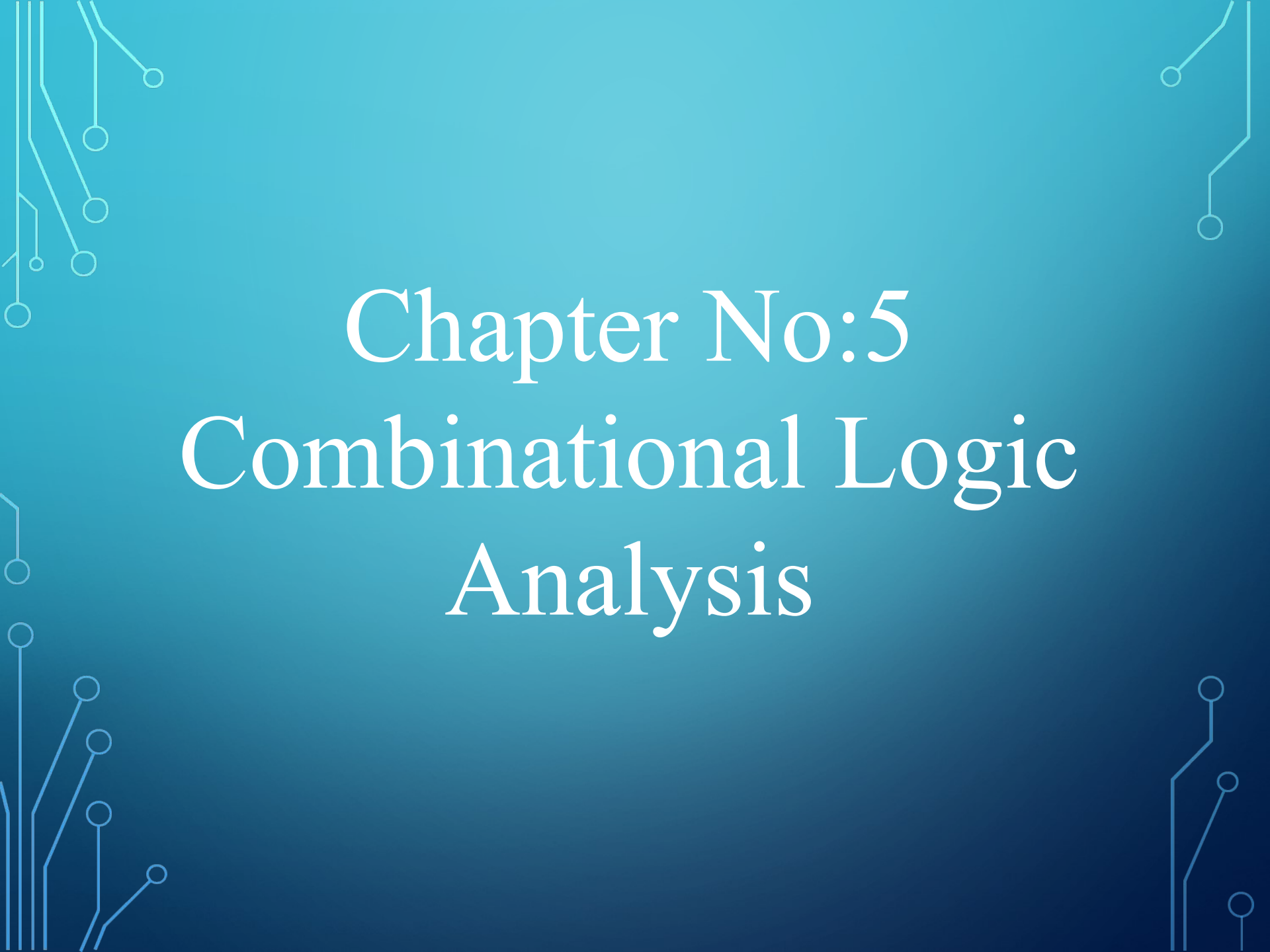




DIGITAL LOGIC DESIGN

EE(1005)

LECTURE-16-17

The background is a blue gradient with white circuit-like lines and circles in the corners. The text is centered in a white serif font.

Chapter No:5

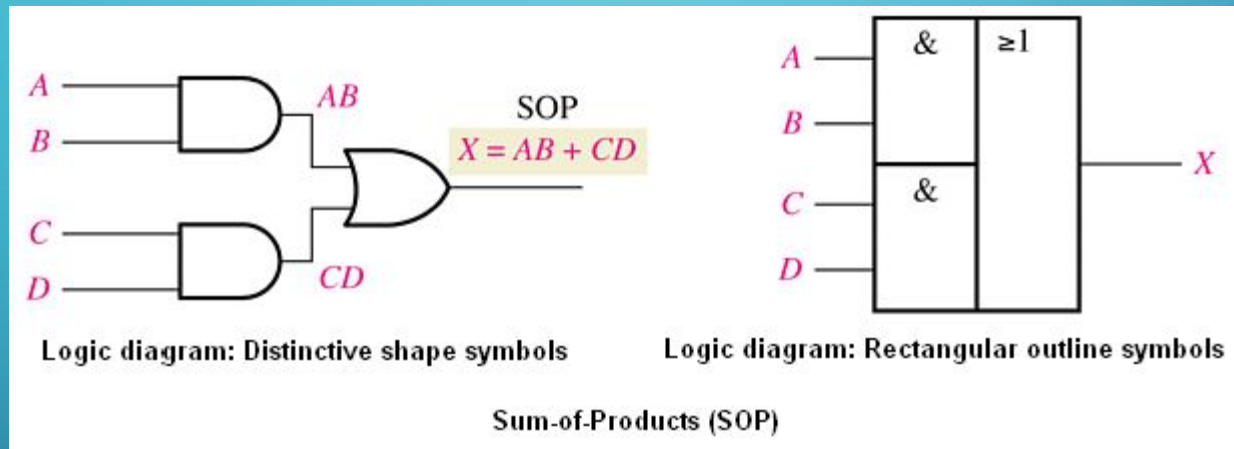
Combinational Logic Analysis

CHAPTER OUTLINE

- Basic Combinational Logic Circuits
- Implementing Combinational Logic
- The Universal Property of NAND and NOR Gates
- Combinational Logic Using NAND and NOR Gates
- Pulse Waveform Operation

BASIC COMBINATIONAL LOGIC CIRCUITS

- AND-OR Logic



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).

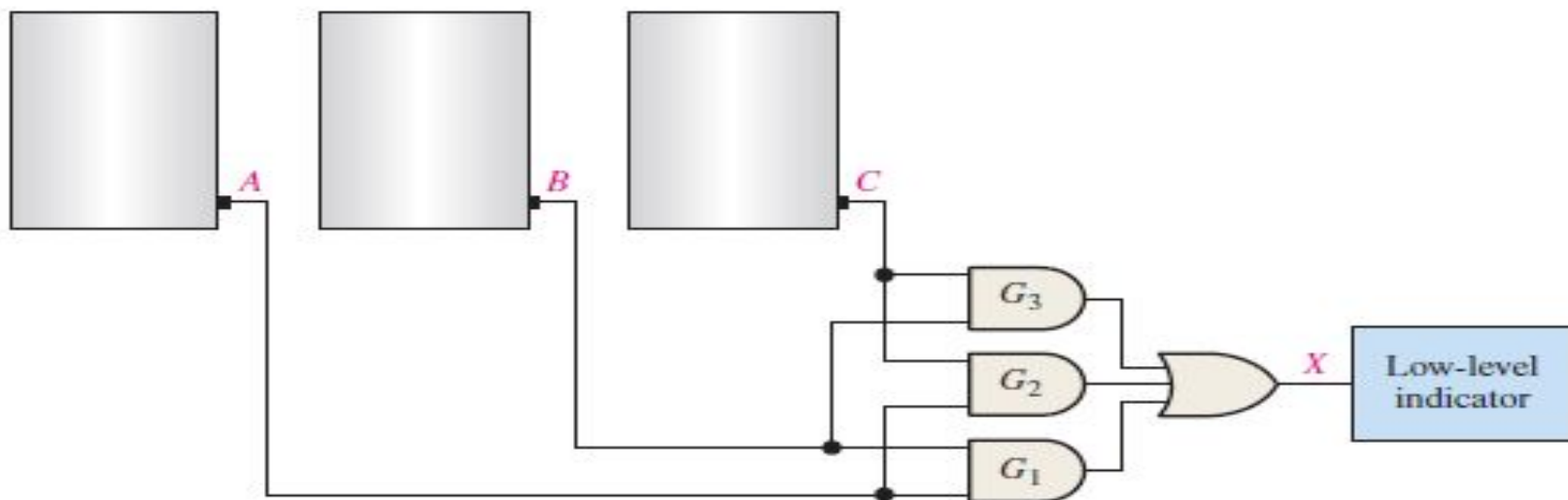
EXAMPLE

In a certain chemical-processing plant, a liquid chemical is used in a manufacturing process. The chemical is stored in three different tanks. A level sensor in each tank produces a HIGH voltage when the level of chemical in the tank drops below a specified point.

Design a circuit that monitors the chemical level in each tank and indicates when the level in any two of the tanks drops below the specified point.

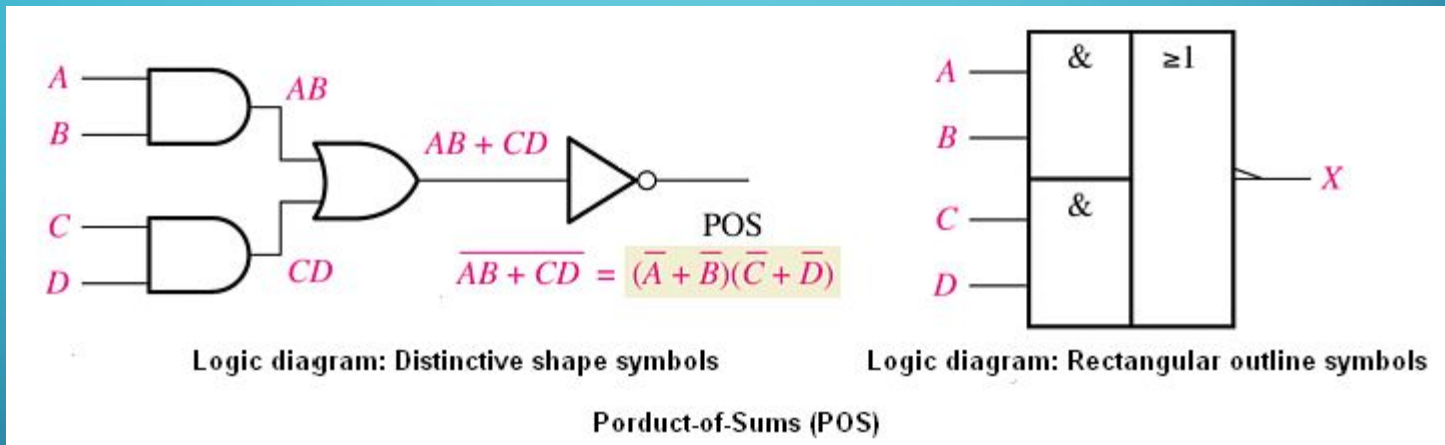
Solution

The AND-OR circuit in Figure 5–2 has inputs from the sensors on tanks *A*, *B*, and *C* as shown. The AND gate G_1 checks the levels in tanks *A* and *B*, gate G_2 checks tanks *A* and *C*, and gate G_3 checks tanks *B* and *C*. When the chemical level in any two of the tanks gets too low, one of the AND gates will have HIGHS on both of its inputs, causing its output to be HIGH; and so the final output *X* from the OR gate is HIGH. This HIGH input is then used to activate an indicator such as a lamp or audible alarm, as shown in the figure.



