面向 21 世纪信息科学最新教材

现 代 数 字 逻 辑

马义忠　常蓬彬　关少颖　　编著

兰 州 大 学 出 版 社

内 容 简 介

本书是原面向21世纪 数字逻辑》课程的革新教材，着重介绍数字系统逻辑设计的基本理论和方法，突出基本分析方法和设计方法与硬件软件高度结合的设计。其内容吸收了当前数字逻辑设计技术的新发展、新思路，引入了电子设计自动化（EDA）的基本思想。从传统的单纯硬件设计转向硬件软件高度渗透的设计方法，从而拓宽软硬件设计的知识面，提高设计能力。书中软硬件结合恰当，概念清晰，取材丰富，有一定的前沿性和新颖性。全书文字流畅，图文并茂，可读性强。

本书系统地阐述了数制与码制，逻辑代数基础，组合逻辑电路和时序逻辑电路的分析与设计，基本的逻辑器件，数字系统设计方法基础，数字系统的基本算法与逻辑实现，以及VHDL语言描述数字系统。

本书可作为计算机科学类、电子信息类、通信类各专业本科、专科学生的教材，同时，也可作为相关专业工程技术人员的参考书。

前　言

现代数字逻辑是计算机科学、数字通信以及电子信息类专业大学本科及专科学生重要的专业基础课。

本书是对原面向21世纪《数字逻辑》课程的革新。首先对数字逻辑的核心内容作了全面系统的阐述，在阐释数字逻辑电路基本概念和原理的基础上，重点讲述数字系统逻辑设计的基本理论及方法。其次集中讨论逻辑设计的重点问题，突出分析方法和设计方法。内容吸收了当前数字逻辑设计技术的新发展、新思路，引入了电子设计自动化(EDA)的基本思想。以传统的单纯硬件设计转向硬件软件高度渗透的设计方法，从而拓宽了学生软硬件设计的知识面并提高了学生的设计能力。

全书共十一章，第一章和第二章介绍数制与码制、逻辑代数基础及逻辑函数化简的方法；第三章至第六章分别讨论组合逻辑电路和时序逻辑电路的分析与设计；第七章结合中、大规模集成电路的应用，论述组合逻辑、时序逻辑和数字系统设计的方法；第八章介绍几种常用集成逻辑部件的原理与特性；第九章和第十章论述采用算法描述数字系统的设计方法和逻辑实现；第十一章讨论采用VHDL语言设计数字系统的方法。本书编写过程中力求遵循既强调基础理论，又注重实际应用；既体现系统性，又突出重点的原则。

在本教材的编写过程中，得到了兰州大学教务处、兰州大学出版社等单位领导的大力支持，并得到了兰州大学信息工程学院王玉曾教授、田亚菲教授的指导，在此一并表示感谢！

由于编者水平有限，书中疏漏和不当之处在所难免，殷切希望广大读者批评指正。

编　者

2002年3月1日

于兰州大学

目 录

第一章　数制与码制

内容提要

本章主要介绍进位计数制的表示方法，各种进制数的相互转换，带符号数的代码表示法，数的定点表示和浮点表示，数码和字符的代码表示等。

§1.1　概述

1.1.1　数字量和模拟量的区别

自然界中形形色色的物理量，尽管它们的性质各异，但就其变化规律的特点而言，可分为两大类。

一类物理量在时间上和数值上的变化都是离散的，它们的变化在时间上是不连续的，总是发生在一系列离散的瞬间。同时，它们的数值大小和每次的增减变化都是某一最小数量单位的整数倍，而小于这个最小数量单位的数值没有任何物理意义。这一类物理量我们把它定义为数字量，把表示数字量的信号叫做数字信号，并且把工作在数字信号下的电子电路称为数字电路。

在现实生活中，用电子电路记录从自动生产线上输出的零件数目时，每输出一个零件便给电子电路一个信号，记为 1，而没有零件输出时便给电子电路的信号是 0，此时不记数。由此可见，零件数目这个信号无论在时间上还是在数量上都是不连续的，因此它是一个数字信号，最小的数量单位就是 1 个。

