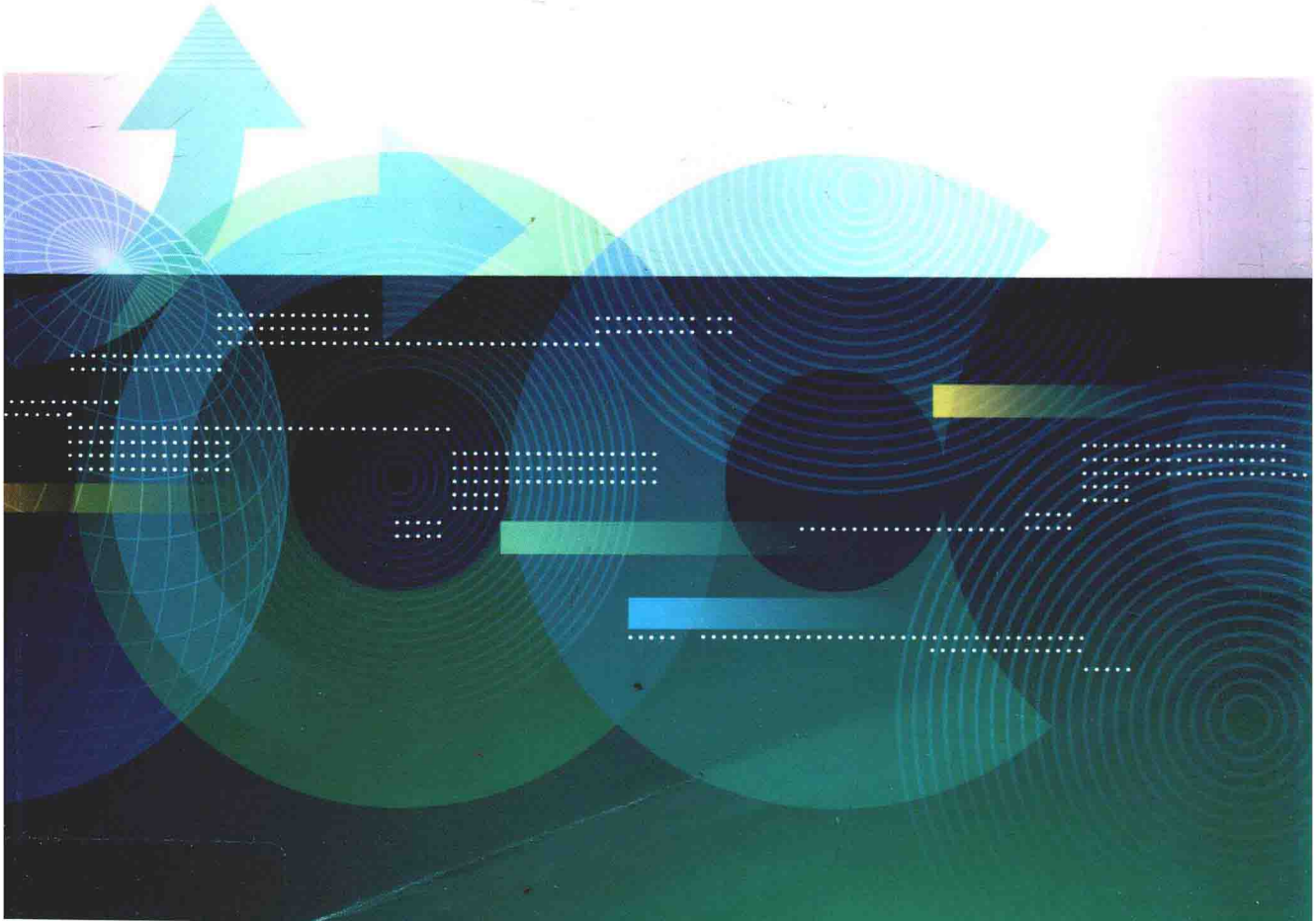
普通高等教育“十二五”规划教材
电子电气基础课程规划教材

数字电子技术

■ 江小安　杨润玲　编著



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY [http://www.phei.com.cn]

普通高等教育“十二五”规划教材
电子电气基础课程规划教材

数字电子技术

江小安　杨润玲　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书共 7 章，内容包括逻辑代数基础、集成逻辑门电路、组合逻辑电路、时序逻辑电路、脉冲波形的产生和变换、半导体存储器和可编程逻辑器件等。各章最后一节为 Multisim 仿真示例。

