应用型特色

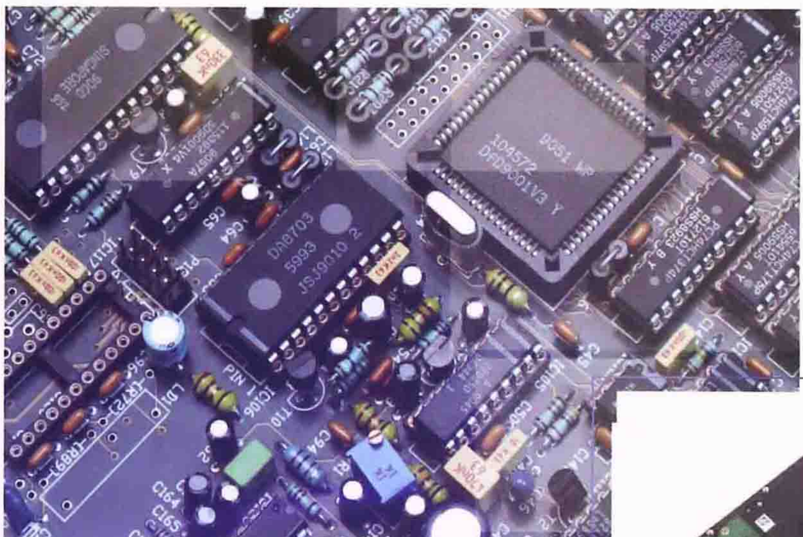
高等学校应用型特色规划教材

河北省精品课程

SHUZI DIANZI JISHU JICHU

数字电子技术基础

谢志远 主 编
尚秋峰 副主编



免费赠送电子课件

- 在借鉴国内教材优点的基础上，对传统教学内容进行了精选与整合，增加了EDA等现代数字电子技术的比重。
- 在保证基础的前提下，更新课程内容，介绍了当代先进的电子技术知识，并重点介绍了典型电路的逻辑功能、外特性和应用，突出了数字集成电路的应用和EDA技术。

清华大学出版社

高等学校应用型特色规划教材

数字电子技术基础

谢志远　主　编

尚秋峰　副主编

清华大学出版社

北　京

内 容 简 介

本书紧紧围绕数字信号的产生、变换、处理、逻辑运算、存储等内容进行讲解，具体内容包括：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、半导体存储器与可编程逻辑器件、脉冲波形的变换与产生、数/模与模/数转换器以及数字系统设计自动化 EDA 等。本书在编写上概念清晰，内容先进、实用，淡化内部电路的分析计算，突出实际电路的应用，引入数字系统设计自动化 EDA 技术，并给出了若干典型数字系统设计实例。为便于读者学习，每章都给出了习题，参考答案可在清华大学出版社网站下载。

本书既可作为高等学校电气信息、电子信息类各专业数字电子技术基础课程的教材，也可作为工程技术人员的参考书。

图书在版编目(CIP)数据

数字电子技术基础/谢志远主编. --北京：清华大学出版社，2014
(高等学校应用型特色规划教材)
ISBN 978-7-302-35715-5

Ⅰ. ①数…　Ⅱ. ①谢…　Ⅲ. ①数字电路—电子技术—高等学校—教材　Ⅳ. ①TN79

中国版本图书馆 CIP 数据核字(2014)第 060807 号

责任编辑：李春明　郑期彤
封面设计：杨玉兰
责任校对：李玉萍
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62791865
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：21.75　　**字　　数**：528 千字
版　　次：2014 年 6 月第 1 版　　**印　　次**：2014 年 6 月第 1 次印刷
印　　数：1～3000
定　　价：39.00 元

