数字电子技术基本教程

（修订版）

宋樟林　陈道铎　王小海　编著

浙江大学出版社

数字电子技术基本教程

（修订版）

宋樟林　陈道铎　王小海　编著

浙江大学出版社

内容简介

本书根据国家教委批准的《高等工业学校电子技术基础课程教学基本要求》，在第一版的基础上修改编写而成。

本书系统地阐述了数制和码制、逻辑代数及逻辑函数化简、MOS 和双极型数字集成逻辑电路、组合逻辑电路、集成触发器、时序逻辑电路、数/模和模/数转换、脉冲的产生和变形、存储器及可编逻辑器件等内容。

本书阐述清楚、理论联系实际、配以较多例题和习题，便于自学。在内容上适量增加了具有发展前途的新技术和新器件。本书可作为工科院校电气类、电子类、自动化类和计算机类等专业的“电子技术基础”课教材，也可作为有关专业工程技术人员的参考书。

数字电子技术基本教程

宋棹林 [redacted] 王小海 编著

责任编辑 龚建勋

* * *

浙江大学出版社出版

（杭州玉古路 20 号 邮政编码 310027）

浙江大学出版社电脑排版中心排版

德清第二印刷厂印刷

浙江省新华书店发行

* * *

787×1092 16开 17.25印张 442千字

1995年1月第2版 1996年4月第3次印刷

印数：2001－10000

ISBN 7-308-01578-5/TN·036 定价：17.00 元

第一版前言

本书是根据国家教委批准的《高等工业学校电子技术基础课程教学基本要求》和总结了组内多年来的教学经验、在几经试用、修改的基础上编写的。本书系统地介绍数字集成电路的分析设计基础、逻辑单元、典型电路和综合应用。本书在保证基本教学内容的前提下，增写了部分加宽加深的内容，均注有 * 号，对标有 * 的内容，可作教师选讲和读者自学参考。

当前，各种类型中、大规模集成电路，不仅应用在数字电子计算机中，而且在通讯、控制、测量仪表、医疗设备和家用电器等各个技术领域中的应用也日益广泛。为了适应科学技术和生产发展的需要，为了给读者学习数字电子计算机及其它各种数字系统打好基础，本书在小规模集成逻辑门电路和触发器的基础上，着重介绍中、大规模集成电路的工作原理、功能和应用。

本书一、二、三、四章由宋梓林编写，六、七章由陈道锋编写，五、八、九章由王小海编写。

本书承浙江大学计算机系张德馨教授仔细地审阅了全部书稿，并提出了许多宝贵的意见。浙江大学电子学教研室方伟、张圣训、许中元、郑元耀、刘保墉、吴燮华、邢建等同志参加了大纲的讨论、本教材的试用，并提出了具体的修改意见。本书编写过程中得到了国家教委电子技术课程指导小组成员邓汉馨教授的关心和指导。对此，一并表示衷心的感谢。

由于我们的水平有限，书中难免存在缺点和错误，恳请读者给予批评指正。

编者　1989 年 8 月

再版序

本书第一版出版至今已六年，这六年来电子技术及其应用有较大发展，教学也有发展，为了适应形势，对原书作了必要的修改和充实。这次修改是参照电子技术课程教学指导小组1992年8月制订的《高等工业学校电子技术基础教学基本要求(修订稿)》进行的。

通过这次修订，增强了CMOS电路的比重，加强了中、大规模集成电路应用，对PAL和GAL等可编逻辑器件作了初步介绍；对习题进行删、补，适度增加了习题。鉴于国家标准局要求全面推行电气制图及图形符号新国家标准，它与国际电工委员会IEC 617-12(1983)的图形符号一致，正在国内外逐渐普及，本书对二进制逻辑单元(数字集成电路)中的门电路和触发器的逻辑(图形)符号全部采用新国家标准；对中、大规模集成逻辑电路采用示例和简化框图。由于GB4728.12-85《电气图用图形符号 二进制逻辑单元》与旧标准差异很大，为了便于读者逐步熟悉和掌握，本书一方面在采用新国家标准逻辑符号的同时，也介绍了我国过去常用符号和美、日等国符号，作为过渡；另一方面，在分析和设计逻辑电路的有关章节中，结合电路实例，逐一给出逻辑符号；书中对逻辑状态、逻辑电平、逻辑约定等概念及定性符号等作简要讲解，并在附录中简单介绍逻辑符号的绘制原则。

