刘英娴　宗德忠　杨延双　编著

数字逻辑

机械工业出版社
China Machine Press

数字逻辑

刘英娴　宗德忠　杨延双　编著

机械工业出版社

《数字逻辑》是大学计算机专业本科生的主干基础课程，使学生掌握数字逻辑电路的分析和设计方法。包括门级、触发器级、中大规模集成电路及可编程逻辑器件的分析、使用和设计方法等内容。全书系统全面地介绍了逻辑电路和逻辑功能器件，侧重于应用，着重介绍分析、设计方法是其特点，举例丰富，习题安排得当，图文并茂。

本书适合于高等教育及高等职业教育的本专科师生，和对计算机及其他数字系统设计感兴趣的科技人员。

图书在版编目（CIP）数据

数字逻辑/刘英娴等编著. －北京：机械工业出版社，2000
ISBN 7-111-01318-2

Ⅰ. 数… Ⅱ. 刘… Ⅲ.①数字电路-高等学校-教材 ②逻辑电路-高等学校-教材 Ⅳ.TN79

中国版本图书馆 CIP 数据核字（2000）第 63187 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
责任编辑：何文军 版式设计：张世琴 责任校对：张 佳
封面设计：方 芬 责任印制：郭景龙
三河市宏达印刷厂印刷·新华书店北京发行所发行
2000 年 8 月第 1 版·第 1 次印刷
787mm×1092mm 1/16·17.5 印张·432 千字
0 001—5 000 册
定价：27.00 元

前　言

计算机的发明和广泛应用，对20世纪人类社会的进步做出了巨大贡献。它使社会生产力发生了深刻的、新的飞跃；使社会生活产生重大变革；对社会各行各业直至家庭生活都产生了深刻影响。

计算机本身硬件和软件技术的发展日新月异，短短的半个世纪，从占地面积几百平方米，速度只有几千次/秒的电子管计算机发展到目前能在几平方毫米面积的芯片上实现上千兆次/秒、几千万个元器件的高速多机系统。这些高技术产品的产生恰恰又反映了当代数字系统设计水平和微电子技术水平的迅速提高。

数字系统设计技术的基础就是《数字逻辑》这门课要讨论的内容。

《数字逻辑》是计算机专业众多专业基础课之一，是本科生的一门主干课程。学习本课的主要目的是使学生掌握对数字逻辑电路的分析和设计方法。其中包括用门和触发器的逻辑分析及设计方法，中大规模集成电路的原理、使用方法和可编程逻辑器件的逻辑设计方法。它可为学生今后从事数字系统设计和学习“计算机组成”、“计算机体系结构”、“接口与通信”等专业课程打下良好的硬件基础。要使我国计算机技术的总体水平尽快赶上发达国家，很关键的工作就是要提高系统设计水平和集成电路生产水平，这都需要高水平的专业人才。

本书作者都是有多年第一线教学经验和实践经验的教师。对内容已几经筛选，取优汰劣。目的是在加强基础内容的基础上，适量介绍新的逻辑设计技术。按照教学大纲的营求，力图对经典内容叙述准确简捷、重点突出，同时又强调实用性，略去了很多电路的内部结构，只强调其外特性及应用。作者力求对本书的叙述深入浅出，笔调通俗易懂。便于学生预习和自学。

现在的通用标准器件种类多得数不胜数，本书不可能面面俱到。作者只能对几种常用典型产品作深入解剖，并辅以大量例题，使学生学会分析问题和解决问题的方法，便于提高独立思考和举一反三的能力。

本书的第1章由杨延双，第2、3、4章由宗德忠，第5、6、7、8章由刘英娴编写；周海旗、张维善等教授提供了大量资料；刘津瑜、尹志忠、刘怡、李福观、张旭帮助审阅和校对；刘志婷、张军帮忙录入，同时得到李大友、严化南教授的指导，特此致谢。

