Digital Logic and Combinational Circuits

CHAPTER 8

8.1 INTRODUCTION

We have discussed so far linear circuits, where the signals have a continuous range of values, known as the *analog signal*. Analog signal processing and displays are normally done using various linear circuits. With the advent of microelectronics, microprocessors and digital computers, the emphasis has now shifted to digital processing, including control and display. The real-life signals are analog and will remain so. Analog signals are obtained from transducers. The digital circuits interface with linear circuits from analog to digital converters. Once an electrical signal is converted into digital form, further processing is done digitally.

In contrast to an analog signal, a *digital signal* exists only at specific levels or states and changes its level in discrete steps. In this chapter, we deal with digital signals having only two states. The two-state system allows the application of Boolean logic and binary number representation, which form the foundation for the design of all digital devices and circuits. These two levels of 0 and 1 may represent levels of on or off, open or closed, yes or no, true or false. *Binary signals* have the great advantage of being far less susceptible to noise disturbance compared to analog signals.

In general, digital signals or numbers are processed by means of digital systems using the concepts of binary numbers and Boolean algebra. The operation of a digital system is controlled by logic gate. In this chapter, we shall study various logic gates, namely NOT, OR, AND, NAND, NOR and Ex-OR. Boolean algebra defines theorems which are used to simplify the logic statements. We shall describe here logical functions and circuits, which implement these logical functions. We shall also deal with switching applications of BJT and MOS devices, and the implementation of different logic gates using BJT, NMOS and CMOS. Besides, we shall cover various logic families such as DTL, RTL, TTL and CMOS here.

The term *combinational logic* is used for combining two or more basic logic gates to form a required function where the output at a given time depends only on the input. In addition to a discussion of combinational logic, this chapter includes arithmetic operation of logic gates by connecting them in certain combinations.

8.2 BINARY NUMBER SYSTEMS AND CODES

We are very much familiar with the *decimal number system* with base 10. The *base* of the number system indicates the number of different symbols required for representing whole numbers and fractions. In base 10, the symbols are: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. To represent numbers greater than 9, the digits are arranged by columns on the left of a decimal point, each column having a different weight or multiplying factor in different powers of 10 according to

$$d_n d_{n-1} \cdots d_2 d_1 d_0 = d_n 10^n + d_{n-1} 10^{n-1} + \cdots + d_2 10^2 + d_1 10^1 + d_0 10^0 \quad (8.1)$$

where each digit d_i is one of the 10 symbols. As an example,

$$4231 = 4 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 1 \times 10^0 \quad (8.2)$$

Fractions (numbers less than 1) may also be included if digits for negative powers of 10 are included (d_{-1}, d_{-2}, . . .). For example, the number 231.65 is written as

$$231.65_{10} = 2 \times 10^2 + 3 \times 10^1 + 1 \times 10^0 + 6 \times 10^{-1} + 5 \times 10^{-2} \quad (8.3)$$

The binary number system is the most important one in the digital system. 'Binary' means 'two'. The binary number system uses only two digits 0 and 1 in contrast to 10 digits of the decimal system. These two digits are termed *binary digits* (*bits*). The *base* (sometimes called *radix*) for this system is 2 because the operation of digital devices is based on transistors that switch between two states: the ON or saturated state and OFF or cut-off state. These states are designated by the symbols 1 (ON) and 0 (OFF) in the base 2 system. The digits in a binary number correspond to different powers of the base 2. A binary number can be expanded as

$$d_n d_{n-1} \cdots d_2 d_1 d_0 = d_n 2^n + d_{n-1} 2^{n-1} + \cdots + d_2 2^2 + d_1 2^1 + d_0 2^0 \quad (8.4)$$

where each d_i is one of the two symbols 0 and 1. As an example of Eq. (8.4), the binary number 1110 can be expanded as

$$1110_2 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8_{10} + 4_{10} + 2_{10} + 0_{10} = 14_{10} \quad (8.5)$$

In a binary number system, the first or the left-most bit is known as the *most significant bit* (MSB) since it represents the largest power of 2. The last or the right-most bit is known as the *least significant bit* (LSB) since it represents the smallest power of 2.

8.2.1 Binary-to-decimal Conversion

Binary-to-decimal conversion process has already been illustrated above, as in Eq. (8.5). It is shown that each bit carries a certain weight (power of 2) based on its position relative to the LSB. Any binary number can be converted to its decimal equivalent simply by summing together the weights of the various positions in the binary number, which contain a 1. For example, to convert 10111_2 to decimal, we write

$$10111_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 0 + 4 + 2 + 1 = 23_{10} \quad (8.6)$$

The MSB has a weight 2^4 even though it is the fifth bit because the LSB is the first bit and has a weight 2^0. For mixed binary number, the positions to the left or right of the binary point carry weights increasing or decreasing in powers of 2. For example, the number

$$\begin{aligned} 1101.1101_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} \\ &= 8_{10} + 4_{10} + 1_{10} + 0.5_{10} + 0.25_{10} + 0.062_{10} = 13.812_{10} \end{aligned} \quad (8.7)$$

8.2.2 Decimal-to-binary Conversion

To convert a decimal number (base 10) to binary number (base 2), the decimal number is successively divided by base 2 and the remainders are recorded after each division until a quotient of 0 is

obtained. The remainders, when written in reverse order from left to right, form the binary number. To illustrate the procedure, the decimal number 123 is converted into its binary equivalent in Table 8.1.

