4

Logic Gates and Related Devices

Logic gates are electronic circuits that can be used to implement the most elementary logic expressions, also known as Boolean expressions. The logic gate is the most basic building block of combinational logic. There are three basic logic gates, namely the OR gate, the AND gate and the NOT gate. Other logic gates that are derived from these basic gates are the NAND gate, the NOR gate, the EXCLUSIVE-OR gate and the EXCLUSIVE-NOR gate. This chapter deals with logic gates and some related devices such as buffers, drivers, etc., as regards their basic functions. The treatment of the subject matter is mainly with the help of respective truth tables and Boolean expressions. The chapter is adequately illustrated with the help of solved examples. Towards the end, the chapter contains application-relevant information in terms of popular type numbers of logic gates from different logic families and their functional description to help application engineers in choosing the right device for their application. Pin connection diagrams are given on the companion website at http://www.wiley.com/go/maini_digital. Different logic families used to hardware-implement different logic functions in the form of digital integrated circuits are discussed in the following chapter.

4.1 Positive and Negative Logic

The binary variables, as we know, can have either of the two states, i.e. the logic '0' state or the logic '1' state. These logic states in digital systems such as computers, for instance, are represented by two different voltage levels or two different current levels. If the more positive of the two voltage or current levels represents a logic '1' and the less positive of the two levels represents a logic '0', then the logic system is referred to as a *positive logic system*. If the more positive of the two voltage or current levels represents a logic '0' and the less positive of the two levels represents a logic '1', then the logic system is referred to as a *negative logic system*. The following examples further illustrate this concept.

If the two voltage levels are 0 V and +5 V, then in the positive logic system the 0 V represents a logic ‘0’ and the +5 V represents a logic ‘1’. In the negative logic system, 0 V represents a logic ‘1’ and +5 V represents a logic ‘0’.

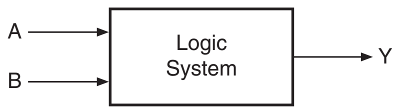
If the two voltage levels are 0 V and −5 V, then in the positive logic system the 0 V represents a logic ‘1’ and the −5 V represents a logic ‘0’. In the negative logic system, 0 V represents a logic ‘0’ and −5 V represents a logic ‘1’.

It is interesting to note, as we will discover in the latter part of the chapter, that a positive OR is a negative AND. That is, OR gate hardware in the positive logic system behaves like an AND gate in the negative logic system. The reverse is also true. Similarly, a positive NOR is a negative NAND, and vice versa.

4.2 Truth Table

A truth table lists all possible combinations of input binary variables and the corresponding outputs of a logic system. The logic system output can be found from the logic expression, often referred to as the Boolean expression, that relates the output with the inputs of that very logic system.

When the number of input binary variables is only one, then there are only two possible inputs, i.e. ‘0’ and ‘1’. If the number of inputs is two, there can be four possible input combinations, i.e. 00, 01, 10 and 11. Figure 4.1(b) shows the truth table of the two-input logic system represented by Fig. 4.1(a). The logic system of Fig. 4.1(a) is such that $Y = 0$ only when both $A = 0$ and $B = 0$. For all other possible input combinations, output $Y = 1$. Similarly, for three input binary variables, the number of possible input combinations becomes eight, i.e. 000, 001, 010, 011, 100, 101, 110 and 111. This statement can be generalized to say that, if a logic circuit has n binary inputs, its truth table will have 2^n possible input combinations, or in other words 2^n rows. Figure 4.2 shows the truth table of a three-input logic circuit, and it has 8 $(= 2^3)$ rows. Incidentally, as we will see later in the chapter, this is the truth table of a three-input AND gate. It may be mentioned here that the truth table of a three-input AND gate as given in Fig. 4.2 is drawn following the positive logic system, and also that, in all further discussion throughout the book, we will use a positive logic system unless otherwise specified.



(a)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(b)

Figure 4.1 Two-input logic system.

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Figure 4.2 Truth table of a three-input logic system

4.3 Logic Gates

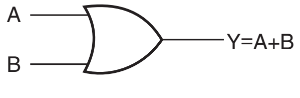
The logic gate is the most basic building block of any digital system, including computers. Each one of the basic logic gates is a piece of hardware or an electronic circuit that can be used to implement some basic logic expression. While laws of Boolean algebra could be used to do manipulation with binary variables and simplify logic expressions, these are actually implemented in a digital system with the help of electronic circuits called logic gates. The three basic logic gates are the OR gate, the AND gate and the NOT gate.

4.3.1 OR Gate

An OR gate performs an ORing operation on two or more than two logic variables. The OR operation on two independent logic variables A and B is written as $Y = A + B$ and reads as Y equals A OR B and not as A plus B. An OR gate is a logic circuit with two or more inputs and one output. The output of an OR gate is LOW only when all of its inputs are LOW. For all other possible input combinations, the output is HIGH. This statement when interpreted for a positive logic system means the following. The output of an OR gate is a logic '0' only when all of its inputs are at logic '0'. For all other possible input combinations, the output is a logic '1'. Figure 4.3 shows the circuit symbol and the truth table of a two-input OR gate. The operation of a two-input OR gate is explained by the logic expression

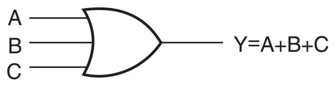
$$Y = A + B \tag{4.1}$$

As an illustration, if we have four logic variables and we want to know the logical output of $(A + B + C + D)$, then it would be the output of a four-input OR gate with A, B, C and D as its inputs.

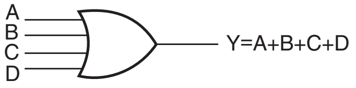


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Figure 4.3 Two-input OR gate.



(a)



(b)

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(c)

Figure 4.4 (a) Three-input OR gate, (b) four-input OR gate and (c) the truth table of a three-input OR gate.

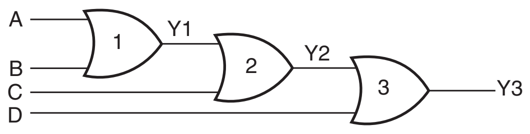
Figures 4.4(a) and (b) show the circuit symbol of three-input and four-input OR gates. Figure 4.4(c) shows the truth table of a three-input OR gate. Logic expressions explaining the functioning of three-input and four-input OR gates are $Y = A + B + C$ and $Y = A + B + C + D$.

Example 4.1

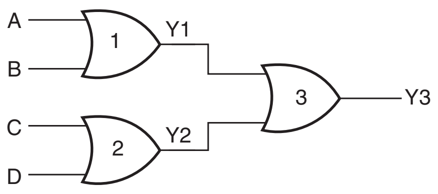
How would you hardware-implement a four-input OR gate using two-input OR gates only?

Solution

Figure 4.5(a) shows one possible arrangement of two-input OR gates that simulates a four-input OR gate. A, B, C and D are logic inputs and $Y3$ is the output. Figure 4.5(b) shows another possible arrangement. In the case of Fig. 4.5(a), the output of OR gate 1 is $Y1 = (A + B)$. The second



(a)



(b)

Figure 4.5 Example 4.1.

OR gate produces the output $Y2 = (Y1 + C) = (A + B + C)$. Similarly, the output of OR gate 3 is $Y3 = (Y2 + D) = (A + B + C + D)$. In the case of Fig. 4.5(b), the output of OR gate 1 is $Y1 = (A + B)$. The second OR gate produces the output $Y2 = (C + D)$. Output $Y3$ of the third OR gate is given by $(Y1 + Y2) = (A + B + C + D)$.

Example 4.2

Draw the output waveform for the OR gate and the given pulsed input waveforms of Fig. 4.6(a).

Solution

Figure 4.6(b) shows the output waveform. It can be drawn by following the truth table of the OR gate.

4.3.2 AND Gate

An AND gate is a logic circuit having two or more inputs and one output. The output of an AND gate is HIGH only when all of its inputs are in the HIGH state. In all other cases, the output is LOW. When interpreted for a positive logic system, this means that the output of the AND gate is a logic '1' only when all of its inputs are in logic '1' state. In all other cases, the output is logic '0'. The logic symbol and truth table of a two-input AND gate are shown in Figs 4.7(a) and (b) respectively. Figures 4.8(a) and (b) show the logic symbols of three-input and four-input AND gates respectively. Figure 4.8(c) gives the truth table of a four-input AND gate.

The AND operation on two independent logic variables A and B is written as $Y = A.B$ and reads as Y equals A AND B and not as A multiplied by B. Here, A and B are input logic variables and Y is the output. An AND gate performs an ANDing operation:

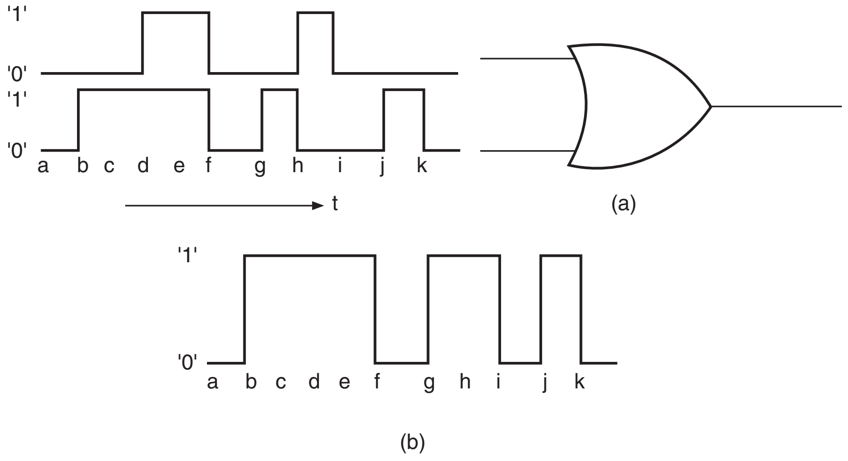
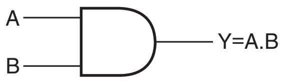


Figure 4.6 Example 4.2.

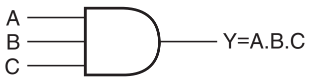


(a)

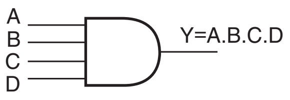
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(b)

Figure 4.7 Two-input AND gate.



(a)



(b)

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
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

(c)

Figure 4.8 (a) Three-input AND gate, (b) four-input AND gate and (c) the truth table of a four-input AND gate.

- for a two-input AND gate, $Y = A.B$;
- for a three-input AND gate, $Y = A.B.C$;
- for a four-input AND gate, $Y = A.B.C.D$.

If we interpret the basic definition of OR and AND gates for a negative logic system, we have an interesting observation. We find that an OR gate in a positive logic system is an AND gate in a negative logic system. Also, a positive AND is a negative OR.

Example 4.3

Show the logic arrangement for implementing a four-input AND gate using two-input AND gates only.

Solution

Figure 4.9 shows the hardware implementation of a four-input AND gate using two-input AND gates. The output of AND gate 1 is $Y1 = A.B$. The second AND gate produces an output $Y2$ given by $Y2 = Y1.C = A.B.C$. Similarly, the output of AND gate 3 is $Y = Y2.D = A.B.C.D$ and hence the result.

4.3.3 NOT Gate

A NOT gate is a one-input, one-output logic circuit whose output is always the complement of the input. That is, a LOW input produces a HIGH output, and vice versa. When interpreted for a positive logic system, a logic '0' at the input produces a logic '1' at the output, and vice versa. It is also known as a 'complementing circuit' or an 'inverting circuit'. Figure 4.10 shows the circuit symbol and the truth table.

The NOT operation on a logic variable X is denoted as $\overline{X}$ or X'. That is, if X is the input to a NOT circuit, then its output Y is given by $Y = \overline{X}$ or X' and reads as Y equals NOT X. Thus, if $X = 0$, $Y = 1$ and if $X = 1$, $Y = 0$.

Example 4.4

For the logic circuit arrangements of Figs 4.11(a) and (b), draw the output waveform.

