第 1 章 数制与码制

内容提要 本章首先介绍数字电路的一些基本概念及数字电路中常用的数制与码制；然后介绍二进制数的算术运算及数字逻辑中的基本逻辑运算；最后介绍硬件描述语言。

随着数字电路与系统的开发和应用，现在众多的电子系统，如电子计算机、通信系统、自动控制系统、影视音响系统等，均使用了数字电路。相对于模拟电路而言，数字电路具有不可比拟的突出优势。数字电路中使用了二进制，通常采用“0”和“1”构成的代码来描述各种有关对象。而硬件描述语言则是对数字电路和系统进行性能描述和模拟的语言。

1.1 数字信号与数字电路概述

数字电路同模拟电路一样，也经历了由分立器件电路到集成电路的发展，且集成度已达到超大规模集成电路的水平。数字电路与系统的体积小，工作可靠性高，应用极其广泛。

1.1.1 数字信号

数字电路中处理的信号是数字信号。在现代技术的信号处理中，数字信号发挥的作用越来越大，几乎复杂的信号处理都离不开数字信号。

信号数据可用于表示任何信息，如符号、文字、语音、图像等。从表现形式上可归结为两类：模拟信号和数字信号。模拟信号与数字信号的区别可根据幅度取值是否离散来确定。

模拟信号指幅度的取值是连续的（幅值可由无限个数值表示）。模拟信号常常是物理现象中被测量对变化的响应。例如，声音、光、温度、位移、压强，这些物理量可以使用传感器测量。时间上离散的模拟信号是一种抽样信号，它是对模拟信号每隔时间 T 抽样一次所得到的信号。虽然其波形在时间上是不连续的，但其幅度取值是连续的，所以仍是模拟信号。当然，在电学上所提的模拟信号往往是指幅度和相位都连续的电信号，此信号可以被模拟电路进行各种运算，如放大、相加和相乘等。图 1.1.1 就是模拟信号的例子，正弦波信号是典型的模拟信号。

数字信号指人们抽象出来的时间上不连续的信号，其幅度的取值是离散的，且幅值被限制在有限个数值之内。十字路口的交通信号灯、数字式电子仪表、自动生产线上产品数量的统计等都是数字信号。图 1.1.2 就是数字信号的例子，矩形波信号是典型的数字信号。由图

1.1.2 可以看出，数字信号的特点是突变和不连续。数字电路中的波形都是这类不连续的波形，通常这类波形又称为脉冲。

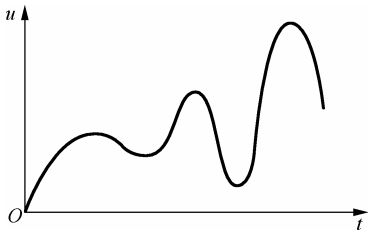

图 1.1.1　模拟信号

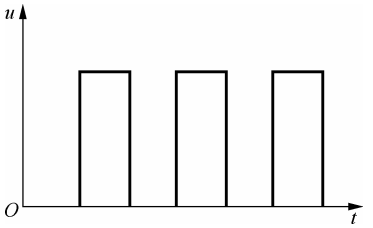

图 1.1.2　数字信号

1.1.2　数字电路与系统

传递与处理数字信号的电子电路称为数字电路或数字系统。由于它具有逻辑运算和逻辑处理功能，所以又称数字逻辑电路。数字电路与模拟电路相比主要有下列优点。

① 数字电路是以二值数字逻辑为基础的，只有“0”和“1”两个基本数字，易于用电路来实现。例如，可用二极管、三极管的导通与截止这两个对立的状态来表示数字信号的逻辑“0”和逻辑“1”。

② 由数字电路组成的数字系统工作可靠，精度较高，抗干扰能力强。它可以通过整形很方便地去除叠加于传输信号上的噪声与干扰，还可利用差错控制技术对传输信号进行查错和纠错。

③ 数字电路不仅能完成数值运算，而且能进行逻辑判断和运算，这在控制系统中是不可缺少的。

④ 数字信息便于长期保存。例如，可将数字信息存入磁盘、光盘等长期保存。

⑤ 数字集成电路产品系列多、通用性强、成本低。

由于具有一系列优点，数字电路在电子设备或电子系统中得到了越来越广泛的应用，计算机、计算器、电视机、音响系统、视频记录设备、光碟、长途电信及卫星系统等，无一不采用了数字系统。

