TABLE 1.4 Symbolic Representations of Standard Boolean Operators

Boolean Operation	Operators
AND	*, &
OR	+, |, #
XOR	⊕, ^
NOT	!, ~, $\overline{A}$

inverting the base function. Two equally valid ways of representing NAND are $Y = \overline{A \,\&\, B}$ and Y = !(AB). Similarly, an XNOR might be written as $Y = \overline{A \oplus B}$.

When logical functions are converted into circuits, graphical representations of the seven basic operators are commonly used. In circuit terminology, the logical operators are called *gates*. Figure 1.1 shows how the basic logic gates are drawn on a circuit diagram. Naming the inputs of each gate A and B and the output Y is for reference only; any name can be chosen for convenience. A small bubble is drawn at a gate's output to indicate a logical inversion.

More complex Boolean functions are created by combining Boolean operators in the same way that arithmetic operators are combined in normal mathematics. Parentheses are useful to explicitly convey precedence information so that there is no ambiguity over how two variables should be treated. A Boolean function might be written as

$$Y = (AB + \overline{C} + D) \& \overline{E \oplus F}$$

This same equation could be represented graphically in a circuit diagram, also called a *schematic diagram*, as shown in Fig. 1.2. This representation uses only two-input logic gates. As already mentioned, binary operators can be chained together to implement functions of more than two variables.

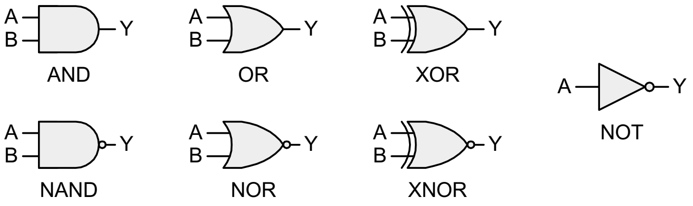

FIGURE 1.1 Graphical representation of basic logic gates.

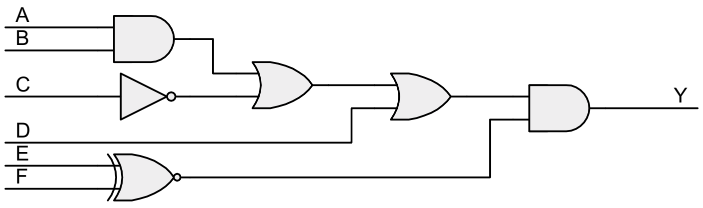

FIGURE 1.2 Schematic diagram of logic function.

An alternative graphical representation would use a three-input OR gate by collapsing the two-input OR gates into a single entity.

1.2 BOOLEAN MANIPULATION

Boolean equations are invaluable when designing digital logic. To properly use and devise such equations, it is helpful to understand certain basic rules that enable simplification and re-expression of Boolean logic. Simplification is perhaps the most practical final result of Boolean manipulation, because it is easier and less expensive to build a circuit that does not contain unnecessary components. When a logical relationship is first set down on paper, it often is not in its most simplified form. Such a circuit will function but may be unnecessarily complex. Re-expression of a Boolean equation is a useful skill, because it can enable you to take better advantage of the logic resources at your disposal instead of always having to use new components each time the logic is expanded or otherwise modified to function in a different manner. As will soon be shown, an OR gate can be made to behave as an AND gate, and vice versa. Such knowledge can enable you to build a less-complex implementation of a Boolean equation.

First, it is useful to mention two basic identities:

$$A \,\&\, \overline{A} = 0 \text{ and } A + \overline{A} = 1$$

The first identity states that the product of any variable and its logical negation must always be false. It has already been shown that both operands of an AND function must be true for the result to be true. Therefore, the first identity holds true, because it is impossible for both operands to be true when one is the negation of the other. The second identity states that the sum of any variable and its logical negation must always be true. At least one operand of an OR function must be true for the result to be true. As with the first identity, it is guaranteed that one operand will be true, and the other will be false.

Boolean algebra also has commutative, associative, and distributive properties as listed below:

- Commutative: $A \,\&\, B = B \,\&\, A$ and $A + B = B + A$
- Associative: $(A \,\&\, B) \,\&\, C = A \,\&\, (B \,\&\, C)$ and $(A + B) + C = A + (B + C)$
- Distributive: $A \,\&\, (B + C) = A \,\&\, B + A \,\&\, C$

The aforementioned identities, combined with these basic properties, can be used to simplify logic. For example,

$$A \,\&\, B \,\&\, C + A \,\&\, B \,\&\, \overline{C}$$

can be re-expressed using the distributive property as

$$A \,\&\, B \,\&\, (C + \overline{C})$$

which we know by identity equals

$$A \,\&\, B \,\&\, (1) = A \,\&\, B$$

Another useful identity, $A + \overline{A}B = A + B$, can be illustrated using the truth table shown in Table 1.5.