Solution

In the case of the OR gate arrangement of Fig. 4.11(a), the output will be permanently in logic '1' state as the two inputs can never be in logic '0' state together owing to the presence of the inverter. In the case of the AND gate arrangement of Fig. 4.11(b), the output will be permanently in logic '0' state as the two inputs can never be in logic '1' state together owing to the presence of the inverter.

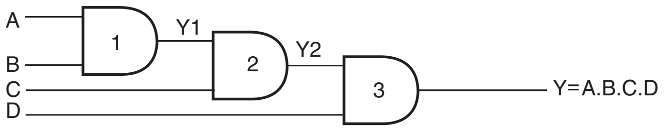
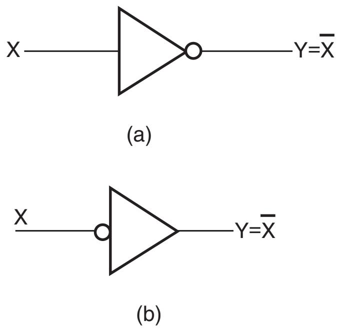


Figure 4.9 Implementation of a four-input AND gate using two-input AND gates.



X	Y
0	1
1	0

Figure 4.10 (a) Circuit symbol of a NOT circuit and (b) the truth table of a NOT circuit.

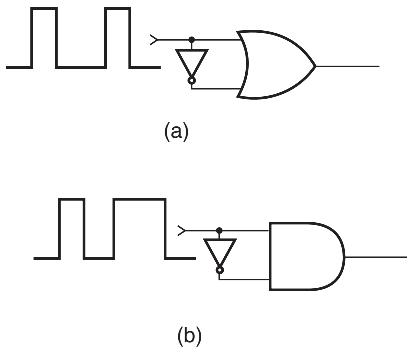
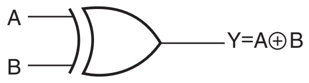


Figure 4.11 Example 4.4.

4.3.4 EXCLUSIVE-OR Gate

The EXCLUSIVE-OR gate, commonly written as EX-OR gate, is a two-input, one-output gate. Figures 4.12(a) and (b) respectively show the logic symbol and truth table of a two-input EX-OR gate. As can be seen from the truth table, the output of an EX-OR gate is a logic '1' when the inputs are unlike and a logic '0' when the inputs are like. Although EX-OR gates are available in integrated circuit form only as two-input gates, unlike other gates which are available in multiple inputs also, multiple-input EX-OR logic functions can be implemented using more than one two-input gates. The truth table of a multiple-input EX-OR function can be expressed as follows. The output of a multiple-input EX-OR logic function is a logic '1' when the number of 1s in the input sequence is odd and a logic '0' when the number of 1s in the input sequence is even, including zero. That is, an all 0s input sequence also produces a logic '0' at the output. Figure 4.12(c) shows the truth table of a four-input EX-OR function. The output of a two-input EX-OR gate is expressed by

$$Y = (A \oplus B) = \overline{A}B + A\overline{B} \tag{4.2}$$



(a)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

(b)

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

(c)

Figure 4.12 (a) Circuit symbol of a two-input EXCLUSIVE-OR gate, (b) the truth table of a two-input EXCLUSIVE-OR gate and (c) the truth table of a four-input EXCLUSIVE-OR gate

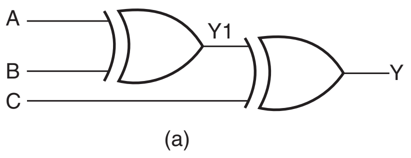
Example 4.5

How do you implement three-input and four-input EX-OR logic functions with the help of two-input EX-OR gates?

Solution

Figures 4.13(a) and (b) show the implementation of a three-input EX-OR logic function and a four-input EX-OR logic function using two-input logic gates:

- For Fig. 4.13(a), the output $Y1$ is given by $A \oplus B$. The final output Y is given by $Y = (Y1 \oplus C) = (A \oplus B) \oplus C = A \oplus B \oplus C$.
- Figure 4.13(b) can be explained on similar lines.



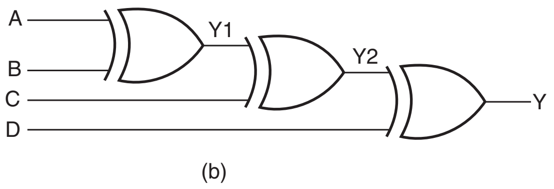


Figure 4.13 (a) Three-input EX-OR gate and (b) a four-input EX-OR gate.

Example 4.6

How can you implement a NOT circuit using a two-input EX-OR gate?

Solution

Refer to the truth table of a two-input EX-OR gate reproduced in Fig. 4.14(a). It is clear from the truth table that, if one of the inputs of the gate is permanently tied to logic '1' level, then the other input and output perform the function of a NOT circuit. Figure 4.14(b) shows the implementation.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

(a)

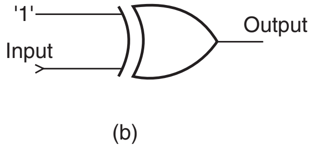


Figure 4.14 Implementation of a NOT circuit using an EX-OR gate.

4.3.5 NAND Gate

NAND stands for NOT AND. An AND gate followed by a NOT circuit makes it a NAND gate [Fig. 4.15(a)]. Figure 4.15(b) shows the circuit symbol of a two-input NAND gate. The truth table of a NAND gate is obtained from the truth table of an AND gate by complementing the output entries [Fig. 4.15(c)]. The output of a NAND gate is a logic ‘0’ when all its inputs are a logic ‘1’. For all other input combinations, the output is a logic ‘1’. NAND gate operation is logically expressed as

$$Y = \overline{A.B} \tag{4.3}$$

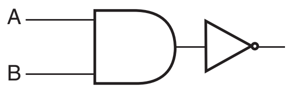
In general, the Boolean expression for a NAND gate with more than two inputs can be written as

$$Y = \overline{(A.B.C.D...)} \tag{4.4}$$

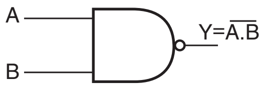
4.3.6 NOR Gate

NOR stands for NOT OR. An OR gate followed by a NOT circuit makes it a NOR gate [Fig. 4.16(a)]. The truth table of a NOR gate is obtained from the truth table of an OR gate by complementing the output entries. The output of a NOR gate is a logic ‘1’ when all its inputs are logic ‘0’. For all other input combinations, the output is a logic ‘0’. The output of a two-input NOR gate is logically expressed as

$$Y = \overline{(A+B)} \tag{4.5}$$



(a)

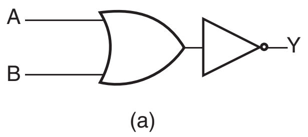


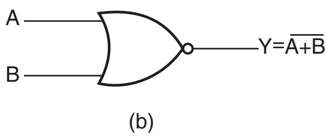
(b)

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(c)

Figure 4.15 (a) Two-input NAND implementation using an AND gate and a NOT circuit, (b) the circuit symbol of a two-input NAND gate and (c) the truth table of a two-input NAND gate.





A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(c)

Figure 4.16 (a) Two-input NOR implementation using an OR gate and a NOT circuit, (b) the circuit symbol of a two-input NOR gate and (c) the truth table of a two-input NOR gate.

In general, the Boolean expression for a NOR gate with more than two inputs can be written as

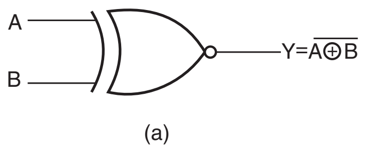
$$Y = \overline{(A + B + C + D...)} \tag{4.6}$$

4.3.7 EXCLUSIVE-NOR Gate

EXCLUSIVE-NOR (commonly written as EX-NOR) means NOT of EX-OR, i.e. the logic gate that we get by complementing the output of an EX-OR gate. Figure 4.17 shows its circuit symbol along with its truth table.

The truth table of an EX-NOR gate is obtained from the truth table of an EX-OR gate by complementing the output entries. Logically,

$$Y = (\overline{A \oplus B}) = (A.B + \overline{A}.\overline{B}) \tag{4.7}$$



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

(b)

Figure 4.17 (a) Circuit symbol of a two-input EXCLUSIVE-NOR gate and (b) the truth table of a two-input EXCLUSIVE-NOR gate.

The output of a two-input EX-NOR gate is a logic ‘1’ when the inputs are like and a logic ‘0’ when they are unlike. In general, the output of a multiple-input EX-NOR logic function is a logic ‘0’ when the number of 1s in the input sequence is odd and a logic ‘1’ when the number of 1s in the input sequence is even including zero. That is, an all 0s input sequence also produces a logic ‘1’ at the output.

Example 4.7

Show the logic arrangements for implementing:

(a) a four-input NAND gate using two-input AND gates and NOT gates;
(b) a three-input NAND gate using two-input NAND gates;
(c) a NOT circuit using a two-input NAND gate;
(d) a NOT circuit using a two-input NOR gate;
(e) a NOT circuit using a two-input EX-NOR gate.

Solution

(a) Figure 4.18(a) shows the arrangement. The logic diagram is self-explanatory. The first step is to get a four-input AND gate using two-input AND gates. The output thus obtained is then complemented using a NOT circuit as shown.
(b) Figure 4.18(b) shows the arrangement, which is again self-explanatory. The first step is to get a two-input AND from a two-input NAND. The output of the two-input AND gate and the third input then feed the inputs of another two-input NAND to get the desired output.
(c) Shorting the inputs of the NAND gives a one-input, one-output NOT circuit. This is because when all inputs to a NAND are at logic ‘0’ level the output is a logic ‘1’, and when all inputs to a NAND are at logic ‘1’ level the output is a logic ‘0’. Figure 4.18(c) shows the implementation.
(d) Again, shorting the inputs of a NOR gate gives a NOT circuit. From the truth table of a NOR gate it is evident that an all 0s input to a NOR gate gives a logic ‘1’ output and an all 1s input gives a logic ‘0’ output. Figure 4.18(d) shows the implementation.
(e) It is evident from the truth table of a two-input EX-NOR gate that, if one of the inputs is permanently tied to a logic ‘0’ level and the other input is treated as the input, then it behaves as a NOT circuit between input and output [Fig. 4.18(e)]. When the input is a logic ‘0’, the two inputs become 00, which produces a logic ‘1’ at the output. When the input is at logic ‘1’ level, a 01 input produces a logic ‘0’ at the output.

Example 4.8

How do you implement a three-input EX-NOR function using only two-input EX-NOR gates?

Solution

Figure 4.19 shows the arrangement. The first two EX-NOR gates implement a two-input EX-OR gate using two-input EX-NOR gates. The second EX-NOR gate here has been wired as a NOT circuit. The output of the second gate and the third input are fed to the two inputs of the third EX-NOR gate.

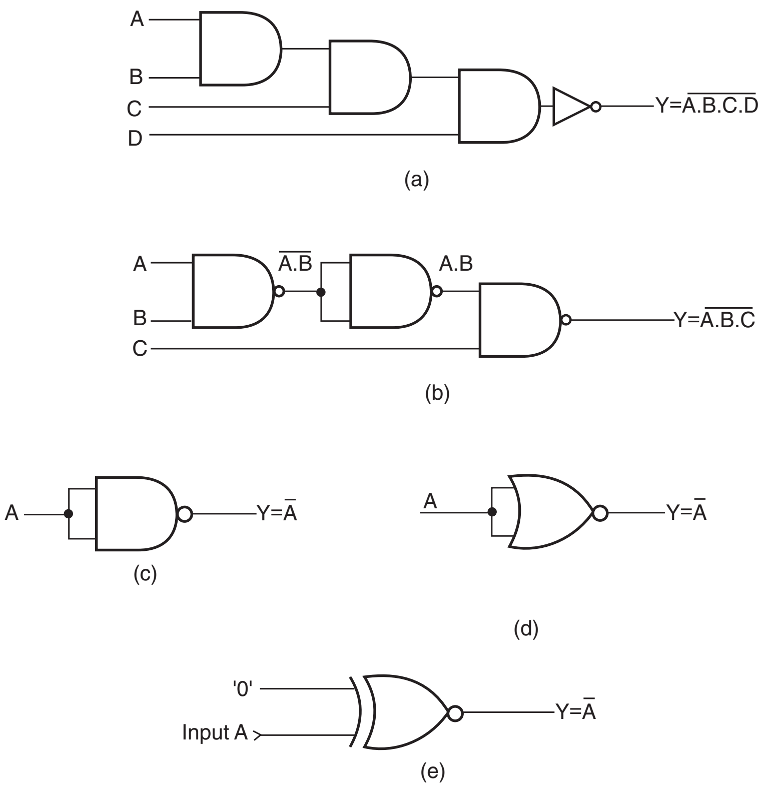


Figure 4.18 Example 4.7.