而另一类物理量在时间上和数值上的变化则是连续的。这一类物理量我们把它叫做模拟量，把表示模拟量的信号叫做模拟信号，并把工作在模拟信号下的电子电路称为模拟电路。

在实际工作中，热电偶在工作时输出的电压信号就属于模拟信号，因为在任何情况下被测温度都不可能发生突跳，所以测得的电压信号无论在时间上还是在数值上都是连续的。而且，这个电压信号在连续变化过程中的任何一个取值都有具体的物理意义，即表示一个相应的温度。

1.1.2 数字信号系统的特点

数字信号系统可处理的信号都是参数元素，这些元素可以是各种数字、字母、算符及符号等。离散元素按不同方式排例可以表示各种信号。如字母 C，O，M，P，U，T，E 和 R 可组合成 COMPUTER 单词，而数字 2001 表示一个确定的数，等等。

1.1.3 在数字系统中信号的表示

信息的离散元素是以称为信号的物理量来表示的，电压和电流就是最长用的电信号。通常在数字系统中的信号为“有”或“无”也有称“真”或“假”）两个离散量。这两个离散量称为二进制信号。由于只有导通和截止两种工作状态的电子器件（二极管、三极管、CDMS 管）能可靠地反映这两个离散量，并且在工程上很容易实现。加上人类的逻辑思维方式也倾向于二值性。所以数字系统常要用于进制信号。

1.1.4 离散信号的形成

信号的离散量有的是自然形成的，有的则是对连续过程有意加以量化后而得到的。如一份学生成绩单就是由离散量形成的，上面有学生的学号、姓名、课程名称、得分等。而科学工作者在工作时常常要观察连续的过程（如温度的变化过程），把一些特定的数值记录下来并列成表，将连续的信息离散并且量化了。表中每一个数字都已成为离散量。

§1.2 进位计数制

1.2.1 十进制数的表示

用数字量表示物理量的大小时，仅用一位数码往往不够用。因此，用一组统一的符号和规则表示数的方法，常用进位计数的方法组成多位数码使用。我们把一组多位数码中每一位的构成方法以及从低位到高位的进位规则称为数制。

原则上说，一个数可以用任何一种进位计数制来表示和运算，但不同数制其运算方法及难易程度互不相同。选择什么样的进位计数制来表示数，对数字系统的性能影响很大。

在日常生活中，人们通常采用十进制数来计数，每位数可用十个数码之一来表示，即 0，1，2，3，4，5，6，7，8，9。十进制的基数为 10，基数表示进位制所具有的数字符号个数，十进制具有的数字符号个数为 10。

十进制数的计算规律是由低位向高位进位时“逢十进一”，也就是说，每位累计不能超过 9，计满 10 就应向高位进 1。

当人们看到一个十进制数，如 632.45 时，就会立刻想到：这个数的最左位（即第一位）为百位（6 代表 600），第二位为十位（3 代表 30），第三位为个位（2 代表 2），小数点右

面第一位为十分位（代表 4×10^{0}），第二位为百分位（代表 5×10^{-2}）。这里百、十、个、十分之一和百分之一都是 10 的次幂。在一个进位制表示的数中，不同数位上的固定常数称之为“权”。十进制数 632.45 按权展开的形式如下：

$$632.45=6\times10^{2}+3\times10^{1}+2\times10^{0}+4\times10^{-1}+5\times10^{-2}$$

等式左边的表示方法称为位置计数表示法，等式右边则是其按权展开表示法。

一般说来，对于任意一个十进制数 N，可用位置计数表示如下：

$$(N)_{10}=(k_{n-1}k_{n-2}\cdots k_1k_0.k_{-1}k_{-2}\cdots k_{-m})_{10}$$

也可用按权展开表示法表示如下：

$$\begin{aligned}(N)_{10}&=k_{n-1}\times10^{n-1}+k_{n-2}\times10^{n-2}+\cdots k_1\times10^{1}+k_0\times10^{0}\\&\quad+k_{-1}\times10^{-1}+k_{-2}\times10^{-2}+\cdots k_{-m}\times10^{-m}\\&=\sum_{i=-m}^{n-1}k_i\times10^{i}\end{aligned}$$

