21世纪高等院校规划教材

数字电子技术（第2版）

SHUZI DIANZI JISHU

◎ 主编 康晓明

国防工业出版社
National Defense Industry Press

21世纪高等院校规划教材

数字电子技术

（第2版）

主　编　康晓明
副主编　王荣莉　丁建石　杨国庆　黄　玮
参　编　唐广芝　李　莉　夏春茂　安海霞

国防工业出版社
·北京·

内容简介

本书共分9章，详细地介绍了数字逻辑基础、逻辑门电路、组合逻辑电路、触发器、时序逻辑电路、脉冲波形的产生和整形、半导体存储器、数/模和模/数转换电路、基于EDA技术的数字系统设计等内容。本书内容丰富，叙述清楚，概念明确，例题围绕重点，便于理解和学习，每章后附有精选的典型习题，可供练习和提高之用。

本书可作为高等学校及成人教育学校相关专业的教材，也可供工程人员参阅。

图书在版编目(CIP)数据

数字电子技术/康晓明主编. —2版. —北京：国防工业出版社，2007.1

ISBN 7-118-03760-5

Ⅰ.数… Ⅱ.康… Ⅲ.数字电路－电子技术

Ⅳ.TN79

中国版本图书馆CIP数据核字(2006)第115241号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路23号 邮政编码100044)

天利华印刷装订有限公司印刷

新华书店经售

*

开本 787×1092 1/16 **印张** 14¾ **字数** 335千字

2007年1月第2版第1次印刷 **印数** 1—4000册 **定价** 25.00元

(本书如有印装错误，我社负责调换)

国防书店：(010)68428422 发行邮购：(010)68414474

发行传真：(010)68411535 发行业务：(010)68472764

前　言

“数字电子技术”是一门重要的专业技术基础课程。本书是在第1版的基础上修订而成的。初版书一经出版，受到广大读者的欢迎和好评，并已被多所院校选为指定教材。本书是根据“电子技术基础课程教学基本要求”并在认真研讨和论证教学体系的基础上，组织长期讲授“数字电子技术”课的教师针对当前高等院校教育教学的特点并结合学生的实际情况而编写的。

本书对初版书“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、脉冲波形的产生和整形、半导体存储器、数/模和模/数转换电路”章节的内容做了进一步完善，并根据当前数字电路的发展需要，增添了“基于EDA技术的数字系统逻辑设计与VHDL描述，MAX+plus Ⅱ软件的基本操作”等内容。

本书在编写过程中力求简明扼要、深入浅出、结合实际、便于自学且注重对学生分析能力和解决问题的能力的培养。

本书由康晓明副教授担任主编。康巨珍教授，刘锡安教授担任主审。

参加本书编写的有：王荣莉编写了第1、6章，李莉编写了第2章，唐广芝编写了第3章，康晓明编写了第4、5、9章，黄玮编写了第7章，杨国庆编写了第8章和附录。丁建石参加了部分章节的编写和修改。夏春茂、安海霞为本书编写了大量习题。全书由康晓明统稿。

李春华副教授对本书的编写给予了极大的支持，在此也表示衷心感谢！

另外，本书在编写过程中还得到了我所在学校教务处领导的支持以及广大同行及学生的支持和帮助，在此也表示由衷感谢！

最后，特别感谢国防工业出版社刘炯编辑在本书出版过程中给予的大力支持。

由于编者水平所限，书中不妥之处恳请批评与指正。

编　者

2006年9月

目　录

第 1 章　数字逻辑基础

1.1　概　述

1.1.1　数字电路及其特点

1. 数字电路

电子电路中的信号可分为两大类:一类是时间的连续函数,称为模拟信号,模拟信号来自自然界客观存在的一些物理量,如速度、压力、温度、声音等;另一类是时间和幅度都是离散的数字信号,如电子表的秒信号、生产上记录零件个数的计数信号等。这些信号的变化发生在一系列离散的瞬间,其值也是离散的。这种数字信号只有两个离散值,常用数字 0 和 1 来表示。注意,这里的 0 和 1 没有大小之分,只代表两种对立的状态,称为逻辑 0 和逻辑 1,也称为二值数字逻辑。处理数字信号的电路称为数字电路。

