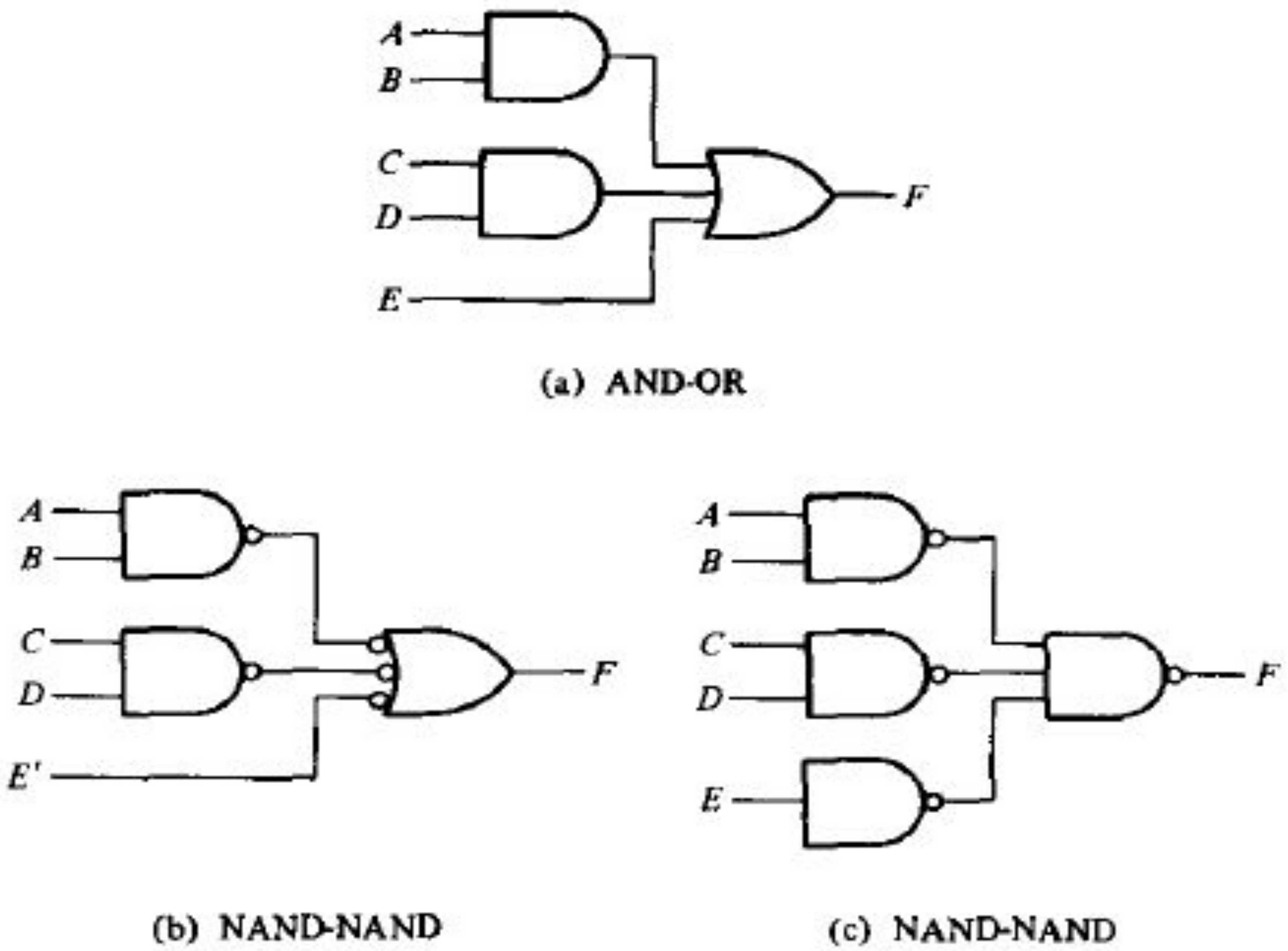


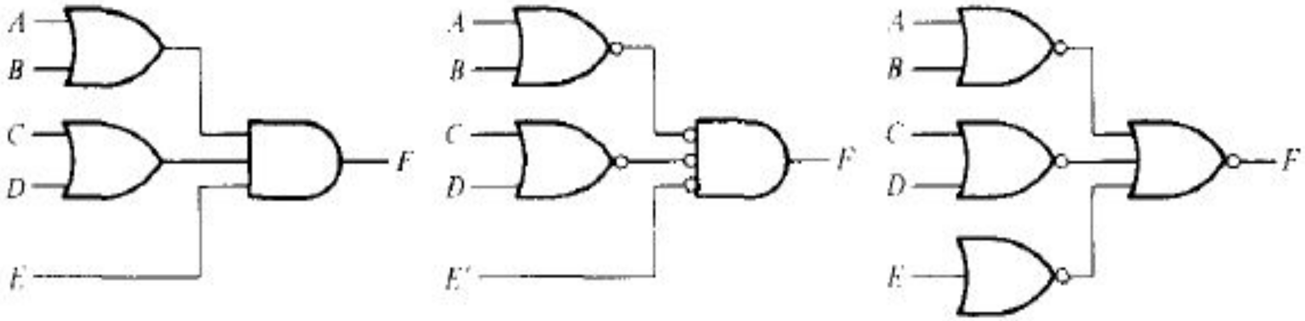
(a) AND-OR

(b) NAND-NAND

(c) NAND-NAND

FIGURE 3-18

Three ways to implement $F = AB + CD + E$



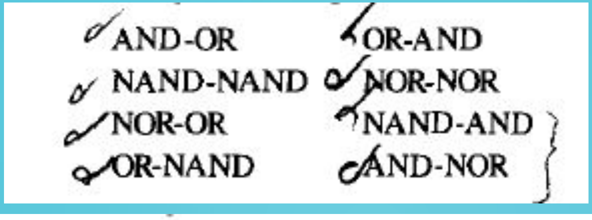
(a) OR-AND

(b) NOR-NOR

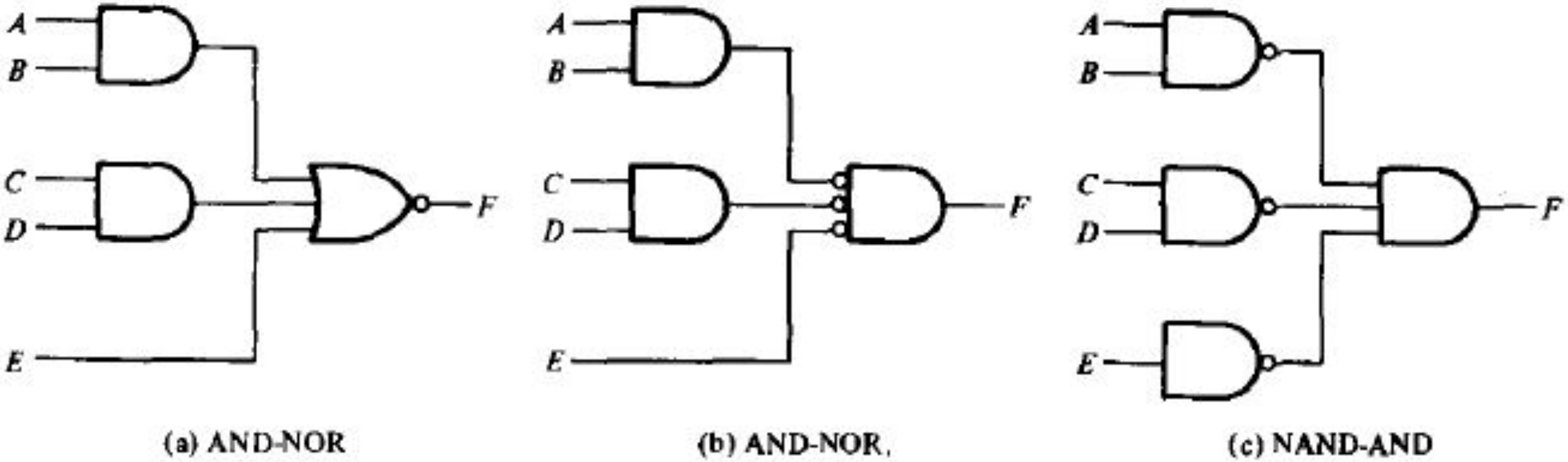
(c) NOR-NOR

FIGURE 3-20

Three ways to implement $F = (A + B)(C + D)E$



- ✓ AND-OR
- ✓ NAND-NAND
- ✓ NOR-OR
- ✓ OR-NAND
- ✓ OR-AND
- ✓ NOR-NOR
- ✓ NAND-AND
- ✓ AND-NOR

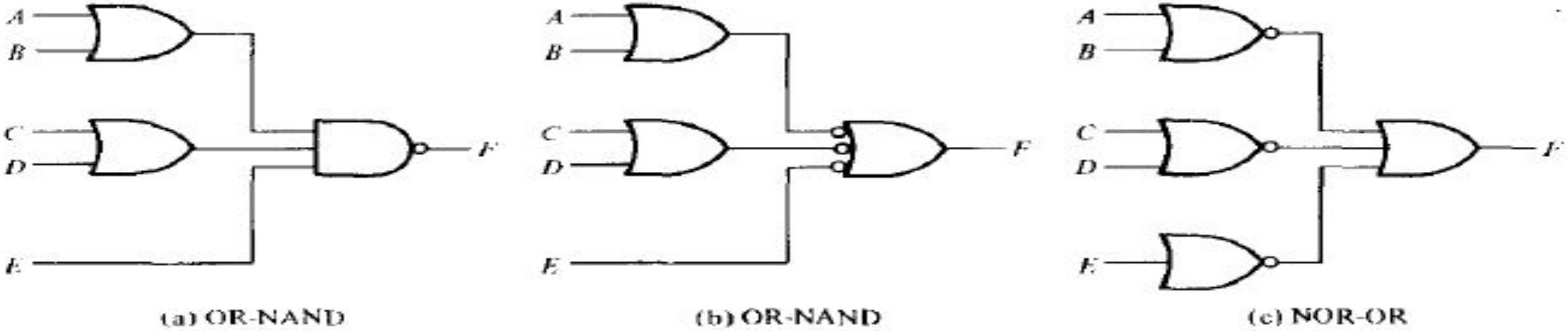


(a) AND-NOR

(b) AND-NOR.

