Digital Logic & Design (Theory) Lecture No.7

LECTURE OUTLINE

5–1 Basic Combinational Logic Circuits

You have learned that SOP expressions are implemented with an AND gate for each product term and one OR gate for summing all of the product terms. As you know, this SOP implementation is called AND-OR logic and is the basic form for realizing standard Boolean functions. In this section, the AND-OR and the AND-OR-Invert are examined; the exclusive-OR and exclusive-NOR gates, which are actually a form of AND-OR logic, are also covered.

AND-OR Logic

Figure 5–1 shows an AND-OR circuit consisting of two 2-input AND gates and one 2-input OR gate. The Boolean expressions for the AND gate outputs and the resulting SOP expression for the output *X* are shown on the diagram. In general, an AND-OR circuit can have any number of AND gates, each with any number of inputs.

The truth table for a 4-input AND-OR logic circuit is shown in Table 5–1. The intermediate AND gate outputs (the *AB* and *CD* columns) are also shown in the table.

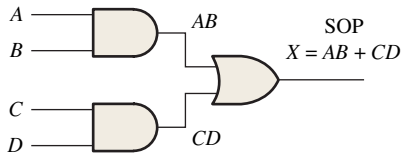


FIGURE 5–1 An example of AND-OR logic. Open file F05-01 to verify the operation. *A Multisim tutorial is available on the website.*

An AND-OR circuit directly implements an SOP expression, assuming the complements (if any) of the variables are available. The operation of the AND-OR circuit in Figure 5–1 is stated as follows:

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).

TABLE 5–1

Truth table for the AND-OR logic in Figure 5–1.

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. Recall that AND-OR logic directly implements SOP expressions. POS expressions can be implemented with AND-OR-Invert logic. This is illustrated as follows, starting with a POS expression and developing the corresponding AND-OR-Invert (AOI) expression.

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

The logic diagram in Figure 5–3 shows an AND-OR-Invert circuit with four inputs and the development of the POS output expression. In general, an AND-OR-Invert circuit can have any number of AND gates, each with any number of inputs.

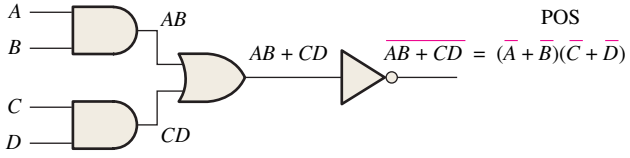


FIGURE 5–3 An AND-OR-Invert circuit produces a POS output. Open file F05-03 to verify the operation.

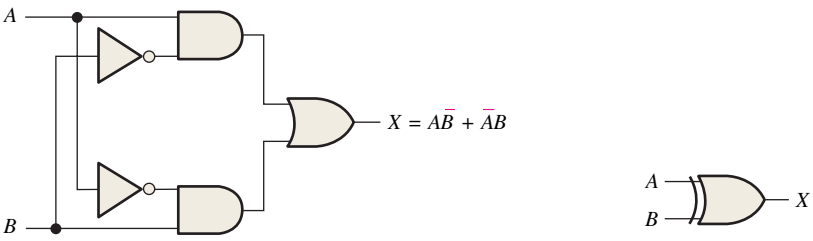
The operation of the AND-OR-Invert circuit in Figure 5–3 is stated as follows:

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).

A truth table can be developed from the AND-OR truth table in Table 5–1 by simply changing all 1s to 0s and all 0s to 1s in the output column.

Exclusive-OR Logic

The exclusive-OR gate was introduced in Chapter 3. Although this circuit is considered a type of logic gate with its own unique symbol, it is actually a combination of two AND gates, one OR gate, and two inverters, as shown in Figure 5–5(a). The ANSI standard exclusive-OR logic symbol is shown in part (b).



(a) Logic diagram

(b) ANSI distinctive shape symbol

FIGURE 5–5 Exclusive-OR logic diagram and symbols.

The output expression for the circuit in Figure 5–5 is

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

Evaluation of this expression results in the truth table in Table 5–2. Notice that the output is HIGH only when the two inputs are at opposite levels. A special exclusive-OR operator $\oplus$ is often used, so the expression $X = A\overline{B} + \overline{A}B$ can be stated as "X is equal to A exclusive-OR B" and can be written as

$$X = A \oplus B$$

TABLE 5–2
Truth table for an exclusive-OR.

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

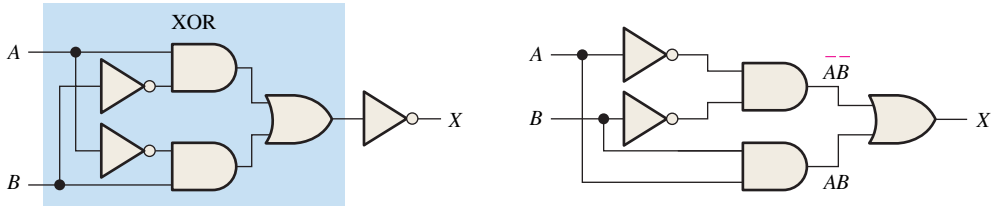
Exclusive-NOR Logic

As you know, the complement of the exclusive-OR function is the exclusive-NOR, which is derived as follows:

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

Notice that the output X is HIGH only when the two inputs, A and B, are at the same level.

The exclusive-NOR can be implemented by simply inverting the output of an exclusive-OR, as shown in Figure 5–6(a), or by directly implementing the expression $\overline{A}\,\overline{B} + AB$, as shown in part (b).



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

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

FIGURE 5–6 Two equivalent ways of implementing the exclusive-NOR.

EXAMPLE 5–3

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.

