第 1 章　绪　　论

1.1　脉冲信号

1. 脉冲波形

脉冲(pulse)这个词包含着脉动和短促的意思。脉冲信号通常是一些具有断续性和突发性特点的、短暂出现的、周期或非周期性时间函数的电压或电流。从广义上讲，凡是非正弦形状的电压(或电流)信号都可称为脉冲信号。图 1.1.1 示出了几种常见的脉冲信号的波形：矩形波、尖顶波、锯齿波、钟形波、梯形波和阶梯波。电压波形图的横向坐标为时间，纵向坐标为电压(为了波形清晰，没有画出坐标轴线)。

脉冲电路是各种波形的脉冲的产生以及脉冲波形间相互变换的电路。

下面举两个矩形脉冲的应用实例。

图 1.1.2(a)是一个发电报的简单装置，电键 K、电阻 R 和电池 E 串联。按下电键，电阻上电压 v_O 等于 E；不按电键，v_O 等于 0。于是，每按一次电键，在电阻上就可以产生一个脉冲电压。根据电文内容，不断忽长忽短地按动电键，就可以得到一系列幅度为 E 而宽度不同的矩形脉冲(见图 1.1.2(b))。

图 1.1.3(a)示出了一个通用的商品条形代码标志。每一个数字由两根黑线条和两根白线条组成，利用线条宽度的不同来表示不同的数字。每一个数字的黑和白线条的总宽度为 7 个单位尺寸(见图 1.1.3(b))，即每个数字的黑白线条由宽度均为单位尺寸的 7 个黑、白窄线条组成。若把每个窄线条的颜色用一个数码表示：0 表示白色，1 表示黑色，则通用商品代码的编码规则如表 1.1.1 所示。在图 1.1.3(a)中，最左边和最右边的三条(黑-白-黑)为保护条组，中间的五条(白-黑-白-黑-白)也是保护条组。在中间保护条组左边的数字按表 1.1.1 中左部编码规则编码；右边的数字按表 1.1.1 中右部编码规则编码。例如，图 1.1.3(b)示出了数字 7 按左部编码规则的黑白条纹。此外，最左边的数字和最右边的数字分别写在编码条纹的左侧和右侧(见图 1.1.3(a))。商品条形代码标志的黑白条纹经过读入机变成电信号。若将白色对

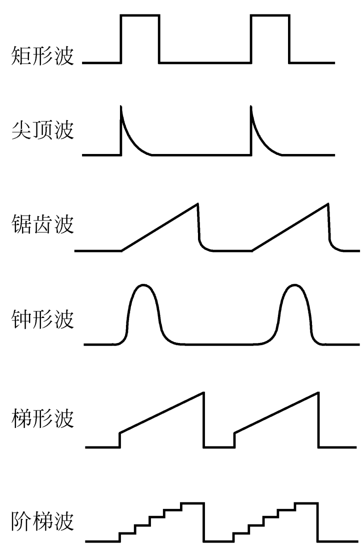

图 1.1.1　常见的脉冲波形

应低电位 0V,黑色对应高电位 $+E$,则宽窄不同的黑白条纹就变成类似图 1.1.2(b)的宽窄不同的矩形脉冲信号。

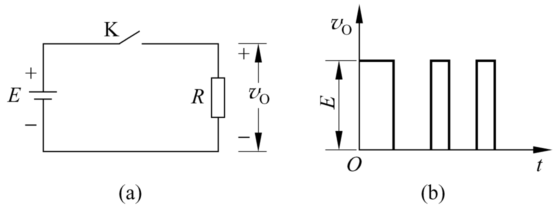

图 1.1.2 发报示意图

(a) 电路;(b) 所代表的电文

图 1.1.3 通用的商品条形代码标志

(a) 条形代码标志实例;(b) 7 的条形代码

表 1.1.1 编码规则

数字	左部编码	右部编码
0	0 0 0 1 1 0 1	1 1 1 0 0 1 0
1	0 0 1 1 0 0 1	1 1 0 0 1 1 0
2	0 0 1 0 0 1 1	1 1 0 1 1 0 0
3	0 1 1 1 1 0 1	1 0 0 0 0 1 0
4	0 1 0 0 0 1 1	1 0 1 1 1 0 0
5	0 1 1 0 0 0 1	1 0 0 1 1 1 0
6	0 1 0 1 1 1 1	1 0 1 0 0 0 0
7	0 1 1 1 0 1 1	1 0 0 0 1 0 0
8	0 1 1 0 1 1 1	1 0 0 1 0 0 0
9	0 0 0 1 0 1 1	1 1 1 0 1 0 0

