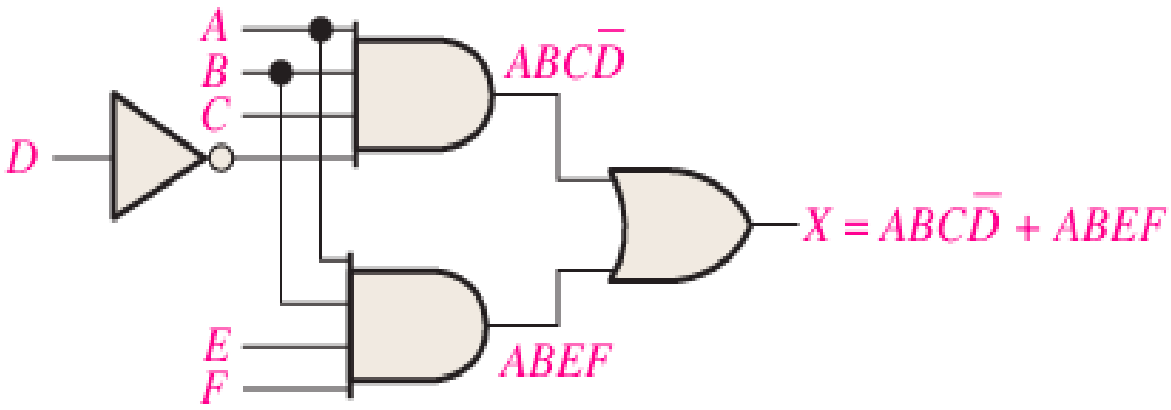
$$X = AB(C\overline{D} + EF)$$



1. One inverter to form $\overline{D}$
2. Two 2-input AND gates to form $C\overline{D}$ and EF
3. One 2-input OR gate to form $C\overline{D} + EF$
4. One 3-input AND gate to form X



(a)



(b) Sum-of-products implementation of the circuit in part (a)

The NAND Gate as a Universal Logic Element

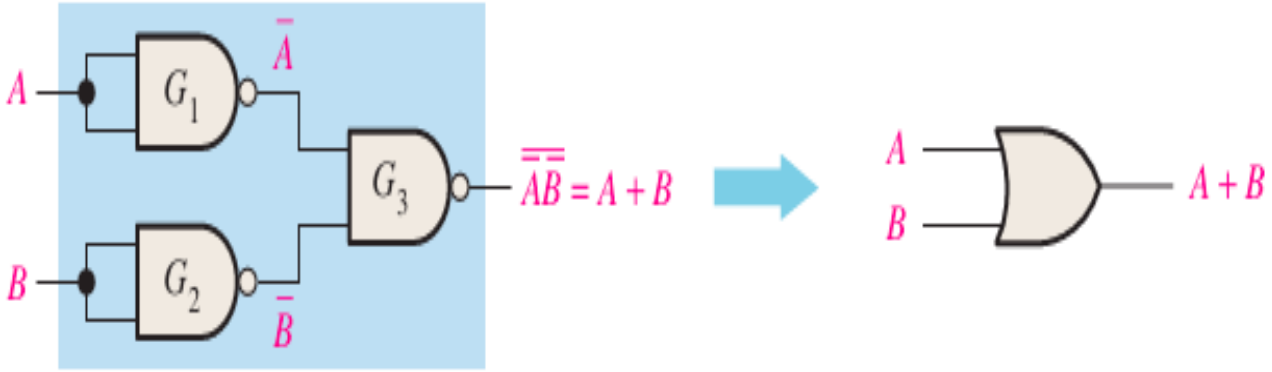
- Combinations of NAND gates can be used to produce any logic function.