本教材适应面较宽，可作为高等工科院校有关专业的本科生、高职高专学生及自考生的教材，也可供电子技术领域的工程技术人员学习参考。

图书在版编目（CIP）数据

数字电子技术 / 江小安，杨润玲编著. —北京：电子工业出版社, 2015.2
电子电气基础课程规划教材
ISBN 978-7-121-25265-5

Ⅰ. ①数… Ⅱ. ①江… ②杨… 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 303381 号

责任编辑：韩同平　　特约编辑：李佩乾
印　　刷：北京季蜂印刷有限公司
装　　订：北京季蜂印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　印张：12　字数：360 千字
版　　次：2015 年 2 月第 1 版
印　　次：2015 年 2 月第 1 次印刷
印　　数：2 500 册　　定价：35.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

数字电子技术不仅是电类专业重要的专业基础课，也是相关专业学习后续课程的基础，与此同时也是一门很重要的技能基础课程。因此该课程的学习是十分重要的。

本教材在内容上倾向于集成电路的原理及应用，如中规模集成组合电路、集成触发器、集成计数器及其应用等，而较大地削减了用门电路实现的内容。本书淡化理论推导，强化实际应用；淡化内部原理和电路结构，强化外部特性和功能描述；淡化文字内容，强化实例讲解。

每章最后都安排了一节 Multisim 仿真示例，每个例题都是与本章内容紧密相关的器件的应用实例，仿真的结果也有利于学生进一步掌握该课程的系统知识，并激发学生的学习兴趣。

本教材中标注有“*”的部分是建议选讲的内容，正常教学中略去这些内容并不影响理论体系的完整性和内容的连贯性。

本教材适合作为高等院校相关专业“数字电子技术”课程的教材，也可供自学考试、夜大、函大和远程教学相关专业的学生选用。

由于作者水平有限，书中一定存在错误和不妥之处，恳请广大学者批评指正，有关意见可以发至作者电子邮箱 yrl@xauat.edu.cn。

作　者

目　录

第 1 章　逻辑代数基础

各种数字设备，不能对人们熟悉的十进制数所直接接受，而只能对以二进制形式表示的数或代码进行运算和处理。因此为实现人机交互，必须掌握各种进制的数、各种代码的特点及相互之间的转换方法。

“逻辑学”主要研究推理判断的规律和方法。“逻辑代数”是用数学语言来描述逻辑思维的一门科学。虽然逻辑代数和普通代数在书写形式上有相同之处，但是含义截然不同。逻辑代数中的“量”和“值”均无大小的含义。

本章主要讲述各种数制和常用代码及相互之间的转换方法；各种逻辑运算及常用的逻辑门；逻辑代数的基本规则、基本公式和常用公式；逻辑关系的表示形式及逻辑函数的化简方法。

1.1　数制与码制

1.1.1　数制

用数码表示数量的大小时，仅仅用一位数是不够的，因此常采用多位数。多位数码的构成及从低位向高位的进位规则称为进位计数制，简称数制。几种常用的数制是十进制、二进制、八进制和十六进制等。

1．十进制

在十进制数中，每个数位规定使用 0～9 这 10 个数码，故其基数是 10。大于 9 的数就需要用两位以上的多位数来表示，其计数规则是“逢十进一”的进位计数关系。

多位数中不同位置上的 1 所表征的数值大小称为这一位的权值。若 i 是各数位的序号，则按照这个方法来确定：以小数点为中心，整数部分，自右向左依次为第 $0, 1, 2, \cdots, m-1$ 位；小数部分，自左向右依次为第 $-1, -2, -3, \cdots, -n$ 位，m 和 n 分别为整数部分和小数部分的位数。第 i 位的权值是 10^i，因此，一个多位数所表示的数值等于每一位上的数码与该位权值的乘积之后的累加和。例如

$$292.16 = 2\times 10^2 + 9\times 10^1 + 2\times 10^0 + 1\times 10^{-1} + 6\times 10^{-2}$$

将上述关系式写成通式，也就是“按权展开式”，则任意一个十进制数 $(N)_{\mathrm{D}}$ 均可表示为

$$(N)_{\mathrm{D}} = \sum_{i=-n}^{m-1} k_i \times 10^i$$

其中 k_i 是第 i 位的数码，下脚标 D(Decimal)表示括号内的数 N 是十进制数，有时也用 10 表示。

同理，任意一个 r 进制数，都可以写成“按权展开”式，并可求出等值的相应十进制数为

$$(N)_r = \sum_{i=-n}^{m-1} k_i \times r^i$$

其中 r 是该数制的基数，r^i 是第 i 位的权值。

在数字电路中，一般都不直接采用十进制，因为要用 10 个不同的电路状态来表示十进制的 10 个数码既困难又不经济。

2．二进制

在二进制数中，每个数位规定使用 0 和 1 这两个数码，故其基数是 2。其计数规则是“逢二进一”。二进制数用下脚标 B(Binary)或者 2 来表示。把一个二进制数“按权展开”就能得到它所对应的十进制数。例如

$$\begin{aligned}(110.101)_B &= 1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}\\ &=4+2+0+0.5+0+0.125\\ &=(6.625)_D\end{aligned}$$