按照电路有无集成元器件来分，数字电路可分为分立元件数字电路和集成数字电路。现代的数字电路大多是由半导体工艺制成的若干数字集成器件构造而成。按集成电路的集成度进行分类，数字电路可分为小规模集成数字电路（SSI，Small Scale Integrated Circuits）、中规模集成数字电路（MSI，Medium Scale Integrated Circuits）、大规模集成数字电路（LSI，Large Scale Integrated Circuits）和超大规模集成数字电路（VLSI，Very Large Scale Integrated Circuits）。

数字电路根据逻辑功能的不同特点，又可以分成两大类，一类叫组合逻辑电路（简称组合电路），另一类叫时序逻辑电路（简称时序电路）。组合逻辑电路在逻辑功能上的特点是任意时刻的输出仅仅取决于该时刻的输入，与电路原来的状态无关。而时序逻辑电路在逻辑功能上的特点是任意时刻的输出不仅取决于当时的输入信号，而且还取决于电路原来的状态，或者说，还与以前的输入有关。

1.2 数制

数制是计数体制（即计数的方法）的简称，有累加计数制和进位计数制两种。累加计数制是原始的计数方法，计多大的数就要使用与所计数目个数相等的各不相同的计数符号，很不方便。比较方便的计数方法是进位计数制，本章所述数制即指进位计数制。常用的进位计数制有十进位计数制（十进制）以及二进制、八进制、十六进制。

1.2.1 数制的基本知识

在介绍各种数制之前，首先介绍数制的基础知识。

1．基本数码

基本数码是指计数制中使用的基本数字符号，简称数码。例如，十进制中的 0、1、2、3、4、5、6、7、8、9 便是数码。

2．基数（*R*）

计数制中所使用的数码的个数称为基数，亦称底数。基数常用 *R* 表示，在十进制中 *R*=10。在基数为 *R* 的数制中，每一个数位上可以使用的数码包括 0 在内共有 *R* 个，最大数码是 *R*–1，而没有 *R*，因此计数时当某位数计到 *R* 时，则在该位记作 0，并向高位进一，即逢 *R* 进一，故基数为 *R* 的计数制称为 *R* 进制计数制。

3．数位（*i*）

在由一串数码构成的数中，数码所在的位置称数位。数位的排序用 *i* 表示，*i* 的计算以小数点为界，向左依次为第 0 位、第 1 位……向右依次为第–1 位、第–2 位……例如，在十进制数 123.45 中，3 是第 0 位数，2 是第 1 位数，4 是第–1 位数等。

4．位权（Weight）

位权亦称权值。在进位计数制的由一串数码构成的数中，各个数位上的数码所表示的数值的大小不但和该数码本身的大小有关，而且还和该数码所处的数位有关。例如，在十进制数 44 中，十位数 4 表示 4×10^{1}，个位数 4 表示 4×10^{0}。可见，不同的数位赋予该位上的数码以不同的表示数的大小的权力。我们把数位上的数码在表示数时所乘的倍数称为该数位的位权。

在 *R* 进制数中，第 *i* 位数位的权值用 W_i 表示，$W_i=R^i$。其中，*R* 是基数，*i* 是数位的位数。例如，十进制数的第 0 位（个位）、第 1 位、第–1 位的位权分别为 10^{0}、10^{1}、10^{-1}。

在由一串数码表示的数中，相邻两个数位中左边数位的位权是右边的 *R* 倍。

5．数的表示方式

在进位计数制中，数的表示方式有位置记数法、按权展开式、和式 3 种。

以十进制数 123.45 为例，分别可以表示为

$$(N)_{10}=123.45\cdots\cdots\text{位置计数法}$$

$$=1\times10^{2}+2\times10^{1}+3\times10^{0}+4\times10^{-1}+5\times10^{-2}\cdots\cdots\text{按权展开式}$$

$$=\sum_{i=-2}^{2}D_i\times10^{i}\cdots\cdots\text{和式}$$

其中，$(N)_{10}$ 的脚标 10 表示 N 为十进制数。二进制、八进制、十进制、十六进制数分别用脚标 2、8、10、16 或 B、O、D、H 表示。位置记数法是用一串数码来表示数，也称并列记数法；按权展开式是用多项式来表示一个数，又称多项式表示法，多项式中的各个乘积项由各个数位上的数码加权（即乘上权值）构成；和式则是把按权展开式表示为 ∑的形式，D_i 表示第 i 位数位上的十进制数码，10^i 为第 i 位的权值。

