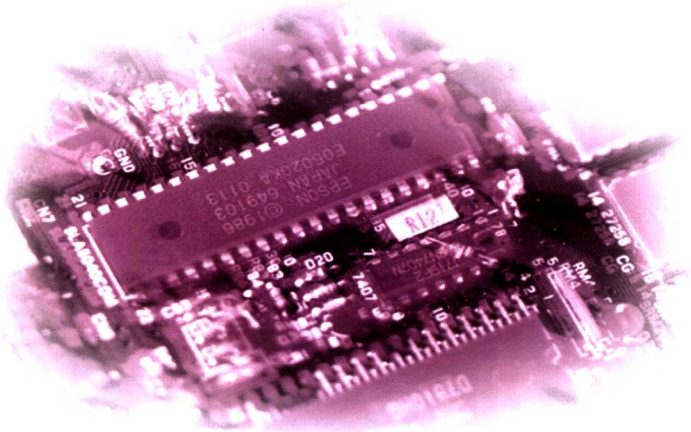
高职高专电子技术系列教材

数字电路

SHUZI DIANLU

副主编 马 杰 王海云
主 编 唐 颖



重庆大学出版社

数字电路

主　编　唐　颖
副主编　马　杰　王海云

重庆大学出版社

内 容 简 介

本书为高职高专电子技术专业系列教材之一。全书共分8章，即数字逻辑基础、门电路、组合逻辑电路、触发器、时序逻辑电路、脉冲信号的产生与整形、A/D与D/A转换器、半导体存储器及可编程逻辑器件简介。其中第1章为数字电路的理论基础，包含数制的转换、补码运算、逻辑代数等。第3章、5章是全书的重点，各章后均配有习题。

本书可作为高等专科学校、高等职业学校信息类、电类专业的技术基础课教材，也可作为从事电子技术的工程人员的参考书。

图书在版编目(CIP)数据

数字电路/唐颖主编. —重庆：重庆大学出版社，2004.7
(高职高专电子技术系列教材)
ISBN 7-5624-3085-3

Ⅰ.数… Ⅱ.唐… Ⅲ.数字电路—高等学校：技术学校—教材 Ⅳ.TN79

中国版本图书馆CIP数据核字(2004)第050937号

数字电路

主 编 唐 颖
副主编 马 杰 王海云
责任编辑：周 立 版式设计：周 立
责任校对：廖应碧 责任印制：张立全

*

重庆大学出版社出版发行
出版人：张鸽盛
社址：重庆市沙坪坝正街174号重庆大学(A区)内
邮编：400030
电话：(023) 65102378 65105781
传真：(023) 65103686 65105565
网址：http://www.cqup.com.cn
邮箱：fxk@cqup.com.cn (市场营销部)
全国新华书店经销
重庆华林天美彩色报刊印务有限公司印刷

*

开本：787×1092 1/16 印张：13.75 字数：343千
2004年7月第1版 2004年7月第1次印刷
印数：1—5 000
ISBN 7-5624-3085-3/TN·81 定价：19.00元

前言

本书是根据教育部高职高专培养应用型、技能型人才的目标及对本课程的基本要求编写的21世纪高职高专电子技术专业系列教材之一。

本课程的主要任务是阐明数字逻辑电路的基本概念、基本原理和基本分析方法。全书共分8章，主要内容包括数字逻辑基础（含数制转换和逻辑代数）、门电路（含TTL、MOS集成逻辑门电路）、组合逻辑电路、触发器、时序逻辑电路、脉冲信号的产生与整形、数/模（D/A）及模/数（A/D）转换、半导体存储器及可编程逻辑器件简介（含RAM、ROM、PAL、GAL等）。

为突出高职高专的人才培养目标和教学特点，本书本着"必须、够用"的原则，从职业岗位对专业知识的需要来确定教材的知识深度及范围，基本概念、基本原理以讲明为度，淡化集成电路内部结构及内部工作原理的论述，主要介绍其外部特性及使用方法。同时注重知识的应用价值在教材中的科学体现。因随着中、大规模集成电路的快速发展和广泛应用，数字电路的设计方法在概念上发生了根本的变化。因此，本书以介绍逻辑分析方法为主，逻辑电路的设计主要考虑中、大规模集成电路的选用和运用。编写时力求简明扼要，通俗易懂，便于自学。各章附有小结和习题，书末附有部分习题答案。