2. 矩形脉冲

矩形波是数字电路中最常用的脉冲波形。矩形脉冲具有两个固定电平,其电平转换时间与每个电平的持续时间相比可以忽略。图 1.1.4 示出一周期性的矩形脉冲的波形,其主要参数如下。

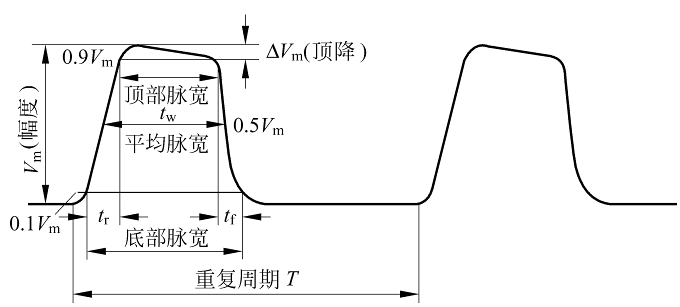

图 1.1.4 矩形脉冲的波形参数

脉冲幅度 V_m：矩形脉冲的高电平和低电平之差，它反映了脉冲信号的大小。

脉宽 t_w（或称为脉冲持续期）：矩形脉冲起始和终了时刻之间的时间间隔。若将脉冲前、后沿上瞬时值为 $0.5V_m$ 的对应点之间的时间间隔定为脉宽，则称作平均脉宽。此外，还有顶部脉宽和底部脉宽。

重复周期 T：相邻两个脉冲对应点之间的时间间隔。重复周期的倒数称为重复频率 f。

脉冲占空系数 Q^{-1}（或称脉冲工作比）：脉宽 t_w 和重复周期 T 的比值。Q^{-1} 的倒数 Q 称为空度比。

上升沿 t_r：脉冲电压从 $0.1V_m$ 上升到 $0.9V_m$ 所需的时间。

下降沿 t_f：脉冲电压从 $0.9V_m$ 下降到 $0.1V_m$ 所需的时间。

理想的矩形脉冲的 t_r 和 t_f 等于 0。

1.2 数字信号和数字电路

人们在现实生活中遇到的许多物理量，如温度、压力、距离、时间等，一般都具有连续变化的特点。它们可以在一定范围内取任意实数值，这类物理量称为模拟量。在工程应用中，为了测量、传递和处理这些物理量，常把它们通过传感器转换成与之成比例的电压（或电流）。这些电信号表示和模拟了实际的物理量，故称为模拟信号，其电压值（或电流值）在一定范围内是连续的变量。模拟信号所传送的内容称为模拟信息。处理模拟信号的电路称为模拟电路（analog circuit）。

数字量是离散的，只能按有限个或可数的量化单位（又称为增量、量化层）取值。例如，某一实际距离的值为 2567.82326…km，若取量化单位为 1km，则代表此距离的数字量为 2568km；若取量化单位为 1m，则数字量为 2567823m。显然，量化单位越小，精度越高。但是，这是以增加数字量的位数为代价的。量化单位的选择，取决于实际工作所要求的精度。与数字量相对应的电信号称为数字信号。数字信号所传送的内容称为数字信息。处理数字信号的电路称为数字电路（digital circuit）。

在进行信息传递和处理时，数字方式和模拟方式相比有下述优点。

（1）精度和可靠性高

同一物理量可以用连续的模拟信号表示，也可以用离散的数字信号表示。用数字信号进行信息传送和处理，容易达到高精度和高可靠性。例如，当温度信息用模拟信号传送时，模拟信号的电压值正比于温度值。若要求精度为对应最高温度的电压的千分之一，则必须要求传送途径上的干扰电压低于所传送信号电压最大值的千分之一。在干扰严重的地方，这样的要求往往难以实现。若利用数字信号传送，就容易达到这一精度要求。数字信号传输时，把所传送的数字量按照一定的编码规则编成一组矩形脉冲序列。数字量的精度越高，则量化单位越小，所需数码的位数越多。但这只需增加每组脉冲序列所能容纳的脉冲个数，而脉冲的幅值无需改变。脉冲幅度与抗干扰能力有关。当脉冲幅度足够大时，只有当传送途中遇到相当大的干扰，才能改变信号中脉冲的有无，破坏信息的内容。因此，数字信号具有较强的抗干扰能力，容易达到较高的精度。