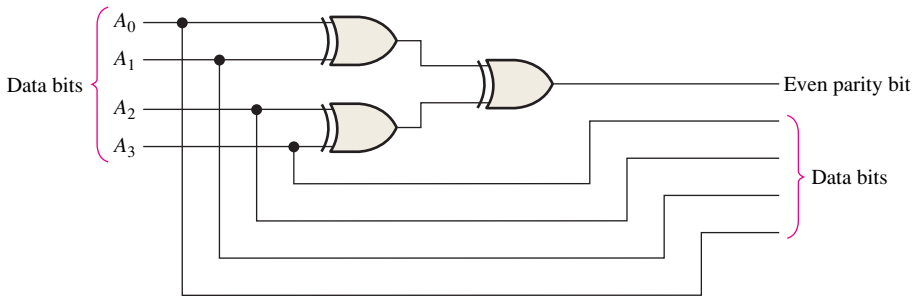


FIGURE 5–7 Even-parity generator.

EXAMPLE 5–4

Use exlusive-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.

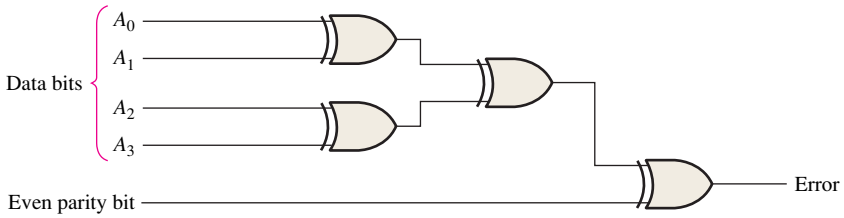


FIGURE 5–8 Even-parity checker.

5–2 Implementing Combinational Logic

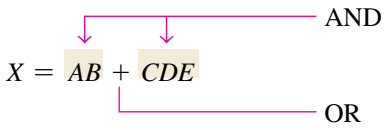
In this section, examples are used to illustrate how to implement a logic circuit from a Boolean expression or a truth table. Minimization of a logic circuit using the methods covered in Chapter 4 is also included.

From a Boolean Expression to a Logic Circuit

Let's examine the following Boolean expression:

$$X = AB + CDE$$

A brief inspection shows that this expression is composed of two terms, AB and CDE, with a domain of five variables. The first term is formed by ANDing A with B, and the second term is formed by ANDing C, D, and E. The two terms are then ORed to form the output X. These operations are indicated in the structure of the expression as follows:



Note that in this particular expression, the AND operations forming the two individual terms, AB and CDE, must be performed *before* the terms can be ORed.

To implement this Boolean expression, a 2-input AND gate is required to form the term AB, and a 3-input AND gate is needed to form the term CDE. A 2-input OR gate is then required to combine the two AND terms. The resulting logic circuit is shown in Figure 5–9.

As another example, let's implement the following expression:

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

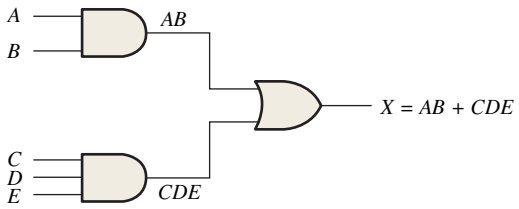
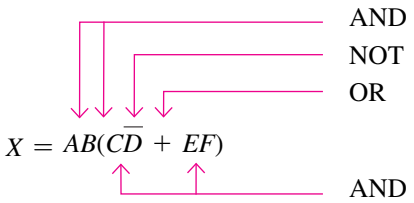


FIGURE 5–9 Logic circuit for $X = AB + CDE$.

A breakdown of this expression shows that the terms AB and $(C\overline{D} + EF)$ are ANDed. The term $C\overline{D} + EF$ is formed by first ANDing C and $\overline{D}$ and ANDing E and F, and then ORing these two terms. This structure is indicated in relation to the expression as follows:



Before you can implement the final expression, you must create the sum term $C\overline{D} + EF$; but before you can get this term; you must create the product terms $C\overline{D}$ and EF; but before you can get the term $C\overline{D}$, you must create $\overline{D}$. So, as you can see, the logic operations must be done in the proper order.

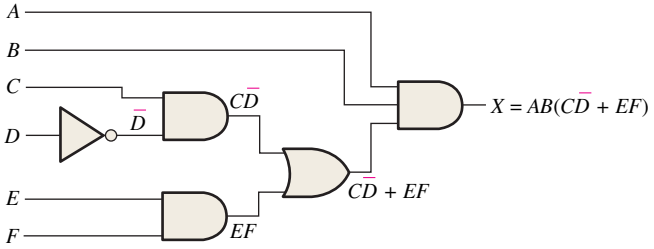
The logic gates required to implement $X = AB(C\overline{D} + EF)$ are as follows:

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

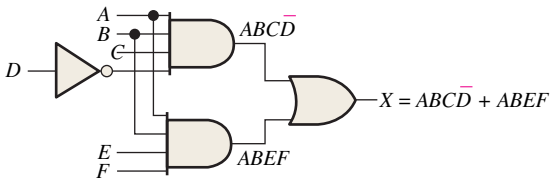
The logic circuit for this expression is shown in Figure 5–10(a). Notice that there is a maximum of four gates and an inverter between an input and output in this circuit (from input D to output). Often the total propagation delay time through a logic circuit is a major consideration. Propagation delays are additive, so the more gates or inverters between input and output, the greater the propagation delay time.