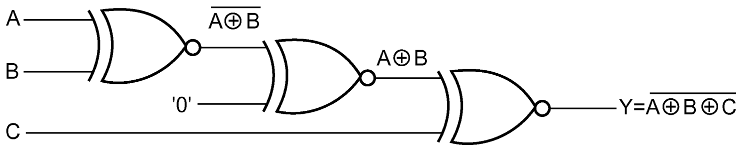


Figure 4.19 Example 4.8.

4.3.8 INHIBIT Gate

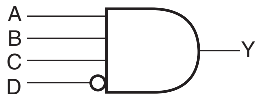
There are many situations in digital circuit design where the passage of a logic signal needs to be either enabled or inhibited depending upon certain other control inputs. INHIBIT here means that the gate produces a certain fixed logic level at the output irrespective of changes in the input logic level. As an illustration, if one of the inputs of a four-input NOR gate is permanently tied to logic '1' level, then the output will always be at logic '0' level irrespective of the logic status of other inputs. This gate will behave as a NOR gate only when this control input is at logic '0' level. This is an example of the INHIBIT function. The INHIBIT function is available in integrated circuit form for an AND gate, which is basically an AND gate with one of its inputs negated by an inverter. The negated input acts to inhibit the gate. In other words, the gate will behave like an AND gate only when the negated input is driven to a logic ‘0’. Figure 4.20 shows the circuit symbol and truth table of a four-input INHIBIT gate.

Example 4.9

Refer to the INHIBIT gate of Fig. 4.21(a). If the waveform of Fig. 4.21(b) is applied to the INHIBIT input, draw the waveform at the output.

Solution

Since all other inputs of the gate have been permanently tied to logic ‘1’ level, a logic ‘0’ at the INHIBIT input would produce a logic ‘1’ at the output and a logic ‘1’ at the INHIBIT input would produce a logic ‘0’ at the output. The output waveform is therefore the inversion of the input waveform and is shown in Fig. 4.22.



(a)

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

(b)

Figure 4.20 INHIBIT gate.

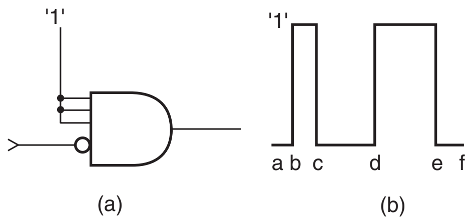


Figure 4.21 Example 4.9.

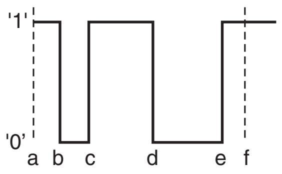


Figure 4.22 Solution to example 4.9.

Example 4.10

Refer to the INHIBIT gate shown in Fig. 4.23(a) and the INHIBIT input waveform shown in Fig. 4.23(b). Sketch the output waveform.

Solution

The output will always be at logic '1' level as two of the inputs of the logic gate, which is a NAND, are permanently tied to logic '0' level. This would have been so even if one of the inputs of the gate were at logic '0' level.

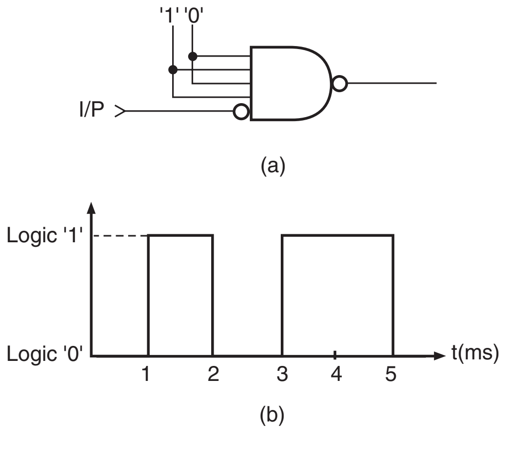


Figure 4.23 Example 4.10.

4.4 Universal Gates

OR, AND and NOT gates are the three basic logic gates as they together can be used to construct the logic circuit for any given Boolean expression. NOR and NAND gates have the property that they individually can be used to hardware-implement a logic circuit corresponding to any given Boolean expression. That is, it is possible to use either only NAND gates or only NOR gates to implement any Boolean expression. This is so because a combination of NAND gates or a combination of NOR gates can be used to perform functions of any of the basic logic gates. It is for this reason that NAND and NOR gates are universal gates.

As an illustration, Fig. 4.24 shows how two-input NAND gates can be used to construct a NOT circuit [Fig. 4.24(a)], a two-input AND gate [Fig. 4.24(b)] and a two-input OR gate [Fig. 4.24(c)]. Figure 4.25 shows the same using NOR gates. Understanding the conversion of NAND to OR and NOR to AND requires the use of DeMorgan's theorem, which is discussed in Chapter 6 on Boolean algebra.

4.5 Gates with Open Collector/Drain Outputs

These are gates where we need to connect an external resistor, called the pull-up resistor, between the output and the DC power supply to make the logic gate perform the intended logic function. Depending on the logic family used to construct the logic gate, they are referred to as gates with open collector output (in the case of the TTL logic family) or open drain output (in the case of the MOS logic family). Logic families are discussed in detail in Chapter 5.

The advantage of using open collector/open drain gates lies in their capability of providing an ANDing operation when outputs of several gates are tied together through a common pull-up resistor,

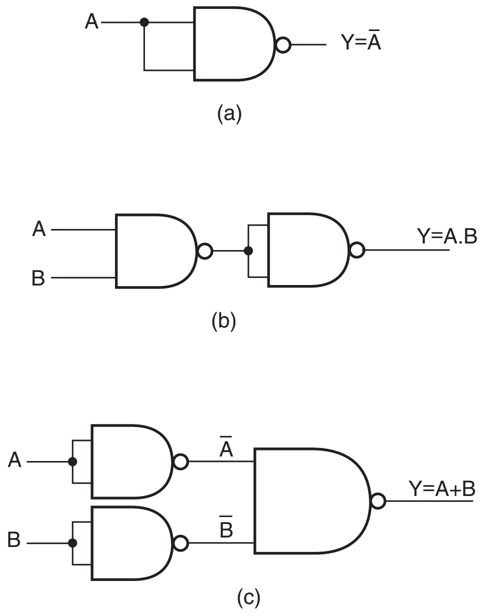


Figure 4.24 Implementation of basic logic gates using only NAND gates.

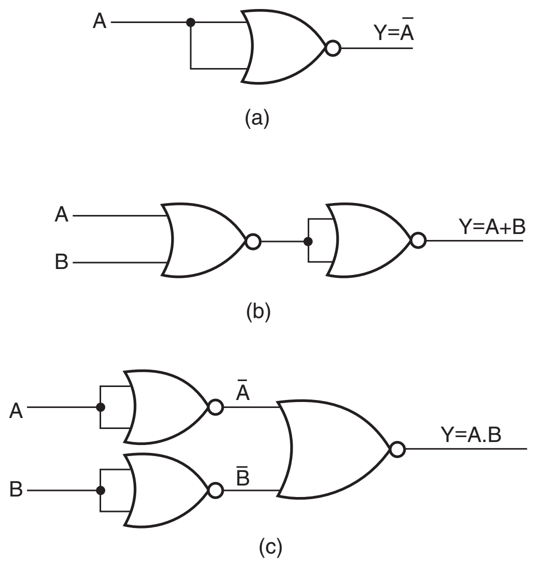


Figure 4.25 Implementation of basic logic gates using only NOR gates.

without having to use an AND gate for the purpose. This connection is also referred to as WIRE-AND connection. Figure 4.26(a) shows such a connection for open collector NAND gates. The output in this case would be

$$Y = \overline{AB}.\overline{CD}.\overline{EF} \tag{4.8}$$

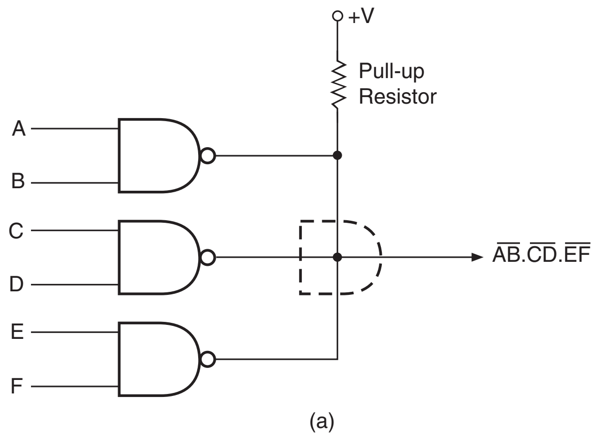


Figure 4.26 WIRE-AND connection with open collector/drain devices.

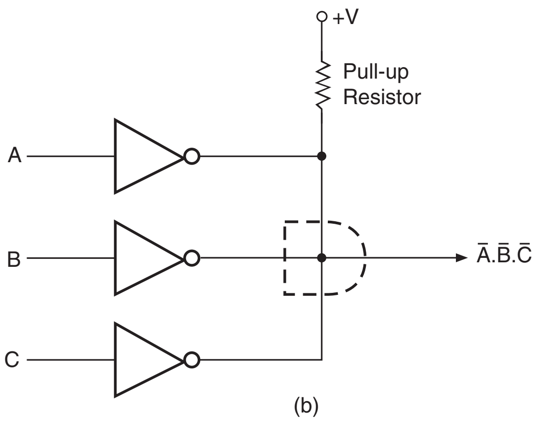


Figure 4.26 (*continued*).

Figure 4.26(b) shows a similar arrangement for NOT gates. The disadvantage is that they are relatively slower and noisier. Open collector/drain devices are therefore not recommended for applications where speed is an important consideration.

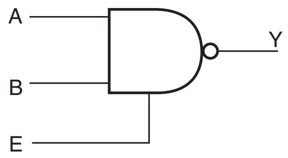
4.6 Tristate Logic Gates

Tristate logic gates have three possible output states, i.e. the logic ‘1’ state, the logic ‘0’ state and a high-impedance state. The high-impedance state is controlled by an external ENABLE input. The ENABLE input decides whether the gate is active or in the high-impedance state. When active, it can be ‘0’ or ‘1’ depending upon input conditions. One of the main advantages of these gates is that their inputs and outputs can be connected in parallel to a common bus line. Figure 4.27(a) shows the circuit symbol of a tristate NAND gate with active HIGH ENABLE input, along with its truth table. The one shown in Fig. 4.27(b) has active LOW ENABLE input. When tristate devices are paralleled, only one of them is enabled at a time. Figure 4.28 shows paralleling of tristate inverters having active HIGH ENABLE inputs.

4.7 AND-OR-INVERT Gates

AND-OR and OR-AND gates can be usefully employed to implement sum-of-products and product-of-sums Boolean expressions respectively. Figures 4.29(a) and (b) respectively show the symbols of AND-OR-INVERT and OR-AND-INVERT gates.

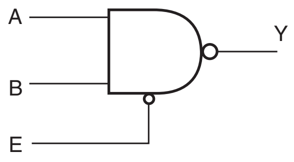
Another method for designating the gates shown in Fig. 4.29 is to call them two-wide, two-input AND-OR-INVERT or OR-AND-INVERT gates as the case may be. The gate is two-wide as there are two gates at the input, and two-input as each of the gates has two inputs. Other varieties such as two-wide, four-input AND-OR-INVERT (Fig. 4.30) and four-wide, two-input AND-OR-INVERT (Fig. 4.31) are also available in IC form.