对于任意一个 R 进制数$(N)_R$，可以用三种方式表示为

$$(N)_R=a_{m-1}a_{m-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-n}\cdots\cdots\text{位置计数法}$$

$$=a_{m-1}\times R^{m-1}+a_{m-2}\times R^{m-2}+\cdots+a_1\times R^{1}+a_0\times R^{0}+$$

$$a_{-1}\times R^{-1}+a_{-2}\times R^{-2}\cdots a_{-n}\times R^{-n}\cdots\cdots\text{按权展开式}$$

$$=\sum_{i=-n}^{m-1}a_i\times R^{i}\cdots\cdots\text{和式}$$

式中，$a_{m-1},a_{m-2},\cdots,a_{-n}$ 分别为第 $m-1$，$m-2$，…，$-n$ 位上的数码；R 为基数；m 和 n 分别为整数部分和小数部分的位数；i 为数位的序号。

1.2.2 常用数制

人们通常采用的数制有十进制、二进制、八进制和十六进制。下面将分别介绍二进制、八进制和十六进制。

1．二进制（Binary）

数码：0、1；

基数：$R=2$；

第 i 位的权值：$W_i=2^i$。

表示方式示例：

$$(101.01)_2=1\times2^{2}+0\times2^{1}+1\times2^{0}+0\times2^{-1}+1\times2^{-2}$$

$$=\sum_{i=-2}^{2}B_i\times2^{i}$$

2．八进制（Octal）

数码：0、1、2、3、4、5、6、7；

基数：$R=8$；

第 i 位的权值：$W_i=8^i$。

表示方式示例：

$$(25.6)_8 = 2\times8^1 + 5\times8^0 + 6\times8^{-1}$$
$$= \sum_{i=-1}^{1} O_i \times 8^i$$

3．十六进制（Hexadecimal）

数码：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F；

基数：$R = 16$；

第 i 位的权值：$W_i = 16^i$。

其中，数码 A、B、C、D、E、F 依次与十进制数 10、11、12、13、14、15 等值，但 A、B、C、D、E、F 在十六进制中是 1 位数。

表示方式示例：

$$(12D.23)_{16} = 1\times16^2 + 2\times16^1 + 13\times16^0 + 2\times16^{-1} + 3\times16^{-2}$$
$$= \sum_{i=-2}^{2} H_i \times 16^i$$

需要说明的是，在数字技术中用得最多的是二进制数。这是因为二进制中只有两个数码，很容易用高低电平来表示；如果采用十进制，则需要用 10 个不同的状态来表示十进制中 10 个不同的数码，很不容易。此外，二进制数的运算也比较简单，因此机器数都用二进制数表示。但是用二进制表示的数往往很长，不便于书写和记忆，相对而言，八进制、十六进制数的书写和记忆比较方便，而且和二进制数之间的转换也很方便，因此，又常用八进制和十六进制数作为二进制数的缩写形式用于书写、输入和显示。

1.2.3 数制转换

下面将介绍各种数制间的转换方法。

1．二（八、十六）进制数→十进制数

将二进制、八进制、十六进制数转换成十进制数，只要把原数写成按权展开式再相加即可。

例 1.2.1 分别将$(101.01)_2$、$(74.5)_8$、$(3C.A)_{16}$转换成十进制数。

解： $(101.01)_2 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} = (5.25)_{10}$

$(74.5)_8 = 7\times8^1 + 4\times8^0 + 5\times8^{-1} = (60.625)_{10}$

$(3C.A)_{16} = 3\times16^1 + 12\times16^0 + 10\times16^{-1} = (60.625)_{10}$

2．十进制数→二进制数

十进制数转换成二进制数只需将整数部分和小数部分分别转换成二进制数，再将转换结果连接在一起即可。

整数的转换用“除 2 取余法”：将十进制数除以基数 2，得商和余数，取下余数作为目标数制数的最低位数码；将所得的商再除以基数又得商和余数，取下余数作为目标数制数的次低位数码……如此连续进行，直至商为 0，余数小于目标数制的基数 2 为止，末次相除所得的余数为目标数制数的最高位数码。

小数的转换用“乘 2 取整法”：将被转换的十进制小数乘以目标数制的基数 2，取下所得乘积中的整数作为目标数小数的最高位；再将乘积中的小数部分乘以基数 2，取下乘积中的整数作为目标数小数的次高位……如此反复进行，直到乘积的小数部分为 0 或达到所需精度为止。各次相乘所得的整数即可构成目标数的小数，第一次相乘所得的整数为小数的最高位，末次相乘所得的整数为小数的最低位。

例 1.2.2 将$(60.625)_{10}$转换成二进制数。

解：（1）对整数 60 进行“除 2 取余”

$$
\begin{array}{r|l}
2 & 60 \\
2 & 30 \quad \cdots\cdots\cdots\cdots 余0\cdots\cdots \text{ 最低位} \\
2 & 15 \quad\quad 0 \\
2 & 7 \quad\quad 1 \\
2 & 3 \quad\quad 1 \\
2 & 1 \quad\quad 1 \\
 & 0 \quad \cdots\cdots\cdots 余1\cdots\cdots \text{ 最高位}
\end{array}
$$