本书参考学时为60学时，对标有*的内容，可作教师选讲和读者自学参考。本书安排在模拟电子技术课程之前或之后讲授均可。

本书第一、二、三、四章由宋樟林改编，第六、七章由陈道铎改编，第五、八章由王小海改编。

浙江大学电子学教研室数字电子技术教学组的老师们对本书修改提出了有益的意见，也得到原国家教委电子技术课程指导小组成员邓汉馨教授的关心和指导。谨对上述同志及对本书编写给予支持的同志表示衷心的感谢。

限于编者水平，缺点和错误在所难免，恳请读者批评指正。

编　者

1994年10月于杭州浙江大学

目　录

第八章　半导体存储器和可编程器件

第一章　数字电路基础

1.1　概述

一、数字量和模拟量

在我们周围存在许多物理量，其中一类物理量的变化在时间上和数量上是不连续的，是离散的，是以某最小量为单位成整数倍变化的，而小于这个最小数量单位的数值没有任何物理意义。我们将这一类物理量称为数字量。另一类物理量是随时间作连续变化的，如温度、速度、气压等。将连续变化的量称为模拟量。

二、数字电路

用以传递和加工处理数字信号形式的电路称为数字电路(Digital Circuit)。用以传递和处理模拟信号形式的电路称为模拟电路。在传递和处理信息过程中，模拟信号和数字信号可以转换。通常将模拟量和数字量互相转换的技术归属在数字技术(Digital Technigue) 领域。

三、数字电路的特点

在数字电路中，传递和处理信息时，采用二进制的“0” 和“1” 作为工作信号。“0” 和“1” 对应于电路中电压的“低” 和“高” 两种状态。数字电路中的半导体器件，在稳态时工作在饱和导通(“开”) 或截止(“关”) 状态。

相对于模拟电路而言，数字电路具有抗干扰能力强、精度高、通用性强、功耗较低、效率较高、便于长期保存信息和保密性好(经过加密处理) 等特点。正由于数字电路的许多特点，它已被人们广泛应用(如数字电子计算机、数字通讯、自动控制、数字仪表、导航、人造卫星、民用电器和医疗仪器等领域)，而且正在朝着更广泛的领域更迅猛地发展和普及。

1.2　数制和编码

在数字电路中广泛使用二进制(Binary) 数，但它有读起来困难、写起来很长的缺点，为了弥补这一缺点，常采用八进制(Octal) 数和十六进制(Hexadecimal) 数。本节从大家习惯的十进制(Decimal) 数开始，进而讨论这些进位计数制和不同数制之间的相互转换方法。在此基础上再介绍带符号数的代码表示法及几种数字电路的常用编码。

数字电子

1.2.1. 数制

数制(Number System)是计数进位的简称。例如,常用的十进制数,它由 0,1,……,9 等十个数码和一个小数点符号组成。在记数时,采用位值法则,就是对每一个数位赋以一定的位值——或称"权"(weight),相邻两位,高位比低位的位值大十倍。应用这种位值法则就能够用排列适当的数码来表示任意大小的数目,例如十进制数 794.5,写成

$$794.5 = \underset{(百)}{7 \times 10^2} + \underset{(拾)}{9 \times 10^1} + \underset{(个)}{4 \times 10^0} + \underset{(十分之一)}{5 \times 10^{-1}}$$

式中 10^2、10^1、10^0、10^{-1} 分别表示相应数位的"权"。十进制数的基数(Radix)为 10,计数规律是"逢十进一"。

因此,一个具有 n 位整数及 m 位小数的十进制数

$$(N)_{10} = (K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_{10}$$

可按权展开:

$$\begin{aligned}(N)_{10} &= K_{n-1} \times 10^{n-1} + \cdots + K_1 \times 10^1 + K_0 \times 10^0 + K_{-1} \times 10^{-1} + \cdots + K_{-m} \\ &\quad \times 10^{-m} \\ &= \sum_{i=-m}^{n-1} K_i \times 10^i \end{aligned} \tag{1.2.1}$$