Table 8.1 Decimal-to-binary Conversion

Successive division	Quotient	Remainder
123/2	61	1 LSB
61/2	30	1
30/2	15	0
15/2	7	1
7/2	3	1
3/2	1	1
1/2	0	↑ 1 MSB
Result:	$123_{10} = 1111011_2$	

Fractional part of a decimal number is converted into binary by repeated multiplication of fractional part by 2 until the result is a full integer number. The integer numbers obtained after multiplication are recorded in forward order (from left to right). An example of conversion of mixed number, 59.4375 (integer and fraction) to its binary equivalent is shown below.

Following the procedure mentioned above, integer part is converted into binary number and we have found that

$$59_{10} = 111011_2$$

Fractional part is converted as under:

$$0.4375 = 0.4375 \times 2 = 0.8750 \rightarrow \text{integer is } 0 \downarrow$$
$$= 0.8750 \times 2 = 1.7500 \rightarrow \text{integer is } 1$$
$$= 0.7500 \times 2 = 1.5000 \rightarrow \text{integer is } 1$$
$$= 0.5000 \times 2 = 1.0000 \rightarrow \text{integer is } 1$$

Thus, $0.4375_{10} = .0111$, and the decimal number $59.4375_{10} = 111011.0111_2$.

8.2.3 Hexadecimal and Octal Number System

Since binary numbers can be long and cumbersome to write and display, often the hexadecimal ('base 16') and Octal ('base 8') number systems are used as an alternative representation. The octal number system is very important in digital computer work and it has a base of 8, meaning that it has eight possible digits: 0, 1, 2, 3, 4, 5, 6, 7. The hexadecimal number system expresses the binary number further concisely and is commonly used in computers. It has 16 possible digit symbols: 0 through 9 plus the letters *A, B, C, D, E, F*. It should be noted that the letters from *A* to *F* are used to represent the digits larger than 9. Table 8.2 shows the relationships among symbols for the octal, hexadecimal, binary and decimal number systems.

Table 8.2 Decimal, Binary and Hexadecimal Equivalents

Binary	Hexadecimal	Decimal	Octal
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	8	8	10
1001	9	9	11
1010	*A*	10	12
1011	*B*	11	13
1100	*C*	12	14
1101	*D*	13	15
1110	*E*	14	16
1111	*F*	15	17

In general, the value of a number represented in any base can be expanded and computed with

$$(d_n d_{n-1} \cdots d_2 d_1 d_0)_b = d_n b^n + d_{n-1} b^{n-1} + \cdots + d_2 b^2 + d_1 b^1 + d_0 b^0 \qquad (8.8)$$

where b is the base. It is often necessary to convert from one base system to another. Equation (8.2) provides a mechanism to convert a number with base 2 into that of base 10.

For binary-to-octal conversion, the bits of the binary numbers are grouped into groups of three bits starting at the LSB. Then each group is converted to its octal equivalent. Similarly, for binary-to-hexadecimal conversion, the binary number is grouped into groups of four bits and each group is converted to its equivalent hex digit. Zeros are added as needed to complete a 4-bit group.

8.2.4 Binary Coded Decimal (BCD) Code

A group of symbols is called a *code* when they represent certain numbers or letters or words. One of the most familiar codes is the *Morse code*, where series of dots and dashes represent letters of the alphabet. The group of 0's and 1's in the binary number can be thought of as a code representing the decimal number. Thus, a code is simply a system of symbols by means of which meaningful communication can be effected. In computer and microprocessor applications, the conversion of decimal-to-binary and binary-to-decimal are performed often. We have seen that conversion of large decimal number to binary becomes long and complicated. For this reason, a method of encoding decimal numbers, which combines some features of both the decimal and binary systems, is used in certain situations. *Binary coded decimal* is a digital representation where each digit of a decimal number is represented by its binary equivalent. Since a decimal digit can be as large as

9, four bits are required to code each digit (binary code of 9 is 1001). BCD is a convenient mechanism for representing decimal numbers in a binary format, but it is inefficient since only 10 of the 16 possible states of the 4-bit number are used. It does not use the numbers 1010, 1011, 1100, 1101, 1110, and 1111. To illustrate the BCD code, we take a decimal number 123. Each digit is changed to its binary equivalent as follows:

1 2 3 (decimal) = 0111 1011 (binary)

↓ ↓ ↓

0001 0010 0011 (BCD)

To get information into and out of a computer, it must handle non-numerical data in addition to numerical data. Some kind of alphanumeric information (alphabet, punctuation mark, special characters, etc.) must be recognized by a computer. These codes are known as *alphanumeric code*, which are different at one time for different manufacturers of computers. To avoid confusion, industry settled on an input-output code known as the *American Standard Code for Information Interchange* (ASCII), which allows the manufacturer to standardize computer hardware such as keyboards, printers, and video displays. The ASCII code is 7-bit code whose format is $X_6X_5X_4X_3X_2X_1X_0$, where X_i is 0 or 1. For instance, the letter *A* is coded as 100 0001. ASCII code has $2^7 = 128$ possible code groups. This is more than enough to represent the entire standard keyboard characters as well as control function. One should consult the listing of ASCII code to know all possible 128-code groups.

8.3 BASIC LOGIC GATES AND TRUTH TABLES

The simplest digital signals are binary signals having two levels. The two levels may be defined arbitrarily to any set of two values of voltages, but the signal is confined to these two values only. For example, it may be just a presence or an absence of voltage that is used as the two-level signals. On and off state of electronics devices like diode, BJT, and MOS are a very convenient way of setting up the binary signals. A combination of 'true' or 'false' statements had been studied a long time ago by mathematicians for variables having only two possible values. Once the logical ideas are expressed using two-valued functions, the binary number system is highly suitable for such a situation. Although a large number of symbols have been used to represent the status of the binary variables like true and false, on and off, positive and negative, high and low, etc., the symbols '1' and '0' have become quite popular. Thus, if a logical statement *A* is true we say that *A* has a value equal to 1. Similarly if the statement is false, the variable *A* is equal to 0. Only logical significance to 1 and 0 is considered instead of their numerical significance.

Logic gates are digital devices that convert binary inputs into binary outputs based on the rules of mathematical logic operations. These devices are called *gates* because they control the flow of signals from the inputs to the single output. There are three basic logic operations, namely, OR, AND and NOT. Other logic operations such as NOR, NAND and XOR are realized from the basic logic operations. In these logic operations, each combination of the input variables will give an output depending on the operations. The statement of the input and output is presented in a tabular form and is called the *truth table*. Logic gates are basically digital circuits constructed using diodes, transistors and resistors. A truth table is a means for describing how the output of a logic circuit depends on the logic levels present at the inputs of the circuit. Usually, the combination of inputs is written as the ascending list of binary numbers whose number of bits correspond to

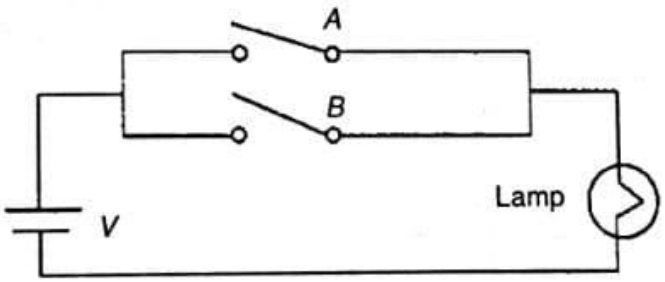
the number of inputs (e.g. 00, 01, 10, 11 for two-input gate). Each of the logic operations is discussed in the following sections.

8.3.1 OR Operation

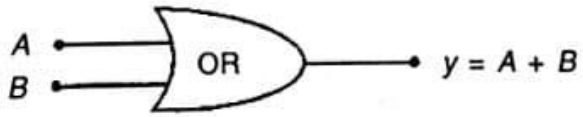
An OR operation with two independent logic variables A, B gives an output of 1 when A or B is 1. We can visualize such a gate as an electrical circuit involving two switches in parallel, as shown in Fig. 8.1(a). There is a current when either switch A or B is closed. OR gate is represented by the equation

$$A + B = y \tag{8.9}$$

In this expression, the + sign does not stand for ordinary addition; it stands for the OR operation. The logic symbol and the truth table of a two-input OR gate are shown as in Figs. 8.1(b) and 8.1(c), respectively. The OR gate operates in such a way that its output is high (logic 1) if either input A or B or both are at a logic 1 level. OR gates can also have more than two inputs. In the case of a multiple input OR gate, the output is 0 if and only if all inputs are 0; otherwise, it is 1.



(a)



(b)

Inputs		Output
A	B	y
0	0	0
0	1	1
1	0	1
1	1	1

(c)

Fig. 8.1 OR gate: (a) electrical circuit analog, (b) logic symbol and (c) truth table.

8.3.2 AND Operation

In the case of AND operation, the output is high (logic 1) only when both the inputs A and B are high, for all other conditions it gives a low output (logic 0). The electrical circuit equivalent of a