由于作者水平有限，书中难免有不妥之处或缺点错误。殷切希望广大读者批评指正。

编　者

2000年4月

目　录

第4章 组合逻辑电路

第5章 同步时序逻辑电路

第6章 异步时序逻辑电路

第 7 章　逻辑功能部件的应用

第 8 章　可编程逻辑器件的原理和逻辑设计

第 1 章　数制和编码

本章从人们常用的十进制数开始，分析推导出一般的进位计数制规则和各种不同的进位计数制之间的转换方法。对于在计算机及其他数字系统中广泛使用的二进制计数方法作了重点讨论。

本章将介绍几种常用的十进制数不同代码的表示方法、可靠性编码和字符代码。

1.1　进位计数制

在计算机和其他数字系统中普遍采用二进制数，有时也常采用八进制数和十六进制数，而在日常生活中，人们习惯于用十进制数。

大家都熟悉，十进制数采用“逢十进一”的进位规则，二进制数采用“逢二进一”的进位规则……。所谓进位计数制，就是按一定的进位方式实现计数的规则，简称为进位制。

1.1.1　十进制计数制

在十进制计数制中，有 0，1，2，3，4，5，6，7，8，9 十个数字符号。即基数为 10，其特点是逢十进一。

任何一个数都可以用上述 10 个数字符号按一定规律排列起来表示。每一数字符号处于不同的位置时，它代表的数值是不同的，这就是说有着不同的位权。

例如十进制数 126.5 可以写为

$$126.5 = 1\times10^2 + 2\times10^1 + 6\times10^0 + 5\times10^{-1}$$

式中，小数点左边第一位为个位，其权为 10^0，6 代表 6（$6=6\times10^0$）；第二位为十位，其权为 10^1，2 代表 20（$20=2\times10^1$）；第三位为百位，其权为 10^2，1 代表 100（$100=1\times10^2$）；小数点右边第一位为十分位，其权为 10^{-1}，5 代表 0.5（$0.5=5\times10^{-1}$）。由上可看出，相邻位的权值相差 10 倍，从右到左递增。

一般来说，任一个十进制数 N 都可用下式表示：

$$\begin{aligned}(N)_{10} &= k_{n-1}10^{n-1} + k_{n-2}10^{n-2} + \cdots + k_1 10^1 + k_0 10^0 \\ &\quad + k_{-1}10^{-1} + \cdots + k_{-m}10^{-m} \\ &= \sum_{i=-m}^{n-1} k_i 10^i \end{aligned} \tag{1.1}$$

上式称为十进制数 N 的多项式表示法，或称为按权展开式。式中，n 代表整数位数，m 代表小数位数，k_i 为 0～9 中任意一个，记作 $0\leqslant k_i\leqslant 9$。括号外下标为十进制数的基数（10）。

十进制数 N 还有一种表示法，即位置表示法，也叫并列表示法。

数 N 的位置表示法写为：

$$(N)_{10} = (k_{n-1}k_{n-2}\cdots k_1 k_0 . k_{-1}\cdots k_{-m})_{10} \tag{1.2}$$

实际的数字系统中用到的计数制并不限于十进制，其他进制的计数规律可以看成十进制计数规律的推广。对于任意 R 进制，数 N 可用下列多项式表示法表示：

$$(N)_R = (k_{n-1}R^{n-1} + k_{n-2}R^{n-2} + \cdots + k_1 R^1$$

$$+ k_0R^0 + k_{-1}R^{-1} + \cdots + k_{-m}R^{-m})_R$$

$$= \sum_{i=-m}^{n-1} k_iR^i \tag{1.3}$$

用位置表示法写为：

$$(N)_R = (k_{n-1}k_{n-2}\cdots k_1k_0k_{-1}\cdots k_{-m})_R \tag{1.4}$$

式中，n 代表整数的位数，m 代表小数位数，k_i 是 R 进制中 R 个数字符号中的任意一个，即 $0 \leqslant k_i \leqslant R-1$。