BASIC COMBINATIONAL LOGIC CIRCUITS

- AND-OR Invert Logic



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).

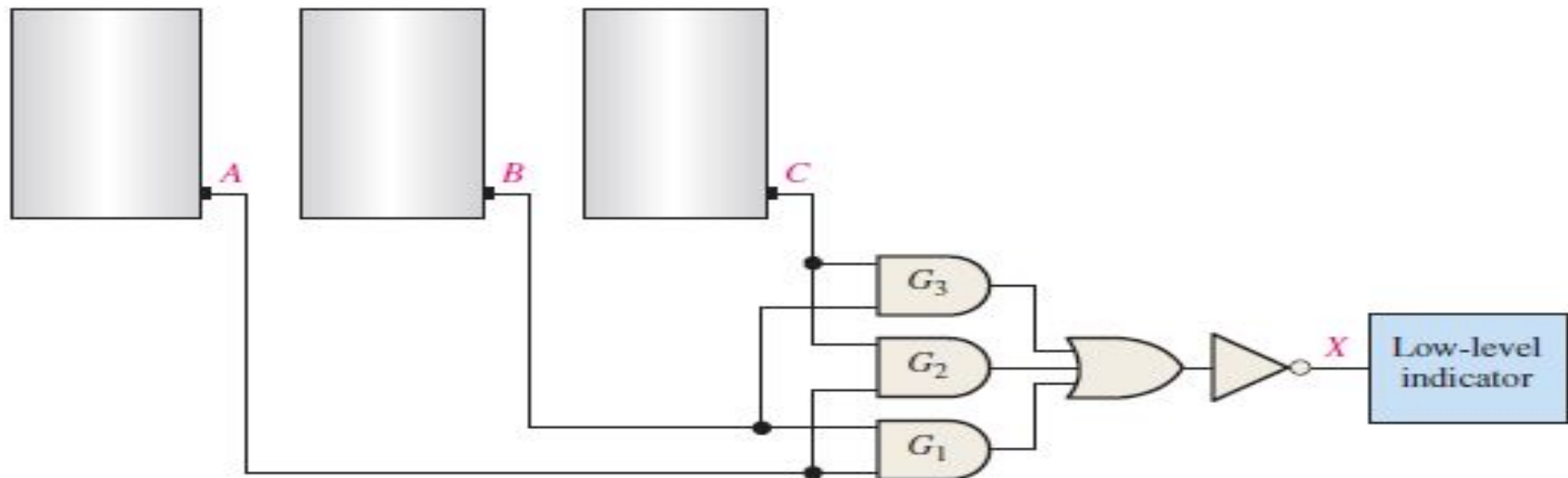
EXAMPLE

The sensors in the chemical tanks of Example 5–1 are being replaced by a new model that produces a LOW voltage instead of a HIGH voltage when the level of the chemical in the tank drops below a critical point.

Modify the circuit in Figure 5–2 to operate with the different input levels and still produce a HIGH output to activate the indicator when the level in any two of the tanks drops below the critical point. Show the logic diagram.

Solution

The AND-OR-Invert circuit in Figure 5–4 has inputs from the sensors on tanks *A*, *B*, and *C* as shown. The AND gate G_1 checks the levels in tanks *A* and *B*, gate G_2 checks tanks *A* and *C*, and gate G_3 checks tanks *B* and *C*. When the chemical level in any two of the tanks gets too low, each AND gate will have a LOW on at least one input, causing its output to be LOW and, thus, the final output *X* from the inverter is HIGH. This HIGH output is then used to activate an indicator.



EXAMPLE

Use exclusive-OR gates to implement an even-parity code generator for an original 4-bit code.

Solution

Recall from Chapter 2 that a parity bit is added to a binary code in order to provide error detection. For even parity, a parity bit is added to the original code to make the total number of 1s in the code even. The circuit in Figure 5–7 produces a 1 output when there is an odd number of 1s on the inputs in order to make the total number of 1s in the output code even. A 0 output is produced when there is an even number of 1s on the inputs.

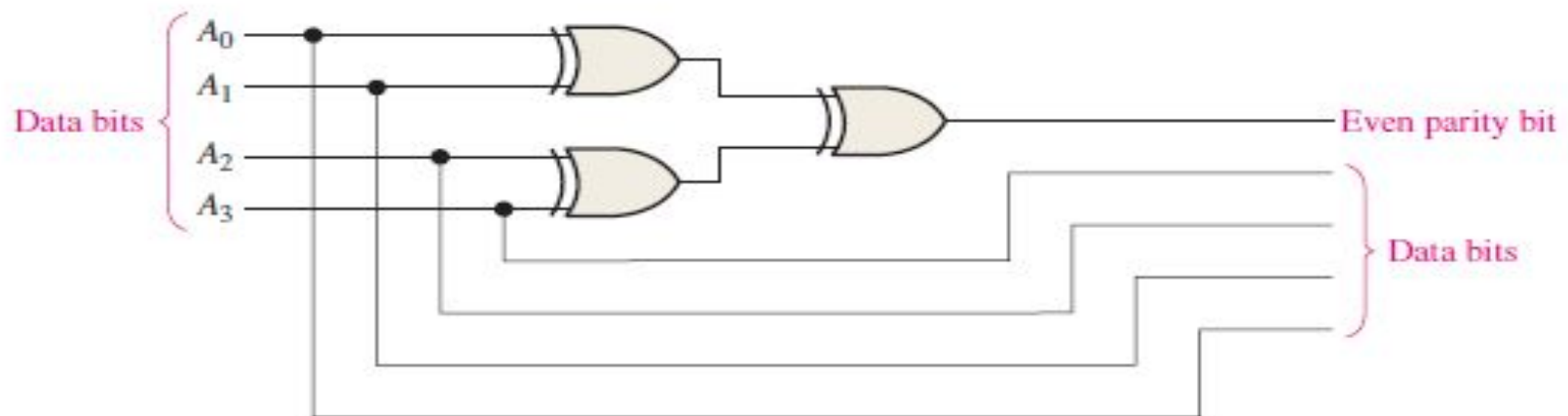


FIGURE 5-7 Even-parity generator.

EXAMPLE

Use exclusive-OR gates to implement an even-parity checker for the 5-bit code generated by the circuit in Example 5–3.

Solution

The circuit in Figure 5–8 produces a 1 output when there is an error in the five-bit code and a 0 when there is no error.

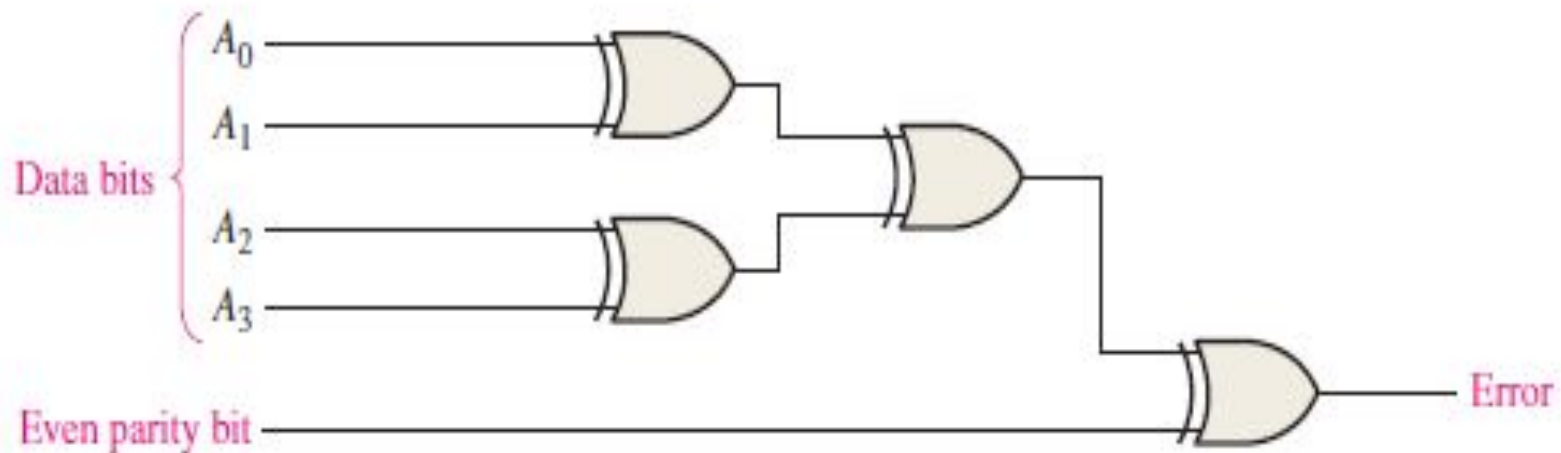
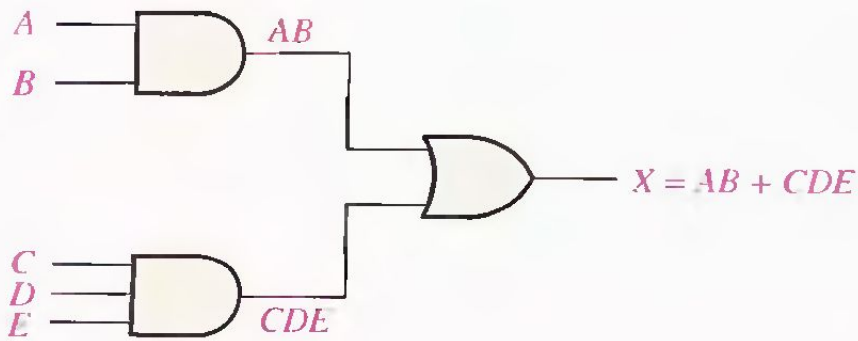


FIGURE 5-8 Even-parity checker.

