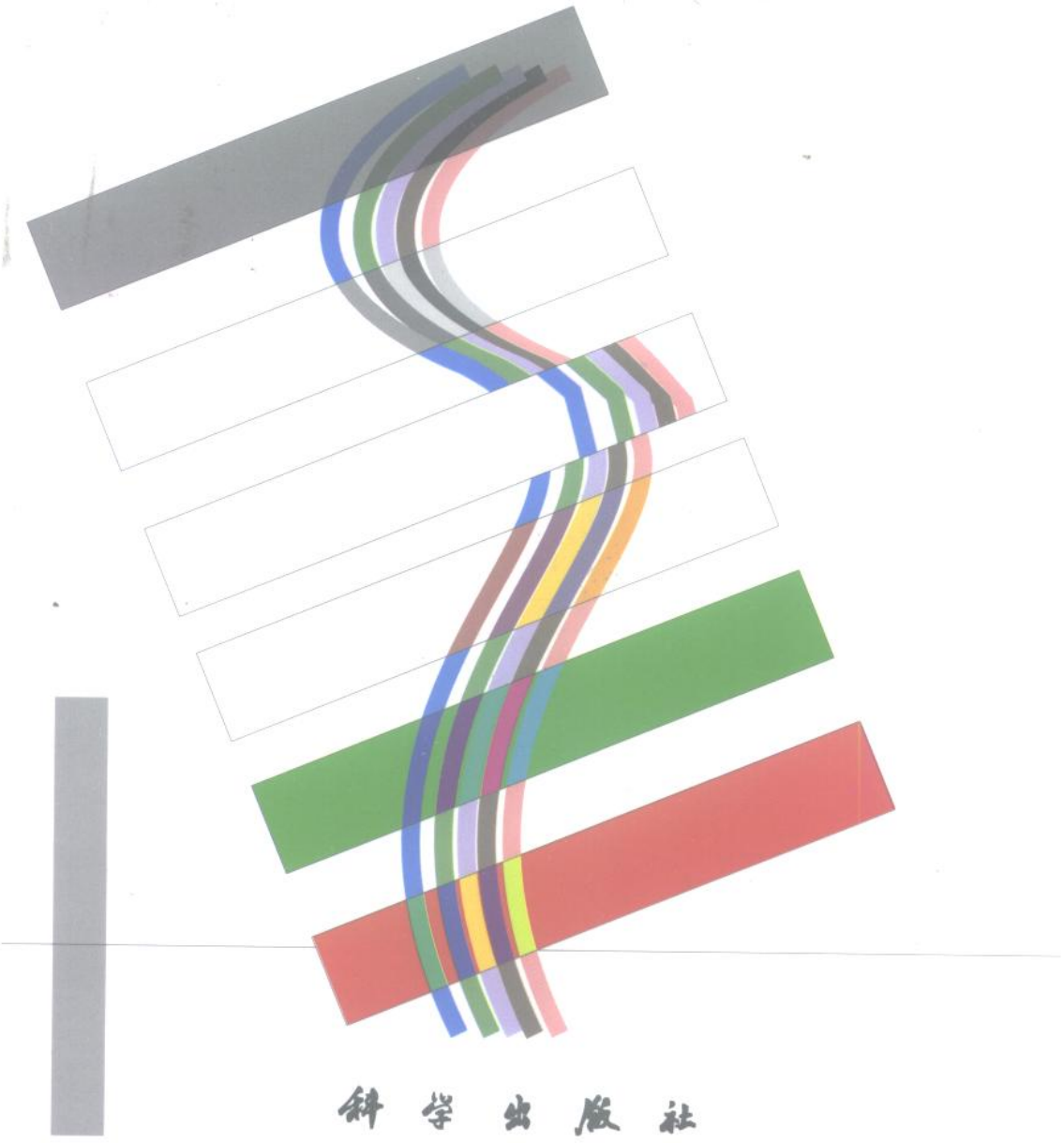
高等院校选用教材系列

数字逻辑电路

刘必虎　沈建国　编著



科学出版社

高等院校选用教材系列

数字逻辑电路

刘必虎　沈建国 编著

科学出版社
1999

内 容 简 介

本书介绍数字逻辑电路设计的原理、方法和应用，着重论述中大规模集成电路及EPROM存储器、GAL、iSPLSI等可编程逻辑器件的原理、编程和应用实例．书中众多在实际应用中取得成功的例子可启发读者把学到的基础知识用于解决实际问题．

全书共十二章，由逻辑代数、组合电路、时序电路的分析和设计、集成脉冲电路、可编程逻辑器件、A/D与D/A集成块及其应用等六部分组成，每章均附有适量习题．

本书可作为高校计算机、无线电、自控等专业的教材．

图书在版编目(CIP)数据

数字逻辑电路/刘必虎，沈建国编著．-北京：科学出版社，1999．8
(高等院校选用教材系列)
ISBN 7-03-007336-3

Ⅰ．数… Ⅱ．①刘… ②沈… Ⅲ．①逻辑电路-高等学校-教材 ②逻辑集成电路-高等学校-教材 Ⅳ．TN79

中国版本图书馆CIP数据核字（1999）第04632号

科学出版社 出版
北京东黄城根北街16号
邮政编码：100717
新蕾印刷厂印刷
新华书店北京发行所发行 各地新华书店经售
*
1999年8月第 一 版 开本：787×1092 1/16
1999年8月第一次印刷 印张：26 1/2
印数：1—4 000 字数：614 000
定价：39.80元

(如有印装质量问题，我社负责调换〈新欣〉)

前　言

数字系统和数字电路设备已广泛应用于各个社会领域，它使各种仪器设备获得众多的功能，满足智能化的要求，因此电子技术领域的工程师和技术员必须掌握数字系统的基础知识.

