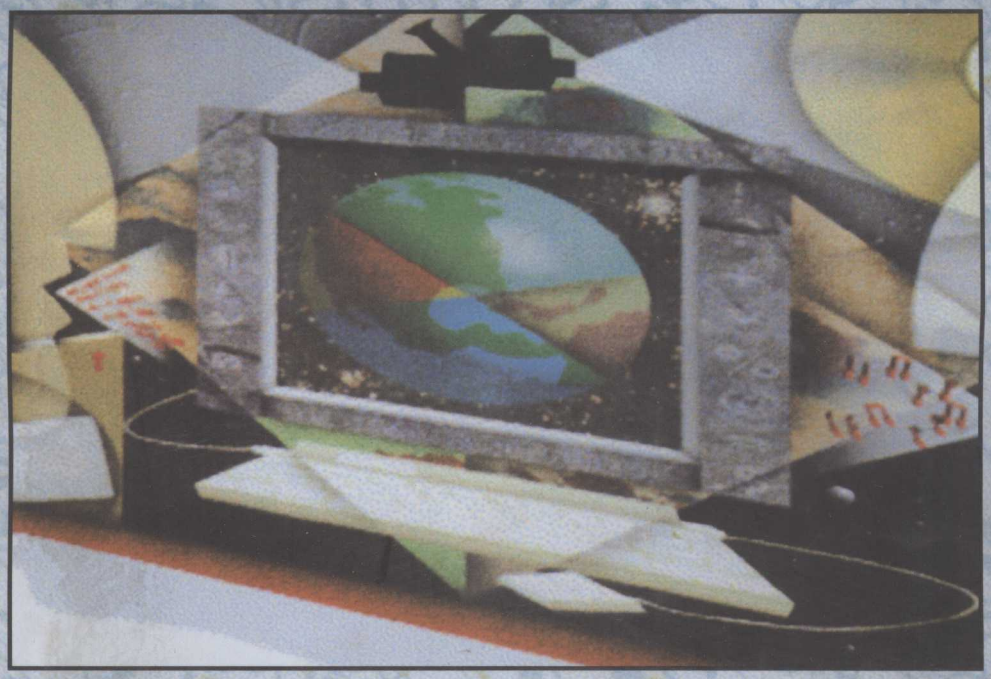
数字逻辑电路分析与设计



主编 鲍可进

东南大学出版社

数字逻辑电路分析与设计

主　编　鲍可进

参　编　赵不贿
　　赵念强

东南大学出版社

内 容 提 要

本书把传统的数字电路、数字逻辑两门技术基础课融合在一起，同时考虑数字器件的飞速发展背景，加强了可编程逻辑器件、在系统编程技术方面的内容，在扩大知识面、追踪电子器件发展方面，作了详尽的阐述。全书从数字电路的基础知识出发，介绍逻辑门电路、组合逻辑、时序逻辑、编程逻辑、在系统编程技术、数字系统设计等内容。每章末有小结并附有一定数量的习题与思考题。

本书从基础入门，逐步深化，紧扣数字系统发展的历程，使读者能从该课程而进入到数字系统的广阔领域中去。

该书可作为高等院校计算机应用、通信、计算机软件、自动控制等专业的教材，也可作为成人教育的教材及相关技术人员的参考书。

图书在版编目(CIP)数据

数字逻辑电路的分析与设计 / 鲍可进编 . —南京:东南大学出版社,1999.12

ISBN 7-81050-594-7

Ⅰ.数... Ⅱ.鲍...Ⅲ.① 逻辑电路-电路分析②逻辑电路-电路设计 Ⅳ.TN79

中国版本图书馆 CIP 数据核字(2000)第 10023 号

东南大学出版社出版发行
(南京四牌楼 2 号 邮编 210096)
出版人:宋增民
江苏省新华书店经销 丹阳兴华印刷厂印刷
开本:787 mm×1092 mm 1/16 印张:20 字数:487 千字
1999 年 12 月第 1 版 1999 年 12 月第 1 次印刷
印数:1-3000 册 定价:26.00 元