产品编号：038935-01

前　言

电子技术基础课程是一门工程性、实践性和应用性很强的课程，对于培养学生的工程实践能力和创新意识具有重要的意义。其中，数字技术是当前发展最为迅速的学科之一，在这一领域内知识更新的速度远远高于整个科技领域发展的平均速度。随着数字电子技术的不断发展，新技术、新器件、新电路不断涌现，传统的数字电子技术教材在内容安排上已经不能适应电子技术发展的要求。为此，华北电力大学电子学教研室根据数字电子技术发展趋势，在总结“电子技术基础系列课程”省级精品课程建设成果和经验的基础上，对以往教材内容进行了修改和更新，将启发式教学、研究性教学和优秀生培养思路及成果融入教材编写之中，使教材体系更加突出实际电路的应用。

本书紧紧围绕数字信号的产生、变换、处理、逻辑运算、存储等内容进行讲解。全书共 9 章，具体内容包括：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、半导体存储器与可编程逻辑器件、脉冲波形的变换与产生、数/模与模/数转换器以及数字系统设计自动化 EDA。

本书的特点如下。

(1) 在每章开头的“本章要点”中将本章内容以问题的形式提出，引导学生进行探究式学习，培养学生的自主学习能力和创新思维能力。在每章结尾的“本章小结”中对引言中的问题给予较精练的解答，提炼数字电子技术基础中的基本概念、基本原理和基本方法。

(2) 在借鉴国内数字电子技术基础教材优点的基础上，对传统教学内容进行了精选与整合，精简和优化了经典的数字电子技术基础知识，增加了 EDA 等现代数字电子技术的比重。具体表现为：在保证基础的前提下，更新课程内容，介绍当代先进的电子技术知识；进一步淡化内部电路的分析和计算，重点介绍典型电路的逻辑功能、外特性和应用；突出数字集成电路的应用和 EDA 技术。

(3) 独立地将数字系统设计自动化 EDA 技术引入书中，详细分析了硬件描述语言，给出了若干典型数字系统设计实例并在 EDA 平台上以及数字电路实验中给予充分验证，有助于读者系统掌握 EDA 技术设计方法。

(4) 将教学与科研紧密结合，充分发挥科研优势对本科教学的促进作用，适度将科研实践中的一些实用电路引入书中。

(5) 安排了适量的例题和习题，以便帮助学生更好地理解数字电子技术基础的基本概念、基本电路和基本方法。

(6) 编写体系新颖。为了方便读者学习，加深对内容的理解，在编写过程中引入了“注意”模块、“提示”模块以及“知识拓展”模块。

本书由华北电力大学电子学教研室共同编写，谢志远任主编，尚秋峰任副主编。参加编写的老师有尼俊红、胡正伟、刘童娜、黄怡然、刘立、段东兴等。

限于作者水平有限，书中难免存在许多缺点和不足，敬请使用本书的师生和读者批评指正。

编　者

目　录

第 1 章　数字逻辑基础

本章要点

- 常用的数制有哪几种？不同数制之间如何相互转换？
- 数字系统中数的表示形式有哪几种？如何对二进制数进行算术运算？
- 常用的数字编码有哪些？
- 基本逻辑运算有哪些？逻辑函数的概念是什么？
- 逻辑函数的化简方法有哪些？
- 逻辑函数的常用表示方法有哪些？

模拟电子技术的研究对象是在时间上或数值上都连续变化的电信号，研究的主要内容包括信号的产生、放大、运算、处理与变换等。数字电子技术的研究对象是在时间和数值上都不连续的数字信号(脉冲信号)，研究的主要内容包括数字逻辑基础、各种逻辑门电路、组合逻辑电路与时序逻辑电路的分析和设计、脉冲产生与变换、A/D 和 D/A 转换等。

1.1　几种常用的数制

数字电路是处理数字信号的电路，而数字信号通常用数码形式来表示，数码可以表示数量的大小。用一位数码去表示数量的大小往往是不够用的，因此经常需要用进位计数制的方法组成多位数码使用。数制就是多位数码中每一位的构成方法以及从低位到高位的进位规则，也就是通常所说的计数体制。下面介绍几种常用的数制。