本书第3、6章及附录由唐颖编写；第5章由马杰编写；第7章由王海云编写；第1章由范泽良编写；第2章由朱亚利编写；第4章由廖雄燕编写；第8章由贺天柱编写。唐颖负责全书的统稿。

限于编者水平，疏漏欠妥之处在所难免，敬请读者批评指正。

编　者

2004年1月

目录

第 1 章
数字逻辑基础

本章首先给出数字电路的定义，然后从常用的十进制数开始，分析推导出各种不同数制的表示方法以及各种数制之间的转换方法，进而讨论了几种常用的编码和符号数的补码表示法。在介绍逻辑代数的基本概念、公式和定理的基础上，着重讲解逻辑函数的公式化简法和图形化简法，最后介绍和归纳了逻辑函数的5种常用表示方法及其相互转换。

1.1 数字电路概述

1.1.1 模拟电路与数字电路

电子线路中的信号可分为两类：一类是时间的连续函数，称为模拟信号，例如，模拟声音的音频信号和模拟图像的视频信号、温度、速度、压力、电磁场等物理量转变成的电信号等等。对模拟信号进行发送、传输、接收和处理的电子线路称为模拟电路，如交、直流放大器、滤波器、信号发生器等。另一类是时间和幅度都是离散的信号，称为数字信号，例如，汽车上的速度读数表，工厂产品数量的统计等，都属于数字信号。而对数字信号进行发送、传输、接收和处理的电子线路称为数字电路，这就是本书数字部分将要讨论的内容。

1.1.2 数字电路举例

数字电路大致包括信号的产生、放大、整形、传输、控制、存储、计数和运算等等。图1.1为一个数字频率计的方框图。

它是用来测量周期信号频率的。假定被测信号为正弦波，它的频率为f_x。为了要把被测信号的频率用数字直接显示出来，首先得将被测的模拟信号放大、整形，使被测信号变换成同频率的矩形脉冲信号。既然是测量频率，则还需要有个时间标准，以秒为单位，把1秒内通过的脉冲个数记录下来，就得出了被测信号的频率。这个时间标准由脉冲发生器产生，它是宽度为1秒的矩形脉冲。由秒脉冲来控制门电路，又由门电路来控制电路的开通与关断。这样秒脉冲把门电路打开1秒，在这1秒内，整形后的矩形脉冲通过门电路进入计数器，计数器累计的信号个数就是被测信号在1秒内重复的次数，即信号的频率。最后通过显示器显示出来。

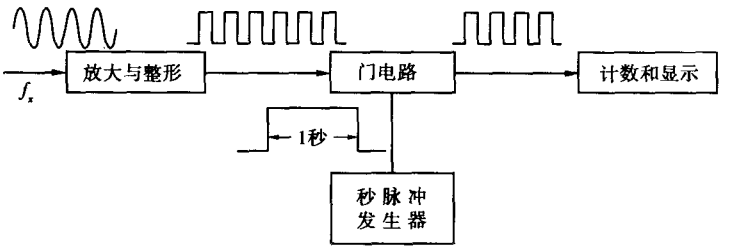


图 1.1　数字频率计方框图

1.1.3　数字电路的特点

数字电路的工作信号是离散变化的数字信号，所以在数字电路中工作的半导体管多数工作在开关状态，即工作在饱和区和截止区，而放大区只是其过渡状态。

在数字电路中，通常将高电位称为高电平，低电位称为低电平。在实际的数字电路中高电平在 3.5 V 左右，低电平在 0.3 V 左右。由于在数字电路中是用二进制来传输和处理信息，因而用 1 和 0 来代表高电平和低电平，这种用 1 表示高电平，0 表示低电平的对应关系称为正逻辑关系。反之，用 0 代表高电平，1 代表低电平的对应关系称为负逻辑关系，本书一般采用正逻辑关系（注意：这里的 1 和 0 只代表状态，不表示数字的大小关系）。

