数字系统逻辑设计

王维华 曲兆瑞 编著

山东大学出版社
Shandong University Press

责任编辑
刘旭东

封面设计
牛钧

版式设计
赵岩

ISBN 7-5607-2456-6

9 787560 724560 >

定价：21.00元

数字系统逻辑设计

王维华　曲兆瑞　编著

山东大学出版社

图书在版编目(CIP)数据

数字系统逻辑设计/王维华,曲兆瑞编著.—济南:山东大学出版社,2003.1
ISBN 7-5607-2456-6

Ⅰ.数… Ⅱ.①王…②曲… Ⅲ.数字系统—逻辑设计 Ⅳ.TP302.2

中国版本图书馆 CIP 数据核字(2003)第 000468 号

山东大学出版社出版发行
(山东省济南市山大南路 27 号 邮政编码:250100)
山东省新华书店经销
安丘市艺中印务有限责任公司印刷
787×1092 毫米 1/16 14 印张 326 千字
2003 年 1 月第 1 版 2003 年 1 月第 1 次印刷
印数:1—3000 册
定价:21.00 元

内 容 简 介

本书系统地讲述了数字逻辑和数字系统的基本概念和理论，着重介绍了数字系统分析和设计的基本方法。主要内容有数制和编码，逻辑代数基础知识，组合逻辑电路和时序逻辑电路的分析方法和设计方法，中大规模集成电路的应用等。

本书结构合理，叙述清楚，方法实用，题例丰富，还附有部分习题答案。本书可作为高等院校计算机、通信工程、电子工程、自动控制等专业的教材，也可作为相关专业科技人员的参考书。

前　言

我们正生活在信息化、数字化的时代。数码相机、智能冰箱、数控系统、数字通讯、国际互联网……各种数字化信息化的新技术、新产品层出不穷。作为数字系统的杰出代表，计算机更是广泛地应用于科学和工程计算、过程和实时控制、辅助设计和辅助制造、数据采集和信息处理、办公自动化等各个领域。数字技术极大地改变了我们的生产和生活方式。

在科学技术高速发展的今天，数字电子技术也在飞速发展中，新器件、新技术、新方法不断涌现。在 20 世纪 70 年代，数字系统的设计主要利用逻辑代数、状态分析等方法作出原理电路图，再选用适当的通用型集成电路，进行连接，构成系统。随着大规模可编程逻辑器件（PLD）的出现，硬件描述语言（HDL）及电子设计自动化（EDA）技术的发展，数字系统的设计进入了“编制”专用集成电路（ASIC）的阶段。设计人员使用硬件描述语言编写程序，由 DEA 工具编译生成“配置数据”，写入可编程逻辑器件即可。数字系统的全部或大部可以“编制”在一片或数片可编程逻辑器件中。这些新技术新器件的应用给数字系统的设计带来了很大的变化，使得设计人员可以更为快速高效地设计出高性能的数字电子产品。但是这些新技术的开发和应用都离不开数字逻辑的基础知识。

数字逻辑是数字技术的基础，是电子信息、通讯工程、自动控制等专业的主要技术基础课程之一。对于从事数字系统的研制、开发和应用的科学工作者来说，熟悉和掌握数字逻辑的理论和方法是十分重要的。

本书根据对数字技术领域专业技术人员的知识结构的要求，以及技术进步、课程体系的总体考虑，按照注重基础知识和基本方法，注重培养能力的要求，系统地讲述了数字逻辑和数字系统的基本概念和理论，着重介绍了数字系统分析和设计的基本方法。主要内容有数制和编码，逻辑代数基础知识，组合逻辑电路和时序逻辑电路的分析方法和设计方法，中大规模集成电路的应用等。本书结构合理，叙述清楚，方法实用，题例丰富，还附有部分习题答案，便于读者学习和掌握。

在本书的编写过程中得到了山东大学计算机学院殷晓峰、任守华、李铁军等教师的多方协助，刘荣兴教授审阅了本书的全稿并提出了许多宝贵意见。另外，本书的出版还得到了山东大学出版基金的资助。在此一并表示感谢。