数码、基数和位权称为数制的三要素。数码，就是通常所用到的数字或字符，如 1、2、3、4、…以及 A、B、C 等。基数是数制的根本，是指在进位计数制中，每个数位所允许使用的数码的个数。例如，十进制数的数码为 0～9 共十个不同的数字，因此十进制的基数是 10。同理，二进制数的数码只有 0 和 1，因此二进制的基数是 2。以此类推，八进制的基数是 8，十六进制的基数是 16。位权(或称权)，通常是指某个固定位置上的计数单位，它指出了一个数中每个数码所代表的量值，是由数码本身和该数码所处的位置共同决定的。以我们熟悉的十进制为例，数码 5 在百位数位置上表示 500，在小数点后第一位则表示 0.5。

在日常生活中，人们习惯于用十进制进行计数；而在数字电路中，通常使用二进制和十六进制，有时也使用八进制。任意 R 进制数可以表示为

$$(N)_R = \sum_{i=-\infty}^{+\infty} K_i \times R^i \tag{1.1}$$

式中，R 为基数；R^i 为第 i 位的位权；K_i 为基数 R 的第 i 次幂的系数。

1. 十进制

十进制(Decimal)是以 10 为基数的计数体制，是日常生活和工作中最常使用的进位计数制。其计数规律是“逢十进一”，有 0～9 十个数码，通常用 $(N)_\mathrm{D}$ 或 $(N)_{10}$ 表示十进制数。

任意十进制数可以表示为

$$(N)_D = \sum_{i=-\infty}^{+\infty} K_i \times 10^i \tag{1.2}$$

式中，10 为基数；10^i 为第 i 位的权；K_i 为基数 10 的第 i 次幂的系数，K_i 的取值为 0～9，共 10 个数码。例如：

$$526.79 = 5\times10^2 + 2\times10^1 + 6\times10^0 + 7\times10^{-1} + 9\times10^{-2}$$

式中，10^2、10^1、10^0、10^{-1} 和 10^{-2} 分别表示百位、十位、个位、小数点后第一位和小数点后第二位数码的权。

用数字电路来存储或处理十进制是不方便的，它要求电路中必须有 10 个完全不同的状态，这样会使电路变得非常复杂，因此在数字电路中不直接处理十进制数。

2. 二进制

构成数字电路的基本思路是把电路状态与数码对应起来。二进制(Binary)就是以 2 为基数的计数体制，只有 0 和 1 两个数码，可以分别用电路中两种不同的稳定状态来表示，因此广泛应用在数字电路中。二进制数的计数规律是“逢二进一”，通常用 $(N)_B$ 或 $(N)_2$ 来表示。

任意二进制数可以表示为

$$(N)_B = \sum_{i=-\infty}^{+\infty} K_i \times 2^i \tag{1.3}$$

式中，2 为基数；2^i 为第 i 位的权；K_i 为基数 2 的第 i 次幂的系数，K_i 的取值只有 0 和 1 两种可能。式(1.3)也可以用作二进制数到十进制数的转换公式。例如：

$$\begin{aligned}(1011.01)_B &= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} \\ &= (11.25)_D\end{aligned}$$

与十进制相比较，二进制具有以下优点。

(1) 二进制的数字装置简单可靠，所用元件少。二进制中只有两个数码 0 和 1 ，因此，BJT(双极结型晶体管)的饱和与截止、继电器触点的闭合与断开、灯泡的亮与不亮等，只要规定了其中一种状态为“0”，则另一种状态就可以用“1”来表示，这就给设计电路带来了很大的方便。

(2) 二进制数的基本运算规则简单，运算操作方便。

提示： 基于二进制的上述特点，人们习惯于用十进制数表示原始数据，在运算过程中将十进制数转换成数字系统可以接受的二进制数，在运算结束后，再将二进制数转换成十进制数来表示最终结果。

二进制只有 0 和 1 两个数码，因此用于表示数字时位数多。例如，十进制数 55 用二进制表示为 110111，不便于书写和表示，因此在数字计算机的资料中常采用十六进制或八进制。