前　言

数字逻辑电路分析与设计是计算机科学与技术专业类必修的技术基础课。主要讲述数字系统的基础知识及应用数字电路来进行数字系统的分析与设计的基本理论和方法。

随着微电子技术的飞速发展，数字逻辑设计的物质基础电子元器件不断更新，设计方法也不断改进。电子器件从传统的小规模集成芯片到中、大规模集成芯片；从复杂可编程器件到高密度可编程器件。设计方法从经典的手工设计方法到计算机辅助设计方法以及电子设计自动化(EDA)。原来的教材及教学内容已跟不上微电子技术的发展，尤其计算机科学与技术专业类与微电子技术的发展更为密切，教材及教学内容的改革势在必行。

电子设计自动化(EDA)技术是20世纪90年代以后发展起来的，它打破了传统的由固定集成芯片组成数字系统的模式，对数字系统设计带来了革命性的变化。计算机科学与技术专业类的学生掌握这个新技术十分必要，所以该书中增加了在系统可编程技术的内容，以美国Lattice公司的在系统可编程芯片(ISP芯片)为模型来讲述EDA技术的设计思想。

本书共分8章，在内容安排上按循序渐进的原则，前面五章主要是讲述数字电路的基础知识及逻辑电路的分析与设计，这是学习数字逻辑电路课程必需的知识，也是学习可编程器件及EDA技术的基础。在这个基础上第6章讨论了中大规模集成电路的逻辑设计及可编程逻辑器件。第7章讲述了在系统可编程技术，重点放在讲述在系统可编程器件的结构及编程原理上，同时，将在系统可编程的开发软件的使用介绍放到与该书配套的实验指导书中讲解。第8章介绍了数字系统设计的概念及基本设计方法。为方便读者学习，每章附有小结与思考题。整个内容建议安排60～80学时讲授，并配以一定学时的实验课及课程设计，以加深对基本理论的理解和对新技术的掌握。

本书中数字逻辑电路的经典部分与新的EDA技术内容的结合得到了东南大学黄正谨教授的指点；该课程实验设备的设计得到了南京雨田公司的支持；Lattice公司的陈恒先生提供了部分资料；在该书的编写过程中，江苏理工大学教务处、杂志社、计算机系的领导及其同事们给了很大的支持和帮助；徐剑小姐为本书绘制了大部分的插图，在此一并表示感谢。

本书由鲍可进主编，其中第1章、第5章、第7章由鲍可进编写；第2章、第4章由赵不贿编写；第3章、第6章、第8章由赵念强编写，附录由鲍可进整理并对全书进行了统稿。

由于水平有限，加之时间较仓促，书中难免有一些缺点和错误，殷切希望广大读者批评指正。

编　者

1999年11月15日

目　录

1 数字逻辑基础

在数字系统中，信息的离散元素是以称为信号的物理量来表示的，电压和电流就是最常用的电信号。通常，数字系统的信号只有两个量“有”或“无”，称为二进制信号。具有导通和截止两种工作状态的电子器件能十分可靠地反映两个离散量，且在工程上较容易实现，加上人类的逻辑思维方式也倾向于二值，所以，数字系统常采用二进制信号。

本章主要围绕数字系统中的二进制信号，讨论数字系统中数的表示方法、数字系统中的编码、逻辑代数基础、逻辑函数化简等数字逻辑的基础知识。

1.1 数制与编码

1.1.1 进位计数制

所谓进位计数制，就是按进位方式实现计数的一种规则，简称进位制。在日常生活中我们就是按这种进位制计数的，如十进制、十二进制、六十进制等等。

对于任何一个数，我们可以用不同的进位制来表示。我们先从熟悉的十进制开始，分析各种进位制的特点和表示方法。

十进制有十个数字符号，即 0、1、2、3、4、5、6、7、8、9。将若干个这样的符号并列在一起可以表示一个进制数，每位不超过“9”，由低位向高位进位是“逢十进一”。这是十进制的特点。

这是要引两个术语：一个叫“基数”，它表示某种进位制所具有的数字符号的个数，如十进制的基数为“10”；另一个叫“位权”或“权”，它表示某种进位制的数中不同位置上数字的单位数值，如十进制数 135.79，最左位为百位（1 代表 100），权为 10^2；第二位为十位（3 代表 30），权为 10^1；第三位为个位（5 代表 5），权为 10^0；小数点左边第一位为十分位（7 代表 7/10），权为 10^{-1}；第二位为百分位（9 代表 9/100），权为 10^{-2}。