编　者

2002.9

目　录

第一章　数据信息的二进制表示

数字系统是对数字量进行加工、处理的系统。目前数字系统中广泛采用二进制信号（也称二值量、开关量），即每个信号只可有两种状态（常用 0 和 1 表示）。这种二进制信号很容易和电路状态对应，且技术实现方便。一位二进制信号只能表示两种状态，多位二进制信号组合可表示多种状态。任何信息都必须表示成二进制形式才能用数字系统处理。本章讨论数字、字符等常用信息的二进制表示。

1.1　进位记数制

1.1.1　进位记数制的基本概念

进位记数制是指按一定的规则和符号表示数量的方法，其基本要素是数码、位权和运算规则。我们熟悉的十进制数是以 10 为基数的记数制，所用的数码是 0，1，…，9，各数位的位权是 10 的整次幂，运算规则是“逢十进一”、“借一当十”。例如十进制数 368.25 从左边（高位）开始各位的位权依次为 10^2，10^1，10^0，10^{-1}，10^{-2}，这个十进制数所表示的数量可以写成

$$(368.25)_{10} = 3\times10^2 + 6\times10^1 + 8\times10^0 + 2\times10^{-1} + 5\times10^{-2}$$

一般十进制数 $N = d_n d_{n-1}\cdots d_0 \cdot d_{-1}\cdots d_{-m}$ 所表示的数量可写作

$$(N)_{10} = d_n\times10^n + d_{n-1}\times10^{n-1} + \cdots + d_0\times10^0 + d_1\times10^{-1} + \cdots + d_{-m}\times10^{-m}$$

$$= \sum_{i=-m}^{n} d_i \times 10^i \qquad (1.1-1)$$

上式称为位权展开式，其中 $d_i \in \{0, 1, \cdots, 9\}$，m 和 n 都是正整数。

由十进制记数制可以推知一般的 R 进制记数制就是以 R 为基数的记数制，采用的数码应是 0，1，…，R-1，各数位的位权应是 R 的整次幂，运算规则应是“逢 R 进一”，其位权展开式为

$$(N)_R = (q_n q_{n-1}\cdots q_0 \cdot q_{-1}\cdots q_{-m})_R = \sum_{i=-m}^{n} q_i \times R^i \qquad (1.1-2)$$

其中 $q_i \in \{0, 1, \cdots, R-1\}$，m 和 n 都是正整数。

1.1.2 数字系统中的常用数制

数字系统中普遍采用二进制，而在人工读写时常用八进制和十六进制，为便于人机交换也使用十进制，但都是以二进制为基础的。

1. 二进制记数制

二进制（Binary）记数制是以 2 为基数的记数制，所用的数码为 0 和 1，各数位的位权是 2 的整次幂，运算规则是“逢二进一”，其位权展开式为

$$(N)_2 = (q_n q_{n-1} \cdots q_0 . q_{-1} \cdots q_{-m})_2 = \sum_{i=-m}^{n} q_i \times 2^i \qquad (1.1-3)$$

其中 $q_i \in \{0, 1\}$，m 和 n 都是正整数。

例如二进制数 11010.01 所表示的数量可以写作

$$(11010.01)_2 = 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2}$$
$$= 2^4 + 2^3 + 2^1 + 2^{-2}$$

二进制数的运算过程如下

	(被加数)	1	0	1	0	1	1	0	1		(被减数)	1	0	1	0	1	1	0	1
+	(加数)	0	0	1	1	1	0	0	1	−	(减数)	0	0	1	1	1	0	0	1
	(进位)	0	1	1	1	0	0	1			(借位)	1	1	1	0	0	0	0	
	(和)	1	1	1	0	0	1	1	0		(差)	0	1	1	1	0	1	0	0

由 1111 1111 + 1 = 10000 0000 = 2^8 可知，8 位二进制无符号正整数所能表示的数值范围为 0 ~ 255。n 位二进制无符号正整数所能表示的数值范围为 0 ~ 2^n -1。