(a) One NAND gate used as an inverter

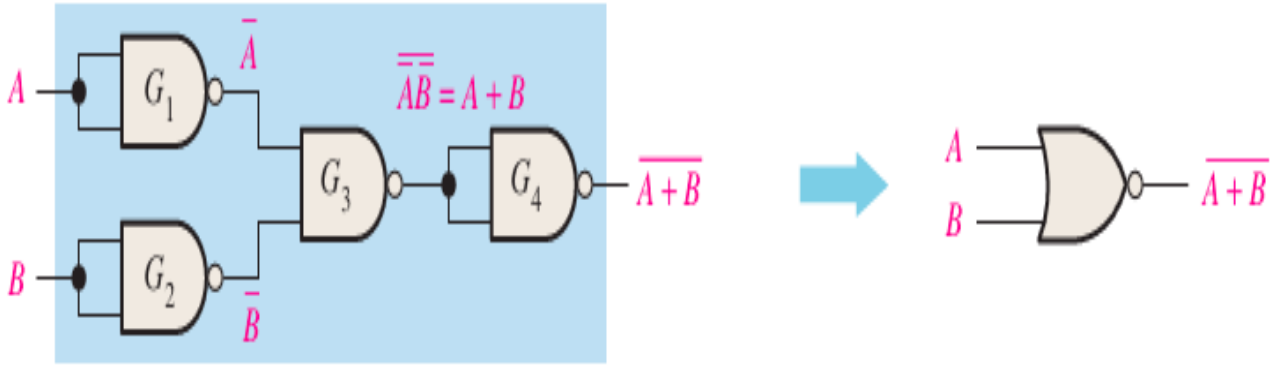
If A = B

(b) Two NAND gates used as an AND gate

(c) Three NAND gates used as an OR gate



(D) Four NAND gates used as a NOR gate



NAND gate is used to invert (complement) a NAND output to form the AND function, as indicated in the following equation:

$$X = \overline{\overline{AB}} = AB$$

NAND gates G1 and G2 are used to invert the two input variables before they are applied to NAND gate G3. The final OR output is derived as follows by application of De Morgan's theorem:

$$X = \overline{\overline{A}\,\overline{B}} = A + B$$

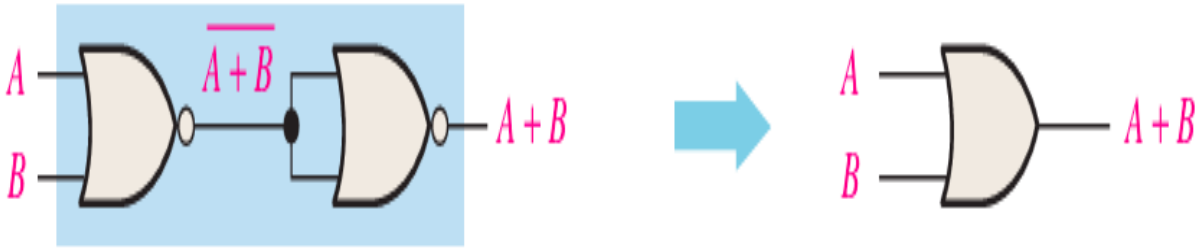
NAND gate G4 is used as an inverter connected to the circuit of part (c) to produce the NOR operation.

$$\overline{A + B}.$$

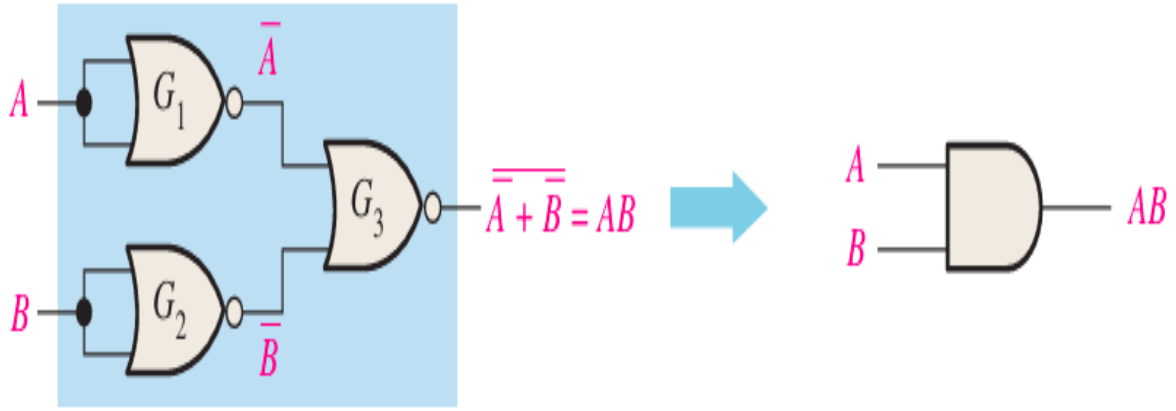
The NOR Gate as a Universal Logic Element