数字电子技术在早期的分立元件和 60 年代后的小规模集成电路(SSI)时，逻辑设计的化简理论发挥了重要作用.70 年代以后，中规模集成电路(MSI)和大规模集成电路(LSI)相继研制成功，它提高了电路的性能，降低了成本，但其设计方法却大部分是探索性和经验性的.

目前，大规模和超大规模的可编程逻辑器件得到了越来越广泛的实际应用，它们的设计采用的是计算机辅助设计技术，使电子系统的研制时间大大缩短，特别是系统可编程逻辑器件，可以在不改变硬件设置的情况下，在现场对系统进行组态，并可实现电子系统的遥控升级.

集成电路技术和计算机辅助设计的迅猛发展改变了电子系统的传统设计方法，它使电子设计自动化(EDA)和电子系统设计自动化(ESDA)成为现代电子系统设计和制造中的主要技术手段，使电子系统的设计从传统的单纯硬件设计方法变为计算机软硬件协同设计的方法，由此可设计制造出实现各种功能的专用集成电路(ASIC).

EDA 和 ESDA 技术是现代电子工程师进行电子系统和电子工程设计所必须掌握的技术.

为了适应电子系统设计技术的发展、培养面向 21 世纪、参与国内外市场竞争的电子科技人才，本书对原数字逻辑电路方面的内容做了较大增删，除了讲述必要的数字逻辑设计原理的基础知识以外，对小规模电路的内容做了精简，加强了中大规模组件方面的内容，特别是对在系统可编程逻辑器件的编程及使用做了较详细的介绍，使读者能掌握具体的使用技术.

此外，本书还讨论了数模和模数转换技术，最后介绍了脉冲电路的基本知识.

本书是编著者在科研与教学基础上，为适应数字技术的迅猛发展、新器件不断出现的新形势而编写的.本书的原稿已在华东师范大学电子系作为教材多次使用，成书过程中得到翁默颖教授的指导及教研组中多位老师的帮助，本专业的林昇等同学也做了许多工作，编者在此一并致谢.

编著者水平有限，敬请读者对本书的缺点及不足之处批评指正.

编　著　者

1998 年 7 月

目　录

第一章　数制和编码

数字逻辑电路主要研究数字信号的产生、存储、变换及运算等，其分析及设计方法是电子工程技术人员所必备的专业基础知识. 二进制和逻辑代数是目前数字逻辑电路中进行算术运算及逻辑运算的主要数学工具.

本章以二进制为重点，讨论了各种不同的进位计数制及其相互之间的联系与转换方法，进而讨论了二进制负数的表示法以及用二进码来表示十进制数的编码方法.

§1.1　进位计数制及其相互转换

一、数字信号及数字电路

电信号可分为模拟信号和数字信号两大类. 模拟信号是指在时间上和幅值上都是连续变化的电压或电流信号. 处理这种模拟信号的电路统称为模拟电路.

所谓数字信号是指在时间上和幅值上都是离散的(不连续的)电压或电流信号. 如图 1-1 所示. 一方面，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 $t_i(i=0,1,2,\cdots)$；另一方面，它们的幅值大小及增减变化都采取数字形式，譬如，图中 t_2 时幅值

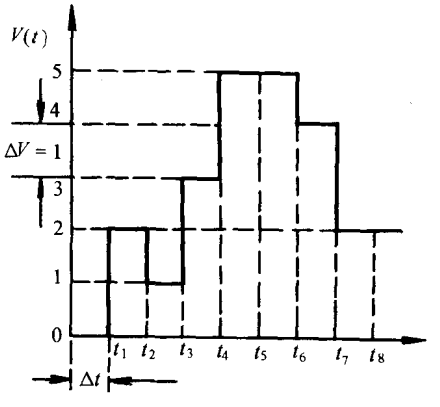


图 1-1　数字信号例一

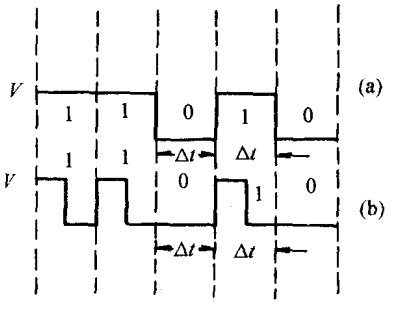


图 1 2　数字信号例二

由 2 跳变为 1，t_4 时幅值由 3 跳变为 5 等等. 实际上，目前在数字电路中最常采用的是只有 0 和 1 两种数值组成的数字信号，如图 1-2 所示. 这种两值数字信号又可分为电位型和脉冲型两种. 图 1-2(a)是电位型数字信号，用高电位表示数字 1，低电位表示数字 0；每个 1、0 信号所占的时间间隔 Δt 都相等，并称之为一拍. 一般一个 1 或一个 0 称为 1 比特(Bit)或 1 位. 几个连续的高(低)电位，就是几拍(位)长的 1(0)信号. 图 1-2(a)所示就是 11010 五位数字信号. 图 1-2(b)是脉冲型数字信号，它是以每拍内有无脉冲来表示的信号，在一拍时间内有脉冲信号表示数字 1，无脉冲信号表示数字 0，图 1-2(b)所示也是 11010 五位数字信号.

