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 EXCLUSIVEOR 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.

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'.

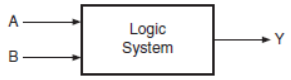
If the two voltage levels are 0 V and −5 V, then in the positive logic system the 0 V represents a logic '1' and the −5 V represents a logic '0'. In the negative logic system, 0 V represents a logic '0' and −5 V represents a logic '1'.

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 2n 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 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

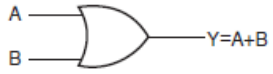
4.3 Logic Gates

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.



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

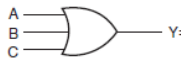
Figure 4.3 Two-input OR gate.

The operation of a two-input OR gate is explained by the logic expression

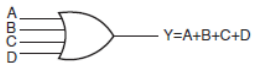
Y = A+B… (4.1)

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.

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.



(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.

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 Y 3 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

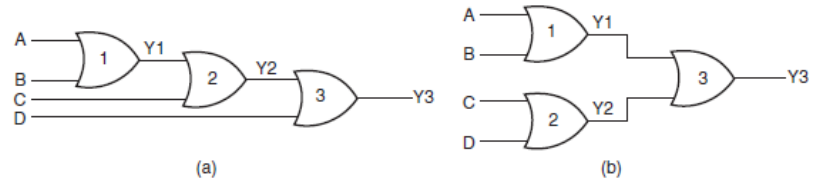


Figure 4.5 Example 4.1.

OR gate produces the output Y 2 = (Y 1+C) = (A+B+C). Similarly, the output of OR gate 3 is Y 3 = (Y 2+D) = (A+B+C+D). In the case of Fig. 4.5(b), the output of OR gate 1 is Y 1 = (A+B). The second OR gate produces the output Y 2 = (C +D). Output Y 3 of the third OR gate is given by (Y 1+Y 2) = (A+B+C +D).

Example 4.2

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

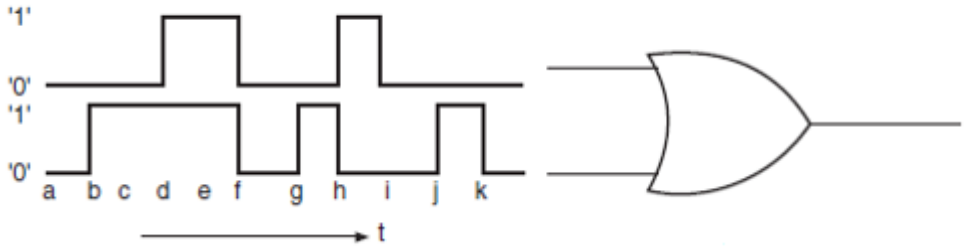


Fig 4,6 (a)

Solution

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

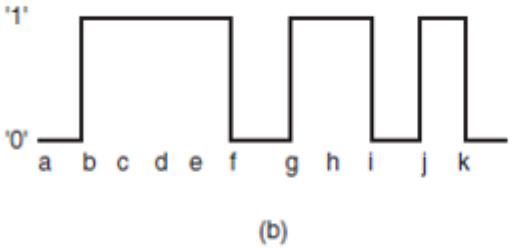


Fig 4.6 (b)

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.

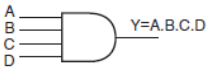
The AND operation on two independent logic variables A and B is written as Y = A.B and reads as Y equals A AND B and not as A multiplied by B. Here, A and B are input logic variables and Y is the output. An AND gate performs an ANDing operation:

(a)

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

(b)

Figure 4.7 Two-input AND gate.



(b)

(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	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 Y 1 = A.B. The second AND gate produces an output Y 2 given by

Y 2 = Y 1.C = A.B.C. Similarly, the output of AND gate 3 is Y = Y 2.D = A.B.C.D and hence the result.