Implementing Combinational Logic

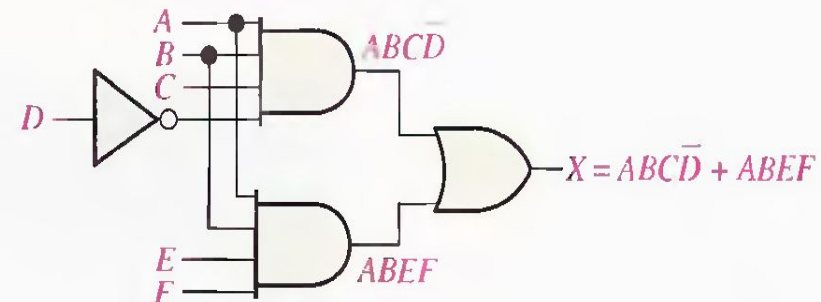
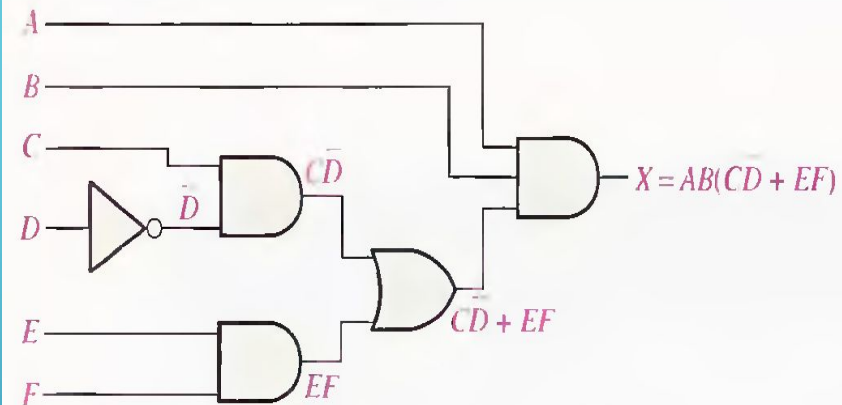
From a Boolean Expression to a Logic Circuit

$$X = AB + CDE$$



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

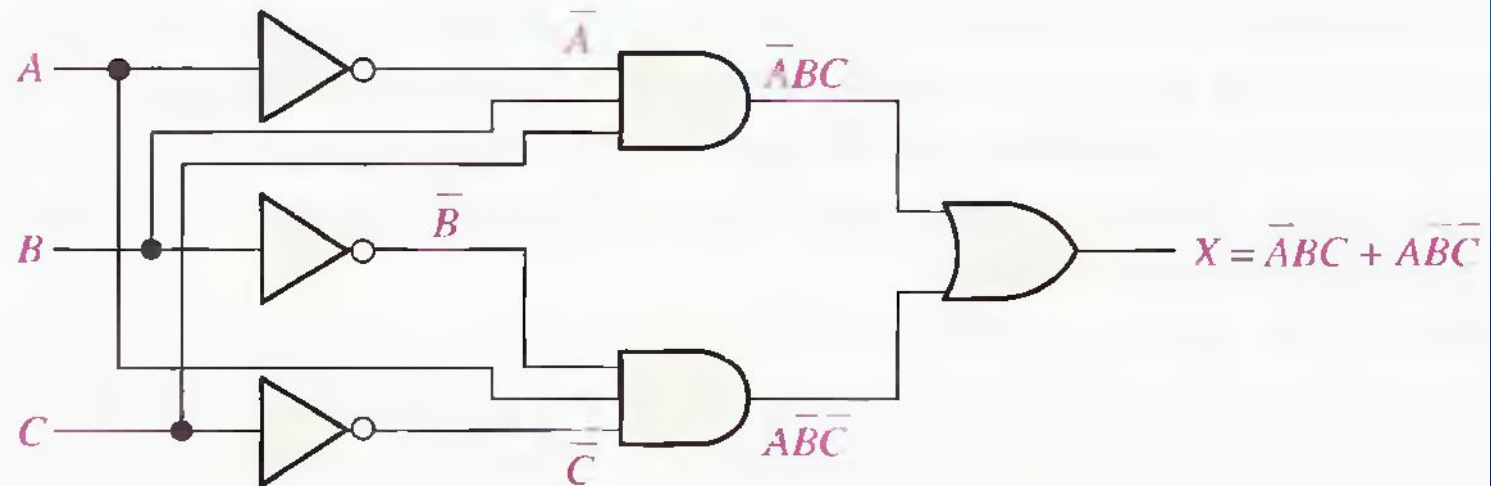
1. One inverter to form \overline{D}
2. Two 2-input AND gates to form \overline{CD} and EF
3. One 2-input OR gate to form $\overline{CD} + EF$
4. One 3-input AND gate to form X



From a Truth Table to a Logic Circuit

$$X = \bar{A}BC + A\bar{B}\bar{C}$$

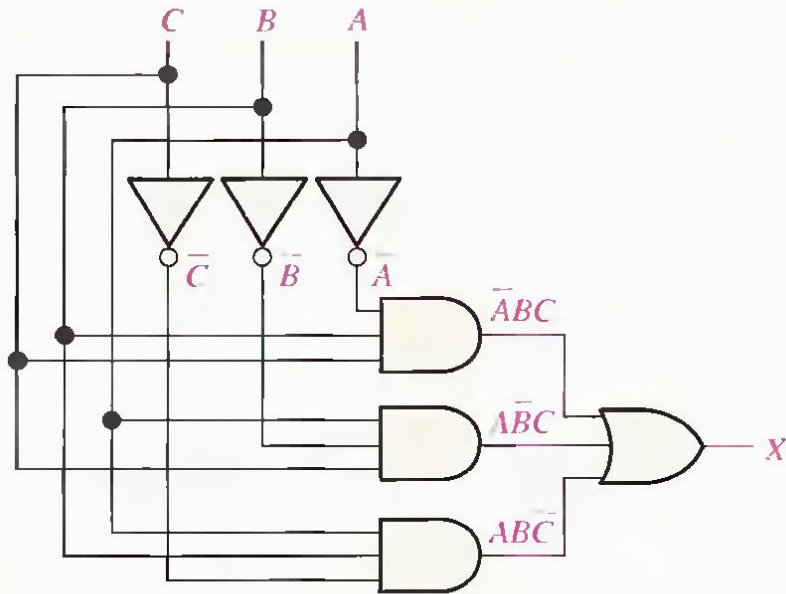
INPUTS			OUTPUT	PRODUCT TERM
A	B	C	X	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}BC$
1	0	0	1	$A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	



EXAMPLE

Design a logic circuit to implement the operation specified in the truth table

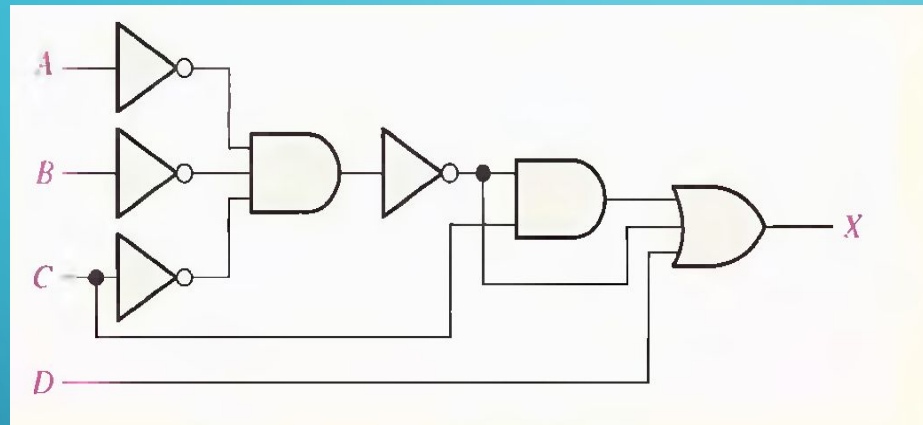
$$X = \bar{A}BC + A\bar{B}C + ABC\bar{C}$$



INPUTS			OUTPUT
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Example

Reduce the combinational logic circuit in Figure

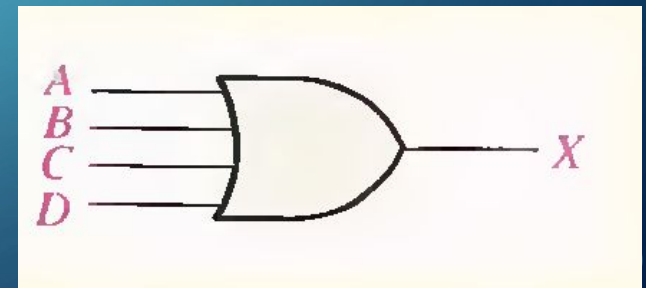


The expression for the output of the circuit is

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

Applying DeMorgan's theorem and Boolean algebra,

$$\begin{aligned} X &= (\overline{\overline{A} + \overline{B} + \overline{C}})C + \overline{\overline{A} + \overline{B} + \overline{C}} + D \\ &= AC + BC + CC + A + B + C + D \\ &= AC + BC + C + A + B + \cancel{C} + D \\ &= C(A + B + 1) + A + B + D \\ X &= A + B + C + D \end{aligned}$$



Example

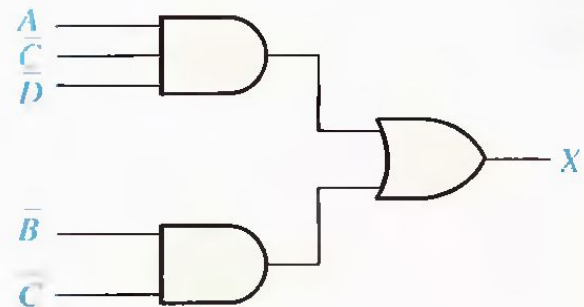
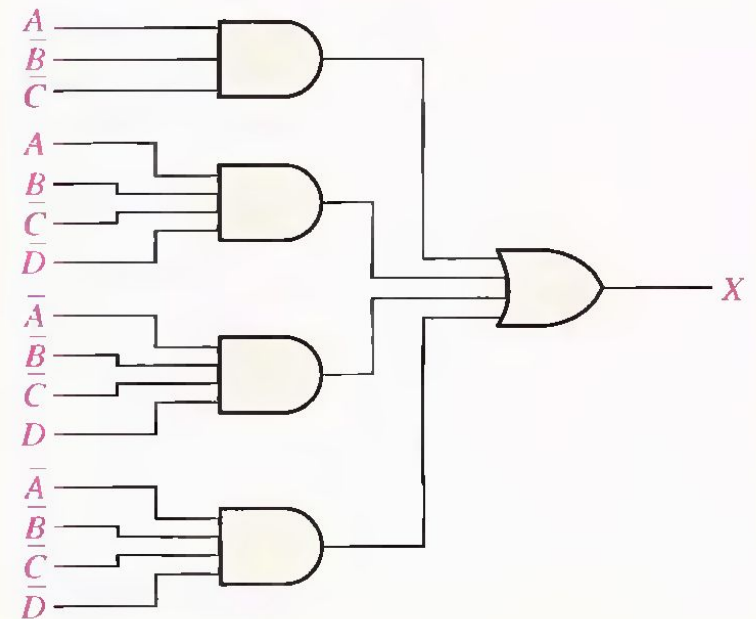
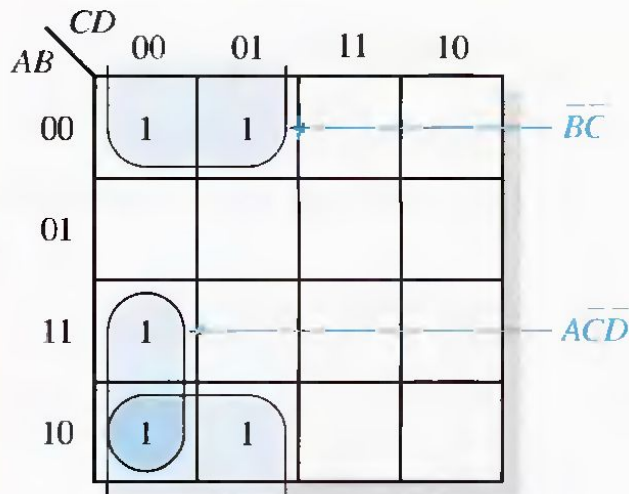
Minimize the combinational logic circuit in Figure .