处理数字信号的电路称为数字电路. 它能产生、存贮、变换、传送数字信号以及具有对数字信号进行算术运算、逻辑运算、计数、显示等功能. 由于数字电路的输入变量与输出变量都是数字信号，而且相互之间都有一定的逻辑关系，这些逻辑关系常用一系列逻辑符号来表示，所以数字电路又称做数字逻辑电路，或逻辑电路.

二、十进计数制(十进制)

数制是计数进位制的简称. 只有 0 和 1 两种数值组成的数字信号称做二进制信号. 目前在数字电路中处理的就是二进制数字信号，有时也用八进制和十六进制来表示，但人们熟悉的是十进制数字.

在十进制数中，每一位用 0、1、2、…、9 十个不同的数符来表示. 在数制中，数符的个数称为基数，十个数符即基数为十. 计数时，超过 9 的数要用多位数来表示，其中低位数和高一位数之间的关系是“逢十进一”或“借一当十”，故称为十进制.

任意一个正的十进制数 N 都可以写成按位权展开的表示式

$$\begin{aligned}(N)_{10}=\sum_{i=-m}^{n-1}a_i\times 10^i &= a_{n-1}\times 10^{n-1}+a_{n-2}\times 10^{n-2}+\cdots+a_1\times 10^1\\ &\quad +a_0\times 10^0+a_{-1}\times 10^{-1}+a_{-2}\times 10^{-2}+\cdots+a_{-m}\times 10^{-m}\\ &=(a_{n-1}a_{n-2}\cdots a_1a_0a_{-1}a_{-2}\cdots a_{-m})_{10}\end{aligned}\tag{1-1}$$

式中 a_i 是第 i 位的系数，它可能是 0～9 十个数符中的任何一个；10^i 是第 i 位的位值，称为“权”；n 为整数的位数，m 为小数的位数. a_{n-1}是最高位，a_{-m}是最低位.

如 $38.65=3\times10^1+8\times10^0+6\times10^{-1}+5\times10^{-2}$. 各位权值分别为 $10^1=10$，$10^0=1$，$10^{-1}=0.1$，$10^{-2}=0.01$；把 10^0 称为个位，10^1 称为十位，10^{-1}及 10^{-2}分别称为十分位及百分位等等.

若以基数 R 代替上式中的 10，就可以得到 R 进制(任意一个进制)的普遍表示式

$$\begin{aligned}(N)_R=\sum_{i=-m}^{n-1}A_iR^i &= A_{n-1}\times R^{n-1}+A_{n-2}\times R^{n-2}+\cdots+A_1\times R^1+A_0\times R^0+A_{-1}\times R^{-1}\\ &\quad +A_{-2}\times R^{-2}+\cdots+A_{-m}\times R^{-m}\\ &=(A_{n-1}A_{n-2}\cdots A_1A_0A_{-1}A_{-2}\cdots A_{-m})_R\end{aligned}\tag{1-2}$$

式中 A_i 是系数，它可能是 0、1、2、…、$R-1$ 等 R 个数符中的任何一个.

三、二进制及 2^K 进位计数制

1. 二进制

在二进制中，基数 $R=2$，每一位仅有 0 和 1 两个可能的数符，低位和高一位之间的关系是逢二进一，第 i 位的权值为 2^i，一个任意的二进制数$(N)_2$可写成

$$\begin{aligned}(N)_2=\sum_{i=-m}^{n-1}b_i2^i&=(b_{n-1}b_{n-2}\cdots b_1b_0\,b_{-1}b_{-2}\cdots b_{-m})_2\\ &=b_{n-1}\times 2^{n-1}+b_{n-2}\times 2^{n-2}+\cdots+b_1\times 2^1+b_0\\ &\quad \times 2^0+b_{-1}\times 2^{-1}+b_{-2}\times 2^{-2}+\cdots+b_{-m}\times 2^{-m}\end{aligned}\tag{1-3}$$

这里 b_i 的取值只有 0 和 1 两种可能. 一个 n 位的二进制数可表示 2^n 个数，其中最大

的数为 2^n-1；例如，两位二进制数可表示 $2^2=4$ 个数，即 00、01、10、11 分别对应于十进制数 0、1、2、3，其最大的数为 $2^2-1=3$.

由于二进制可以很方便地用具有两个稳定状态的电子器件如晶体管的饱和及截止来实现（其中一个状态表示“0”，另一个状态表示“1”），而且二进制运算简单，所以在数字电路及计算机中得到广泛的应用.

2. 2^K 进位计数制

用二进制来表示较大的数值时，位数很多，例如 $(1024)_{10}=(10000000000)_2$，读写时容易出错. 为了简捷地表示一个很长的二进制数，常常采用基数为 2^K 的进位计数制，K 是正整数. 其中常用的有八（2^3）进计数制及十六（2^4）进计数制.