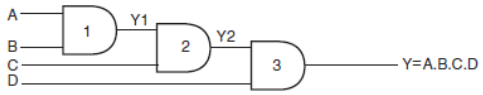
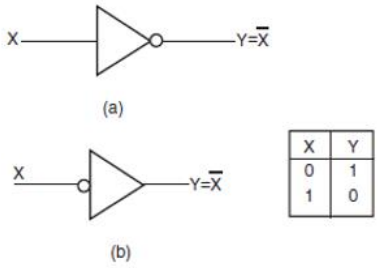


Figure 4.9 Implementation of a four-input AND gate using two-input AND gates.

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



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.

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.

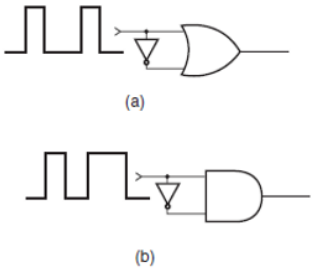
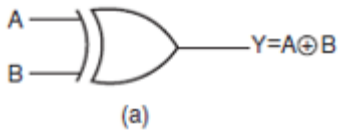


Figure 4.11 Example 4.4.

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.



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

(b)

4.12(a) and (b)

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.

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(c)

The output of a two-input EX-OR gate is expressed by

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

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 Y 1 is given by A ⊕ B. The final output Y is given by Y = (Y 1⊕C)= (A⊕B)⊕C = A⊕B⊕C.

• Figure 4.13(b) can be explained on similar lines

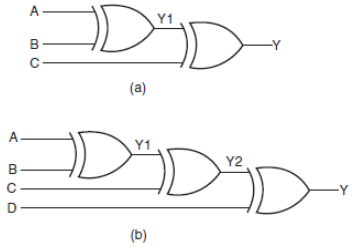


Figure 4.13 (a) Three-input EX-OR gate and (b) a four-input EX-OR gate.

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)

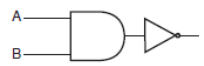
(b)

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

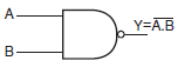
4.3.5 NAND Gate

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)].

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



(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.

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

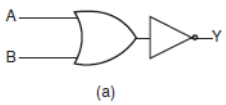
$Y = \underline{A.B}$ (4.3)

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

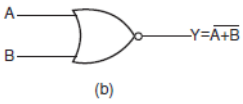
$$Y = \overline{(A.B.C.D...)} \qquad (4.4)$$

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



(a)



(b)

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

(c)

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

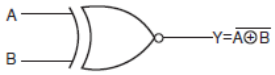
$$Y = \overline{(A + B + C + D...)} \qquad (4.6)$$

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,

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



(a)

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.

