集成数字电路原理与应用

钱民康　编著

湖北科学技术出版社

一九八六年

内 容 提 要

本书系统地阐述了数字电路和逻辑设计的基础知识、基本分析方法和设计方法。对各种规模的数字集成组件，尤其是微型计算机中常用的数字集成组件，比较详细地介绍了功能分析、性能参数和应用举例。本书重点讨论了以中规模集成组件为中心的逻辑设计方法，还用一定篇幅介绍了脉冲集成组件。本书内容丰富、理论联系实际，反映了最新技术成果。

本书可作为高等工科院校无线电、自动控制等专业的试用教材，也可供工程技术人员和有关专业师生参考。

集成数字电路原理与应用

钱民康 编著

*

湖北科学技术出版社出版 新华书店湖北发行所发行

湖北省广济县印刷厂印刷

787×1092毫米 16开本 44印张 1,110,000字

1986年6月第1版 1986年6月第1次印刷

印数：1——5,000

统一书号：15304·111 定价：7.60元

前　言

随着科学技术的进步，数字集成电路在近十几年里有了较大发展，数字技术在雷达、通讯、自动控制、电视、无线电测量和数据处理等各个方面的应用越来越广泛，尤其是微型计算机在各个领域的推广应用，使每个工程技术人员都必须了解数字技术的基础知识。

本书从小、中、大规模集成电路并存的实际出发组织内容，使读者在学习数字技术基础理论的同时，了解集成电路产品的实际，并结合产品实际学习逻辑设计。

本书在编著过程中力求体现以下特点：

1.突出中、大规模集成电路。目前，数字集成电路中，中、大规模集成电路的比重越来越大。本书在内容结构上，压缩以小规模集成电路为中心的逻辑设计内容，加强中、大规模集成器件的功能分析，性能参数和使用方法介绍，并通过实例介绍以中、大规模集成器件为中心的逻辑设计方法。

2.增大CMOS型组件的比重。目前，CMOS技术有了重大突破，有些系列在工作速度上已与LSTTL器件相当，加上它特有的微功耗、电源电压使用范围宽、高干扰容限等特点，一些原是TTL组件的传统领域正逐渐被CMOS组件占领，它的重要性越来越大。因此，许多人，把CMOS集成电路称为“八十年代的集成电路”。为了适应这种情况，本书大幅度地增加了CMOS器件的比重。当然，TTL型组件目前仍是最重要的一种数字集成组件。

在国际上，TTL数字集成电路一般以美国德克萨斯仪器公司（TEXAS INSTRUMENTS）的产品SN54/74系列为公认的通用系列；CMOS数字集成电路一般以美国无线电公司（RCA）的产品CD4000B（电源3～18伏）系列和美国国家半导体公司（NS）的高速54/74HC、54/74HCT系列为公认的通用系列。本书在涉及集成组件时，一般都应用上述通用系列。

3.加强与微型计算机的联系。本书对微型计算机中常用的数字集成器件都用一定篇幅进行介绍。

4.突出数字技术的应用。本书在内容选取上加强了应用部分，对小规模器件重点进行了外特性研究，对中、大规模器件着重进行功能分析，并用较多篇幅讲述选取原则和运用集成器件的实际例子。为了便于应用，本书列有大量的器件资料。

本书第十二章“微型计算机在数字技术中的应用”，由俞国梁同志编写。

本书由王士俊副教授担任主审，薛凤鸣副教授审阅了第十章，王长胤副教授审阅了第十二章，张桃源同志校阅了本书的大部分章节，他们对初稿提出过很多宝贵意见。

空军雷达学院脉冲与数字电路教研室的部分同志参加了本书清样的校对，郭伟同志做了许多具体工作。

在这里，谨向上述单位和个人表示深切谢意。

由于数字技术近年来发展非常迅速，加上作者水平有限，因此本书一定会有不少缺点和错误，殷切希望读者给予批评指正。

钱　民　康

一九八五年二月

目　录

第一章　数制和编码

数字技术是电子技术的一个重要分支。它主要用于对数字信号进行存贮、变换和运算。数字技术中的运算，除了通常意义的算术运算之外，更主要的是进行逻辑运算。在数学技术中，对数字电路与数字系统的设计，主要的是进行逻辑设计。而逻辑设计的主要数学工具又是二进制和逻辑代数。