3. 八进制

八进制(Octal)是以 8 为基数的计数体制。其计数规律是“逢八进一”，有 0～7 共八个数码，通常用 $(N)_O$ 或 $(N)_8$ 表示八进制数。

任意八进制数可以表示为

$$(N)_O = \sum_{i=-\infty}^{+\infty} K_i \times 8^i \tag{1.4}$$

式中，8 为基数；8^i 为第 i 位的权；K_i 为基数 8 的第 i 次幂的系数，K_i 的取值为 0～7，共 8 个数码。式(1.4)也可以用作八进制数到十进制数的转换公式。例如：

$$\begin{aligned}(35.67)_O &= 3\times8^1 + 5\times8^0 + 6\times8^{-1} + 7\times8^{-2}\\ &= (29.859375)_D\end{aligned}$$

4. 十六进制

十六进制(Hexadecimal)是以 16 为基数的计数体制。其计数规律是“逢十六进一”，除了 0～9 十个数码外，还有 A、B、C、D、E、F 六个数码，它们依次相当于十进制数中的 10、11、12、13、14、15。通常用 $(N)_H$ 或 $(N)_{16}$ 表示十六进制数。

任意十六进制数可以表示为

$$(N)_H = \sum_{i=-\infty}^{+\infty} K_i \times 16^i \tag{1.5}$$

式中，16 为基数；16^i 为第 i 位的权；K_i 为基数 16 的第 i 次幂的系数。式(1.5)也可以用作十六进制数到十进制数的转换公式。例如：

$$\begin{aligned}(1B.C8)_H &= 1\times16^1 + 11\times16^0 + 12\times16^{-1} + 8\times16^{-2}\\ &= (27.78125)_D\end{aligned}$$

在计算机中普遍采用 8 位、16 位和 32 位二进制并行计算，可以分别用 2 位、4 位和 8 位十六进制数表示，因而用十六进制书写程序非常方便。除此之外，十六进制还具有计数容量大的优点。例如，同样采用 4 位数码，二进制最多可计至 $(1111)_B = (15)_D$；八进制可计至 $(7777)_O = (2800)_D$；十进制可计至 $(9999)_D$；十六进制可计至 $(FFFF)_H = (65535)_D$，即 64K。

十进制数 0～15 与等值二进制、八进制、十六进制数的对照表如表 1.1 所示。

表 1.1 不同数制之间的对照

十 进 制	二 进 制	八 进 制	十六进制
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	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

1.2 数制之间的相互转换

同一个数能够用不同的计数体制来表示，根据实际需要在各种数制之间进行转换。例如，在数字计算机中，它的基本运算和操作可执行的代码都是以二进制为基础的，而日常使用时，人们更习惯于使用十进制，所以各种数据、操作命令等在进入计算机前必须转化成二进制代码，而处理完的结果一般使用十进制表示。同时为了书写方便，在一些资料中常采用十六进制或八进制表示。

任何一个 N 进制数都可以按式(1.1)转换成十进制数。先把 N 进制数按位权展开的方式写成多项式和的形式，若基数和位权已知，则多项式的和即为该 N 进制数所对应的十进制数，这种转换的方法称为多项式法，这种方法在前面已经介绍过，在此不再重复。

1.2.1 十进制数转换为二进制数

将十进制数转换为二进制数时，对整数部分和小数部分的处理方法不同，下面分别加以介绍。

假设十进制整数为 $(N)_D$，等值的二进制整数为 $(b_n b_{n-1} \cdots b_0)_B$，按照式(1.3)可写成

$$\begin{aligned}(N)_D &= b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_1 2^1 + b_0 2^0 \\ &= 2(b_n 2^{n-1} + b_{n-1} 2^{n-2} + \cdots + b_1) + b_0\end{aligned} \tag{1.6}$$