由于二进制比较简单，只有 0 和 1 两个数码，所以在数字电路中很容易实现。例如三极管的饱和和截止，灯泡的亮和灭，继电器开关的闭合和断开等。只要规定其中一种状态为 1，另一种状态为 0，就可用来表示二进制数。

二进制也有缺点，例如用它来表示一个数时，位数多，书写长。而且在人机交互中，还需解决二进制和十进制的换算问题。

3．八进制

在八进制数中，每个数位规定使用 0～7 这 8 个数码，故其基数是 8。其计数规则是“逢八进一”。八进制数用下脚标 O(Octal)或者 8 来表示。把一个八进制数“按权展开”就能得到它所对应的十进制数。例如

$$\begin{aligned}(457.01)_O &= 4\times8^2+5\times8^1+7\times8^0+0\times8^{-1}+1\times8^{-2}\\ &=256+40+7+0+0.015625\\ &=(303.015625)_D\end{aligned}$$

因为 $2^3=8$，所以三位二进制数可用一位八进制数来表示。后面将举例介绍二进制数和八进制数之间特殊的转换方法。

4．十六进制

在十六进制数中，每个数位规定使用 0～9，A(10)，B(11)，C(12)，D(13)，E(14)，F(15)这 16 个数码，故其基数是 16。其计数规则是“逢十六进一”。十六进制数用下脚标 H(Hexadecimal)或者 16 来表示。把一个十六进制数“按权展开”就能得到它所对应的十进制数。例如

$$\begin{aligned}(F0D.3A)_H &= F\times16^2+0\times16^1+D\times16^0+3\times16^{-1}+A\times16^{-2}\\ &=15\times16^2+0\times16^1+13\times16^0+3\times16^{-1}+10\times16^{-2}\\ &=3804+0+13+0.1875+0.0390625\\ &=(3817.2265625)_D\end{aligned}$$

因为 $2^4=16$，所以四位二进制数可用一位十六进制数来表示。后面将举例介绍二进制数和十六进制数之间特殊的转换方法。

在计算机应用系统中，二进制主要用于机器内部的数据处理，八进制和十六进制主要用于书写程序，十进制主要用于输出最终运算结果。

以上四种计数制的对照表如表 1-1 所示。

表 1-1　四种计数制的对照表

十进制数	二进制数	八进制数	十六进制数
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	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
16	10000	20	10

1.1.2 数制转换

1．非十进制数转换成十进制数

前面已经讲过，只要将非十进制数写成按权展开的多项式，然后根据十进制数的运算规则求其和，即可转换成为等值的十进制数。

2．十进制数转换成其他进制数

十进制数的整数部分转换，采用“基数连除法”。把十进制整数 N 转换成 r 进制数的步骤是：

（1）将 N 除以 r，记下所得的商和余数；

（2）将（1）所得的商再除以 r，记下所得的商和余数；

（3）重复第（2）步，直到商为 0 为止；

（4）将各个余数转换成 r 进制的数码，并按照运算过程所得的余数逆序排列，即得 r 进制的整数。

【例 1】 $(12)_D=(?)_B$

解：

2 | 12	余数		
2 | 6	…… 0	最低位	
2 | 3	…… 0		
2 | 1	…… 1		
0	…… 1	最高位	

即 $(12)_D=(1100)_B$

【例 2】 $(429)_D=(?)_O$

解：

8 | 429	余数		
8 | 53	…… 5	最低位	
8 | 6	…… 5		
0	…… 6	最高位	

即 $(429)_D=(655)_O$

【例 3】 $(429)_D=(?)_H$

解：

16 | 429	余数		
16 | 26	…… 13=D	最低位	
16 | 1	…… 10=A		
0	…… 1	最高位	

即 $(429)_D=(1AD)_H$

十进制数的小数部分，即纯小数的转换，采用“基数连乘法”。把十进制的纯小数 M 转换成 r 进制数的步骤是：

（1）将 M 乘以 r，记下乘积的整数部分；

（2）将（1）所得的积中小数部分再乘以 r，记下乘积的整数部分；

（3）重复第（2）步，直到小数部分为 0 或者满足要求的精度为止；

（4）将各步求得的整数部分转换成 r 进制的数码，并按照运算过程所得的整数顺序排列，即为 r 进制的小数。

【例 4】 $(0.375)_D=(?)_B$

解：

0.375			
× 2			
0.750	…… 整数部分 =0	最高位	
0.750			
× 2			
1.500	…… 整数部分 =1		
0.500			
× 2			
1.000	…… 整数部分 =1	最低位	