本章首先从常用的十进制数开始分析，进而说明各种不同的进位计数制之间的相互联系和相互转换。在此基础上再讨论二进制负数的三种表示法，即原码、反码和补码，以及用二进制码来表示十进制数和常用字符的编码方法。

第一节　进位计数制

在我们日常生活中，通常使用的数制是十进制，它采用0、1、…、9等十个数码，并按“逢十进一”的规律进行计数。因此，十进制是基数为十的计数制。

数码在数中处于不同的数位，其代表的意义是不同的。例如，7419.5这个数，它可以写成　$7419.5=7\times10^3+4\times10^2+1\times10^1+9\times10^0+5\times10^{-1}$

在该数中，最左的位为千位，第二位是百位，第三位是十位，第四位是个位，小数点右边第一位是十分位。通常把上述各数位所代表的数值称为权。上面7419.5中各位的权，自左至右分别为10^3、10^2、10^1、10^0、10^{-1}。因此，任何一个十进制数N_{10}均可写成按权展开的一般形式（也称为数的多项式表示法）：

$$\begin{aligned}N_{10}&=a_{n-1}a_{n-2}\cdots a_1\ a_0.a_{-1}\cdots a_{-m}\\&=a_{n-1}\times10^{n-1}+\cdots\cdots+a_1\times10^1+a_0\times10^0+a_{-1}\times10^{-1}+\cdots+a_{-m}\times10^{-m}\\&=\sum_{i=-m}^{n-1}a_i\times10^i\end{aligned}$$

式中，N右下角的角注表示数N的基数为10，a表示0～9数码中的任意一个，a_i表示第i位的数码，10^i表示第i位的权，n表示整数的位数（n只取正整数），m表示小数位数（m只取正整数）。

除了十进制以外，在生产和生活中还会碰到非十进制的计数制。例如，时钟以60进位：60秒为1分，60分为1小时。又如，12支铅笔为1打，7天为1周，它们分别是12进位和7进位。对于任意的r进制，数Nr也可写成按权展开的一般形式：

$$\begin{aligned}Nr&=a_{n-1}a_{n-2}\cdots a_2a_1a_{-1}\cdots a_{-m}\\&=a_{n-1}\times r^{n-1}+a_{n-2}\times r^{n-2}+\cdots+a_1\times r^1+a_0\times r^0+a_{-1}\times r^{-1}+\cdots+a_{-m}\times r^{-m}\\&=\sum_{i=-m}^{n-1}a_ir^i\end{aligned}$$

式中，a_i为第i位的数码，它可取0，1，…r－1中的任意正整数。

在数字电路中大量采用的是二进制。八进制和十六进制也常被涉及，下面对这三种进位计数制分别作一介绍。

在二进制中，数码只有两个：0和1，而且按“逢二进一”规律进行运算。因此，它的基数为2。任意二进制数N_2可按权展开为：

$$\begin{aligned} N_2 &= b_{n-1}b_{n-2}\cdots b_1 b_0 . b_{-1}\cdots b_{-m} \\ &= b_{n-1}\times 2^{n-1}+b_{n-2}\times 2^{n-2}+\cdots+b_1\times 2^1+b_0\times 2^0+b_{-1}\times 2^{-1}+\cdots+b_{-m}\times 2^{-m} \\ &= \sum_{i=-m}^{n-1} b_i 2^i \end{aligned}$$

这里，b_i是第i位的数码，它只能取0或1；2^i为第i位的权值。例如，二进制数 110101.1 可写成如下形式

$$\begin{aligned} N_2 &= 110101.1 \\ &= 1\times 2^5+1\times 2^4+0\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0+1\times 2^{-1} \end{aligned}$$