(c) NAND-AND

FIGURE 3-23
AND-OR-INVERT circuits; $F = (AB + CD + E)'$

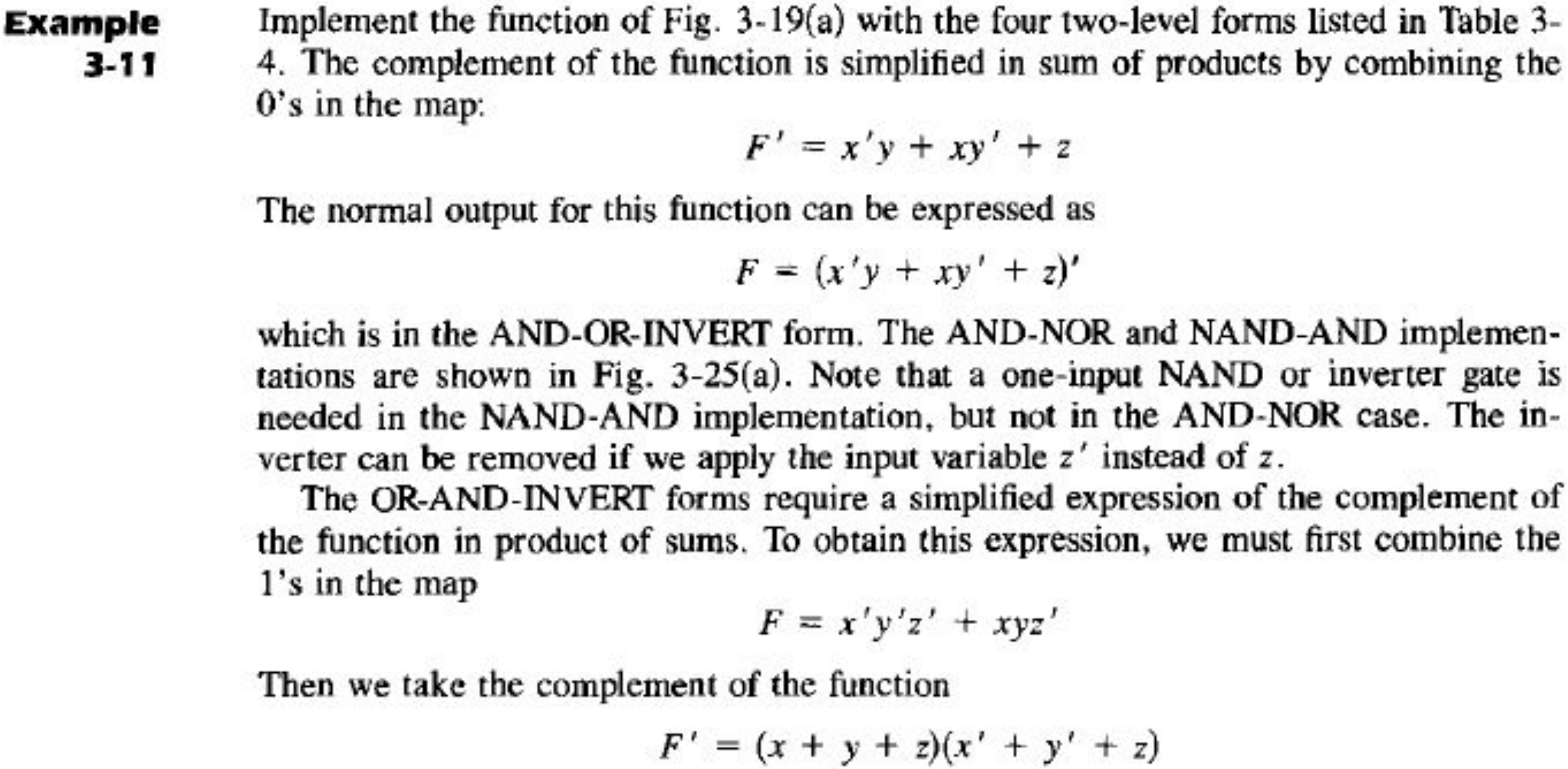


(a) OR-NAND

(b) OR-NAND

(c) NOR-OR

FIGURE 3-24
OR-AND-INVERT circuits; $F = [(A + B)(C + D)E]'$



Example 3-11

Implement the function of Fig. 3-19(a) with the four two-level forms listed in Table 3-4. The complement of the function is simplified in sum of products by combining the 0's in the map:

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

The normal output for this function can be expressed as

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

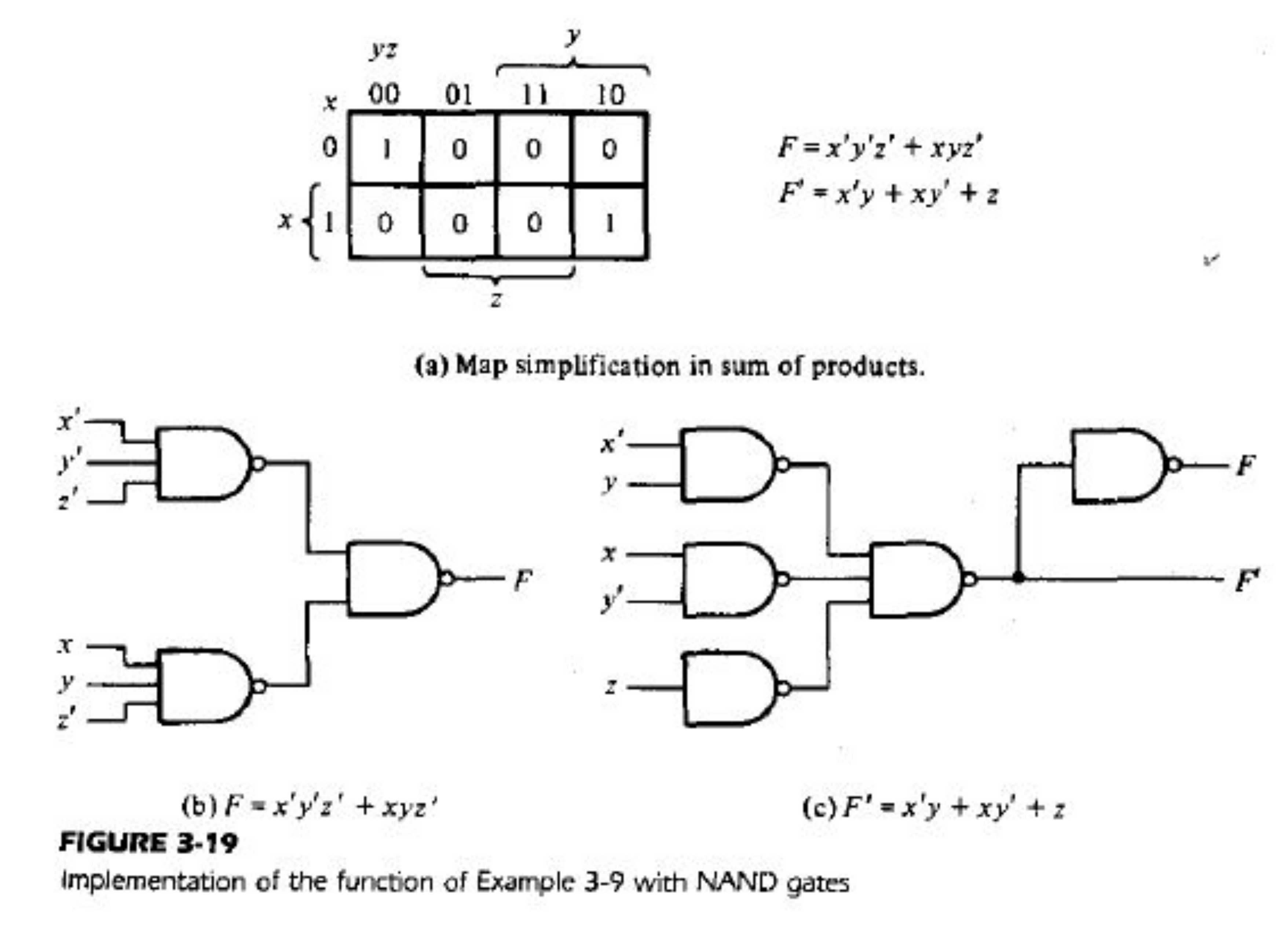
which is in the AND-OR-INVERT form. The AND-NOR and NAND-AND implementations are shown in Fig. 3-25(a). Note that a one-input NAND or inverter gate is needed in the NAND-AND implementation, but not in the AND-NOR case. The inverter can be removed if we apply the input variable z' instead of z.

The OR-AND-INVERT forms require a simplified expression of the complement of the function in product of sums. To obtain this expression, we must first combine the 1's in the map

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

Then we take the complement of the function

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



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

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

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

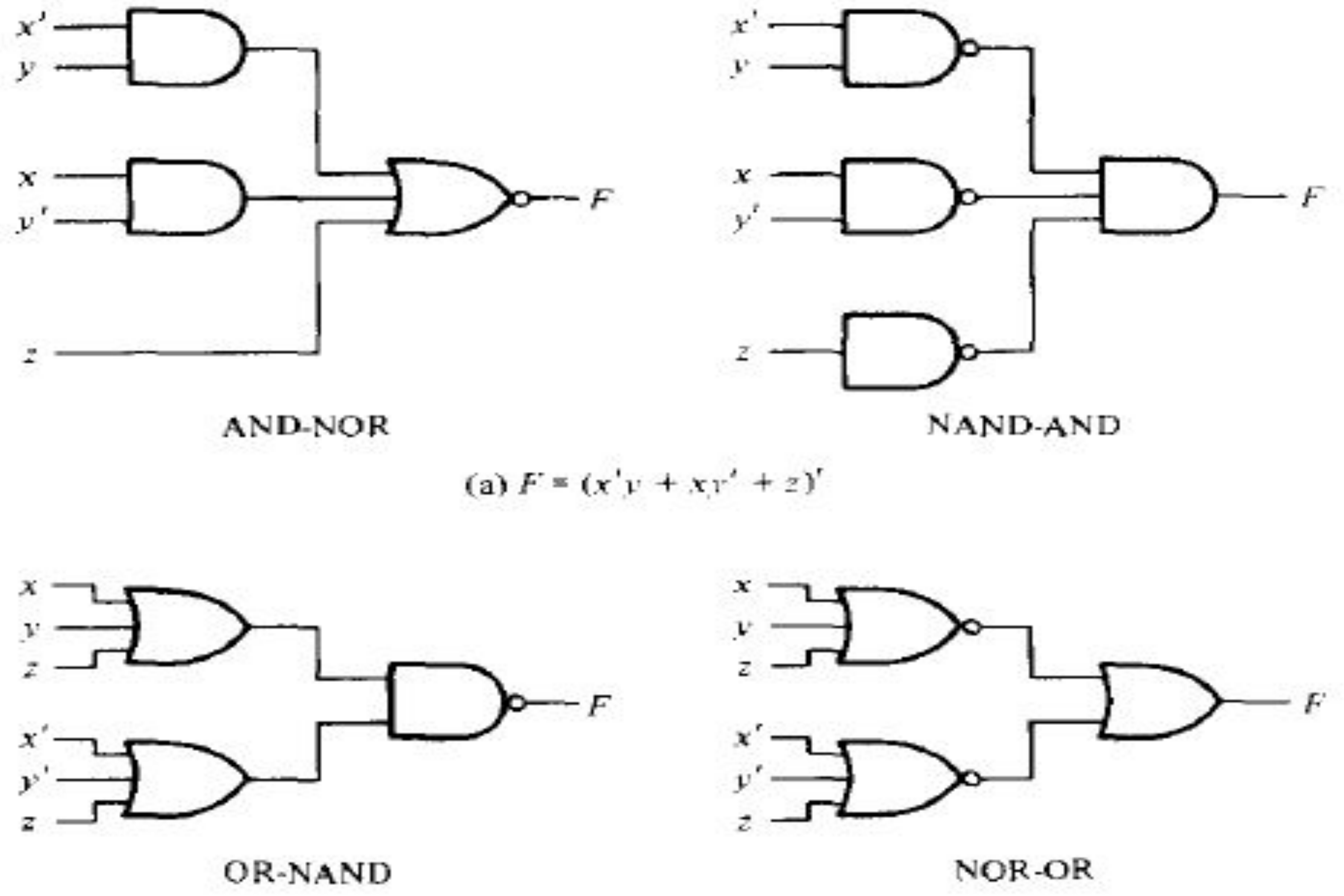
(a) Map simplification in sum of products.

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

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

FIGURE 3-19

Implementation of the function of Example 3-9 with NAND gates



AND-NOR

NAND-AND

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

OR-NAND

NOR-OR

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

FIGURE 3-25
Other two-level implementations

The normal output F can now be expressed in the form

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

which is in the OR-AND-INVERT form. From this expression, we can implement the function in the OR-NAND and NOR-OR forms, as shown in Fig. 3-25(b). ■

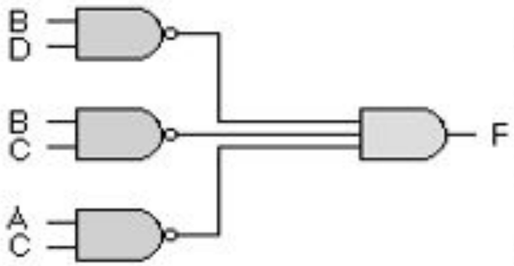
Implement the function *F* with the following two-level forms: NAND-AND, AND-NOR, OR-NAND, and NOR-OR.