数字电路主要研究的是输出信号的状态（0 或 1）与输入信号的状态之间的关系，因而不能采用模拟电路的分析方法，例如，微变等效电路法等就不适用了。数字电路的主要分析工具是逻辑代数，而数字电路功能的主要表达形式有真值表、逻辑表达式、卡诺图、逻辑电路图以及波形图 5 种形式。

1.2　数制与编码

1.2.1　进位计数制

数字电路中经常遇到计数问题。

按进位的原则进行计数，称为进位计数制。每一种进位计数制都有一组特定的数码，例如十进制有 10 个数码，二进制只有两个数码，而十六进制数却有 16 个数码。每一种进位计数制中允许使用的数码的总数称为基数或底数，对应地，十进制数的基数为 10，二进制的基数为 2，十六进制的基数则为 16。在任何一种进位计数制中，任何一个数都由整数和小数两部分组成，并且具有两种表示方式：位置记数法和多项式表示法。

例如十进制数　100（位置计数法）$=1\times10^2+0\times10^1+0\times10^0$（多项式表示法）

在日常生活中，我们习惯于用十进制来计数，而在数字系统中则多采用二进制计数，有时也采用八进制和十六进制数。

（1）十进制数（Decimal）

所谓十进制就是以 10 为基数的计数体制。它是采用十个不同的数码 0，1，2，3，…，9 以及小数点（.）来表示的数。任何一个数都可以用这十个数码按一定的排列规律来表示，其进位

规律为“逢十进一”，即 9 + 1 = 10，右边的“0”为个位数，左边的“1”为十位数，也就是 $10 = 1 \times 10^1 + 0 \times 10^0$。这样，每一个数码处在不同的位置时，它代表的数值是不同的。

例如，将十进数 425.713 写成幂的形式为

$$425.713 = 4 \times 10^2 + 2 \times 10^1 + 5 \times 10^0 + 7 \times 10^{-1} + 1 \times 10^{-2} + 3 \times 10^{-3}$$

一般来说，任意十进制数可表示为

$$(N)_{10} = \sum_{i=-\infty}^{\infty} K_i \times 10^i \tag{1.1}$$

式中，$(N)_{10}$中的下标 10 表示该数为十进制数，以后的数均采用这种表示方式。K_i为基数“10”的第 i 次幂的系数。

在数字电路中采用十进制计数是不方便的。因为数字电路的基本出发点就是把电路的状态用数码一一对应起来，而十进制有十个数码，就必须在电路中找到十个严格区别的状态与之对应，这样对技术要求很高不说，而且也不经济。这点大家将在以后的学习中逐渐体会到。因而在数字电路中一般不直接采用十进制，而多采用二进制数。

(2)二进制数(Binary)

和十进制类似，所谓二进制就是以 2 为基数的计数体制。它仅有两个数码 0 和 1，其计数规律为“逢二进一”，即 1 + 1 = 10(读为“壹零”)，左边的“1”代表 1×2^1，右边的“0”代表 0×2^0。每个数码处于不同的位置时，它代表的数值是不同的，这与十进制相同，只是基数由“10”换为了“2”。

例如，二进制数的 1010.01 可表示为

$$\begin{aligned}(1010.01)_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= 8 + 0 + 2 + 0 + 0 + 0.25 = (10.25)_{10}\end{aligned}$$

同样地，任意二进制数可表示为

$$(N)_2 = \sum_{i=-\infty}^{\infty} K_i \times 2^i \tag{1.2}$$

式中，$(N)_2$中的下标 2 表示该数为二进制数，K_i为基数“2”的第 i 次幂的系数。这样，就可以用该公式将任何二进制数转换成十进制数。

在数字电路中，与十进制相比，二进制具有一定的优点：

1)用二进制设计的数字电路简单可靠，所用的元件少。

二进制只有两个数码 0 和 1，因此它的每一个位数都可以用任何具有两个不同稳定状态的元件来表示，而这类元件是非常普通的。例如，二极管的导通与断开，三极管的饱和与截止，开关的闭合与断开等。我们只需规定其中一种状态为 1，则另一种状态就可以用 0 来表示。由于只有两种状态，数码的传输与存储都非常简单。