上式表明，若将 $(N)_D$ 除以 2，则得到的商为 $b_n 2^{n-1} + b_{n-1} 2^{n-2} + \cdots + b_1$，而余数即为 b_0。

同理，将式(1.6)除以 2 得到的商写成

$$b_n 2^{n-1} + b_{n-1} 2^{n-2} + \cdots + b_1 = 2(b_n 2^{n-2} + b_{n-1} 2^{n-3} + \cdots + b_2) + b_1 \tag{1.7}$$

由式(1.7)不难看出，若将 $(N)_D$ 除以 2 所得的商再次除以 2，则所得余数即为 b_1。

以此类推，反复将每次得到的商再除以 2，就能得到所对应的二进制整数，这种方法被称为“辗转相除法”，也就是将十进制整数连续不断地除以 2，直至商为 0，所得余数由低位到高位排列，即为所求的二进制整数。

【例 1.1】 将十进制数 $(141)_D$ 转换为二进制数。

解 1： 辗转相除法

2|141 ……………………余1…… b_0

2|70 ……………………余0…… b_1

2|35 ……………………余1…… b_2

2|17 ……………………余1…… b_3

2|8 ……………………余0…… b_4

2|4 ……………………余0…… b_5

2|2 ……………………余0…… b_6

2|1 ……………………余1…… b_7

由上可得 $(141)_D = (10001101)_B$。

这种方法虽然简单，但当十进制整数很大时，计算比较麻烦。为了简化计算过程，可以先将十进制整数和 2 的乘幂项对比。

解 2： $2^7=128$，$141-128=13=2^3+2^2+2^0$，由此可以写出对应的二进制数 $b_7=1$，$b_3=1$，$b_2=1$，$b_0=1$，其余各位权系数为 0，因此

$$(141)_D=(10001101)_B$$

注意： 在数字系统或计算机中只处理 4、8、16、32 或 64 位等二进制数据，因此数据的位数需配成规格化的位数。若要将二进制数 1100101 配成 8 位，则相应的高幂项补 0，其余不变，即

$$1100101=01100101$$

下面介绍将十进制小数转换为二进制小数的方法。

假设 $(N)_D$ 是一个十进制数的小数部分，其对应的二进制小数为 $(0.b_{-1}b_{-2}\cdots b_{-m})_B$，则根据式(1.3)可知

$$(N)_D=b_{-1}2^{-1}+b_{-2}2^{-2}+\cdots+b_{-m}2^{-m}$$

将上式两边分别乘以 2，得

$$2\times(N)_D=b_{-1}\times2^0+b_{-2}\times2^{-1}+\cdots+b_{-m}\times2^{-(m-1)} \tag{1.8}$$

将十进制小数乘以 2，所得乘积的整数即为 b_{-1}。因此，将十进制小数每次减去上一次所得积中的整数再乘以 2，直到满足误差要求为止，就可以完成由十进制小数到二进制小数的转换，这种方法称为“乘 2 取整法”。

注意： 十进制小数不一定能完全准确地转换成二进制小数，在这种情况下，可以根据精度要求只转换到小数点后某一位为止。

【例 1.2】 将十进制数 $(11.718)_D$ 转换为二进制数，要求精度达到 0.1%。

解： 由于转换精度要求达到 0.1%，即误差 $\varepsilon<0.001$，因此需要精确到二进制数小数点后 10 位，误差 $\varepsilon=\dfrac{1}{2^{10}}=\dfrac{1}{1024}<0.001$。

先对整数部分进行转换，得到 $(11)_D=(1011)_B$。

再对小数部分进行转换，将 $(0.718)_D$ 转换为二进制小数，保留小数点后 10 位有效数字，即

$0.718\times2=1.436$　　则 $b_{-1}=1$
$0.436\times2=0.872$　　则 $b_{-2}=0$
$0.872\times2=1.744$　　则 $b_{-3}=1$
$0.744\times2=1.488$　　则 $b_{-4}=1$
$0.488\times2=0.976$　　则 $b_{-5}=0$
$0.976\times2=1.952$　　则 $b_{-6}=1$
$0.952\times2=1.904$　　则 $b_{-7}=1$
$0.904\times2=1.808$　　则 $b_{-8}=1$
$0.808\times2=1.616$　　则 $b_{-9}=1$
$0.616\times2=1.232$　　则 $b_{-10}=1$