式中：k_i 表示各个数字符号，为 0 ～ 9 这 10 个数码中的任意一个；n 为整数部分的位数；m 为小数部分的位数。

通常，对十进制数的表示，可以在数字的右下角标注 10 或 D。

1.2.2 二进制数的表示

数字系统中使用的进位制并不仅限于十进制，当进位基数为 2 时，称为二进制。在二进制中，只有 0 和 1 两个数码。二进制的计数规则是由低位向高位“逢二进一”，即每位计满 2 就向高位进 1，例如，$(101)_2$ 就是一个二进制数。不同数位的数码表示的值不同，各位的权值是以 2 为底的连续整数幂，从右向左递增。

对于任意一个二进制数 N，用位置计数法表示为：

$$(N)_2=(k_{n-1}k_{n-2}\cdots k_1k_0.k_{-1}k_{-2}\cdots k_{-m})_2$$

用按权展开法表示为：

$$\begin{aligned}(N)_2&=k_{n-1}\times2^{n-1}+k_{n-2}\times2^{n-2}+\cdots+k_1\times2^{1}+k_0\times2^{0}+k_{-1}\times2^{-1}\\&\quad+k_{-2}\times2^{-2}+\cdots+k_{-m}\times2^{-m}\\&=\sum_{i=-m}^{n-1}k_i\times2^{i}\end{aligned}$$

式中：k_i 表示各个数字符号，为 0 或 1；n 为整数部分的位数；m 为小数部分的位数。

通常，对二进制数的表示，可以在数字右下角标注 2 或 B。

在数字系统中，常用二进制来表示数字和进行运算。因为它具有如下特点：

（1）二进制数只有 0 或 1 两个数码，任何具有两个不同稳定状态的元件都可用来表示 1 位二制数，例如，晶体管的导通和截止，脉冲信号的“有”或“无”等。

(1)二进制运算规则简单。其运算规则如下：

加法规则：

$0+0=0$　　$0+1=1$

$1+0=1$　　$1+1=0$（同时向相邻高位进 1）

减法规则：

$0-0=0$　　$0-1=1$（同时向相邻高位借 1）

$1-0=1$　　$1-1=0$

乘法规则：

$0\times0=0$　　$0\times1=0$

$1\times0=0$　　$1\times1=1$

除法规则：

$0\div1=0$　　$1\div1=1$

下面举例说明：

例 1.2.1　进行 $1101+1011$ 运算。

解：

$$\begin{array}{rr} & 1\ 1\ 0\ 1 \\ +) & 1\ 0\ 1\ 1 \\ \hline & 1\ 1\ 0\ 0\ 0 \end{array}$$

两个二进制数的加法运算和十进制数的加法运算相似，但采用‘逢二进一’的法则，每位数累计到 2 时，本位就记为 0，同时向相邻高位进 1。

例 1.2.2　进行 $11101-10011$ 运算。

解：

$$\begin{array}{rr} & 1\ 1\ 1\ 0\ 1 \\ -) & 1\ 0\ 0\ 1\ 1 \\ \hline & 0\ 1\ 0\ 1\ 0 \end{array}$$

二进制减法运算从低位起按位进行，在遇到 0 减 1 时，就要采用‘借一当二’的法则向相邻高位借 1，也就是从那一高位减去 1。

例 1.2.3　进行 1101×1001 运算。

解：

$$\begin{array}{rr} & 1\ 1\ 0\ 1 \\ \times) & 1\ 0\ 0\ 1 \\ \hline & 1\ 1\ 0\ 1 \\ & 0\ 0\ 0\ 0 \\ & 0\ 0\ 0\ 0 \\ & 1\ 1\ 0\ 1 \\ \hline & 1\ 1\ 1\ 0\ 1\ 0\ 1 \end{array}$$