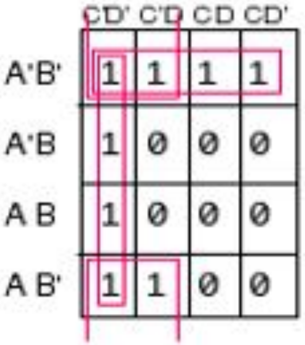
F(A,B,C,D) = Σ(O, I. 2, 3, 4, 8, 9,12)   (NAND – AND == AND –NOR)

(NOR-OR == OR-NAND)

F(A,B,C,D) = Σ(0,1,2,3,4,8,9,12)
F = A'B' + C'D' + B'C'
F'= BD + BC + AC

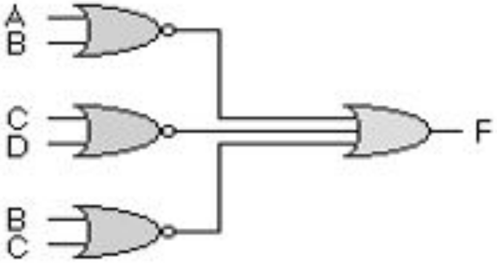
NAND-AND

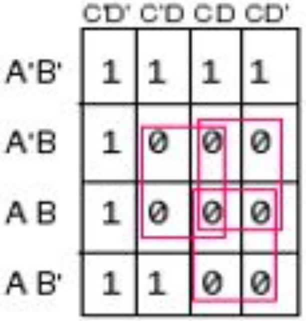




	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B	1	0	0	0
A B	1	0	0	0
A B'	1	1	0	0

NOR-OR

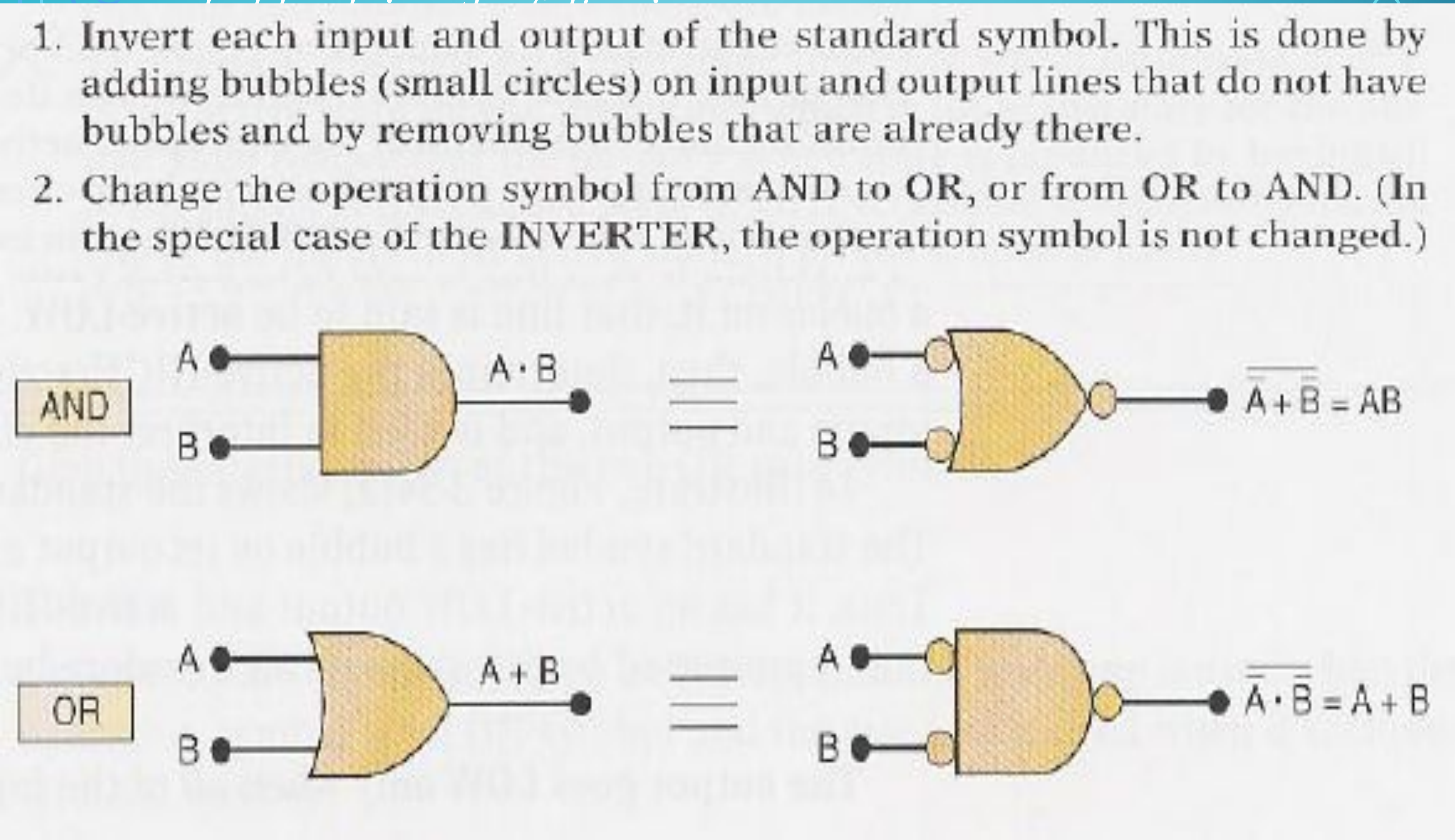




	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B	1	0	0	0
A B	1	0	0	0
A B'	1	1	0	0

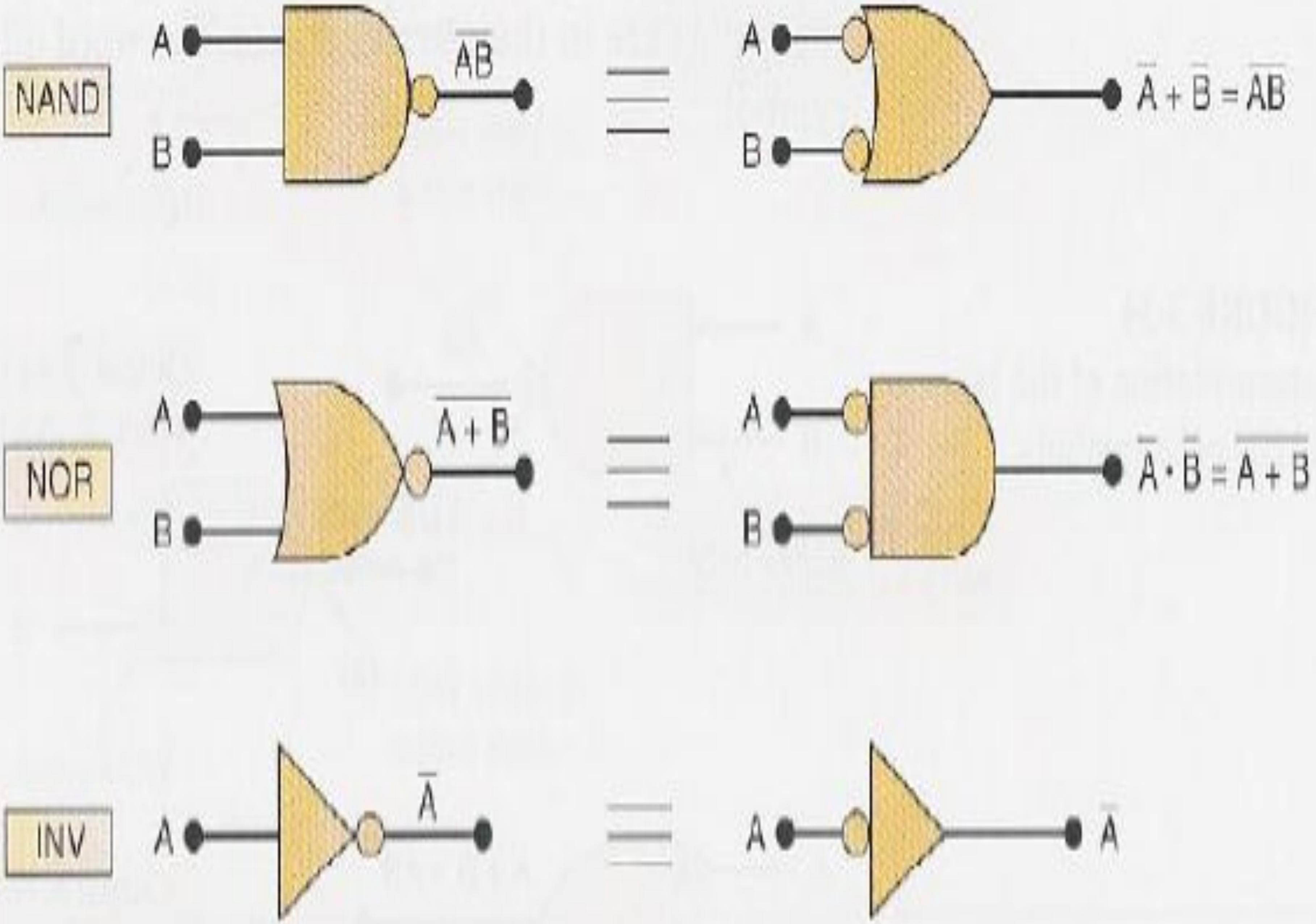
ALTERNATE LOGIC GATE REPRESENTATIONS

The alternate symbol for each gate is obtained from the standard symbol by doing the following:

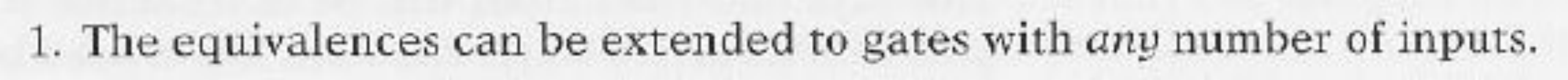


1. Invert each input and output of the standard symbol. This is done by adding bubbles (small circles) on input and output lines that do not have bubbles and by removing bubbles that are already there.
2. Change the operation symbol from AND to OR, or from OR to AND. (In the special case of the INVERTER, the operation symbol is not changed.)

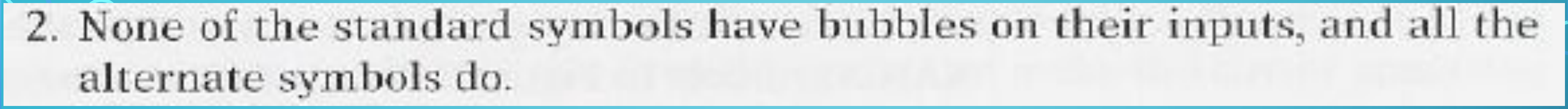
AND
A
B
A · B
≡
A
B
$\overline{\overline{A}+\overline{B}} = AB$
OR
A
B
A + B
≡
A
B
$\overline{\overline{A}\cdot\overline{B}} = A + B$


NAND
A
B
$\overline{AB}$
≡
A
B
$\overline{A} + \overline{B} = \overline{AB}$
NOR
A
B
$\overline{A + B}$
≡
A
B
$\overline{A} \cdot \overline{B} = \overline{A + B}$
INV
A
$\overline{A}$
≡
A
$\overline{A}$

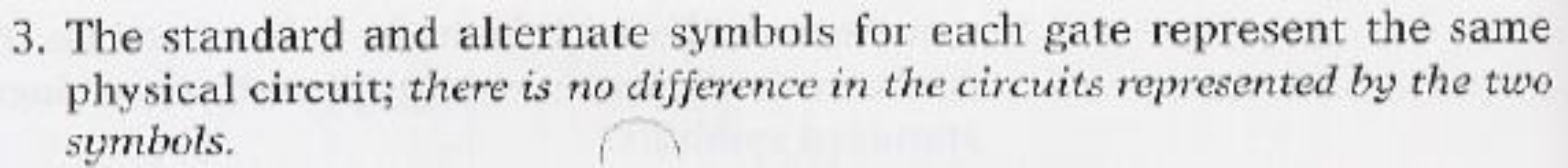
Several point should be stressed regarding the logic symbol equivalences:



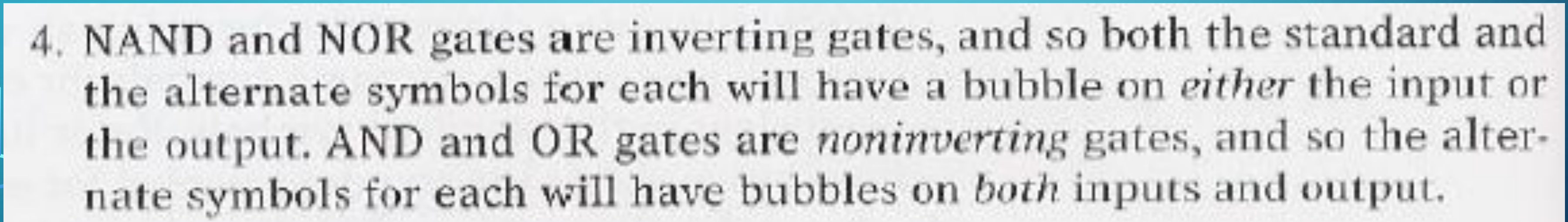
1. The equivalences can be extended to gates with *any* number of inputs.