Unless an intermediate term, such as $C\overline{D} + EF$ in Figure 5–10(a), is required as an output for some other purpose, it is usually best to reduce a circuit to its SOP form in order to reduce the overall propagation delay time. The expression is converted to SOP as follows, and the resulting circuit is shown in Figure 5–10(b).

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



(a)



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

FIGURE 5–10 Logic circuits for $X = AB(C\overline{D} + EF) = ABC\overline{D} + ABEF$.

TABLE 5–3

Inputs			Output	
A	*B*	*C*	*X*	Product Term
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	

From a Truth Table to a Logic Circuit

If you begin with a truth table instead of an expression, you can write the SOP expression from the truth table and then implement the logic circuit. Table 5–3 specifies a logic function.

The Boolean SOP expression obtained from the truth table by ORing the product terms for which $X = 1$ is

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

The first term in the expression is formed by ANDing the three variables $\overline{A}$, B, and C. The second term is formed by ANDing the three variables A, $\overline{B}$, and $\overline{C}$.

The logic gates required to implement this expression are as follows: three inverters to form the $\overline{A}$, $\overline{B}$, and $\overline{C}$ variables; two 3-input AND gates to form the terms $\overline{A}BC$ and $A\overline{B}\,C$; and one 2-input OR gate to form the final output function, $\overline{A}BC + A\overline{B}\,\overline{C}$.

The implementation of this logic function is illustrated in Figure 5–11.

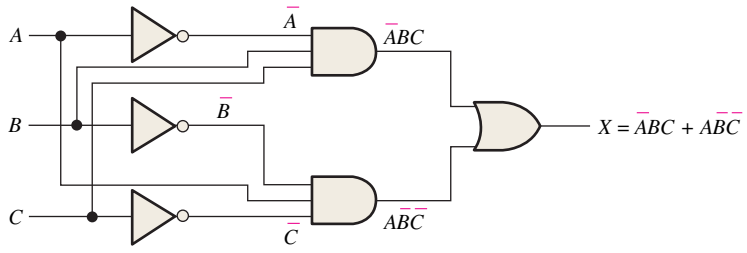


FIGURE 5–11 Logic circuit for $X = \overline{A}BC + A\overline{B}\,\overline{C}$. Open file F05-11 to verify the operation.

EXAMPLE 5–5

Design a logic circuit to implement the operation specified in the truth table of Table 5–4.

TABLE 5–4

Inputs			Output	
A	*B*	*C*	*X*	Product Term
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	0	

Solution

Notice that $X = 1$ for only three of the input conditions. Therefore, the logic expression is

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

The logic gates required are three inverters, three 3-input AND gates and one 3-input OR gate. The logic circuit is shown in Figure 5–12.

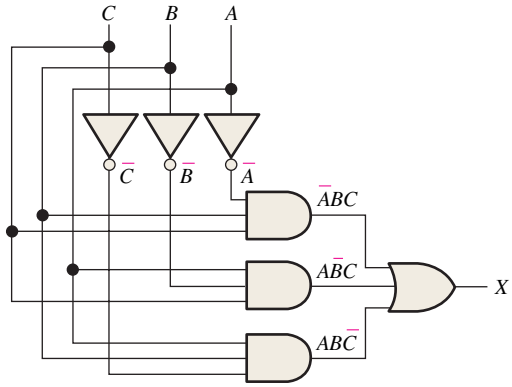


FIGURE 5–12 Open file F05-12 to verify the operation.

EXAMPLE 5–6

Develop a logic circuit with four input variables that will only produce a 1 output when exactly three input variables are 1s.

Solution

Out of sixteen possible combinations of four variables, the combinations in which there are exactly three 1s are listed in Table 5–5, along with the corresponding product term for each.

TABLE 5–5

A	*B*	*C*	*D*	Product Term
0	1	1	1	$\overline{A}BCD$
1	0	1	1	$A\overline{B}CD$
1	1	0	1	$AB\overline{C}D$
1	1	1	0	$ABC\overline{D}$

The product terms are ORed to get the following expression:

$$X = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D}$$

This expression is implemented in Figure 5–13 with AND-OR logic.

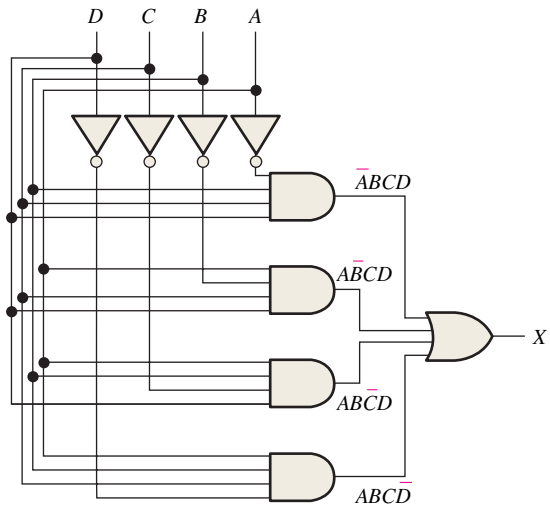


FIGURE 5–13 Open file F05-13 to verify the operation.

EXAMPLE 5–7

Reduce the combinational logic circuit in Figure 5–14 to a minimum form.

Solution

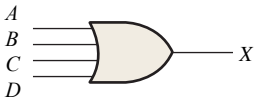
The expression for the output of the circuit is

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

Applying DeMorgan's theorem and Boolean algebra,

$$\begin{aligned} X &= (\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}})C + \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} + D \\ &= AC + BC + CC + A + B + C + D \\ &= AC + BC + C + A + B + \not{C} + D \\ &= C(A + B + 1) + A + B + D \\ X &= A + B + C + D \end{aligned}$$

The simplified circuit is a 4-input OR gate as shown in Figure 5–15.