二进制的乘法运算和十进制数的乘法运算相似，所不同的是对部分积进行累加时要按‘逢二进一’的法则。

例 1.2.4 **进行** 10010001÷1011 **运算。**

解：

```
          1 1 0 1 ……商
      ┌──────────
1 0 1 1 √1 0 0 1 0 0 0 1
        1 0 1 1
        ─────
          1 1 1 0
          1 0 1 1
          ─────
            1 1 0 1
            1 0 1 1
            ─────
              1 0 ……余数
```

二进制数的除法运算同十进制数的除法运算类似，但采用二进制数的运算规则。

(3)二进制数只有两个状态，数字的传输和处理不容易出错，可靠性高。

(4)二进制数的数码 0 **和** 1**，可与逻辑代数中逻辑变量的值‘假’和‘真’对应起来。也就是说，可用一个逻辑变量来表示一个二进制数码。这样，在逻辑运算中就可以使用逻辑代数这一数学工具。**

1.2.3 其它进制数的表示

二进制数运算规则简单，便于电路实现，它是数字系统中广泛应用的一种数制。但用二进制表示一个数时，所用的位数比用十进制数表示的位数多，人们读写很不方便，容易出错，不便记忆。因此，人们常采用八进制数和十六进制数，这两种数制不但容易书写和阅读，便于记忆，而且具有二进制数的特点，十分容易将它们转换成二进制数。

八进制数的基数是 8**，采用的数码是** 0,1,2,3,4,5,6,7**。计数规则是从低位向高位‘逢八进一’，对于相邻两位来说，高位的权值是低位权值的** 8 **倍。例如，数** $(7.6)_8$ **就表示一个八进制数。由于八进制的数码和十进制前** 8 **个数码相同，为了便于区分，通常在八进制数字的右下角标注** 8 **或数字后连接** O**。**

十六进制数的基数为 16**，采用的数码是** 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**。其中** A,B,C,D,E,F **分别代表十进制数字** 10,11,12,13,14,15**。十六进制的计算规则是从低位向高位‘逢十六进一’，对于相邻两位来说，高位的权值是低位权值的** 16 **倍。例如，**$(4AF.8B)_{16}$**就是一个十六进制数。通常，在十六进制数字的右下角标注** 16 **或数字后连接** H**。**

与二进制数一样，八进制数和十六进制数均可用位置计数法的形式和按权展开法的形式表示。

一般说来，对于任意的数 N**，都能表示成以** R **为基数的** R **进制数。数** N **的位置记数法为：**

$$(N)_R = (k_{n-1}k_{n-2}\cdots k_1k_0 \cdot k_{-1}k_{-2}\cdots k_{-m})_R$$

而相应的按权展开表示法为：

$$(N)_R = k_{n-1} \times R^{n-1} + k_{n-2} \times R^{n-2} + \cdots + k_1 \times R^1 + k_0 \times R^0 + k_{-1} \times R^{-1} + k_{-2} \times R^{-2} + \cdots k_{-m} \times R^{-m} = \sum_{i=-m}^{n-1} k_i R^i$$

式中：k_i 表示各个位的数字符号，为 $0 \sim R-1$ 数码中任意一个；R 为进位制的基数；n 为整数部分的位数；m 为小数部分的位数。

R 进制的计数规则是从低位向高位“逢 R 进一”。

常用的不同数制的各种数码见表 1.2.1，该表列出了当 R 为 10、2、8、16 时各种进位计数制中开头的 16 个自然数。

表 1.2.1 不同进位计数制的各种数码

十进制数 ($R=10$)	二进制数 ($R=2$)	八进制数 ($R=8$)	十六进制数 ($R=16$)
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

§1.3 数制转换

1.3.1 二进制数与十进制数的转换

在计算机和其他数字系统中，普遍使用二进制数，采用二进制数的数字系统只能处理二进制数和用二进制代码形式表示的其它进制数。而人们习惯于使用十进制数，所以，在信息处理中首先必须把十进制数转换成计算机能加工和处理的二进制数进行运算，然后再将二进制数的计算结果转换成人们习惯的十进制数。

二进制数转换成十进制数是很方便的，只要将二进制数写成按权展开式，并将式中

各乘积项的积算出来，然后各项相加，即可得到与该二进制数相对应的十进制数。例如

$$\begin{aligned}(1010.101)_2 &= 1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0\\ &\quad+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}\\ &=16+8+2+0.5+0.125\\ &=(10.625)_{10}\end{aligned}$$