(a) One NOR gate used as an inverter

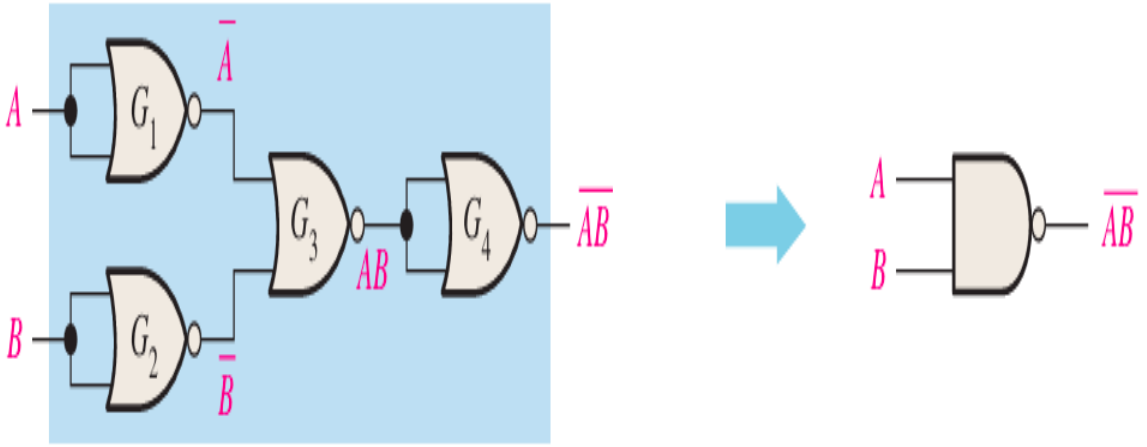
(b) Two NOR gates used as an OR gate



(c) Three NOR gates used as an AND gate



(d) Four NOR gates used as a NAND gate



Combinational Logic Using NAND and NOR Gates