即 $(0.375)_D=(0.011)_B$

【例 5】 $(0.85)_D=(?)_H$

解：

0.85		
× 16		
13.6	…… 整数部分 =13 即 D	最高位
0.6		
× 16		
9.6	…… 整数部分 =9	
0.6		
× 16		
9.6	…… 整数部分 =9	
⋮	⋮	最低位

即 $(0.85)_D=(0.D99\ldots)_H$

如果十进制数既有整数部分又有小数部分，则将整数部分和小数部分分别转换，再求其和即可。

3．二进制数转换成八进制数或十六进制数

当二进制数转换成八进制数(或十六进制数)时，其整数部分和小数部分可以同时进行转换，方法是：以二进制数的小数点为起点，整数部分自右至左，小数部分自左至右，每三位(或四位)分为一组，位数不足时补零(整数部分在前面补零，小数部分在后面补零)。然后把每一组二进制数转换成八进制数(或十六进制数)。

【例 6】 $(111100010.011011)_B=(?)_O=(?)_H$

解： $(111100010.011011)_B=(111100010.011011)_B=(742.33)_O$

$(111100010.011011)_B=(000111100010.01101100)_B=(1E2.6C)_H$

即$(111100010.011011)_B=(742.33)_O=(1E2.6C)_H$

4．八进制数或十六进制数转换成二进制数

把八进制数(或十六进制数)转换成二进制数，只要把八进制数(或十六进制数)的每一位，分别转换成三位(或四位)二进制数，并保持原来的顺序即可。

【例 7】 $(50.13)_O=(?)_B$

解： $(50.13)_O=(101000.001011)_B$

即$(50.13)_O=(101000.001011)_B$

【例 8】 $(1A9.0E)_H=(?)_B$

解： $(1A9.0E)_H=(000110101001.00001110)_B$

即$(1A9.0E)_H=(110101001.0000111)_B$

1.1.3 码制

按照一定的规律给不同的事物或事物的不同状态指定一个代表符号的过程叫做编码。习惯上把这些符号叫做代码。在数字系统中，所有的代码都是用若干位二进制代码“0”和“1”的不同组合构成的，因此称其为二进制代码。这些代码已不再具有数量大小的含义了。例如，n 位二进制代码，一共有 2^n 种不同的组合，可以代表 2^n 种不同的事物或信息。

在进行编码时，我们可以根据具体情况编制所需的代码。为了便于识别，编制的代码通常都有一定的规则和规律，这些规则也叫码制。下面介绍几种常见的通用代码的组成和特点。

1．十进制代码

数字系统中用十进制给出数据时，必须用 10 个二进制代码来表示 0～9 这 10 个状态。需要用到的二进制代码至少有 4 位，而 4 位二进制数有 16 种组合，从中选取 10 个与 0～9 进行对应，又会有不同的排列方式，所以编码的方案有许多种。由于这些代码都是用二进制码元来表示十进制数的，也就是用四位二进制代码表示一位十进制数，所以这些编码方案统称为二-十进制代码(Binary Code Decimal)，简称 BCD 码，它其实是一种十进制代码。表 1-2 给出了几种常见的 BCD 码。

8421 码是有权码，各位的权值分别是 8, 4, 2, 1。虽然 8421 码和 4 位自然二进制数的权值相同，但是二者有本质的区别。8421 码仅仅取用了 4 位自然二进制数的前 10 种组合，因此后面的 6 种组合在 8421 码中并不出现，这里称之为伪码。

表 1-2 几种常见的十进制代码

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

余 3 码比相应的 8421 码多 3(即 0011)。因此余 3 码所代表的 0 和 9、1 和 8、2 和 7、3 和 6、4 和 5，各对码组之和均为 1111，具有这种特性的代码称为自补代码。可以看出，余 3 码的各位无固定权值，所以它属于无权码。

5421 码是有权码，只不过每位的权值分别是 5, 4, 2, 1。

2421 码也是有权码，各位的权值分别是 2, 4, 2, 1。这种编码方案具有上下自补性的关系。

当用 BCD 码表示十进制数时，只需要把十进制的每一位数码，分别用 4 位的 BCD 码取代即可。反之亦然。下面举例说明。

【例 9】 $(902.26)_{D}=(?)_{8421BCD}=(?)_{5421BCD}$

解： 由表 1-2 可知

$(902.26)_{D}=(100100000010.00100110)_{8421BCD}=(110000000010.00101001)_{5421BCD}$

【例 10】 $(10100110.0011)_{余3码}=(?)_{D}$

解： 由表 1-2 可知

$(10100110.0011)_{余3码}=(73.0)_{D}$

2．可靠性代码

以数字形式传送信息比以模拟形式传送信息能更好地抗干扰，但是在进行大量数据交换、远距离数据传输以及存储数据的过程中免不了出现错误。为此，人们在具体的编码形式上想办法减少出错，或者一旦出错就易于发现和纠正。因此，可靠性代码(也叫纠错编码)受到普遍重视和使用。目前常用到格雷码和奇偶校验码。