二进制的加法和乘法规则是：

$0+0=0$　　$0\times 0=0$

$0+1=1+0=1$　　$0\times 1=1\times 0=0$

$1+1=10$　　$1\times 1=1$

运用二进制的加法和乘法规则，就可以进行二进制的算术运算。下面举例说明：

例1　已知x＝10111，y＝1011，求：（1）　x＋y；（2）　x－y

解：（1）求（x＋y）　　（2）求（x－y）

```
    1 1 1 1      ←进位          1 0 0 0     ←借位
    1 0 1 1 1    ←x             1 0 1 1 1   ←x
+     1 0 1 1    ←y         −     1 0 1 1   ←y
  1 0 0 0 1 0    ←x+y             1 1 0 0   ←x−y
```

即　　x＋y＝100010，　　x－y＝1100。

例2　求1011×101的值。

解：

```
      1 0 1 1
  ×     1 0 1
  -------------
      1 0 1 1
    0 0 0 0
+ 1 0 1 1
  -------------
  1 1 0 1 1 1
```

即　　1011×101＝110111

在数字电路中为什么要采用二进制呢？因为在数字电路中究竟采用什么样的进位制，取决于该进位制在机器中是否容易实现，是否计算简便，节省器材。我们知道，十进制有十个数码，在机器中要制造出有十个稳态的装置是比较困难的，因而十进制是很难采用的。而二进制只有两个数码，它可以用开关的通断、触发器的两个稳态等简便的形式来表示。同时，二进制的算术运算也比较简便，只要分别记住$\frac{2\times(2+1)}{2}=3$个和及3个积，就可以进行运算。而十进制的算术运算，它们要分别记住$\frac{10\times(10+1)}{2}=55$个和及55个积（即九九表）。

因而，二进制比十进制运算要方便得多。由于二进制数具有这种运算规律简便的特点，就为简化运算电路和控制电路的结构创造了十分有利的条件。所以数字电路中采用二进制作为主要数制。当然，我们也应看到，二进制也有书写起来很长，读起来不易懂的缺点。为此，常用八进制和十六进制来书写，从而部分地弥补了这一缺点。

八进制有0、1、2、…、7等八个数码，分别代表0～7八个数值。当数值为8时，则向相邻的高位进一，即7＋1＝10。这里10所代表的数值为8，即进位规律为“逢八进一”。因此，八进制数675.4可表示为

$$(675.4)_8=6\times8^2+7\times8^1+5\times8^0+4\times8^{-1}$$

八进制数书写起来不象二进制那么长，近似于十进制，而且以后将要讲到，它与二进制之间的转换相当方便，因此它常用来书写计算机的指令和输入数据。

十六进制有十六个数码。其中，10～19的各个数码常用字母或其它符号来表示。其常用的表示方法为：

十进制数：　0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

十六进制数：0 1 2 3 4 5 6 7 8 9 A B C D E F

在十六进制中，当数值为16时，则向相邻的高位进位，即F＋1＝10。这里，10代表16，即“逢十六进一”。因此，十六进制数3AF.7可表示为：

$$(3AF.7)_{16}=3\times16^2+10\times16^1+15\times16^0+7\times16^{-1}$$

十六进制数也是微型计算机中常用来书写指令的工具，它也具有与二进制数互换简便的优点。

第二节　数制转换

使用微型计算机等数字设备时，经常需要确定二进制数的十进制值。此外，也需要将特定的十进制数转换成与它等效的二进制数。下面研究数制之间的转换方法及其原理。

非十进制数换算为十进制数的基本方法是按权相加法。这种方法是把这些非十进制数按权展开，再按十进制的运算规则相加。

例1　把$(101101.01)_2$换算为十进制数。

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

这种转换的基本原理是：二进制数中每一位数码所代表的大小决定于它的权，而二进制的权（2^i）又是用十进制表示的，所以按权相加后即为换算后的十进制数。

为了便于换算，表1—1中列出了与2^n和2^{-n}相应的十进制数，即二进制数中各位的权。

利用按权相加法同样可把八进制数和十六进制数换算成十进制数。

例2　把$(372.01)_8$换算成十进制数。

解.
$$\begin{aligned}(372.01)_8&=3\times8^2+7\times8^1+2\times8^0+1\times8^{-2}\\&=192+56+2+0.015625\\&=(250.015625)_{10}\end{aligned}$$