A	B	E	Y
0	0	0	Z
0	0	1	1
0	1	0	Z
0	1	1	1
1	0	0	Z
1	0	1	1
1	1	0	Z
1	1	1	0

Z= High Impedance state

(a)



A	B	E	Y
0	0	0	1
0	0	1	Z
0	1	0	1
0	1	1	Z
1	0	0	1
1	0	1	Z
1	1	0	0
1	1	1	Z

Z= High Impedance state

(b)

Figure 4.27 Tristate devices.

4.8 Schmitt Gates

The logic gates discussed so far have a single-input threshold voltage level. This threshold is the same for both LOW-to-HIGH and HIGH-to-LOW output transitions. This threshold voltage lies somewhere between the highest LOW voltage level and the lowest HIGH voltage level guaranteed by the manufacturer of the device. These logic gates can produce an erratic output when fed with a slow

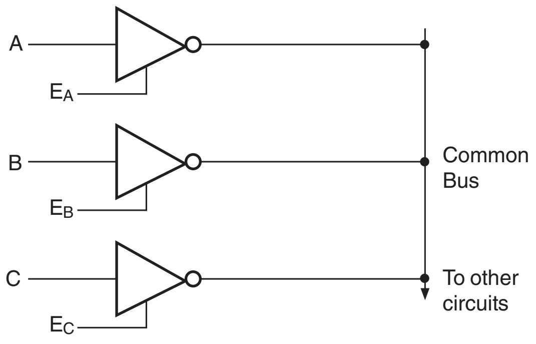


Figure 4.28 Paralleling of tristate inverters.

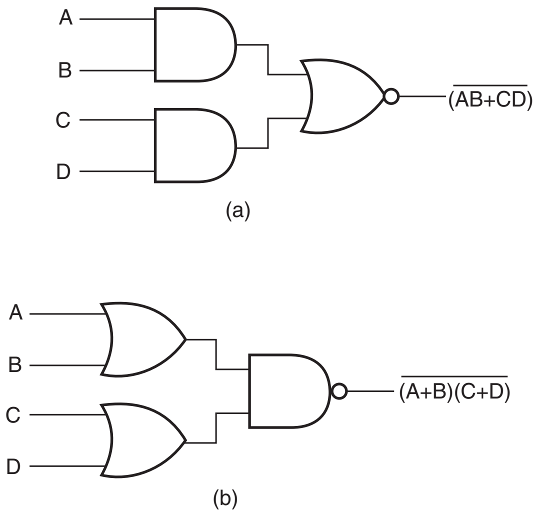


Figure 4.29 AND-OR-INVERT and OR-AND-INVERT gates.

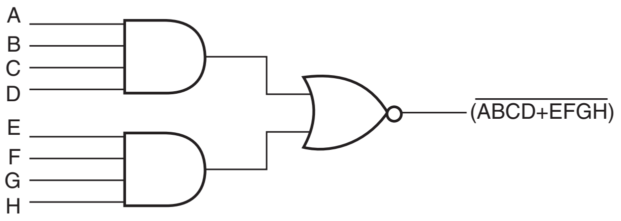


Figure 4.30 Two-wide, four-input AND-OR-INVERT gate.

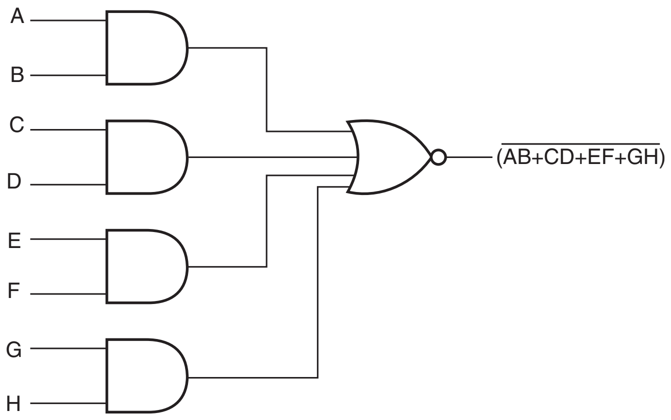


Figure 4.31 Four-wide, two-input AND-OR-INVERT gate.

varying input. Figure 4.32 shows the response of an inverter circuit when fed with a slow varying input both in the case of an ideal signal [Fig. 4.32(a)] and in the case of a practical signal having a small amount of AC noise superimposed on it [Fig. 4.32(b)]. A possible solution to this problem lies in having two different threshold voltage levels, one for LOW-to-HIGH transition and the other for HIGH-to-LOW transition, by introducing some positive feedback in the internal gate circuitry, a phenomenon called hysteresis.

There are some logic gate varieties, mainly in NAND gates and inverters, that are available with built-in hysteresis. These are called Schmitt gates, which interpret varying input voltages according to two threshold voltages, one for LOW-to-HIGH and the other for HIGH-to-LOW output transition. Figures 4.33(a) and (b) respectively show circuit symbols of Schmitt NAND and Schmitt inverter. Schmitt gates are distinguished from conventional gates by the small 'hysteresis' symbol reminiscent of the $B-H$ loop for a ferromagnetic material. Figure 4.33(c) shows typical transfer characteristics for such a device. The difference between the two threshold levels is

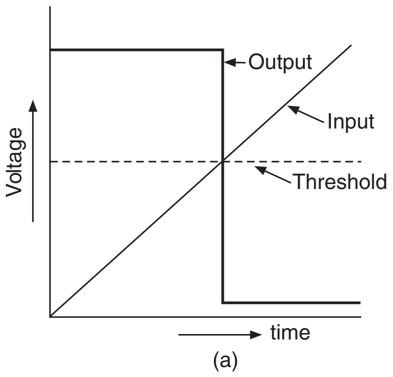
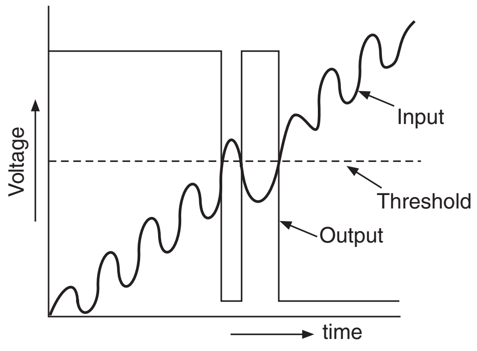


Figure 4.32 Response of conventional inverters to slow varying input.



(b)

Figure 4.32 (*continued*).

the hysteresis. These characteristics have been reproduced from the data sheet of IC 74LS132, which is a quad two-input Schmitt NAND belonging to the low-power Schottky TTL family. Figure 4.33(d) shows the response of a Schmitt inverter to a slow varying noisy input signal. We will learn more about different logic families in Chapter 5. It may be mentioned here that hysteresis increases noise immunity and is used in applications where noise is expected on input signal lines.

4.9 Special Output Gates

There are many applications where it is desirable to have both noninverted and inverted outputs. Examples include a single-input gate that is both an inverter and a noninverting buffer, or a two-input logic gate that is both an AND and a NAND. Such gates are called complementary output gates and are particularly useful in circuits where PCB space is at a premium. These are also useful in circuits where the addition of an inverter to obtain the inverted output introduces an undesirable time delay between inverted and noninverted outputs. Figure 4.34 shows the circuit symbols of complementary buffer, AND, OR and EX-OR gates.

Example 4.11

Draw the circuit symbols for (a) a two-wide, four-input OR-AND-INVERT gate and (b) a four-wide, two-input OR-AND-INVERT gate.

Solution

(a) Refer to Fig. 4.35(a).
(b) Refer to Fig. 4.35(b).

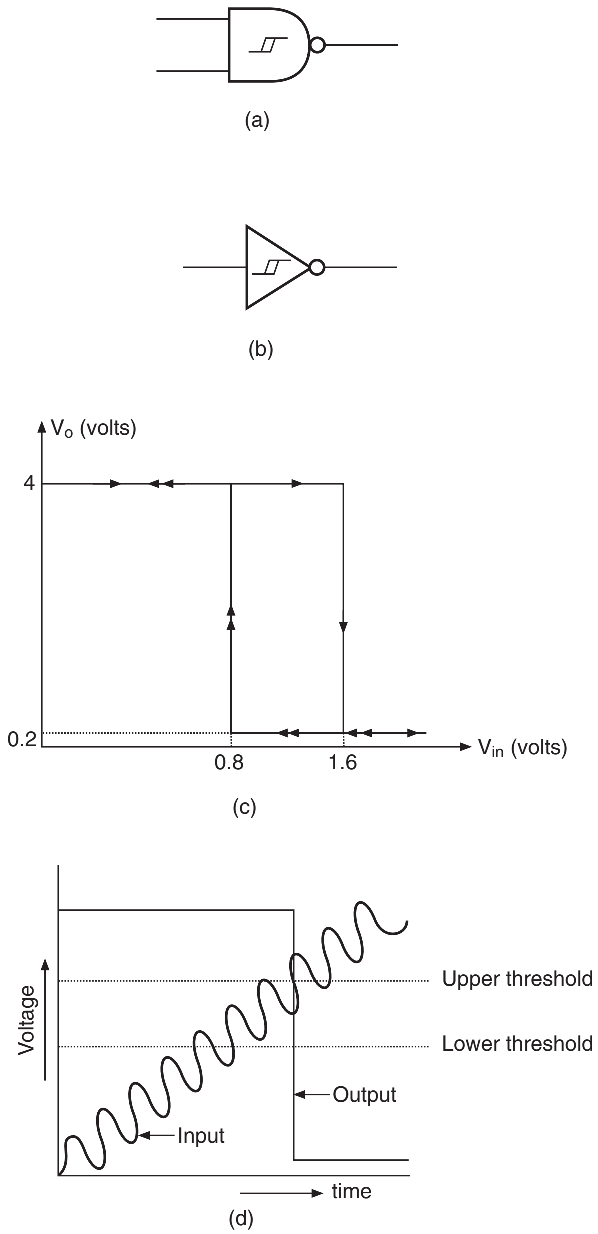


Figure 4.33 Schmitt gates.

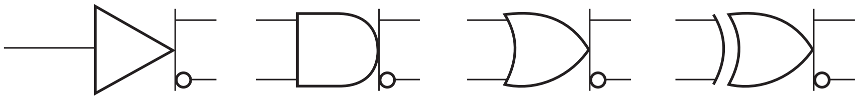


Figure 4.34 Complementary gates.

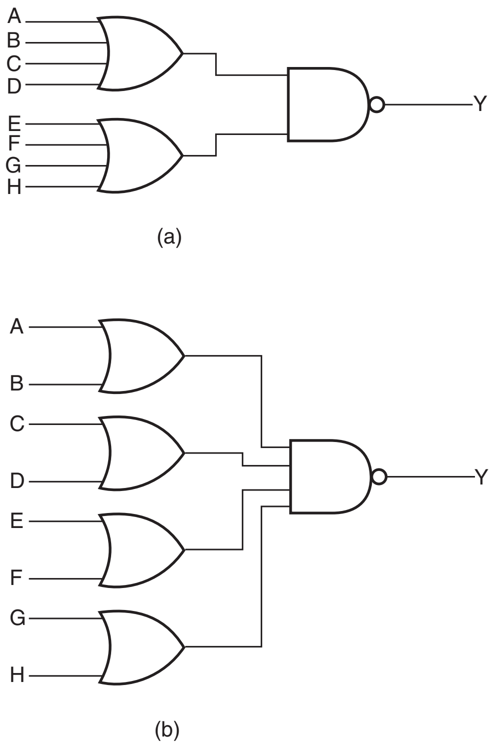


Figure 4.35 Example 4.11.

Example 4.12

Refer to Fig. 4.36(a). If the NAND gate used has the transfer characteristics of Fig. 4.36(b), sketch the expected output waveform.

Solution

The output waveform is shown in Fig. 4.36(c). The output is initially in logic '1' state. It goes from logic '1' to logic '0' state as the input exceeds 2 V. The output goes from logic '0' to logic '1' state as the input drops below 1 V.