在 R 进制中，其进位规则是逢 R 进一。

1.1.2 二进制计数制

在计算机和数字系统中，数量关系大多采用二进制数字表示。在二进制计数制中，每个数位只有两个不同的取值，0 和 1。即基数为 2，其特点是逢二进一。

由于二进制计数制简单，所以其数字系统容易采用数字电路实现，且工作可靠。在二进制中，数 N 的小数点左边第一位的权为 2^0，第二位的权为 2^1，第三位的权为 $2^2\cdots$，小数点右边第一位的权为 $2^{-1}\cdots$。例如二进制数 1001.1 可以写成：

$$(1001.1)_2 = 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1}$$

上式右边即是二进制数 1001.1 的按权展开式。

对于任一个二进制数 N 都可以表示为

$$(N)_2 = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \cdots + k_12^1 + k_02^0 + k_{-1}2^{-1} + \cdots + k_{-m}2^{-m}$$

$$= \sum_{i=-m}^{n-1} k_i2^i \tag{1.5}$$

式中，n 代表整数位数，m 代表小数位数，k_i 为 0 或 1。

二进制数的好处还有其算术运算简便。

在二进制运算中，四则运算的规则有：

(1) 加法规则

$$0+0=0$$
$$0+1=1$$
$$1+0=1$$
$$1+1=10$$

(2) 减法规则

$$0-0=0$$
$$1-0=1$$
$$1-1=0$$
$$0-1=1(\text{借一当二})$$

(3) 乘法规则

$$0\times0=0$$
$$0\times1=0$$
$$1\times0=0$$
$$1\times1=1$$

（4）除法运算　二进制数的除法运算是乘法运算的逆运算。只要利用乘法规则和减法规则可以容易地实现二进制数的除法运算。

例 1.1　1001＋101＝1110

$$\begin{array}{r} 1\ 0\ 0\ 1 \\ +\quad 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 0 \end{array}$$

例 1.2　1011－101＝110

$$\begin{array}{r} 1\ 0\ 1\ 1 \\ -\quad 1\ 0\ 1 \\ \hline 1\ 1\ 0 \end{array}$$

例 1.3　1101×110＝1001110

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ \times\quad 1\ 1\ 0 \\ \hline 0\ 0\ 0\ 0 \\ 1\ 1\ 0\ 1 \\ 1\ 1\ 0\ 1 \\ \hline 1\ 0\ 0\ 1\ 1\ 1\ 0 \end{array}$$

例 1.4　110110÷101＝1010

$$\begin{array}{r} \quad\quad 1\ 0\ 1\ 0\ \text{（商）} \\ 101\ \overline{)\,1\ 1\ 0\ 1\ 1\ 0} \\ \quad 1\ 0\ 1 \\ \hline \quad\quad 1\ 1\ 1 \\ \quad\quad 1\ 0\ 1 \\ \hline \quad\quad\quad 1\ 0\ 0\ \text{（余数）} \end{array}$$

1.1.3　八进制计数制

在八进制计数制中，有 0，1，2，3，4，5，6，7 八个数字符号，即基数为 8，其特点是逢八进一。

因为 $2^3=8$，一位八进制数相当于三位二进制数，所以八进制与二进制的转换是很直观的，只要把每位八进制数用三位二进制数来表示，再组合在一起就成了相等的二进制数。例如，$(1010011011)_2$ 可表示为

$$\underbrace{1}_{1},\ \underbrace{0\ 1\ 0}_{2},\ \underbrace{0\ 1\ 1}_{3},\ \underbrace{0\ 1\ 1}_{3}\quad\text{（三位一组）}$$

（每组对应的八进制数）

即 $(1010011011)_2=(1233)_8$

例 1.5　将八进制数716转换成二进制数。

八进制：　7　　1　　6

二进制：$\overbrace{1\ 1\ 1}$　$\overbrace{0\ 0\ 1}$　$\overbrace{1\ 1\ 0}$

所以

$$(716)_8 = (111001110)_2$$

八进制数比二进制数位少，便于书写和记忆。

1.1.4 十六进制计数制

十六进制数的基数为16，采用了16个数字符号：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F。其中A对应十进制数中的10，B对应11，C对应12，D对应13，E对应14，F对应15。其特点是逢十六进一。