2. 数字电路的特点

数字电路的工作信号是不连续变化的数字信号,所以在数字电路中工作的半导体三极管、二极管、MOS 管及由它们构成的集成电路多数工作在开关状态,即工作在饱和区和截止区,而放大区只是其过渡状态。

数字电路的主要研究对象是电路的输入和输出之间的逻辑关系,因此数字电路又可称做数字逻辑电路。它的主要分析工具是逻辑代数,表达电路的功能主要是真值表、逻辑表达式等。

1.1.2　数的进制

多位数码中,每位的构成方法以及从低位到高位的进位规则称为数制。

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。

(1) 基数

它是计数制中用到的数码个数,例如在十进制中,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。

(2) 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这位的一个固定常数,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。

1. 十进制(Decimal)

十进制的基数 $R=10$,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

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

$$K_{-1}10^{-1}+\cdots+K_{-m}10^{-m})=\sum_{i=-m}^{n-1}K_i 10^i$$

式中,K_i 为 0~9 中任一数字,i 可为 $-\infty$ 到 $+\infty$ 之间的任意整数。

例如:十进制数 328.213 可展开为

$$(328.213)_{10}=3\times10^2+2\times10^1+8\times10^0+2\times10^{-1}+1\times10^{-2}+3\times10^{-3}$$

十进制数 9 用二进制数表示为 1001,即表示 1 位十进制数需要 4 位二进制数。

十进制虽然是人们最习惯的计数制,却很难用电路来实现。因为要使一个电路或一个电子器件具有能严格区分的 10 个状态来与十进制的 10 个不同的数字符号一一对应,是比较困难的,因此在计数电路中一般不直接使用十进制。

2. 二进制(Binary)

二进制的基数 $R=2$,第 i 位的位权是 2^i,有两个数字符号 0 和 1(二进制中的 1 一般读作"幺")。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

$$(N)_2=(K_{n-1}2^{n-1}+K_{n-2}2^{n-2}+\cdots+K_1 2^1+K_0 2^0+$$

$$K_{-1}2^{-1}+\cdots+K_{-m}2^{-m})=\sum_{i=-m}^{n-1}K_i 2^i$$

式中,K_i 为 0 或者 1,i 可为 $-\infty$ 到 $+\infty$ 之间的任意整数。

例如:二进制数 1101.101 可展开为

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

二进制具有一定的优点,因此在计算技术中被广泛采用。

(1) 二进制数只有两个数字符号 0 和 1,因此很容易用电路元件的状态来表示。例如,三极管的截止和饱和、继电器的接通和断开、灯泡的亮和灭、电平的高和低等,都可以将其中一个状态规定为 0,另一个状态规定为 1,用来表示二进制数。这种表示简单可靠,所用元件少,存储和传送二进制数也十分可靠。

(2) 二进制的基本运算规则与十进制运算规则相似,但要简单得多。例如两个一位十进数相乘,其规律要用"九九乘法表"才能表示,而两个一位二进制数相乘,只有 4 种组合。因此用电路来实现二进制运算十分方便可靠。

但同样表示一个数,二进制数要比十进制数位数多。例如,2 位的十进制数 87 变为二进制数为 1010111,需 7 位。

因此,用数字系统运算时,通常先将人们习惯的十进制原始数据转换成二进制数。运算结束后,再转换成人们所易接受的十进制数。

由于用二进制表示一个数的位数很多,不便于书写和记忆。因此,在数字系统中,通常采用十六进制作为二进制的缩写。

3. 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 的多项式表示为

$$(N)_{16} = (K_{n-1}16^{n-1} + K_{n-2}16^{n-2} + \cdots + K_1 16^1 + K_0 16^0 +$$

$$K_{-1}16^{-1} + \cdots + K_{-m}16^{-m} = \sum_{i=-m}^{n-1} K_i 16^i$$

式中，K_i 为 0～9 和 $A \sim F$ 数符中任一数字，i 为 $-\infty$ 到 $+\infty$ 之间的任意整数。

例如：$(4CD)_{16} = 4\times16^2 + 12\times16^1 + 13\times16^0$

十六进制数很容易转换为二进制数，因为十六进制的数可用 4 位二进制来表示：

$$(4CD)_{16} = (0100,1100,1101)_2$$

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，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：

$$(1101011.101)_2$$

$$0110,1011.1010$$

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

$$6 \quad B \quad A$$

即：$(1101011.101)_2 = (6B.A)_{16}$

由于目前在微型计算机中普遍采用 8 位、16 位和 32 位二进制并行运算，而 8 位、16 位和 32 位的二进制数可以用 2 位、4 位和 8 位的十六进制数表示，因而用十六进制符号书写程序十分简便，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。常用数制对照表如表 1.1 所示。