2)二进制的基本运算非常简单，这一点大家将在随后的学习中体会到。

二进制数最大的缺点是表述一个数时位数太多，书写和记忆都不方便。十进制数虽然可以表示二进制数，但十进制数与二进制数之间的转换却较为复杂，一般不被人们所采用，因而在数字电路中引进了十六进制数和八进制数来表示二进制数。

(3)十六进制(Hexadecimal)和八进制(Octal)

同十进制数和二进制数一样，十六进制数就是以 16 为基数的计数体制。它有 16 个不同

的数码:0,1,2,3,…,9,A(与十进制的10相对应),B(11),C(12),D(13),E(14),F(15),其进位规律为“逢十六进一”,即F+1=10,左边的“1”表示1×16^1,右边的“0”表示0×16^0。

例如,将十六进制数15E转换成十进制数为

$$(15E)_{16}=1\times16^2+5\times16^1+14\times16^0=(350)_{10}$$

任意十六进制数可用公式表示为

$$(N)_{16}=\sum_{i=-\infty}^{\infty}K_i\times16^i \tag{1.3}$$

同理,八进制就是以8为基数的计数体制;它有8个数码:0,1,2,3,4,5,6,7,进位规律为“逢八进一”,即7+1=10,左边的“1”表示1×8^1,右边的“0”表示0×8^0。

例如,将八进制数624转换为十进制为

$$(624)_8=6\times8^2+2\times8^1+4\times8^0=(404)_{10}$$

用公式可将八进制数表示为

$$(N)_8=\sum_{i=-\infty}^{\infty}K_i\times8^i \tag{1.4}$$

十进制、二进制、八进制、十六进制之间的关系如表1.1所示。

表1.1 几种进制之间的关系对照表

十进制	二进制	八进制	十六进制
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	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*

1.2.2 数制之间的相互转换

既然同一个数可以用上述的几种数制来表示,那么这几种数制之间就必然有一定的转换关系,下面将介绍这几种数制之间的转换方法。

(1)二进制与十进制之间的转换

1)二进制数转换成十进制数:只要将二进制数按式(1.2)展开,然后将各项数值按十进制数相加便可以得到等值的十进制数。

例1.1 $(10110.11)_2=1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+1\times2^{-2}$

$$=16+0+4+2+0+0.5+0.25$$
$$=(22.75)_{10}$$

2）十进制数转换成二进制数

①整数部分的转换——除2取余法。若将二进制整数$(M)_2=b_n\times2^n+b_{n-1}\times2^{n-1}+\cdots+b_1\times2^1+b_0\times2^0$，转换成十进制数$(N)_{10}$，则可以写成$(N)_{10}=(M)_2$，即

$$(N)_{10}=b_n\times2^n+b_{n-1}\times2^{n-1}+\cdots+b_1\times2^1+b_0\times2^0$$
$$=2\times(b_n\times2^{n-1}+b_{n-1}\times2^{n-2}+\cdots+b_1\times2^0)+b_0$$
$$=2Q+b_0$$

若将等式两端同时除以2，则商为$Q=b_n\times2^{n-1}+b_{n-1}\times2^{n-2}+\cdots+b_1\times2^0$，余数为$b_0$。如果再将$Q$除以2，其余数为$b_1$，依此类推，将十进制数的整数每除以一次2就可以根据余数得到一位二进制数。因此，只要连续除以2直到商为0，就可由所得的余数按$b_nb_{n-1}\cdots b_1b_0$的顺序组合成对应的二进制数。

例1.2　将$(42)_{10}$转换成二进制数：

	商	余数	位置
2 | 42	21	0	b_0
2 | 21	10	1	b_1
2 | 10	5	0	b_2
2 | 5	2	1	b_3
2 | 2	1	0	b_4
2 | 1	0	1	b_5

所以　$(42)_{10}=(b_5b_4b_3b_2b_1b_0)_2=(101010)_2$

②小数部分的转换——乘2取整法。若将二进制数的小数$(M)_2=b_{-1}\times2^{-1}+b_{-2}\times2^{-2}+\cdots+b_{-n}\times2^{-n}$转换成十进制的小数$(N)_{10}$，则可以写成

$$(N)_{10}=b_{-1}\times2^{-1}+b_{-2}\times2^{-2}+\cdots+b_{-n}\times2^{-n}$$

将等式两边同时乘以2可得

$$2(N)_{10}=b_{-1}+2(b_{-2}\times2^{-1}+b_{-3}\times2^{-2}+\cdots+b_{-n}\times2^{-(n-1)})$$