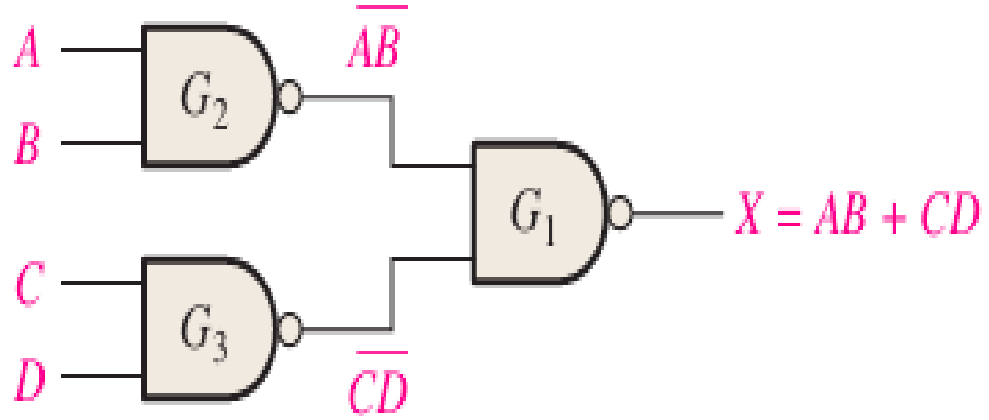
NAND Logic

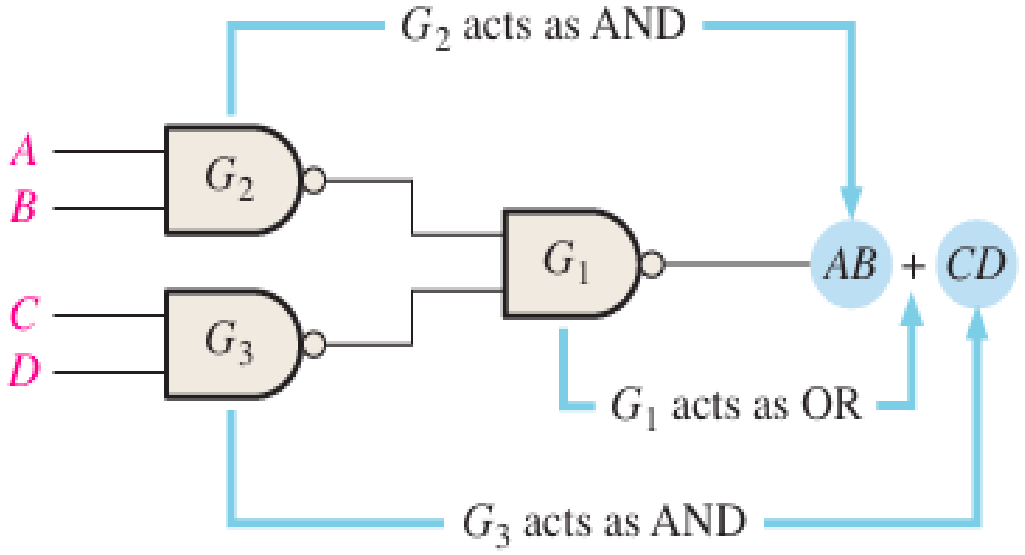
NAND gate can function as either a NAND or a negative-OR because, by De Morgan's theorem,

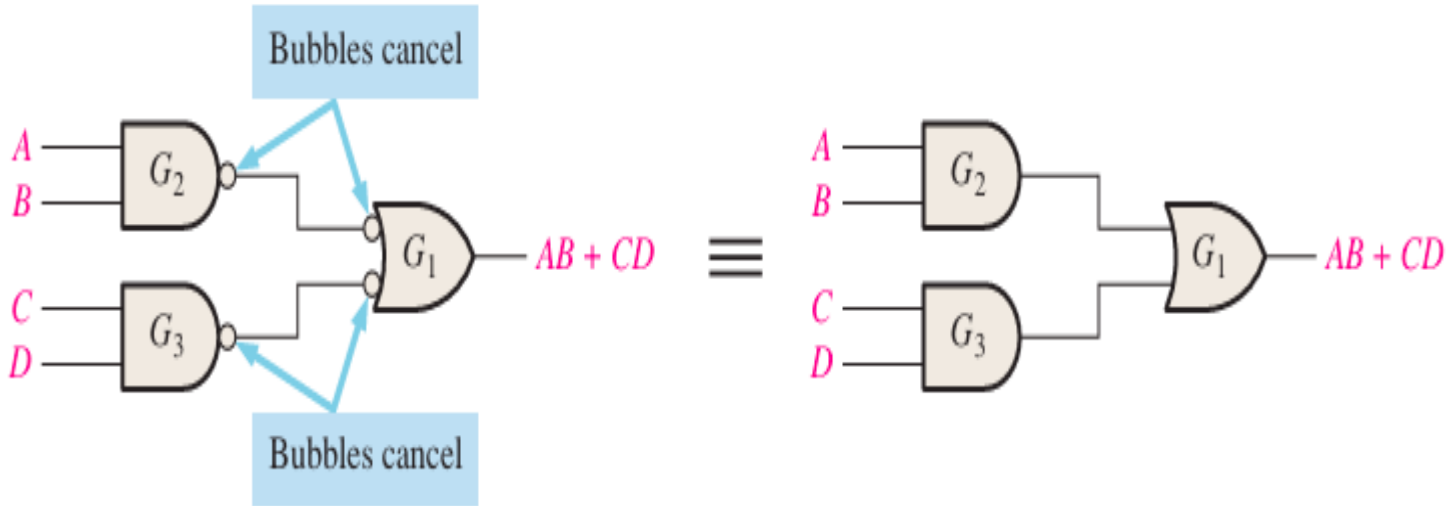
$$\overline{AB} = \overline{A} + \overline{B}$$