式中系数 K_i 可取 0 ~ 9 十个数中的任何一个。等式左边的下标 10(有时写 D) 代表十进制数。

同理,对于任意的 r 进制数来说,它的基数为 r,由 r 个数码组成,计数规律是"逢 r 进一"。则一个 n 位整数及 m 位小数 r 进制数

$$(N)_r = (K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_r$$

可按权展开:

$$\begin{aligned}(N)_r &= K_{n-1} \times r^{n-1} + \cdots + K_1 \times r^1 + K_0 \times r^0 + K_{-1} \times r^{-1} + \cdots + K_{-m} \times \\ &\quad r^{-m} \\ &= \sum_{i=-m}^{n-1} K_i \times r^i \end{aligned} \tag{1.2.2}$$

式中系数 K_i 可取 0 ~ $(r-1)$ 之间的任意一个整数。

基数 $r = 2$,为二进制数,数码只有 0 和 1 两个,计数规律是"逢二进一";$r = 8$,为八进制数,有 0 ~ 7 八个数码,计数规律是"逢八进一";$r = 16$,为十六进制数,用 0 ~ 9 和 A ~ F 来表示,其中 $A = 10, B = 11, C = 12, D = 13, E = 14, F = 15$,计数规律是"逢十六进一"。

表 1.2.1 中列出了几种数制(r 为 10、2、8 和 16) 的若干个数、以资对照。

表 1.2.1 几种数制的对照表

$r=10$ $(N)_{10}$	$r=2$ $(N)_2$								$r=8$ $(N)_8$	$r=16$ $(N)_{16}$
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1	1
2	0	0	0	0	0	0	1	0	2	2
3	0	0	0	0	0	0	1	1	3	3
4	0	0	0	0	0	1	0	0	4	4
5	0	0	0	0	0	1	0	1	5	5
6	0	0	0	0	0	1	1	0	6	6
7	0	0	0	0	0	1	1	1	7	7
8	0	0	0	0	1	0	0	0	10	8
9	0	0	0	0	1	0	0	1	11	9
10	0	0	0	0	1	0	1	0	12	A
11	0	0	0	0	1	0	1	1	13	B
12	0	0	0	0	1	1	0	0	14	C
13	0	0	0	0	1	1	0	1	15	D
14	0	0	0	0	1	1	1	0	16	E
15	0	0	0	0	1	1	1	1	17	F
16	0	0	0	1	0	0	0	0	20	10
21	0	0	0	1	0	1	0	1	25	15
25	0	0	0	1	1	0	0	1	31	19
85	0	1	0	1	0	1	0	1	125	55
170	1	0	1	0	1	0	1	0	252	AA

1.2.2 各种进位制数的互相转换

一、非十进制数转换成十进制数

应用式(1.2.2)，可以把 r 进制数转换成等值的十进制数，现举例说明。

[例 1.2.1] 二进制数$(110.011)_2$，表示的等值十进制数是

$$\begin{aligned}(\underbrace{110}_{\text{整数}}.\underbrace{011}_{\text{小数}})_2 &= (\underbrace{1\times 2^2+1\times 2^1+0\times 2^0}_{\text{整数}} \\ &\quad + \underbrace{0\times 2^{-1}+1\times 2^{-2}+1\times 2^{-3}}_{\text{小数}})_{10} \\ &= (6.375)_{10}\end{aligned}$$

式中的下标 2(有时写 B) 代表二进制。

[例 1.2.2] 写出八进制数$(17.3)_8$，按权展开后的等值十进制数是

$$(17.3)_8 = (\underbrace{1\times 8^1+7\times 8^0}_{\text{整数}}+\underbrace{3\times 8^{-1}}_{\text{小数}})_{10} = (15.375)_{10}$$