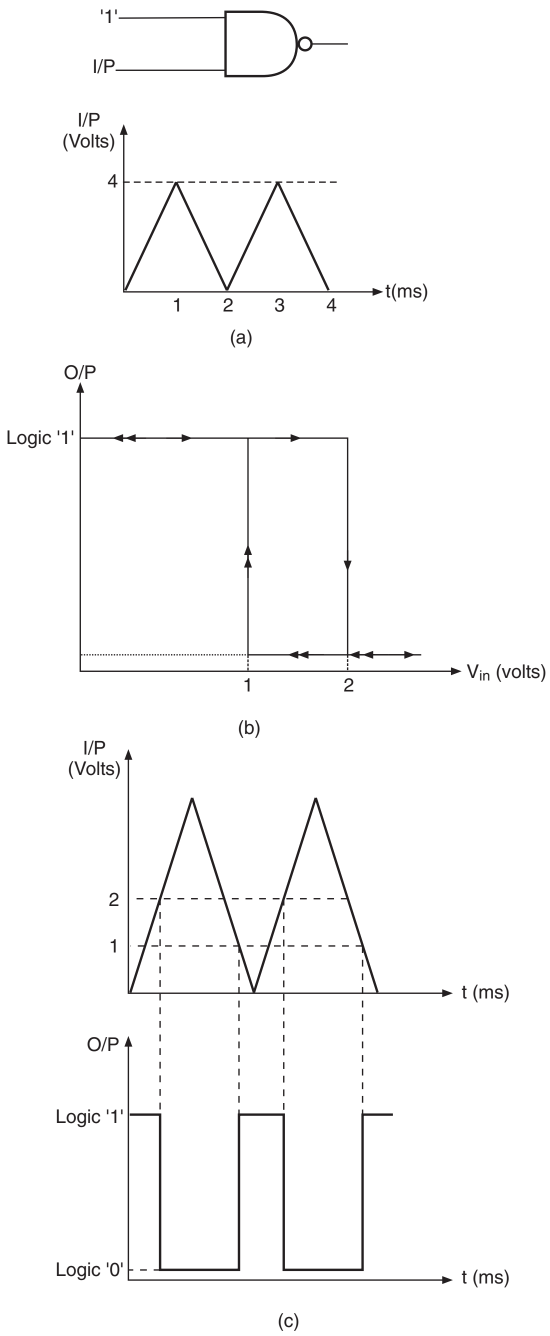


Figure 4.36 Example 4.12.

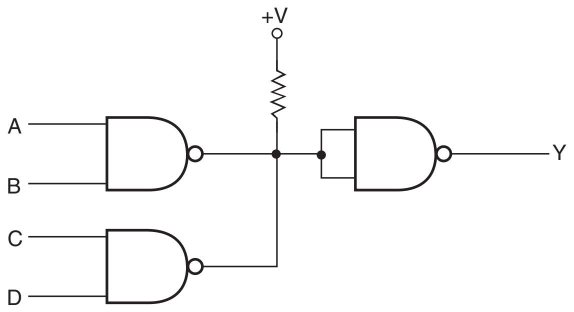


Figure 4.37 Example 4.13.

Example 4.13

Refer to the logic arrangement of Fig. 4.37. Write the logic expression for the output Y.

Solution

The NAND gates used in the circuit are open collector gates. Paralleling of the two NAND gates at the input leads to a WIRE-AND connection. Therefore the logic expression at the point where the two outputs combine is given by the equation

$$(\overline{AB}.\overline{CD}) \qquad (4.9)$$

Using DeMorgan's theorem (discussed in Chapter 6 on Boolean algebra),

$$(\overline{AB}.\overline{CD}) = (\overline{AB + CD}) \qquad (4.10)$$

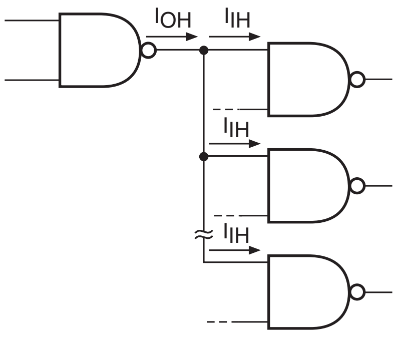
The third NAND is wired as an inverter. Therefore, the final output can be written as

$$Y = (AB + CD) \qquad (4.11)$$

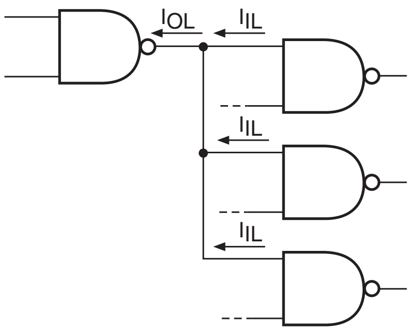
4.10 Fan-Out of Logic Gates

It is a common occurrence in logic circuits that the output of one logic gate feeds the inputs of several others. It is not practical to drive the inputs of an unlimited number of logic gates from the output of a single logic gate. This is limited by the current-sourcing capability of the output when the output of the logic gate is HIGH and by the current-sinking capability of the output when it is LOW, and also by the requirement of the inputs of the logic gates being fed in the two states.

To illustrate the point further, let us say that the current-sourcing capability of a certain NAND gate is I_{OH} when its output is in the logic HIGH state and that each of the inputs of the logic gate that it is driving requires an input current I_{IH}, as shown in Fig. 4.38(a). In this case, the output of the logic gate will be able to drive a maximum of I_{OH}/I_{IH} inputs when it is in the logic HIGH state. When the output of the driving logic gate is in the logic LOW state, let us say that it has a maximum current-sinking capability I_{OL}, and that each of the inputs of the driven logic gates requires a sinking current I_{IL}, as shown in Fig. 4.38(b). In this case the output of the logic gate will be able to drive a maximum of



(a)



(b)

Figure 4.38 Fan-out of logic gates.

I_{OL}/I_{IL} inputs when it is in the logic LOW state. Thus, the number of logic gate inputs that can be driven from the output of a single logic gate will be I_{OH}/I_{IH} in the logic HIGH state and I_{OL}/I_{IL} in the logic LOW state. The number of logic gate inputs that can be driven from the output of a single logic gate without causing any false output is called fan-out. It is the characteristic of the logic family to which the device belongs. If in a certain case the two values I_{OH}/I_{IH} and I_{OL}/I_{IL} are different, the fan-out is taken as the smaller of the two. Figure 4.39 shows the actual circuit diagram where the output of a single NAND gate belonging to a standard TTL logic family feeds the inputs of multiple NAND gates belonging to the same family when the output of the feeding gate is in the logic HIGH state [Fig. 4.39(a)] and the logic LOW state [Fig. 4.39(b)]. We will learn in Chapter 5 on logic families that the maximum HIGH-state output sourcing current $(I_{OH})_{max}$ and maximum HIGH-state input current $(I_{IH})_{max}$ specifications of standard TTL family devices are 400 μA and 40 μA respectively. Also, the maximum LOW-state output sinking current $(I_{OL})_{max}$ and maximum LOW-state input current $(I_{IL})_{max}$ specifications are 16 mA and 1.6 mA respectively. Considering both the sourcing and sinking capability of standard TTL family devices, we obtain a fan-out figure of 10 both for HIGH and for LOW logic states. If the maximum sourcing and sinking current specifications are exceeded, the output voltage levels in the logic HIGH and LOW states will go out of the specified ranges.

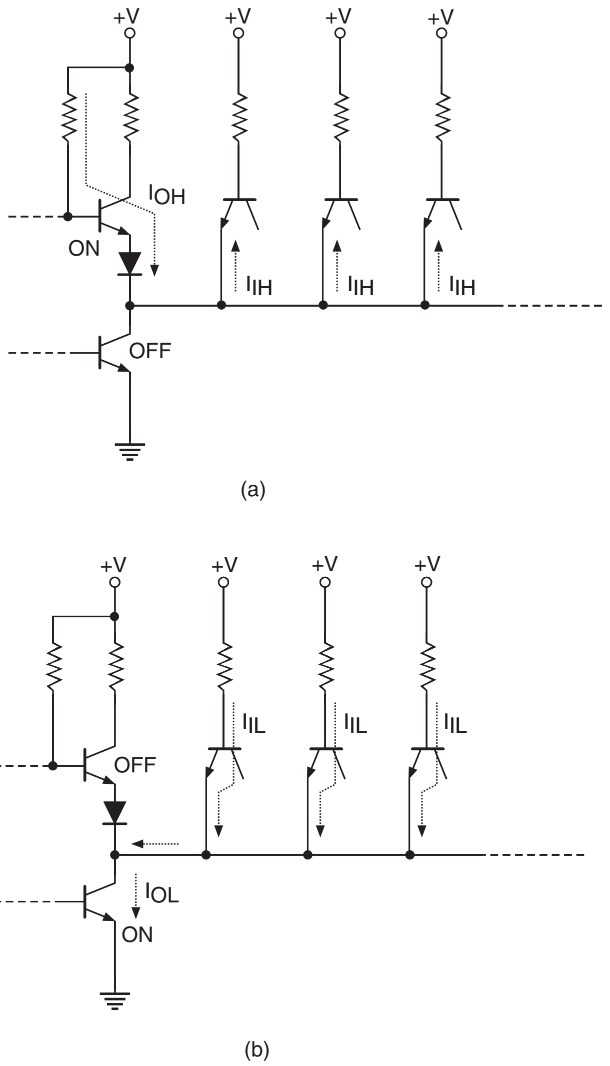


Figure 4.39 Fan-out of the standard TTL logic family.

Example 4.14

A certain logic family has the following input and output current specifications:

1. *The maximum output HIGH-state current = 1 mA.*
2. *The maximum output LOW-state current = 20 mA.*
3. *The maximum input HIGH-state current = 50 μA.*
4. *The maximum input LOW-state current = 2 mA.*

The output of an inverter belonging to this family feeds the clock inputs of various flip-flops belonging to the same family. How many flip-flops can be driven by the output of this inverter providing the clock signal? Incidentally, the data given above are taken from the data sheet of a Schottky TTL family.

Solution

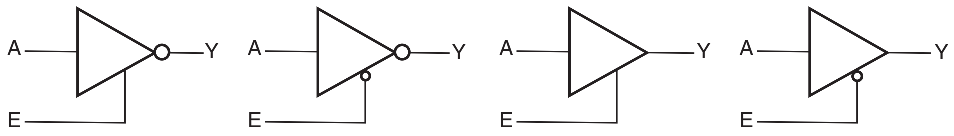
- The HIGH-state fan-out = (1/0.05) = 20 and the LOW-state fan-out = (20/2) = 10.
- Since the lower of the two fan-out values is 10, the inverter output can drive a maximum of 10 flip-flops.

4.11 Buffers and Transceivers

Logic gates, discussed in the previous pages, have a limited load-driving capability. A buffer has a larger load-driving capability than a logic gate. It could be an inverting or noninverting buffer with a single input, a NAND buffer, a NOR buffer, an OR buffer or an AND buffer. ‘Driver’ is another name for a buffer. A driver is sometimes used to designate a circuit that has even larger drive capability than a buffer. Buffers are usually tristate devices to facilitate their use in bus-oriented systems. Figure 4.40 shows the symbols and functional tables of inverting and noninverting buffers of the tristate type.

A transceiver is a bidirectional buffer with additional direction control and ENABLE inputs. It allows flow of data in both directions, depending upon the logic status of the control inputs. Transceivers, like buffers, are tristate devices to make them compatible with bus-oriented systems. Figures 4.41(a) and (b) respectively show the circuit symbols of inverting and noninverting transceivers. Figure 4.42 shows a typical logic circuit arrangement of a tristate noninverting transceiver with its functional table [Fig. 4.42(b)].

Some of the common applications of inverting and noninverting buffers are as follows. Buffers are used to drive circuits that need more drive current. Noninverting buffers are also used to increase the fan-out of a given logic gate. This means that the buffer can be used to increase the number of logic gate inputs to which the output of a given logic gate can be connected. Yet another application of a noninverting buffer is its use as a delay line. It delays the signal by an amount equal to the propagation delay of the device. More than one device can be connected in cascade to get larger delays.



A	E	Y
X	0	Z
0	1	1
1	1	0

A	E	Y
X	1	Z
0	0	1
1	0	0

A	E	Y
X	0	Z
0	1	0
1	1	1

A	E	Y
X	1	Z
0	0	0
1	0	1

Z = High Impedance State

Figure 4.40 (a) Inverting tristate buffers and (b) noninverting tristate buffers.