2. 八进制记数制

八进制（Octal）记数制是以 8 为基数的记数制，所用的数码为 0，1，…，7，各数位的位权是 8 的整次幂，运算规则是“逢八进一”，其位权展开式为

$$(N)_8 = (q_n q_{n-1} \cdots q_0 . q_{-1} \cdots q_{-m})_8 = \sum_{i=-m}^{n} q_i \times 8^i \qquad (1.1-4)$$

其中 $q_i \in \{0, 1, \cdots, 7\}$，m 和 n 都是正整数。

例如八进制数 350.2 所表示的数量可以写作

$$(350.2)_8 = 3\times 8^2 + 5\times 8^1 + 0\times 8^0 + 2\times 8^{-1}$$

八进制数的运算过程如下

	(被加数)	2	5	5		(被减数)	2	5	5
+)	(加数)		7	3	-)	(减数)		7	3
	(进位)	1	1			(借位)	1	0	
	(和)	3	5	0		(差)	1	6	2

n 位八进制无符号正整数所能表示的数值范围为 0 ~ 8^n-1。

3. 十六进制记数制

十六进制记数制是以 16 为基数的记数制，所用的数码为 0，1，…，9 和 A，B，C，D，E，F，其中 A，B，C，D，E，F 所表示的数量分别是 10，11，12，13，14，15，各数位的位权是 16 的整次幂，运算规则是“逢十六进一”，其位权展开式为

$$(N)_{16} = (q_n q_{n-1} \cdots q_0 . q_{-1} \cdots q_{-m})_{16} = \sum_{i=-m}^{n} q_i \times 16^i \qquad (1.1-5)$$

其中 $q_i \in \{0, 1, \cdots, 9, A, B, C, D, E, F\}$，m 和 n 是正整数。

例如十六进制数 E8.4 所表示的数量可以写作

$$(E8.4)_{16} = E \times 16^1 + 8 \times 16^0 + 4 \times 16^{-1} = 14 \times 16^1 + 8 \times 16^0 + 4 \times 16^{-1}$$

十六进制数的运算过程如下

	(被加数)	A	9		(被减数)	A	9
+)	(加数)	3	F	-)	(减数)	3	F
	(进位)	1			(借位)	1	
	(和)	E	8		(差)	6	A

由 $16^n - 1 = 100 \cdots 0 - 1 = FF \cdots F$ 可知，n 位十六进制无符号正整数所能表示的数值范围为 0 ~ $16^n - 1$。

4. 不同数制的标识

在同时使用不同数制的场合,为避免混淆常使用数字下标或字母后缀对不同数制加以标识，如下所示

数　制	下标标识	后缀标识
二进制(Binary)	$(n)_2$	nB
八进制(Octal)	$(n)_8$	nQ
十进制(Decimal)	$(n)_{10}$	nD
十六进制(Hexadecimal)	$(n)_{16}$	nH

1.1.3 不同数制的相互转换

1. 任意进制数转为十进制数

任意进制数转为十进制数一般采用按权相加法，就是按十进制数的运算规则计算其位权展开式，就可得到相应的十进制数。例如

$$(11101000.01)_2 = 2^7 + 2^6 + 2^5 + 2^3 + 2^{-2} = (232.25)_{10}$$

$$(350.2)_8 = 3 \times 8^2 + 5 \times 8^1 + 2 \times 8^{-1} = (232.25)_{10}$$

$$(E8.4)_{16} = 14 \times 16^1 + 8 \times 16^0 + 4 \times 16^{-1} = (232.25)_{10}$$

2. 十进制数转为其他进制数

要把十进制数转为其他进制数，需对其整数部分和小数部分分别进行不同的处理。

十进制整数的转换可采用除基取余法。就是用所要转成数制的基数连续除十进制整数，直到商为 0，将所得一系列余数按先得为低位，后得为高位的顺序排列起来即为所要转换数制的数。例如