表 1.1　数制对照表

对照内容 / 数制 / 特点	十进制(Decimal)	二进制(Binary)	十六进制(Hexadecimal)
数字符号	0、1、2、3、4、5、6、7、8、9	0、1	0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F
计数规律	逢十进一	逢二进一	逢十六进一
基数	10	2	16
位权	10^i	2^i	16^i
表达式	$(N)_D = \sum_{i=-\infty}^{\infty} K_i \times 10^i$	$(N)_B = \sum_{i=-\infty}^{\infty} K_i \times 2^i$	$(N)_H = \sum_{i=-\infty}^{\infty} K_i \times 16^i$

4．二进制与十进制之间的相互转换

(1) 二进制转换为十进制

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：

$$(1010.101)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} =$$

$$8 + 0 + 2 + 0 + 0.5 + 0 + 0.125 = (10.625)_{10}$$

为了方便地把二进制转换为十进制，应熟记二进制数各位的权。表 1.2 列出了常用

的二进制位权。

表 1.2　二进制各位的权

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	0	2	4	8	16	32	64	128	256	512	1024

(2) 十进制转换成二进制

任何十进制数的整数部分可用辗转相除法转换成二进制数,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

$$(K_2K_1K_0)_2 = (K_2 \times 2^2 + K_1 \times 2^1 + K_0 \times 2^0)_{10} =$$

$$[2(K_2 \times 2^1 + K_1) + K_0]_{10}$$

$$(N)_{10} \div 2 = K_2 \times 2 + K_1 \quad \cdots\cdots \text{余数为 } K_0$$

用商再除以 2 得　　$(K_2 \times 2 + K_1) \div 2 = K_2$　　……余数为 K_1

不断用前次的商除以 2,直到最后的商为零。

$$K_2 \div 2 = 0 \quad \cdots\cdots \text{余数为 } K_2$$

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

[例 1.1]　将十进制数 13 转换为二进制数。

解:将十进制数 13 除以 2 取其余数。

	余数	
2 | 13	1	k_0
2 | 6	0	k_1
2 | 3	1	k_2
2 | 1	1	k_3
0		

所以　　$(13)_{10} = (k_3k_2k_1k_0) = (1101)_2$

对于任何十进制数的净小数部分可用乘 2 取整法求相应的二进制数。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

$$2 \times (N)_{10} = K_{-1} \times 2^0 + K_{-2} \times 2^{-1} + \cdots + K_{-m} \times 2^{-(m-1)} = K_{-1} + N_1$$

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

$$2 \times N_1 = K_{-2} \times 2^{-0} + K_{-3} \times 2^{-1} + \cdots + K_{-m} \times 2^{-(m-1)}$$

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_{-i},直至求得 K_{-m}。

上述计算过程中,如还未求到 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;如求到 K_{-m}时,小数部分仍不为零,这表示转换有误差,可按“四舍五入”的

原则取舍。

[例 1.2] 将纯小数$(0.625)_{10}$转换为二进制数。

解：

		取整	
$0.625\times2=1.250$	……	1	k_{-1}
$0.250\times2=0.500$	……	0	k_{-2}
$0.50\times2=1.00$	……	1	k_{-3}

所以 $(0.625)_{10}=(0.k_{-1}k_{-2}k_{-3})_2=0.101$

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 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 1.3] 将十进制数 11.625 转化为二进制数。