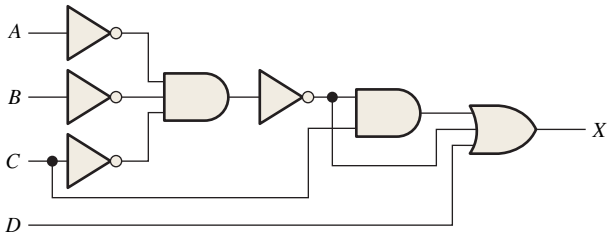


FIGURE 5–14
Open file F05-14 to verify that this circuit is equivalent to the gate in Figure 5–15.

EXAMPLE 5–8

Minimize the combinational logic circuit in Figure 5–16. Inverters for the complemented variables are not shown.

Solution

The output expression is

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

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

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

This expanded SOP expression is mapped and simplified on the Karnaugh map in Fig- ure 5–17(a). The simplified implementation is shown in part (b). Inverters are not shown.

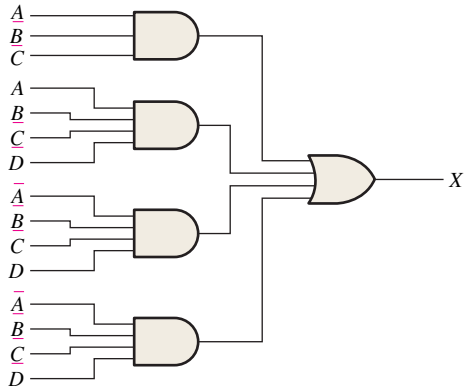
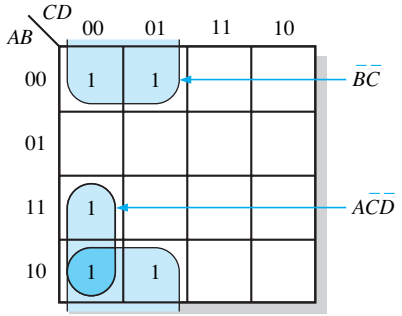
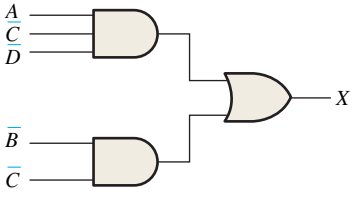


FIGURE 5–16



(a)



(b)

FIGURE 5–17

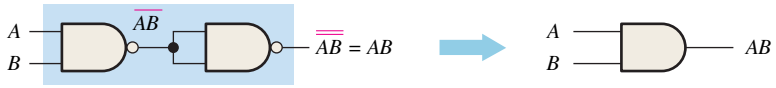
5–3 The Universal Property of NAND and NOR Gates

Up to this point, you have studied combinational circuits implemented with AND gates, OR gates, and inverters. In this section, the universal property of the NAND gate and the NOR gate is discussed. The universality of the NAND gate means that it can be used as an inverter and that combinations of NAND gates can be used to implement the AND, OR, and NOR operations. Similarly, the NOR gate can be used to implement the inverter (NOT), AND, OR, and NAND operations.

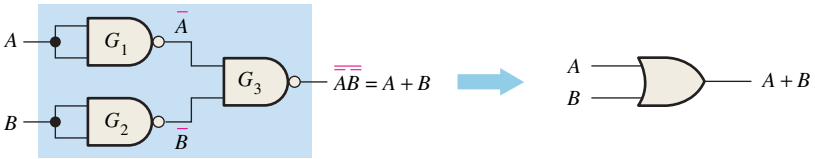
The NAND Gate as a Universal Logic Element

The NAND gate is a **universal gate** because it can be used to produce the NOT, the AND, the OR, and the NOR functions. An inverter can be made from a NAND gate by connecting all of the inputs together and creating, in effect, a single input, as shown in Figure 5–18(a) for a 2-input gate. An AND function can be generated by the use of NAND gates alone, as shown in Figure 5–18(b). An OR function can be produced with only NAND gates, as illustrated in part (c). Finally, a NOR function is produced as shown in part (d).

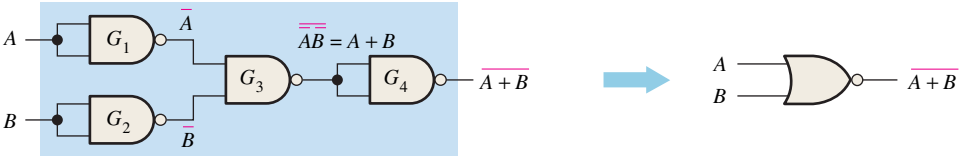
(a) One NAND gate used as an inverter



(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

FIGURE 5–18 Universal application of NAND gates. Open files F05-18(a), (b), (c), and (d) to verify each of the equivalencies.