式中的下标 8(有时写 0) 代表八进制数。

[例 1.2.3] 写出十六制数$(CD.E)_{16}$，按权展开后的等值十进制数是

$$(CD.E)_{16} = C\times 16^1 + D\times 16^0 + E\times 16^{-1}$$

$$= 12 \times 16^1 + 13 \times 16^0 + 14 \times 16^{-1}$$
$$= 192 + 13 + 0.875$$
$$= (205.875)_{10}$$

式中的下标 16(有时写 H) 代表十六进制数。

由上可见,非十进制数转换成等值十进制数的基本方法是按权展开再相加的方法。

二、十进制数转换成非十进制数

1. 十进制数转换成二进制数

十进制数转换成二进制数时,需将待转换数的整数部分和小数部分,分别加以转换。

(1) 整数部分

若要将十进制整数$(N)_{10}$,转换成等值二进制数$(K_{n-1}K_{n-2}\cdots\cdots K_1K_0)_2$,则可写成下列等式。

$$(N)_{10} = K_{n-1} \times 2^{n-1} + K_{n-2} \times 2^{n-2} + \cdots + K_1 \times 2^1 + K_0 \times 2^0$$
$$= 2 \times (K_{n-1} \times 2^{n-2} + K_{n-2} \times 2^{n-3} + \cdots + K_2 \times 2^1 + K_1) + K_0$$

将上式两边均除以 2,可得商为$(K_{n-1} \times 2^{n-2} + K_{n-2} \times 2^{n-3} + \cdots + K_2 \times 2^1 + K_1)$,余数为 K_0。将商再除以 2,得新余数 K_1,依次类推,继续用 2 除,直到最后的商为零,就可以分别求出 K_0、K_1,K_2,……,K_{n-1}。

由上分析可知,其转换方法,就是将十进制整数逐次除以 2,并依次记下余数,直到其商得 0 为止。第一次余数表示最低位(LSB),最后的余数表示最高位(MSB)。把全部余数按最高位到最低位次序排列,就得到等值的二进制数。

[**例** 1.2.4] 将十进制数$(19)_{10}$ 转换为二进制数。

解

			余数	
2	19			
2	9	…………	1	(LSB)
2	4	…………	1	
2	2	…………	0	
2	1	…………	0	
	0	…………	1	(MSB)

余数 $(19)_{10}=(10011)_2$

上述将十进制整数转换成二进制整数的方法,称为"除 2 取余"法。

(2) 小数部分

若要将十进制小数$(N)_{10}$,转换成等值二进制数$(0.K_{-1}K_{-2}\cdots\cdots K_{-m})_2$,则可写成下列等式

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

将上式两边均乘以 2,则得

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

上式整数部分为 K_{-1}，小数部分为$(K_{-2}\times 2^{-1}+K_{-3}\times 2^{-2}+\cdots+K_{-m}\times 2^{-m+1})$，将小数部分再乘以 2，得新整数 K_{-2}，如此继续下去，可以分别求出 $K_{-3}, K_{-4}, \cdots\cdots, K_{-m}$。

由此可见，其转换方法，就是将十进制小数逐次乘以 2(每次只将小数部分乘以 2)，并每次记下整数，直到积的小数部分出现全 0 为止。第一次得到的“积的整数”为最高位，而最后一次得到的“积的整数”为最低位。把全部整数按次序排列，就得到等值的二进制数。

[**例** 1.2.5] 将十进制小数$(0.375)_{10}$转换成二进制小数。

$$\begin{array}{lrr}
 & & 0.375\\
 & \times & 2\\
\hline
\text{(小数MSB)} & & \boxed{0}.750\\
 & \times & 2\\
\hline
 & & \boxed{1}.500\\
 & \times & 2\\
\hline
\text{(小数LSB)} & & \boxed{1}.\underbrace{000}_{\text{全零}}
\end{array}$$