所以，$(60)_{10}=(111100)_2$。

（2）对小数 0.625 进行“乘 2 取整”

$$
\begin{array}{rl}
 & 0.625 \\
 & \times \quad 2 \\
\hline
\text{最高位} & (1).250 \\
 & \times \quad 2 \\
\hline
 & (0).500 \\
 & \times \quad 2 \\
\hline
\text{最低位} & (1).000
\end{array}
$$

所以，$(0.625)_{10}=(0.101)_2$。

（3）整数和小数部分的转换结果连接在一起

$$(60.625)_{10}=(111100.101)_2$$

采用乘基数取整法将十进制小数转换成二进制、八进制、十六进制小数时，可能出现多次相乘后乘积的小数部分仍不为 0 的情况。这时应按照所需的精度来确定位数。

一个 R 进制 n 位小数的精度为 R^{-n}，如$(0.95)_{10}$的精度为 10^{-2}。

例 1.2.3 $(0.39)_{10}=(?)_2$，要求精度达到 1%。

解：设转换后的二进制数小数点后面有 n 位小数，则其精度为 2^{-n}，由题意可知

$$2^{-n} \leqslant 1\%$$

解得 $n \geqslant 7$，取 $n=7$。

```
          0.39
        ×    2
最高位  (0).78
        ×    2
        (1).56
        ×    2
        (1).12
        ×    2
        (0).24
        ×    2
        (0).48
        ×    2
        (0).96
        ×    2
最低位  (1).92
```

所以，$(0.39)_{10}=(0.0110001)_2$。

3．二进制数和十六进制数之间的相互转换

十六进制数的进位基数是 $16=2^4$，因此二进制数和十六进制数之间的转换非常简单。将二进制数转换成十六进制数时，整数部分从低位起每 4 位分成一组，最高位一组不足 4 位时以零补足，小数部分从高位起每 4 位分成一组，最低位一组不足 4 位时也以零补足，然后，依次以 1 位 16 进制数替换 4 位二进制数即可。

例 1.2.4 将二进制数 101101.01001 转换成十六进制数。

解：$(101101.01001)_2=(\underline{0010}\ \underline{1101}.\underline{0100}\ \underline{1000})_2=(2D.48)_{16}$

将十六进制数转换成二进制数时，其过程正好相反，即用 4 位二进制数替换 1 位 16 进制数。

例 1.2.5 将十六进制数$(23A.D)_{16}$转换成二进制数。

解：$(23A.D)_{16}=(\underline{0010}\ \underline{0011}\ \underline{1010}.\underline{1101})_2=(1000111010.1101)_2$

4．二进制数和八进制数之间的相互转换

八进制数的进位基数是 $8=2^3$，将二进制数转换成八进制数时，整数部分从低位起每 3 位分成一组，最高位一组不足 3 位时以零补足，小数部分从高位起每 3 位分成一组，最低位一组不足 3 位时也以零补足，然后，依次以 1 位 8 进制数替换 3 位二进制数即可。

例 1.2.6 将二进制数 10101.00111 转换成八进制数。

解：$(10101.00111)_2=(\underline{010}\ \underline{101}.\underline{001}\ \underline{110})_2=(25.16)_8$

将八进制转数换成二进制数时，用 3 位二进制数依次替换 1 位 8 进制数即可。

例 1.2.7 将八进制数$(27.3)_8$转换成二进制数。

解：$(27.3)_8=(\underline{010}\ \underline{111}.\underline{011})_2=(10111.011)_2$

5．十进制数转换成十六进制数、八进制数

十进制数转换成十六进制数和八进制数时，通常采用的方法是首先把十进制数转换成二进制数，然后再把得到的二进制数转换成十六进制数或八进制数。所用到的转换方法上面均已介绍过。

例 1.2.8 将十进制数 215.625 转换成十六进制数和八进制数。

解： $(215.625)_{10}=(11010111.101)_2=(D7.A)_{16}=(327.5)_8$

1.3 码制

用文字、符号或数码来表示各个特定对象的过程称为编码，编码所得的每组符号称为代码或码字，代码中的每个符号称为基本代码或码元。在数字电路中通常用二进制数码构成的代码来表示各有关对象（如十进制数、字符等）。码制是指编码的制式，不同的码制编码时遵循不同的规则。

