新世纪高等学校计算机专业教材系列

数字逻辑与数字系统

（第三版·网络版）

白中英 主编

科学出版社
2002

内 容 简 介

本书根据数字逻辑器件的发展历程，系统地讲述了数字逻辑和数字系统的基本概念、分析方法和设计原理。全书共分六章，分别介绍了开关理论基础、组合逻辑、时序逻辑、可编程逻辑器件、在系统编程技术、数字系统等六个方面。

本书根据《计算机学科教学计划》大纲编写，体系新颖，取材先进，内容精练，文字流畅，题例丰富，并和CAI、远程网络教材、试题库、实验、课程设计等综合配套，形成了“理论、抽象、设计”三个过程相统一的教学体系，有利于教师执教，有利于学生学习。

本书是作者对“数字逻辑”课程体系、教学内容、教学方法和教学手段进行综合改革的具体成果。

本课程综合教学改革获2001年北京市教学成果二等奖。

本书可作为高等院校计算机、信息、电子工程、自动控制等专业的教材，也可作为成人教育的教材和相关专业科技人员的参考书。

图书在版编目(CIP)数据

数字逻辑与数字系统（网络版）/白中英主编. —北京：科学出版社，2002

（新世纪高等学校计算机专业教材系列）

ISBN 7-03-010371-8

Ⅰ.数… Ⅱ.白… Ⅲ.① 数字逻辑②数字系统 Ⅳ.TP302.2

中国版本图书馆CIP数据核字（2002）第022868号

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

源海印刷厂 印刷

科学出版社发行 各地新华书店经销

*

1998年7月第 一 版 开本：787×1092 1/16

1999年12月第 二 版 印张：14 1/2 插页：1

2001年8月第 三 版 字数：800 000（含光盘）

2002年4月第三版（网络版） 印数：35 001—45 000

2002年4月第五次印刷

定价：30.00元（含网络光盘）

（如有印装质量问题，我社负责调换〈环伟〉）

第三版前言

《中国教育改革和发展纲要》指出：**世界范围的经济竞争，综合国力竞争，实质上是科学技术的竞争和民族素质的竞争。从这个意义上讲，谁掌握了面向21世纪的教育，谁就能在21世纪的国际竞争中处于战略主动地位。**

处于跨世纪过程中的中国高等教育，必须花大力气对专业结构、课程体系、教学内容和教学方法进行系统的、整体的改革。可以说，全国高等学校计算机教育研究会推荐的《计算机学科教学计划》在这方面进行了卓有成效的努力，并做出了贡献。这种努力不是单纯的学时压缩，而是从时代发展、技术进步、专业结构、课程体系上的总体考虑，并跟踪了著名的美国 ACM/IEEE-CS 联合教程。

现代科学技术的发展速度真可谓一日千里。电子技术每隔5年淘汰40%，逼人更新知识而不息；新理论、新发现从提出到实际应用的周期大大缩短，催人策马紧追而不及。就数字逻辑器件的功能和使用方法来说，20世纪60年代末期出现标准通用片，70年代中后期出现现场片(PROM，PLA，PAL)，80年代初期出现半用户片(门阵列片)，80年代中后期出现通用阵列逻辑(GAL)和现场可更改的门阵列片(FPGA)，90年代又出现在系统编程(ISP)的用户片。在这样的发展历程中，用户逐步由被动地对厂商提供的标准片进行选择，发展到半主动乃至全主动地投入对芯片的设计和选择。数字器件这种更新换代的迅速发展，一方面使数字系统的设计方法发生了革命性变化，另一方面也为传统的“数字电路”课程的教学体系、教学内容、人才培养模式和任课教师提出了挑战。教材内容陈旧的局面，再也不能继续下去了。

《数字逻辑与数字系统》一书，就是作者考虑数字器件的飞速发展背景，以及计算机学科教学计划中“数字逻辑与数字系统”课程教学大纲而编写的教材。作者认为，一本好的“数字逻辑与数字系统”教材应具备以下特点：

(1) 内容全面，概念清楚；

(2) 系统性强，使学生能建立数字系统完整的总体概念；

(3) 有合理的知识结构，为进一步深入学习有关后续课程打下良好基础；

(4) 理论教学与实践教学结合，注重学生的智力开发和能力培养；

(5) 有较广的适应性，以满足学生从事开发和应用各类数字系统的需要；

(6) 力图反映新技术、新动向，以适应数字技术快速发展的需要。

根据作者多年从事理论教学与实践教学的经验，从传授知识和培养能力的目标出发，并结合本课程教学的特点和难点，本教材采用主教材、辅教材、CAI、远程网络课件、试题库、实验、课程设计等综合配套，形成了“理论、抽象、设计”三个过程相统一的教学体系。教学计划64学时左右，实践教学单独安排。

本教材第一版于1998年，第二版于1999年由科学出版社出版发行。为了在课程体系、教学内容、教学方法、教学手段上进行系统的综合改革；为了使教材精益求精，便于教师在多功能教室中上课，同时考虑远程网络课程建设的需要，我们决定编写本教材第三版，并研制配套的教学软件和实验设备。