结果$(0.375)_{10}=(0.011)_2$

上述将十进制小数转换成二进制小数的方法，称为“乘 2 取整”法。

有时乘 2 取整积的小数部分永不等于零，就产生转换误差，此时可根据所需精度取二进制小数的位数。

由上可见，若把$(19.375)_{10}$转换成二进制数，其结果是

$$(19.375)_{10}=(10011.011)_2$$

2. 十进制数转换成八进制数

转换方法是将十进制数的整数部分“除 8 取余”，小数部分“乘 8 取整”。

[**例** 1.2.6] 把十进制数$(379.375)_{10}$转换成八进制数。

解

整数部分

$$\begin{array}{r|l l}
 & & \text{余数}\\
8 & 379 & \\
\hline
8 & 47 & 3\\
\hline
8 & 5 & 7\\
\hline
 & 0 & 5
\end{array}\quad \uparrow\ \text{读数方向}$$

小数部分

$$\begin{array}{rr}
 & 0.375\\
\times & 8\\
\hline
 & \boxed{3}.000
\end{array}$$

结果$(379.375)_{10}=(573.3)_8$

至于十进制数转换成十六制数，可用“除 16 取余”及“乘 16 取整”法。例如，

$$(7141.25)_{10}=(1BE5.4)_{16}$$

三、八进制数、十六进制数和二进制数的相互转换

由于 $2^3=8$，因此，每位八进制数可以用三位二进制数来表示，例如，$(0)_8=(000)_2,\cdots\cdots,(7)_8=(111)_2$。同样，由于 $2^4=16$，故可用四位二进制数来表示一位十六进制数，例如，$(0)_{16}=(0000)_2,\cdots\cdots,(F)_{16}=(1111)_2$。因而，二进制、八进制和十六进制之间可直接进行转换。例如

八进制	3	5	7	·	1	2	4	6
二进制	011	101	111	·	001	010	100	110
十六进制	E		F	·	2	A		6

即 $(357.1246)_8=(11101111.00101010011)_2=(EF.2A6)_{16}$

由上可见，二进制转换成八进制（或十六进制）时，二进制数的整数部分从低位开始，每三位（或四位）分为一组，若最左边一组不足三位（或四位），可在左边添 0 补足。二进制小数部分从小数点向右每三位（或四位）分为一组，最后不足三位（或四位），可在右边添 0 补足。然后，将每组二进制数转换为八进制（或十六进制）数。采用上述的逆过程，可将八进制（或十六进制）转换成二进制。八进制和十六进制之间转换，可用二进制作桥梁。

1.2.3　带符号数的代码表示

通常，在数值（绝对值）左面加上符号“+”或“-”，以表示数的正或负，例如负数 9，表示“-9”。但机器只能识别二进制数，因此也只能用二进制数来表示“+”或“-”，习惯上以“0”表示正数符号，以“1”表示负数符号。将符号位放在数值的左边，就成为一个带有符号位的数，例如

$$N_1=+1001(\text{或 }1001)\qquad\qquad N_2=-1001$$

在机器中分别表示为：

$$\underset{\text{符号}}{\boxed{0}}\underbrace{\boxed{1001}}_{\text{数值部分}}\qquad\qquad\underset{\text{符号}}{\boxed{1}}\underbrace{\boxed{1001}}_{\text{数值部分}}$$

这种将数值部分及符号部分统一用代码表示的带符号数称为机器数，如 N_1、N_2 的机器数表示形式就是 01001，11001。而把原来的数值形式称为机器数的真值，如 N_1、N_2 的真值表示形式就是 +1001，-1001。

在数字电路中，机器数表示方法最常用的有原码、反码和补码三种*

一、原码(True form)

原码表示方法是将带符号位用数码 0 表示正号，用 1 表示负号，而对数值位不作任何改变，仍然采用原来的二进制数表示。例如，两个带符号的二进制数 $N_1=+1001101$ 及 $N_2=-1001101$ 的原码表示形式为：

$[N_1]_{原}=01001101 \qquad [N_2]_{原}=11001101$

原码表示法简单易懂，且真值转换也较方便。但是原码的加、减运算较复杂、例如，两数相加时，如果同号，则数值相加，符号不变；如果异号，就要进行减法，而在相减时，需先比较两数绝对值的大小，然后从绝对值较大的数中减去绝对值较小的数，差值的符号与绝对值较大的数的符号一致。在机器中，为了判断同号还是异号，比较数的绝对值大小等，就要增加机器中的设备和降低运算速度。为了解决这一问题，采用补码表示法。