在八进制中，基数 $R=8$，即有 0、1、2、3、4、5、6、7 等八个数符，低位和高一位之间的关系是逢八进一. 据(1-2)式可知，一个八进制数，如 $(13.24)_8$，可表示为 $(13.24)_8=1\times8^1+3\times8^0+2\times8^{-1}+4\times8^{-2}$.

在十六进制中，基数 $R=16$，即有 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 共十六个数符，低位和高一位之间的关系是逢十六进一. 任意一个十六进制数同样可以用展开式表示，如 $(3AB.98)_{16}=(3AB.98)_H=3\times16^2+A\times16^1+B\times16^0+9\times16^{-1}+8\times16^{-2}$.

式中 H 表示十六进制数.

现将十进制、二进制、八进制及十六进制的对照表列于表 1-1.

表 1-1　几种常见的数制对照表

十进制数	二进制数	八进制数	十六进制数
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	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*
16	10000	20	10

四、不同数制的相互转换与运算

由于二进制数不便于读写，而人们又习惯于使用十进制数，因此二进制数往往只在数字电路、计算机内部使用. 这样就需要在机器的输入端及输出端进行不同数制的转换.

1. 二进制转换为十进制

二进制数转换为十进制数时，可由(1-3)式把二进制数按权展开，然后按十进制的运算规则求和.

例 1-1 将二进制数$(101101.01)_2$换算为十进制数.

解 $(101101.01)_2=1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$
$+0\times2^{-1}+1\times2^{-2}$
$=32+8+4+1+0.25=(45.25)_{10}$

为便于计算，表 1-2 列出了不同 n 时二进制的权值 2^n 以及 2^{-n}.

2. 十进制转换为二进制

(1)十进制整数换算为二进制整数

十进制整数转换为二进制数常采用“除 2 取余”的方法，现举例说明如下.

例 1-2 把十进制整数$(13)_{10}$转换为二进制数$(N)_2$.

解 由于转换后的二进制数是整数，故$(N)_2$的展开式应为

表 1-2 二进制的权

n	2^n	2^{-n}
0	1	1.0
1	2	0.5
2	4	0.25
3	8	0.125
4	16	0.0625
5	32	0.03125
6	64	0.015625
7	128	0.0078125
8	256	0.00390625
9	512	0.001953125
10	1024	0.0009765625
11	2048	0.00048828125
12	4096	0.000244140625
13	8192	0.0001220703125
14	16384	0.00006103515625
15	32768	0.000030517578125
16	65536	0.0000152587890625

$$(N)_2=b_{n-1}\times2^{n-1}+b_{n-2}\times2^{n-2}+\cdots+b_1\times2^1+b_0\times2^0$$

据题意得

$$(13)_{10}=b_{n-1}\times2^{n-1}+b_{n-2}\times2^{n-2}+\cdots+b_1\times2^1+b_0\times2^0$$
$$=2(b_{n-1}\times2^{n-2}+b_{n-2}\times2^{n-3}+\cdots+b_1)+b_0$$

等式两边同时除以 2，便得

$$\frac{13}{2}=6+\frac{1}{2}=b_{n-1}\times 2^{n-2}+b_{n-2}\times 2^{n-3}+\cdots+b_1+\frac{b_0}{2}$$

由于等式两边整数与小数必须对应相等，所以由上式可得 $\frac{b_0}{2}=\frac{1}{2}$，即 $b_0=1$，b_0 正好是 13 除以 2 的余数.

$$6=b_{n-1}\times 2^{n-2}+b_{n-2}\times 2^{n-3}+\cdots+b_2\times 2^1+b_1$$

同样，将上式两边除以 2，可得

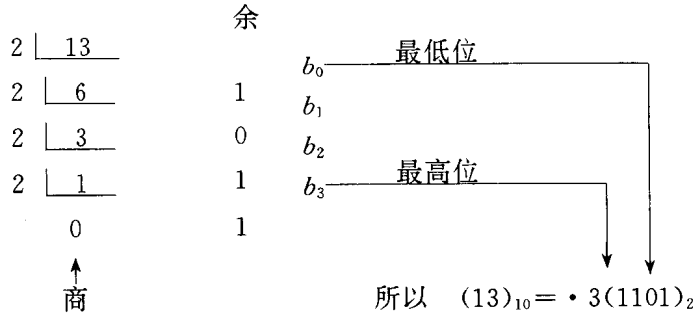
$$3=b_{n-1}\times 2^{n-3}+b_{n-2}\times 2^{n-4}+\cdots+b_2+\frac{b_1}{2}$$

可知 $b_1=0$，即 6 除以 2 的余数为 0，

$$3=b_{n-1}\times 2^{n-3}+b_{n-2}\times 2^{n-4}+\cdots+b_3\times 2^1+b_2$$

上式两边再除以 2 可得 $b_2=1$，$1=b_{n-1}\times 2^{n-4}+b_{n-2}\times 2^{n-5}+\cdots+b_4\times 2^1+b_3$，等式两边再除以 2 便得 $\frac{1}{2}=b_{n-1}\times 2^{n-5}+b_{n-2}\times 2^{n-6}+\cdots+b_4+\frac{b_3}{2}$，由于整数部分为 0，所以 b_{n-1}、b_{n-2}、$\cdots b_4$ 皆为 0，而 $b_3=1$；最后可得 $(N)_2=(b_3b_2b_1b_0)_2=(1101)_2=(13)_{10}$.