The output expression is

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

Expanding the first term to include the missing variables D and \overline{D} ,

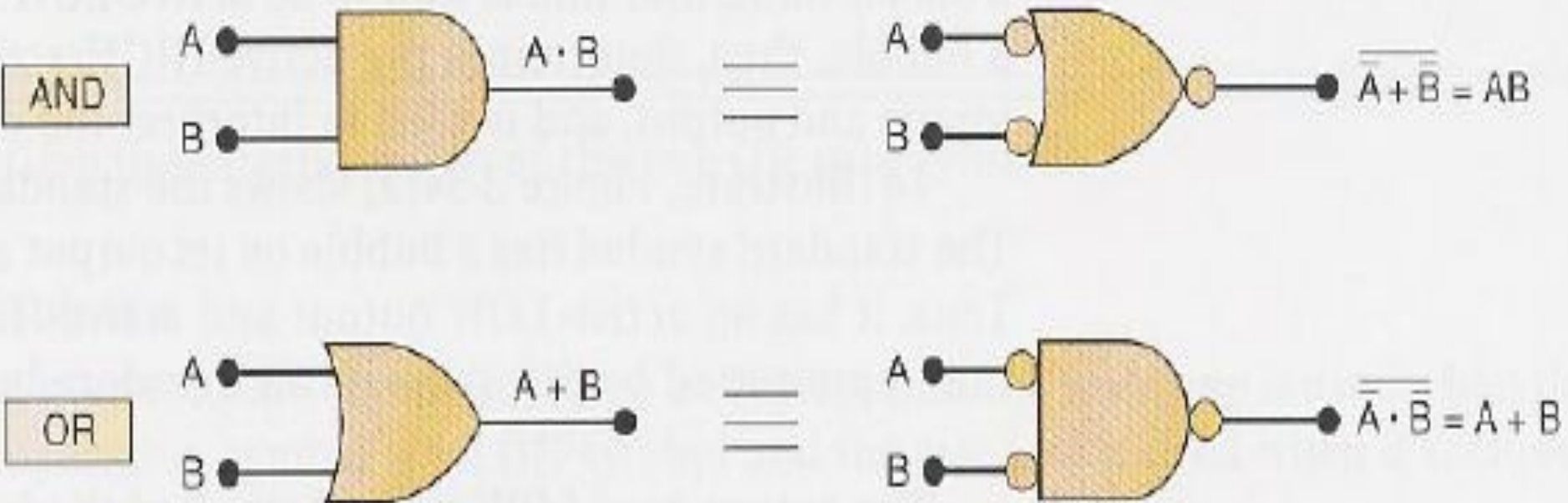
$$\begin{aligned} X &= \overline{A}\overline{B}\overline{C}(D + \overline{D}) + A\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D} \\ &= \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D}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



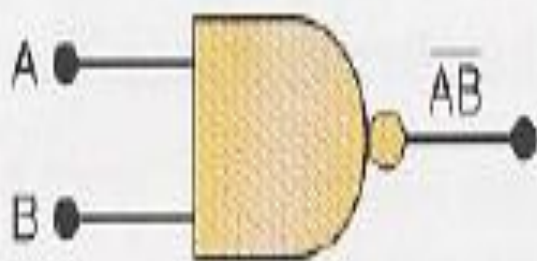
ALTERNATE LOGIC GATE REPRESENTATIONS

The alternate symbol for each gate is obtained from the standard

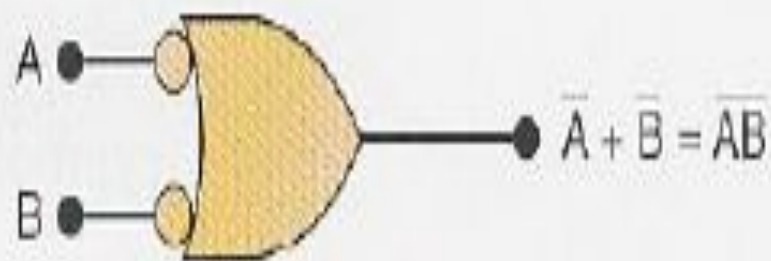
1. Invert each input and output of the standard symbol. This is done by adding bubbles (small circles) on input and output lines that do not have bubbles and by removing bubbles that are already there.
2. Change the operation symbol from AND to OR, or from OR to AND. (In the special case of the INVERTER,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



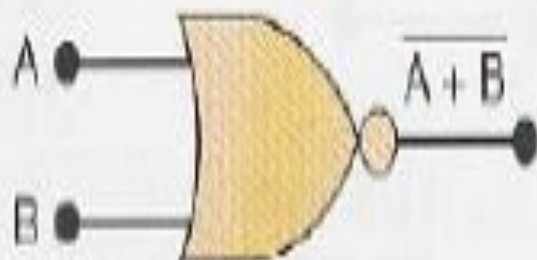
NAND



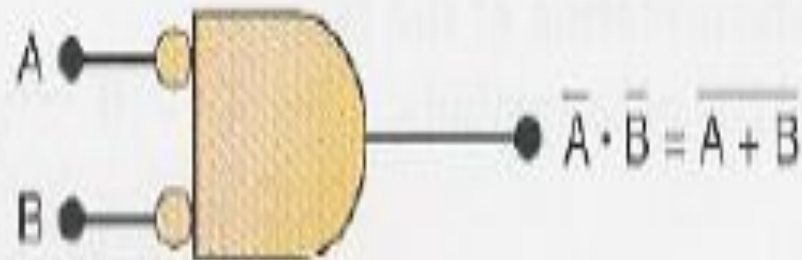
\equiv



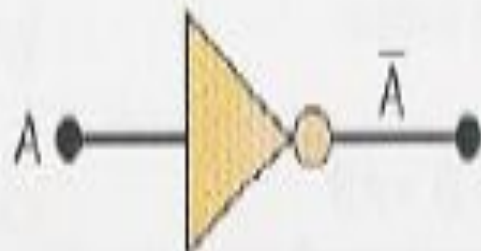
NOR



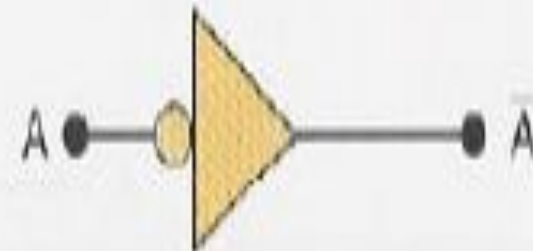
\equiv



INV



\equiv



Several point should be stressed regarding the logic symbol equivalences:

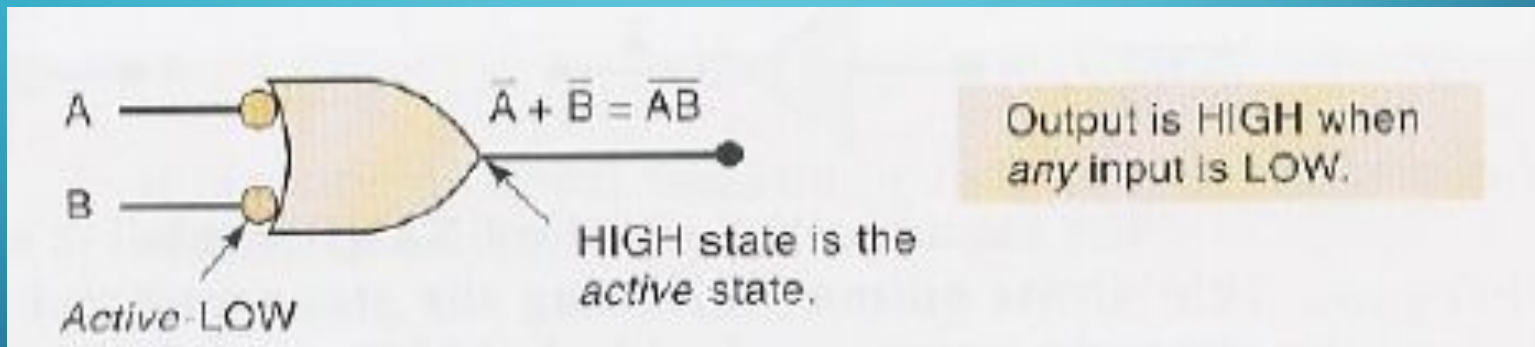
1. The equivalences can be extended to gates with *any* number of inputs.
2. None of the standard symbols have bubbles on their inputs, and all the alternate symbols do.
3. The standard and alternate symbols for each gate represent the same physical circuit; *there is no difference in the circuits represented 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



*Word **all** used because of the AND symbol*

*Word **any** used because of the OR symbol*

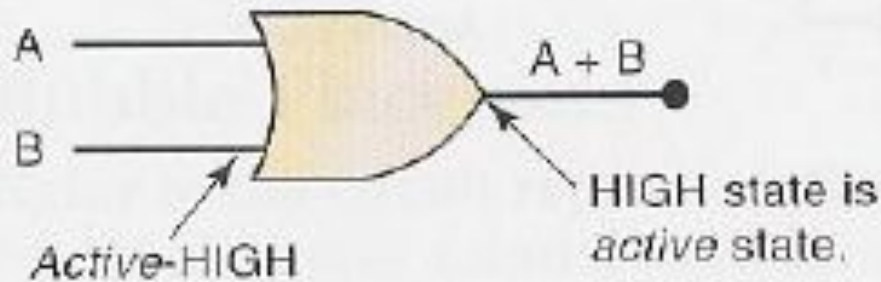
We can see that the two interpretations of the NAND symbols in figure are different ways of saying the same thing.