(2) 使用灵活,易于器件标准化

随着半导体工艺的发展,数字电路器件的体积越来越小,集成度越来越高。今天,可以在一块硅片上制造几千个、几万个甚至几千万个元件,并可制造单片的数字计算机、单片的信号处理器等功能很强的标准化的通用器件,也可以由使用者定制专用的芯片。这些器件的出现,使数字系统体积小、重量轻、耗电省,并且提高了工作速度和可靠性。目前,数字电路广泛应用在军事、工业、商业、企业管理、交通运输、医疗保健和文化教育等各个领域,同时已进入到人们的日常生活中。

第 2 章　数制与编码

2.1　数制

人们在日常生活中已习惯使用十进制数。例如，7492 读作七千四百九十二。7 在千位上，读作七千；4 在百位上，读作四百；9 在十位上，读作九十；2 在个位上，读作二。这是一种位置记数法，将千、百、十、个称为权(weight)；7，4，9，2 称为系数(digit)。每个系数所在的位置不同，则所对应的权值不同。于是有

$$(7492)_{10} = (7 \times 10^3 + 4 \times 10^2 + 9 \times 10^1 + 2 \times 10^0)_{10}$$

式中的下角标 10 表示十进制。十进制数以 10 为基数(base)，其一般形式为

$$\begin{aligned}(N)_{10} &= (a_{n-1}a_{n-2}\cdots a_1 a_0)_{10} \\ &= (a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} + \cdots + a_1 \times 10^1 + a_0 \times 10^0)_{10}\end{aligned}$$

带小数的数写为

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

式中，a_0 和 a_{-1} 之间的点称为小数点，n 和 m 分别是整数的位数和小数的位数。由上式看到十进制每位的权是 10 的幂。十进制中有 10 个数字：0，1，2，3，4，5，6，7，8，9。

在数字系统中常常采用二进制数，这是因为二进制数的基数为 2，它只有 0 和 1 两个数字，运算规则简单，便于电路实现。二进制数也采用位置记数法，每位的权是 2 的幂(见表 2.1.1)。二进制数的一般形式为

表 2.1.1　二进制各位的权

二进制的位置号	12	11	10	9	8	7	6	5	4	3	2	1	0
权(十进制表示)	2^{12} (4096)	2^{11} (2048)	2^{10} (1024)	2^9 (512)	2^8 (256)	2^7 (128)	2^6 (64)	2^5 (32)	2^4 (16)	2^3 (8)	2^2 (4)	2^1 (2)	2^0 (1)

二进制的位置号	−1	−2	−3	−4	−5	−6
权(十进制表示)	2^{-1} (0.5)	2^{-2} (0.25)	2^{-3} (0.125)	2^{-4} (0.0625)	2^{-5} (0.03125)	2^{-6} (0.015625)

$$(N)_2 = (b_{n-1}b_{n-2}\cdots b_1 b_0 \cdot b_{-1}b_{-2}\cdots b_{-m})_2$$

式中的下角标2表示二进制。$(N)_2$ 等值的十进制数为

$$\begin{aligned}(N)_2 &= (b_{n-1}\times 2^{n-1} + b_{n-2}\times 2^{n-2} + \cdots + b_1\times 2^1 + b_0\times 2^0 \\ &\quad + b_{-1}\times 2^{-1} + b_{-2}\times 2^{-2} + \cdots + b_{-m}\times 2^{-m})_{10} \\ &= (\sum_{i=-m}^{n-1} b_i 2^i)_{10}\end{aligned}$$

【例 2.1.1】

$$\begin{aligned}(11010.11)_2 &= (1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 \\ &\quad + 1\times 2^{-1} + 1\times 2^{-2})_{10} \\ &= (26.75)_{10}\end{aligned}$$