In Figure 5–18(b), a NAND gate is used to invert (complement) a NAND output to form the AND function, as indicated in the following equation:

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

In Figure 5–18(c), NAND gates G_1 and G_2 are used to invert the two input variables before they are applied to NAND gate G_3. The final OR output is derived as follows by application of DeMorgan's theorem:

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

In Figure 5–18(d), NAND gate G_4 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

Like the NAND gate, the NOR gate can be used to produce the NOT, AND, OR, and NAND functions. A NOT circuit, or inverter, can be made from a NOR gate by connecting all of the inputs together to effectively create a single input, as shown in Figure 5–19(a) with a 2-input example. Also, an OR gate can be produced from NOR gates, as illustrated in Figure 5–19(b). An AND gate can be constructed by the use of NOR gates, as shown in

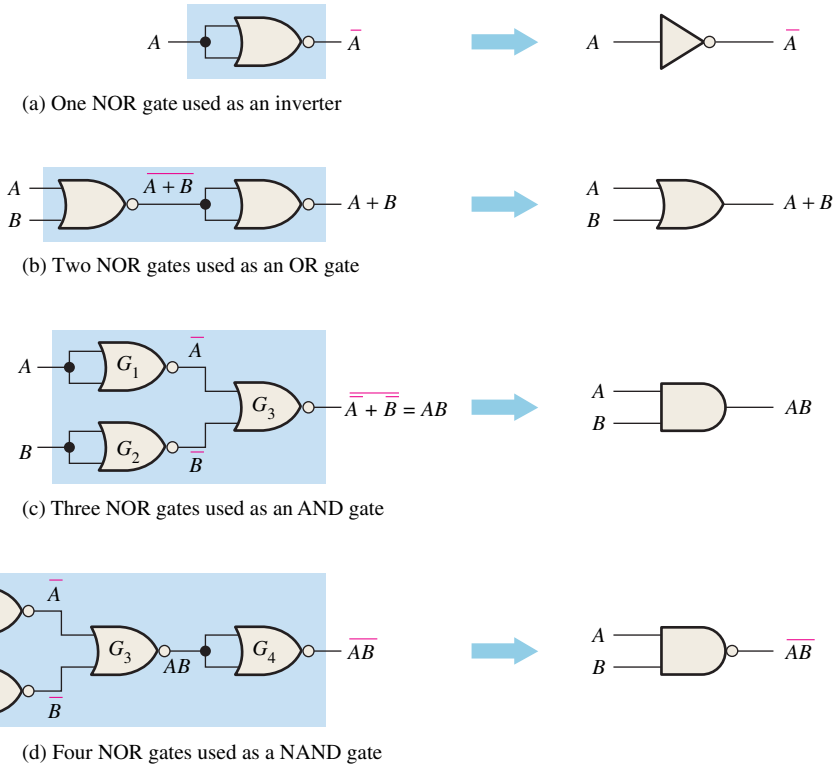


FIGURE 5–19 Universal application of NOR gates. Open files F05-19(a), (b), (c), and (d) to verify each of the equivalencies.

Figure 5–19(c). In this case the NOR gates G_1 and G_2 are used as inverters, and the final output is derived by the use of DeMorgan's theorem as follows:

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

Figure 5–19(d) shows how NOR gates are used to form a NAND function.

5–4 Combinational Logic Using NAND and NOR Gates

In this section, you will see how NAND and NOR gates can be used to implement a logic function. Recall from Chapter 3 that the NAND gate also exhibits an equivalent operation called the negative-OR and that the NOR gate exhibits an equivalent operation called the negative-AND. You will see how the use of the appropriate symbols to represent the equivalent operations makes "reading" a logic diagram easier.

NAND Logic

As you have learned, a NAND gate can function as either a NAND or a negative-OR because, by DeMorgan's theorem,

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

NAND — negative-OR

Consider the NAND logic in Figure 5–20. The output expression is developed in the following steps:

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

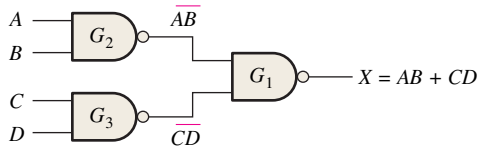
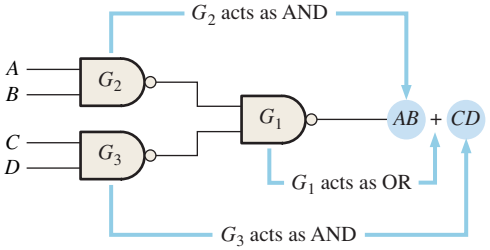


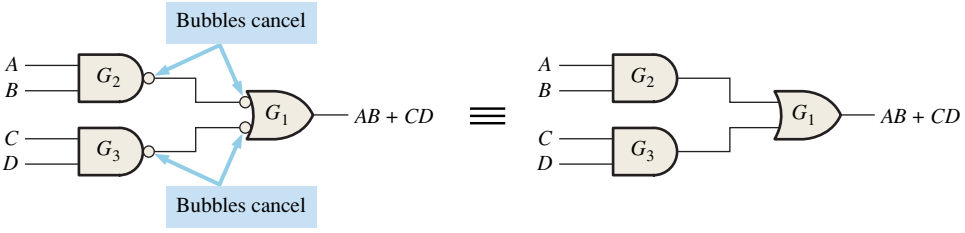
FIGURE 5–20 NAND logic for $X = AB + CD$.

As you can see in Figure 5–20, the output expression, $AB + CD$, is in the form of two AND terms ORed together. This shows that gates G_2 and G_3 act as AND gates and that gate G_1 acts as an OR gate, as illustrated in Figure 5–21(a). This circuit is redrawn in part (b) with NAND symbols for gates G_2 and G_3 and a negative-OR symbol for gate G_1.