表1—2列出了8的乘方表。

表1—1

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.015625
128	7	0.0078125
256	8	0.00390625
512	9	0.001953125
1024	10	0.0009765625
2048	11	0.00048828125
4096	12	0.000244140625

表1—2

8^n	n	8^{-n}
1	0	1.0
8	1	0.125
64	2	0.015625
512	3	0.001953125
4096	4	0.000244140625
32768	5	0.000030517578125
262144	6	0.000003814697265625

例3 把$(E5D7.A3)_{16}$换算为十进制数。

解：
$$(E5D7.A3)_{16}=14\times16^3+5\times16^2+13\times16^1+7\times16^0+10\times16^{-1}+3\times16^{-2}$$
$$=14\times4096+5\times256+13\times16+7+10\times0.0625$$
$$+3\times0.00390625$$
$$=(58839.63671875)_{10}$$

表1—3列出了16的乘方表。

十进制数换算为非十进制数的常用方法是基数乘除法。这种方法，对于十进制数的整数部分和小数部分是分别进行换算的。

对于十进制整数N，它换算成r进制数时，利用除基数r取余法。它们的运算是按十进制的运算规律进行的。其一般方法为：

表1—3

16^n	n	16^{-n}
1	0	1.0
16	1	0.0625
256	2	0.00390625
4096	3	0.000244140625
65536	4	0.0000152587890625

	余　数
r | N	
r | N_1	A_0
r | N_2	A_1
⋮	⋮
r | N_{n-1}	A_{n-2}
0	A_{n-1}

上述商N_i和A_{i-1}分别为第i次除法所得到的商和余数，且$A_{i-1}<r$。因此，被除数N和商N_i又可写成：

$$N = r \cdot N_1 + A_0$$
$$N_1 = r \cdot N_2 + A_1$$
$$\vdots$$
$$N_{n-1} = r \cdot 0 + A_{n-1}$$

所以N又可写成：

$$N = r \cdot (rN_2 + A_1) + A_0$$
$$= r^2 \cdot N_2 + rA_1 + A_0$$
$$\vdots$$
$$= A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \cdots + A_1 r^1 + A_0$$
$$= (A_{n-1}A_{n-2}\cdots A_1 A_0)_r$$

例4　把$(653)_{10}$分别换算为：（1）二进制数；　（2）八进制数。

解：（1）换算为二进制数

2 | 653		
2 | 326	1	
2 | 163	0	
2 | 81	1	
2 | 40	1	
2 | 20	0	
2 | 10	0	
2 | 5	0	
2 | 2	1	
2 | 1	0	
0	1	

验算：

$$(1010001101)_2 = 1\times2^9 + 1\times2^7 + 1\times2^3 + 1\times2^2 + 1\times2^0 = 512 + 128 + 8 + 4 + 1 = (653)_{10}$$

因此，有$(653)_{10}=(1010001101)_2$。

由上述例子可看出，把十进制数换算成二进制数时，利用除2取余法，其余数逆序排列（即最后一次余数为二进制数的最高位），就得到所求的二进制数。

（2）换算成八进制数

```
8 | 653                验算：
  | 81        5        (1215)8 = 1×8³ + 2×8² + 1×8¹ + 5×8⁰
8 | 10        1                = 512 + 128 + 8 + 5
8 |  1        2                = (653)10
  |  0        1
```

即 $(653)_{10}=(1215)_8$

例5 把$(1626)_{10}$换算成十六进制数。

解：利用除16取余法，余数逆序排列：

```
16 | 1626               验算：
16 |  101      A        (65A)16 = 6×16² + 5×16¹ + 10×16⁰
16 |    6      5                = 1536 + 80 + 10
   |    0      6                = (1626)10
```