2. None of the standard symbols have bubbles on their inputs, and all the alternate symbols do.



3. The standard and alternate symbols for each gate represent the same physical circuit; *there is no difference in the circuits represented by the two symbols.*



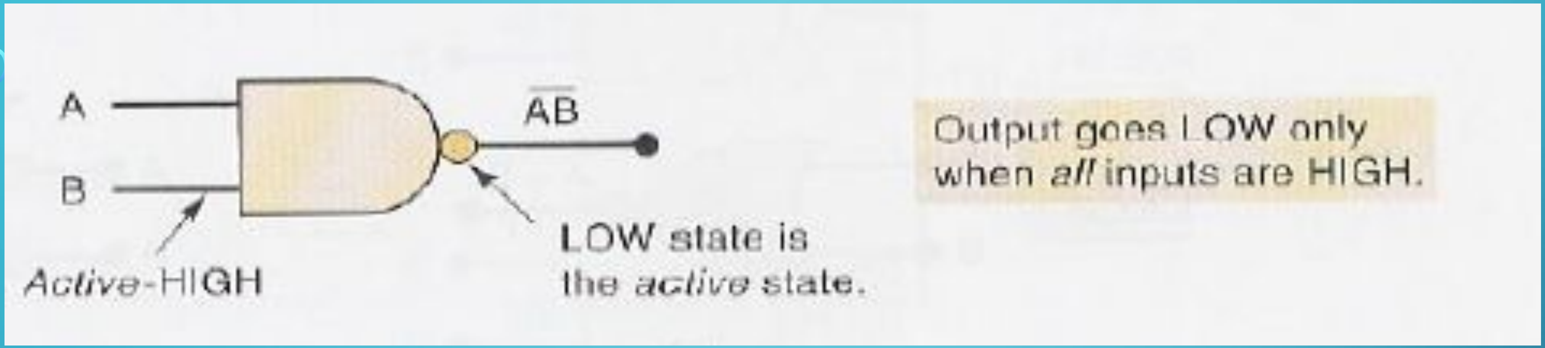
4. NAND and NOR gates are inverting gates, and so both the standard and the alternate symbols for each will have a bubble on *either* the input or the output. AND and OR gates are *noninverting* gates, and so the alternate symbols for each will have bubbles on *both* inputs and output.

Active Logic Levels: *There are two active logic levels*

Active High : *When an input or output line on a logic circuit symbol has no bubble on it ,that line is said to be active High.*

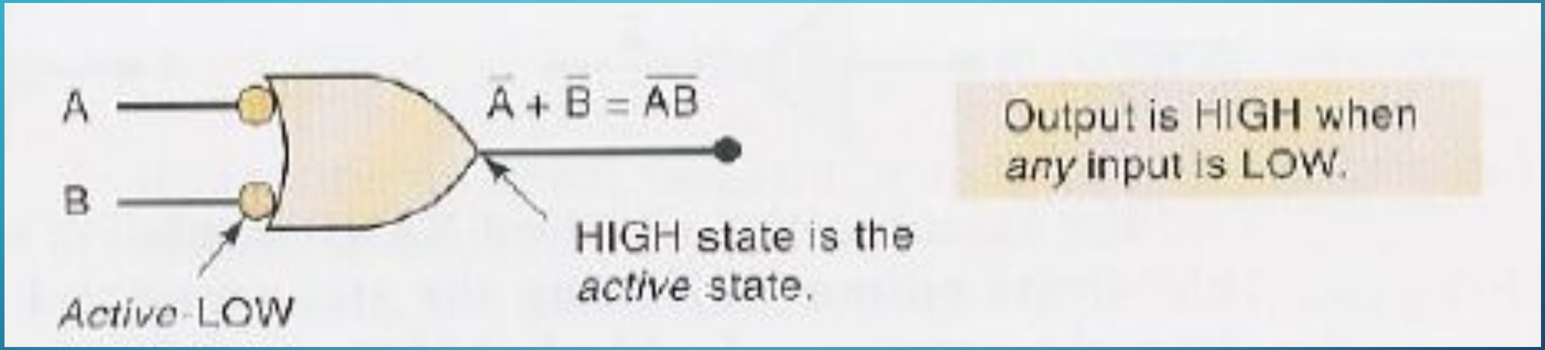
Active Low: *When an input or output line does have a bubble on it ,that line is said to be active Low.*

LOGIC SYMBOL INTERPRETATION



LOW state is the *active* state.

Output goes LOW only when *all* inputs are HIGH.

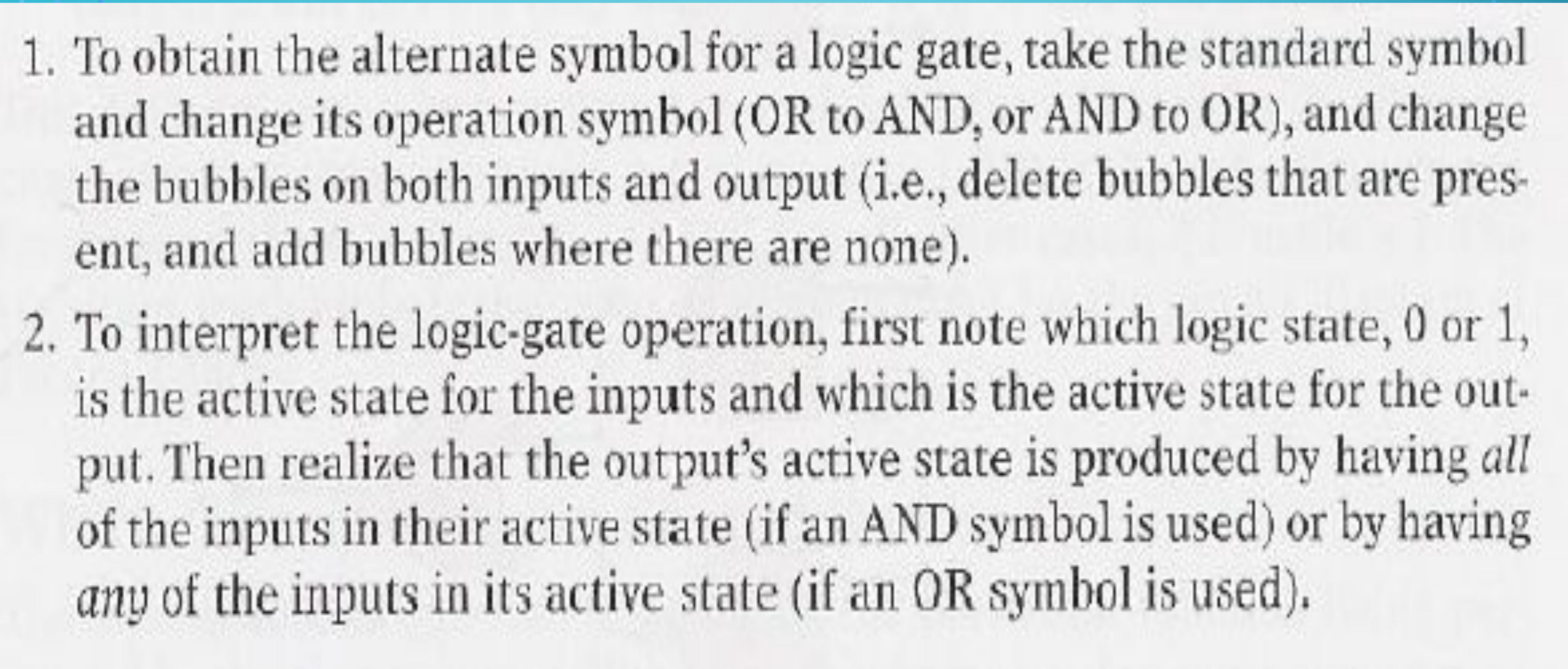


HIGH state is the *active* state.

Output is HIGH when *any* input is LOW.

*Word **all** used because of the AND symbol*
*Word **any** used because of the OR symbol*
We can see that the two interpretations of the NAND symbols in figure are different ways of saying the same thing.

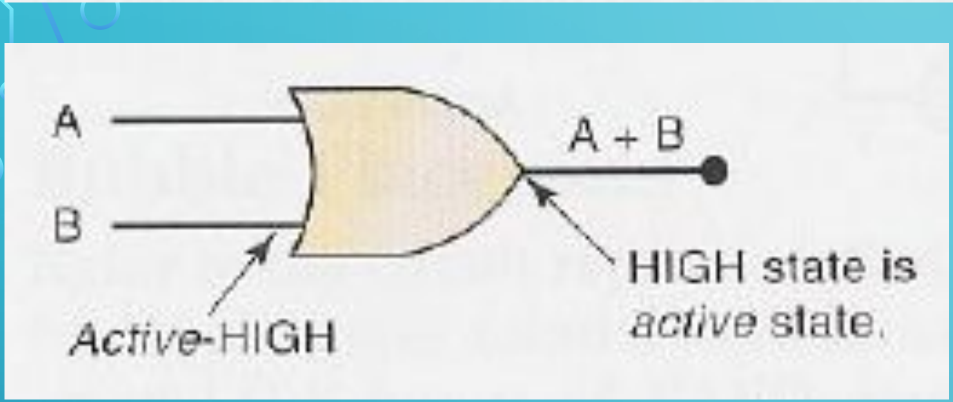
SUMMARY



1. To obtain the alternate symbol for a logic gate, take the standard symbol and change its operation symbol (OR to AND, or AND to OR), and change the bubbles on both inputs and output (i.e., delete bubbles that are present, and add bubbles where there are none).
2. To interpret the logic-gate operation, first note which logic state, 0 or 1, is the active state for the inputs and which is the active state for the output. Then realize that the output's active state is produced by having *all* of the inputs in their active state (if an AND symbol is used) or by having *any* of the inputs in its active state (if an OR symbol is used).

EXAMPLE

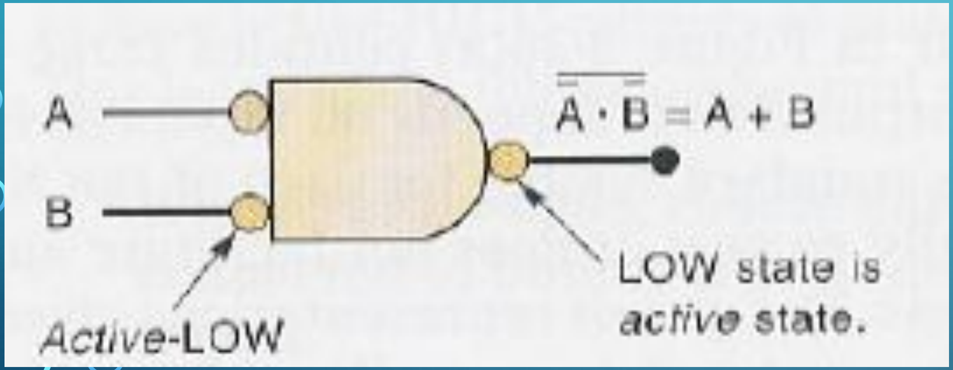
Give the interpretation of the two OR gate symbols.



Active-HIGH

HIGH state is *active* state.

Output goes HIGH when *any* input is HIGH.



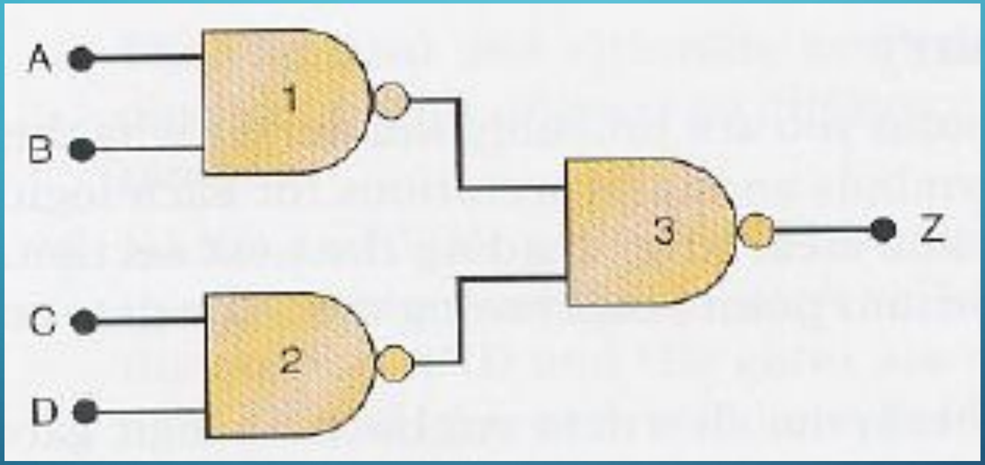
Active-LOW

LOW state is *active* state.

Output goes LOW only when *all* inputs are LOW.

WHICH GATE REPRESENTATION TO USE

Proper use of alternate gate symbol in the circuit diagram can make the circuit operation much clearer.



The circuit diagram uses the standard symbol for each of the NAND gates. While this diagram is logically correct, it does not facilitate an understanding of how the circuit functions.