十进制数转换成二进制数时，需将待转换的数分成整数部分和小数部分，并分别加以转换。一个十进制数可写成

$$(N)_{10}=(\text{整数部分})_{10}.(\text{小数部分})_{10}$$

转换时，首先将 $(\text{整数部分})_{10}$ 转换成 $(\text{整数部分})_2$，然后再将 $(\text{小数部分})_{10}$ 转换成 $(\text{小数部分})_2$。待整数部分和小数部分确定后，就可写成

$$(N)_2=(\text{整数部分})_2.(\text{小数部分})_2$$

一、整数转换

十进制数的整数部分采用‘除 2 取余’法进行转换，即把十进制数除以 2，取出余数 1 或 0 作为相应二进制数的最低位，把得到的商再除以 2，取余数 1 或 0 作为二进制数的次低位，依次类推，继续上述过程，直到商为 0，所得余数为最高位。

例如，将十进制整数 58 转换为二进制整数，先把它写成如下形式；

$$\begin{aligned}(58)_{10}&=(k_{n-1}k_{n-2}\cdots k_1k_0)_2\\ &=k_{n-1}\times2^{n-1}+k_{n-2}\times2^{n-2}+\cdots+k_1\times2^1+k_0\times2^0\\ &=2(k_{n-1}\times2^{n-2}+k_{n-2}\times2^{n-3}+\cdots+k_1)+k_0\end{aligned}$$

只要求出等式中的各个系数 k_{n-1}，k_{n-2}，…，k_1，k_0 便得到二进制整数。将上式两边除以 2，得

$$(29)_{10}=k_{n-1}\times2^{n-2}+k_{n-2}\times2^{n-3}+\cdots+k_1+\frac{k_0}{2}$$

两数相等，整数部分和小数部分必须对应相等，等式左边余数为 0，则 k_0 为 0。

因而得

$$(29)_{10}=2(k_{n-1}\times2^{n-3}+k_{n-2}\times2^{n-4}+\cdots+k_2)+k_1$$

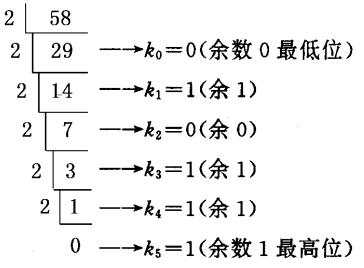
将等式两边再除以 2，得

$$(14+\frac{1}{2})_{10}=k_{n-1}\times2^{n-3}+k_{n-2}\times2^{n-4}+\cdots+k_2+\frac{k_1}{2}$$

比较等式两边，等式左边余数为 1，则取 k_1 为 1。

依次类推，可得系数 k_2，k_3，…，k_{n-2}，k_{n-1}。

根据上面讨论的方法，可用下列形式很方便地将十进制整数转换成二进制整数。



因此，$(58)_{10}=(11010)_2$。

二、纯小数转换

十进制数的小数部分采用‘乘 2 取整’法进行转换，即先将十进制小数乘以 2，取其整数 1 或 0 作为二进制小数的最高位；然后将乘积的小数部分再乘以 2，并再取整数作为次高位。重复上述过程，直到小数部分为 0 或达到所要求的精度。

例如，将十进制小数 0.625 转换为二进制小数，需把它写成如下形式：

$$\begin{aligned}(0.625)_{10}&=(0.k_{-1}k_{-2}\cdots k_{-m})_2\\&=k_{-1}\times 2^{-1}+k_{-2}\times 2^{-2}+\cdots+k_{-m}\times 2^{-m}\\&=\frac{k_{-1}}{2}+\frac{1}{2}(k_{-2}+\cdots+k_{-m}\times 2^{-m+1})\end{aligned}$$