即 $(1626)_{10}=(65A)_{16}$

由于某进制的小数换算成其它进制的数时，仍将保持小数的性质。因此，将十进制小数N_F换算成r进制数时，N_F可表示成：

$$N_F=A_{-1}r^{-1}+A_{-2}r^{-2}+A_{-3}r^{-3}+\cdots$$

式中，系数A_{-1}、A_{-2}、A_{-3}…是整数。下面讨论确定系数A_{-1}、A_{-2}、A_{-3}…等的方法。

首先，用r乘N_F，得到

$$rN_F=A_{-1}+A_{-2}r^{-1}+A_{-3}r^{-2}+\cdots$$

因此，rN_F的整数部分是A_{-1}。我们从rN_F中减去A_{-1}，再用r乘，得到

$$r(rN_F-A_{-1})=A_{-2}+A_{-3}r^{-1}+A_{-4}r^{-2}+\cdots$$

由此确定了系数A_{-2}。这个过程可继续进行到获得所希望的系数为止。应当看到，上述过程有时是无穷尽的。

例6 把数$(0.61)_{10}$换算成二进制数。

解：利用上述乘基数2取整数法进行：

$2\times(0.61)=1.22$　　$A_{-1}=1$

$2\times(0.22)=0.44$　　$A_{-2}=0$

$2\times(0.44)=0.88$　　$A_{-3}=0$

$2\times(0.88)=1.76$　　$A_{-4}=1$

$2\times(0.76)=1.52$　　$A_{-5}=1$

$2\times(0.52)=1.04$ $\quad A_{-6}=1$

$2\times(0.04)=0.08$ $\quad A_{-7}=0$

⋮

即 $(0.61)_{10}=(0.1001110\cdots)_2$

验算 $(0.1001110\cdots)_2=1\times2^{-1}+1\times2^{-4}+1\times2^{-5}+1\times2^{-6}+\cdots$

$=0.5+0.0625+0.03125+0.015625+\cdots$

$=(0.609375\cdots)_{10}$

如果要把一个既有整数部分又有小数部分的十进制数换算成为非十进制的数，则可将整数部分和小数部分分别转换，再把结果合并。例如：

$(653.61)_{10}=(1010001101.1001110\cdots)_2$

上述换算是用十进制运算实现的。由于我们熟悉十进制的运算，因此这类换算是很方便的。

假如，要求把基数为r（r≠10）的数换算成基数为d的数，它有两种方法：1.运用r进制运算实现换算；2.先把r进制数换算成十进制数，然后再把该十进制数换算成d进制数。一般而言，我们对于非十进制的运算则不太熟悉，因而常不采用第一种方法，而采用第二种方法来实现换算。

例7 把$(12012)_3$换算成五进制数

解：我们分别用上述给出的两种方法进行换算。

1.运用三进制运算规则，直接实现转换时，采用除基数5取余法。

5 | 12012		
5 | 1001	$A_0=0$	
5 | 12	$A_1=3$	
5 | 1	$A_2=0$	
0	$A_3=1$	

即 $(12012)_3=(1030)_5$

2.先将$(12012)_3$换算成十进制数，再把该十进制数换算成五进制数。

$(12012)_3=1\times3^4+2\times3^3+1\times3^1+2\times3^0$

$=81+54+3+2$

$=(140)_{10}$

5 | 140		
5 | 28	$B_0=0$	
5 | 5	$B_1=3$	
5 | 1	$B_2=0$	
0	$B_3=1$	

即 $(12012)_3=(140)_{10}=(1030)_5$

由于八进制和十六进制与二进制间有着十分紧密的联系，因而它们在微型计算机等数字设备中具有特殊重要的作用，下面研究它们的互换。

因为$8=2^3$，所以每个八进制数与三位二进制数相对应，即

八进制数：	7	6	5	4	3	2	1
二进制数：	111	110	101	100	011	010	001

把二进制数转换成八进制数的步骤是：首先把二进制数从小数点开始划成三位一组，然后对每组指定相应的八进制数。

例如，$(111\ \ 001\ \ 010\ 011\ \ .010\ \ 110\ \ 011)_2=(7123.263)_8$

$$\underbrace{111}_{7}\ \underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ .\ \underbrace{010}_{2}\ \underbrace{110}_{6}\ \underbrace{011}_{3}$$