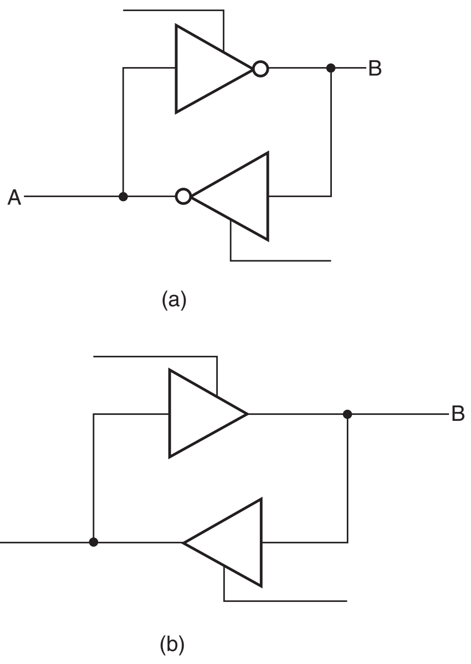


Figure 4.41 (a) Inverting transceivers and (b) noninverting transceivers.

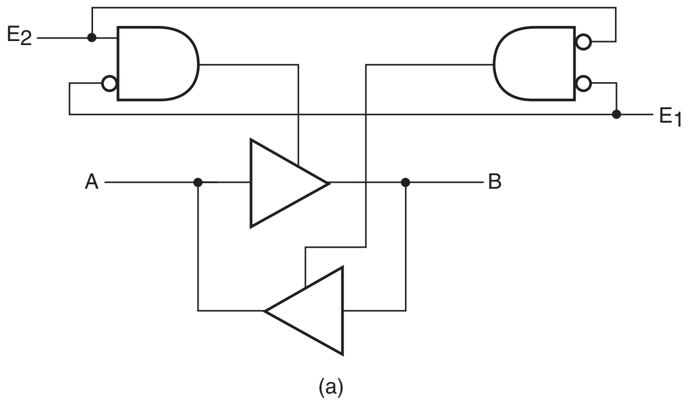


Figure 4.42 Tristate noninverting transceiver.

E_1	E_2	Operation
L	L	Data flow from B to A
L	H	Data flow from A to B
H	X	Isolation

(b)

Figure 4.42 (*continued*).

4.12 IEEE/ANSI Standard Symbols

The symbols used thus far in the chapter for representing different types of gate are the ones that are better known to all of us and have been in use for many years. Each logic gate has a symbol with a distinct shape. However, for more complex logic devices, e.g. sequential logic devices like flip-flops, counters, registers or arithmetic circuits, such as adders, subtractors, etc., these symbols do not carry any useful information. A new set of standard symbols was introduced in 1984 under IEEE/ANSI Standard 91–1984. The logic symbols given under this standard are being increasingly used now and have even started appearing in the literature published by manufacturers of digital integrated circuits. The utility of this new standard will be more evident in the following paragraphs as we go through its salient features and illustrate them with practical examples.

4.12.1 IEEE/ANSI Standards – Salient Features

This standard uses a rectangular symbol for all devices instead of a different symbol shape for each device. For instance, all logic gates (OR, AND, NAND, NOR) will be represented by a rectangular block.

A right triangle is used instead of a bubble to indicate inversion of a logic level. Also, the right triangle is used to indicate whether a given input or output is active LOW. The absence of a triangle indicates an active HIGH input or output. As far as logic gates are concerned, a special notation inside the rectangular block describes the logic relationship between output and inputs. A '1' inside the block indicates that the device has only one input. An AND operation is expressed by '&', and an OR operation is expressed by the symbol '$\geq$1'. Figure 4.43 shows the ANSI counterparts of various logic gates. A '$\geq$1' symbol indicates that the output is HIGH when one or more than one input is HIGH. An '&' symbol indicates that the output is HIGH only when all the inputs are HIGH. The two-input EX-OR is represented by the symbol '=1' which implies that the output is HIGH only when one of its inputs is HIGH.

A special dependency notation system is used to indicate how the outputs depend upon the input. This notation contains almost the entire functional information of the logic device in question. This will be more clear as we illustrate this new standard with the help of ANSI symbols for some of the actual devices belonging to the category of flip-flops, counters, etc., in the following chapters. All those control inputs that control the timing of change in output states as per logic status of inputs are designated by the letter 'C'. These are ENABLE inputs in latches or CLOCK inputs in flip-flops.

Most of the digital ICs contain more than one similar function on one chip such as IC 7400 (quad two-input NAND), IC 7404 (hex inverter), IC 74112 (dual-edge triggered JK flip-flop), IC 7474 (dual D-type latch), IC 7475 (quad D-type latch) and so on. Those inputs to such ICs that are common to

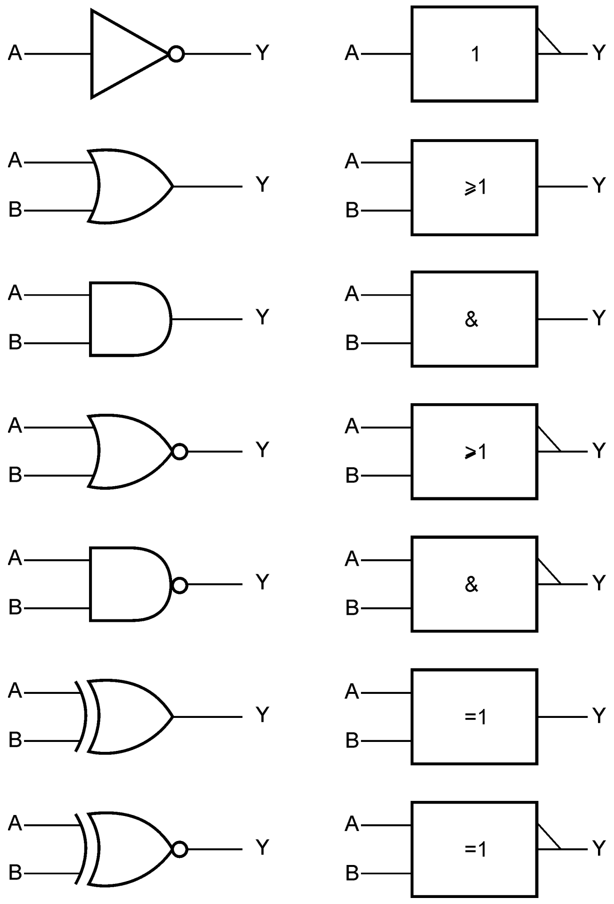


Figure 4.43 IEEE / ANSI symbols.

all the functional blocks or in other words similarly affect various individual but similar functions are represented by a separate notched rectangle on the top of the main rectangle.

4.12.2 ANSI Symbols for Logic Gate ICs

Figure 4.44 shows the ANSI symbol for IC 7400, which is a quad two-input NAND gate. The figure is self-explanatory with the background given in the preceding paragraphs. Any other similar device, i.e. another quad two-input NAND gate belonging to another logic family, would also be represented by the same ANSI symbol. As another illustration, Fig. 4.45 shows the ANSI symbol for IC 7420, which is a dual four-input NAND gate.

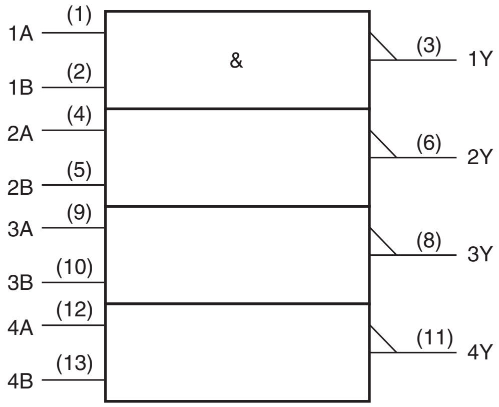


Figure 4.44 ANSI symbol for IC 7400.

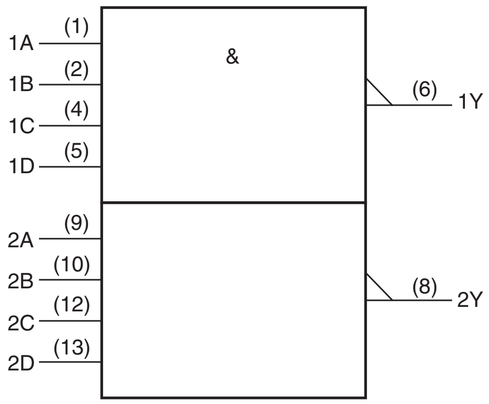


Figure 4.45 ANSI symbol for IC 7420.

Example 4.15

Draw the IEEE/ANSI symbol representation of the logic circuit shown in Fig. 4.46.

Solution

Figure 4.47 shows the circuit using IEEE/ANSI symbols.

4.13 Some Common Applications of Logic Gates

In this section, we will briefly look at some common applications of basic logic gates. The applications discussed here include those where these devices are used to provide a specific function in a larger digital circuit. These also include those where one or more logic gates, along with or without some external components, can be used to build some digital building blocks.

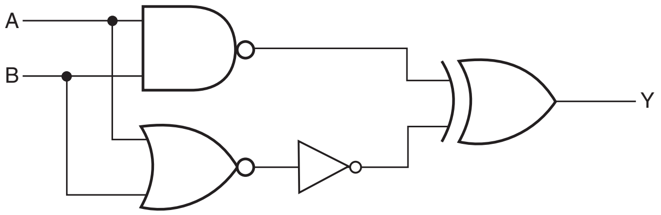


Figure 4.46 Example 4.15.

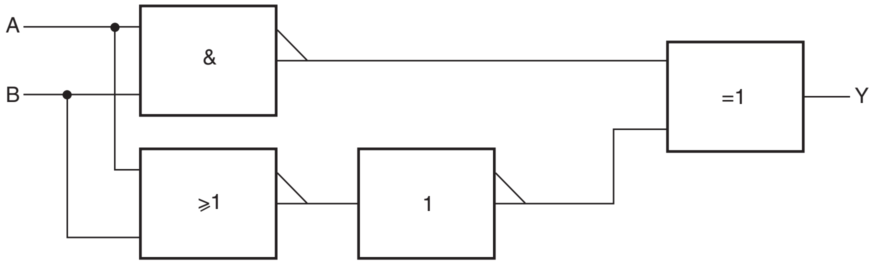


Figure 4.47 Solution to example 4.15.

4.13.1 OR Gate

An OR gate can be used in all those situations where the occurrence of any one or more than one event needs to be detected or acted upon. One such example is an industrial plant where any one or more than one parameter exceeding a preset limiting value should lead to initiation of some kind of protective action. Figure 4.48 shows a typical schematic where the OR gate is used to detect either temperature or pressure exceeding a preset threshold value and produce the necessary command signal for the system.

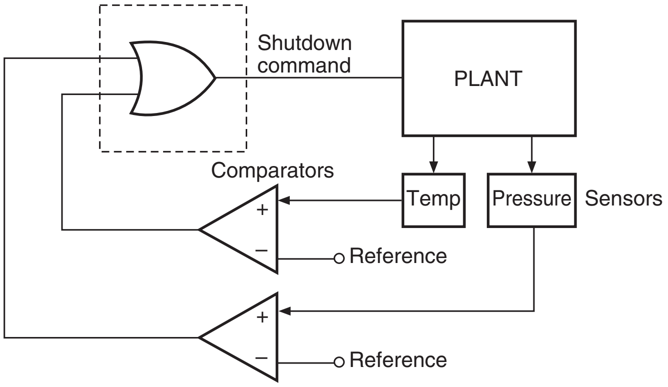


Figure 4.48 Application of an OR gate.

4.13.2 AND Gate

An AND gate is commonly used as an ENABLE or INHIBIT gate to allow or disallow passage of data from one point in the circuit to another. One such application of enabling operation, for instance, is in the measurement of the frequency of a pulsed waveform or the width of a given pulse with the help of a counter. In the case of frequency measurement, a gating pulse of known width is used to enable the passage of the pulse waveform to the counter's clock input. In the case of pulse width measurement, the pulse is used to enable the passage of the clock input to the counter. Figure 4.49 shows the arrangement.