NAND — negative-OR

$$\begin{aligned} X &= \overline{(\overline{AB})(\overline{CD})} \\ &= \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{D})} \\ &= \overline{(\overline{A} + \overline{B})} + \overline{(\overline{C} + \overline{D})} \\ &= \overline{\overline{A}}\,\overline{\overline{B}} + \overline{\overline{C}}\,\overline{\overline{D}} \\ &= AB + CD \end{aligned}$$







(b) Equivalent NAND/Negative-OR logic diagram

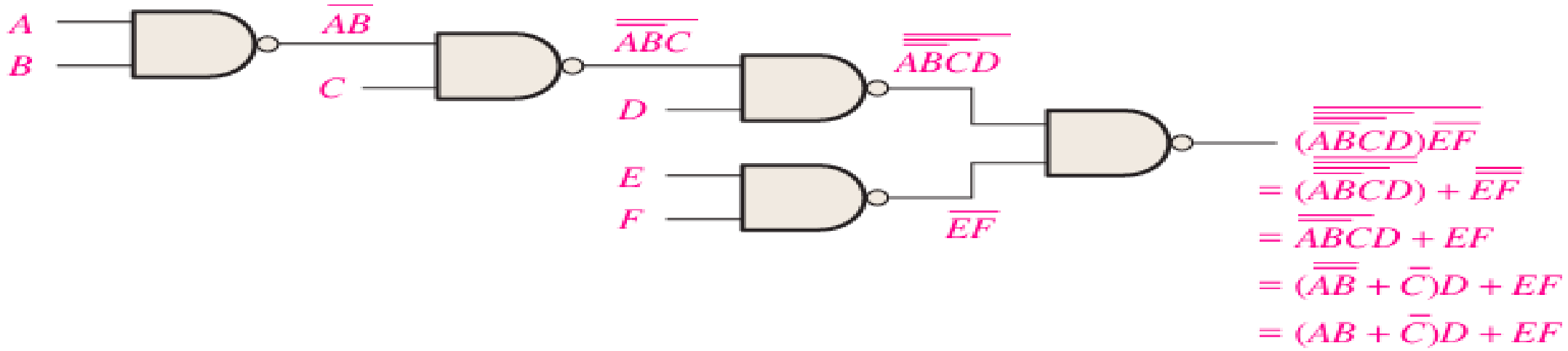
(c) AND-OR equivalent

NAND Logic Diagrams Using Dual Symbols

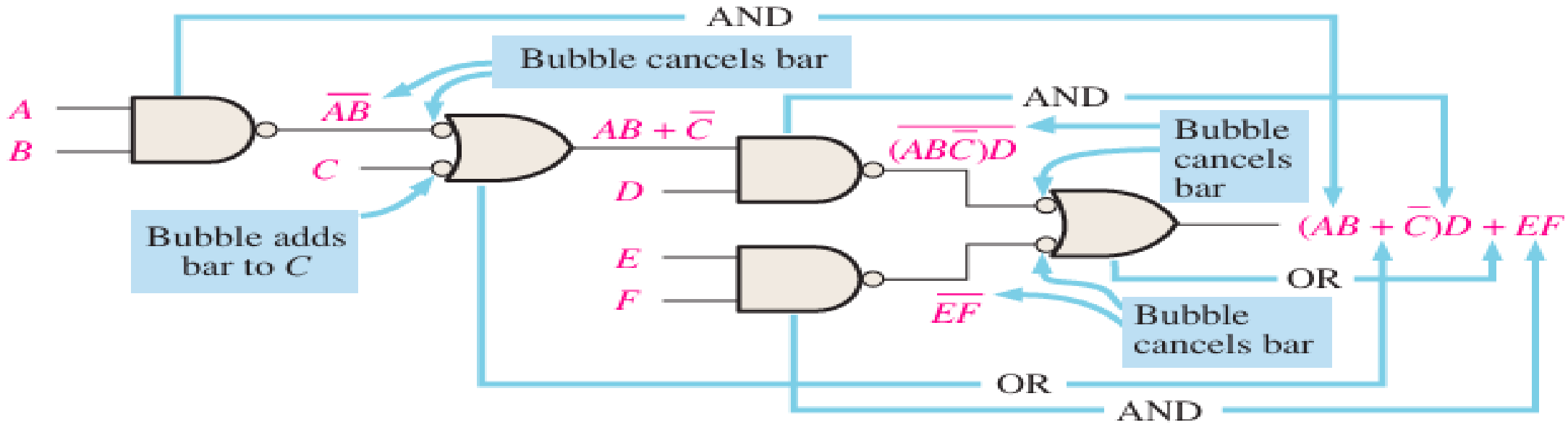
All logic diagrams using NAND gates should be drawn with each gate represented by either a NAND symbol or the equivalent negative-OR symbol to reflect the operation of the gate within the logic circuit.