因为$2^4=16$，所以一位十六进制数可以用四位二进制数来表示。十六进制数0～F与二进制数的对应关系如表1.1.1所示：

表 1.1.1

十六进制数	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
二进制数	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

例 1.6 将十六进制数2AF用二进制数来表示。

十六进制：　2　　A　　F

二进制：$\overbrace{0\ 0\ 1\ 0}^{2}\ \overbrace{1\ 0\ 1\ 0}^{A}\ \overbrace{1\ 1\ 1\ 1}^{F}$

所以

$$(2AF)_{16} = (001010101111)_2$$

例 1.7 将二进制数　1111010101用十六进制数来表示。

二进制　$\underbrace{1\ 1}_{3}\ \underbrace{1\ 1\ 0\ 1}_{D}\ \underbrace{0\ 1\ 0\ 1}_{5}$

十六进制　3　　D　　5

所以，

$$(1111010101)_2 = (3D5)_{16}$$

1.2 数制转换

一个数从一种进位计数制表示法转换成另一种进位计数制表示法，称为数制转换。通常采用多项式替代法和基数乘除法。下面将这两种方法分别予以介绍。

1.2.1 多项式替代法

数N从α进制转换成β进制，可以叙述如下：首先将$(N)_\alpha$在α进制中用多项式表示法写出，然后在β进制进行计算。

例 1.8 将二进制数101101转换为十进制数。

$$\begin{aligned}(101101)_2 &= (2^5 + 2^3 + 2^2 + 2^0)_{10}\\ &= (32 + 8 + 4 + 1)_{10}\\ &= (45)_{10}\end{aligned}$$

例 1.9 将十六进制数2BC.4转换为十进制数。

$$\begin{aligned}(2BC.4)_{16} &= (2\times16^2 + B\times16^1 + C\times16^0 + 4\times16^{-1})_{10}\\ &= (512 + 176 + 12 + 0.25)_{10}\\ &= (700.25)_{10}\end{aligned}$$

多项式替代法适用于数 N 由任意进制数转换成十进制数的场合。

1.2.2 基数乘除法

数 N 从 α 进制转换成 β 进制时，若采用基数乘除法，则它的计算是在 α 进制中进行的。而且，数 N 的整数部分的转换方法采用基数除法，小数部分的转换方法采用基数乘法。分别转换后将它们合并起来。

1. 基数除法（整数转换）

基数除法就是将整数部分采用“除基取余”法。即把数 $(N)_\alpha$ 的整数部分在 α 进制中连续用 β 进制的基数去除，直到商为 0 止。再把每次除得余数依次排列起来（最后得到的余数为最高位），便得到了 β 进制的整数部分。

例 1.10 将十进制数27转换成二进制数。

```
2 | 2  7  ..... 余数 1      (最低位)
  2 | 1  3  ......1
    2 | 6  ......0
      2 | 3  ......1
        2 | 1  ......1      (最高位)
          | 0
```

所以，

$$(27)_{10} = (11011)_2$$

例 1.11 将十进制数1935转换为十六进制数。

```
16 | 1 9 3 5  ..... 余数 F    (最低位)
   16 | 1 2 0  ......8
      16 | 7  ......7         (最高位)
         | 0
```

所以，

$$(1935)_{10} = (78F)_{16}$$

2. 基数乘法（小数转换）

基数乘法就是将小数部分采用“乘基取整”法。即把数 $(N)_\alpha$ 的小数部分在 α 进制中连续用 β 进制的基数去乘，直到所要求的精度为止。再将每次乘得积的整数部分依次排列起来（最先得到的整数部分为小数点后的第一位），便得到了 β 进制的小数部分。

例 1.12 将十进制数0.125转换成二进制数

```
   0.125
×      2
---------
  0.250   ……整数部分＝0
```

$$
\begin{array}{r}
0.25 \\
\times \quad 2 \\
\hline
0.50 \quad \cdots\cdots \text{整数部分}=0 \\
0.5 \\
\times \quad 2 \\
\hline
1.0 \quad \cdots\cdots \text{整数部分}=1
\end{array}
$$