所以，$(11.718)_D=(1011.1011011111)_B$，转换误差 $\varepsilon<0.001$。

可以采用类似的方法，将十进制数转换为八进制数或十六进制数，在此不再赘述。

1.2.2 二进制数、十六进制数和八进制数之间的转换

使用二进制表示一个数所使用的位数要比十进制表示时所使用的位数长得多，书写不方便，不好读也不容易记忆。在计算机科学中，为了口读与书写方便，也经常采用十六进制或八进制表示，因为十六进制或八进制与二进制之间有着直接而方便的换算关系。

16 和 8 都是 2 的整数次幂，因此，每 4 位二进制数相当于 1 位十六进制数，每 3 位二进制数相当于 1 位八进制数。

1. 二进制数与十六进制数之间的转换

二进制数转换为十六进制数非常简单：以小数点为基准，将二进制数的整数部分从右到左每 4 位一组，不足 4 位的在高位补 0；小数部分从左到右每 4 位一组，不足 4 位的在低位补 0，然后写出每一组对应的 1 位十六进制数。

【例 1.3】将 $(101011.01101)_B$ 转换为十六进制数。

解： 将二进制数每 4 位分为一组，用相应的十六进制数代替，可得

$$(10\ 1011.0110\ 1000)_B = (2B.68)_H$$

【例 1.4】 将 $(C5.A)_H$ 转换为二进制数。

解： 将每位十六进制数用 4 位二进制数代替，可得

$$(C5.A)_H = (1100\ 0101.1010)_B$$

2. 二进制数与八进制数之间的转换

同理，二进制数转换为八进制数时，将二进制数每 3 位分为一组，每组对应 1 位八进制数即可。例 1.3 中的二进制数转换为八进制数为

$$(101\ 011.011\ 010)_B = (53.32)_O$$

将十进制数转换为十六进制数或八进制数时，也可以先将十进制数转换为二进制数，再进行相应的转换。

1.3 二进制数的算术运算

在数字电路中，0 和 1 既可以表示电路中两种对立的逻辑状态，也可以表示数量大小。在表示数量时，两个二进制数可以进行算术运算。二进制数的进位规则是“逢二进一”，其运算规则和十进制数的运算规则基本相同。

1.3.1 无符号二进制数的算术运算

二进制数的加、减、乘、除四种运算规则与十进制数类似，两者的区别在于进位或借位的规则不同。

1. 二进制加法

无符号二进制数的加法规则是

$$0+0=0,\ 0+1=1,\ 1+0=1,\ 1+1=\boxed{1}0$$

式中，方框中的 1 是进位，表示向高位进 1。

【例 1.5】计算两个二进制数 1001 和 0101 的和。

解:

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

因此，$1001+0101=1110$。

2. 二进制减法

无符号二进制数的减法规则是

$$0-0=0,\ 0-1=\boxed{1}1,\ 1-0=1,\ 1-1=0$$

式中，方框中的 1 是借位，表示 0 减 1 时不够减，向高位借 1。

【例 1.6】计算两个二进制数 1001 和 0101 的差。

解:

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

因此，$1001-0101=0100$。

在无符号减法中无法表示负数，因此要求被减数一定要大于减数。

3. 二进制乘法

【例 1.7】计算两个二进制数 1001 和 0101 的积。

解:

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

因此，$1001\times0101=101101$。

二进制数的乘法运算是由左移被乘数与加法运算组成的。

4. 二进制除法

【例 1.8】计算两个二进制数 1010 和 100 的商。

解：

$$\begin{array}{r} 10.1 \\ 100\overline{)1010} \\ \underline{100} \\ 010 \\ \underline{000} \\ 100 \\ \underline{100} \\ 0 \end{array}$$