However , these circuits can be analyzed more easily to determine the circuit operation. For all circuits resultant Truth Table is same

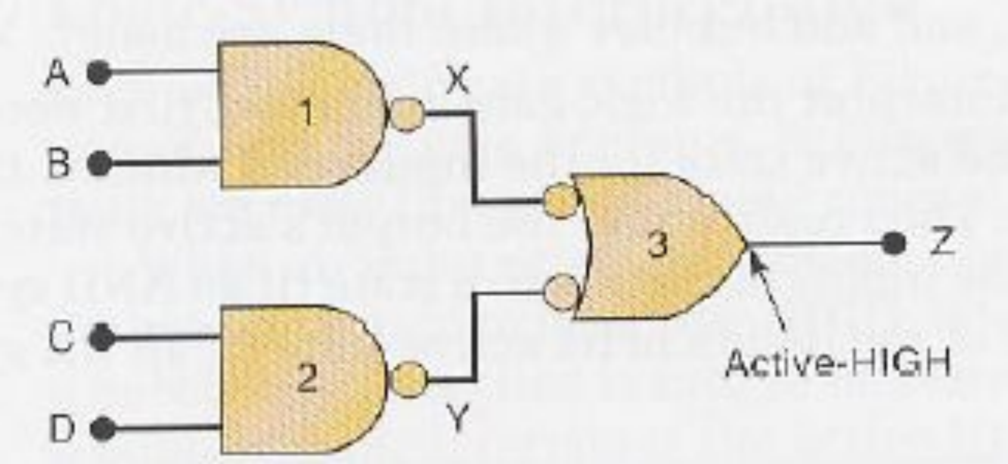


Figure-b

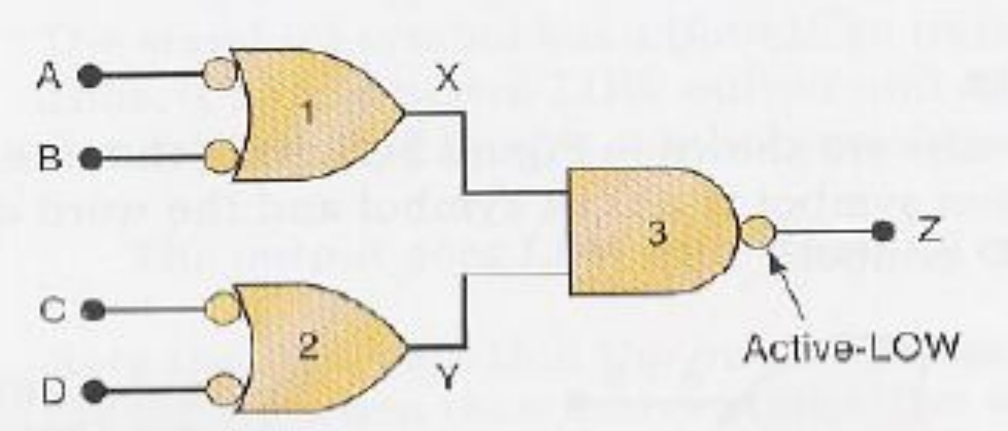
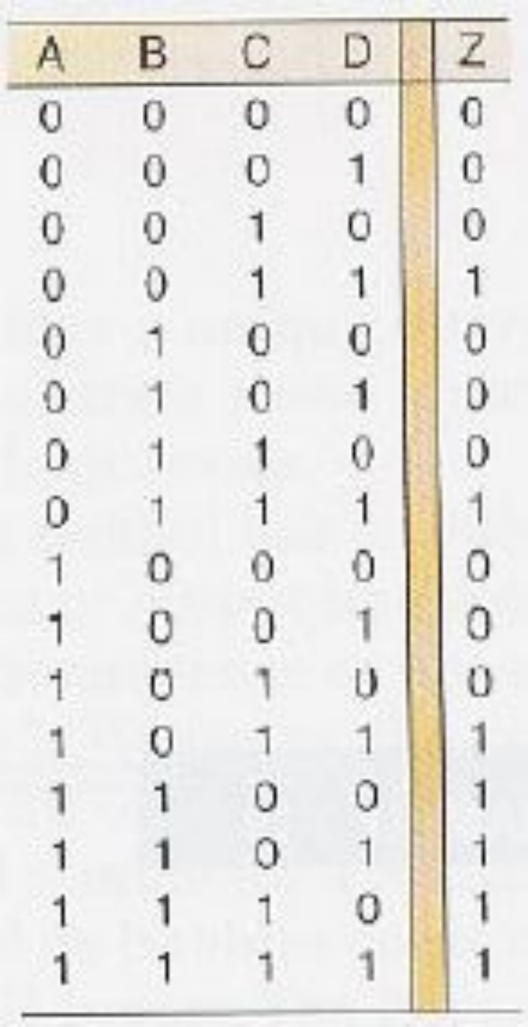


Figure-c



A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

The representation of the figure is obtained from original circuit diagram by replacing NAND gate 3 with its alternate symbol.

Output Z will go active high of X or Y is low.

X will go low only if A=B= 1 and Y will go low only if C=D= 1.

Putting this all together , we can describe the circuit operation as follows:

Output Z will go HIGH whenever either A=B=1 or C=D = 1

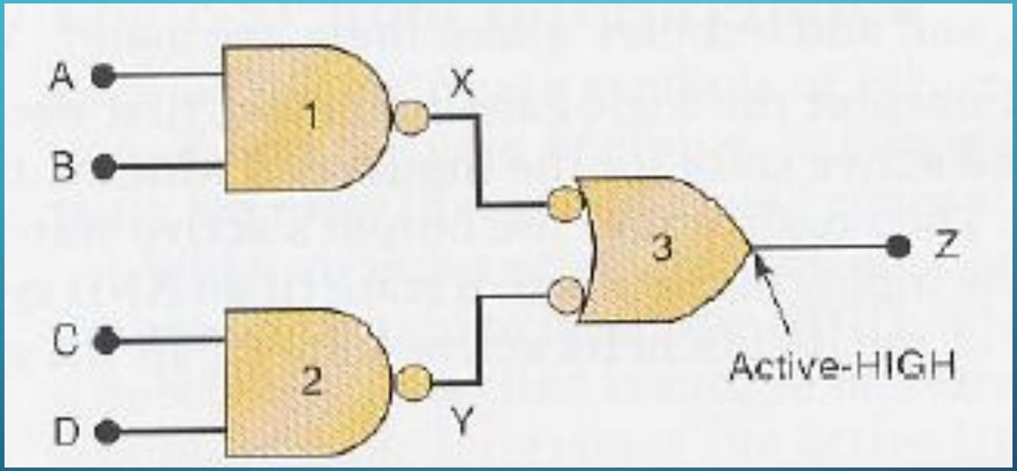


Figure - a

The representation of this figure is obtained from original circuit diagram by replacing NAND gate 1 and 2 with its alternate symbol.

Output Z will go active Low only when X = Y =1.

X will go High only if A or B is low and Y will go High only if C or D is low

Putting this all together , we can describe the circuit operation as follows:

Output Z will go LOW only when A or B is Low and C or D is Low.

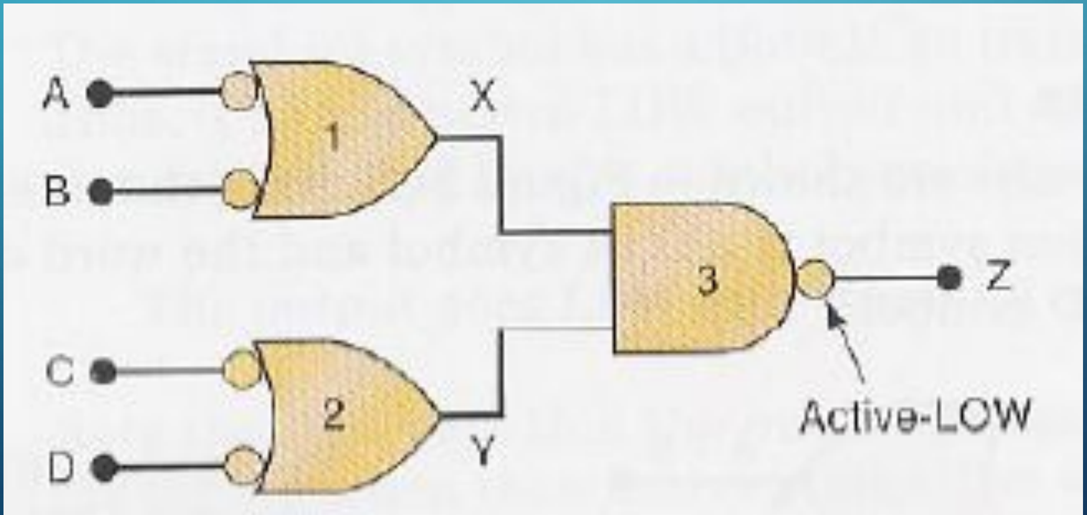
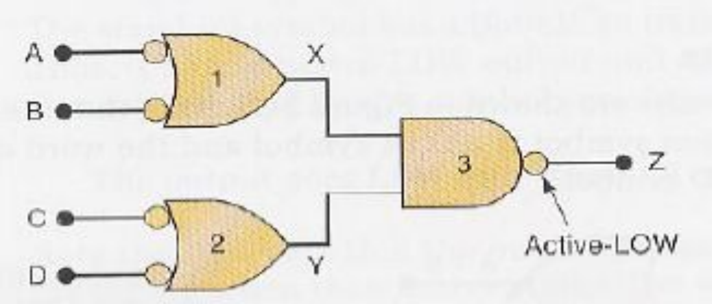


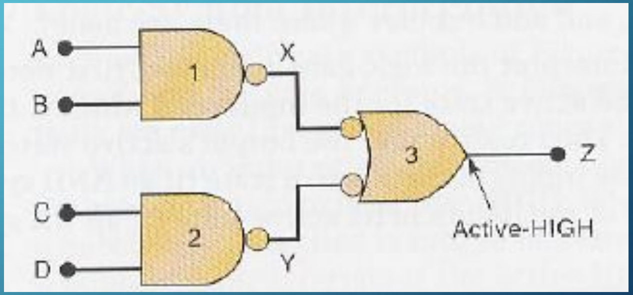
Figure – c

WHICH CIRCUIT DIAGRAM SHOULD BE USED

The answer to this question depends on the particular function being performed by the circuit output.

If the circuit is being used to cause some action (e.g , turn on an LED or activate another logic circuit) when output Z goes to the 1 state , then we say that Z is to be active –HIGH , and the circuit diagram of figure –b should be used.



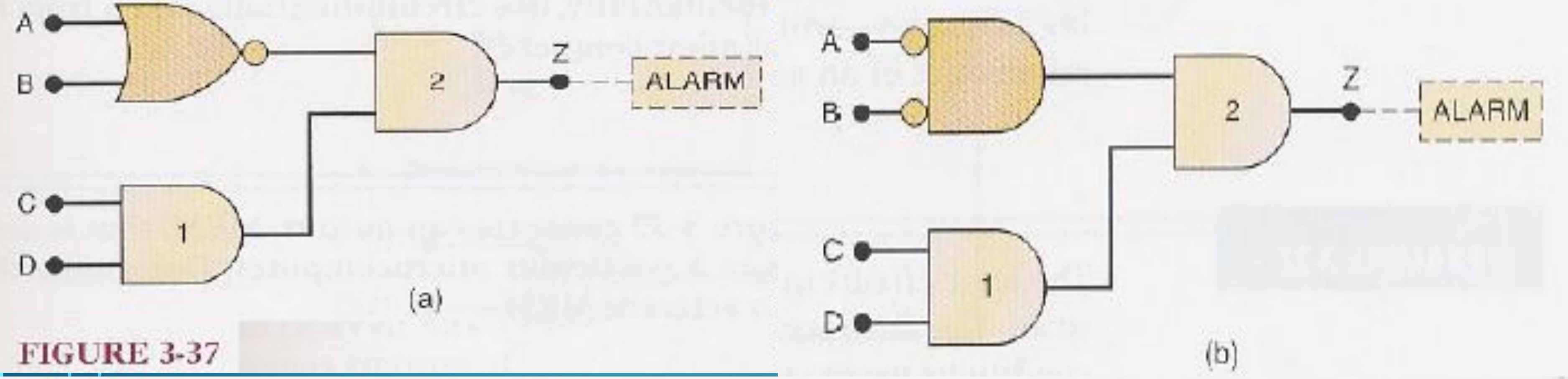


On the other hand, if the circuit is being used to cause some when output Z goes to the 0 state , then we say that Z is to be active –LOW , and the circuit diagram of figure – c should be used.

BUBBLE PLACEMENT

Whenever possible, choose gate symbols so that bubble outputs are connected to bubble inputs, and nonbubble outputs to nonbubble inputs.

Example: The logic circuit in Figure 3-37(a) is being used to activate an alarm when its output Z goes HIGH. Modify the circuit diagram so that it represents the circuit operation more effectively.