所以，

$$(0.125)_{10} = (0.001)_2$$

最后，再举一个例子，数 N 既有整数部分又有小数部分，则将它们分别转换，然后合并，就得到了所求结果。

例 1.13 将十进制数 2803.4321 转换成十六进制数。(取小数点后四位)

$$(2803.4321)_{10} = (2803)_{10} + (0.4321)_{10}$$

先将整数部分 $(2803)_{10}$采用“除 16 取余法”转换成十六进制，得

$$(2803)_{10} = (AF3)_{16}$$

再将小数部分 $(0.4321)_{10}$采用“乘 16 取整法”转换成十六进制，得

$$(0.4321)_{10} = (0.6E9E)_{16}$$

所以，

$$(2803.4321)_{10} = (AF3.6E9E)_{16}$$

基数乘除法通常适用于数 N 从十进制数转换成任意进制数。

如果一个数在两种任意进制（两者都是非十进制数）之间进行转换，那么一种方便的方法就是利用十进制作桥梁。即先将 α 进制数转换成十进制数，再将相应的十进制数转换为 β 进制数。

1.3 带符号数的代码表示

前面讨论的二进制数，都没有考虑符号，也就是无符号数的表示。这里介绍带符号数的表示。

1.3.1 机器数与真值

一个带符号的数的表示，通常是在其数值的左面写上符号，即正数为“+”号（可省略），负数为“-”号。例如

$$x_1 = +0.1011 = 0.1011$$

$$x_2 = -0.1101$$

这样一个连同符号位在一起作为一个数的表示形式，称为带符号数的真值。

数的符号在机器中的表示法为，正数符号用“0”表示；负数符号用“1”表示。这样就把数的符号数字化了，将一个数的数值部分和符号部分用统一代码表示的形式，称之为机器数。例如 $x = -0.0111$，其机器数的表示为 1.0111。

机器数的表示形式有三种，即原码、反码和补码。

1.3.2 原码

一个二进制数加上符号位，且正数的符号位用“0”表示，负数的符号位用“1”表示，

就得了这个数的原码。带符号二进制数的原码表示与它的真值表示很相似。

若有小数 $x=\pm 0.x_{-1}x_{-2}\cdots x_{-m}$，它的原码表示可记为

$$[x]_{原}=x_0.x_{-1}x_{-2}\cdots x_{-m}$$

其中，x_0 表示该小数的符号。

$$x_0=\begin{cases}0 & (x\geqslant 0)\\ 1 & (x\leqslant 0)\end{cases}$$

小数 x 原码的表达形式是：

$$[x]_{原}=\begin{cases}x & (1>x\geqslant 0)\\ 1-x & (0\geqslant x>-1)\end{cases} \tag{1.6}$$

例 1.14 写出 $x=+0.0101$ 的原码

$$[x]_{原}=0.0101$$

例 1.15 写出 $x=-0.0101$ 的原码

$$\begin{aligned}[x]_{原}&=1-(-0.0101)\\ &=1+0.0101\\ &=1.0101\end{aligned}$$

在原码表示中，真值零有两种表示形式，即 +0（正零）和 −0（负零）。

$$[+0]_{原}=0.00\cdots 0 \qquad [-0]_{原}=1.00\cdots 0$$

它们的值是相等的。

原码表示简明直观，与原来的数码转换方便。但原码的算术运算比较复杂，特别是两个异号数相加（或两个同号数相减），就要做减法。在做数值的减法运算时，还必须比较两个数中哪个绝对值大，才能确定哪个是被减数、哪个是减数，这就增加了复杂性。为了把减法运算转化为加法运算就引进了反码和补码。