(1) 格雷码

任何相邻的两个码组(包括首和尾两个码组)中，只有一个码元不同，具有这种特点的代码叫做格雷（Gray）码。

在编码技术中，把两个码组中不相同的码元个数叫做这两个码组的距离，简称码距。因此，格雷码也是单位距离码。此外，由于首和尾两个码组也具有单位距离性，因而格雷码也叫循环码。并且发现，格雷码属于无权码。

格雷码的编码方案有很多，典型的格雷码如表 1-3 所示，表中同时给出了 4 位自然二进制数。

表 1-3　典型的格雷码

十进制数	二进制数	格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

相比其他 BCD 代码，格雷码的单位距离性可以降低其产生错误的概率，并且能提高其运行速度。例如，为完成十进制数 7 加 1 的运算，当采用 4 位自然二进制数时，计数器应由 0111 变为 1000。由于计数器中各个元件特性不可能完全相同，从而导致各位上的码元不会同时发生变化，可能会出现瞬间过渡性的错码，变化过程可能是 0111→1011→1010→1000。虽然最终结果是正确的，但是在运算过程中出现了错码 1011 和 1010，这会造成数字系统的逻辑错误，并且会降低其运算速度。如果采用格雷码，由 7 变成 8，只有一位码元发生变化，就不会出现上述错误，而且运算速度也会明显提高。

(2) 奇偶校验码

奇偶校验码是一种可以检测一位错误的代码，它由信息位和按一定规律编写的校验位两部分组成。校验位是为了发现和纠正错误而添加的冗余位。

奇偶校验码是一种最简单的校验码。它的编码规律是在有效信息位之前或之后添加一位校验位，使编码中“1”的个数是奇数或者偶数。整个编码中“1”的个数之和是奇数的称为奇校验，“1”的个数之和是偶数的称为偶校验。表 1-4 给出了 8421BCD 码的奇校验码和偶校验码。

接收方对接收到的奇偶校验码进行检测，来判断每个码组中“1”的个数是否与约定相符。若不相符，则为错码。

若代码中同时出现多位错误，则奇偶校验码无法检测，它只能检测一位错码，既不能测定哪一位出错，也不能自行纠正错误。尽管如此，由于多位码同时出错的概率要比一位码出错的概率小得多，并且奇偶校验实现起来容易，信息传送率高，所以它仍被应用于误码率不高的许多场合。

表 1-4　带奇偶校验的 8421BCD 码

十进制数	奇校验码		偶校验码	
	信息位	校验位	信息位	校验位
0	0000	1	0000	0
1	0001	0	0001	1
2	0010	0	0010	1
3	0011	1	0011	0
4	0100	0	0100	1
5	0101	1	0101	0
6	0110	1	0110	0
7	0111	0	0111	1
8	1000	0	1000	1
9	1001	1	1001	0

3. 字符代码

ASCII 码是美国国家信息交换标准代码(American National Standard Code for Information Interchange)的简称，是当前计算机中使用最广泛的一种字符编码，主要用来为英文字符编码。

ASCII 码包含 52 个大、小写英文字母，10 个十进制数字符，32 个标点符号、运算符号、特殊符号和 34 个不可显示打印的控制字符，总共有 128 个，如表 1-5 所示。正好用 7 位二进制数 $b_7b_6b_5b_4b_3b_2b_1$ 进行编码表示。先读列码 $b_7b_6b_5$，再读行码 $b_4b_3b_2b_1$，则可读出某个字符的 7 位 ASCII 码。

表 1-5　ASCII 码

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	、	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	-	o	DEL

1.2　二进制数的运算

1.2.1　二进制正负数的表示方法

由于计算机只能识别 0 和 1 两种数码，这就要求数的符号也要数码化。这种在计算机中使

用的连同符号位一起数码化的数叫做机器码，也叫机器数。机器数的最高位为符号位，一般用0表示正数，用1表示负数。这种表示方法称为二进制数的原码表示。

【例11】 若机器字长为5，求$(1101)_{原码}=(?)$，$(-1101)_{原码}=(?)$

解： $(1101)_{原码}=(01101)$，$(-1101)_{原码}=(11101)$

原码表示简单直观，但是符号位不能参与算术运算。

1.2.2 二进制补码运算

将两个带符号的数相加时，有时需要将两个数的绝对值相加(当两个数的符号相同时)，有时需要将两个数的绝对值相减(当两个数的符号不同时)。而且当两个数的符号不同时，首先要判断两个数的绝对值的大小，然后才能确定哪一个是减数，哪一个是被减数。这些操作显然十分烦琐，为简化运算，在数字系统中通常采用补码相加的方法来实现有符号数的加减法运算。