基数和权是进位制的两个要素，根据基数和权的概念，我们可以将任何一个数表示成多项式的形式。例如：

$$135.79 = 1\times10^2 + 3\times10^1 + 5\times10^0 + 7\times10^{-1} + 9\times10^{-2}$$

对于一个一般的十进制数 N，它可表示成

$$(N)_{10} = (d_{n-1}d_{n-2}\cdots d_1 d_0 \cdot d_{-1}d_{-2}\cdots d_{-m})_{10} \tag{1.1}$$

或

$$(N)_{10} = d_{n-1}(10)^{n-1} + d_{n-2}(10)^{n-2} + \cdots + d_1(10^1) + d_0(10)^0 + d_{-1}(10)^{-1} + d_{-2}(10)^{-2} + \cdots + d_{-m}(10)^{-m} = \sum_{i=-m}^{n-1} d_i(10)^i \tag{1.2}$$

式中，n 表示整数部分的位数；m 表示小数部分的位数；10 表示基数，$(10)^i$ 为第 i 位的权；d_i 表示各个数字符号，在十进制中有

$$d_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

通常，我们把式(1.1)称为并列表示法，把式(1.2)称为多项式表示法或按权展开式。

一般地，对于任意进制数可表示为：

$$\begin{aligned}(N)_R &= (r_{n-1}r_{n-2}\cdots r_1 r_0 \cdot r_{-1}r_{-2}\cdots r_{-m})_R \\ &= r_{n-1}R^{n-1} + r_{n-2}R^{n-2} + \cdots + r_1R^1 + r_0R^0 + r_{-2}R^{-2} + \cdots + r_{-m}R^{-m} \\ &= \sum_{i=-m}^{n-1} r_i R^i\end{aligned}$$

式中，n 表示整数的位数，m 表示小数的位数；R 为基数，在 10 进制中 R 应写成“10”；r_i 是 R 进制中各个数字符号，即有

$$r_i \in \{0, 1, 2, \cdots, R-1\}$$

在数字系统中，常用二制数来表示数和进行运算。这时 R 写成“2”，$r_i \in \{0, 1\}$。

二进制算术运算十分简单，规则如下：

加法规则　$0+0=0$，　$0+1=1+0=1$，　$1+1=10$

乘法规则　$0\times0=0$，　$0\times1=1\times0=0$，　$1\times1=1$

下面举几个二进制数四则运算的例子，从中领会它的运算规则。

【例 1-1】 两个二进制数相加，采用“逢二进一”的法则。

解

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

【例 1-2】 两个二进制数相减，采用“借一当二”的法则。

解

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

【例 1-3】 两个二进制数相乘，其方法与十进制乘法运算相似，但采用二进制运算规则。

解

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

【例 1-4】 两个二进制数相除，其方法与十进制除法运算相似，但采用二进制运算规则。

解

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

虽然数字系统广泛采用二进制，但当二进制数的位数很多时，书写和阅读很不方便，容易出错。为此，人们通常采用二进制的缩写形式——八进制和十六进制。

八进制的基数 $R=8$，每位可取 8 个不同的数字符号（即 0,1,2,3,4,5,6,7），其进位规则是“逢八进一”。

由于一位八进制的 8 个数字符号正好相应于三位二进制数的八种不同组合，所以八进制与二进制之间有简单的对应关系：

八进制	0	1	2	3	4	5	6	7
二进制	000	001	010	011	100	101	110	111

这样，八进制与二进制之间数的转换就极为方便。

例如　将二进制数 11010.1101 转换为八进制数。

$$\underbrace{011}_{3}\ \underbrace{010}_{2}\ .\ \underbrace{110}_{6}\ \underbrace{100}_{4}$$

所以　$(11010.1101)_2=(32.64)_8$

由二进制转换成八进制的方法是：以小数为界，将二进制数的整数部分从低位开始，小数部分从高位开始，每三位分成一组、头尾不足三位的补 0；然后将每组的三位二进制转换为一位八进制数。由八进制数转换成二进制数同样很方便。