(a)

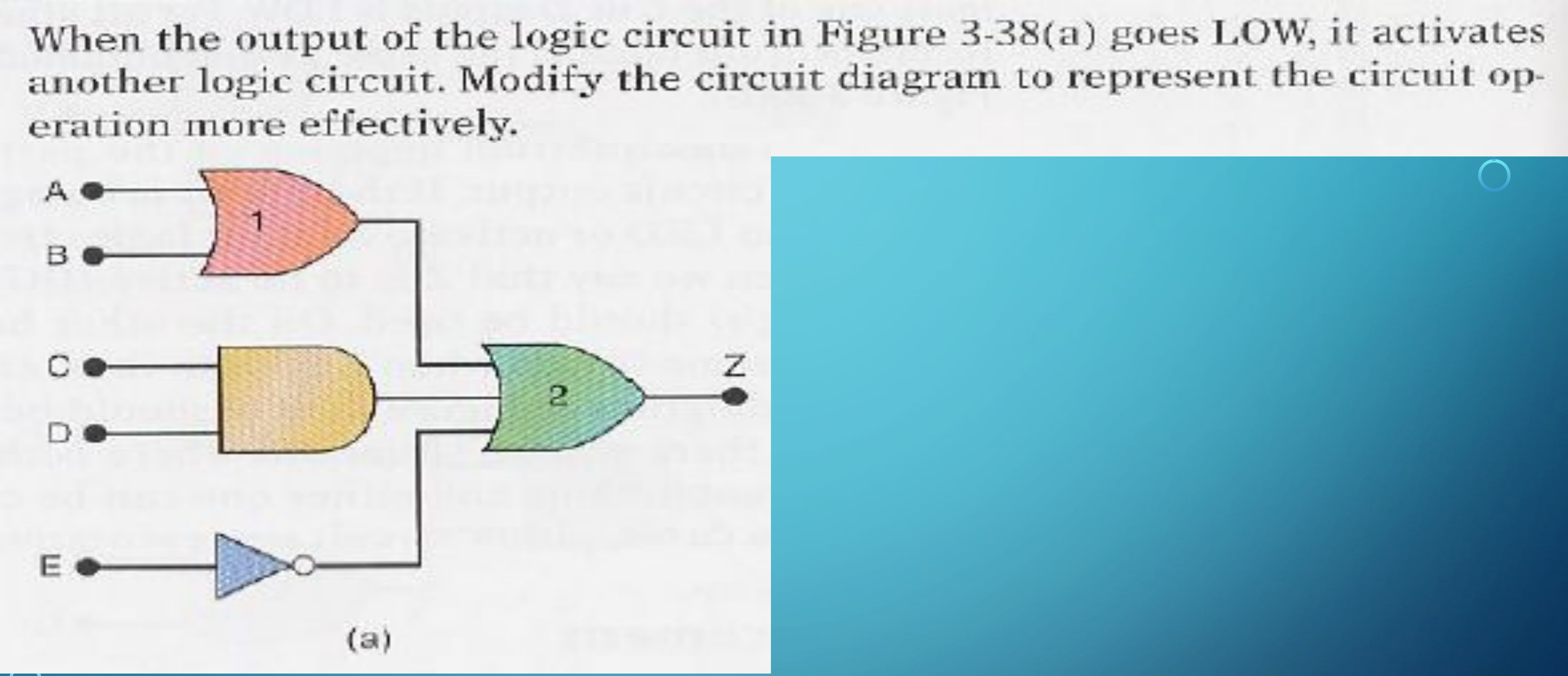
(b)

FIGURE 3-37

Solution

Because Z = 1 will activate the alarm, Z is to be active-HIGH. Thus, the AND gate 2 symbol does not have to be changed. The NOR gate symbol should be changed to the alternate symbol with a nonbubble (active-HIGH) output to match the nonbubble input of AND gate 2, as shown in Figure 3-37(b). Note that the circuit now has nonbubble outputs connected to the nonbubble inputs of gate 2.

EXAMPLE:

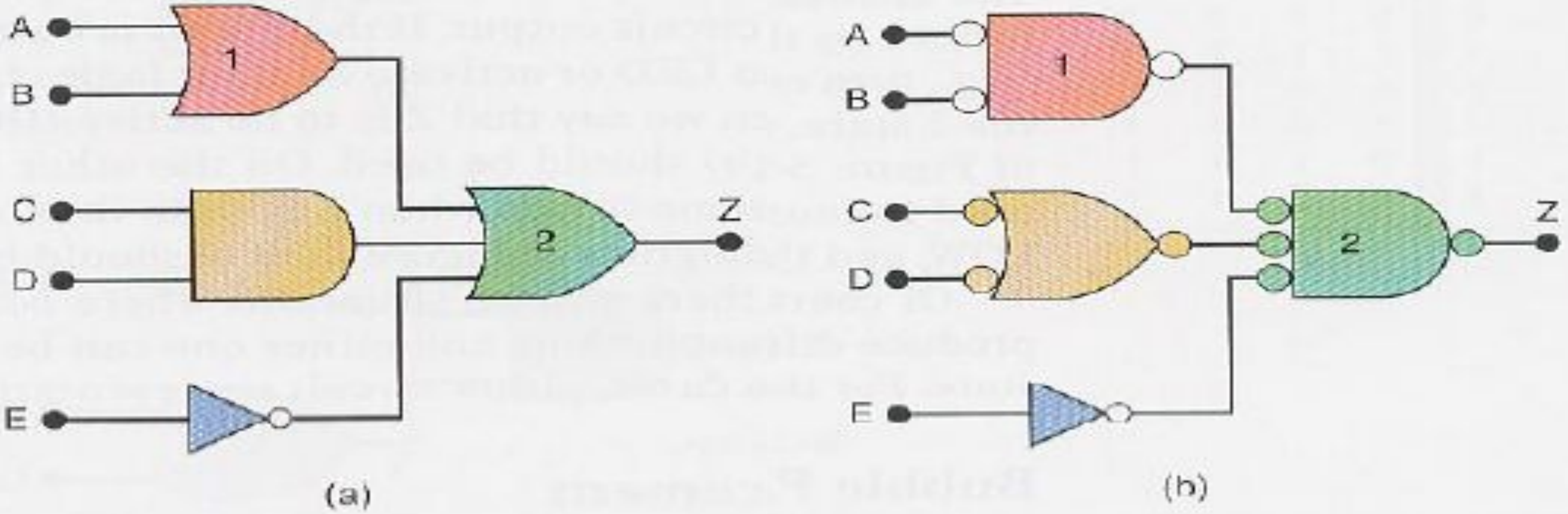


When the output of the logic circuit in Figure 3-38(a) goes LOW, it activates another logic circuit. Modify the circuit diagram to represent the circuit operation more effectively.

(a)

EXAMPLE:

When the output of the logic circuit in Figure 3-38(a) goes LOW, it activates another logic circuit. Modify the circuit diagram to represent the circuit operation more effectively.



(a)

(b)

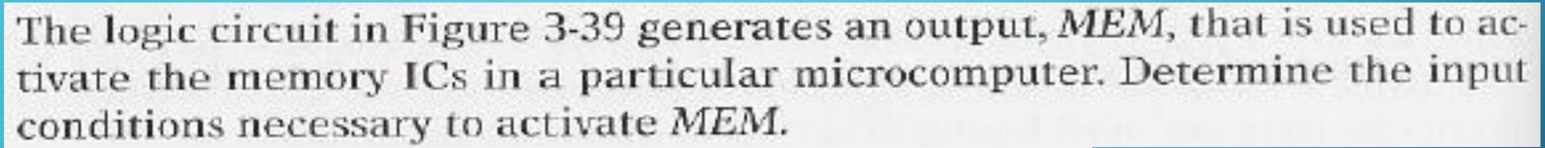
Solution

Because Z is to be active-LOW, the symbol for OR gate 2 must be changed to its alternate symbol, as shown in Figure 3-38(b). The new OR gate 2 symbol has bubble inputs, and so the AND gate and OR gate 1 symbols must be changed to bubbled outputs, as shown in Figure 3-38(b). The INVERTER already has a bubble output. Now the circuit has all bubble outputs connected to bubble inputs of gate 2.

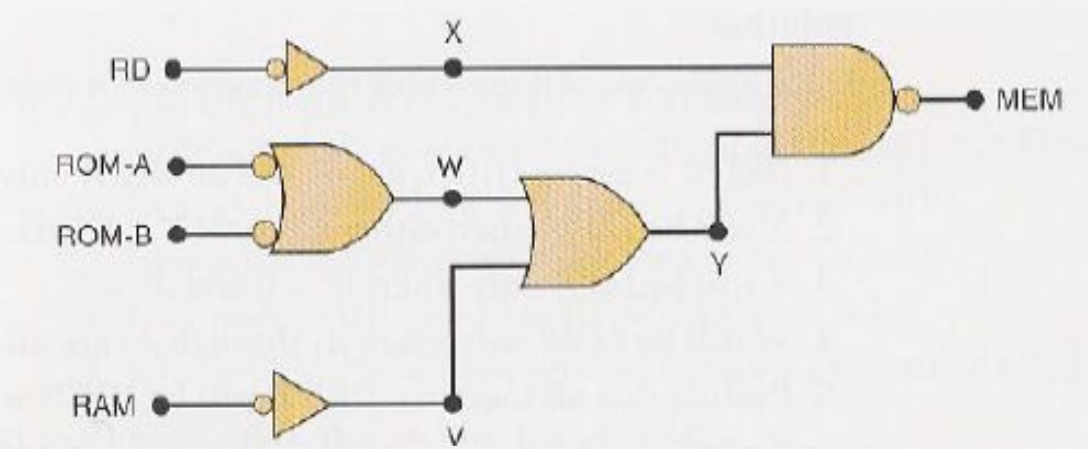
ANALYZING CIRCUITS

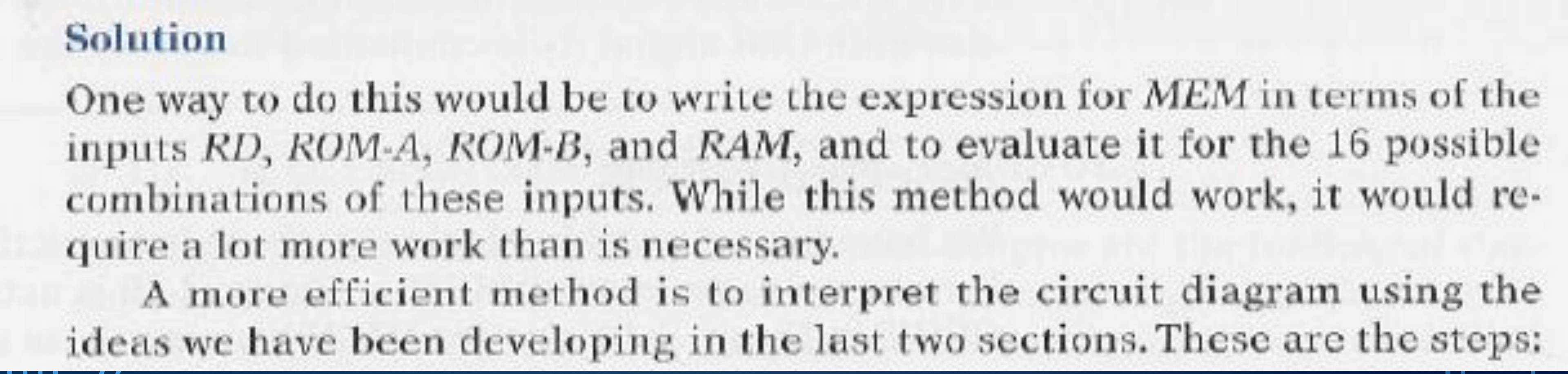
When a logic circuit schematic is drawn using the rules we followed in these examples, it is much easier for an engineer or technician (or student) to follow the signal flow through the circuit and to determine the input condition that are needed to activate the output.

Example:



The logic circuit in Figure 3-39 generates an output, *MEM*, that is used to activate the memory ICs in a particular microcomputer. Determine the input conditions necessary to activate *MEM*.

