21世纪高等学校电子信息类教材

# 数字电子技术基础

● 张申科　崔葛瑾　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

21世纪高等学校电子信息类教材

# 数字电子技术基础

张申科　崔葛瑾　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

## 内 容 简 介

本书共 8 章，简明、系统地介绍数字逻辑基础、逻辑门电路、组合逻辑电路、触发器、时序逻辑电路及其应用、脉冲信号的产生和整形、可编程逻辑器件、模数转换和数模转换等内容。

本书的语言精练，适宜少学时教学，但又保持内容的完整性，同时适量地介绍新技术、新器件。书中有较多的实用性例题，以拓宽知识面，有利于培养学生的应用能力。

本书可作为高等院校电子信息类各专业的教材，也可供其他相关专业选用和工程技术人员参考。



**图书在版编目(CIP)数据**

数字电子技术基础/张申科，崔葛瑾编著. —北京：电子工业出版社，2005.5
21 世纪高等学校电子信息类教材
ISBN 7-121-01153-0

Ⅰ.数… Ⅱ.①张…②崔… Ⅲ.数字电路－电子技术－高等学校－教材 Ⅳ.TN79

中国版本图书馆 CIP 数据核字(2005)第 040273 号

责任编辑：凌 毅
印 刷：北京牛山世兴印刷厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
经 销：各地新华书店
开 本：787×1092 1/16 印张：18 字数：461 千字
印 次：2005 年 5 月第 1 次印刷
印 数：4000 册 定价：22.80 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：(010)68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

# 前　言

“电子技术基础”课程是电子信息类专业、计算机专业及相关专业的技术基础课程。当前电子技术的发展日新月异，电子技术的发展促进了电子教学的发展，也促进着电子技术教材内容的不断更新，以培养出适应电子信息时代新形势需要的技术人才。

上海市高校电子技术研究会经多次讨论，决定编写一本适应当前新形势需要的电子技术教材，供高校电子技术基础课使用。通过多年的教学改革和实践，本书对以往教学内容进行了修改更新。为适应当前电子信息发展的新形势和培养电子技术人才的迫切需要，对常规教材内容做了精选，增加新技术、新器件及其应用，注重理论联系实际的应用实例，加强新概念、新方法的讲授与训练。内容由浅入深，减少分立元件内容，强化集成电路及应用。文字上力求做到深入浅出，简明通俗。