例如　将八进制数 357.6 转换为二进制数。

```
3   5   7   .   6
↓   ↓   ↓       ↓
011 101 111 .   110
```

所以　$(357.6)_8=(11101111.11)_2$

十六进制的基数 $R=16$，每位可取 16 个不同的数字符号（即 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F），其进位规则是“逢十六进一”。

同理，由于一位十六进制的 16 个数字符号正好相应于四位二进制数的十六种不同的组合，所以，十六进制与二进制之间有简单的对应关系：

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

这样，十六进制与二进制之间数的转换也很方便。

例如　将二进制数 1010110110.110111 转换为十六进制数。

$$\underbrace{0010}_{2}\ \underbrace{1011}_{B}\ \underbrace{0110}_{6}\ .\ \underbrace{1101}_{D}\ \underbrace{1100}_{C}$$

所以　　$(1010110110.110111)_2 = (2B6.DC)_{16}$

例如　将十六进制数 5D.6E 转换为二进制数。

$$\begin{array}{cccccc} 5 & D & . & 6 & E \\ \downarrow & \downarrow & & \downarrow & \downarrow \\ 0101 & 1101 & . & 0110 & 1110 \end{array}$$

所以　$(5D.6E)_{16} = (1011101.0110111)_2$

由此可见，采用八进制和十六进制要比用二进制书写简短，易读易记，而且转换也方便。因此，计算机工作者普遍采用八进制或十六进制来书写和表达。

1.1.2　数制转换

在计算机和其它数字系统中普遍采用二进制，采用二进制的数字系统只能处理二进制数或用二进制编码形式表示的其它进位制数。由于人们习惯于使用十进制数，所以在用计算机进行信息处理时，首先必须把十进制数转换成二进制数才能被计算机所接受，然后进行运算，运算结果又必须从二进制转换成人们习惯的十进制数。

1）二进制数和十进制数的转换

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

$$\begin{aligned}(11010.101)_2 &= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 \\ &\quad + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} \\ &= 16 + 8 + 2 + 0.5 + 0.125 \\ &= (26.626)_{10}\end{aligned}$$

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。将一个十进制数写成

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

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

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

（1）整数转换

十进制数的整数部分采用“除 2 取余”法进行转换，即把十进位制整数除以 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，再取余数 1 或 0 作为二进制数的次低位，依次类推，继续上述过程，直至商为 0，所得余数为最高位。

例如，要将十进制整数 58 转换为二进制整数，就要把它成如下形式：

$$\begin{aligned}(58)_{10} &= (a_{n-1}a_{n-2}\cdots a_1a_0)_2 \\ &= a_{n-1}\times2^{n-1} + a_{n-2}\times2^{n-2} + \cdots + a_1\times2^1 + a_0\times2^0 \\ &= 2(a_{n-1}\times2^{n-2} + a_{n-2}\times2^{n-3} + \cdots + a_1) + a_0\end{aligned}$$

只要求出等式中的各个系数 $a_{n-1}, a_{n-2}, \cdots, a_1, a_0$，便得到二进制数。

将上式两边除以 2，得

$$(29)_{10} = a_{n-1} \times 2^{n-2} + a_{n-2} \times 2^{n-3} + \cdots + a_1 + \frac{a_0}{2}$$

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，则取 a_0 为 0。因而得

$$(29)_{10} = 2(a_{n-1} \times 2^{n-3} + a_{n-2} \times 2^{n-4} + \cdots + a_2) + a_1$$

将等式两边再除以 2，得

$$\left(14 + \frac{1}{2}\right)_{10} = a_{n-1} \times 2^{n-3} + a_{n-2} \times 2^{n-4} + \cdots + a_2 + \frac{a_1}{2}$$

比较等式两边，等式左边余数为 1，则取 a_1 为 1。

依次类推，可得系数 $a_2, a_3, \cdots, a_{n-2}, a_{n-1}$。

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数。

2 | 58
2 | 29　　余数 0(a_0)最低位
2 | 14　　余数 1(a_1)
2 | 7　　余数 0(a_2)
2 | 3　　余数 1(a_3)
2 | 1　　余数 1(a_4)
0　　余数 1(a_5)最高位