1.3.1 二进制码

所谓二进制码是指用二进制数码“0”和“1”构成的代码。n 位的二进制码可以有 2^n 个代码。

表 1.3.1 给出了几种典型的二进制码。

表 1.3.1　　典型二进制码

十进制数	4 位自然二进制码	典型格雷码（循环码）	8421 奇（偶）校验码	
			信息码　P（奇）	信息码　P（偶）
0	0 0 0 0	0 0 0 0	0 0 0 0 1	0 0 0 0 0
1	0 0 0 1	0 0 0 1	0 0 0 1 0	0 0 0 1 1
2	0 0 1 0	0 0 1 1	0 0 1 0 0	0 0 1 0 1
3	0 0 1 1	0 0 1 0	0 0 1 1 1	0 0 1 1 0
4	0 1 0 0	0 1 1 0	0 1 0 0 0	0 1 0 0 1
5	0 1 0 1	0 1 1 1	0 1 0 1 1	0 1 0 1 0
6	0 1 1 0	0 1 0 1	0 1 1 0 1	0 1 1 0 0
7	0 1 1 1	0 1 0 0	0 1 1 1 0	0 1 1 1 1
8	1 0 0 0	1 1 0 0	1 0 0 0 0	1 0 0 0 1
9	1 0 0 1	1 1 0 1	1 0 0 1 1	1 0 0 1 0
10	1 0 1 0	1 1 1 1	1 0 1 0 1	1 0 1 0 0
11	1 0 1 1	1 1 1 0	1 0 1 1 0	1 0 1 1 1
12	1 1 0 0	1 0 1 0	1 1 0 0 1	1 1 0 0 0
13	1 1 0 1	1 0 1 1	1 1 0 1 0	1 1 0 1 1
14	1 1 1 0	1 0 0 1	1 1 1 0 0	1 1 1 0 1
15	1 1 1 1	1 0 0 0	1 1 1 1 1	1 1 1 1 0

1．自然二进制码

自然二进制码是通常用以表示数值的一种二进制码。从编码的角度看，二进制数也是一种表示数的代码，称为自然二进制码。例如，1100 既可以说它是数 12 的二进制数，又可以说它是数 12 的自然二进制码。不过，虽然一个数的自然二进制码和其二进制数在写法上完全一样，但在概念上是不一样的，前者是码制中的概念，后者是数制中的概念。表 1.3.1 中给出

了 4 位自然二进制码，代码中每个码元的位权自左至右分别为 8、4、2、1，16 个代码依次分别用来表示数 0～15。

2．格雷码、循环码

在一组数的编码中，若任意两个相邻数的代码中只有一位对应的码元不同，则称这种编码为格雷码（Gray Code）。格雷码的种类很多，循环码是其中的典型。在循环码中，不仅相邻的两个代码只有 1 位码元不同，而且首尾两个代码也是如此，表 1.3.1 中给出了 4 位循环码。使用循环码可以减少电路工作时出错的可能。通常把两个代码中取值不同的码元的位数称为两个代码的间距，把两个相邻代码中只有一位对应码元取值不同的特点称为单位间距特性。格雷码和循环码都具有单位间距特性，因此，它们都是单位间距码。

循环码的各个代码除最左位以外，其他各位均以最左位 0 和 1 的水平分界线为轴镜像对称；循环码的最左位，分界线以上均为 0，以下均为 1。上述特点称为循环码的反射特性。利用反射特性可以由 n 位循环码方便地写出 $n+1$ 位循环码。例如，1 位、2 位、3 位、4 位循环码如图 1.3.1 所示。

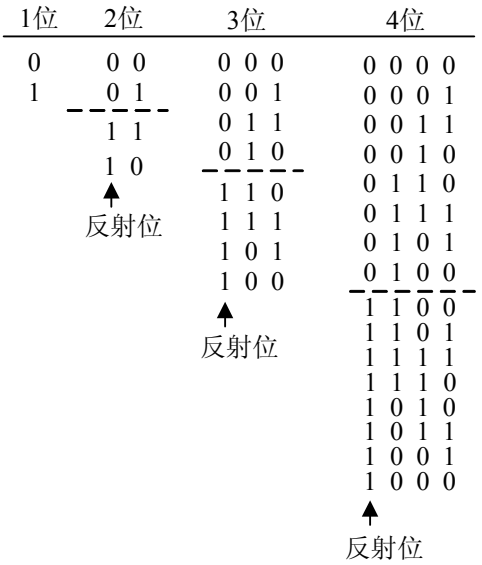

图 1.3.1　1 位、2 位、3 位、4 位循环码

3．奇（偶）校验码

奇（偶）校验码示于表 1.3.1 中。这种码由信息码加一个奇校验位 P（奇）或偶校验位 P（偶）构成，其中校验位的取值（0 或 1）将使各个代码（包括信息码和校验位）中“1”的个数为奇数或偶数。若使“1”的个数为奇数，称为奇校验码；为偶数，则称为偶校验码。奇（偶）校验码的用处是通过检测经传输以后的代码中“1”的个数是否为奇数（或是否为偶数），即进行奇校验（或偶校验）来判断信息在传输过程中是否有一位码元出错。