则可以得到整数b_{-1}。用同样的方法，先去掉十进制数上次乘以2后的整数部分，用剩下的小数部分再乘一次2，就又可以根据整数得到一位二进制数，依此连续乘以2，直到满足误差要求进行“四舍五入”为止，然后按照$b_{-1}b_{-2}\cdots b_{-n}$的顺序排列后在高位添上0和小数点(.)就完成了十进制小数向二进制小数的转换。

例1.3　可按如下步骤将$(0.73)_{10}$转换成误差ε不大于2^{-6}的二进制数：

	小数	整数	位置
$0.73\times2=1.46$	0.46	1	b_{-1}
$0.46\times2=0.92$	0.92	0	b_{-2}
$0.92\times2=1.84$	0.84	1	b_{-3}
$0.84\times2=1.68$	0.68	1	b_{-4}
$0.68\times2=1.36$	0.36	1	b_{-5}

最后的小数为0.36小于0.5，根据“四舍五入”的原则，b_{-6}应该为0，所以，$(0.73)_{10}=(0.10111)_2$，其误差ε小于2^{-6}。

综上可知，如果要将既有小数又有整数的十进制数转换成二进制数，则只需将小数与整数分别转换成二进制后，再求和即可。

例1.4 将$(10.25)_{10}$转换成二进制数：

解： 因 $(10.25)_{10}=(10)_{10}+(0.25)_{10}$

又因 $(10)_{10}=(1010)_2,(0.25)_{10}=(0.01)_2$

故 $(10.25)_{10}=(1010)_2+(0.01)_2=(1010.01)_2$

同理，十进制转换成八进制、十六进制时，整数部分分别采用除8取余与除16取余法，小数部分则分别采用乘8取整与乘16取整法。

(2)二进制与八进制、十六进制之间的转换

1)二进制与八进制之间的转换

八进制数的基数恰好是$8=2^3$，所以三位二进制数恰好对应一位八进制数，它们之间的转换很方便。

二进制转换成八进制：①对于二进制的整数部分，由低位向高位，每三位为一组，若高位不足三位，则在高位添“0”补足三位，这样每三位二进制就对应一位八进制。②对于二进制小数部分，则由高位向低位，每三位为一组，低位不足三位，在低位添“0”补足三位，这样每三位二进制就对应一位八进制。

例1.5 求$(10010101.1011)_2$等值的八进制数：

二进制 010 010 101. 101 100

八进制 2 2 5 . 5 4

所以 $(10010101.1011)_2=(225.54)_8$

八进制转换成二进制：每一位八进制对应三位二进制，将转换后的二进制数去掉整数部分高位的“0”和小数部分低位的“0”就是最终结果。

例1.6 求$(326.74)_8$等值的二进制数。

八进制 3 2 6 . 7 4

二进制 011 010 110 . 111 100

去掉整数部分高位的“0”和小数部分低位的“0”后得11010110.1111

即 $(326.74)_8=(11010110.1111)_2$

2)二进制与十六进制之间的转换

同理，十六进制数的基数恰好是$16=2^4$，所以四位二进制数恰好对应一位十六进制数，它们之间的转换规则同二进制与八进制之间的转换类似。只需在转换中将三位二进制为一组换成四位二进制为一组即可。