全书共分 8 章，由张申科、崔葛瑾执笔。在本书中，逻辑符号采用的是国标符号。目录中注有“*”号的部分建议作为选讲的内容，在学时较少的情况下，可以删减这些内容。

在本书编写的过程中，参考了一些已经出版的教材和文献，在此表示衷心的感谢！

在书稿的定稿过程中，复旦大学王勇副教授、华东师范大学刘必虎教授、劳五一教授仔细阅读了初稿，并提出了详细的修改意见，在此致以诚挚的谢意。

由于编者水平有限，虽然初稿作为教材已在一些高校的相关专业使用，并进行过适当的修改，但错误仍在所难免，热忱希望使用本书的师生和广大读者提出批评和改进意见。

编著者

2005 年 5 月

# 目　录

# 第 1 章　数字逻辑基础

## 1.1　脉冲信号及其参数

在数字系统中，经常用到的是矩形脉冲，矩形脉冲的特性直接关系到数字系统能否正常工作。如图 1-1-1 所示标注了矩形脉冲特性的几个主要参数。

(1)脉冲周期 $T$：是指周期性重复的脉冲系列中，两个相邻脉冲之间的时间间隔。也可用频率 $f = \frac{1}{T}$ 表示单位时间内脉冲重复的次数。

(2) 脉冲幅度 $V_m$：是指脉冲电压的最大变化值。

(3) 脉冲宽度 $T_W$：是指从脉冲电压由低到高的 50% $V_m$ 起，到脉冲电压由高到低的 50% $V_m$ 为止的时间。

(4) 上升时间 $t_r$：是指脉冲由低到高的上升沿从 10% $V_m$ 上升到 90% $V_m$ 所需要的时间。

(5) 下降时间 $t_f$：是指脉冲由高到低的下降沿从 90% $V_m$ 下降到 10% $V_m$ 所需要的时间。

(6) 占空比 $q$：是指脉冲宽度与脉冲周期的比值，一般用百分数表示。

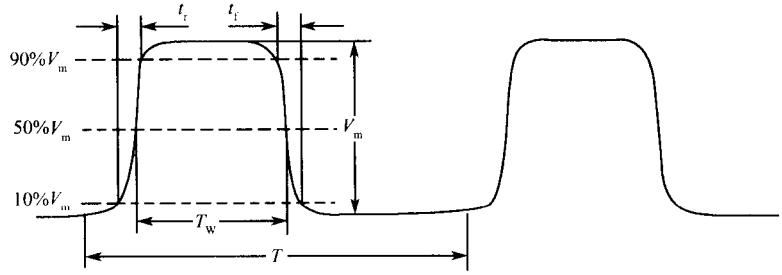


图 1-1-1　矩形脉冲的主要参数

## 1.2　数字系统中数的表示方法

### 1.2.1　数制

数制是人们对数量计数的一种统计规律。在日常生活中，最常用的进位计数制是十进制，而在数字系统中，广泛采用二进制、八进制和十六进制。

在进位计数制中，包含基数和位权两个基本因素。

基数：它是计数制中用到的数码个数，如在十进制中，它用的是 0,1,2,3,4,5,6,7,8,9 十个数码。所以它的基数是 10。一般地说，基数为 $R$ 的进位计数制中，包含有 0,1,2,…,$(R-1)$ 个数码，进位规律是“逢 $R$ 进一”，即每个数位计满 $R$ 就向高位进 1，简称 $R$ 进制。

位权：在一个进位计数制表示的数中，处于不同数位的数码代表不同的数值，某一个数位的数值是由这一位数码的值乘上处于该位的一个固定常数，不同数位上的固定常数称为位权值，简称位权。不同数位有不同的位权值，如十进制个位的位权值是 $10^0$，十位的位权值是 $10^1$，百位的位权值是 $10^2$。

任何进位制数可有两种表示形式。如十进制数125.4，它是4个数码并列在一起表示“壹佰贰拾伍点肆”，这种表示形式称为并列表示法，或称位置记数法。它也可以按权展开为

$$125.4 = 1\times(10)^2 + 2\times(10)^1 + 5\times(10)^0 + 4\times(10)^{-1}$$

等式右边的表示形式称多项式表示式，或称按权展开式。

1. 位置记数法的一般表示式

$$(N)_R = (K_{n-1}K_{n-2}\cdots K_1K_0K_{-1}K_{-2}\cdots K_{-m})_R \tag{1-2-1}$$

式中，$n$ 表示整数位数；$m$ 表示小数位数；$R$ 表示基数；$K_i$ 可以是 $R$ 进制中 $R$ 个数字符中的某一个，$0 \leqslant K_i \leqslant R-1$。

2. 多项式表示法的一般表示式

$$(N)_R = (K_{n-1}R^{n-1} + K_{n-2}R^{n-2} + \cdots + K_1R^1 + K_0R^0 + K_{-1}R^{-1} + K_{-2}R^{-2}\cdots + K_{-m}R^{-m})_R \tag{1-2-2}$$

也可写为

$$(N)_R = \left(\sum_{i=-m}^{n-1} K_iR^i\right)_R \tag{1-2-3}$$

式中，$n$ 表示整数位数；$m$ 表示小数位数；$R$ 表示基数；$K_i$ 可以是 $R$ 进制中 $R$ 个数字符中的某一个，$0 \leqslant K_i \leqslant R-1$。计数时，每一位都是逢 $R$ 进一。

下面介绍几种常用的计数制。

(1) 十进制(Decimal)

十进制的基数是10，位权为 $10^i$，有0,1,2,3,4,5,6,7,8,9十个不同的数字符号；计数时，每一位都是逢十进一。任意一个十进制数 $N$ 的多项式表示为

$$(N)_{10} = (K_{n-1}10^{n-1} + K_{n-2}10^{n-2} + \cdots + K_010^0 + K_{-1}10^{-1}\cdots + K_{-m}10^{-m}) = \sum_{i=-m}^{n-1} K_i10^i$$

式中，$K_i$ 可为 $0 \sim 9$ 数符中某一个数字；$i$ 可为 $-m$ 到 $n-1$ 之间的任意整数。

例如，十进制数524.216可展开为

$$(524.216)_{10} = 5\times10^2 + 2\times10^1 + 4\times10^0 + 2\times10^{-1} + 1\times10^{-2} + 6\times10^{-3}$$

