数字逻辑与数字电路

高晶敏　柴海莉　张金龙　编

科学出版社
www.sciencep.com

数字逻辑与数字电路

高晶敏　柴海莉　张金龙　编

科学出版社

北　京

内 容 简 介

本书在数字电子技术理论体系基础上，介绍了小规模数字集成电路的逻辑设计技术，并重点介绍了中、大规模数字集成电路和可编程逻辑器件。全书共分 8 章，包括数字逻辑基础、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、半导体存储器和可编程逻辑器件、D/A 和 A/D 转换器。每章后还配有适量习题。

本书可作为高等学校电气信息类各专业的基础课教材，也可供从事电子技术工作的工程技术人员参考。

图书在版编目(CIP)数据

数字逻辑与数字电路/高晶敏，柴海莉，张金龙编. —北京：科学出版社，2009

ISBN 978-7-03-025456-6

Ⅰ. 数… Ⅱ. ①高…②柴…③张… Ⅲ. ①数字逻辑-高等学校-教材 ②数字电路：逻辑电路-高等学校-教材 Ⅳ. TP302.2 TN79

中国版本图书馆 CIP 数据核字(2009)第 154263 号

责任编辑：孙 芳 王志欣/责任校对：郑金红

责任印制：赵 博/封面设计：耕者设计工作室

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

铭浩彩色印装有限公司 印刷

科学出版社发行 各地新华书店经销

*

2009 年 9 月第 一 版 开本：B5(720×1000)

2009 年 9 月第一次印刷 印张：17 3/4

印数：1—4 000 字数：344 000

定价：32.00 元

(如有印装质量问题，我社负责调换〈路通〉)

前　　言

数字电子技术是目前发展最为迅速的技术之一，从计算机到通信、广播、电视、医疗仪器和航空航天，几乎所有领域都在应用数字电子技术。随着数字电子技术的发展，数字集成电路经历了从分立元件、小规模、中规模、大规模到超大规模的发展。随着集成电路的密度不断提高，功能日益复杂，新型器件的相继诞生，相应的数字设计方法也在不断地演变和发展，传统的设计方法已不能完全适应器件的发展。鉴于上述情况，本书在保持数字电子技术理论体系的基础上，不仅介绍了用小规模数字集成电路为基础的数字电路和逻辑设计技术，还重点介绍了中、大规模数字集成电路和可编程逻辑器件。

在本书的编写过程中，我们总结了多年的教学实践经验，加强了基础理论和基本概念的论述，同时，也强调了现代电子技术的基本方法及其工程应用，使读者能更好地适应实际工作的需要。在内容的安排上，注意贯彻从实际出发，由浅入深、由特殊到一般、从感性上升到理性等原则。文字叙述尽量做到通俗易懂，逻辑性强。同时，每章末都附有一定数量的习题，帮助读者加深对本书内容的理解。

本书编写安排为：第 2、4～6、8 章由高晶敏、柴海莉编写，第 7 章由高晶敏、陈福彬编写，第 1、3 章由张金龙编写。

北京信息科技大学李邓化教授在百忙之中审阅了全书，并对本书的编写提出了宝贵的意见，在此表示感谢。本书由北京市属市管高等学校人才强教计划资助项目（项目编号：PHR200907124）资助。本书的编写工作还得到了北京信息科技大学教学改革基金的支持，并且在编写过程中还得到了北京信息科技大学电工电子实验教学中心全体教师的帮助，在此谨向他们表示衷心的感谢。

由于编者水平有限，书中难免存在不妥之处，恳请读者批评指正。

编　者

2009 年 6 月

目　　录

第 1 章　数字逻辑基础

1.1　数字信号与数字电路

自然界中的物理量就其变化规律而言，不外乎以下两大类。

（1）在时间上和数量上都是离散的，其数值的变化都是某一个最小数量单位的整数倍，这一类物理量称为数字量，把表示数字量的信号称为数字信号，并把工作在数字信号下的电子电路称为数字电路。