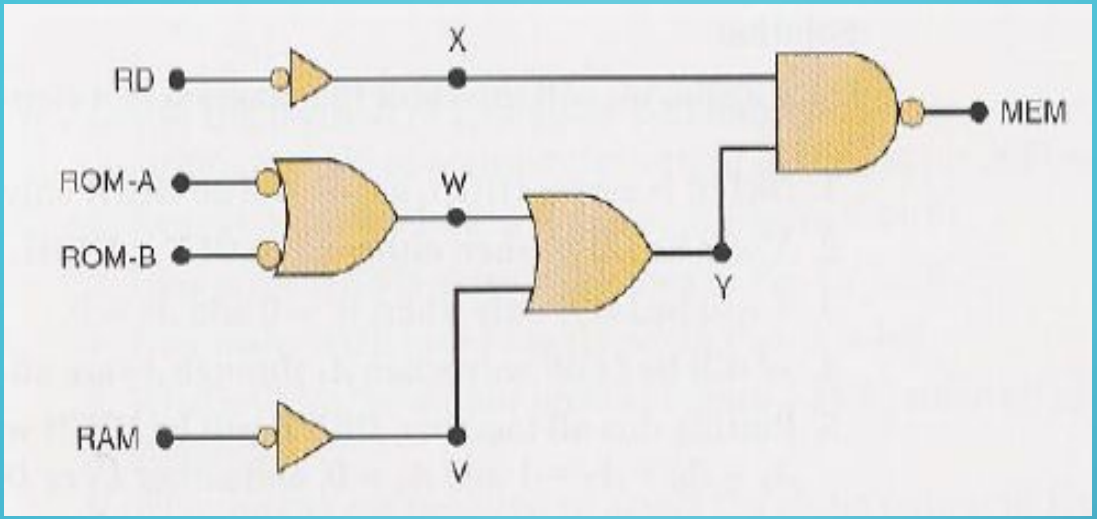


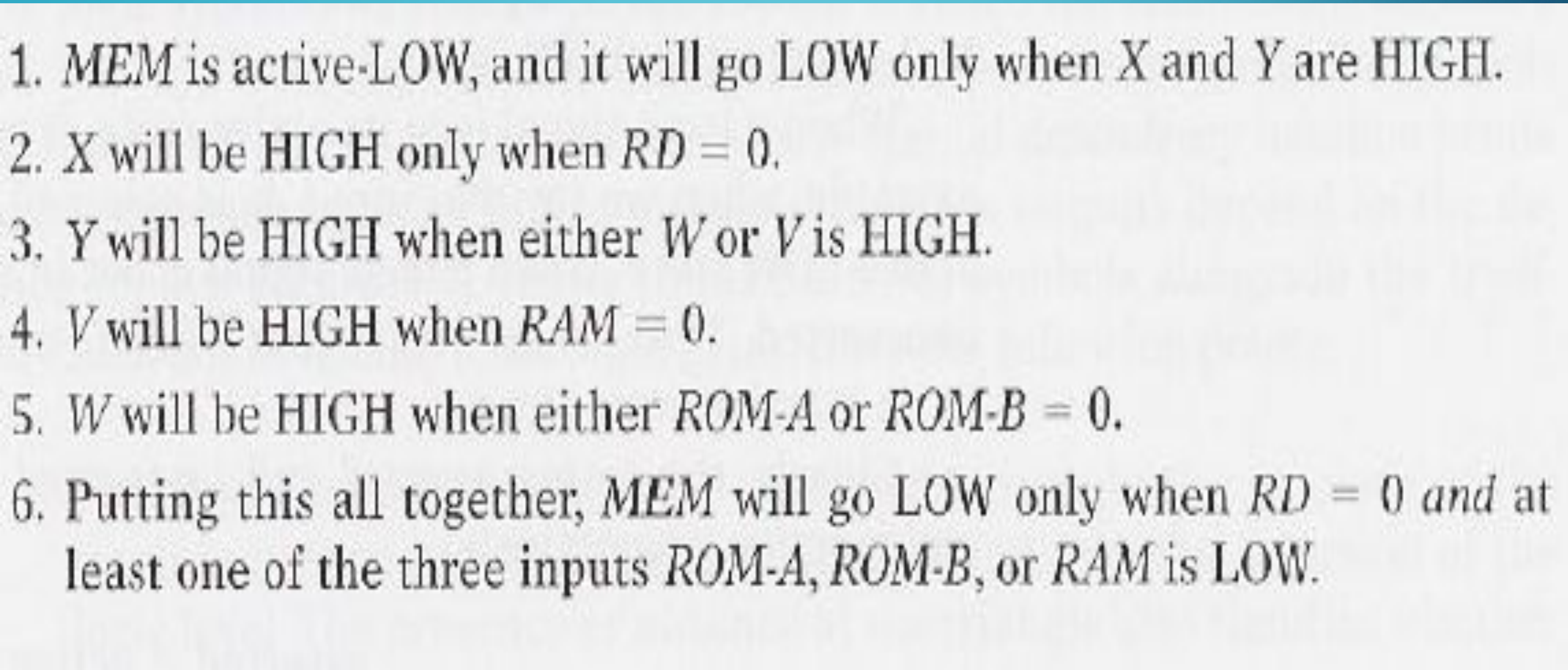


Solution

One way to do this would be to write the expression for *MEM* in terms of the inputs *RD*, *ROM-A*, *ROM-B*, and *RAM*, and to evaluate it for the 16 possible combinations of these inputs. While this method would work, it would require a lot more work than is necessary.

A more efficient method is to interpret the circuit diagram using the ideas we have been developing in the last two sections. These are the steps:

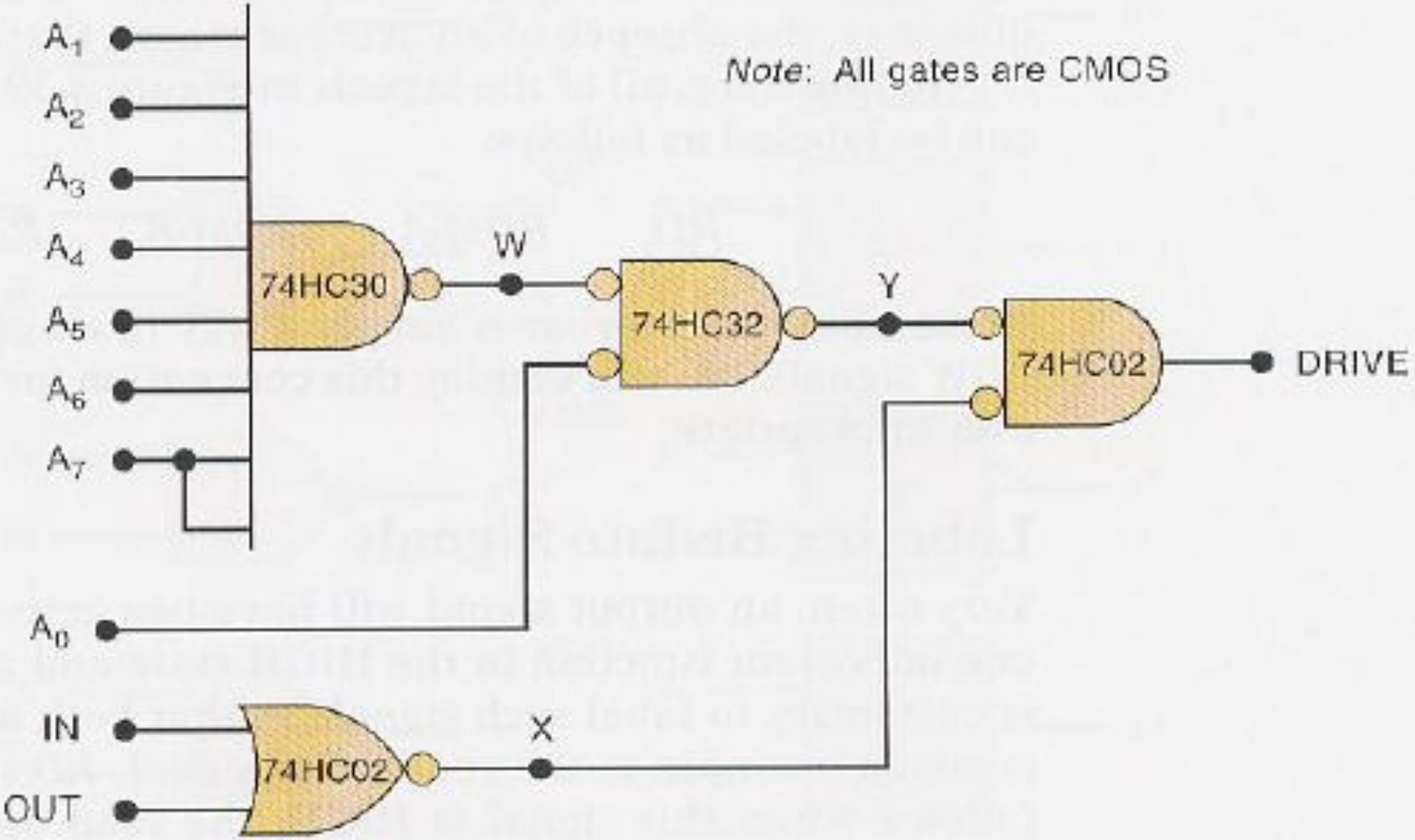


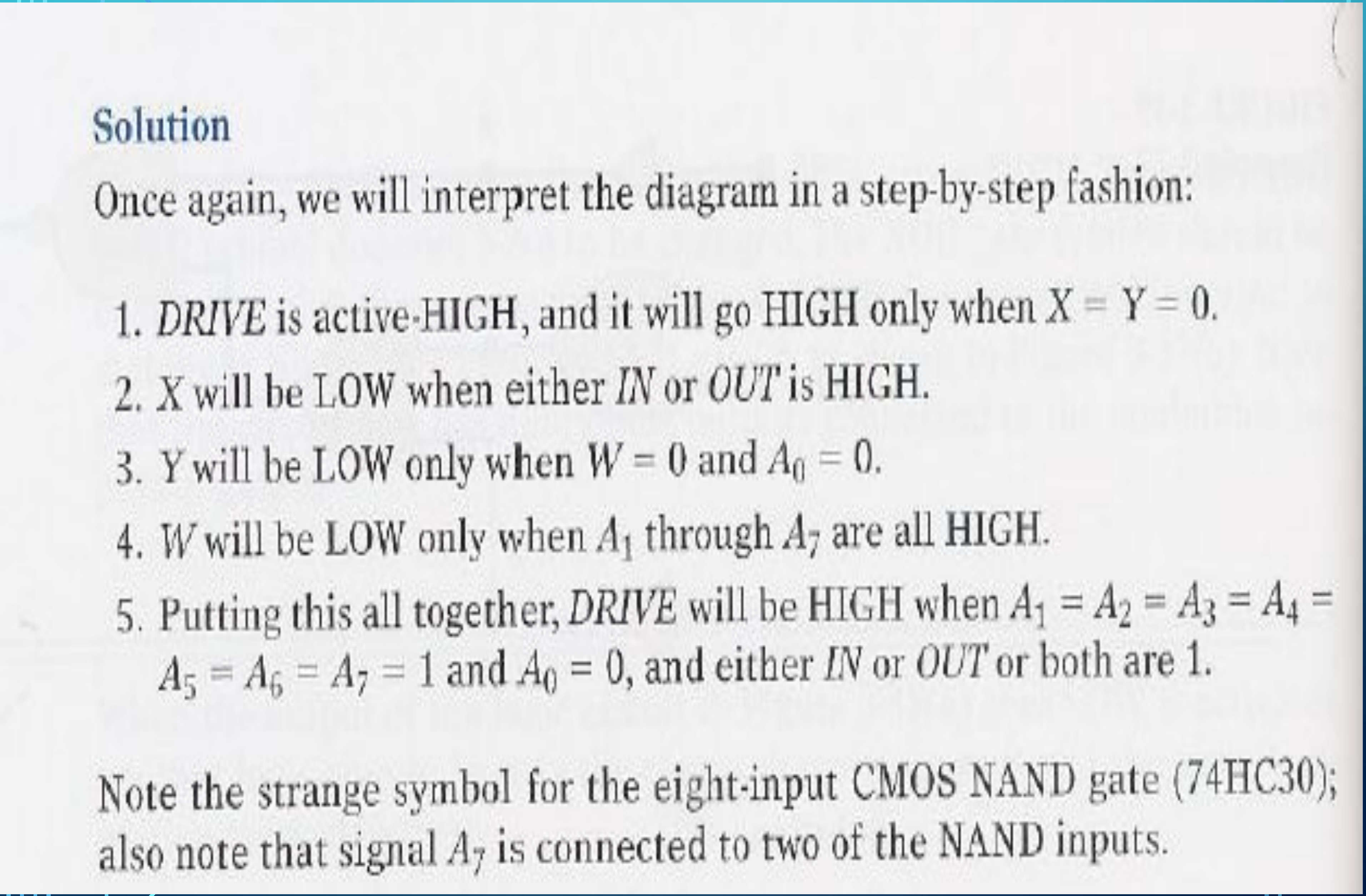


1. *MEM* is active-LOW, and it will go LOW only when X and Y are HIGH.
2. *X* will be HIGH only when *RD* = 0.
3. *Y* will be HIGH when either *W* or *V* is HIGH.
4. *V* will be HIGH when *RAM* = 0.
5. *W* will be HIGH when either *ROM-A* or *ROM-B* = 0.
6. Putting this all together, *MEM* will go LOW only when *RD* = 0 *and* at least one of the three inputs *ROM-A*, *ROM-B*, or *RAM* is LOW.