格雷码、循环码、奇（偶）校验码都是一种可靠性编码。奇、偶校验码具有一定的检测错码的能力，是一种误差检验码，但只能检错而不能纠错。使用海明码则可以达到检测并纠

正错码的要求。

1.3.2 二-十进制（BCD）码

所谓二-十进制码又称 BCD（Binary Coded Decimal）码，是指用二进制数码 0 和 1 来表示的十进制数码 0，1，2，…，9，或者说是十进制数码 0，1，2，…，9 的二进制编码。要表示 0～9 这 10 个十进制数码，使用的代码至少需要有 4 位码元。由于 4 位二进制数可以构成 16 个不同的代码，因此，构成 BCD 码的方案可以有多种，不过最常用的也只是其中的几种。BCD 码可以分为有权码和无权码两大类，表 1.3.2 列出了几种最常用的 BCD 码，其编码规则各不相同。

表 1.3.2　　几种常用的 BCD 码

十进制数	8421 码	5421 码	2421 码	余 3 码	余 3 循环码	格雷码
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 0 0 1	0 1 0 0	0 1 1 0	0 0 0 1
2	0 0 1 0	0 0 1 0	0 0 1 0	0 1 0 1	0 1 1 1	0 0 1 1
3	0 0 1 1	0 0 1 1	0 0 1 1	0 1 1 0	0 1 0 1	0 0 1 0
4	0 1 0 0	0 1 0 0	0 1 0 0	0 1 1 1	0 1 0 0	0 1 1 0
5	0 1 0 1	1 0 0 0	1 0 1 1	1 0 0 0	1 1 0 0	0 1 1 1
6	0 1 1 0	1 0 0 1	1 1 0 0	1 0 0 1	1 1 0 1	0 1 0 1
7	0 1 1 1	1 0 1 0	1 1 0 1	1 0 1 0	1 1 1 1	0 1 0 0
8	1 0 0 0	1 0 1 1	1 1 1 0	1 0 1 1	1 1 1 0	1 1 0 0
9	1 0 0 1	1 1 0 0	1 1 1 1	1 1 0 0	1 0 1 0	1 0 0 0

1．有权码

代码中的每一位都有固定权值的代码称为有权码或恒权码。有权码的名称通常用 4 个码位的位权来命名。表 1.3.2 中的 8421BCD、5421BCD、2421BCD 都是有权码。

各种有权 BCD 码所表示的十进制数 D 可以由按权展开式求得。例如，8421BCD 码 $b_3b_2b_1b_0$ 所表示的十进制数码为 $D = 8b_3+4b_2+2b_1+1b_0$。

2．无权码

代码中的各位没有固定权值的代码称为无权码。表 1.3.2 中的余 3 码、余 3 循环码和格雷码都是无权码。

余 3 BCD 码由 8421BCD 码加 3 得到，其主要特点是 0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 的代码互为反码，具有这种特点的代码称为自补码（Self Complementing Code）。

表 1.3.2 中的余 3 BCD 循环码和格雷 BCD 码分别由表 1.3.1 的 4 位二进制代码中的典型格雷码（循环码）去掉 6 个代码构成，仍具有原代码所具有的单位间距特性或检错特性。其中，余 3 BCD 循环码是从循环码的第 4 个码字开始依次取 10 个而形成。

每种 4 位二进制代码共有 16 个代码，在表 1.3.2 所列的各种常用 BCD 码中，未列出的代码是该码制中不允许出现的代码，称为禁用码或非法码，因为它们不与单个十进制数码相对应，是不允许出现的。如 8421BCD 码中的 1010～1111 六个代码即是 8421BCD 码中的禁用

码（非法码）。

一个多位的十进制数可以用多个 BCD 代码表示，表示时代码之间应有间隔。例如，用 8421BCD 码表示$(123)_{10}$时，其书写形式为

$$(123)_{10}=(0001\ 0010\ 0011)_{8421BCD}$$

应该注意，用 BCD 码表示的十进制数不是二进制数，也不能直接转化为二进制数。要转换，应先将其转换成十进制数，再由十进制数转换成二进制数。

3．8421BCD 码的加法运算

两个 8421BCD 码相加时，可把其当作自然二进制码相加，如在相加所得结果中未出现 8421BCD 码的非法码，则该结果就是所需的 8421BCD 码。但如在相加结果中出现了 8421BCD 码的非法码或在相加过程中在 BCD 数位上出现了向高位的进位，则应对非法码及产生进位的代码进行“加 6(0110)修正”。在作多位 8421BCD 码相加时，如果在“加 6(0110)修正”过程中又出现非法码，则还需继续作“加 6 修正”。