因此，$1010 \div 100 = 10.1$。

二进制数的除法运算是由右移除数与减法运算组成的。

提示： 计算机进行数值计算时，往往把二进制数的减法运算操作转化为某种形式的加法运算操作，那么加、减、乘、除运算就全部可以用“移位”和“相加”两种操作来实现，从而使运算电路的结构大大简化。

1.3.2 带符号二进制数的减法运算

带符号数可以表示为：符号+数值。一个带符号二进制数的最高位就是符号位，其余部分为数值位。通常用“0”代表正数，用“1”代表负数。例如，$(+26)_D$ 的 8 位带符号二进制数为 $(00011010)_B$，而 $(-26)_D$ 的 8 位带符号二进制数为 $(10011010)_B$。在数字系统中，引入原码、反码、补码的概念，进行负数求补将减法运算变为加法运算，以达到简化电路的目的。

1. 带符号二进制数的原码、反码和补码表示

正数的原码、反码和补码三种形式完全相同。例如，$(+26)_D$ 的 8 位二进制原码为 $(00011010)_B$，其反码 $(N)_{INV}$ 为 00011010，补码 $(N)_{COMP}$ 为 00011010。

负数的反码 $(N)_{INV}$ 是通过将最高位符号位保留为 1，而其余所有数值位的原码逐位求反得到的，在电路中可以很容易地实现这一运算过程。

负数的补码 $(N)_{COMP}$ 为该负数的反码加 1，即

$$(N)_{COMP} = (N)_{INV} + 1 \tag{1.9}$$

例如，$(-26)_D$ 的 8 位二进制原码为 $(10011010)_B$，其反码 $(N)_{INV}$ 为 11100101，补码 $(N)_{COMP}$ 为 11100110。

注意： 在原码表示中，0 的原码有两种表达方式，即 00000000 和 10000000。也就是说，0 占用了两个编码。因此，8 位二进制数原码的表示范围为 $-127 \sim -0$ 和 $+0 \sim +127$，共 255 个数。

在反码表示中，0 的反码也有两种表达方式，即 00000000 和 11111111。因此，8 位带符号数反码的表示范围也为 $-127 \sim -0$ 和 $+0 \sim 127$，共 255 个数。

总之，在原码、反码中，n 位二进制数的表示范围为 $-2^{n-1}+1 \sim -0$ 和 $+0 \sim$

$2^{n-1}-1$。

在补码表示中，0 的补码只有一种表达方式，即 $(+0)_{COMP}=(-0)_{COMP}=$ 00000000，而 10000000 表示 −128。因此，8 位带符号数补码的表示范围为 −128 ～ 127，共 256 个数。

2. 二进制补码的运算

在数字系统中，带符号数一律用补码进行存储和计算。

【例 1.9】试用 4 位二进制补码计算 $6-2$。

解：

$$
\begin{aligned}
(6-2)_{COMP} &= (6)_{COMP}+(-2)_{COMP} \\
&= 0110+1110 \\
&= 0100
\end{aligned}
$$

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

注意： 进行二进制补码运算时，被加数的补码和加数的补码的位数要相等。两个用补码表示的数相加时，如果最高位(符号位)有进位，则进位被舍弃。因此，上例方框中的 1 应该舍弃，得 $6-2=4$。

【例 1.10】试用 4 位二进制补码计算 $6+7$。

解：

$$
\begin{aligned}
(6+7)_{COMP} &= (6)_{COMP}+(7)_{COMP} \\
&= 0110+0111 \\
&= 1101
\end{aligned}
$$

例 1.10 的求解结果1101 表示 −5，显然是错误的。这是因为，在 4 位二进制补码中，只有 3 位是数值位，它所表示的数值范围为 −8 ～ +7，本例的正确结果 13 已经超出了其数值表示范围，因此产生了溢出。解决溢出的方法是进行位扩展。