求 $(232)_{10} = (?)_2$　　$(74)_{10} = (?)_2$

$(232)_{10} = (?)_8$　　$(232)_{10} = (?)_{16}$

转换计算过程（除基取余）如下

2	232	余数
2	116 …	0
2	58 …	0
2	29 …	0
2	14 …	1
2	7 …	0
2	3 …	1
2	1 …	1
	0 …	1

低位 ↑ 高位

2	74	余数
2	37 …	0
2	18 …	1
2	9 …	0
2	4 …	1
2	2 …	0
2	1 …	0
	0 …	1

低位 ↑ 高位

即 $(232)_{10} = (11101000)_2$　　$(74)_{10} = (1001010)_2$

8	232	余数
8	29 …	0
8	3 …	5
	0 …	3

16	232	余数
16	14 …	8
	0 …	E

即 $(232)_{10} = (350)_8$　　$(232)_{10} = (E8)_{16}$

十进制小数的转换可采用乘基取整法。就是用要转成数制的基数连续乘十进制小数，直到十进制小数为 0 或达到所要求的转换位数，将所得一系列整数按先得为高位，后得为低位的顺序排列起来即为所要转换数制的小数。注意十进制小数转换成其他进制数仍然是小数。例如

求　$(0.25)_{10}=(?)_2$

$(0.6875)_{10}=(?)_2$

$(0.3125)_{10}=(?)_8$

$(0.25)_{10}=(?)_{16}$

转换计算过程（乘基取整）如下

整数	小数
	25 × 2
0	5 × 2
1	0

整数	小数
	6875 × 2
1	375 × 2
0	75 × 2
1	5 × 2
1	0

即　$(0.25)_{10}=(0.01)_2$　　$(0.6875)_{10}=(0.1011)_2$

整数	小数
	3125 × 8
2	5 × 8
4	0

整数	小数
	25 × 16
4	0

即　$(0.3125)_{10}=(0.24)_8$　　$(0.25)_{10}=(0.4)_{16}$

一般来说连续乘 2、8 或 16 难以使十进制小数达到 0，这时可按“精度相当”原则确定转换位数。例如一位十进制小数表示数的精度为十分之一，转为二进制小数要得到相应精度可取三位或四位(精度为八分之一或十六分之一)；两位十进制小数表示数的精度为百分之一，相应的二进制小数可取六位或七位(精度为 64 分之一或 128 分之一)。

对既有整数又有小数的十进制数，转换时需分别处理。例如

$$(232.25)_{10}=(232)_{10}+(0.25)_{10}=(11101000)_2+(0.01)_2$$
$$=(11101000.01)_2$$

3. 二进制数和八进制数、十六进制数的相互转换

由于一位八进制数对应三位二进制数(见表 1.1-1)，一位十六进制数对应四位二进制数(见表 1.1-2)，二进制数和八进制数、十六进制数的相互转换非常方便。

表 1.1-1 八进制数和二进制数的对应关系

八进制	0	1	2	3	4	5	6	7
二进制	000	001	010	011	100	101	110	111

表 1.1-2 十六进制数和二进制数的对应关系

十六进制	0	1	2	3	4	5	6	7
二进制	0000	0001	0010	0011	0100	0101	0110	0111
十六进制	8	9	A	B	C	D	E	F
二进制	1000	1001	1010	1011	1100	1101	1110	1111

要把八进制数转为二进制数，只需将每位八进制数写成相应的三位二进制数即可；要把十六进制数转为二进制数，只需将每位十六进制数写成相应的四位二进制数即可。

例如 求 $(56.73)_8=(?)_2$ ， $(6A.D3)_{16}=(?)_2$

则 $(56.73)_8=(101\ 110.111\ 011)_2$

$(6A.D3)_{16}=(0110\ 1010.1101\ 0011)_2$

把二进制数转为八进制数或十六进制数，需要以小数点为基准进行分组，转为八进制数时每三位一组，转为十六进制数时每四位一组，位数不足时需用 0 补位（补位应不改变原二进制数的数值，即整数在高位补 0，小数在低位补 0）；按原顺序写出每组二进制数所对应的八进制数或十六进制数即可。