例 1.3.1 试用 8421BCD 码分别求 1+9 及 8+8。

解：①

```
    0 0 0 1
+   1 0 0 1
-------------
    1 0 1 0   ←— 非法码
+   0 1 1 0   ←— 加6修正
-------------
0 0 0 1 0 0 0 0
```

所以，1+9=$(0001\ 0000)_{8421BCD}=(10)_{10}$。

②

```
    1 0 0 0
+   1 0 0 0
-------------
  1 0 0 0 0   ←— 个位产生进位
+   0 1 1 0   ←— 加6修正
-------------
0 0 0 1 0 1 1 0
```

所以，8+8=$(0001\ 0110)_{8421BCD}=(16)_{10}$。

例 1.3.2 试用 8421BCD 码求 712+989。

```
      0 1 1 1   0 0 0 1   0 0 1 0
+     1 0 0 1   1 0 0 0   1 0 0 1
-------------------------------------
0 0 0 1   0 0 0 0   1 0 0 1   1 0 1 1   ←— 个位是非法码，百位产生进位
+     0 1 1 0               0 1 1 0     ←— 在个位和百位上加6修正
-------------------------------------
0 0 0 1   0 1 1 0   1 0 1 0   0 0 0 1   ←— 十位是非法码
+             0 1 1 0                   ←— 在十位上加6修正
-------------------------------------
0 0 0 1   0 1 1 1   0 0 0 0   0 0 0 1
```

所以，712+989=$(0001\ 0111\ 0000\ 0001)_{8421BCD}=(1701)_{10}$。

4．8421BCD 码的减法运算

8421BCD 码相减时，由于在进行减法运算时使用了四位二进制减法运算“借一当十六”的规则，而 8421BCD 码相减时应为“借一当十”，故当发生借位时应进行“减 6 修正”。

例 1.3.3 试用 8421BCD 码求 113−55。

解：

```
    0001  0001  0011
−   0000  0101  0101
-------------------
    0000  1011  1110  ←— 个位、十位有借位
−         0110  0110  ←— 在个位和十位上减6修正
-------------------
    0000  0101  1000
```

所以，$113-55=(0000\ 0101\ 1000)_{8421BCD}=(58)_{10}$。

1.4 算术运算与逻辑运算

在数字电路中，1 位二进制数码的 0 和 1 不仅可以表示数量的大小，而且可以表示两种不同的逻辑状态。当两个二进制数码表示两个数量大小时，它们之间可以进行数值运算，这种运算称为算术运算；当两个二进制数码表示两种不同逻辑状态时，它们之间可以进行逻辑运算。

1.4.1 算术运算

二进制数算术运算的法则和十进制数的运算法则基本相同。唯一的区别在于十进制数是逢十进一、借一当十，而二进制数是逢二进一、借一当二。

以两个二进制数 1001 和 0101 为例，其算术运算如下。

加法运算

```
  1001
+ 0101
------
  1110
```

减法运算

```
  1001
− 0101
------
  0100
```

乘法运算

```
   1001
×  0101
-------
   1001
  0000
 1001
+0000
-------
 0101101
```

除法运算

```
          1.11……
       ------------
0101 ) 1001
       −0101
       ------
         1000
        −0101
        ------
          0110
         −0101
         ------
           001
```

需要说明的是，在数字电路中为了简化运算电路，通常二数相减的运算是用其补码的加法来实现的，乘法运算则用移位和加法两种操作来完成，而除法运算是用移位和减法操作来完成，因此，二进制数的加、减、乘、除都可以用加法电路完成。所以在数字设备中加法器是极为重要的运算部件。

1.4.2 逻辑运算

在数字电路中，1 位二进制数码的 0 和 1 不仅可以表示数量的大小，而且可以表示两种不同的逻辑状态。例如，可以用 1 和 0 分别表示一件事情的是和非、真和伪、有和无、好和坏，或者表示电路的通和断、电灯的亮和暗等。这种只有两种对立逻辑状态的逻辑关系称为

二值逻辑。

逻辑代数中只有三种基本运算：与、或、非。只有当决定一件事情的条件全部具备之后，这件事情才会发生，这种因果关系称为与逻辑；当决定一件事情的几个条件中，只要有一个或一个以上条件具备，事情就会发生，这种因果关系称为或逻辑；某事情发生与否，仅取决于一个条件，且条件具备时事情不发生，条件不具备时事情才发生，这种因果关系称为非逻辑。