例1.7 将$(11110010001.11011)_2$转换成等值的十六进制。

二 进 制 0111 1001 0001. 1101 1000

十六进制 7 9 1 . D 8

所以 $(11110010001.11011)_2=(791.D8)_{16}$

例1.8 将$(A2.D6)_{16}$转换成等值的二进制数。

十六进制　A　2　.　D　6

二 进 制 1010 0010 . 1101 0110

即　$(A2.D6)_{16}=(10100010.1101011)_2$

另外，十进制数转换成八进制和十六进制数，可以先将其转换成二进制数，然后由二进制转换成八进制或十六进制数。

例1.9　将$(60.75)_{10}$转换成二进制、八进制和十六进制数。

解： 1)转换成二进制

$$(60.75)_{10}=(60)_{10}+(0.75)_{10}=(111100)_2+(0.11)_2=(111100.11)_2$$

2)转换成八进制

二进制：111 100. 110

八进制：7　4．6

3)转换成十六进制

二 进 制：0011 1100. 1100

十六进制：3　C　.　C

所以　$(60.75)_{10}=(111100.11)_2=(74.6)_8=(3C.C)_{16}$

1.2.3　编　码

在数字系统中，任何数据和信息都是用若干位“0”和“1”组成的二进制来表示的，n位二进制可以构成2^n种不同的组合，代表2^n种不同的数据或信息。编码就是用若干位二进制的码元按一定的规律排列起来表示给定信息或数据的过程。若需要编码的信息有N项，则与所需二进制的位数n之间的关系如下：

$$2^n \geqslant N$$

下面就介绍一下几种常见的编码。

(1)二-十进制编码(BCD码)

在这种编码中，用四位二进制$b_3b_2b_1b_0$来表示十进制中的0～9共十个数码($2^4 \geqslant 10$)。由于四位二进制有16种不同的组合，而只需其中10种组合来表示十进制的十个码元，余下6种组合不用。由16种组合中选10种组合有

$$C_{16}^{10}=\frac{16!}{(16-10)!}\approx 2.9\times 10^{10}$$

种编码方案，但并不是所有的方案都有实用价值。在此，仅介绍几种人们常用的BCD码的编码方式。

1)8421 BCD码

8421 BCD码是最基本、最常用的BCD码，它和自然的二进制码相似，各位的权值(二进制数码每位的值称为权或位权)分别为8,4,2,1，故称为有权码；和自然二进制码不同的是，它只选了四位二进制码中的前十组代码，即用0000～1001来分别表示它所对应的十进制数的十个码元0,1,2～9；余下1010～1111六种组合不用。8421 BCD码的编码方式是惟一的。

2)5421 BCD码、2421BCD码和余3码

5421BCD码和2421BCD码也是有权码，各位的权值分别为5、4、2、1和2、4、2、1。在5421 BCD码中，有一些数字，如5，既可以用0101(即$0\times5+1\times4+0\times2+1\times1=5$)表示，也可以用

1000(即 $1\times5+0\times4+0\times2+0\times1=5$)表示;同样 2421 BCD 码中也有这样的数字;这说明这两种编码的编码方式不是惟一的。在表 1.2 中只列出了其中的一种方案。

余 3 码是由 8421 BCD 码的每组数码分别加 0011(即十进制的 3)得到的,是一种无权码,它的编码方式是惟一的。

例 1.10 用 8421 BCD 码和余 3 码来表示十进制数 $(572.38)_{10}$

解: 1)转换成 8421 BCD 码:一位十进制用四位二进制来表示,故

十进制: 5 7 2 . 3 8

8421 BCD 码:0101 0111 0010.0011 1000

即 $(572.38)_{10}=(0101\ 0111\ 0010.0011\ 1000)_{8421\ BCD}$

2)转换成余 3 码:将 8421 BCD 码以小数点为分界线,按四位二进制为一组分组,然后将每一组加 3(即二进制的 0011)即可。

8421 BCD 码:0101 0111 0010.0011 1000

每组加 3: 0011 0011 0011.0011 0011

余 3 码: 1000 1010 0101.0110 1011

所以 $(572.38)_{10}=(0101\ 0111\ 0010.0011\ 1000)_{8421\ BCD}$

$=(1000\ 1010\ 0101.0110\ 1011)_{余3}$

例 1.11 将二进制数 $(1001\ 0111)_2$ 用 8421 BCD 码表示。

表 1.2 几种常见的码

$b_3b_2b_1b_0$ $2^3\,2^2\,2^1\,2^0$	对应的十进制数				
	十进制数	二-十进制数			
		8421 码	2421 码	5421 码	余 3 码
0000	0	0	0	0	×
0001	1	1	1	1	×
0010	2	2	2	2	×
0011	3	3	3	3	0
0100	4	4	4	4	1
0101	5	5	×	×	2
0110	6	6	×	×	3
0111	7	7	×	×	4
1000	8	8	×	5	5
1001	9	9	×	6	6
1010	10	×	×	7	7
1011	11	×	5	8	8
1100	12	×	6	9	9
1101	13	×	7	×	×
1110	14	×	8	×	×
1111	15	×	9	×	×