二进制数的缺点是：同一个数值用二进制数表示，比用十进制数表示的位数要多。为了书写简便，在数字系统中，常采用以2的幂为基数的八进制数和十六进制数。八进制有8个数：0,1,2,3,4,5,6,7；十六进制有16个数：0,1,2,3,4,5,6,7,8,9,A,B,C,D,E和F，其中A,B,C,D,E和F分别对应十进制的10,11,12,13,14和15(即二进制的1010,1011,1100,1101,1110,1111)。八进制的基数为8；十六进制的基数为16。二进制、八进制、十进制和十六进制常以B、O、D和H表示，它们是binary、octal、decimal和hexadecimal的字头。在VHDL语言中，二进制、八进制和十六进制分别用B、O和X表示。用二进制表示的八进制、十进制、十六进制见2.4节。

2.2 不同数制之间的相互转换

本节介绍借助于十进制运算实现的数制转换方法。

1. 十进制转换成其他进制

(1) 整数的数制转换

十进制整数转换成二进制整数时，可采用基数除法，其步骤如下。

① 将给定的十进制数除以基数2，余数就是等值的二进制数的最低位。

② 将上一步的商再除以基数2，余数是等值的二进制数的次低位。

③ 重复步骤②，直到最后所得的商等于0为止。各次除得的余数，便是二进制各位的数(最后一次的余数是最高位)。

【例 2.2.1】

$$(41)_{10} = (101001)_2$$

解 算式如图2.2.1所示。

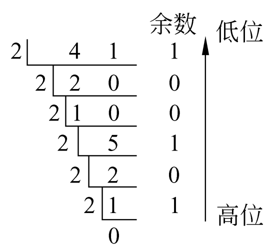

图2.2.1 例2.2.1算式

“基数除法”的原理是：二进制整数每次被2除，相当于将其向右移动一位，并将移位前的最低位从整数中移出。例如，二进制整数$(101001)_2$右移一位后为$(10100.1)_2$，这个右移到小数部分的最低位恰好对应(该二进制数移位前的)等值十进制数被2除后的余数。

类似地，利用基数除法原理可实现十进制整数向八进制和十六进制整数的转换。

(2) 纯小数的数制转换

十进制纯小数转换成二进制纯小数可采用基数乘法，其步骤如下。

① 将给定的十进制纯小数乘以基数 2，其积的整数部分就是等值二进制纯小数的最高位。

② 将上一步乘积的小数部分再乘以基数 2，所得乘积的整数部分是等值二进制纯小数的次高位。

③ 重复步骤②，直至乘积的小数部分为 0，或者达到要求的精度为止(舍入误差小于最低位对应的数值)。各次乘积的整数部分便是二进制纯小数的各位(最后一次乘积的整数部分是最低位)。

【例 2.2.2】

$$(0.89)_{10}=(0.11100011)_2，舍入误差小于 2^{-8}\approx 4\times 10^{-3}。$$

解 算式如图 2.2.2 所示。

基数乘法的原理是：二进制小数每次乘 2，相当于将其向左移动一位，并将移位前的最高位从小数部分移到整数部分。例如，二进制小数$(0.11100011)_2$左移一位后为$(1.1100011)_2$，这个左移到整数部分的最高位恰好对应(该二进制小数移位前的)等值十进制小数乘 2 后的整数值。

类似地，利用基数乘法原理可实现十进制小数向八进制和十六进制小数的转换。

(3) 带小数的数制转换

一个带有小数的十进制数转换成带有小数的二进制数时，将其整数部分和小数部分分开计算，整数部分用基数除法，小数部分用基数乘法，最后，将求得的等值整数部分和等值的小数部分合起来，即可得到等值的带有小数的二进制数。

【例 2.2.3】

$(41.89)_{10}=(101001.11100011)_2$，舍入误差小于 2^{-8}。

采用同样的方法，可实现带小数的十进制数向八进制、十六进制数的转换。

$$
\begin{array}{l|r}
 & 0.89 \\
 & \times\quad 2 \\ \hline
\text{高位}\downarrow & (1).78 \\
 & \times\quad 2 \\ \hline
 & (1).56 \\
 & \times\quad 2 \\ \hline
 & (1).12 \\
 & \times\quad 2 \\ \hline
 & (0).24 \\
 & \times\quad 2 \\ \hline
 & (0).48 \\
 & \times\quad 2 \\ \hline
 & (0).96 \\
 & \times\quad 2 \\ \hline
 & (1).92 \\
 & \times\quad 2 \\ \hline
\text{低位} & (1).84
\end{array}
$$