Notice in Figure 5–21(b) the bubble-to-bubble connections between the outputs of gates G_2 and G_3 and the inputs of gate G_1. *Since a bubble represents an inversion, two*



(a) Original NAND logic diagram showing effective gate operation relative to the output expression



(b) Equivalent NAND/Negative-OR logic diagram (c) AND-OR equivalent

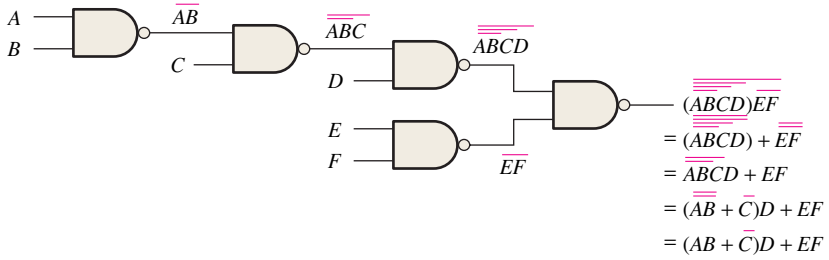
FIGURE 5–21 Development of the AND-OR equivalent of the circuit in Figure 5–20.

connected bubbles represent a double inversion and therefore cancel each other. This inversion cancellation can be seen in the previous development of the output expression $AB + CD$ and is indicated by the absence of barred terms in the output expression. Thus, the circuit in Figure 5–21(b) is *effectively* an AND-OR circuit, as shown in Figure 5–21(c).

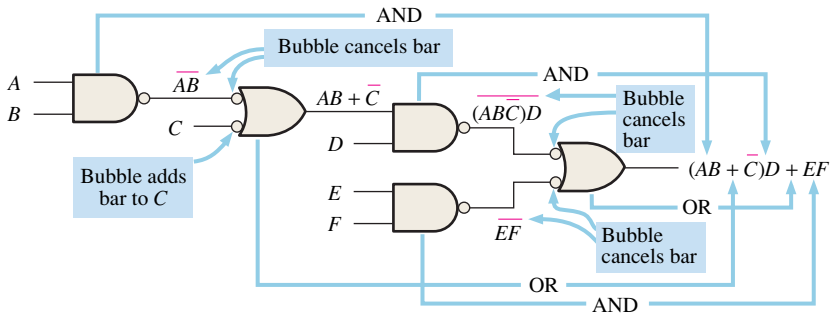
NAND Logic Diagrams Using Dual Symbols

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 nonbubble-to-nonbubble. In general, a bubble output should not be connected to a nonbubble input or vice versa in a logic diagram.

Figure 5–22 shows an arrangement of gates to illustrate the procedure of using the appropriate dual symbols for a NAND circuit with several gate levels. Although using all NAND symbols as in Figure 5–22(a) is correct, the diagram in part (b) is much easier to "read" and is the preferred method. As shown in Figure 5–22(b), the output gate is represented with a negative-OR symbol. Then the NAND symbol is used for the level of gates right before the output gate and the symbols for successive levels of gates are alternated as you move away from the output.



(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.

FIGURE 5–22 Illustration of the use of the appropriate dual symbols in a NAND logic diagram.

The shape of the gate indicates the way its inputs will appear in the output expression and thus shows how the gate functions within the logic circuit. For a NAND symbol, the inputs appear ANDed in the output expression; and for a negative-OR symbol, the inputs appear ORed in the output expression, as Figure 5–22(b) illustrates. The dual-symbol diagram in part (b) makes it easier to determine the output expression directly from the logic diagram because each gate symbol indicates the relationship of its input variables as they appear in the output expression.

EXAMPLE 5–9

Redraw the logic diagram and develop the output expression for the circuit in Figure 5–23 using the appropriate dual symbols.

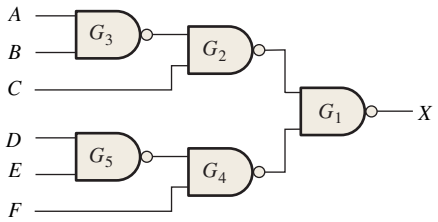
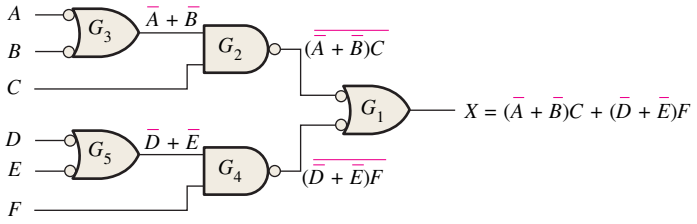


FIGURE 5–23

Solution

Redraw the logic diagram in Figure 5–23 with the use of equivalent negative-OR symbols as shown in Figure 5–24. Writing the expression for X directly from the indicated logic operation of each gate gives $X = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F$.



EXAMPLE 5–10

Implement each expression with NAND logic using appropriate dual symbols:

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

Solution

See Figure 5–25.

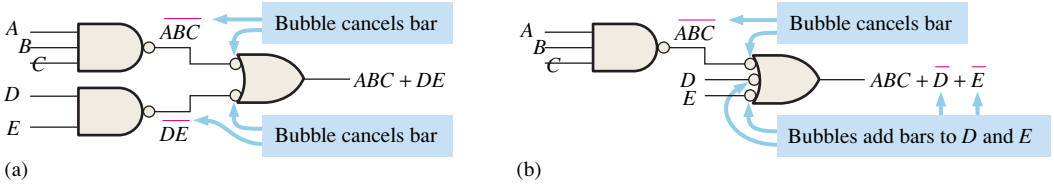


FIGURE 5–25

NOR Logic

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

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

NOR — negative-AND

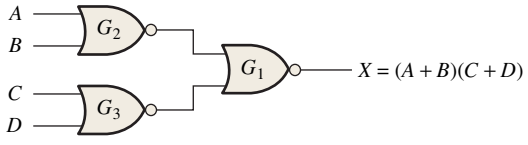


FIGURE 5–26 NOR logic for $X = (A + B)(C + D)$.