（2）在时间上或在数值上是连续的，这一类物理量称为模拟量，把表示模拟量的信号称为模拟信号，并把工作在模拟信号下的电子电路称为模拟电路。

1.2　数制和码制

1.2.1　数制

数字信号通常以数码形式给出。不同的数码可以用来表示数量的大小。用数码表示数量大小时，经常需要用进位计数制的方法组成多位数码使用。多位数码中，每一位的构成方法及从低位到高位的进位规则称为数制。经常使用的计数制除了十进制以外，还有二进制和十六进制，有时也用到八进制。

1. 十进制

十进制数中，每一位有 0～9 共 10 个数码，计数的基数是 10，其中，低位和相邻高位之间的关系是"逢十进一"，故称为十进制。例如，

$$5185.68 = 5 \times 10^3 + 1 \times 10^2 + 8 \times 10^1 + 5 \times 10^0 + 6 \times 10^{-1} + 8 \times 10^{-2}$$

通常，任意一个十进制数 D 均可展开为

$$D = \sum d_i \times 10^i \tag{1.2.1}$$

式中，d_i 是第 i 位的系数，可以是 0～9 中的任何一个。

若以 r 取代式(1.2.1)中的 10，即可得到任意进制（r 进制）数按十进制的展开式，即

$$D = \sum d_i \times r^i \tag{1.2.2}$$

式中，i 的取值与式(1.2.1)的规定相同；r 称为计数的基数；d_i 为第 i 位的系数；r^i 称为第 i 位的权。

2. 二进制

二进制数中，每一位仅有 0 和 1 两个可能的数码，所以，计数基数为 2。低位和相邻高位间的进位关系是“逢二进一”，故称为二进制。

根据式(1.2.2)，任何一个二进制数均可展开为

$$D = \sum d_i \times 2^i \tag{1.2.3}$$

这样，就可以计算出二进制数所表示的十进制数的大小。例如，

$$(101.001)_2 = 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3} = (5.125)_{10}$$

上式中分别使用下脚注 2 和 10 表示括号里的数是二进制数和十进制数，有时也用 B(binary)和 D(decimal)代替 2 和 10 这两个脚注。

3. 八进制

八进制数的每一位有 0～7 共 8 个不同的数码，计数的基数为 8。低位和相邻的高位之间的进位关系是“逢八进一”。任意一个八进制数可以按十进制数展开为

$$D = \sum d_i \times 8^i \tag{1.2.4}$$

利用上式计算出与之等效的十进制数值，例如，

$$(436.5)_8 = 4\times 8^2 + 3\times 8^1 + 6\times 8^0 + 5\times 8^{-1} = (286.625)_{10}$$

有时也有 O(octal)代替下脚注 8，表示八进制数。

4. 十六进制

十六进制数的每一位有 16 个不同的数码，分别用 0～9、A、B、C、D、E、F 表示。任意一个十六进制数均可展开为

$$D = \sum d_i \times 16^i \tag{1.2.5}$$

利用上式计算出它所表示的十进制数值，例如，

$$(1CE8)_8 = 1\times 16^3 + C\times 16^2 + E\times 16^1 + 8\times 16^0 = (7400)_{10}$$

式中的下脚注 16 表示括号里的数是十六进制数，有时也用 H(hexadecimal)代替这个脚注。

表 1.2.1 列出了几种常用数制的对照表。

表 1.2.1　常用进制数的对照表

十进制	二进制	八进制	十六进制
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3

续表

十进制	二进制	八进制	十六进制
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	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

1.2.2 数制之间的转换

数制转换就是一个数从一种进位制表示形式转换成等值的另一种进位制表示形式，其实质为权值的转换。

1. 二进制数转换为十进制数

二进制数按式(1.2.3)展开，然后将所有各项的数值按十进制数相加，就转换成相应等值的十进制数了。例如，

$$(10011)_2 = 1\times 2^4 + 0\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 = (19)_{10}$$

2. 十进制数转换成二进制数

通常使用“除基取余”法将十进制数的整数部分转换成二进制数，用“乘基取整”法将十进制数的小数部分转换成二进制数。任意十进制整数都可以写成

$$(D)_{10} = b_n\times 2^n + b_{n-1}\times 2^{n-1} + \cdots + b_1\times 2^1 + b_0\times 2^0 \qquad (1.2.6)$$

上式表明，若将$(D)_{10}$除以 2，则得到的商为 $b_n\times 2^{n-1}+b_{n-1}\times 2^{n-2}+\cdots+b_1$，而余数即 b_0。