图 2.2.2 例 2.2.2 算式

2. 由其他进制转换成十进制

由 r 进制转换成十进制可用下式直接求得：

$$
\begin{aligned}
(N)_r &= (k_{n-1}k_{n-2}\cdots k_1k_0\cdot k_{-1}k_{-2}\cdots k_{-m})_r \\
&= (k_{n-1}\times r^{n-1}+k_{n-2}\times r^{n-2}+\cdots+k_1\times r^1+k_0\times r^0 \\
&\quad +k_{-1}\times r^{-1}+k_{-2}\times r^{-2}+\cdots+k_{-m}\times r^{-m})_{10}
\end{aligned}
$$

式中的 r 为基数值，二进制、八进制和十六进制的 r 分别为 2，8 和 16。

【例 2.2.4】

$$(101001.1101)_2=(1\times 2^5+1\times 2^3+1\times 2^0+1\times 2^{-1}+1\times 2^{-2}+1\times 2^{-4})_{10}$$

$=(32+8+1+0.5+0.25+0.0625)_{10}$
$=(41.8125)_{10}$

2.3 有符号的二进制数

2.3.1 反码和补码

本节讨论无符号的二进制数的反码和补码,下几节再讨论有符号的二进制数的表示方法,以及与其相关的加、减运算和算术移位等问题。

1. 反码

一个二进制数的反码就是将该数的每一位取反,即0变1,1变0。反码又称1补(one's complement)。

【例2.3.1】 $N=101100_2$ 的反码为

$$N_{反}=010011_2$$

对一个反码 $N_{反}$ 求反可得原数 N。无符号二进制数 N 也称为该数的原码 $N_{原}$。

2. 补码

一个二进制数的补码就是在该数的反码的最低位加1。补码又称2补(two's complement)。

【例2.3.2】 $N=101100_2$ 的补码为

$$N_{补}=(010011+1)_2=010100_2$$

对一个补码 $N_{补}$ 求补可得原数 N(即原码 $N_{原}$)。如例2.3.2中对 010100_2($N_{补}$)求补,得到 101100_2(N)。

求一个二进制数的补码也可按下述方法进行:从最低位开始,保留所有的低位0不变;遇到第一个1时,保持这个1不变;以后各高位均取反,即1换0,0换1。

n 位二进制数 N(即原码 $N_{原}$)与补码 $N_{补}$ 之和为等值十进制数 2^n,换句话说,n 位二进制数 N 的补码 $N_{补}=2^n-N_{原}$,这便是称为补码的原因。如在例2.3.2中,$N_{原}=(101100)_2$,$n=6$,$(2^n)_{10}=(1000000)_2$,$N_{补}=(1000000-101100)_2=(010100)_2$。

2.3.2 二进制正、负数的表示法

在数字系统中,数的符号规定用0表示正号,用1表示负号。有符号的二进制数有以下三种表示法(见表2.3.1):

(1) 原码(符号-绝对值)表示法;

(2) 反码表示法;

(3) 补码表示法。

表 2.3.1　三种不同的 3 位二进制正、负数表示法

<table>
<tr><th rowspan="2">十进制数</th><th colspan="3">二　进　制　数</th></tr>
<tr><th>原　码</th><th>反　码</th><th>补　码</th></tr>
<tr><td>−8</td><td></td><td></td><td>1 0 0 0</td></tr>
<tr><td>−7</td><td>1 1 1 1</td><td>1 0 0 0</td><td>1 0 0 1</td></tr>
<tr><td>−6</td><td>1 1 1 0</td><td>1 0 0 1</td><td>1 0 1 0</td></tr>
<tr><td>−5</td><td>1 1 0 1</td><td>1 0 1 0</td><td>1 0 1 1</td></tr>
<tr><td>−4</td><td>1 1 0 0</td><td>1 0 1 1</td><td>1 1 0 0</td></tr>
<tr><td>−3</td><td>1 0 1 1</td><td>1 1 0 0</td><td>1 1 0 1</td></tr>
<tr><td>−2</td><td>1 0 1 0</td><td>1 1 0 1</td><td>1 1 1 0</td></tr>
<tr><td>−1</td><td>1 0 0 1</td><td>1 1 1 0</td><td>1 1 1 1</td></tr>
<tr><td>−0</td><td>1 0 0 0</td><td>1 1 1 1</td><td rowspan="2">0 0 0 0</td></tr>
<tr><td>+0</td><td>0 0 0 0</td><td>0 0 0 0</td></tr>
<tr><td>1</td><td>0 0 0 1</td><td>0 0 0 1</td><td>0 0 0 1</td></tr>
<tr><td>2</td><td>0 0 1 0</td><td>0 0 1 0</td><td>0 0 1 0</td></tr>
<tr><td>3</td><td>0 0 1 1</td><td>0 0 1 1</td><td>0 0 1 1</td></tr>
<tr><td>4</td><td>0 1 0 0</td><td>0 1 0 0</td><td>0 1 0 0</td></tr>
<tr><td>5</td><td>0 1 0 1</td><td>0 1 0 1</td><td>0 1 0 1</td></tr>
<tr><td>6</td><td>0 1 1 0</td><td>0 1 1 0</td><td>0 1 1 0</td></tr>
<tr><td>7</td><td>0 1 1 1</td><td>0 1 1 1</td><td>0 1 1 1</td></tr>
</table>