现将整个过程与结果演算如下：



总之，采用“除 2 取余”法时，只要连续除以 2，直除到商为 0 为止；然后取其余，取余时由下而上排列，即最下面的余数为最高位，最上面的余数为最低位.

(2)十进制小数换算为二进制小数

十进制小数换算为二进制小数常采用“乘 2 取整”的方法，现举例说明如下.

例 1-3 把十进制小数 $(0.625)_{10}$ 转换为二进制小数 $(N)_2$.

解 由于转换后的二进制是小数，故 $(N)_2$ 的展开式应为

$$(N)_2=0.b_{-1}b_{-2}\cdots b_{-m}=b_{-1}\times 2^{-1}+b_{-2}\times 2^{-2}+b_{-3}\times 2^{-3}+\cdots+b_{-m}\times 2^{-m}$$

据题意得

$$(0.625)_{10}=b_{-1}\times 2^{-1}+b_{-2}\times 2^{-2}+b_{-3}\times 2^{-3}+\cdots+b_{-m}\times 2^{-m}$$

等式两边都乘以 2，得

$$2\times 0.625=1.25=1+0.25=b_{-1}+b_{-2}\times 2^{-1}+b_{-3}\times 2^{-2}+\cdots+b_{-m}\times 2^{-m+1}$$

由于等式两边的整数与小数必须对应相等，所以

$$b_{-1}=1,0.25=b_{-2}\times 2^{-1}+b_{-3}\times 2^{-2}+\cdots+b_{-m}\times 2^{-m+1}$$

继续将上式两边都乘以 2，得

$$2\times 0.25=0.5=0+0.5=b_{-2}+b_{-3}\times 2^{-1}+b_{-4}\times 2^{-2}+\cdots+b_{-m}\times 2^{-m+2}$$

可见

$$b_{-2}=0,0.5=b_{-3}\times 2^{-1}+b_{-4}\times 2^{-2}+\cdots+b_{-m}\times 2^{-m+2}$$

将此等式两边再乘以 2，得

$$2\times 0.5=1+0.0=b_{-3}+b_{-4}\times 2^{-1}+\cdots+b_{-m}\times 2^{-m+3}$$

可见 $b_{-3}=1$，而 b_{-4}、…、b_{-m}都为 0，所以$(0.625)_{10}=(0.101)_2$.

如果按上述方法进行下去，十进制小数部分始终不为 0，则只要换算到所需要的精度为止.

例 1-4 将十进制数$(0.56)_{10}$ 换算为二进制数，精确到小数点后八位.

解

$2\times 0.56=1.12$	$b_{-1}=1$
$2\times 0.12=0.24$	$b_{-2}=0$
$2\times 0.24=0.48$	$b_{-3}=0$
$2\times 0.48=0.96$	$b_{-4}=0$
$2\times 0.96=1.92$	$b_{-5}=1$
$2\times 0.92=1.84$	$b_{-6}=1$
$2\times 0.84=1.68$	$b_{-7}=1$
$2\times 0.68=1.36$	$b_{-8}=1$

所以$(0.56)_{10}=(0.10001111+e)_2$，$e$ 为$<2^{-8}$的剩余误差.

如果一个数既有整数部分，又有小数部分，则可将整数部分与小数部分分别进行转换，然后相加即得其所需结果. 例如

$$(13.625)_{10}=(13+0.625)_{10}=(1101+0.101)_2=(1101.101)_2$$

根据上面的分析，不难得到，一个 R 进制表示的数 A 要转换成十进制数时，可以按(1-2)式将其按权展开后相加即可. 而一个十进制数要转换成 R 进制数时，只要把除 2、乘 2 改成除 R、乘 R，然后取其余数或取其整数就可以了.

对于两种非十进制的不同数制间的转换，一般都是先把其中一种数制的数转换成十进制数，然后再把该十进制数转换成相应的另一种数制的数. 但二进制数与 2^K 进制数之间可以很方便地直接进行转换，而不必经过十进制数. 当二进制数转换为 2^K 进制数时，只要从二进制整数的最低位开始，每 K 个二进制数分成一组，然后写出每组对应的 2^K 进制数即可. 反之，将 2^K 进制数转换为二进制数时，只要将每位 2^K 进制数值转换为相应的 K 位二进制数即可.

例 1-5 将二进制数$(11001101)_2$ 转换为 2^3(八)进制数$(N)_8$.

解 $(11001101)_2=(011'001'101)_2=(315)_8$，所以

$(11001101)_2=(315)_8$

例 1-6 将八(2^3)进制数$(167)_8$ 转换为二进制$(N)_2$.

解 $(167)_8=(001'110'111)_2$，所以$(167)_8=(1110111)_2$

例 1-7 将二进制数$(1011100110)_2$ 转换为十六(2^4)进制数$(N)_{16}$

解 $(1011100110)_2=(0010'1110'0110)_2=(2E6)_{16}$，所以

$(1011100110)_2=(2E6)_{16}$

例 1-8 将十六(2^4)进制数$(4DF)_{16}$转换为二进制数$(N)_2$.

解 $(4DF)_{16}=(0100'1101'1111)_2$，所以

$(4DF)_{16}=(10011011111)_2$

3. 二进制数的算术运算

二进制数进行加、减、乘、除四则算术运算时，其运算规则与十进制数相同，只是由于其进位基数为 2，使得和数表与积数表比后者简单得多，如下所示.