1.3.3 反码

二进制小数的反码表示法规定如下：

(1) 对于正数，反码的表示形式和原码一样。

(2) 对于负数，符号位为“1”，数值部分按位取反（即 1 变 0，0 变 1）。

二进制小数 x 的反码的表达形式为

$$[x]_{反}=\begin{cases}x & (1>x\geqslant 0)\\ 2-2^{-m}+x & (0\geqslant x>-1)\end{cases} \tag{1.7}$$

其中，m 代表二进制小数数值的位数。

例 1.16 写出下列各数的反码。

$$x_1=+0.0101 \quad x_2=-0.0101$$

则

$$[x_1]_{反}=0.0101 \quad [x_2]_{反}=1.1010$$

也可通过式 (1.6) 得到 $[x_2]_{反}$。

$$\begin{aligned}[x_2]_{反}&=2-2^{-4}+(-0.0101)\\ &=2-0.0001-0.0101\\ &=1.1111-0.0101\\ &=1.1010\end{aligned}$$

在反码表示中，零有两种形式，即正零和负零。

$$[+0]_{反} = 0.00\cdots0 \qquad [-0]_{反} = 1.11\cdots1$$

它们的真值是相等的。

1.3.4 补码

二进制小数的补码表示法规定如下：

(1) 对于正数，其补码的形式和原码一样。

(2) 对于负数，其符号位为“1”，数值部分则是按位求反，最低位加 1，也就是“变反加 1”。

二进制小数 x 的补码的表达形式为

$$[x]_{补} = \begin{cases} x & (0 \leqslant x < 1) \\ 2 + x & (-1 \leqslant x < 0) \end{cases} \tag{1.8}$$

例 1.17 写出 $x = -0.0101$ 的补码。

$$[x]_{补} = 1.1011$$

也可通过式 1.8 得到：

$$[x]_{补} = 2 + (-0.0101) = 2 - 0.0101 = 1.1011$$

在补码表示中，真值零的补码是唯一的，即 0.00…0。

用补码进行运算的结果仍为补码。对补码再变补可以得到原码。

1.3.5 二进制整数的代码表示

对于二进制整数的代码表示，与二进制小数的代码表示是类似的。对 n 位二进制整数，在它左边添加一位符号位，即其代码表示共有 $n+1$ 位。符号位用“0”表示正数，用“1”表示负数。

(1) 二进制整数 x 的原码表达形式是：

$$[x]_{原} = \begin{cases} x & (2^n > x \geqslant 0) \\ 2^n - x & (0 \geqslant x > -2^n) \end{cases}$$

例如

$$x_1 = +1101001$$

$$x_2 = -1101001$$

则

$$[x_1]_{原} = 01101001$$

$$[x_2]_{原} = 2^7 - (-1101001)$$

$$= 10000000 + 1101001$$

$$= 11101001$$

(2) 二进制整数 x 的反码的表达形式是：

$$[x]_{反} = \begin{cases} x & (2^n > x \geqslant 0) \\ (2^{n+1} - 1) + x & (0 \geqslant x > -2^n) \end{cases}$$

例如 写出 $x = -1100011$ 的反码。

则

$$[x]_{反} = (2^{7+1} - 1) + (-1100011)$$

$$= (100000000 - 1) + (-1100011)$$

$$= 11111111 - 1100011$$

$$= 10011100$$

(3) 二进制整数 x 的补码的表达形式是：

$$[x]_{补}=\begin{cases}x & (2^n > x \geqslant 0)\\ 2^{n+1}+x & (0 > x \geqslant -2^n)\end{cases}$$

例 1.18 $[x]=-1001101$ 求 $[x]_{补}$。

$$\begin{aligned}[x]_{补}&=2^{7+1}+(-1001101)\\&=100000000-1001101\\&=10110011\end{aligned}$$

1.3.6 带符号二进制数的加、减运算

原码的加减法运算有不同的规则，运算比较复杂，为了简化运算，通常采用补码运算或反码运算。

例 1.19 $x=+0.0101$，$y=+0.0011$，求 $x-y$。

$$x-y=x+(-y)$$

(1) 补码运算

$$[x]_{补}=0.0101 \qquad [-y]_{补}=1.1101$$

$$\begin{array}{r} 0.\ 0\ 1\ 0\ 1\\ +1.\ 1\ 1\ 0\ 1\\ \hline 1\ 0.\ 0\ 0\ 1\ 0\end{array}$$