同理，可将式(1.2.6)除以 2 得到的商写成

$$b_n\times 2^{n-1} + b_{n-1}\times 2^{n-2} + \cdots + b_1 = 2\times(b_n\times 2^{n-2} + b_{n-1}\times 2^{n-3} + \cdots + b_2) + b_1 \qquad (1.2.7)$$

由式(1.2.7)不难看出，若将$(D)_{10}$除以 2 所得的商再次除以 2，则所得余数即 b_1。依此类推，反将每次得到的商再除以 2，就可求得二进制数的每一位了。

任意十进制小数都可以写成

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

将上式两边同乘以 2 得到

$$2\times(D)_{10}=b_{-1}+b_{-2}\times2^{-1}+b_{-3}\times2^{-2}+\cdots+b_{-m}\times2^{-m+1} \tag{1.2.8}$$

式(1.2.8)说明，将小数$(D)_{10}$乘以 2 所得乘积的整数部分即 b_{-1}。

同理，将乘积的小数部分再乘以 2 又可得到

$$2\times(b_{-2}2^{-1}+b_{-3}2^{-2}+\cdots+b_{-m}2^{-m+1})=b_{-2}+(b_{-3}2^{-1}+\cdots+b_{-m}2^{-m+2}) \tag{1.2.9}$$

亦即乘积的整数部分就是 b_{-2}。

依此类推，将每次乘 2 后所得乘积的小数部分再乘以 2，便可求出二进制小数的每一位了。

3. 二进制数转换成十六进制转换数

观察表 1.2.1 可知，4 位二进制数就相当于 1 位十六进制数。因此，可用"一一对应"法将二进制数转换成十六进制数，所以，只要从低位到高位将整数部分每 4 位二进制数分为一组并代之以等值的十六进制数，同时，从高位到低位将小数部分的每 4 位数分为一组并代之以等值的十六进制数，即可得到对应的十六进制数。

例如，将$(100011001110)_2$化为十六进制数时可得

$$(100011001110)_2=(100011001110)_2=(8CE)_{16}$$

4. 十六进制数转换成二进制数

十六进制数转换成二进制数是指将十六进制数转换为等值的二进制数。转换时，只需将十六进制数的每一位用等值的 4 位二进制数得到，例如，

$$(1DBA9)_{16}=(00011101101110101001)_2=(11101101110101001)_2$$

5. 八进制数与二进制数的相互转换

二进制数与八进制数相互转换与二进制数与十六进制数相互转换的方法基本相同，按位的高低依次排列将每一位八进制数与等值的 3 位二进制数"一一对应"就可以了。

6. 十六进制数与十进制数的相互转换

将十六进制数转换为十进制数时，可根据式(1.2.5)将各位按权展开后相加求得。在将十进制数转换为十六进制数时，可以先转换为二进制数，然后再将得到的二进制数转换为等值的十六进制数。这两种转换方法上面已经讲过了。

1.2.3 二进制算术运算

两个数码同时还可以进行数量间的加、减、乘、除等运算，这种运算称为算术

运算。

1. 二进制算术运算的方法

当两个二进制数码表示两个数量大小时，它们之间可以进行数值运算，这种运算称为算术运算。二进制算术运算和十进制算术运算的规则基本相同，唯一的区别在于二进制数是“逢二进一”，而不是十进制数的“逢十进一”。

加法运算

$$\begin{array}{r} 10101101 \\ +\quad 00101100 \\ \hline 11011001 \end{array}$$

减法运算

$$\begin{array}{r} 11011101 \\ -\quad 01001100 \\ \hline 10010001 \end{array}$$

2. 反码、补码和补码运算

各种数制的数都有原码、反码和补码，而二进制数的原码、反码及补码是经常使用的。在数字电路中，用逻辑电路输出的高、低电平表示二进制数的 1 和 0，而数的正、负通常采用的方法是在二进制数的前面增加一位符号位，符号位为 0 表示这个数是正数，符号位为 1 表示这个数是负数，这种形式的数称为原码。