加法表三个：0+0=0，0+1=1+0=1，1+1=10.

乘积表三个：0×0=0，0×1=1×0=0，1×1=1.

下面举例说明二进制正整数的运算规则：

1)
$$\begin{array}{r} 1011 \\ +\quad 11 \\ \hline 1110 \end{array}$$

2)
$$\begin{array}{r} 1001 \\ -\quad 110 \\ \hline 0011 \end{array}$$

3)
$$\begin{array}{r} 1011 \\ \times\quad 101 \\ \hline 1011 \\ 0000 \\ 1011 \\ \hline 110111 \end{array}$$

4)
$$\begin{array}{r|l} & 110 \leftarrow \text{商} \\ \hline 111 & 101101 \\ & 111 \\ \hline & 1000 \\ & 111 \\ \hline & 11 \leftarrow \text{余数} \end{array}$$

§1.2 原码、反码和补码

一、真值与机器数

前面讨论的数都没有考虑符号，可看做是正数，但在算术运算中总会有负数，那末在数字系统中如何来表示带符号的二进制数(即数的正负)呢？

不带符号的数是数的绝对值，在绝对值左面加上表示正负的符号就成了带符号数. 直接用“+”号和“−”号来表示正、负的二进制数，叫做符号数的真值. 如+1001 表示绝对值为 1001 的正数；−1001 表示绝对值为 1001 的负数. 正数前面的“+”号可以省略. 这种符号数不能直接用于数字系统中.

“+”号和“−”号也是表示了两种不同的状态，同样，它也可以用一位二进制数来表示. 习惯上，以 0 表示符号“+”(正数)，以 1 表示符号“−”(负数)，在此符号位之后便为数值位. 这种把数的符号也数值化(0 或 1)的带符号数叫做机器数，它可以在数字设备中使用. 为了表示清楚，在符号位和数值位之间加上逗号“,”，而当逗号“,”和小数点在同一位置时，则省去该逗号“,”. 当然，在数字设备中是没有这逗号“,”的.

在数字系统中表示机器数的方法很多，常用的有原码、反码及补码三种. 在这三种表示形式中，符号部分的规定是相同的，所不同的只是其数值部分的表示形式.

二、原码

在原码表示的正数和负数中，第一位为符号位，0 表示正数，1 表示负数，其余为数值位.

一个具有 n 位整数、m 位分数的二进制数 N，即

$N=\pm\sum_{i=-m}^{n-1} b_i \times 2^i$，其原码的一般表示式为

$$[N]_{原}=\begin{cases}N & 0\leqslant N<2^n\\ 2^n-N & -2^n<N\leqslant 0\end{cases}\tag{1-4}$$

这里$[N]_{原}$为原码表示的机器数，N 为机器数的真值.

如果 N 是一个真值小于 1 的小数，即 $n=0$，则其原码表示式为

$$[N]_{原}=\begin{cases}N & 0\leqslant N<1\\ 1-N & -1<N\leqslant 0\end{cases}\tag{1-5}$$

从原码的一般表示式可知，当 N 为正数时，$[N]_{原}$与 N 的区别只是在数的左边增加一位用 0 表示的符号位，而 0 对该数的数值并无影响，所以$[N]_{原}$就是 N 本身；当 N 为负数时，$[N]_{原}$和 N 的区别是增加一位用 1 表示的符号位. 在原码中，有两种不同形式的零，即正零和负零，如表 1.3 所示. 正零和负零的值是相等的.

原码表示法易懂易读，与真值的变换也很方便. 但是用原码来做加、减运算就比较复杂. 例如，若两个数相加，需首先比较两个数的符号，若两数的符号相同，就将两个数的数值相加，最后给结果加上与两数相同的符号；若两数的符号不同，就需先比较两数绝对值的大小，然后使绝对值大的数减去绝对值小的数，差值的符号与绝对值大的数的符号一致. 要在数字系统中实现这一过程，所需电路就较复杂，运算速度也较慢. 为了简化运算，在数字系统中常把机器数表示成反码及补码形式.

三、反码

用反码表示时，左边第一位也为符号位，0 代表正数，1 代表负数. 反码数值位与它的符号有关. 若是正数，则反码数值与原码数值相同；若是负数，则反码数值是将原码数值按位变反（即原码的某位为 0，反码的对应位就为 1，或者原码的某位为 1，反码的对应位就为 0）.

例如：$N_1=+11001$，$N_2=-10101$，则其原码和反码分别为

$[N_1]_{原}=0,11001$，　　$[N_2]_{原}=1,10101$

$[N_1]_{反}=0,11001$，　　$[N_2]_{反}=1,01010$

一个具有 n 位整数、m 位分数的二进制数 N，即

$N=\pm\sum_{i=-m}^{n-1}b_i\times 2^i$，其反码的一般表示式为

$$[N]_{反}=\begin{cases}N & 0\leqslant N<2^n\\ (2^{n+1}-2^{-m})+N & -2^n<N\leqslant 0\end{cases}\tag{1-6}$$

如果 N 是一个真值小于 1 的 m 位小数，即 $n=0$，则其反码的表示式为

$$[N]_{反}=\begin{cases}N & 0\leqslant N<1\\ (2-2^{-m})+N & -1<N\leqslant 0\end{cases}\tag{1-7}$$

由反码的一般表示式可知，正数 N 的反码$[N]_{反}$与原码$[N]_{原}$相同；对于负数 N，其反码$[N]_{反}$的符号位为 1，而数值部分是将原码$[N]_{原}$的数值按位变反. 如表 1.3 所示，零的反码有正零和负零两种表示形式.