SUMMARY

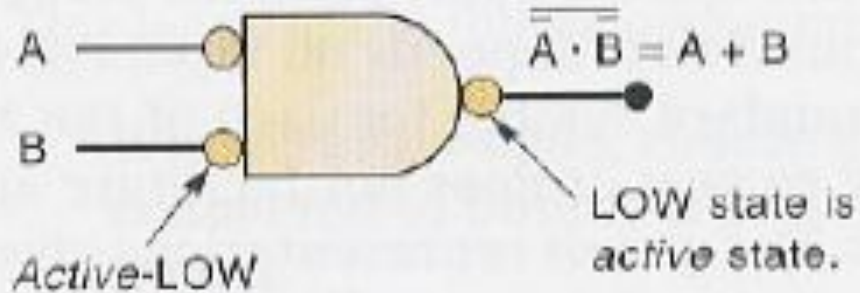
1. To obtain the alternate symbol for a logic gate, take the standard symbol and change its operation symbol (OR to AND, or AND to OR), and change the bubbles on both inputs and output (i.e., delete bubbles that are present, and add bubbles where there are none).
2. To interpret the logic-gate operation, 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.



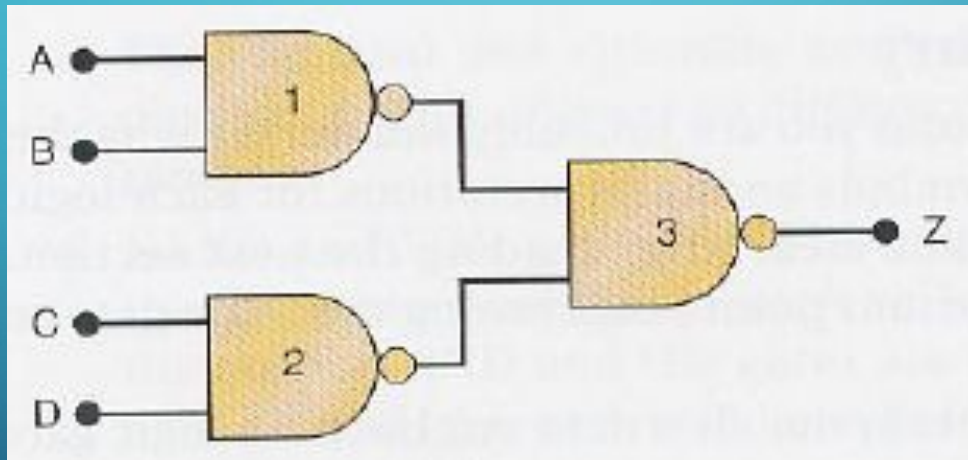
Output goes HIGH when *any* input is HIGH.



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

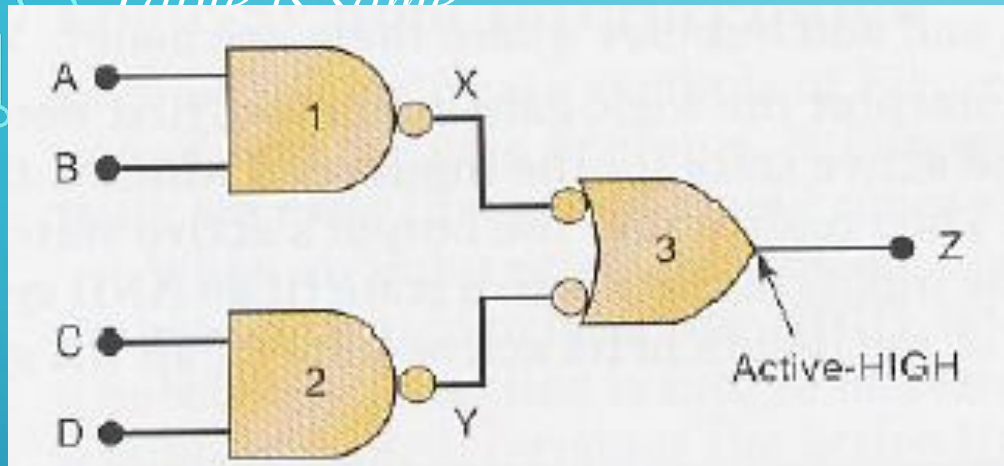


Figure-b

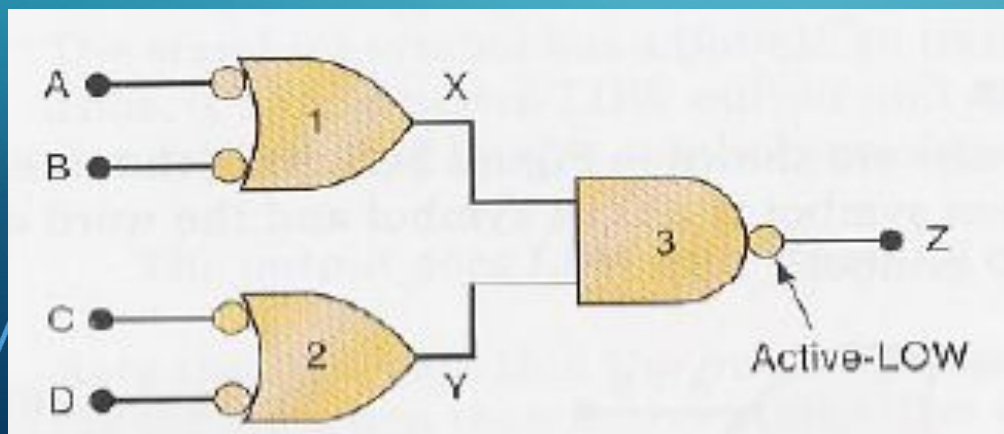


Figure-c

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 if 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 $C=D=1$

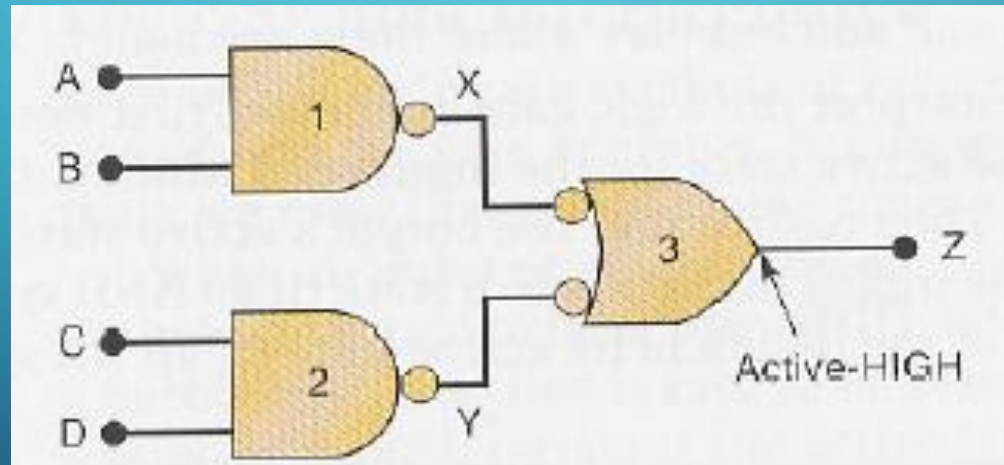


Figure - a

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.

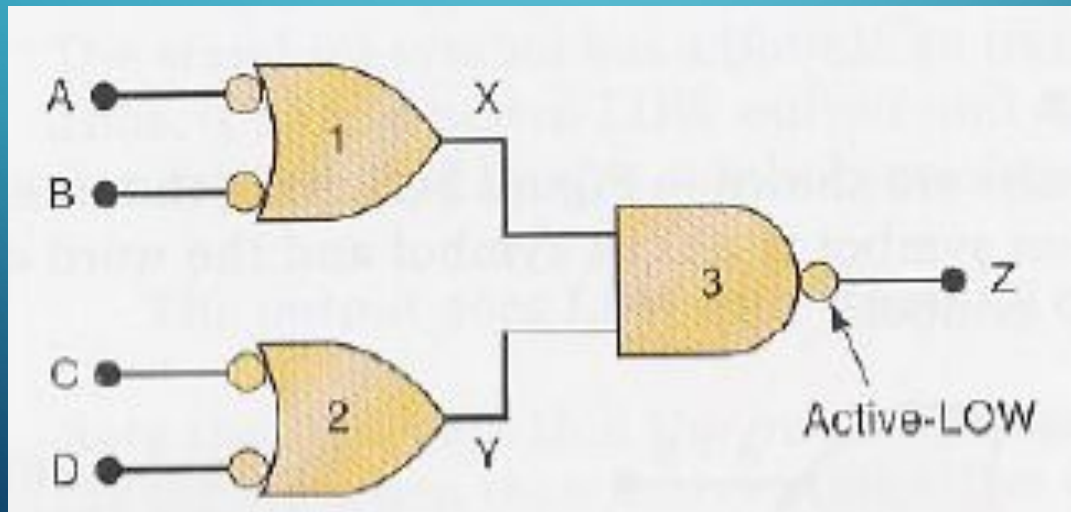
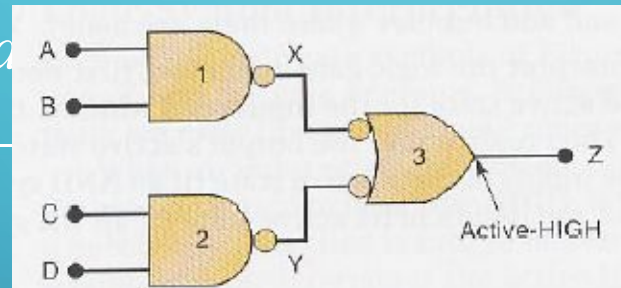


Figure – c

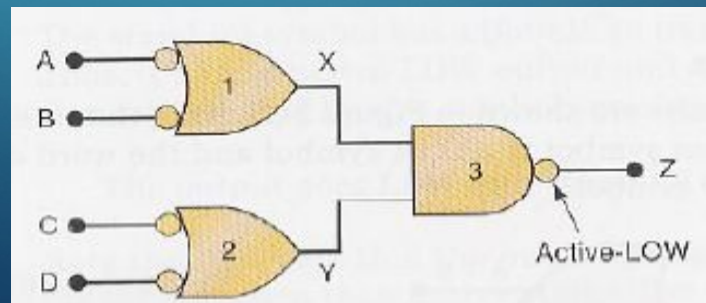
WHICH CIRCUIT DIAGRAM SHOULD BE USED

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 -



On the other hand, when output Z goes to the 0 state then we say that Z is to be active



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.