(2) 二进制(Binary)

二进制的基数 $R=2$，第 $i$ 位的位权是 $2^i$，有0和1两个数字符号；计数时，每一位都是逢二进一。任意一个二进制数 $N$ 的多项式表示为

$$(N)_2 = (K_{n-1}2^{n-1} + K_{n-2}2^{n-2} + \cdots + K_02^0 + K_{-1}2^{-1}\cdots + K_{-m}2^{-m}) = \sum_{i=-m}^{n-1} K_i2^i$$

式中，$K_i$ 为0或者1；$i$ 可为 $-m$ 到 $n-1$ 之间的任意整数。

例如，二进制数1101.101可展开为

$$(1101.101)_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$$

由于二进制数每位只可能有0或1两种数值，在数字系统中，可用电子器件的两种不同状态来表示一位二进制数。例如，晶体管的截止和饱和、继电器的接通和断开、灯泡的亮和灭、电平的高和低，都可将其中一个状态定为0，另一个状态定为1。因此，二进制电路的实现较简单、可靠，而且存储和传输也方便。

二进制数的基本运算规则与十进制数的运算相似，但要简单得多，只要记住两个二进制数

的和与积就可以了。

加法规律：$0+0=0$　　$0+1=1+0=1$　　$1+1=10$

乘法规律：$0\times0=0$　　$0\times1=1\times0=0$　　$1\times1=1$

与十进制相比较，二进制的缺点是人们对它不熟悉，使用不习惯。因此，在数字系统中，通常把熟悉的十进制数先进行“十翻二”的运算，变成数字系统能接受的二进制数输入，再把运算的结果进行“二翻十”的转换，变成人们熟悉的十进制数。

二进制的另一缺点是：用二进制表示的一个数的位数很多，不便于书写和记忆。因此，在数字系统中，通常采用八进制和十六进制作为二进制的缩写。

(3) 八进制(Octal)

八进制的基数 $R=8$，它有 8 个符号，即 0 ～ 7；计数时，每一位都是逢八进一。任意一个八进制数 $N$ 的多项式可表示为

$$(N)_8=(K_{n-1}8^{n-1}+K_{n-2}8^{n-2}+\cdots+K_18^1+K_08^0+K_{-1}8^{-1}\cdots+K_{-m}8^{-m})=\sum_{i=-m}^{n-1}K_i8^i$$

式中，$K_i$ 为 0 ～ 7 八个数符中的某一个，$i$ 为 $-m$ 到 $n-1$ 之间的任意整数。例如

$$(23.4)_8=2\times8^1+3\times8^0+4\times8^{-1}$$

八进制数很容易转换为二进制数，八进制数中任一数码均可用 3 位二进制数表示，如

$$(23.4)_8=(010\ 011.100)_2$$

同样地，二进制数也很容易改写成八进制数，只要将二进制数的整数部分从低位向高位每 3 位分一组，最高一组不足 3 位时在高位用 0 补足；小数部分从高位向低位每 3 位一组，最后一组不足 3 位的，在低位补 0，然后把 3 位二进制数用相应的八进制数来表示。

例如，二进制数$(10101.11)_2$，改写成八进制数

$$\begin{array}{c}10\ 101.11\\ 010\ 101.110\\ \downarrow\quad\downarrow\quad\downarrow\\ 2\quad5\ .\ 6\end{array}$$

即

$$(10101.11)_2=(25.6)_8$$

(4) 十六进制(Hexadecimal)

十六进制的基数 $R=16$，位权是 $16^i$，它有 16 个数字符号，即 0 ～ 9 和 A(10)，B(11)，C(12)，D(13)，E(14)，F(15)。计数时，每一位都是逢十六进一。任意一个十六进制数 $N$ 的多项式可表示为

$$(N)_{16}=(K_{n-1}16^{n-1}+K_{n-2}16^{n-2}+\cdots+K_116^1+K_016^0+K_{-1}16^{-1}\cdots+K_{-m}16^{-m})$$

$$=\sum_{i=-m}^{n-1}K_i16^i$$

式中，$K$ 为 0 ～ 9 和 A ～ F 十六个数符中的某一个，$i$ 为 $-m$ 到 $n-1$ 之间的任意整数。例如

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

同样地，十六进制数也很容易转换为二进制数，因为十六进制的数可用 4 位二进制数来表示，如

$$(5AE)_{16}=(0101\ 1010\ 1110)_2$$

二进制数转换为十六进制数也很方便，只要把二进制数中的整数部分从低位向高位每 4 位分一组，最高一组不足 4 位时，在高位用 0 补足；小数部分从高位向低位每 4 位一组，最后一