只要求出各系数 k_{-1}，k_{-2}，…，k_{-m}，便得到二进制小数。

将上式两边乘 2，得

$$(1.25)_{10}=k_{-1}+(k_{-2}\times 2^{-1}+\cdots+k_{-m}\times 2^{-m+1})$$

根据两个数相等，其整数部分和小数部分必须分别相等的道理，k_{-1}等于左边的整数，则 k_{-1}为 1。

等式右边括号内的数仍为小数，因而

$$(0.25)_{10}=\frac{k_{-2}}{2}+\frac{1}{2}(k_{-3}\times 2^{-1}+\cdots+k_{-m}\times 2^{-m+2})$$

再将等式两边乘 2，得

$$(0.5)_{10}=k_{-2}+(k_{-3}\times 2^{-1}+\cdots+k_{-m}\times 2^{-m+2})$$

比较等式两边的整数，又取 k_{-2}为 0。如此连续乘 2，直到小数部分等于 0，即可求得系数 k_{-1}，k_{-2}，…，k_{-m}。

根据上面讨论的方法，可用下列形式很方便地将十进制小数转换成二进制小数：

$$
\begin{array}{rl}
0.625 & \\
\times) \quad 2 & \\
\hline
\boxed{1}.250 & \text{整数 } 1\ (k_{-1}) \text{最高小数位} \\
\times) \quad 2 & \\
\hline
\boxed{0}.500 & \text{整数 } 0\ (k_{-2}) \text{次高位小数位} \\
\times) \quad 2 & \\
\hline
\boxed{1}.000 & \text{整数 } 1\ (k_{-3}) \text{最低位小数位}
\end{array}
$$

最后得：$(0.625)_{10} = (0.101)_2$。

必须指出：运算时，式中的整数不参加连乘。

在十进制的小数部分转换中，有时连续乘 2 不一定能使小数部分等于 0，这说明该十进制小数不能用有限位二进制小数表示。这时，只要取足够多的位数，使其误差达到所要求的精度就可以了。下面举例说明。

例 1.3.1　将十进制数 0.18 转换成二进制数，精确到小数点后 4 位。

解：

$$
\begin{array}{rl}
0.18 & \\
\times) \quad 2 & \\
\hline
\boxed{0}.36 & \text{整数 } 0\ (k_{-1}) \\
\times) \quad 2 & \\
\hline
\boxed{0}.72 & \text{整数 } 0\ (k_{-2}) \\
\times) \quad 2 & \\
\hline
\boxed{1}.44 & \text{整数 } 1\ (k_{-3}) \\
\times) \quad 2 & \\
\hline
\boxed{0}.88 & \text{整数 } 0\ (k_{-4}) \\
\times) \quad 2 & \\
\hline
\boxed{1}.76 & \text{整数 } 1\ (k_{-5})
\end{array}
$$

十进制数 0.18 连续四次乘 2 后，其小数部分等于 0.88，仍不为 0。由于要求精确到小数点后 4 位，因此将 0.88 再乘 2，小数点后第 5 位为 1，得

$$(0.18)_{10} \approx (0.00101)_2$$

如果一个十进制数既有整数部分又有小数部分，转换时，整数部分采用“除 2 取余”法，小数部分采用“乘 2 取整”法，然后再把转换的结果合并起来即可。

1.3.2　二进制数与八进制数、十六进制数的转换

八进制数的基数 $R=8$，3 位二进制数恰好是 8 个状态，即 $8=2^3$；十六进制数的基

数为 $R=16$，4 位二进制数恰好是 16 个状态，即 $16=2^4$。二进制数、八进制数和十六进制数之间具有 2 的整指数倍关系。因而可直接进行转换。

将二进制数转换为八进制或十六进制的方法是：从小数点开始，分别向左、右按 3 位一组转换为八进制或按 4 位一组转换为十六进制，最后不满 3 位或 4 位的则需补 0。将每组以对应的等值的八进制数或十六进制数代替，举例如下：

二进制数： 010 101 111 · 000 101 101 100

八进制数： 2 5 7 · 0 5 5 4

二进制数： 1010 1111 · 0001 0110 1100

十六进制数： A F · 1 6 C

最后得：$(257.0054)_8=(0101111.0001011011)_2=(AF.16C)_{16}$

若将八进制数或十六进制数转换成二进制数时，可按上述方法的逆过程进行。

1.3.3 码制

不同的数码不仅可以表示数量的不同大小，而且还能用来表示不同的事物。在后一种情况下，这些数码已没有表示数量大小的含意，只是表示不同事物的代号而已，这些数码称为代码。