丢掉← 1

则得：$x-y=0.0010$

在补码运算时，符号位产生的进位要丢掉。

(2) 反码运算

$$[x]_{反}=0.0101 \qquad [-y]_{反}=1.1100$$

$$\begin{array}{r} 0.\ 0\ 1\ 0\ 1\\ +1.\ 1\ 1\ 0\ 0\\ \hline 1\ 0.\ 0\ 0\ 0\ 1\\ \qquad\qquad\longrightarrow 1\\ \hline 0.\ 0\ 0\ 1\ 0\end{array}$$

则得：$x-y=0.0010$

在反码运算时，符号位产生的进位要加到数位的最低位上去，才能保证运算结果的正确性。

1.4 常用十进制数的二进制编码

用 4 位二进制数来表示一位十进制数的方法，称为十进制数的二进制编码，简称 BCD 码。它具有二进制的形式，又具有十进制数的特点。这里介绍几种常用的 BCD 码。

1.4.1 8421 码

8421 码也称为二-十进制码或 BCD 码。它将十进制数的每个数字用四位二进制数表示。该码的位权值从左到右分别为 8 (2^3)、4 (2^2)、2 (2^1)、1 (2^0)。它在十进制十个数字的表示中与普通的二进制中的表示完全一样。这样，十进制数 0～9 用二进制数的 0000～1001 来

分别表示。因此在8421码中，1010～1111这六种代码是不可能出现的。

设8421码四位二进制数码分别为 $a_3a_2a_1a_0$，则它所表示的一位十进制数 D 为：

$$D = 8a_3 + 4a_2 + 2a_1 + 1a_0$$

8421码和十进制数之间的转换是一种直接按位（或组）来进行的。

例 1.20 将十进制数36转换成8421码表示。

$$(36)_{10} = (\underline{0011}\ \ \underline{0110})8421$$
$$\uparrow \quad \uparrow$$
$$3 \quad 6$$

例 1.21 将8421码001110010101转换成十进制数表示。

$$(\underline{0011}\ \ \underline{1001}\ \ \underline{0101})_{8421} = (395)_{10}$$
$$\uparrow \quad \uparrow \quad \uparrow$$
$$3 \quad 9 \quad 5$$

1.4.2 2421码

2421码也是将十进制数的每个数字用四位二进数表示。该码的位权值从左到右分别为2、4、2、1。2421码有多种编码方案，例如：十进制数5的2421码可以编成1011，也可以编成0101。典型的2421码编码方案见表1.4.1所示。

表 1.4.1

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

设2421码的四位二进制数码分别为 $a_3a_2a_1a_0$，则它所表示的一位十进制数 D 为：

$$D = 2a_3 + 4a_2 + 2a_1 + 1a_0$$

2421码和十进制数之间的转换也是直接按位（或组）进行。

例 1.22 将十进制数536转换成2421码表示。

$$(536)_{10} = (\underline{1011}\ \ \underline{0011}\ \ \underline{1100})_{2421}$$
$$\uparrow \quad \uparrow \quad \uparrow$$
$$5 \quad 3 \quad 6$$

2421码是一种自补代码，即用2421码表示的数，只要自身按位求反，就能得到该数对9之补的2421码。

一位十进制数字对9之补，即求取9与该数字的差。例如，2的对9之补是9－2＝7，反之，7对9之补是2；1的对9之补是9－1＝8，反之8对9之补是1。

这样，2的2421码是0010，将其按位求反，得到的是7的2421码1101；而1的2421码是0001，将其按位求反，得到的是8的2421码1110。

1.4.3 余3码

余3码同样也是将十进制数的每个数字用四位二进制数表示。但对同一个十进制数字，余3码和相应的8421码相比多出0011。也就是说，由8421码加上0011（十进制数3）得到