组不足 4 位时，在低位补 0，然后把 4 位二进制数用相应的十六进制数表示。例如

$$(1110101.101)_2$$

$$0111\ 0101.1010$$

$$\downarrow \quad \downarrow \quad \downarrow$$

$$7 \quad 5 \quad . \quad A$$

即 $$(1110101.101)_2 = (75.A)_{16}$$

可见，八进制数、十六进制数比二进制数书写简短，容易读写，也便于记忆，转换为二进制数也较方便，因此，在数字系统中得到普遍应用。

## 1.2.2 不同数制之间的转换

1. 二进制数与十进制数之间的相互转换

(1) 二进制数转换为十进制数

把二进制数按权展开，利用十进制运算法则，求出其值，即为十进制数。例如

$$\begin{aligned}(1011.101)_2 &= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}\\ &= 1\times8 + 1\times2 + 1\times1 + 1\times0.5 + 1\times0.125\\ &= (11.625)_{10}\end{aligned}$$

为了方便地把二进制数转换为十进制数，应熟记二进制数各位的权。表 1-2-1 列出了常用的位权。

**表 1-2-1 二进制数各位的权**

| $i$ | −4 | −3 | −2 | −1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^i$ | 0.0625 | 0.125 | 0.25 | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |

(2) 十进制数转换成二进制数

任何十进制数的整数部分可用辗转除以 2 取余法转换成二进制数，其原理如下：若某一个十进制数 $N$ 可转换为 3 位二进制数

$$(N)_{10} = (K_2K_1K_0)_2$$

把二进制数按权展开，其多项式表示为

$$\begin{aligned}(K_2K_1K_0)_2 &= (K_2\times2^2 + K_1\times2^1 + K_0\times2^0)_{10}\\ &= [2\times(K_2\times2 + K_1) + K_0]_{10}\end{aligned}$$

$$(N)_{10}\div2 = K_2\times2 + K_1 \quad \text{——余数为 } K_0$$

用商再除以 2 得 $$(K_2\times2 + K_1)\div2 = K_2 \quad \text{——余数为 } K_1$$

不断用前次的商除以 2，直到最后的商为零。即

$$K_2\div2 = 0 \quad \text{——余数为 } K_2$$

可见，每次除以 2 所得的余数就是十进制数 $N$ 对应的二进制数 $(K_2K_1K_0)_2$。

**例 1-1** 将十进制数 11 转换为二进制数。

**解** 将十进制数 11 辗转除以 2 取其余数。

```
2 | 11        余数
2 |  5    1   (K0)   ↑
2 |  2    1   (K1)   读
2 |  1    0   (K2)   数
   0      1   (K3)   次
                     序
```

则　　$(11)_{10} = (1011)_2$

任何十进制数的纯小数部分可用“乘 2 取整”的方法，求得相应的二进制数。

对于十进制纯小数化成 $m$ 位二进制纯小数时可用多项式表示为

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

将上式两边分别乘以 2，得

$$2 \times (N)_{10} = K_{-1} + K_{-2} \times 2^{-1} + \cdots + K_{-m} \times 2^{-(m-1)} = K_{-1} + N_1$$

可知上式的整数部分即是 $K_{-1}$，将其小数部分 $N_1$ 再乘以 2，得

$$2 \times N_1 = K_{-2} + K_{-3} \times 2^{-1} + \cdots + K_{-m} \times 2^{-(m-2)}$$

上式右边的整数部分即是 $K_{-2}$。

重复上述乘法运算，即可依次求得 $K_{-1}$，直至求得 $K_{-m}$。

上述计算过程中，如果还未求到 $K_{-m}$，小数部分已为零，说明此小数已精确转换，余下位数均为零；如果求到 $K_{-m}$ 时，小数部分仍不为零，这表示转换有误差，可按照误差要求取舍。