3. 溢出的判别

下面通过一道例题来说明溢出的判别问题。

【例 1.11】试用 4 位二进制补码分别计算 $4+2$、$(-6)+(-2)$、$2+6$、$(-4)+(-5)$。

解：

$$
\begin{array}{r}
+4 \\
+)\quad +2 \\
\hline
+6
\end{array}
\qquad
\begin{array}{r}
0\ 1\ 0\ 0 \\
+\ \ 0\ 0\ 1\ 0 \\
\hline
\boxed{0}\ 0\ 1\ 1\ 0
\end{array}
$$

(a)

$$
\begin{array}{r}
-6 \\
+)\quad -2 \\
\hline
-8
\end{array}
\qquad
\begin{array}{r}
1\ 0\ 1\ 0 \\
+\ \ 1\ 1\ 1\ 0 \\
\hline
\boxed{1}\ 1\ 0\ 0\ 0
\end{array}
$$

(b)

$$
\begin{array}{r}
+2 \\
+)\quad +6 \\
\hline
+8
\end{array}
\qquad
\begin{array}{r}
0\ 0\ 1\ 0 \\
+\ \ 0\ 1\ 1\ 0 \\
\hline
\boxed{0}\ 1\ 0\ 0\ 0
\end{array}
$$

(c)

$$
\begin{array}{r}
-4 \\
+)\quad -5 \\
\hline
-9
\end{array}
\qquad
\begin{array}{r}
1\ 1\ 0\ 0 \\
+\ \ 1\ 0\ 1\ 1 \\
\hline
\boxed{1}\ 0\ 1\ 1\ 1
\end{array}
$$

(d)

4 位二进制补码的表示范围为 −8 ～ +7，所以例 1.11 中，(a)、(b)无溢出；而(c)、(d)的运算结果应分别为 +8 和 −9，均超过了允许范围，产生溢出。

从例 1.11 可以看出，如果进位位与和数的符号位相反，就意味着运算结果是错误的，产生了溢出。

1.4 二进制代码

数字系统中的信息分为两类，一类是数值，另一类是文字符号(包括控制符)。因此，计算机中的二进制代码不仅可以表示数值的大小，还可以表示文字、符号(包括控制符)等信息。为了表示这些信息，往往用一定位数的二进制数码表示，此时这些数码并不代表数值的大小，而是用于区别不同的事务。这些特定的二进制数码称为代码。n 位二进制代码可以表示 2^n 个不同的信息。以一定的规则编制代码，用以表示十进制数值、字母、符号等不同信息的过程称为编码。将代码还原成所表示的十进制数值、字母、符号等的过程称为解码或者译码。

1.4.1 二-十进制码

用 4 位二进制代码表示 1 位十进制数中 0～9 这十个状态，简称 BCD(Binary-Coded Decimal)码。4 位二进制码共有 16 种代码，可以根据不同的规则，从中选出 10 种来表示 10 个十进制数码。下面介绍几种常用的 BCD 码，如表 1.2 所示。

表 1.2 几种常用的 BCD 码

十进制数	有 权 码			无 权 码	
	8421 码	2421 码	5421 码	余 3 码	余 3 循环码
0	0000	0000	0000	0011	0010
1	0001	0001	0001	0100	0110
2	0010	0010	0010	0101	0111
3	0011	0011	0011	0110	0101
4	0100	0100	0100	0111	0100
5	0101	1011	1000	1000	1100
6	0110	1100	1001	1001	1101
7	0111	1101	1010	1010	1111
8	1000	1110	1011	1011	1110
9	1001	1111	1100	1100	1010

8421BCD 码是最常用的一种 BCD 码，它使用 0～9 这十个数值的二进制码来表示相应的十进制数，其余 6 种组合是无效的。在这种编码方式中，每一位的值都是固定数，即每位都有位权。由于 b_3、b_2、b_1、b_0 的位权分别为 8、4、2、1，故称 8421BCD 码，它属于有权码。