除了以上三种基本逻辑运算外，还有一些逻辑运算是综合运用了上述三种基本运算，形成了各种复合逻辑运算。

关于逻辑运算的详细讨论将在第 2 章中进行。

1.5 HDL

HDL（Hardware Description Language）是硬件描述语言。硬件描述语言是一种对数字电路和系统进行性能描述和模拟的语言，即利用高级语言来描述硬件电路的功能、信号连接关系以及各器件间的时序关系。数字电路和数字系统设计者利用这种语言来描述自己的设计思想，然后利用电子设计自动化工具进行仿真、综合，最后利用专用集成电路或可编程逻辑器件来实现其设计功能。其设计理念是将硬件设计软件化，即采用软件的方式来描述硬件电路。

硬件描述语言（HDL）有很多种，目前在我国广泛应用的硬件描述语言主要有 ABEL 语言、AHDL 语言、VerilogHDL 语言和 VHDL 语言等。其中，VerilogHDL 语言和 VHDL 语言最为流行，它们作为 IEEE 标准化的硬件描述语言，具有许多优点：能够抽象地描述电路结构和行为，支持逻辑设计层次与领域的描述，硬件描述与实现工艺无关，易于理解和可移植性好等。但是 VHDL 相对于 Verilog HDL 而言，语法上更严谨些。虽然这样也使它失去了一些灵活性和多样性，但是从文档记录、综合性以及器件和系统级的仿真上讲，VHDL 是一种更好的选择。

VHDL 的英文全名是 Very-High-Speed Integrated Circuit Hardware Description Language，诞生于 1982 年。1987 年底，VHDL 被 IEEE 和美国国防部确认为标准硬件描述语言。自 IEEE 公布了 VHDL 的标准版本 IEEE-1076（简称 87 版）之后，各 EDA 公司相继推出了自己的 VHDL 设计环境，或宣布自己的设计工具可以和 VHDL 接口。此后 VHDL 在电子设计领域得到了广泛的接受，并逐步取代了原有的非标准的硬件描述语言。1993 年，IEEE 对 VHDL 进行了修订，从更高的抽象层次和系统描述能力上扩展 VHDL 的内容，公布了新版本的 VHDL，即 IEEE 标准的 1076-1993 版本（简称 93 版）。

习题

1.1 数字信号和模拟信号有何区别？

1.2 数字电路有哪些特点？

1.3 将下列各数写成按权展开式。

$(352.6)_{10}$，$(101.101)_{2}$，$(54.6)_{8}$，$(13A.4F)_{16}$

1.4 按十进制数 0～17 的顺序，列表写出相应的二进制、八进制、十六进制数。

1.5 二进制数 00000000～11111111 和 0000000000～1111111111 分别可以代表多少个数？

1.6 将下列各数分别转换成十进制数。

$(1111101000)_2$，$(1750)_8$，$(3E8)_{16}$

1.7 将下列各数分别转换成二进制数。

$(210)_8$，$(136)_{10}$，$(88)_{16}$

1.8 将下列各数分别转换成八进制数。

$(111111)_2$，$(63)_{10}$，$(3F)_{16}$

1.9 将下列各数分别转换成十六进制数。

$(11111111)_2$，$(377)_8$，$(255)_{10}$

1.10 转换下列各数，要求转换后保持原精度。

$(1.125)_{10}=(\quad\quad)_2$

$(0010\ 1011\ 0010)_{2421BCD}=(\quad\quad)_2$

$(0110.1010)_{余3循环BCD}=(\quad\quad)_2$

1.11 用下列代码表示$(123)_{10}$，$(1011.01)_2$：

（1）8421BCD 码；　（2）余 3 BCD 码数。

1.12 将下列十进制数分别转换成二进制、八进制、十六进制数。

$(127)_{10}$，$(130.525)_{10}$，$(218.5)_{10}$

1.13 已知 $A=(1011010)_2$，$B=(101111)_2$，$C=(1010100)_2$，$D=(110)_2$。

（1）按二进制运算规律求 $A+B$，$A-B$，$C\times D$，$C\div D$；

（2）将 A、B、C、D 转换成十进制数后，求 $A+B$，$A-B$，$C\times D$，$C\div D$，并将结果与（1）进行比较。

1.14 试用 8421BCD 码完成下列十进制数的运算。

（1）5+8；　（2）9+8；　（3）58+2；　（4）9–3；　（5）87–25；　（6）843–348。

1.15 试导出 1 位余 3 BCD 码加法运算的规则。