**例 1-2**　将纯小数$(0.625)_{10}$转换为二进制数。

**解**

| | 取整 | | |
|---|---|---|---|
| $0.625 \times 2 = 1.250$ | 1 | $(K_{-1})$ | 读数次序 ↓ |
| $0.250 \times 2 = 0.50$ | 0 | $(K_{-2})$ | |
| $0.50 \times 2 = 1.00$ | 1 | $(K_{-3})$ | |

即　　$(0.625)_{10} = (0.101)_2$

**例 1-3**　将纯小数 $(0.562)_{10}$ 转换成误差 $\varepsilon$ 不大于 $2^{-6}$ 的二进制纯小数。

**解**

| | 取整 | | |
|---|---|---|---|
| $0.562 \times 2 = 1.124$ | 1 | $(K_{-1})$ | 读数次序 ↓ |
| $0.124 \times 2 = 0.248$ | 0 | $(K_{-2})$ | |
| $0.248 \times 2 = 0.496$ | 0 | $(K_{-3})$ | |
| $0.496 \times 2 = 0.992$ | 0 | $(K_{-4})$ | |
| $0.992 \times 2 = 1.984$ | 1 | $(K_{-5})$ | |
| $0.994 \times 2 = 1.968$ | 1 | $(K_{-6})$ | |

所以，$(0.562)_{10} = (0.100011)_2$，且其误差 $\varepsilon < 2^{-6}$。

任何十进制数均可将其整数部分和纯小数部分分开，分别用“辗转除以 2 取余”法和“乘 2 取整”法化成二进制数形式，然后将二进制形式的整数部分和纯小数部分合成相应的十进制数所对应的二进制数。

**例 1-4**　将十进制数 11.562 转化为二进制数。

**解**　可由　　$(11)_{10} = (1011)_2$

$(0.562)_{10} = (0.100011)_2$

合成　　$(11.562)_{10} = (1011.100011)_2$

2. 任意进制数与十进制数之间的转换

任意进制数与十进制数之间的转换原理及方法与二进制数与十进制数之间的转换原理及

方法相类似，此处不再重复。

任意两种进制之间的转换，一般说来是先由一种进位制转换为十进制，再由十进制转换为另一种进制，把十进制作为桥梁。

### 1.2.3 码制

1. 带符号的二进制数的代码

在通常的算术运算中，用“+”号表示正数，用“-”号表示负数。但在数字系统中，正、负数的表示方法为：把一个数的最高位作为符号位，用“0”表示“+”；用“1”表示“-”。连同符号位一起作为一个数，称为机器数，它原来的数值形式则称为这个机器数的真值。例如

$$X_1 = +0.1101; \qquad X_2 = -0.1101$$

它的机器数为

$$X_1 = 0.1101; \qquad X_2 = 1.1101$$

在数字系统中，表示机器数的方法很多，常用的有原码、反码和补码。

(1) 原码(True Form)

原码表示法又称符号 - 数值表示法。正数的符号位用“0”表示，负数的符号位用“1”表示，数值部分保持不变。

① 小数原码的定义

若二进制数 $X = \pm 0.X_{-1}X_{-2}\cdots X_{-m}$

当 $X > 0$ 时，$X = +0.X_{-1}X_{-2}\cdots X_{-m}$，则

$$[X]_{原} = 0.X_{-1}X_{-2}\cdots X_{-m}$$

当 $X < 0$ 时，$X = -0.X_{-1}X_{-2}\cdots X_{-m}$，则

$$[X]_{原} = 1.X_{-1}X_{-2}\cdots X_{-m} = 1 - (-0.X_{-1}X_{-2}\cdots X_{-m})$$

例如，$X_1 = +0.1101$，则

$$[X_1]_{原} = 0.1101$$

$X_2 = -0.1101$，则

$$[X_2]_{原} = 1 - (-0.1101) = 1.1101$$

零的原码有两种表示形式，即

$$[+0]_{原} = 0.00\cdots0$$
$$[-0]_{原} = 1.00\cdots0$$

因此，小数原码定义为

$$[X]_{原} = \begin{cases} X & \text{当 } 0 \leqslant X < 1 \text{ 时} \\ 1 - X & \text{当 } -1 < X \leqslant 0 \text{ 时} \end{cases} \tag{1-2-4}$$