在日常生活中，也有一些化减为加的例子。例如，时钟是逢12进位，12点也可以看做0点。当指针从10点调整到5点时有以下两种做法(见图1-1)。

图1-1 说明补码运算原理的实例图

(1) 逆时针方向将时针拨动5格，相当于做减法：10-5=5；

(2) 顺时针方向将时针拨动7格，相当于做加法：10+7=17。由于表盘的最大数只有12，超过12以后的进位信号自动消失，于是就只剩下减去12之后的余数了，即17-12=5。

这是由于时钟是以12为模的，在这个前提下，当超过12时，可将12舍去。同理，减4相当于加8，减3相当于加9，……。

在数学分析中，用“同余”的概念描述上述关系，即两个整数A和B用同一个正整数M(M称为模)去除而余数相等，则称A和B对M同余。具有同余关系的两个数为互补关系，其中一个称为另一个的补码。当M=12时，-5和+7、-4和+8、-3和+9就是同余的，它们互为补码。

由上述时钟和同余概念的描述，不难得出这样的结论：对于某一确定的模，用某数减去小于模的另一个数，可以用加上“该数的补码”来代替。因此，减法就可以转化为加法来做。

补码的求出按定义：正数的补码等于原码；负数的补码等于进位数减去该负数的绝对值。它也要用减法，但二进制负数的补码可通过将原码逐位取反（反码）加1得到。注意负号位不变。

【例12】 写出带符号位二进制数010101(+21)和110101(-21)的反码和补码。

解： 正数的反码和补码都和原码相同。所以+21的反码和补码都是010101。

负数的反码是，符号位保持不变，(相对原码的)数值位按位求反。所以-21的反码是101010。

负数的补码是，符号位保持不变，反码加1即可。所以-21的补码是101011。

下面再来讨论用补码表示的二进制数相加时，和的符号位如何得到。为此我们将在例13中列举出两数相加时的四种情况。

【例13】 用二进制补码运算求出13+11、13-11、-13+11和-13-11。

解： 由于两数的绝对值之和为24，所以必须用5位二进制数才能表示，再加上1位符号位，就得到6位的二进制补码。

首先写出+13和-13的补码，分别是001101和110011；+11和-11的补码分别是001011和110101。

计算结果分别为：

+13	0 01101	+13	0 01101
+11	0 01011	−11	1 10101
+24	0 11000	+2	(1)0 00010
−13	1 10011	−13	1 10011
+11	0 01011	−11	1 10101
−2	1 11110	−24	(1)1 01000

从上面的例子可以看出，若将两个加数的符号位和来自最高有效数字位的进位相加，得到的结果(舍弃产生的进位)就是和的符号。当然，如果得到的结果是负数，实际上是负数的补码，那么必须再求一次补运算，才能得到原码结果。

需要强调指出，在两个同符号数相加时，它们的绝对值之和不可超过有效数字位所能表示的最大值，否则会得出错误的计算结果，即溢出(与机器字长有关)。这就是例 13 在一开始就做出判断并分配有效数字位的原因。

1.3 逻辑运算

逻辑运算是逻辑思维和逻辑推理的数学描述。

前面已经讲过，不同的二进制数字可以表示数量的大小，还可以表示不同事物或事物的多种状态，我们称之为逻辑状态。具有“真”和“假”两种可能，并且可以判定其为“真”和“假”的陈述语句叫做逻辑变量。一般用英文大写字母 A, B, C，…表示。在数字逻辑电路中，用 1 位二进制数码的“1 和 0”表示一个事物的两种不同的逻辑状态，它们可以是“是和非”、“有和无”、“通和断”、“亮和暗”、“开和关”等。虽然“1”和“0”叫逻辑值或逻辑常量，但是它们没有数量大小的含义。

一个结论是否成立，取决于与其相关的前提条件是否具备。结论和前提条件之间的因果对应关系就叫做逻辑函数。通常写作

$$Y = f(A,B,C,\cdots)$$

Y 也是一个逻辑变量，叫做因变量或输出变量，相应地把 A, B, C，…叫做自变量或输入变量。f 是一种逻辑运算关系，它表示的是逻辑变量以及逻辑常量之间逻辑状态的推理运算，而不是数量大小之间的运算。

虽然在二值逻辑中，每个逻辑变量的取值只有 0 和 1 两种可能，只能表示两种不同的逻辑状态，但是我们可以使用多变量的不同组合来表示事物的多种逻辑状态，处理任何复杂的逻辑问题。