因此，$(58)_{10} = (111010)_2$。

(2) 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：

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

只要求出各系数 $a_{-1}, a_{-2}, \cdots, a_{-m}$，便得到二进制小数。

将上式两边乘 2，得

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

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，a_{-1} 等于左边的整数，则 a_{-1} 为 1。

等式右边括号内的数仍为小数，因而

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

再将等式两边乘 2，得

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

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

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数。

$$
\begin{array}{lll}
0.625 & & \\
\times) \quad 2 & & \\
\hline
\boxed{1}.250 & & \text{整数 } 1(a_{-1}) \text{最高小数位} \\
\times) \quad 2 & & \\
\hline
\boxed{0}.500 & & \text{整数 } 0(a_{-2}) \\
\times) \quad 2 & & \\
\hline
\boxed{1}.000 & & \text{整数 } 1(a_{-3}) \text{最低小数位}
\end{array}
$$

因此，$(0.625)_{10}=(0.101)_2$。

必须指出：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。

【例 1-5】 将十进制数 0.18 转换成二进制数，精确到小数点后 4 位。

解

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

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 再乘一次 2，小数点后第 5 位四舍五入后得

$$(0.18)_{10}\approx(0.0011)_2$$

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。

【例 1-6】 将$(58.625)_{10}$转换成二进制数。

解 $(58.625)_{10} = (58)_{10} + (0.625)_{10}$

$= (111010)_2 + (0.101)_2$

$= (111010.101)_2$

2) 任意两种进制之间的转换

前面介绍的方法并不局限于十进制与二进制之间的转换，可用于任意 α、β 进制之间的转换。因为人们对十进制运算十分熟悉，所以 α 进制→β 进制，一种比较方便的方法是利用十进制作桥梁，先把 α 进制转换为十进制数，这用按权展开就行了，然后再将十进制数转换为 β 进制数，这时可分为整数(除 β 取余)和小数(乘 β 除整)两部分进行。其示意图如图 1-1 所示

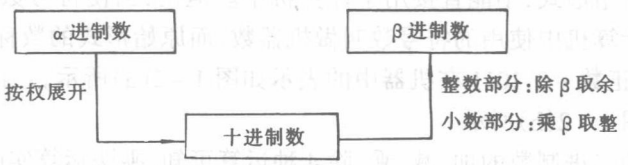


图 1-1 任意进制转换示意图

【例 1-7】 $(121.02)_4 = (\ ?\ )_3$

解 先用按权展开的方法转换成十进制数：

$$(121.02)_4 = (1\times4^2 + 2\times4^1 + 2\times4^0 + 2\times4^{-2})_{10}$$

$$= (16 + 8 + 1 + 0.125)_{10} = (25.125)_{10}$$

再将十进制数的整数部分、小数部分分别进行了转换：

```
3 | 25        余数
3 |  8          1  低位
3 |  2          2
    0           2  高位

 0 . 1 2 5
×)       3      整数
-----------
 0 . 3 7 5      0  高位
×)       3
-----------
 1 . 1 2 5      1
×)       3
-----------
 0 . 3 7 5      0  低位
-----------
    ⋮
```

所以 $(121.02)_4 = (221.010\cdots)_3$

1.1.3 带符号数的代码表示

1）真值与机器数

前面讨论的数都没有考虑符号，一般认为是正数，但在算术运算中总会出现负数。不带符号的数是数的绝对值，在绝对值前加上表示正负的符号就成了带符号数。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。数的符号是一个具有正、负两种值的离散信息，它可以用一位二进制数来表示。习惯上以0表示正数，而以1表示负数。对于一个 n 位二进制数，如果数的第一位为符号位，则乘下的 $n-1$ 位就表示数的数值部分。一般直接用正号“+”和负号“-”来表示符号的二进制数，叫做符号数的真值。数的真值形式是一种原始形式，不能直接用于计算机中。但是，当使符号数值化以后，就可在计算机中使用它。计算机中使用的符号数叫做机器数，而原始形式的数称为真值。