② 整数原码的定义

若 $X = \pm X_{n-1}X_{n-2}\cdots X_0$

当 $X > 0$ 时，$X = +X_{n-1}X_{n-2}\cdots X_0$，则

$$[X]_{原} = 0X_{n-1}X_{n-2}\cdots X_0$$

当 $X < 0$ 时，$X = -X_{n-1}X_{n-2}\cdots X_0$，则

$$[X]_{原} = 1X_{n-1}X_{n-2}\cdots X_0$$

例如，$X = -1101$，则 $[X]_{原} = 11101$。

所以，整数原码定义为

$$[X]_{原}=\begin{cases}X & 当\ 0\leqslant X<2^n\ 时\\ 2^n-X & 当\ -2^n<X\leqslant 0\ 时\end{cases}\tag{1-2-5}$$

原码表示法虽然简单易懂，但在数字系统中，如果进行两个异号原码的加法运算，必须先判别两数的大小，然后从大数中减去小数，最后，还要判别结果的符号位，这样就增加了运算时间。

(2) 反码(One's Complement)

反码的符号位表示法与原码相同，即符号“0”表示正数，“1”表示负数。与原码不同的是数值部分，即正数反码的数值和原码数值相同，而负数反码的数值是原码的数值按位求反。

① 整数的反码

若 $X=\pm X_{n-1}X_{n-2}\cdots X_0$，则定义为

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

例如，$X_1=+1101$，则$[X]_{反}=01101$；$X_2=-1101$，则

$$\begin{aligned}[X_2]_{反}&=(2^5-1)+X\\&=(100000-000001)+(-1101)\\&=11111-1101\\&=10010\end{aligned}$$

最高位是符号位；后 4 位是数值部分，这刚好是原码数值部分按位求反。

② 小数的反码

若 $X=\pm 0.X_{n-1}X_{n-2}\cdots X_0$，则定义为

$$[X]_{反}=\begin{cases}X & 当\ 0\leqslant X<1\ 时\\ 2-2^{-n}+X & 当\ -1<X\leqslant 0\ 时\end{cases}\tag{1-2-7}$$

例如，$X_1=+0.1101$，则$[X_1]_{反}=0.1101$；$X_2=-0.1101$，则$[X_2]_{反}=1.0010$。也可计算$[X_2]_{反}=2-2^{-4}+X=10.0000-0.0001-0.1101=1.0010$。

③ 零的反码

零的反码有两种形式

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

进行反码加减法时，要将运算结果的符号位产生的进位(0 或 1) 加到和的最低位，才能得到最后结果。

例如，$X_1=+1101$，$X_2=-0101$，将 $X_1$ 和 $X_2$ 进行反码加

$$[X_1]_{反}+[X_2]_{反}=01101+11010=\text{“1”}00111$$
$$[X_1+X_2]_{反}=00111+1=01000$$
$$X_1+X_2=1101-0101=+1000$$

这与$[X_1+X_2]_{反}=01000$一致。

(3) 补码(Two's Complement)

补码又称对 2 的补码。补码的符号表示和原码相同，“0”表示正数；“1”表示负数。正数的补码和原码、反码相同，就是二进制数本身。负数的补码是这样得到的：将数值部分按位求反，再在最低位加 1。