4.13.3 EX-OR/EX-NOR Gate

EX-OR and EX-NOR logic gates are commonly used in parity generation and checking circuits. Figures 4.50(a) and (b) respectively show even and odd parity generator circuits for four-bit data. The circuits are self-explanatory.

The parity check operation can also be performed by similar circuits. Figures 4.51(a) and (b) respectively show simple even and odd parity check circuits for a four-bit data stream. In the circuits shown in Fig. 4.51, a logic '0' at the output signifies correct parity and a logic '1' signifies one-bit error. Parity generator/checker circuits are available in IC form. 74180 in TTL and 40101 in CMOS are nine-bit odd/even parity generator/checker ICs. Parity generation and checking circuits are further discussed in Chapter 7 on arithmetic circuits.

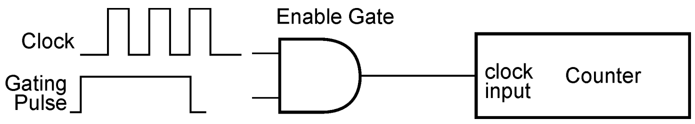


Figure 4.49 Application of an AND gate.

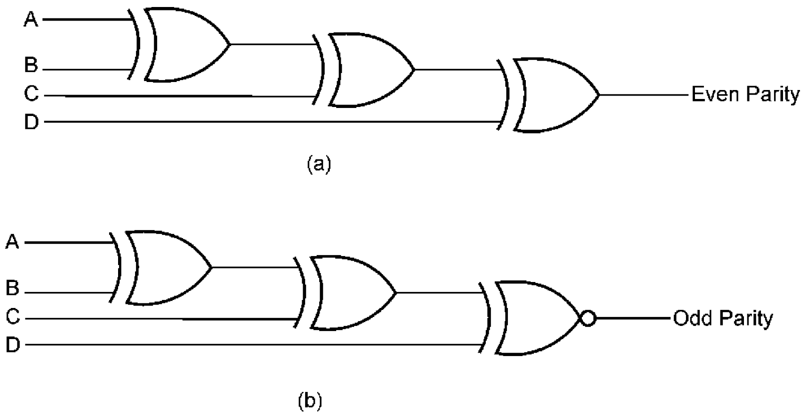
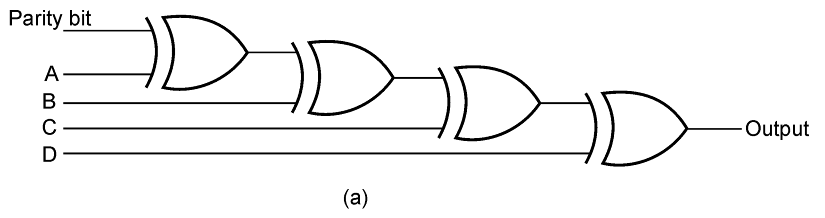


Figure 4.50 Parity generation using EX-OR/EX-NOR gates.



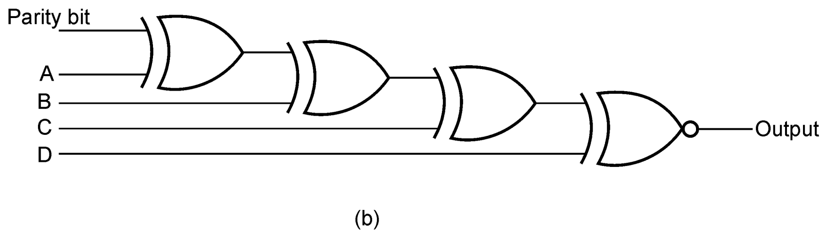


Figure 4.51 Parity check using EX-OR and EX-NOR gates.

4.13.4 Inverter

CMOS inverters are commonly used to build square-wave oscillators for generating clock signals. These clock generators offer good stability, operation over a wide supply voltage range (3–15 V) and frequency range (1 Hz to in excess of 15 MHz), low power consumption and an easy interface to other logic families.

The most fundamental circuit is the ring configuration of any odd number of inverters. Figure 4.52 shows one such circuit using three inverters. Inverting gates such as NAND and NOR gates can also be used instead. This configuration does not make a practical oscillator circuit as its frequency of oscillation is highly susceptible to variation with temperature, supply voltage and external loading. The frequency of oscillation is given by the equation

$$f = 1/(2nt_p) \tag{4.12}$$

where n is the number of inverters and t_p is the propagation delay per gate.

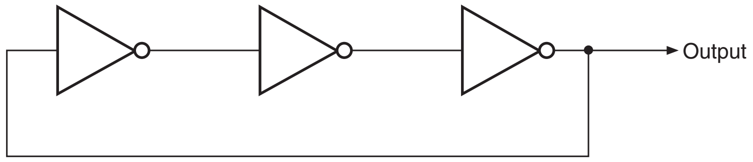


Figure 4.52 Square-wave oscillator using a ring configuration.

Figure 4.53(a) shows a practical oscillator circuit. The frequency of oscillation in this case is given by Equation (4.13) (the duty cycle of the waveform is approximately 50 %):

$$f = 1/2C(0.405R_{eq} + 0.693R_1) \tag{4.13}$$

where $R_{eq} = R_1.R_2/(R_1 + R_2)$.

Figure 4.53(b) shows another circuit using two inverters instead of three inverters. The frequency of oscillation of this circuit is given by the equation

$$f = 1/2.2RC \tag{4.14}$$

The circuits shown in Fig. 4.53 are not as sensitive to supply voltage variations as the one shown in Fig. 4.52. Figure 4.54 shows yet another circuit that is configured around a single Schmitt inverter. The capacitor charges (when the output is HIGH) and discharges (when the output is LOW) between the

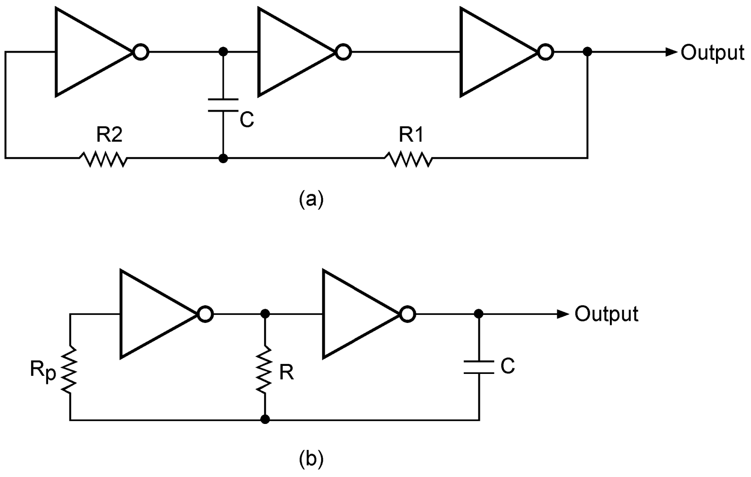


Figure 4.53 Square-wave oscillator with external components.

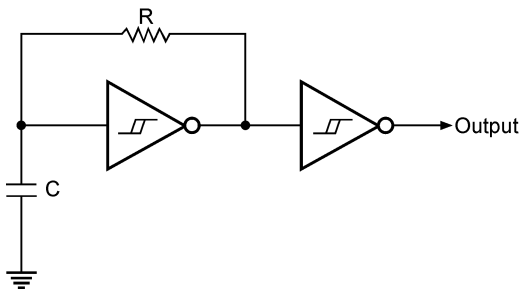


Figure 4.54 Schmitt inverter based oscillator.

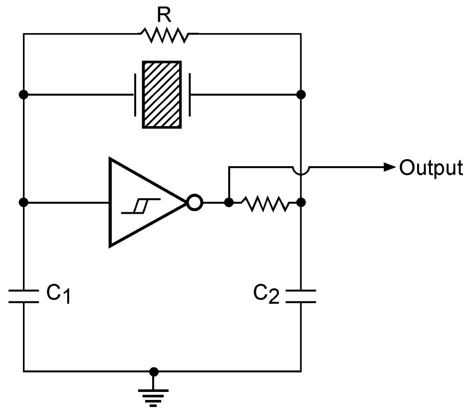


Figure 4.55 Crystal oscillator.

two threshold voltages. The frequency of oscillation, however, is sensitive to supply voltage variations. It is given by the equation

$$f = 1/RC \tag{4.15}$$

Figure 4.55 shows a crystal oscillator configured around a single inverter as the active element. Any odd number of inverters can be used. A larger number of inverters limits the highest attainable frequency of oscillation to a lower value.

4.14 Application-Relevant Information

Table 4.1 lists the commonly used type numbers along with the functional description and the logic family. The pin connection diagrams and the functional tables of the more popular type numbers are given in the companion website.

Table 4.1 Functional index of logic gates.

Type number	Function	Logic family
7400	Quad two-input NAND gate	TTL
7401	Quad two-input NAND gate (open collector)	TTL
7402	Quad two-input NOR gate	TTL
7403	Quad two-input NAND gate (open collector)	TTL
7404	Hex inverter	TTL
7405	Hex inverter (open collector)	TTL
7408	Quad two-input AND gate	TTL
7409	Quad two-input AND gate (open collector)	TTL
7410	Triple three-input NAND gate	TTL

(*Continued overleaf*)

Table 4.1 *(continued).*

Type number	Function	Logic family
7411	Triple three-input AND gate	TTL
7412	Triple three-input NAND gate (open collector)	TTL
7413	Dual four-input Schmitt NAND gate	TTL
7414	Hex Schmitt trigger inverter	TTL
7418	Dual four-input Schmitt NAND gate	TTL
7419	Hex Schmitt trigger inverter	TTL
7420	Dual four-input NAND gate	TTL
7421	Dual four-input AND gate	TTL
7422	Dual four-input NAND gate (open collector)	TTL
7427	Triple three-input NOR gate	TTL
7430	Eight-input NAND gate	TTL
7432	Quad two-input OR gate	TTL
7451	Dual two-wide two-input three-input AND-OR-INVERT gate	TTL
7454	Four-wide two-input AND-OR-INVERT gate	TTL
7455	Two-wide four-input AND-OR-INVERT gate	TTL
7486	Quad two-input EX-OR gate	TTL
74125	Quad tristate noninverting buffer (LOW ENABLE)	TTL
74126	Quad tristate noninverting buffer (HIGH ENABLE)	TTL
74132	Quad two-input Schmitt trigger NAND gate	TTL
74133	13-input NAND gate	TTL
74136	Quad two-input EX-OR gate (open collector)	TTL
74240	Octal tristate inverting bus/line driver	TTL
74241	Octal tristate bus/line driver	TTL
74242	Quad tristate inverting bus transceiver	TTL
74243	Quad tristate noninverting bus transceiver	TTL
74244	Octal tristate noninverting driver	TTL
74245	Octal tristate noninverting bus transceiver	TTL
74266	Quad two-input EXCLUSIVE-NOR gate (open collector)	TTL
74365	Hex tristate noninverting buffer with common ENABLE	TTL
74366	Hex tristate inverting buffer with common ENABLE	TTL
74367	Hex tristate noninverting buffer, four-bit and two-bit	TTL
74368	Hex tristate inverting buffer, four-bit and two-bit	TTL
74386	Quad two-input EX-OR gate	TTL
74465	Octal tristate noninverting buffer Gated ENABLE inverted	TTL
74540	Octal tristate inverting buffer/line driver	TTL
74541	Octal tristate noninverting buffer/line driver	TTL
74640	Octal tristate inverting bus transceiver	TTL
74641	Octal tristate noninverting bus transceiver (open collector)	TTL
74645	Octal tristate noninverting bus transceiver	TTL
4001B	Quad two-input NOR gate	CMOS
4002B	Dual four-input NOR gate	CMOS
4011B	Quad two-input NAND gate	CMOS
4012B	Dual four-input NAND gate	CMOS
4023B	Triple three-input NAND gate	CMOS
4025B	Triple three-input NOR gate	CMOS
4030B	Quad two-input EX-OR gate	CMOS
4049B	Hex inverting buffer	CMOS