1.3.1 三种基本逻辑运算

在逻辑代数中也是用字母表示逻辑变量的。但是它有许多不同于普通代数的运算规则，而且在本书所讨论的二值逻辑电路中，逻辑变量的取值只有 0 和 1 两种可能。

逻辑代数中的三种基本逻辑运算是与逻辑、或逻辑和非逻辑。为直观起见，我们用电路实例图来模拟这三种不同的因果关系或者逻辑关系。如图 1-2 中的三个开关电路所示，把开关的闭合或断开作为条件，把电灯的点亮或熄灭作为结果，则这三个电路分别代表了三种基本的与逻辑、或逻辑和非逻辑运算关系。

在图 1-2(a)中，只有在两个开关同时闭合时，灯才点亮。也就是说，导致结果的所有条件同时都具备，结果才会发生。这种因果关系叫做与逻辑，也叫与运算或逻辑乘。

在图 1-2(b)中，只要有任何一个开关闭合时，灯就会点亮。也就是说，只要导致结果的所有条件当中有任何一个具备，结果就会发生。这种因果关系叫做或逻辑，也叫或运算或逻辑加。

在图 1-2(c)中，开关闭合时灯熄灭，而开关断开时灯反而点亮。也就是说，条件具备时结果不发生，而条件不具备时结果一定发生。这种因果关系叫做非逻辑，也叫非运算或逻辑反。

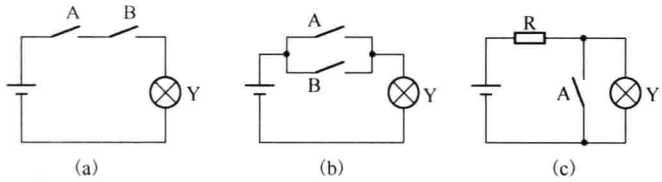


图 1-2　与、或、非逻辑的电路实例图

用“1”代表条件具备或结果发生，用“0”代表条件不具备或结果不发生，即用“1”表示开关闭合或灯点亮，用“0”表示开关断开或灯熄灭。则可列出用 0 和 1 表示的与逻辑、或逻辑及非逻辑运算的真值表，如表 1-6 所示。所谓真值表，就是将输入的所有可能取值的组合与对应的输出变量的值一一列举出来的表格。它是描述逻辑功能的一种重要形式。

表 1-6　与逻辑、或逻辑及非逻辑的真值表

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

A	Y
0	1
1	0

在逻辑代数式中，与运算符号用“·”表示，或运算符号用“+”表示，非运算符号用逻辑变量上面的“¯”表示。在有些文献中，也采用∩、∧、&表示逻辑乘(与运算)，用∪、∨表示逻辑加(或运算)，用′、～表示逻辑反(非运算)。

当 A 和 B 做与运算得到 Y 时，其逻辑表达式(也叫逻辑函数式)为

$$Y = A \cdot B \text{ 或 } Y = AB$$

当 A 和 B 做或运算得到 Y 时，其函数表达式为 $Y = A + B$。

当 A 做非运算得到 Y 时，其函数表达式为 $Y = \overline{A}$。

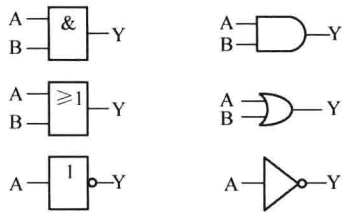
通常称 A 为原变量，$\overline{A}$ 为反变量，二者共同称为互补变量。

利用这些运算符号又可以将表 1-6 的与、或、非运算规则写为

与运算	或运算	非运算
$0 \cdot 0 = 0$	$0 + 0 = 0$	$\overline{0} = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$\overline{1} = 0$
$1 \cdot 0 = 0$	$1 + 0 = 1$	
$1 \cdot 1 = 1$	$1 + 1 = 1$	
$A \cdot A = A$	$A + A = A$	
$1 \cdot A = A$	$1 + A = 1$	
$0 \cdot A = 0$	$0 + A = A$	

实现上述逻辑运算的电路叫做门电路。与门、或门及非门的图形符号如图 1-3 所示。其中图 1-3(a)是矩形轮廓的图形符号(国标符号)，图 1-3(b)是特定外形的图形符号(国外流行符号)。本教材均采用前者。此外，参与逻辑运算的变量，可能不止两个，这里仅仅给出了最简单的情形而已。