采用反码进行加、减运算时，可以把减去一个正数看成是加上一个负数，然后把加数及被加数都用反码表示，从而把减法运算转换成加法运算. 在运算中，符号位也被看成一位数，它与数值位一样按加法进位规则进行运算. 运算结果仍为反码，即两数反码的和等于两数和

的反码. 反码运算时，符号位产生的进位要加到和的最低位上去，即所谓循环进位.

例 1-9 试用反码相加求 $N=0.0111-0.1010$.

解 $0.0111-0.1010=(0.0111)+(-0.1010)$

$$[0.0111]_{反}=0.0111,[-0.1010]_{反}=1.0101,$$

$$[0.0111]_{反}+[-0.1010]_{反}=1.1100=[0.0111-0.1010]_{反}=[N]_{反}$$

上式表示两数反码的和等于该两数和的反码. 所以真值 $N=-0.0011$.

例 1-10 试用反码相加求 $N=1101-0111$.

解 $1101-0111=(1101)+(-0111)$

$$\begin{array}{r} [1101]_{反}=0,1101 \\ +[-0111]_{反}=1,1000 \\ \hline 10,0101 \\ +\ \text{└}\!\!\rightarrow 1 \\ \hline 0,0110=[1101-0111]_{反}=[N]_{反} \end{array}$$

所以真值 $N=+0110=+110$

由于循环进位需要进行二次算术相加，这就给电路带来麻烦，并延长了计算时间. 而采用补码进行运算可避免循环进位的二次计算.

四、补码

用补码表示时，左边第一位也为符号位，0 代表正数，1 代表负数. 补码数值位部分的形成也与其符号有关. 对于正数，补码数值与原码数值的对应位相同；若是负数，则补码数值是将原码数值按位取反后在最低位加 1. 例如：

$$N_1=+11001,\qquad N_2=-10101$$

则其补码为

$$[N_1]_{补}=0,11001,\qquad [N_2]_{补}=1,01011$$

一个具有 n 位整数及 m 位分数的二进制数 N，即

$N=\pm\sum_{i=-m}^{n-1}b_i\times 2^i$，其补码的一般表示式为

$$[N]_{补}=\begin{cases} N & 0\leqslant N<2^n \\ 2^{n+1}+N & -2^n\leqslant N<0 \end{cases} \tag{1-8}$$

如果 N 是一个整数部分为 0 即 $n=0$ 的 m 位小数则补码为

$$[N]_{补}=\begin{cases} N & 0\leqslant N<1 \\ 2+N & -1\leqslant N<0 \end{cases} \tag{1-9}$$

由补码的一般表示式可以得到：正数 N 的补码$[N]_{补}$、反码$[N]_{反}$和原码$[N]_{原}$是相同的，都等于正数 N. 对于负数，补码$[N]_{补}$的符号位为 1，其数值部分为反码数值在末位加 1. 在补码表示法中，零只有一种表示形式，如表 1-3 所示. 表 1-3 列举了带符号二进制整数的原码、反码和补码之间的对应关系，这里设二进制数共四位，其中一位符号位，三位数值位. 表中补码的负数多一个，即$[(-8)_{10}]_{补}=1000$，这是由补码的定义得到的. 表中 $n=3$，其最大负值 $N=-2^3=-(8)_{10}=(-1000)_2$，所以$[N]_{补}=2^{3+1}+N$，即$[(-8)_{10}]_{补}=10000-1000=1000$.

表 1-3　三种不同的二进制正、负数表示法

十进制数	二进制数(共四位)		
	原码	反码	补码
−8			1000
−7	1111	1000	1001
−6	1110	1001	1010
−5	1101	1010	1011
−4	1100	1011	1100
−3	1011	1100	1101
−2	1010	1101	1110
−1	1001	1110	1111
−0	1000	1111	0000
+0	0000	0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0110

例 1-11　分别用二进制原码、反码及补码表示$(-12.625)_{10}$，要求连符号位共八位.

解　$N=(-12.625)_{10}=(-1100.101)_2$

$[N]_{原}=1,1100.101$

$[N]_{反}=1,0011.010$

$[N]_{补}=1,0011.011$

五、补码运算

利用补码，可以把减法运算转换成加法运算. 运算时，符号位也被看成一位数，它与数值部分一样按加法进位规则进行运算，而且可以不考虑符号位产生的进位，这个进位自动丢失，这有利于简化电路的设计.

1. 设备容量和溢出

在数字设备中，能长期存放二进制数的地方叫做内部存贮器，简称内存，能暂时存放二进制数的地方叫做通用寄存器，简称寄存器. 存储器和寄存器都是存放数据的设备，这些设备存放数据的能力(能容纳二进制数多少的能力)称为设备容量，并把存储器一个单元所包含的二进制数的位数 n 叫做数据的字长. 在字长较长的设备中为表示方便，常把 8 位二进制数定义为一个字节，而把 4 位二进制数组成的单元叫做半字节.