Table 4.1 *(continued).*

Type number	Function	Logic family
4050B	Hex noninverting buffer	CMOS
40097B	Tristate hex noninverting buffer	CMOS
40098B	Tristate inverting buffer	CMOS
4069UB	Hex inverter	CMOS
4070B	Quad two-input EX-OR gate	CMOS
4071B	Quad two-input OR gate	CMOS
4081B	Quad two-input AND gate	CMOS
4086B	Four-wide two-input AND-OR-INVERT gate	CMOS
4093B	Quad two-input Schmitt NAND	CMOS
10100	Quad two-input NOR gate with strobe	ECL
10101	Quad two-input OR/NOR gate	ECL
10102	Quad two-input NOR gate	ECL
10103	Quad two-input OR gate	ECL
10104	Quad two-input AND gate	ECL
10113	Quad two-input EX-OR gate	ECL
10114	Triple line receiver	ECL
10115	Quad line Receiver	ECL
10116	Triple Line receiver	ECL
10117	Dual two-wide two- to three-input OR-AND/OR-AND-INVERT gate	ECL
10118	Dual two-wide three-input OR-AND gate	ECL
10123	Triple 4-3-3 input bus driver	
10128	Dual bus driver	ECL
10129	Quad bus driver	ECL
10188	Hex buffer with ENABLE	ECL
10192	Quad bus driver	ECL
10194	Dual simultaneous transceiver	ECL
10195	Hex buffer with invert/noninvert control	ECL

Review Questions

1. How do you distinguish between positive and negative logic systems? Prove that an OR gate in a positive logic system is an AND gate in a negative logic system.
2. Give brief statements that would help one remember the truth table of AND, NAND, OR, NOR, EX-OR and EX-NOR logic gate functions, irrespective of the number of inputs used.
3. Why are NAND and NOR gates called universal gates? Justify your answer with the help of examples.
4. What are Schmitt gates? How does a Schmitt gate overcome the problem of occurrence of an erratic output for slow varying input transitions?
5. What are logic gates with open collector or open drain outputs? What are the major advantages and disadvantages of such devices?
6. Draw the circuit symbol and the associated truth table for the following:

   (a) a tristate noninverting buffer with an active HIGH ENABLE input;
   (b) a tristate inverting buffer with an active LOW ENABLE input;

(c) a three-input NAND with an open collector output;
(d) a four-input INHIBIT gate.

7. Define the fan-out specification of a logic gate. Which parameters would you need to know from the data sheet of a logic gate to determine for yourself the fan-out in case it is not mentioned in the data sheet? Explain the procedure for determining the fan-out specification from those parameters. What are the consequences of exceeding the fan-out specification?
8. What is the main significance of IEEE/ANSI symbols when compared with the conventional ones? Draw the ANSI symbols for four-input OR, two-input AND, two-input EX-OR and two-input NAND gates.

Problems

1. What is the only input combination that:

(a) Will produce a logic '1' at the output of an eight-input AND gate?
(b) Will produce a logic '0' at the output of a four-input NAND gate?
(c) Will produce a logic '1' at the output of an eight-input NOR gate?
(d) Will produce a logic '0' at the output of a four-input OR gate?

(a) 11111111; (b) 1111; (c) 00000000; (d) 0000

2. Draw the truth table of the logic circuit shown in Fig. 4.56.

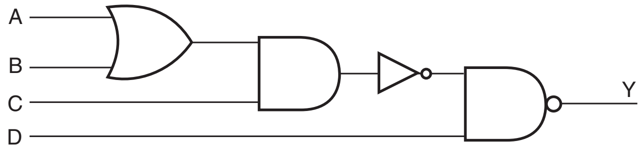


Figure 4.56 Problem 2.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Figure 4.57 Solution of problem 2.

3. Redraw the logic circuit of Fig. 4.56 using IEEE/ANSI symbols.

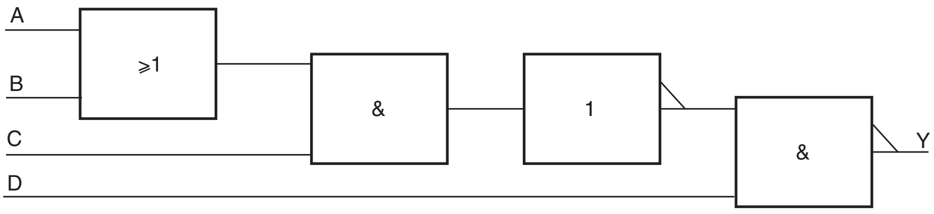
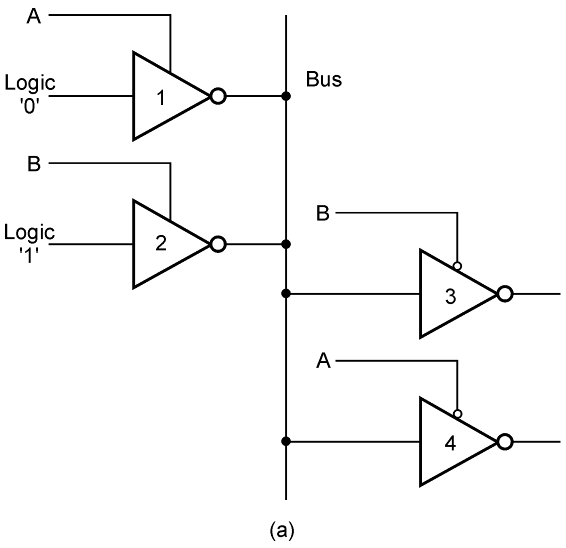


Figure 4.58 Solution to problem 3.

4. Refer to Fig. 4.59(a). The ENABLE waveforms applied at A and B inputs are respectively shown in Figs 4.59(b) and (c). What is the output state of inverter 3 and inverter 4 at (i) $t = 3$ ms and (ii) $t = 5$ ms?

(i) The output of inverter 3 = high Z, while the output of inverter 4 = logic '1'
(ii) The output of inverter 3 = logic '0', while the output of inverter 4 = high Z



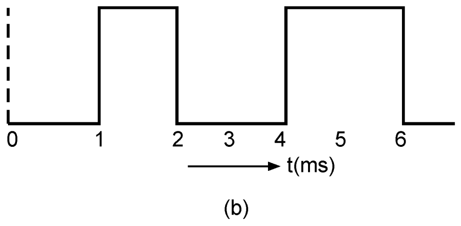
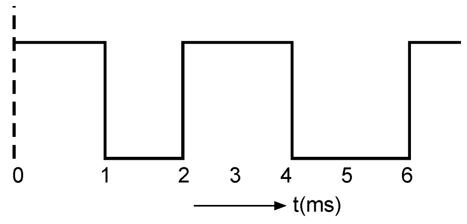
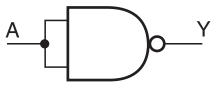


Figure 4.59 Problem 4.

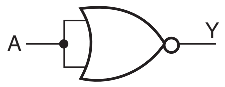


(c)

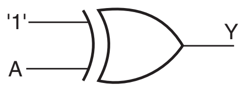
Figure 4.59 (*Continued*)



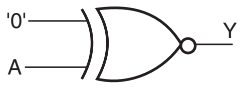
(a)



(b)



(c)

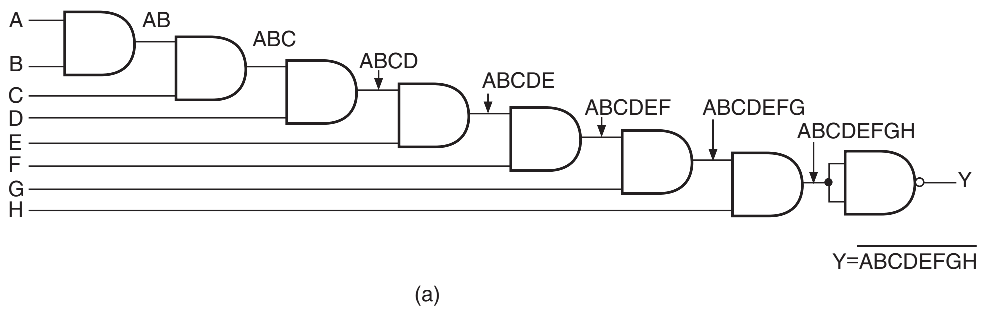


(d)

Figure 4.60 Solution to problem 5.

5. Draw logic implementation of an inverter using (i) two-input NAND, (ii) two-input NOR, (iii) two-input EX-OR and (iv) two-input EX-NOR.

(i) Fig. 4.60(a); (ii) Fig. 4.60(b); (iii) Fig. 4.60(c); (iv) Fig. 4.60(d)



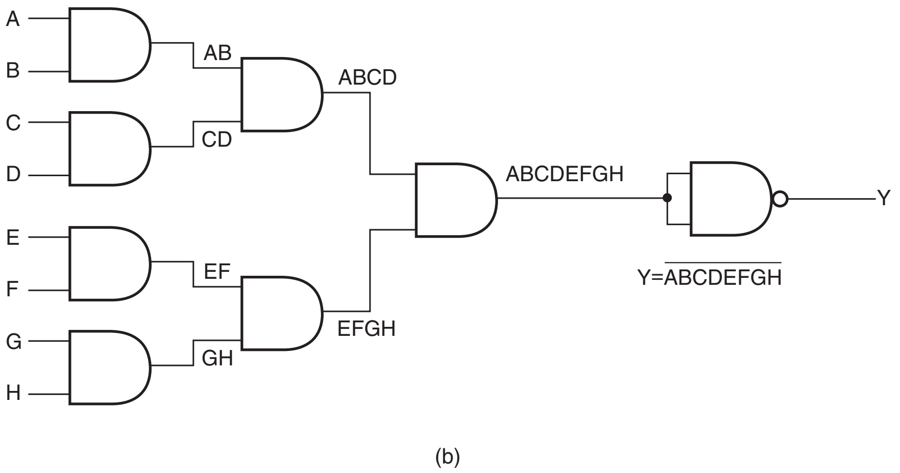


Figure 4.61 Solution to problem 6.

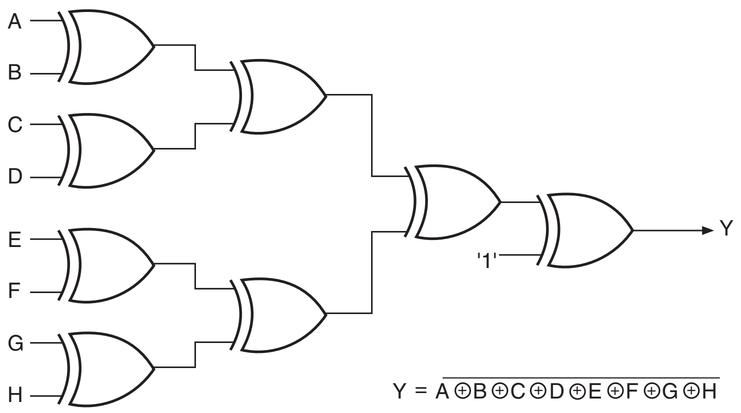

A
B
C
D
E
F
G
H
'1'
Y
Y = $\overline{A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G \oplus H}$

Figure 4.62 Solution to problem 7.

6. It is proposed to construct an eight-input NAND gate using only two-input AND gates and two-input NAND gates. Draw the logic arrangement that uses the minimum number of logic gates.
   The two possible logic circuits are shown in Figs 4.61(a) and (b)

7. Draw the logic diagram to implement an eight-input EX-NOR function using the minimum number of two-input logic gates.

Further Reading

1. Cook, N. P. (2003) *Practical Digital Electronics*, Prentice-Hall, NJ, USA.
2. Fairchild Semiconductor Corporation (October 1974) *CMOS Oscillators*, Application Note 118, South Portland, ME, USA.
3. Holdsworth, B. and Woods, C. (2002) *Digital Logic Design*, Newnes, Oxford, UK.
4. Langholz, G., Mott, J. L. and Kandel, A. (1998) *Foundations of Digital Logic Design*, World Scientific Publ. Co. Inc., Singapore.
5. Chen, W.-K. (2003) *Logic Design*, CRC Press, FL, USA.