二进制数 B 的补码(2 的补码)常简称为补码，其定义为

$$[B]_{补} = 2^i - B \tag{1.2.10}$$

式中，i 是二进制数 B 整数部分的位数。例如，

$$[1100]_{补} = 2^4 - 1100 = 0100$$

二进制数 B 的反码(1 的补码)定义为

$$[B]_{反} = (2^i - 2^{-j}) - B \tag{1.2.11}$$

式中，i 是二进制数 B 整数部分的位数；j 是二进制数 B 小数部分的位数。例如，

$$[1100]_{反} = (2^4 - 2^0) - 1100 = 0011$$

根据定义，二进制数的补码可由反码在最低有效位加 1 得到，即

$$[B]_{补} = [B]_{反} + 1 \tag{1.2.12}$$

无论反码、补码，按定义再求补或者再求反一次将还原成原码。表 1.2.2 为原码、反码、补码的对照表。

表 1.2.2　原码、反码、补码对照表

十进制	二进制		
	原码	反码	补码
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−1	1001	1110	1111
−2	1010	1101	1110
−3	1011	1100	1101
−4	1100	1011	1100
−5	1101	1010	1011
−6	1110	1001	1010
−7	1111	1000	1001
−8	1000	1111	1000

1.2.4　二进制编码

不同的数码不仅可以用来表示数量的不同，还可以用来表示不同的事物或事物的不同状态。在用于表示不同事物的情况下，这些数码已经不再具有表示数量大小的含义了，它们只是不同事物的代号而已，这些数码称为代码。为了便于记忆和查找，在编制代码时总要遵循一定的规则，这些规则就称为码制。

1. 十进制代码

为了用二进制代码表示十进制数 0～9 这 10 个状态，二进制代码至少应当有 4 位。4 位二进制代码一共有 16 个（0000～1111），取其中哪 10 个及如何与 0～9 相对应有许多种方案。表 1.2.3 列出了常见的几种十进制代码，它们的编码规则各不相同。

表 1.2.3　十进制编码

十进制数码	BCD(8421)	2421	余 3 码	5211
0	0000	0000	0011	0000
1	0001	0001	0100	0001
2	0010	0010	0101	0100
3	0011	0011	0110	0101
4	0100	0100	0111	0111
5	0101	1011	1000	1000

续表

十进制数码	BCD(8421)	2421	余 3 码	5211
6	0110	1100	1001	1001
7	0111	1101	1010	1100
8	1000	1110	1011	1101
9	1001	1111	1100	1111
没有用到的码				
	1010	0101	0000	0010
	1011	0110	0001	0011
	1100	0111	0010	0110
	1101	1000	1101	1010
	1110	1001	1110	1011
	1111	1010	1111	1110

8421 码又称 BCD (binary coded decimal)码，是十进制代码中最常用的一种。在这种编码方式中，代码从左到右每一位的 1 分别表示 8、4、2、1，所以将这种代码称为 8421 码。每一位的 1 代表的十进制数称为这一位的权，8421 码中每一位的权是固定不变的，它属于恒权代码。

余 3 码的编码规则是在 8421 码加 3 后得到的，是一种无权码。从表 1.2.3 中还可以看出，0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 的余 3 码互为反码。

2421 码是一种恒权代码，它的 0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 也互为反码，这个特点和余 3 码相仿。

5211 码是另一种恒权代码，代码从左到右每一位的 1 分别表示 5、2、1、1，所以称为 5211 码。

2. 格雷码

格雷(gray)码又称循环码。从表 1.2.4 的 4 位格雷码编码表中可以看出格雷码的构成方法，即每一位的状态变化都按一定的顺序循环。如果从 0000 开始，最右边一位的状态按 0110 顺序循环变化，右边第二位的状态按 00111100 顺序循环变化，右边第三位按 0000111111110000 顺序循环变化。可见，自右向左每一位状态循环中连续的 0、1 数目增加一倍。由于 4 位格雷码只有 6 个，所以，最左边一位的状态只有半个循环，即 0000000011111111。

与普通的二进制代码相比，格雷码的最大优点就在于当它按照表 1.2.4 的编码顺序依次变化时，相邻两个代码之间只有一位发生变化。在代码转换的过程中，由于过渡期间会顺势出现许多别的码组，可能会造成逻辑上的差错，而格雷码就避免了这种瞬间模糊状态，所以，它是错误最小化的代码。