字长为 n 位的二进制数字设备能表示 2^n 个数，并称该设备的模 $M=2^n$，模即为数字设备的容量. 例如四位二进制计数器能计 16 个数(0000～1111)，则其模 $M=2^4=16$. 超出此范围的进位数(如 $2^4=16$，即 10000 中的 1)将自动丢失. 因此计 18 个数时，在该计数器中只表示为 0010，其模 $M=2^4$ 被自动丢失. 又如钟表的模为 12，当报时为 $A(18)$ 点时，指针却指在 $B(6)$ 点上，其模 $M=12$ 也自动进位丢失了. 此时称 $B(6)$ 和 $A(18)$ 对模 $M(12)$ 是同余的(用 12 去同除 6 和 18 这两个数，所得的余数相同)，并称 $A(18)$、$B(6)$ 在以 $M(12)$ 为模时是相等的，即

$$A=B \quad (\text{mod } M)$$

对于四位二进制计数器而言，有

$$2=18 \quad (\text{mod}16)$$

显然，$A=iM+A \quad (\text{mod}M)$ i 为 0,1,2,…等的整数

2. 补码运算.

利用补码可将减法运算转换成加法运算，以达到简化电路、提高运算速度的要求.

由补码定义及模和同余的概念，不难证明，两个数的补码之和等于和的补码，即

$$[N_1+N_2]_{补}=[N_1]_{补}+[N_2]_{补} \tag{1-10}$$

式中 $-2^n\leqslant N_1<2^n$，$-2^n\leqslant N_2<2^n$，且 $-2^n\leqslant(N_1+N_2)<2^n$.

求得 $[N_1+N_2]_{补}$ 后即可得到 $[N_1+N_2]_{原}$，从而求得真值 (N_1+N_2). 若 $[N_1+N_2]_{补}$ 的

符号位是0(正数),则$[N_1+N_2]_{补}=N_1+N_2$;如果$[N_1+N_2]_{补}$的符号位是1(负数),则

$$[N_1+N_2]_{原}=[[N_1+N_2]_{补}]_{补} \tag{1-11}$$

证明如下:设

$$\begin{aligned}[N_1+N_2]_{补}&=1b_{n-1}b_{n-2}\cdots b_1b_0\cdot b_{-1}\cdots b_{-m}\\&=2^n+b_{n-1}b_{n-2}\cdots b_1b_0\cdot b_{-1}\cdots b_{-m}\end{aligned}$$

由于

$$[N]_{原}=2^n-N,[N]_{补}=2^{n+1}+N,[N]_{原}+[N]_{补}=2^{n+1}+2^n$$

所以

$$\begin{aligned}[N_1+N_2]_{原}&=2^{n+1}+2^n-[N_1+N_2]_{补}\\&=2^{n+1}-b_{n-1}b_{n-2}\cdots b_1b_0\cdot b_{-1}\cdots b_{-m}\\&=2^n+2^n-2^{-m}-b_{n-1}b_{n-2}\cdots b_1b_0\cdot b_{-1}\cdots b_{-m}+2^{-m}\\&=2^n+\underbrace{11\cdots1}_{n个1}.\underbrace{11\cdots1}_{m个1}-b_{n-1}b_{n-2}\cdots b_1b_0\cdot b_{-1}\cdots b_{-m}+2^{-m}\\&=2^n+\bar{b}_{n-1}\bar{b}_{n-2}\cdots\bar{b}_1\bar{b}_0\cdot\bar{b}_{-1}\cdots\bar{b}_{-m}+2^{-m}\\&=1\,\bar{b}_{n-1}\bar{b}_{n-2}\cdots\bar{b}_1\,\bar{b}_0\cdot\bar{b}_{-1}\cdots\bar{b}_{-m}+2^{-m}\\&=[[N_1+N_2]_{补}]_{补}\end{aligned}$$

其中

$$\bar{b}_i=\begin{cases}0 & 当\ b_i=1\ 时\\1 & 当\ b_i=0\ 时\end{cases}$$

所以,已知补码求原码时,只要求该补码的补码,即该补码的符号位(1)不变,而将其数值位每位变反,然后在最低位加1.

例1-12 某设备字长为8位,试用补码法求$(125)_{10}-(13)_{10}=?$

解 $(125)_{10}-(13)_{10}=(01111101)_2-(00001101)_2$

$$=(01111101)_2+(-00001101)_2$$

$[(01111101)_2]_{补}=0,1111101$;$[(-00001101)_2]_{补}=1,1110011$

$[(01111101)_2]_{补}+[(-00001101)_2]_{补}$的运算结果为

$$\begin{array}{r}0,1111101\\+\quad 1,1110011\\\hline [1]0,1110000\end{array}$$

因为设备字长为8位,故这里的[1]自动丢失.注意,符号位与数值位一样也参加运算.现结果的符号位是0(正数),故$[N]_{补}=N$,即

$$(01110000)_2=(01111101)_2-(00001101)_2=(112)_{10}$$

例1-13 某设备字长8位,试用补码法求$(13)_{10}-(125)_{10}=?$

解 $[(13)_{10}]_{补}=[(00001101)_2]_{补}=0,0001101$

$[(-125)_{10}]_{补}=[(-01111101)_2]_{补}=1,0000011$

$[(13)_{10}]_{补}+[(-125)_{10}]_{补}$的运算如下:

$$\begin{array}{r}0,0001101\\+\quad 1,0000011\\\hline 1,0010000\end{array}$$