二、反码**(One's complement)

对于正数，反码和原码相同；对于负数，反码表示法，就是将原码符号位右边的二进制数值按位取反，即"1"变"0"及"0"变"1"，而符号位仍为 1。

设两个带符号的二进制数分别为 N_1 和 N_2，其真值形式为：

$N_1=+1001101 \qquad N_2=-1001101$

则 N_1 和 N_2 的反码表示形式为：

$[N_1]_{反}=01001101 \qquad [N_2]_{反}=10110010$

三、补码(Two's complement)

在补码表示法中，正数的表示跟原码和反码的表示相同。负数的补码，可从原码转换而来。转换规则为：符号位仍为 1，数值部分按位取反，并在最低有效位上加 1。例如

$N_1=+1001101 \qquad N_2=-1001101$

则 N_1 和 N_2 的补码表示形式为：

$[N_1]补=01001101 \qquad [N_2]_{补}=10110011$

由于补码加、减运算的规则为：

---

* 设真值 N 为整数($-2^n<N<+2^n$)，则原码、反码及补码的定义如下：

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

例如 $N=-111$，则 $[N]_{原}=2^3-N=1000-(-111)$
$=1111$

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

例如：$N=-111$，则 $[N]_{反}=(2^{3+1}-1)+N$
$=(10000-1)+(-111)=1000$

3. 补码：$[N]_{补}=\begin{cases} N & 当\ 0\leqslant N<2^n \\ 2^{n+1}+N & 当\ -2^n<N\leqslant 0 \end{cases}$

例如：$N=-111$，则 $[N]_{补}=2^{3+1}+N$
$=10000+(-111)=1001$

** 有的文献将反码称为对 1 的补码。

$$[N_1+N_2]_{补}=[N_1]_{补}+[N_2]_{补}$$

故有 $[N_1-N_2]_{补}=[N_1+(-N_2)_{补}]$

$$=[N_1]_{补}+[-N_2]_{补}$$

还有，当已知$[N]_{补}$时，可用$\left([N]_{补}\right)_{补}=[N]_{原}$规则。

应用二进制补码，可使减法运算变成加法运算，举例如下：

[例 1.2.7] 已知 $N_1=+0000101$，$N_2=+0000010$，计算 N_1-N_2 之值。

解 $[N_1]_{补}=00000101$，$[-N_2]_{补}=11111110$，故计算如下：

$$\begin{array}{rcl}
[N_1]_{补} & = & 00000101 \\
+[-N_2]_{补} & = & 11111110 \\
\hline
[N_1-N_2]_{补} & = & 100000011 \longrightarrow \text{结果为正值的补码}
\end{array}$$

符号为正

此1被丢掉

则得

$$N_1-N_2=+0000011$$

[例 1.2.8] 已知 $N_1=+0000101$，$N_2=+0001100$，计算 N_1-N_2 之值。

解 $[N_1]_{补}=00000101$，$[-N_2]_{补}=11110100$，故计算如下：

$$\begin{array}{rcl}
[N_1]_{补} & = & 00000101 \\
+[-N_2]_{补} & = & 11110100 \\
\hline
[N_1-N_2]_{补} & = & 11111001 \longrightarrow \text{结果为负值的补码}
\end{array}$$

符号为负

则得

$$N_1-N_2=-0000111$$

三位二进制正、负数表示方法列于表 1.2.2。表中二进制数码各列的左边第一位是符号位。由表可知，当二进制数为正值时，原码、反码和补码的表示都无区别，而当数为负值时，这三种码的表示就完全不一样。

这里再说明一下：引入补码之后，在数字电子系统(Digital Electronic System)中的减法运算可用加法实现。两个补码表示的带符号位的二进制数相加时，其相加结果为和的补码。运算结果若符号位有进位产生，则应丢掉此进位，才能得到正确结果。(一般在电路中是会自动丢掉的，例如在 8 位字长的计算机中，由于模为 2^8，所以大于等于 2^8 的数就表示不出来)。