反之，把八进制数换算成二进制数时，对每位八进制数用三位二进制来代换。

下面以整数部分转换为例，来进一步说明八进制和十六进制与二进制互换的原理和方法。

设二进制整数N_2和与它等效的八进制整数M_8分别为

$$N_2=K_{n-1}K_{n-2}\cdots K_3K_2K_1K_0$$
$$=K_{n-1}2^{n-1}+\cdots+K_32^3+K_22^2+K_12^1+K_02^0$$
$$M_8=L_{m-1}L_{m-2}\cdots L_2L_1L_0$$
$$=L_{m-1}8^{m-1}+L_{m-2}8^{m-2}+\cdots+L_28^2+L_18^1+L_08_0$$

由于$N_2=M_8$，所以有：

$$K_{n-1}2^{n-1}+K_{n-2}2^{n-2}+\cdots+K_32^3+K_22^2+K_12^1+K_02^0$$
$$=L_{m-1}8^{m-1}+L_{m-2}8^{m-2}+\cdots+L_28^2+L_18^1+L_08^0$$

若等式两边同除以8，则

$$K_{n-1}2^{n-4}+K_{n-2}2^{n-5}+\cdots+K_32^0+\frac{K_22^2+K_12^1+K_02^0}{8}$$
$$=L_{m-1}8^{m-2}+L_{n-2}8^{m-3}+\cdots+L_28^1+L_18^0+\frac{L_08^0}{8}$$

由此得到

$$K_22^2+K_12^1+K_0=L_0$$

因此，将二进制数的低三位按权相加后，即可得到八进制的最低位L_0。

对上面等式的整数部分再除以8，则有

$$K_52^2+K_42^1+K_32_0=L_1$$

由此可见，对于二进制整数，从最低位算起，每三位按权相加，就可得到八进制的一位。如果二进制数的最高有效位不足三位时，可以在高位添零补足三位。

对于小数部分，列出等式后，依次用8去乘，就可求出八进制小数部分的各位数码：二进制数的前三位按权相加后为八进制数的第一位；四、五、六位按权相加后为八进制数的第二位，如此等等。若二进制小数的最低有效位不足三位，可在最低位添零补足三位。

对于八进制数换算成二进制数，同样也可用上述方法：将八进制数的每一位数码用三位二进制数码代替，就可实现转换。

例如，$(4\ \ 3\ \ 1.\ \ 7\ \ 6)_8=(100\ 011\ 001.111\ 11)_2$

$$\begin{array}{ccccccccccc}4 & & 3 & & 1 & . & 7 & & 6\\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow\\ 100 & & 011 & & 001 & & 111 & & 110\end{array}$$

根据$16=2^4$的关系，用四位二进制数代替十六进制数中的每一位，就可实现十六进制数到二进制数的转换；按照同样原理，可实现二进制数到十六进制数的转换。

例如，$(\underbrace{1101}_{D}\ \underbrace{1011}_{B}\ \underbrace{1000}_{8}\ \underbrace{0110}_{6}.\underbrace{1010}_{A}\ \underbrace{0011}_{3})_2=(DB86.A3)_{16}$。

表1—4列出了15以内十进制数与二进制、八进制和十六进制数的对应关系。

表1—4

十进制数	二进制数	八进制数	十六进制数
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

第三节　原码、补码和反码

以上讨论的二进制数的整数和小数，都没有涉及符号，可以认为它们都属于正数。那么，数的符号如何来表示呢？

普通数学中，通常我们在表示带符号数时，是在数值（绝对值）左面加上符号：正号用“+”（也可以省略），负号用“-”。例如，数值为0.1101的正、负数可分别表示为：+0.1101（或0.1101）和-0.1101。

在数字电路中，数是存放在由触发器组成的寄存器中的，二进制数码1和0是用触发器的两种不同状态来表示的。同样对于正号“+”和负号“-”也只能用触发器的两种不同的状态来表示。因此，符号也被数值化了。也就是说，带符号的数，其数值部分和符号部分统一用代码形式来表示。

在数字电路中，用代码形式表示的数常被称为机器数，而原来的数被称为机器数的真值。机器数中最常用的表示法有原码、补码和反码三种。

本节主要研究带符号小数的原码、补码和反码表示法以及它们一些主要性质。

一、原码

带符号数的原码（True form）表示是在数位左面加上符号位。对于正数，符号位记为0；对于负数，符号位记为1，数的数值部分，则保持不变。因此，原码表示又称为符号位加绝对值表示。