Consider the NOR logic in Figure 5–26. The output expression is developed as follows:

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

As you can see in Figure 5–26, the output expression $(A + B)(C + D)$ consists of two OR terms ANDed together. This shows that gates G_2 and G_3 act as OR gates and gate G_1 acts as an AND gate, as illustrated in Figure 5–27(a). This circuit is redrawn in part (b) with a negative-AND symbol for gate G_1.

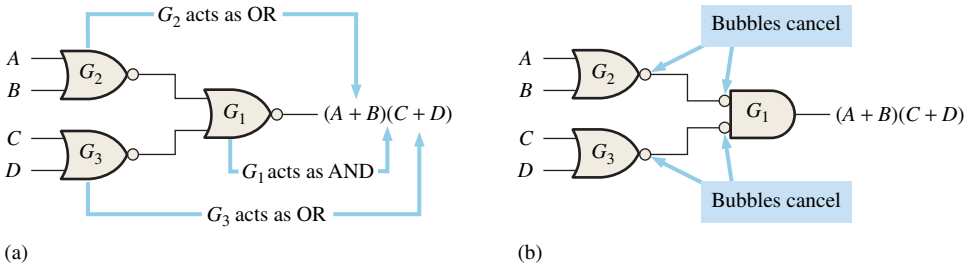
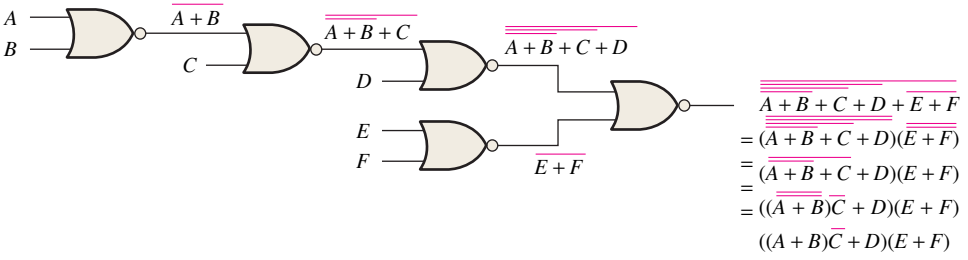


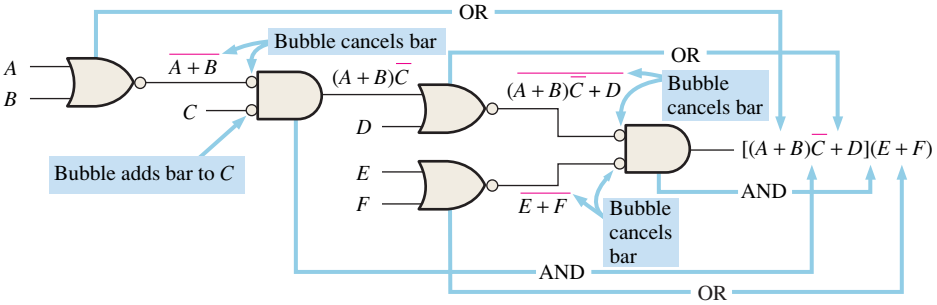
FIGURE 5–27

NOR Logic Diagram Using Dual Symbols

As with NAND logic, the purpose for using the dual symbols is to make the logic diagram easier to read and analyze, as illustrated in the NOR logic circuit in Figure 5–28. When the circuit in part (a) is redrawn with dual symbols in part (b), notice that all output-to-input



(a) Final output expression is obtained after several Boolean steps.



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

FIGURE 5–28 Illustration of the use of the appropriate dual symbols in a NOR logic diagram.

connections between gates are bubble-to-bubble or nonbubble-to-nonbubble. Again, you can see that the shape of each gate symbol indicates the type of term (AND or OR) that it produces in the output expression, thus making the output expression easier to determine and the logic diagram easier to analyze.

EXAMPLE 5–11

Using appropriate dual symbols, redraw the logic diagram and develop the output expression for the circuit in Figure 5–29.

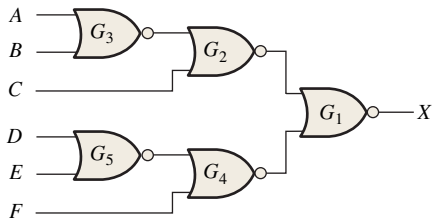


FIGURE 5–29

Solution

Redraw the logic diagram with the equivalent negative-AND symbols as shown in Fig-ure 5–30. Writing the expression for X directly from the indicated operation of each gate,

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

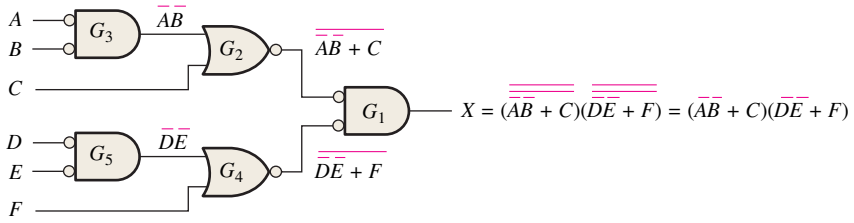


FIGURE 5–30

5–5 Pulse Waveform Operation

General combinational logic circuits with pulse waveform inputs are examined in this section. Keep in mind that the operation of each gate is the same for pulse waveform inputs as for constant-level inputs. The output of a logic circuit at any given time depends on the inputs at that particular time, so the relationship of the time-varying inputs is of primary importance.

The operation of any gate is the same regardless of whether its inputs are pulsed or constant levels. The nature of the inputs (pulsed or constant levels) does not alter the truth table of a circuit. The examples in this section illustrate the analysis of combinational logic circuits with pulse waveform inputs.