① 整数的补码

若 $X = \pm X_{n-1}X_{n-2}\cdots X_0$，则定义为

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

例如，$X_1 = +1101$，则$[X_1]_{补} = 01101$；$X_1 = -1101$，则$[X_2]_{补} = 10011$。

② 小数的补码

若 $X = \pm 0.X_{-1}X_{-2}\cdots X_{-m}$，则定义为

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

③ 零的补码

$$[0]_{补} = 0.000\cdots0$$

对于零，补码仅有一种形式。

引入补码以后，可将数字系统的减法运算用加法实现，并将运算结果产生的进位丢掉，才能得到正确结果。

例如，在补码系统中实现 $X_1 - X_2$ 运算。

$$X_1 = 1101 \qquad X_2 = 0101$$

求得 $[X_1]_{补} = 01101 \qquad [-X_2]_{补} = 11011$

做 $[X_1]_{补} + [-X_2]_{补} = 01101 + 11011 = \text{“1”}01000$

丢掉最高位“1”得 $[X_1 - X_2]_{补} = 01000$

真值为 $X_1 - X_2 = 1000$

可见，两数相减时，用补码求和运算比用原码求和简单，但补码的缺点是负数用补码表示不直观。

表 1-2-2 给出了几个典型数的真值、原码、反码和补码表示。

**表 1-2-2 几个典型数的真值、原码、反码和补码表示**

| $X$ | $[X]_{原}$ | $[X]_{反}$ | $[X]_{补}$ | $X$ | $[X]_{原}$ | $[X]_{反}$ | $[X]_{补}$ |
|---|---|---|---|---|---|---|---|
| +1001 | 01001 | 01001 | 01001 | -0.0000 | 1.0000 | 1.1111 | 0.0000 |
| +0001 | 00001 | 00001 | 00001 | -0.0010 | 1.0010 | 1.1101 | 1.1110 |
| +0.1101 | 0.1101 | 0.1101 | 0.1101 | -0011 | 10011 | 11100 | 11101 |
| +0.0000 | 0.0000 | 0.0000 | 0.0000 | -1010 | 11010 | 10101 | 10110 |

2. 二 - 十进制编码

十进制数除了转换成二进制数外，还可以用代码来表示。这种代码既具有二进制数的形式，又具有十进制数的特点，便于传输和处理。一位十进制数有 0 ~ 9 十个不同的数码，需要用 4 位二进制数才能表示。而 4 位二进制数可有 16 种不同状态，从 16 种状态中取出 10 种状态来表示 0 ~ 9 十个数码。其编码方式很多，一般分为有权码和无权码两种。有权码是指二进制数中的每一位都对应有固定的权值，把每一位的 1 代表的权值加起来，所得的结果就是所表示的

十进制数。无权码是指 4 位二进制数中的每一位无固定的权，它必须遵循另外的规则。最常用的十进制数的二进制编码主要有以下几种。

(1) 8421 码

8421 码是有权码，是十进制代码中最常见的代码，也称二－十进制码，或称 BCD 码(Binary Code Decimal)，4 位二进制码从高位到低位每位的权分别为 $2^3$，$2^2$，$2^1$，$2^0$，即为 8，4，2，1。

(2) 5421 码

5421 码也是一种有权码，自高位到低位，每位的权分别为 5，4，2，1。

(3) 2421 码

它也是一种有权码，每位的权从高位到低位分别为 2，4，2，1。

(4) 余 3 码

余 3 码是一种无权码。十进制数用余 3 码表示，要比 8421 码在二进制数值上多 3，故称余 3 码，它可由 8421 码加 0011 得到。

表 1-2-3 列出了上述 4 种十进制数的编码。编码的形式还有很多，只要在 4 位二进制数的 16 种状态中任取 10 种状态分别表示十进制数的 0 ～ 9 十个数码即可。

**表 1-2-3　4 种十进制数的编码**

| 十进制数 | 8421 码 | 5421 码 | 2421 码 | 余 3 码 |
|---|---|---|---|---|
| 0 | 0000 | 0000 | 0000 | 0011 |
| 1 | 0001 | 0001 | 0001 | 0100 |
| 2 | 0010 | 0010 | 0010 | 0101 |
| 3 | 0011 | 0011 | 0011 | 0110 |
| 4 | 0100 | 0100 | 0100 | 0111 |
| 5 | 0101 | 1000 | 0101 | 1000 |
| 6 | 0110 | 1001 | 0110 | 1001 |
| 7 | 0111 | 1010 | 0111 | 1010 |
| 8 | 1000 | 1011 | 1110 | 1011 |
| 9 | 1001 | 1100 | 1111 | 1100 |

(5) 格雷码(Gray Code)

格雷码是一种无权码，其特点是：任意两个相邻数的代码只有一位不同。这种代码可以减少代码在译码时产生的错误，因此是一种高可靠性的编码。表 1-2-4 给出了 4 种格雷码。从表中可以看出，格雷码 1 有一个特点，即对该组编码的中线(中线位于十进制数 4，5 之间)而言，其最高位的代码一一相反，而其余各位的代码则相同；也就是说，各个代码之间对中线一一“反射”，通常把这种具有反射性的格雷码称反射码。典型格雷码可以从二进制码转换得到($G_i = B_{i+1} \oplus B_i$，$\oplus$ 称为异或运算，将在第 2 章中讨论)。对于十进制计数，典型格雷码的 9(1101) 回到 0(0000) 就有 3 位发生变化，这和格雷码的特点不一致，因此，另外设计了各种十进制格雷码，使得 9 到 0 也只有一位不同。表 1-2-4 列出了其中两种。