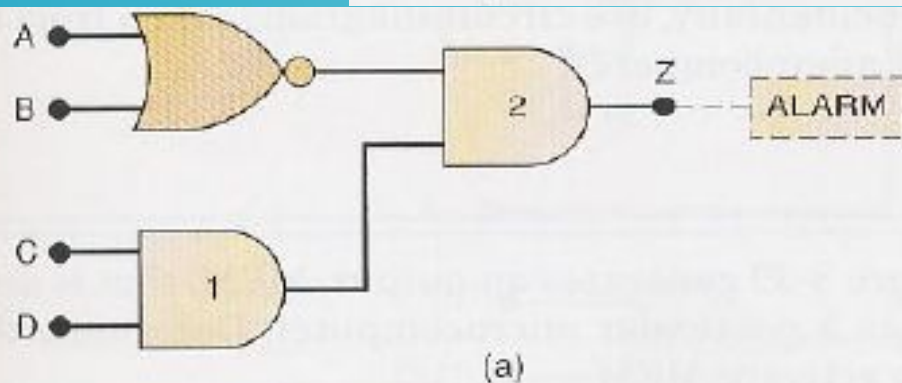
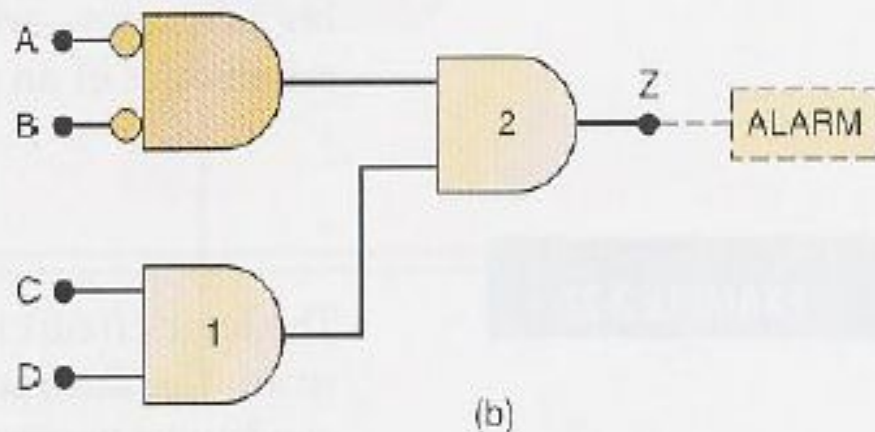


FIGURE 3-37

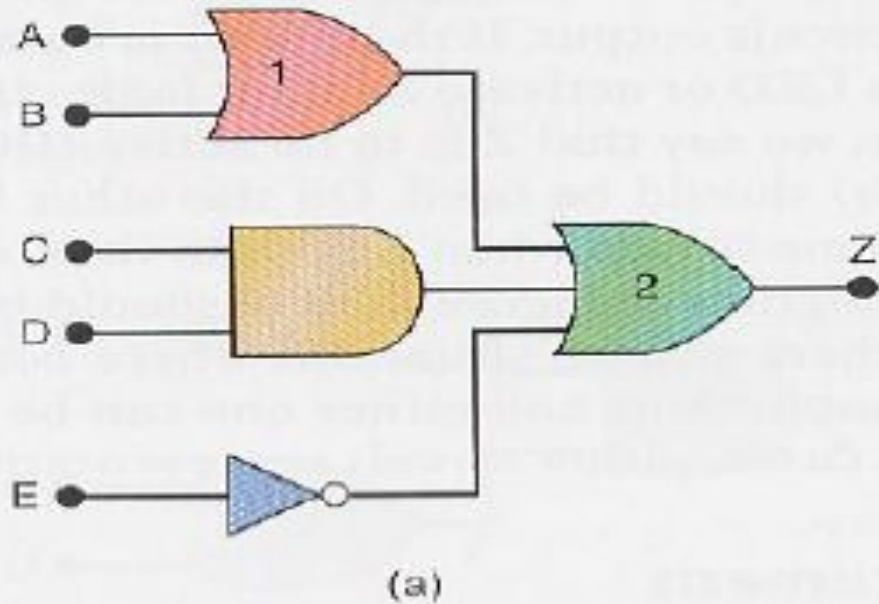


Solution

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.

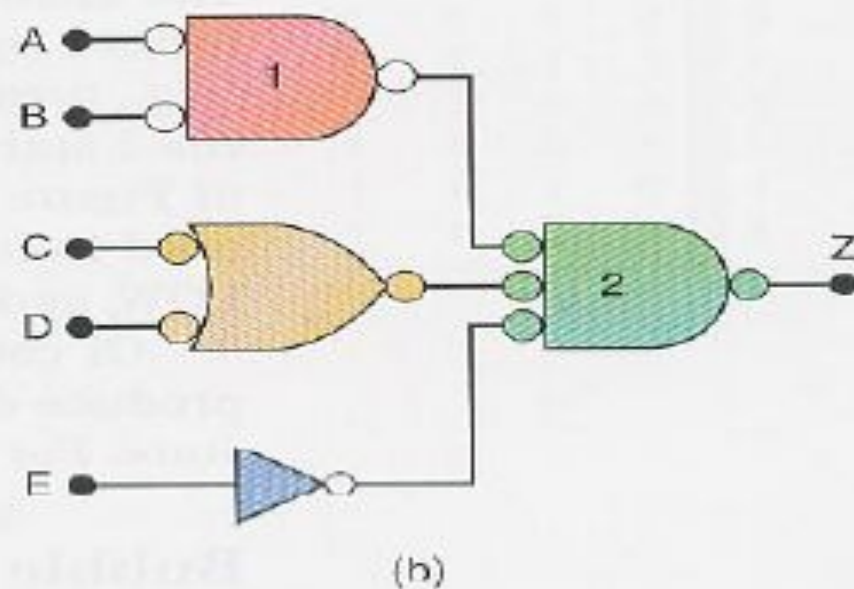
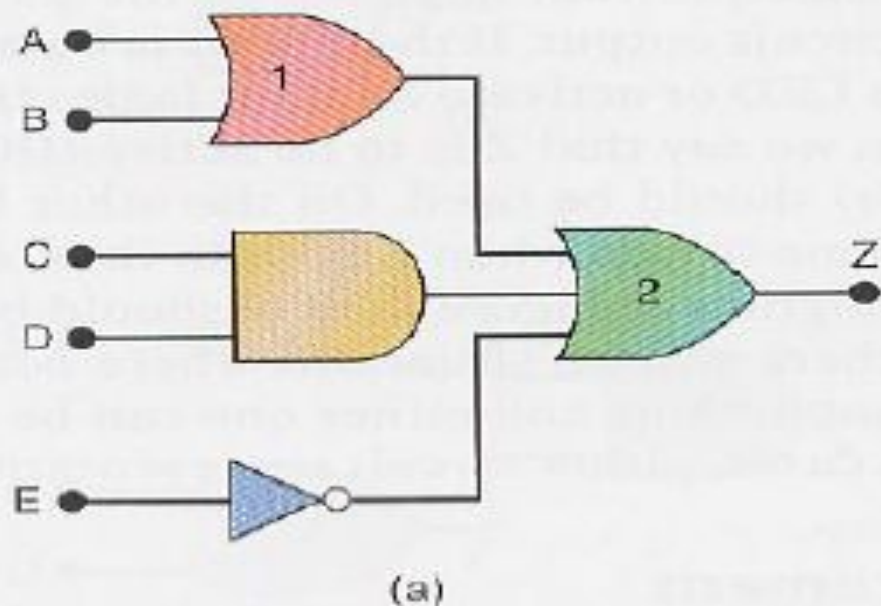
EXAMPLE:

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.



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.



Solution

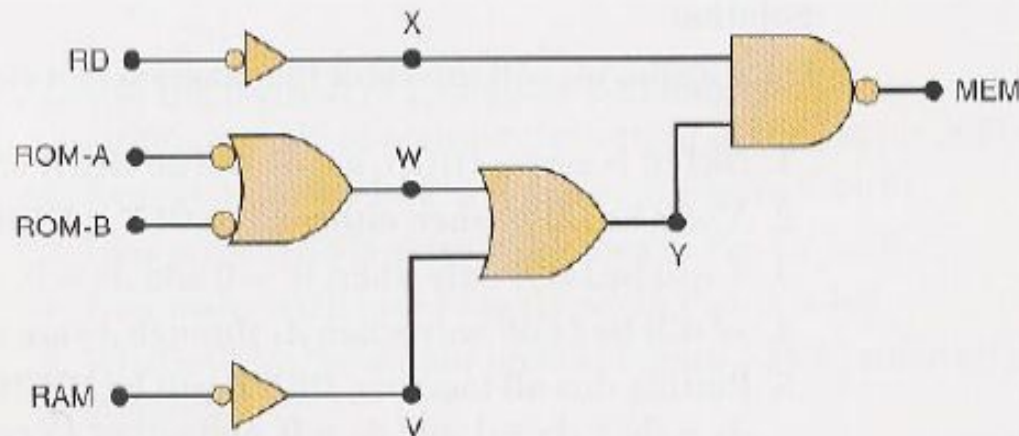
Because Z is to be active-LOW, the symbol for OR gate 2 must be changed to its alternate symbol, as shown in Figure 3-38(b). The new OR gate 2 symbol has bubble inputs, and so the AND gate and OR gate 1 symbols must be changed to bubbled outputs, as shown in Figure 3-38(b). The INVERTER already has a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

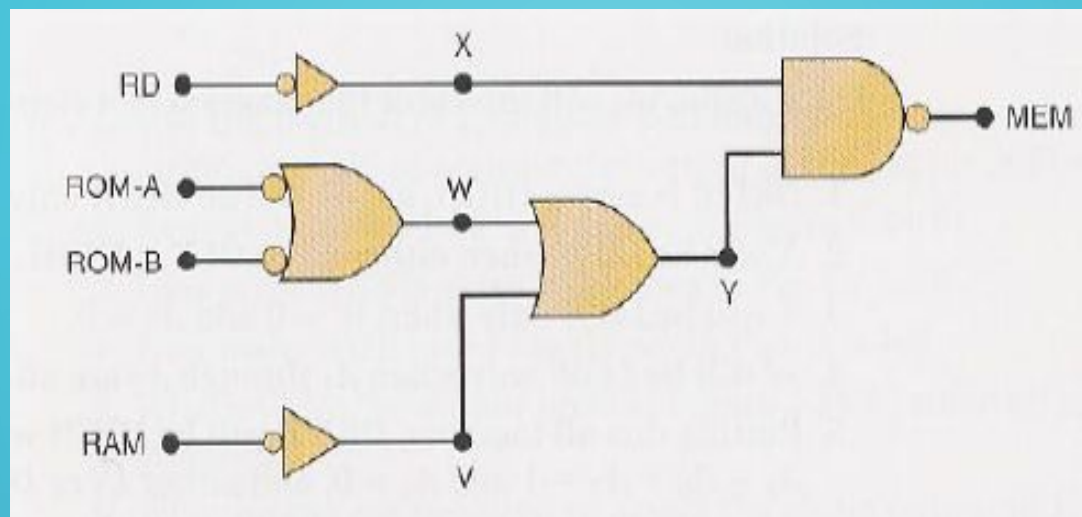
The logic circuit in Figure 3-39 generates an output, *MEM*, that is used to activate the memory ICs in a particular microcomputer. Determine the input conditions necessary to activate *MEM*.