The following is a review of the operation of individual gates for use in analyzing combinational circuits with pulse waveform inputs:

1. The output of an AND gate is HIGH only when all inputs are HIGH at the same time.
2. The output of an OR gate is HIGH only when at least one of its inputs is HIGH.
3. The output of a NAND gate is LOW only when all inputs are HIGH at the same time.
4. The output of a NOR gate is LOW only when at least one of its inputs is HIGH.

EXAMPLE 5–12

Determine the final output waveform X for the circuit in Figure 5–31, with input waveforms A, B, and C as shown.

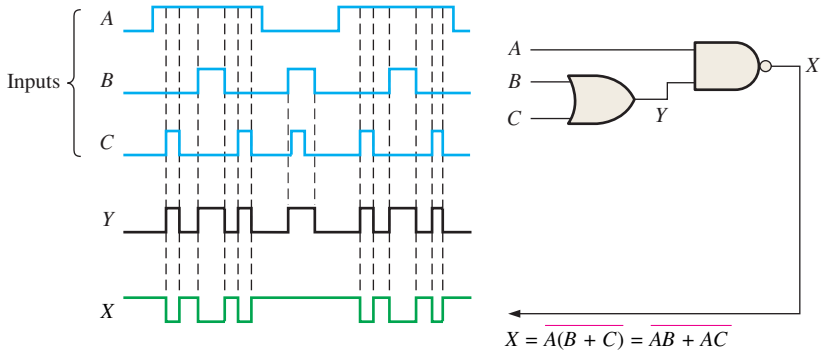


FIGURE 5–31

Solution

The output expression, $\overline{AB + AC}$, indicates that the output X is LOW when both A and B are HIGH or when both A and C are HIGH or when all inputs are HIGH. The output waveform X is shown in the timing diagram of Figure 5–31. The intermediate waveform Y at the output of the OR gate is also shown.

EXAMPLE 5–13

Draw the timing diagram for the circuit in Figure 5–32 showing the outputs of G_1, G_2, and G_3 with the input waveforms, A, and B, as indicated.

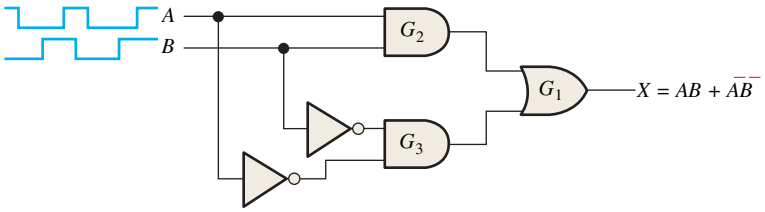


FIGURE 5–32

Solution

When both inputs are HIGH or when both inputs are LOW, the output X is HIGH as shown in Figure 5–33. Notice that this is an exclusive-NOR circuit. The intermediate outputs of gates G_2 and G_3 are also shown in Figure 5–33.

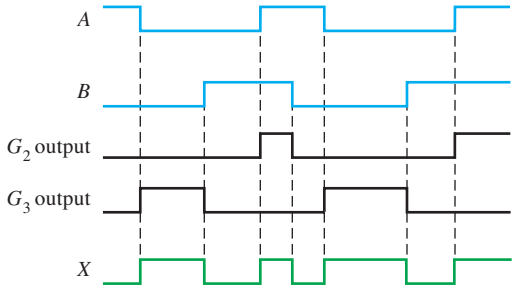


FIGURE 5–33

EXAMPLE 5–14

Determine the output waveform X for the logic circuit in Figure 5–34(a) by first finding the intermediate waveform at each of points Y_1, Y_2, Y_3, and Y_4. The input waveforms are shown in Figure 5–34(b).

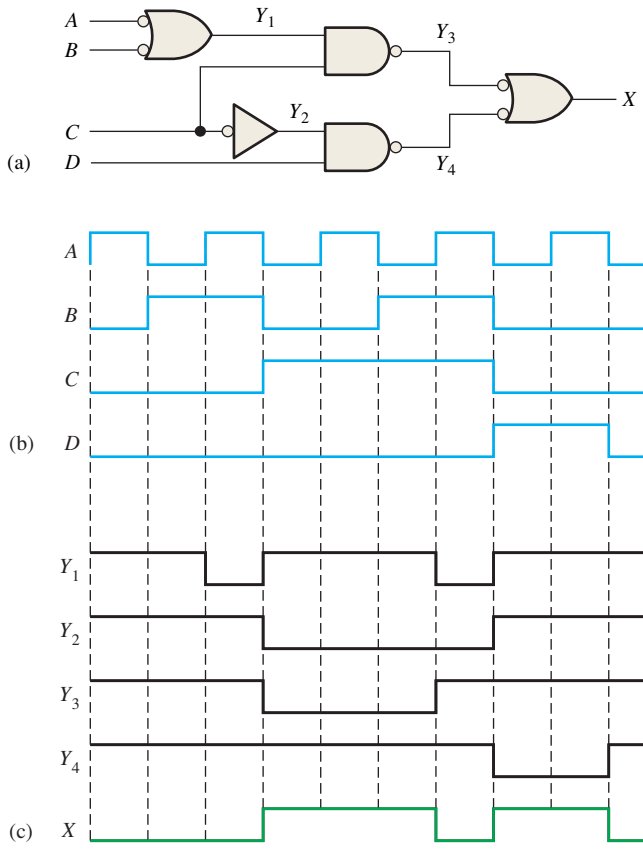


FIGURE 5–34

Solution

All the intermediate waveforms and the final output waveform are shown in the timing diagram of Figure 5–34(c).