比如在举行长跑比赛时，为便于识别运动员，通常给每个运动员编一个号码。显然，这些号码仅仅表示不同的运动员，已失去了数量大小的含意。

为便于记忆和处理，在编制代码时总要遵循一定的规则，这些规则被称为码制。

例如在用 4 位二进制数码表示 1 位十进制数的 0～9 这十个状态时，就有多种不同的码制。通常将这些代码称为二－十进制代码，简称 BCD (Binary Coded Decimal)代码。

表 1.3.1 几种常见的 BCD 代码

编码种类 / 十进制数	8421 码	余 3 码	2421 码	5211 码	余 3 循环码
0	0000	0011	0000	0000	0010
1	0001	0100	0001	0001	0110
2	0010	0101	0010	0100	0111
3	0011	0110	0011	0101	0101
4	0100	0111	0100	0111	0100
5	0101	1000	1011	1000	1100
6	0110	1001	1100	1001	1101
7	0111	1010	1101	1100	1111
8	1000	1011	1110	1101	1110
9	1001	1100	1111	1111	1010
权	8421		2421	5211	

表 1.3.1 中列出了几种常见的 BCD 代码，它们的编码规则各不相同。

8421 码是 BCD 代码中最常见的一种。在这种编码方式中每一位二值代码的 1 都代表一个固定数值，把每一位的 1 代表的十进制数加起来，得到的结果就是它所代表的十进制数。由于代码中从左到右每一位的 1 分别表示 8、4、2、1，所以把这种代码叫做 8421 码。每一位的 1 代表的十进制数称为这一位的权。8421 码中每一位的权是固定不变的，它属于恒权代码。

余 3 位码的编码规则与 8421 码不同，如果把每一个余 3 码看作 4 位二进制数，则它的数值要比它所表示的十进制数码多 3，故而将这种代码叫做余 3 码。

如果将两个余 3 码相加，所得的和将比十进制数之和所对应的二进制数多 6。因此，在用余 3 码作十进制加法运算时，若两数之和为 10，正好等于二进制数的 16，于是便从高位自动产生进位信号。此外，从表 1.3.1 中还可以看到，0 和 9，1 和 8，2 和 7，3 和 6，4 和 5 的余 3 码互为反码，这对于求取对 10 的补码是很方便的。

余 3 码不是恒权代码。如果试图把每个代码视为二进制数，并使它等效的十进制数与所表示的代码相等，那么代码中每一位的 1 所代表的十进制数在各个代码中不是固定的。

2421 码是一种恒权代码，它的 0 和 9，1 和 8，2 和 7，3 和 6，4 和 5 也互为反码，这个特点和余 3 码相仿。

5211 码是另一种恒权代码。

余 3 循环码是一种变权码，每一位的 1 在不同代码中并不代表固定的数值。它的主要特点是相邻的两个代码之间仅有一位的状态不同。因此，按余 3 循环码接成计数器时，每次状态转换过程中只有一个触发器翻转，译码时不会发生竞争－冒险现象（详见 §3.4 节）。

§1.4 带符号数的代码表示

1.4.1 真值与机器数

通常我们都用符号“+”表示正，用符号“－”表示负。“+”和“－”无非是表示两种对立的状态标志。如同计算机中可采用的“0”和“1”一样。因此，在计算机中表示正负号的最简单方法是约定用 0 表示“+”，用 1 表示“－”。

一个带符号的数由两部分组成：一部分表示数的符号；另一部分表示数的数值。对于一个 n 位二进制数。若数的第一位为符号位，则剩下的 $n-1$ 位就表示数的数值部分。一般用正号“+”和负号“－”来表示带符号的二进制数，叫做符号数的真值。数的真值形成是一种原始形式，不能直接用于计算机中。当把符号数值化后，就可在计算机中