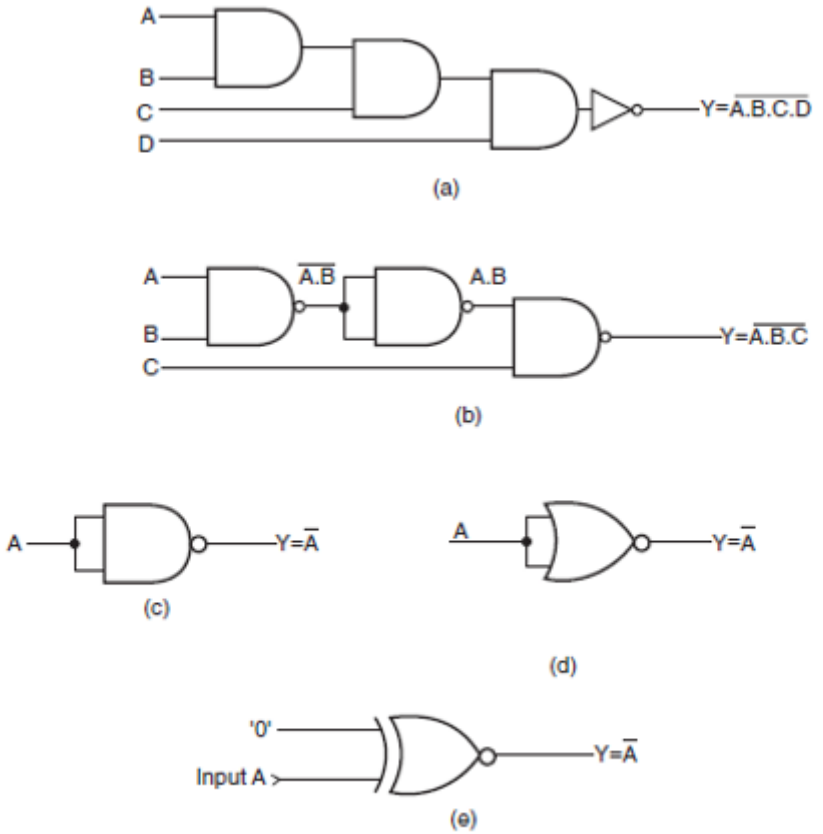


Figure 4.18 Example 4.7.

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.

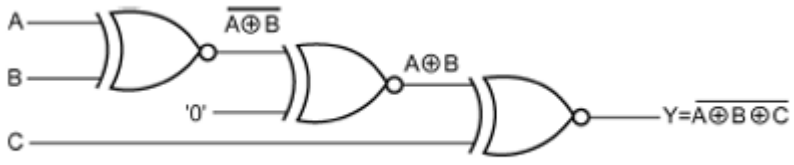


Figure 4.19 Example 4.8.

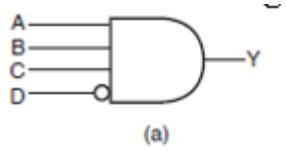
4.3.8 INHIBIT Gate

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.



(a)

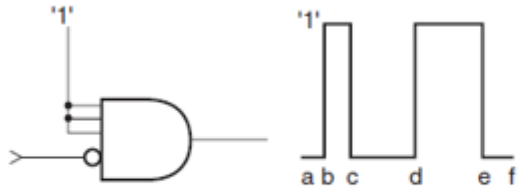
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

(b)

Figure 4.20 INHIBIT gate.

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.



(a) (b)

Figure 4.21 Example 4.9.

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.

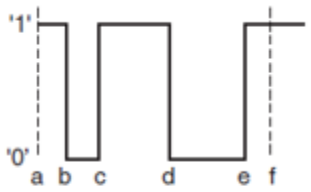


Figure 4.22 Solution to example 4.9.

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.

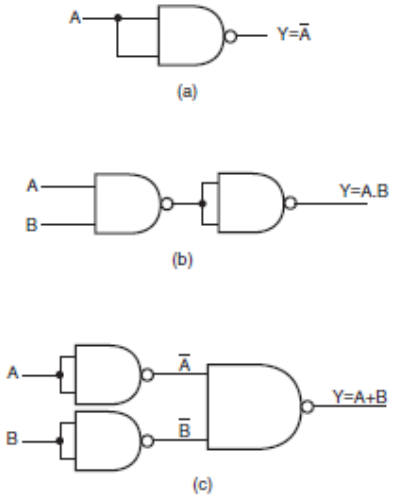


Figure 4.24 Implementation of basic logic gates using only NAND gates.

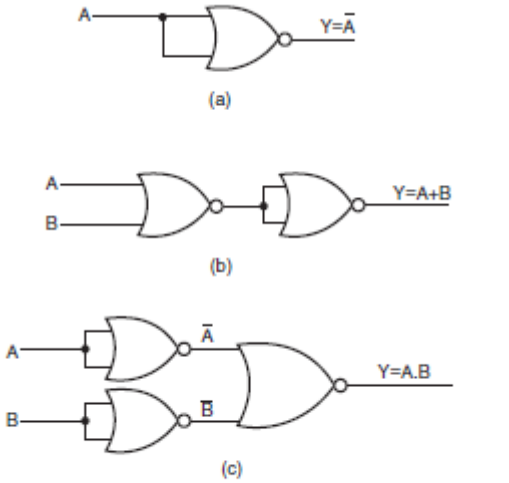


Figure 4.25 Implementation of basic logic gates using only NOR gates.