The NAND symbol and the negative-OR symbol are called dual symbols.

When drawing a NAND logic diagram, always use the gate symbols in such a way that every connection between a gate output and a gate input is either bubble-to bubble or non bubble-to-non bubble. In general, a bubble output should not be connected to a non bubble input or vice versa in a logic diagram.

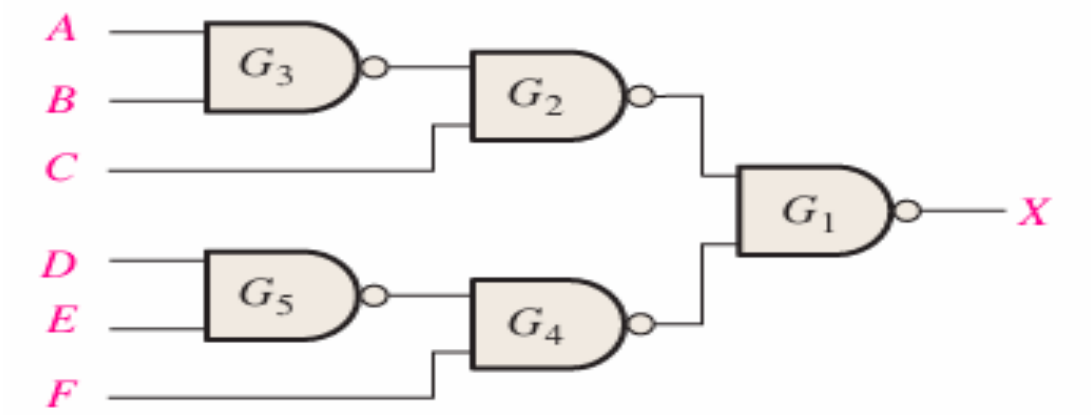


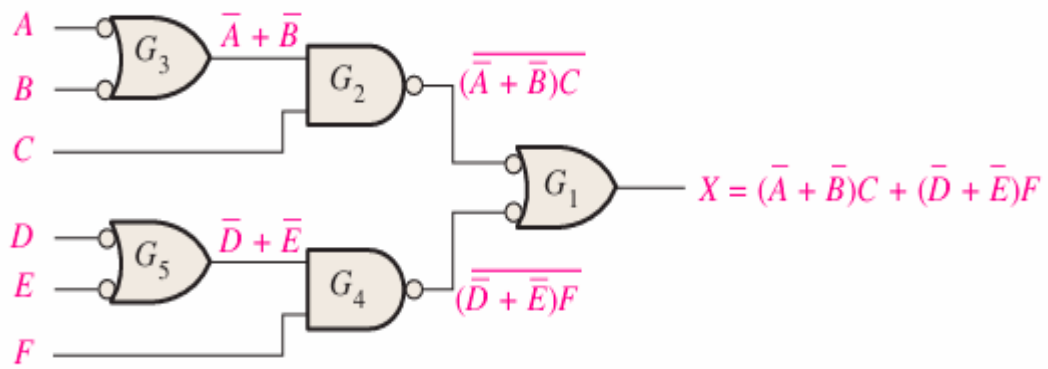
(a) Several Boolean steps are required to arrive at final output expression.



(b) Output expression can be obtained directly from the function of each gate symbol in the diagram.