对于正数，三种表示法是相同的，即符号位为 0，随后的数据部分是二进制数的原码。

对于负数，三种表示法是不同的：

(1) 原码表示法。负数的符号位为 1，随后是二进制数的原码。

(2) 反码表示法。负数的符号位为 1，随后是二进制数的反码。

(3) 补码表示法。负数的符号位为 1，随后是二进制数的补码。

在数字系统中，最常用的是原码表示法和补码表示法。下面将会看到，采用补码表示法可使二进制数的加、减运算容易实现。

根据补码表示法的规则和补码的定义($N_{补}=2^n-N_{原}$)可知，对一个用补码表示法表示的 n 位二进制整数 X：$x_{n-1}x_{n-2}\cdots x_1x_0$，其等值的十进制数为

$$X=-\left(x_{n-1}2^{n-1}-\sum_{i=0}^{n-2}x_i2^i\right)_{10}=\left(-x_{n-1}2^{n-1}+\sum_{i=0}^{n-2}x_i2^i\right)_{10}$$

式中，x_{n-1} 为符号位，$x_{n-2}\cdots x_1x_0$ 为数据位。

2.3.3　带符号位的二进制数的补码运算

1. 补码表示法的加法

补码表示的两个带符号位的二进制数相加时，将两个数(连同它们的符号位)相加，相加结果为和的补码，即

$$(A+B)_{补}=(A)_{补}+(B)_{补}$$

若符号位有进位产生,则将该进位抛弃(事实上,在电路中,这个进位可以自动抛弃)。

【例 2.3.3】

+6	0 0 0 0 1 1 0	+6 的二进制补码
+) +9	+) 0 0 0 1 0 0 1	+9 的二进制补码
+15	0 0 0 1 1 1 1	+15 的二进制补码

【例 2.3.4】

−6	1 1 1 1 0 1 0	−6 的二进制补码
+) +9	+) 0 0 0 1 0 0 1	+9 的二进制补码
+3	1 0 0 0 0 0 1 1	+3 的二进制补码

抛弃

【例 2.3.5】

+6	0 0 0 0 1 1 0	+6 的二进制补码
+) −9	+) 1 1 1 0 1 1 1	−9 的二进制补码
−3	1 1 1 1 1 0 1	−3 的二进制补码

【例 2.3.6】

−6	1 1 1 1 0 1 0	−6 的二进制补码
+) −9	+) 1 1 1 0 1 1 1	−9 的二进制补码
−15	1 1 1 1 0 0 0 1	−15 的二进制补码

抛弃

2. 补码表示法的减法

由表 2.3.1 不难看出补码表示的负数相当于将对应的正数(连同符号位一起)求补,也就是说,把补码表示的正数的符号位看作一位数,并同后面的数据部分一起作为一个数对待,求其补码,可得对应负数的补码表示。例如,补码表示的 $+6_{10}$ 为 0110_2,将其求补为 1010_2,它就是补码表示的 -6_{10}。

由表 2.3.1 还可以看出,对一个补码表示的二进制负数(连同符号位)取补,可得对应正数的补码表示。例如,−6 为 1010_2,取补后为 +6 的补码表示 0110_2。

考虑到

$$(\pm A)-(+B)=(\pm A)+(-B)$$
$$(\pm A)-(-B)=(\pm A)+(+B)$$