注:式中的 × 表示该种二进制组合不用或不会出现。

解: 1)由于 8421 BCD 是用来表示十进制的,因而先得将二进制转换成十进制

$$(1001\ 0111)_2=(151)_{10}$$

2)将对应的十进制转换成 8421BCD 码

十进制：　　1　5　1

8421 BCD 码:0001 0101 0001

即　　$(1001\ 0111)_2 = (1\ 0101\ 0001)_{8421\ BCD}$

同样地,如果用 BCD 码来表示八进制或十六进制数,也得先将其转换成十进制,再将十进制用 BCD 码来表示。

(2)可靠性编码

由于任何代码在传输的过程中都可能会发生错误,为了减少这种错误,因此人们又引进了可靠性编码。下面就将介绍一种常用的可靠性编码——格雷码(Gray)。

格雷码也叫做循环码,其最基本的特征是任何相邻的两组代码中,仅有一位数码不同,因而又叫单位间距码。格雷码常用于模拟量的转换中,当模拟量发生微小变化时,格雷码只改变一位,这就比其他码需要改变两位或多位的情况要可靠,减少了出错的概率。格雷码的编码方式也有多种,典型的编码方式如表 1.3 所示。

表 1.3　格 雷 码

b_3 b_2 b_1 b_0	g_3 g_2 g_1 g_0
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0
1 1 0 0	1 0 1 0
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 1
1 1 1 1	1 0 0 0

1.2.4　数字电路中符号数的表示

在数字系统中,数是用二进制表示的,数的符号也是用二进制数表示的。把一个数连同其符号在机器中的表示加以数值化,这样的数称为机器数。机器数可以用不同的码制来表示,常用的有原码、反码、补码等等。由于在数字系统中大多采用补码表示,故而在此只介绍补码表示法。

(1)数的补码表示

1)正数的补码表示法:一般用二进制数的最高有效位为“0”来表示符号为正,二进制数的其他位则用来表示数的绝对值,即“符号位 + 数的绝对值”。若机器字长为 8 位(也就是说在机器中采用 8 位二进制来表示一个符号数),则 b_7 表示数的符号,$b_6b_5b_4b_3b_2b_1b_0$ 这 7 位二进制表示符号数的绝对值。

例 1.12 设机器字长为 8，用补码表示符号数 +63。

解： 因机器字长为 8，故用 8 位二进制 $b_7b_6b_5b_4b_3b_2b_1b_0$ 来表示补码。又因符号为正，故而：$b_7=0$。

再用 $b_6b_5b_4b_3b_2b_1b_0$ 来表示绝对值：

$$(63)_{10}=(011\ 1111)_2$$

所以

$$[+63]_{补}=0011\ 1111$$

又如，$[+1]_{补}=0000\ 0001$，$[+0]_{补}=0000\ 0000$。

2）负数的补码表示法：设有一负数为 X，则 X 的补码用 $2^n-|X|$ 来表示，其中 n 为机器的字长，$|X|$ 是 X 的绝对值。例如，当 $n=8$ 时，$[-63]_{补}=2^8-|-63|=256-63=1100\ 0001$，$[-1]_{补}=2^8-1=255=1111\ 1111$，可见最高有效位为“1”表示该数为负数，应该注意的是，$[-0]_{补}=2^8-0=1\ 0000\ 0000$，有 9 位二进制，超过了机器字长，这种情况称为溢出，故应将溢出部分去掉，得 $[-0]_{补}=0000\ 0000$。所以符号只对“0”这个数没有影响，均表示为0000 0000。对于 1000 0000 这个数，在补码中被定义成 -128。因此，8 位字长的补码能表示数的范围是 -128 ~ +127（$-2^7\sim2^7-1$）。同理，16 位字长的补码能表示数的范围是 $-2^{15}\sim2^{15}-1$，其他可以依此类推。