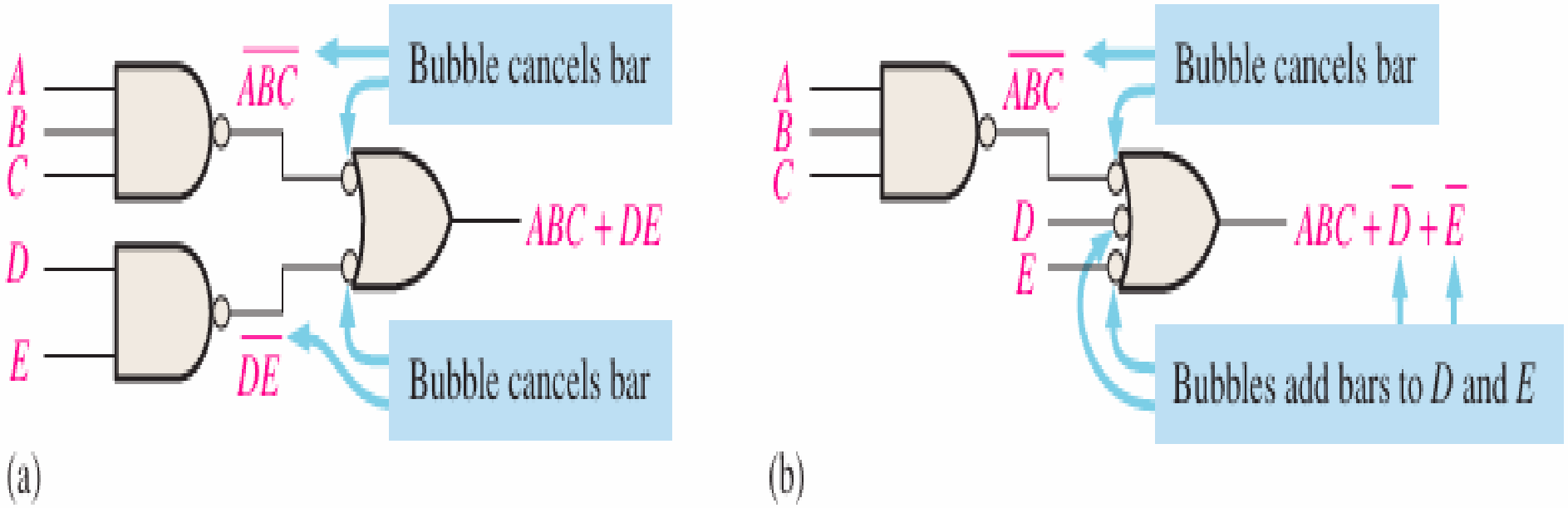
Redraw the logic diagram and develop the output expression for the circuit using the appropriate dual symbols.





Implement each expression with NAND logic using appropriate dual symbols:

(a) $ABC + DE$ (b) $ABC + \overline{D} + \overline{E}$

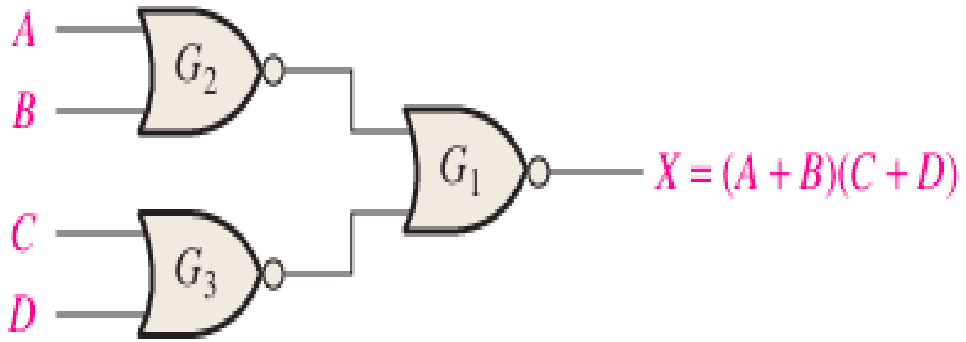


NOR Logic

A NOR gate can function as either a NOR or a negative-AND, as shown by De Morgan's theorem.

$$\overline{A + B} = \overline{A}\,\overline{B}$$

NOR — negative-AND



$$X = \overline{\overline{A + B} + \overline{C + D}} = (\overline{\overline{A + B}})(\overline{\overline{C + D}}) = (A + B)C + D)$$