根据逻辑运算的功能，我们还可以画出其波形图。例如与门的输入信号为 A、B、C，则可画出输出 Y 的波形如图 1-4 所示。



(a) 国标符号　　(b) 国外流行符号

图 1-3　与门、或门、非门图形符号

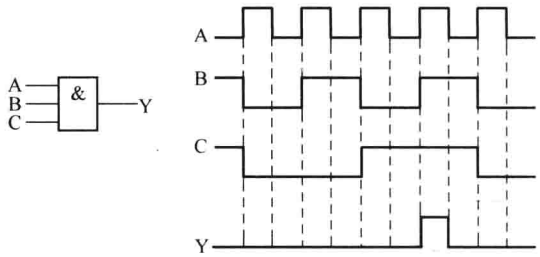


图 1-4　与门波形图

1.3.2　几种复合逻辑运算

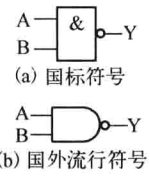
实际的逻辑问题往往要比与、或、非运算复杂得多，它们都可以用与、或、非运算复合而成。有些复合运算大量地、重复地出现，所以也可把它们视为一些基本的运算单元，其中主要有与非、或非、与或非、异或和同或等。

1. 与非

与非逻辑运算，是与逻辑和非逻辑的组合。先与运算，之后再非运算。其逻辑表达式为

$$Y=\overline{A\cdot B}$$

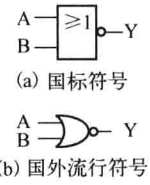
实现与非逻辑运算的门电路叫做与非门。其图形符号如图 1-5 所示，逻辑真值表如表 1-7 所示。



(a) 国标符号

(b) 国外流行符号

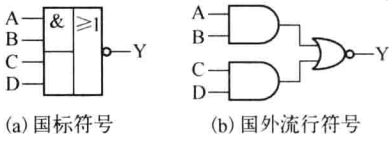
图 1-5　与非门图形符号



(a) 国标符号

(b) 国外流行符号

图 1-6　或非门图形符号



(a) 国标符号　　(b) 国外流行符号

图 1-7　与或非门图形符号

2. 或非

或非逻辑运算，是或逻辑和非逻辑的组合。先或运算，之后再非运算。其逻辑表达式为

$$Y=\overline{A+B}$$

实现或非逻辑运算的门电路叫做或非门。其图形符号如图 1-6 所示，逻辑真值表如表 1-8 所示。

3. 与或非

与或非逻辑运算，是与、或、非三种基本逻辑运算的组合。先与运算，再或运算，最后再非运算。其逻辑表达式为

$$Y=\overline{AB+CD}$$

实现与或非逻辑运算的门电路叫做与或非门。其图形符号如图 1-7 所示，逻辑真值表如表 1-9 所示。

4. 异或

若两个输入变量 A、B 的取值相异，则输出变量 Y 为 1；若两个输入变量 A、B 的取值相同，则输出变量 Y 为 0。这种逻辑关系叫做异或逻辑，其逻辑表达式为

$$Y=A\oplus B=\overline{A}B+A\overline{B}$$

实现异或逻辑运算的门电路叫做异或门。其图形符号如图 1-8 所示，逻辑真值表如表 1-10 所示。

异或的运算规则为

$$0\oplus 0=0 \quad 0\oplus 1=1 \quad 1\oplus 0=1 \quad 1\oplus 1=0 \quad 0\oplus A=A \quad 1\oplus A=\overline{A}$$

当多个变量异或时，可以通过若干个异或门来实现。例如 $Y=A\oplus B\oplus C\oplus D$ 可通过图 1-9 来实现。

多变量异或的逻辑特性是：奇数个 1 相异或，结果为 1；偶数个 1 相异或，结果为 0。异或逻辑应用较广泛。

5. 同或

若两个输入变量 A、B 的取值相同，则输出变量 Y 为 1；若两个输入变量 A、B 的取值相异，则输出变量 Y 为 0。这种逻辑关系叫做同或逻辑，其逻辑表达式为

$$Y=A\odot B=\overline{A}\,\overline{B}+AB$$

实现同或逻辑运算的门电路叫做同或门。其图形符号如图 1-10 所示，逻辑真值表如表 1-11 所示。

表 1-7　与非逻辑的真值表

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

表 1-8　或非逻辑真值表

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

表 1-9　与或非逻辑真值表

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

表 1-10　异或逻辑真值表

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

表 1-11　同或逻辑真值表

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

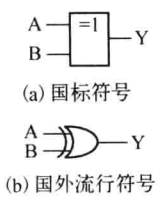


图 1-8　异或门图形符号

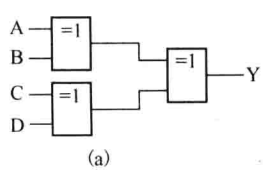
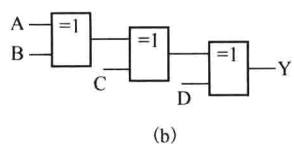


图 1-9　多变量异或的实现



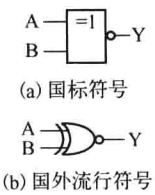


图 1-10　同或门图形符号