设x为二进制小数，那么按照上述定义它的原码$[x]_{原}$可用下式表示：

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

例如，$x=+0.1001$

则 $[x]_{原}=x=0.1001$

又如，$x=-0.1001$

则 $[x]_{原}=1-x=1-(-0.1001)=1+0.1001$

$=1.1001$

由上面讨论可看出原码的简单性质为：

1.若x为正数时，$[x]_{原}$就是x；若x是负数时，$[x]_{原}$是将x数小数点左面用“1”表示，小数点右面部分保持不变。

根据这个性质，x的原码可写成

$$[x]_{原}=x_0+|x|$$

式中，x_0为符号位，它满足

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

2．设$[x]_{原}=x_0.x_1\ x_2\cdots\cdots x_n$，则

$$x=\begin{cases}[x]_{原} & x_0=0\\ 1-[x]_{原} & x_0=1\end{cases}$$

例如，$[x]_{原}=0.1011$，其中$x_0=0$，故

$x=[x]_{原}=0.1011$

又如，$[x]_{原}=1.1011$，其中$x_0=1$，故

$x=1-[x]_{原}=1-1.1011$

$=-0.1011$

3．对于0，可认为它是（+0），也可认为是（-0），这样

$[+0]_{原}=0.0\cdots\cdots0$

$[-0]_{原}=1.0\cdots\cdots0$

所以在原码表示中，0有两种表示形式。

表1—5中列出了小数真值与原码之间的对应关系。

整数的原码表示与小数的原码表示有类似的地方，它也是在数位前面加一位符号位。设x为整数，则

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

因此，整数的原码表示又可写成

$$[x]_{原}=x_0+|x|$$

式中x_0为符号位，它的取值为

$$x_0=\begin{cases}0 & x\geqslant 0\\ 2^n & x\leqslant 0\end{cases}$$

表1—5

X	〔X〕原	〔X〕补	〔X〕反
+0.111	0.111	0.111	0.111
+0.110	0.110	0.110	0.110
+0.101	0.101	0.101	0.101
+0.100	0.100	0.100	0.100
+0.011	0.011	0.011	0.011
+0.010	0.010	0.010	0.010
+0.001	0.001	0.001	0.001
+0.000	0.000	0.000	0.000
−0.000	1.000	0.000	1.111
−0.001	1.001	1.111	1.110
−0.010	1.010	1.110	1.101
−0.011	1.011	1.101	1.100
−0.100	1.100	1.100	1.011
−0.101	1.101	1.011	1.010
−0.110	1.110	1.010	1.001
−0.111	1.111	1.001	1.000
−1.000		1.000	

例如，x = +10101则

$〔x〕_{原} = 0,10101$

上面代码中的逗号“，”是区分符号位和数值位的标志。

又如，x = −10101，则x＜0，有

$$〔x〕_{原} = 2^5 - x = 2^5 - (-10101)$$
$$= 100000 + 10101$$
$$= 1,10101$$

由上述讨论可见，数的原码表示法简单易懂，与真值的转换也比较简便。同时采用原码还可以方便地进行乘、除法的运算。如两数相乘时，将它们的绝对值相乘作为积的数值，再根据它们的符号确定积的符号。

例1 设x = 0.1101，y = −0.1011，求$〔x \cdot y〕_{原}$。

解：$〔x〕_{原} = 0.1101$，$〔y〕_{原} = 1.1011$。

求$〔x \cdot y〕_{原}$时，先求出$|x| \cdot |y|$的值，再决定积的符号位z_0。

$|x| = 0.1101$，$|y| = 0.1011$，因此$|x| \cdot |y| = 0.10001111$。

z_0决定于x_0、y_0的相对关系：当x_0、y_0为同号时，$z_0 = 0$；当x_0、y_0为异号时，$z_0 = 1$。本例中，$x_0 = 0$、$y_0 = 1$。因此，$z_0 = 1$。故

$$〔x \cdot y〕_{原} = 1.10001111$$

但是，原码的加、减运算比较复杂，如进行两数相加时，首先必须判断两数符号是否相同，