EXAMPLE:

The logic circuit in Figure 3-40 is used to control the drive spindle motor for a floppy disk drive when the microcomputer is sending data to or receiving data from the disk. The circuit will turn on the motor when *DRIVE* = 1. Determine the input conditions necessary to turn on the motor.





Solution

Once again, we will interpret the diagram in a step-by-step fashion:

1. *DRIVE* is active-HIGH, and it will go HIGH only when $X = Y = 0$.
2. X will be LOW when either *IN* or *OUT* is HIGH.
3. Y will be LOW only when $W = 0$ and $A_0 = 0$.
4. W will be LOW only when A_1 through A_7 are all HIGH.
5. Putting this all together, *DRIVE* will be HIGH when $A_1 = A_2 = A_3 = A_4 = A_5 = A_6 = A_7 = 1$ and $A_0 = 0$, and either *IN* or *OUT* or both are 1.

Note the strange symbol for the eight-input CMOS NAND gate (74HC30); also note that signal A_7 is connected to two of the NAND inputs.

COMPLETE DESIGN PROCEDURE

Any logic problem can be solved using the following step by step procedure.

1. *Interpret the problem and set up truth tables to describe its operation.*
2. *Write the AND(product) term for each case where the output is 1.*
3. *Write the SOP expression for the output.*

4. *Simplify the output expression if possible.*

5. *Implement the circuit for the final, simplified expression.*

EXAMPLE

Design a logic circuit that has three inputs, *A*, *B*, and *C*, and whose output will be HIGH only when a majority of the inputs are HIGH.

Step 1. Set up the truth table.

A	B	C	x	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	→ $\overline{A}BC$
1	0	0	0	
1	0	1	1	→ $A\overline{B}C$
1	1	0	1	→ $AB\overline{C}$
1	1	1	1	→ ABC

Step 2. Write the AND term for each case where the output is a 1.

$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Step 3. Write the sum-of-products expression for the output.

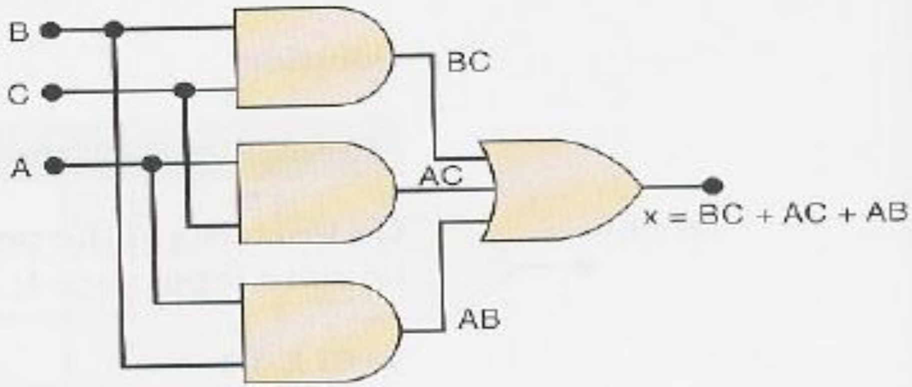
$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Step 4. Simplify the output expression.

$$x = \overline{A}BC + ABC + A\overline{B}C + ABC + AB\overline{C} + ABC$$

$$x = BC(\overline{A} + A) + AC(\overline{B} + B) + AB(\overline{C} + C)$$

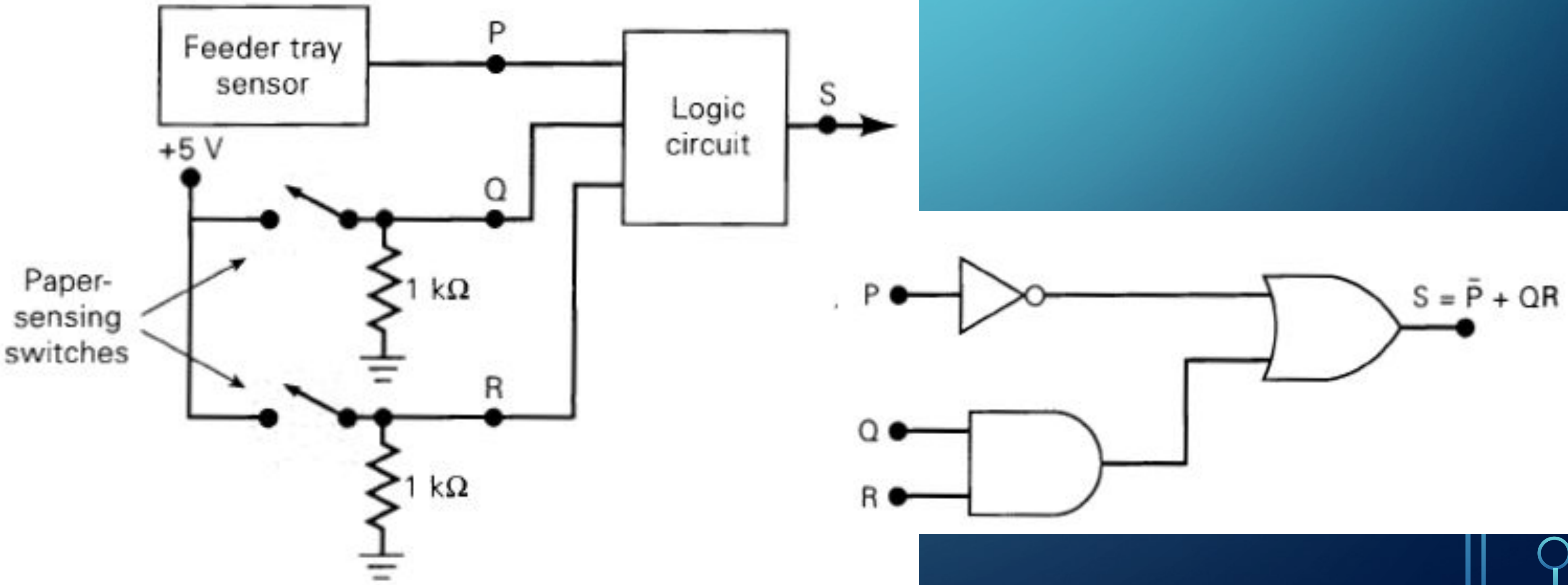
$$x = BC + AC + AB$$

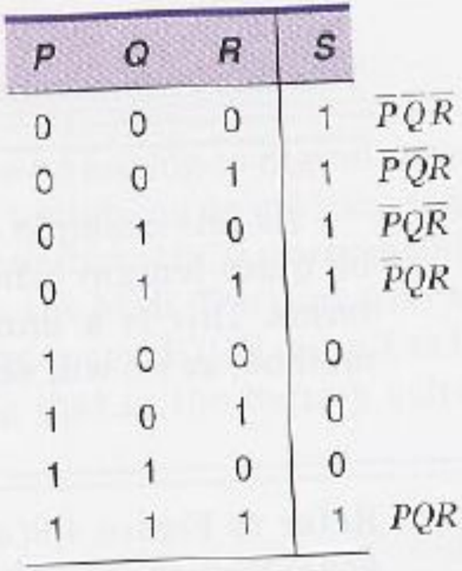


Step 5. Implement the circuit for the final expression.

EXAMPLE

Refer to Figure In a simple copy machine, a stop signal, *S*, is to be generated to stop the machine operation and energize an indicator light whenever either of the following conditions exists: (1) there is no paper in the paper feeder tray; or (2) the two microswitches in the paper path are activated, indicating a jam in the paper path. The presence of paper in the feeder tray is indicated by a HIGH at logic signal *P*. Each of the microswitches produces a logic signal (*Q* and *R*) that goes HIGH whenever paper is passing over the switch to activate it. Design the logic circuit to produce a HIGH at output signal *S* for the stated conditions, and implement it using the 74HC00 CMOS quad two-input NAND chip.





P	Q	R	S	
0	0	0	1	$\overline{P}\,\overline{Q}\,\overline{R}$
0	0	1	1	$\overline{P}\,\overline{Q}R$
0	1	0	1	$\overline{P}Q\overline{R}$
0	1	1	1	$\overline{P}QR$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	PQR

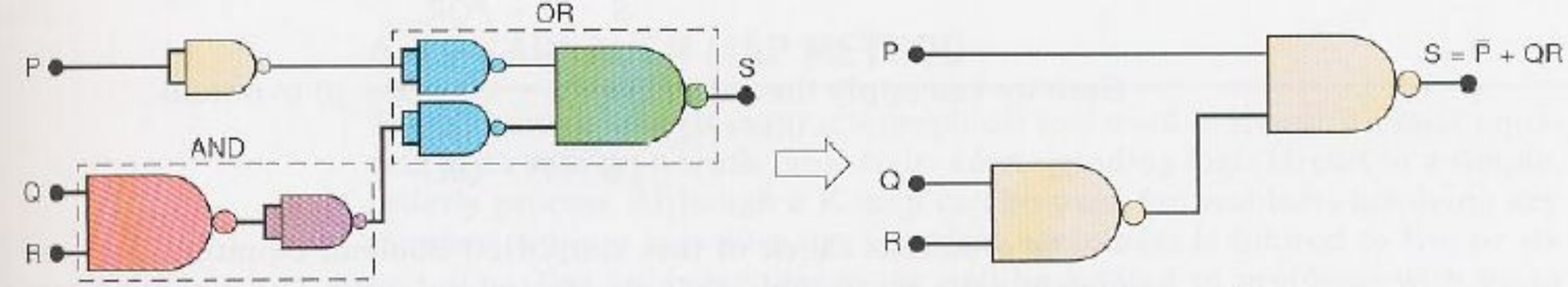
$$S = \overline{P}\,\overline{Q}\,\overline{R} + \overline{P}\,\overline{Q}R + \overline{P}Q\overline{R} + \overline{P}QR + PQR$$

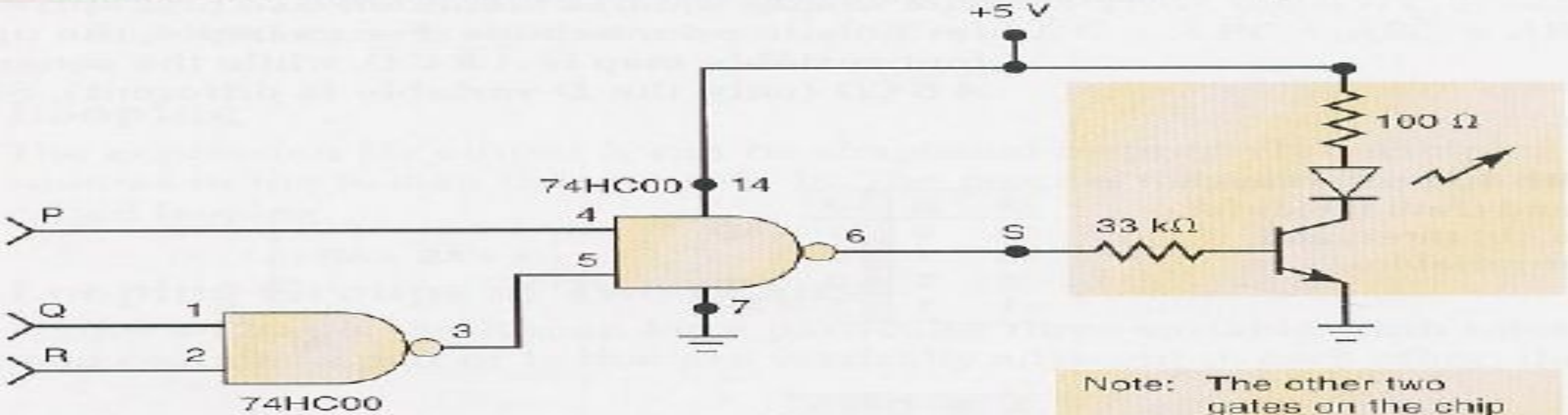
$$S = \overline{P}\,\overline{Q}(\overline{R} + R) + \overline{P}Q(\overline{R} + R) + PQR$$

$$S = \overline{P}\,\overline{Q} + \overline{P}Q + PQR$$

$$S = \overline{P} + PQR$$

$$S = \overline{P} + QR$$





Note: The other two gates on the chip