例如 求 $(1011101.01011)_2=(?)_8$ ， $(1011101.01111)_2=(?)_{16}$

则 $(\underline{00}1\ 011\ 101.010\ 11\underline{0})_2=(135.26)_8$

$(\underline{0}101\ 1101.0111\ 1\underline{000})_2=(5D.78)_{16}$

4. 八进制数和十六进制数的相互转换

八进制数和十六进制数的相互转换可通过二进制数实现。如要将一个八进制数转为十六进制数，可先将其转为二进制数，再将此二进制数转为十六进制数。

例如 $(45.6)_8=(\underline{00}10\ 0101.110\underline{0})_2=(25.C)_{16}$

$(7A9.D)_{16}=(011\ 110\ 101\ 001.110\ 1\underline{00})_2=(3651.64)_8$

1.2　带符号的二进制数的表示

数字系统中所表示的数可能是正数，也可能是负数；可以是纯整数，称为定点整数；也可以是纯小数，称为定点小数；也可以既有整数，也有小数，称为浮点数。浮点数将在 1.3 节讨论,本节讨论定点数。

1.2.1　真值和机器数

带符号的数是用正负号加绝对值表示的数。真值是指带正负号的二进制数，是带符号的二进制数的书写形式。在数字系统中“数”是用寄存器、存储器等硬件设备来表示的，其位数是固定的，即无效 0 也要表示，如无符号二进制数 11011，在 8 位字长的机器中表示为 00011011（8 位二进制数）。数的正负号通常是用位于数码最高位（最左端）的一个二进制位表示，这个二进制位叫作符号位，一般是 0 表示正，1 表示负。这种用一个二进制位表示数的正负号的方法称为符号数码化。机器数是符号数码化的定长二进制数，是带正负号的二进制数在机器中的表示形式。

例如+4 和-4 写成二进制数(真值)分别为+100 和-100，表示成 8 位字长的原码(机器数)分别为 00000100 和 10000100，两机器数的最高位是符号位，0 表示“+”，1 表示“-”。

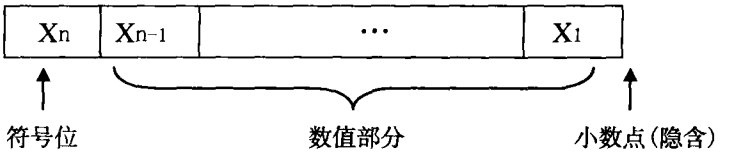
注意：由于机器数的位数是固定的，当实际数的有效数位不足时需要用 0 补位。

常用的机器数有原码，补码和反码。

1.2.2　原码(True from)

原码又叫符号-数值(sign-magnitude)表示法，是用一个二进制位(称为符号位)表示数的正负号，其余各位(称为数值部分)表示数的绝对值。

n 位定点整数原码形式为



n 位定点整数原码的定义式为 (符号位的位权为 2^{n-1})

$$[x]_{原}=\begin{cases} x, & 0\leqslant x\leqslant(2^{n-1}-1) \\ 2^{n-1}+|x|, & -(2^{n-1}-1)\leqslant x\leqslant 0 \end{cases} \qquad (1.2-1)$$

式中$[x]_{原}$ 表示真值 x 的原码。

当 x 为正数时$[x]_{原}=x$，即正数的原码的符号位为 0，数值部分等于真值(若真值的

有效数位不足时需用 0 补位)；当 x 为负数时$[x]_{原}=2^{n-1}+|x|$，即负数的原码的符号位为 1(符号位的位权为2^{n-1})，数值部分等于真值的绝对值(需补位定长)。

按定义可以写出 n 位定点整数原码所能表示的数值如下：

正 数		负 数	
真值	原码	真值	原码
+1	00 … 01	−1	10 … 01
…	…	…	…
$+(2^{n-1}-1)$	01 … 11	$-(2^{n-1}-1)$	11 … 11