例如，二进制正数 +0.1011 在机器中的表示如图 1-2(a)所示，二进制负数 -0.1011 在机器中的表示如图 1-2(b)所示。

由前面介绍的二进制数的加、减、乘、除4种运算可知，乘法运算实际上是做移位加法运算，而除法运算则是做移位减法运算。这就是说，在机器中需要做加、减两种运算。但做减法运算时，必须先比较两个数绝对值的大小，将绝对值大的数减绝对值小的数，最后在相减结果的前面加上正确的符号。虽然逻辑电路可以实现减法运算，但所需的电路复杂，运算时间较长。为了能使减法运算变成加法运算，人们提出了3种机器数的表示形式，即原码、反码和补码。

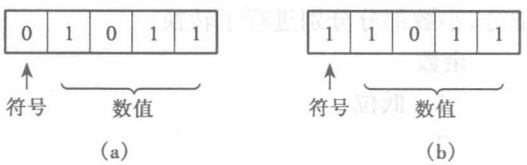


图 1-2　二进制数在机器中的表示

2）原码

原码又称为“符号-数值表示”。在以原码形式表示的正数和负数中，第1位表示符号位，对于正数，符号位记作0，对于负数，符号位记作1，其余各位表示数值部分。

假如两个带符号的二进制数分别为 N_1 和 N_2，其真值形式为

$$N_1 = +10011 \qquad N_2 = -01010$$

则 N_1 和 N_2 的原码表示形式为

$$[N_1]_{原} = 010011 \qquad [N_2]_{原} = 101010$$

根据上述原码形成规则，一个 n 位的整数 N（包括一位符号位）的原码一般表示式为

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

对于定点小数，通常小数点定在最高位的左边，这时，数值小于1。定点小数原码一般表示式为

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

从原码的一般表示式中可以看出：

(1) 当 N 为正数时，$[N]_{原}$和 N 的区别只是增加一位用 0 表示的符号位。由于在数的左边增加一位 0 对该数的数值并无影响，所以$[N]_{原}$就是 N 本身。

(2) 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。

(3) 在原码表示中，有两种不同形式的 0，即

$$[+0]_{原}=0.00\cdots0$$

$$[-0]_{原}=1.00\cdots0$$

例如 $N_1=+0.1101$　　$N_2=-0.1101$

则 $[N_1]_{原}=0.1101$　　$[N_2]_{原}=1.1101$

3) *反码*

反码又称为“对 1 的补数”。用反码表示时，左边第 1 位也为符号位，符号位为 0 代表正数，符号位为 1 代表负数。对于负数，反码的数值是将原码数值按位求反，即原码的某位为 1，反码的相应位就为 0，或者原码的某位为 0，反码的相应位就为 1。而对于正数，反码和原码相同。所以，反码数值的形成与它的符号位有关。

假如两个带符号的二进制数分别为 N_1 和 N_2，其真值形式为

$N_1=+10011$　　$N_2=-01010$

则 N_1 和 N_2 的反码表示形式为

$[N_1]_{反}=010011$　　$[N_2]_{反}=110101$

根据上述的反码形成规则，一个 n 位的整数 N(包括一位符号位)的反码一般表示式为

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

同样，对于定点小数，若小数部分的位数为 m，则它的反码一般表示为

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

从反码的一般表示式可以看出：

(1) 正数 N 的反码$[N]_{反}$与原码$[N]_{原}$相同。

(2) 对于负数 N，其反码$[N]_{反}$的符号位为 1，数值部分是将原码数值按位变反。

(3) 在反码表示中，0 的表示有两种不同的形式，即

$$[+0]_{反}=0.00\cdots0$$

$$[-0]_{反}=1.11\cdots1$$

在进行反码运算时，两数反码的和等于两数和的反码。即

$$[N_1]_{反}+[N_2]_{反}=[N_1+N_2]_{反}$$

符号位也参加运算，当符号位产生进位时，需要循环进位(即把符号位的进位加到和的最低位上去)。

【例 1-8】 已知 $N_1=+1001$，$N_2=-1011$，求 N_1+N_2

解

$$\begin{array}{rr}[N_1]_{反}= & 0\ 1\ 0\ 0\ 1\\ +)\ [N_2]_{反}= & 1\ 0\ 1\ 0\ 0\\ \hline [N_1]_{反}+[N_2]_{反}= & 1\ 1\ 1\ 0\ 1\end{array}$$