解：可由 $(11)_{10}=(1011)_2$

$(0.625)_{10}=(0.101)_2$

合成 $(11.625)_{10}=(1011.101)_2$

任意进制与十进制之间的转换原理及方法，同二进制与十进制之间的转换原理和方法相类似，不再重复。

任意两种进制之间的转换，一般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。

1.1.3 二进制代码

由于数字系统是以二值数字逻辑为基础的，因此数字系统中的信息（包括数值、文字、控制命令等）都是用一定位数的二进制码表示的，这个二进制码称为代码。

1. 带符号的二进制数的代码

(1) 真值

它是指在数值前面用“+”号表示正数，“-”号表示负数的带符号二进制数。

(2) 机器数

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。

例如：真值 $X_1=+0.1101$； $X_2=-0.1101$

机器数 $X_1=\ 0.1101$； $X_2=\ 1.1101$

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 $X=\pm0.X_{-1}X_{-2}\cdots X_{-m}$

当 $X>0$ 时， $X=+0.X_{-1}X_{-2}\cdots X_{-m}$ $(X)_{原}=0.X_{-1}X_{-2}\cdots X_{-m}$

当 $X<0$ 时， $X=-0.X_{-1}X_{-2}\cdots X_{-m}$ $(X)_{原}=1.X_{-1}X_{-2}\cdots X_{-m}=$

$$1-(-0.X_{-1}X_{-2}\cdots X_{-m})$$

例如：$X_1=+0.1101$

则 $(X_1)_{原}=0.1101$

$X_2=-0.1101$

$$(X_1)_{原}=1-(-0.1101)=1.1101$$

零的原码有两种表示形式：

$$(+0)_{原}=0.00\cdots0$$
$$(-0)_{原}=1.00\cdots0$$

因此，小数原码定义为

$$(X)_{原}=\begin{cases}X & 0\leqslant X<1\\ 1-X & -1<X\leqslant 0\end{cases}$$

(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 $(X)_{原}=OX_{n-1}X_{n-2}\cdots X_0$

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：$X=-1101$ $(X)_{原}=11101$

所以，整数源码定义为

$$(X)_{原}=\begin{cases}X & 0\leqslant X<2^n\\ 2^n-X & -2^n<X\leqslant 0\end{cases}$$

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减法运算，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，这就增加了运算时间。

2) 反码(One's complment)

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 $X=\pm X_{n-1}X_{n-2}\cdots X_0$

则整数的反码定义为

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

例如：$X_1=+1101$， $X_2=-1101$

则 $[X_1]_{反}=01101$

$$[X_2]_{反}=(2^5-1)+X_2=$$

$$(100000-000001)+(-1101)=$$
$$11111-1101=10010$$

最高位是符号位,后四位是数值部分,这刚好是原码数值部分按位求反。

(2) 小数的反码

若 $$X=0X_{n1}X_{n2}\cdots X_m$$

则小数的反码定义为

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

例如: $X_1=+0.1101$ $X_2=-0.1101$

则 $$[X_1]_{反}=0.1101$$

(3) 零的反码

有两种形式:

$$[X_2]_{反}=2-2^{-4}+(-0.1101)=$$
$$10.0000-0.0001-0.1101=1.0010$$

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

作反码加减法时,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 作反码加

$$[X_1]_{反}+[X_2]_{反}=01101+11010=\text{“1”}00111$$

则 $$[X_1+X_2]_{反}=00111+1=01000$$

这与 $$X_1+X_2=1101-0101=+1000$$
$$[X_1+X_2]_{反}=01000$$

是一致的。

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,就是二进制数值本身。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 $$X=\pm X_{n-1}X_{n-2}\cdots X_0$$

则整数的补码定义为

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

例如：$X_1 = +1101$，　　则$[X_1]_{补} = 01101$

　　$X_2 = -1101$，　　则$[X_2]_{补} = 10011$

(2) 小数的补码

若　　$X = .X_{n1}X_{n2}\cdots X_m$

则小数的补码定义为

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

(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：实现 $X_1 - X_2$ 的运算。

若　　$X_1 = 1101$，　　$X_2 = 0101$

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

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

丢去最高位“1”　　$[X_1 - X_2]_{补} = 01000$

真值：　　$X_1 - X_2 = 1000$

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 1.3 给出了几个典型数的真值、原码、反码和补码表示。

表 1.3　真值、原码、反码和补码

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. 十进制数的常用代码

十进制数除了转换成二进制数以外，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，便于传送和处理。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，它们编码方式很多。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，4 位权之和为所表示的十进制数；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为 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