用上述办法求负数的补码较为麻烦，这里有一种简单的方法：先写出与该数对应的正数的补码，然后按位求反（即 0 变 1，1 变 0），最后在末位加 1 就可以得到该负数的补码了。

例 1.13 若字长 $n=16$ 位，求 $N=-31$ 的补码。

+31 可表示为	0000	0000	0001	1111
按位求反后为	1111	1111	1110	0000
末位加 1 后为	1111	1111	1110	0001
用十六进制表示为	F	F	E	1

即 $[-31]_{补}=(FFE1)_{16}$

例 1.14 若字长 $n=8$ 位，求 $N=-63$ 的补码。

+63 可表示为	0011	1111
按位求反后为	1100	0000
末位加 1 后为	1100	0001
用十六进制表示为	C	1

即 $[-63]_{补}=(C1)_{16}$

由上面的例子可以知道，用补码表示数时还需注意符号的扩展问题。所谓符号的扩展是指一个数从位数较少扩展到位数较多（例如从 8 位扩展到 16 位、32 位或 16 位扩展到 32 位等）。对于用补码表示的数，正数符号扩展则在高位补“0”，负数则在高位补“1”，直到满足所需的位数为止。例如，将例 1.2.13 的字长由 8 位改为 16 位，由于是负数，则需在高位补“1”，故结果为 $(1111\ 1111\ 1100\ 0001)_2=(FFC1)_{16}$。由于补码能表示的数的范围受机器字长的限制，故一般不能用位数较少的补码来替换位数较多的补码。

（2）补码的加法和减法

补码的加法规则是：$[X+Y]_{补}=[X]_{补}+[Y]_{补}$

补码的减法规则是：$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$

对于这两个规则下面将举例说明，从这些例子中大家将会认识到，由于使用补码来表示

数，使得运算十分方便，它不必判断数的符号，只要符号位能参加运算就能得到正确的结果。

设字长为8位，用下面例子说明补码的加法运算。

例1.15　十进制　　补码

$$\begin{array}{r} 25 \\ +30 \\ \hline 55 \end{array} \qquad \begin{array}{r} 0001\ \ 1001 \\ +0001\ \ 1110 \\ \hline 0011\ \ 0111 \end{array}$$

例1.16

$$\begin{array}{r} 32 \\ +(-25) \\ \hline 7 \end{array} \qquad \begin{array}{r} 0010\ \ 0000 \\ +1110\ \ 0111 \\ \hline 10\,000\ \ 0111 \end{array}$$

由于字长是8，只能取8位二进制，故所得的结果中最高位的“1”必须去掉才是最终结果$(0000\ 0111)_2$。

例1.17

$$\begin{array}{r} 25 \\ +(-32) \\ \hline -7 \end{array} \qquad \begin{array}{r} 0001\ \ 1001 \\ +1110\ \ 0000 \\ \hline 1111\ \ 1001 \end{array}$$

例1.18

$$\begin{array}{r} -32 \\ +(-25) \\ \hline -57 \end{array} \qquad \begin{array}{r} 1110\ \ 0000 \\ +1110\ \ 0111 \\ \hline 1\ 1100\ \ 0111 \end{array}$$

去掉溢出部分后，结果为$(1100\ 0111)_2$。

例1.19

$$\begin{array}{r} 32 \\ -(-25) \\ \hline 57 \end{array} \rightarrow \begin{array}{r} 32 \\ +25 \\ \hline 57 \end{array} \qquad \begin{array}{r} 0010\ \ 0000 \\ +0001\ \ 1001 \\ \hline 0011\ \ 1001 \end{array}$$

可见$[X-(-Y)]_{补}=[X]_{补}+[Y]_{补}=[X+Y]_{补}$，这和代数运算相似。

1.3　逻辑代数基础

逻辑代数是19世纪中叶英国数学家乔治·布尔(George·Boole)创立的一门研究客观事物逻辑关系的代数，故逻辑代数又称为布尔代数。随着数字技术的发展，逻辑代数已成为研究数字逻辑电路必不可少的数学工具。