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.

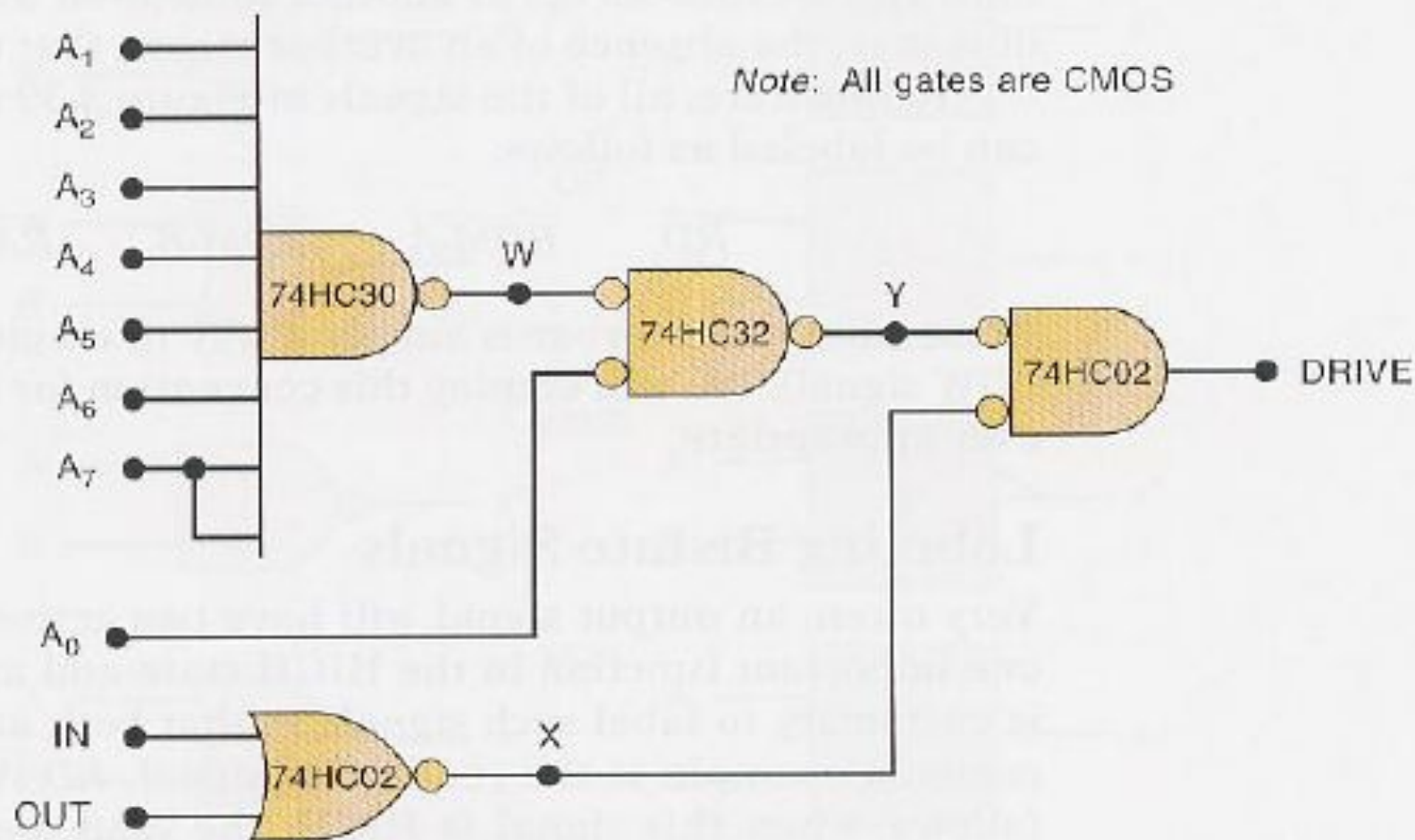
A more efficient method is to interpret the circuit diagram using the ideas we have been developing in the last two sections. These are the steps:



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.

- .Interpret the problem and set up truth tables to describe its operation.*
- .Write the AND(product) term for each case where the output is 1.*
- .Write the SOP expression for the output.*
- .Simplify the output expression if possible.*
- .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 $\rightarrow \bar{A}BC$
1	0	0	0
1	0	1	1 $\rightarrow A\bar{B}C$
1	1	0	1 $\rightarrow AB\bar{C}$
1	1	1	1 $\rightarrow ABC$

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

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

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

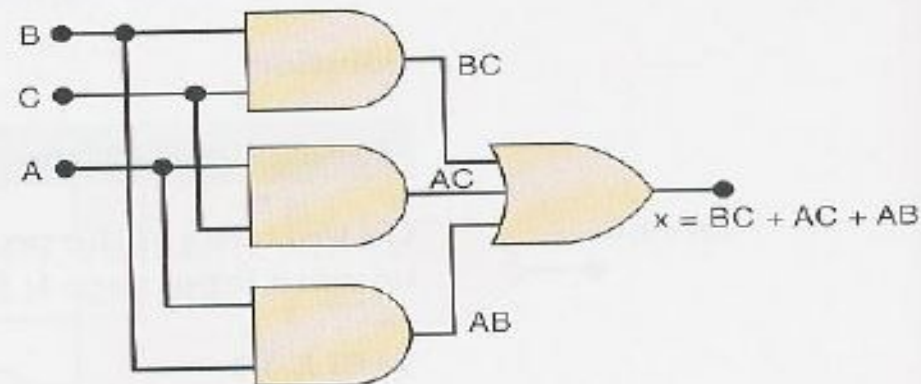
$$x = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

Step 4. Simplify the output expression.

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

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

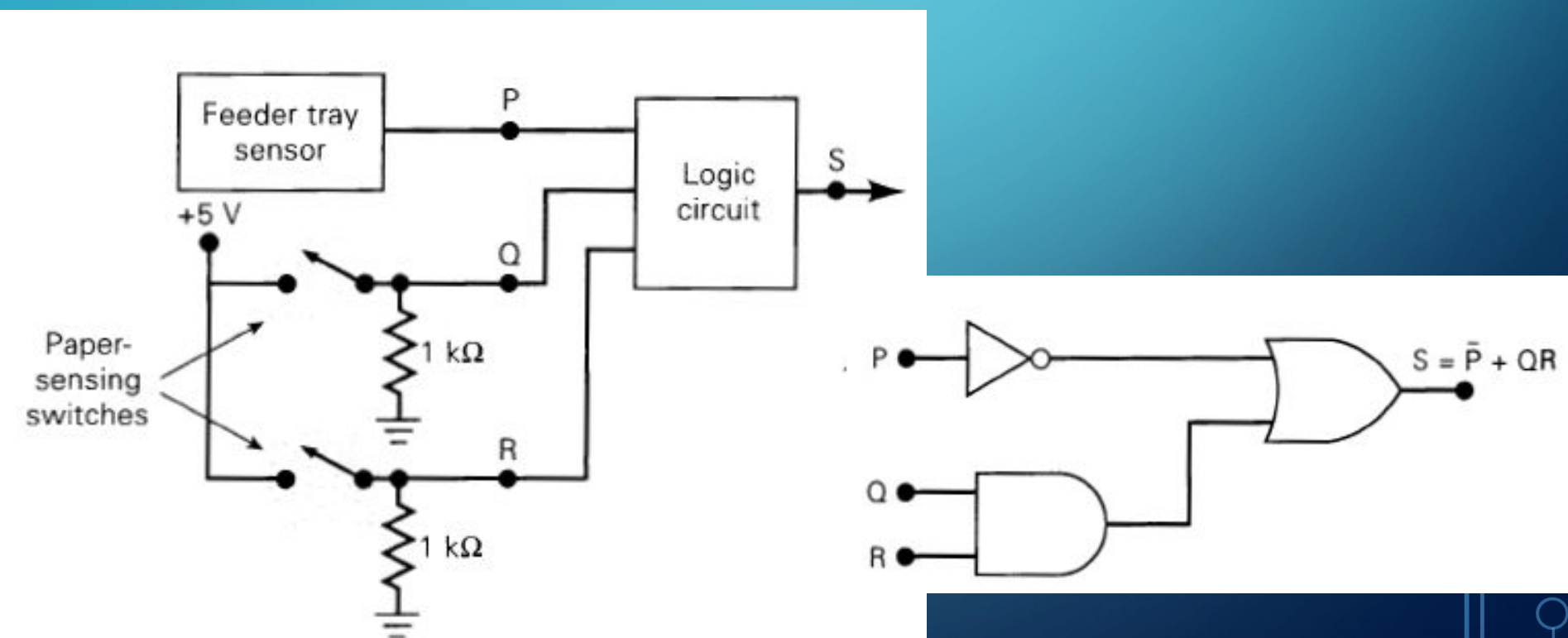
$$x = BC + AC + AB$$



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
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

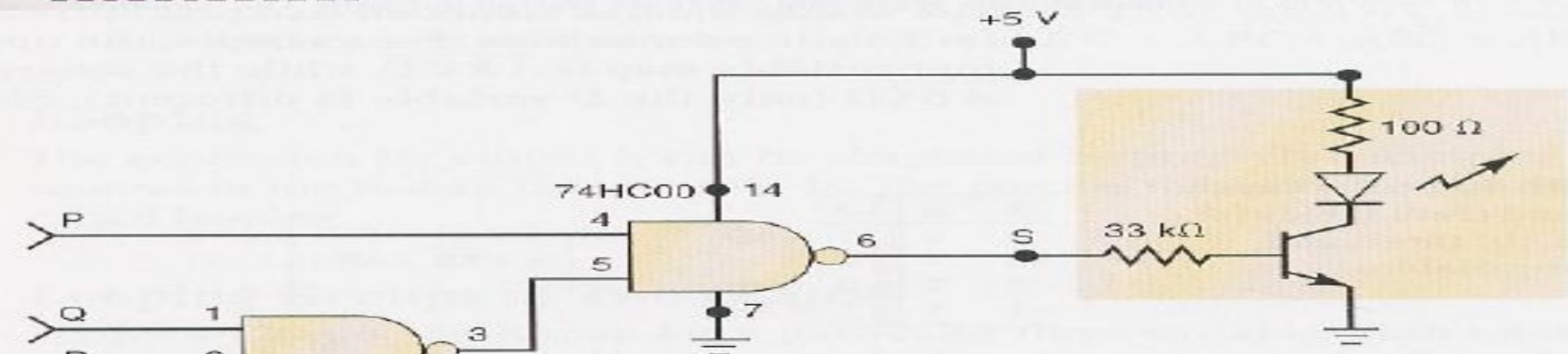
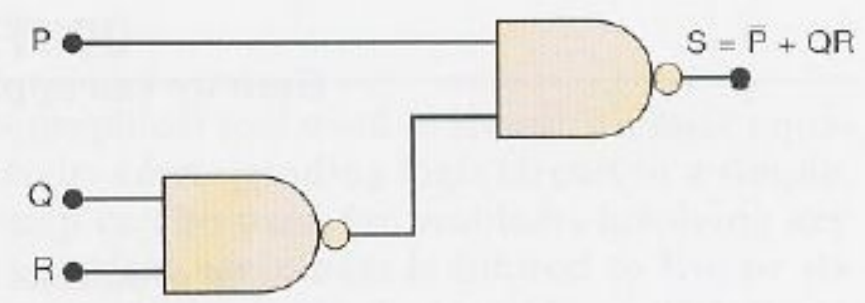
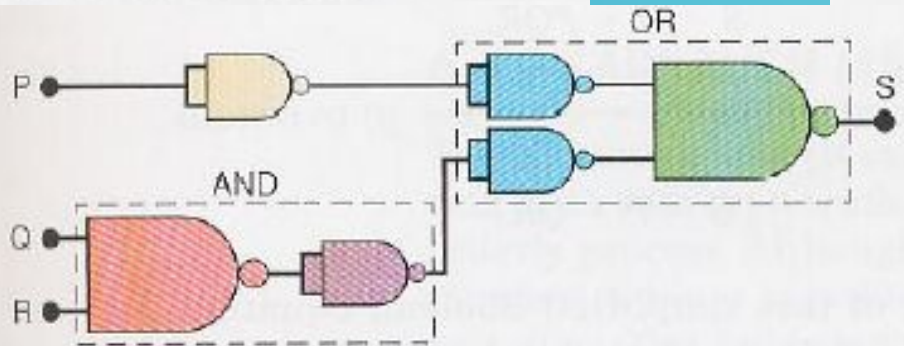
$$S = \overline{P}\overline{Q}\overline{R} + \overline{P}\overline{Q}R + \overline{P}Q\overline{R} + \overline{P}QR + PQR$$