表 1.2.4　4 位格雷码与二进制代码的比较

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

1.3　逻辑代数基础

前面已经讲过，不同的数码不仅可以表示数量的大小，而且还能用来表示不同的事物。在数字逻辑电路中，用 1 位二进制数码的 0 和 1 表示一个事物的两种不同逻辑状态。例如，可以用 1 和 0 分别表示一件事情的真和假、信息的有和无、开关的通和断、电平的高和低、三极管的导通和截止等。

英国数学家布尔首先提出了进行逻辑运算的数学方法——布尔代数。布尔代数后来被广泛应用在开关电路和数字逻辑电路的变换、分析、化简和设计上，所以也将布尔代数称为开关代数。

逻辑代数中也用字母表示变量，这种变量称为逻辑变量。逻辑运算表示的是逻辑变量和常量之间逻辑状态的推理运算，而不是数量之间的运算。

虽然在二值逻辑中每个变量的取值只有 0 和 1 两种可能，只能表示两种不同的逻辑状态，但是，可以用多变量的不同状态组合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　逻辑代数中的基本运算

1. 基本逻辑运算

逻辑代数的基本运算有与(AND)、或(OR)、非(NOT)三种。为便于理解，先来看一个简单的例子。

图 1.3.1 中给出了三个指示灯的控制电路。在图 1.3.1(a)电路中，只有当两个开关同时闭合时，指示灯才会亮；在图 1.3.1(b)电路中，只要有任何一个开关闭合，指示灯就亮；而在图 1.3.1(c)电路中，开关断开时灯亮，开关闭合时灯反而不亮。

如果把开关闭合作为条件(或导致事物结果的原因)，把灯亮作为结果，那么，图 1.3.1 中的三个电路代表了三种不同的因果关系。

图 1.3.1(a)的例子表明，只有决定事物结果的全部条件同时具备时，结果才发生，这种因果关系称为逻辑与。图 1.3.1(b)的例子表明，在决定事物结果的诸条件中，只要有任何一个满足，结果就会发生，这种因果关系称为逻辑或。图 1.3.1(c)的例子表明，只要条件具备了，结果便不会发生；而条件不具备时，结果一定发生，这种因果关系称为逻辑非。

若以 A、B 表示开关的状态，并以 1 表示开关闭合，以 0 表示开关断开；以 Y 表示指示灯的状态，并以 1 表示灯亮，以 0 表示不亮，则可以列出以 0、1 表示的与、或、非逻辑关系的图表，如表 1.3.1～表 1.3.3 所示。这种图表称为逻辑真值表(truth table)，简称真值表。

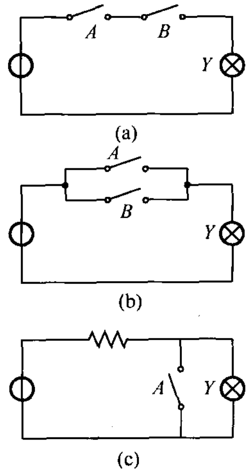

图 1.3.1　与、或、非定义的电路

表 1.3.1　与逻辑真值表

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

表 1.3.2　或逻辑真值表

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

表 1.3.3　非逻辑真值表

A	Y
0	1
1	0

在逻辑代数中，将与、或、非看作是逻辑变量 A、B 间的三种最基本的逻辑运算，并以“·”表示与运算，以“＋”表示或运算，以变量右上角的“′”表示非运算。因此，A 和 B 进行与逻辑运算时可写成

$$Y=A\cdot B \tag{1.3.1}$$

A 和 B 进行或逻辑运算时可写成

$$Y=A+B \tag{1.3.2}$$

对 A 进行非逻辑运算时可写成

$$Y=A' \tag{1.3.3}$$

同时，将实现与逻辑运算的单元电路称为与门，将实现或逻辑运算的单元电路称为或门，将实现非逻辑运算的单元电路称为非门(也称为反相器)。

与、或、非逻辑运算还可以用图形符号表示。图 1.3.2 中给出了被 IEEE 和 IEC(国际电工委员会)认定的两套与、或、非的图形符号。其中一套是目前在国外教材和 EDA 软件中普遍使用的特定外形符号，如图 1.3.2 (a)所示；另一套是矩形