可以看出，n 位定点整数原码所能表示的数值范围为 $-(2^{n-1}-1)\sim+(2^{n-1}-1)$。

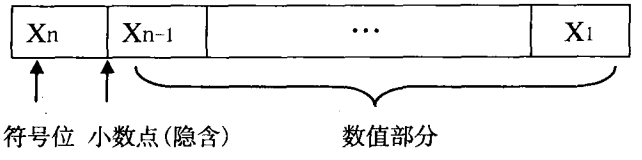
原码和真值的转换可以按原码的符号位对应真值的正负号，原码的数值部分等于真值的绝对值(需补位定长)来完成。

例如 若 x = 10110，y = −10110，按 8 位字长

则 $[x]_{原}=00010110$，$[y]_{原}=10010110$

0 的原码有两种形式 $[+0]_{原}=00\cdots0$，$[-0]_{原}=10\cdots0$

n 位定点小数原码形式为



n 位定点小数原码的定义式为 (符号位的位权为2^0)

$$[x]_{原}=\begin{cases}x, & 0\leqslant x\leqslant 1-2^{-(n-1)}\\ 1+|x|, & -(1-2^{-(n-1)})\leqslant x\leqslant 0\end{cases} \qquad (1.2)-2$$

机器表示定点小数时小数点是隐含的，并不表示出来。但我们在书写时为明确起见常把小数点写出来。

例如 若有 x = 0.1101， y = −0.1101 按 8 位字长

则 $[x]_{原}=0.1101000$， $[y]_{原}=1.1101000$

原码具有符号和数值分离的特点，形式直观易于理解。但运算不便，例如对两个原码表示的数做相加运算，需先检查两数的符号位，若同号则两数的数值部分相加；若异号则需将两数的数值部分相减。数字系统较少采用原码。

1.2.3 补码(Two's complement)

补码是按模定义的一种表示带符号的二进制数的方法，由于补码运算方便，在数字系统中得到了广泛应用。

1. 补码的定义

用来记数的设备所能表示的数总有一个限度，或者说有一定的容量，这个容量我们就称之为模。例如指针式钟表的小时记数，在表盘上有 12 个刻度，可表示 12 种状态或记 12 个数，我们就称其为以 12 为模。对钟表的小时记数来说，12 和 0 的指示相同；另外如果指针原指在 2 的刻度上，将其顺时针方向拨过(加)12 个刻度，指针仍然指在 2 的刻度上。上述情况可表示为

$$12 = 0 \quad (\text{mod } 12)$$
$$2 + 12 = 2 \quad (\text{mod } 12)$$

其中 mod 12 表示以 12 为模。

一般说来若记数设备以 m 为模，则有

$$m = 0 \quad (\text{mod } m)$$
$$a + km = a \quad (\text{mod } m) \qquad (1.2-3)$$

其中 k 是整数。

对钟表来说，如果将指针由 2 的位置逆时针拨过(减)9 个刻度，则指针指向 5；而将指针由 2 的位置顺时针拨过(加)3 个刻度，指针也指向 5。即

$$2 - 9 = 2 + 3 = 5 \quad (\text{mod } 12)$$

按式(1.2-3)，上式也可写成

$$2 - 9 = 2 + (12 - 9) = 2 + 3 \quad (\text{mod } 12)$$

我们称数与模的和为该数对模的补数。如－9 对模 12 的补数为 3，－5 对模 12 的补数为 7，+3 对模 12 的补数为 3。这样，减法操作可由加负数对模的补数来实现。

n 位二进制整数共有 2^n 种状态，可表示 2^n 个数(0 ~ 2^n-1)，而 0 和 2^n 的 n 位二进制表示是相同的，

	X_n	X_{n-1}	···	X_1	数值
	0	0	···	0	0
1	0	0	···	0	2^n

即 n 位二进制整数是以 2^n 为模。对 n 位二进制整数有

$$2^n = 0 \quad (\text{mod } 2^n)$$
$$a + k \times 2^n = a \quad (\text{mod } 2^n) \qquad (1.2-4)$$

其中 k 是整数。