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

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

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

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



Note: The other two gates on the chip

EXAMPLE

Refer to Figure 4-8(a), where an analog-to-digital converter is monitoring the dc voltage of a 12-V storage battery on an orbiting spaceship. The converter's output is a four-bit binary number, $ABCD$, corresponding to the battery voltage in steps of 1 V, with A as the MSB. The converter's binary outputs are fed to a logic circuit that is to produce a HIGH output as long as the binary value is greater than $0110_2 = 6_{10}$; that is, the battery voltage is greater than 6 V. Design this logic circuit.

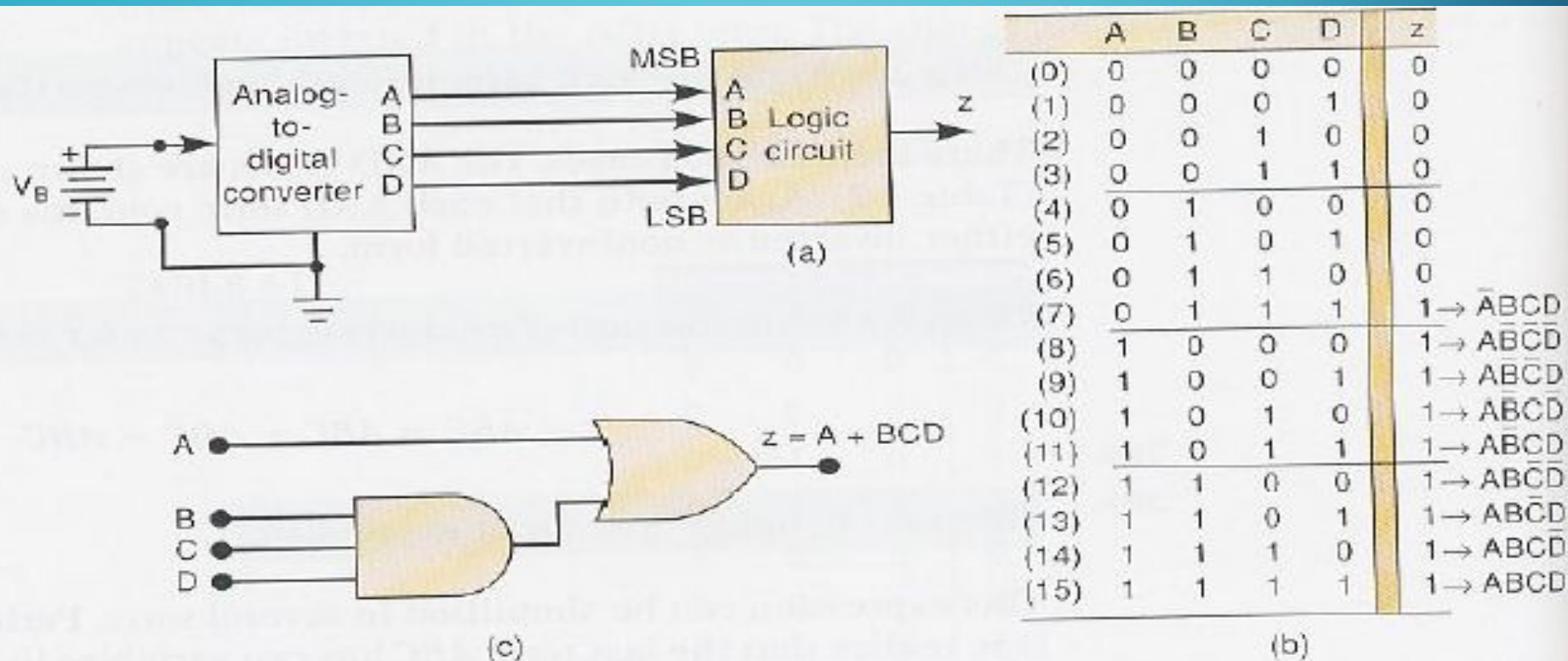


FIGURE 4-8 Example 4-8.

Solution

The truth table is shown in Figure 4-8(b). For each case in the truth table, we have indicated the decimal equivalent of the binary number represented by the $ABCD$ combination.

The output z is set equal to 1 for all those cases where the binary number is greater than 0110. For all other cases, z is set equal to 0. This truth table gives us the following sum-of-products expression:

$$z = \overline{A}BCD + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + A\overline{B}\overline{C}\overline{D} \\ + A\overline{B}C\overline{D} + ABC\overline{D} + ABCD$$

$$\begin{aligned} z &= \overline{A}BCD + \overline{A}\overline{B}\overline{C}(\overline{D} + D) + \overline{A}\overline{B}\overline{C}(\overline{D} + D) + \overline{A}\overline{B}C(\overline{D} + D) + A\overline{B}\overline{C}(\overline{D} + D) \\ &= \overline{A}BCD + \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}\overline{B}C + ABC \\ &= \overline{A}BCD + \overline{A}\overline{B}(\overline{C} + C) + AB(\overline{C} + C) \\ &= \overline{A}BCD + \overline{A}\overline{B} + AB \\ &= \overline{A}BCD + A(\overline{B} + B) \\ &= \overline{A}BCD + A \end{aligned}$$

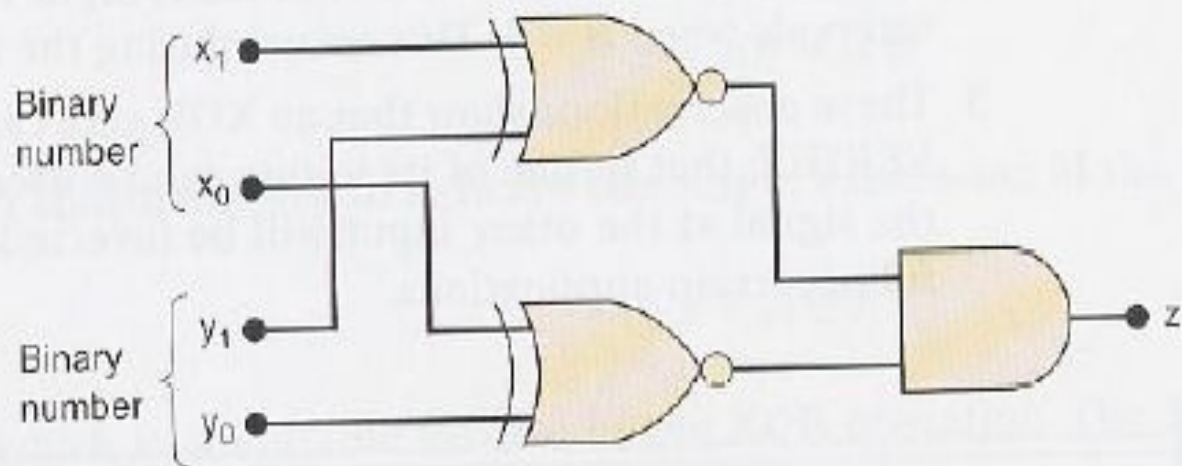
$$z = \overline{A}BCD + A = BCD + A$$

EXAMPLE

The notation x_1x_0 represents a two-bit binary number that can have any value (00, 01, 10, or 11); for example, when $x_1 = 1$ and $x_0 = 0$, the binary number is 10, and so on. Similarly, y_1y_0 represents another two-bit binary number. Design a logic circuit, using x_1 , x_0 , y_1 , and y_0 inputs, whose output will be **HIGH** only when the two binary numbers x_1x_0 and y_1y_0 are *equal*.

The first step is to construct a truth table for the 16 input conditions (Table 4-4). The output z must be **HIGH** whenever the x_1x_0 values match the y_1y_0 values; that is, whenever $x_1 = y_1$ and $x_0 = y_0$. The table shows that there are four such cases. We could now continue with the normal procedure, which would be to obtain a sum-of-products expression for z , attempt to simplify it, and then implement the result. However, the nature of this problem makes it ideally suited for implementation using XNOR gates, and a little thought will produce a simple solution with minimum work. Refer to Figure 4-23; in this logic diagram, x_1 and y_1 are fed to one XNOR gate, and x_0 and y_0 are fed to another XNOR gate. The output of each XNOR will be **HIGH** only when its inputs are equal. Thus, for $x_0 = y_0$ and $x_1 = y_1$, both XNOR outputs will be **HIGH**. This is the condition we are looking for because it means that the two two-bit numbers are equal. The AND gate output will be **HIGH** only for this case, thereby producing the desired output.

x_1	x_0	y_1	y_0	z (Output)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



EXAMPLE

When simplifying the expression for the output of a combinational logic circuit, you may encounter the XOR or XNOR operations as you are factoring. This will often lead to the use of XOR or XNOR gates in the implementation of the final circuit. To illustrate, simplify the circuit of Figure 4-24(a).

Solution

The unsimplified expression for the circuit is obtained as

$$z = ABCD + A\bar{B}\bar{C}D + \bar{A}\bar{D}$$

We can factor AD from the first two terms:

$$z = AD(BC + \bar{B}\bar{C}) + \bar{A}\bar{D}$$