考虑到与软件设计工具保持一致，本书中的逻辑图符采用国际通用符号。

邝坚、祁之力、岳怡、张杰、靳秀国、房鸣、李秀川、郑岩、王春露、冼茂源、王军德、林善轶、张立成、林镇、曲桑、卢宁、吴伟、赵宫明、杜泉、杨猛、杨军、索兴梅等参与了第三版文字教材与配套教学软件的研制，由于幅面所限，封面上未能一一署名。

西安交通大学计算机科学与工程系胡正家教授审定了本教材。清华大学计算机科学与技术系王尔乾教授、北京大学计算机科学与技术系崔光佐教授、北京邮电大学电信工程学院徐惠民教授审定了远程网络教材教学大纲。美国 Lattice 半导体有限公司上海分公司陈恒先生、乐峰先生在各方面给予了很大帮助。在此，作者一并向各位先生表示衷心感谢。

本教材从第一版到第三版虽然不断完善，但仍未达到作者追求的目标，欢迎读者批评指正。最后，我们引用一位哲人的名言与读者共勉：

如果今天你不想生活在未来，那么明天你将生活在过去！

白中英

北京邮电大学计算机科学与技术学院

2001 年 3 月

目　录

第一章　开关理论基础

开关理论是以二进制数为基础的理论，包括二进制数为基础的数制和码制，描述逻辑电路的数学工具、图形和符号语言。开关理论奠基了计算机等现代数字系统的硬件构造基础。本章先讨论数制和码制，然后讨论逻辑函数、布尔代数和卡诺图，最后介绍门电路的外特性。

1.1　数制与码制

1.1.1　进位计数制

1. 十进制计数制

人类的祖先在长期的生产劳动实践中学会了用十个指头计数，因而产生了我们最熟悉的十进制数。任意一个十进制数$(S)_{10}$可以表示为

$$(S)_{10} = k_n 10^{n-1} + k_{n-1}10^{n-2} + \cdots + k_1 10^0 + k_0 10^{-1} + k_{-1}10^{-2} + \cdots + k_{-m}10^{-m-1}$$

$$= \sum_{i=n}^{-m} k_i 10^{i-1} \tag{1.1}$$

其中，k_i 可以是 0～9 十个数码中的任意一个，m 和 n 是正整数，表示权；k_i，m，n 均由$(S)_{10}$决定，(S)的下标与式中的 10 是十进制的基数。由于基数为 10，每个数位计满 10 就向高位进位，即**逢十进一**，所以称它为十进制计数制。

【例 1】 将十进制数 2001.9 写成权表示的形式。

解　$(2001.9)_{10} = 2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0 + 9 \times 10^{-1}$

2. 二进制计数制

在数字系统中，为了便于工程实现，广泛采用二进计数制。这是因为，二进制表示的数的每一位只取数码 0 或 1，因而可以用具有两个不同稳定状态的电子元件来表示，并且数据的存储和传送也可用简单而可靠的方式进行。二进制的基数是 2，其计数规律是**逢二进一**。

任意一个二进制数可以表示成

$$(S)_2 = k_n 2^{n-1} + k_{n-1}2^{n-2} + \cdots + k_1 2^0 + k_0 2^{-1} + k_{-1}2^{-2} + \cdots + k_{-m}2^{-m-1}$$

$$= \sum_{i=n}^{-m} k_i 2^{i-1} \tag{1.2}$$

其中，k_i 只能取 0 或 1，它由$(S)_2$ 决定；m，n 为正整数，表示权。

【例 2】 将二进制数 1101.101 写成权表示的形式。

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

3. 八进制计数制和十六进制计数制

采用二进制计数制，对计算机等数字系统来说，运算、存储和传输极为方便，然而，二进制数书写起来很不方便。为此人们经常采用八进制计数制和十六进制计数制来进行书写或打印。

任意一个八进制数可以表示成

$$(S)_8 = \sum_{i=n}^{m} k_i 8^{i-1} \tag{1.3}$$

其中，k_i 可取 0,1,2,…,7 八个数之一，它由 $(S)_8$ 决定；m 和 n 为正整数，表示权。八进制数的计数规律为**逢八进一**。

【例 3】 将八进制数 $(67.731)_8$ 写成权表示的形式。

解 $(67.731)_8 = 6\times 8^1 + 7\times 8^0 + 7\times 8^{-1} + 3\times 8^{-2} + 1\times 8^{-3}$

任意一个十六进制数可以表示成

$$(S)_{16} = \sum_{i=n}^{m} k_i 16^{i-1} \tag{1.4}$$

其中，k_i 可取 0,1,2,…,9,A,B,C,D,E,F 等十六个数码、字母之一，它由 $(S)_{16}$ 决定；m 和 n 为正整数，表示权。十六进制数的计数规律为**逢十六进一**。

【例 4】 将十六进制数 $(8AE6)_{16}$ 写成权表示的形式。

解 $(8AE6)_{16} = 8\times 16^3 + A\times 16^2 + E\times 16^1 + 6\times 16^0$

1.1.2 进位计数制的相互转换

人们习惯的是十进制数，计算机采用的是二进制数，人们书写时又多采用八进制数或十六进制数，因此，必然产生各种进位计数制间的相互转换问题。

1. 八进制、十六进制与十进制数的转换

一个十进制整数转换成八进制表示的数时，可按**除 8 取余**的方法进行。

【例 5】 $(725)_{10} = (?)_8$

解

8	7	2	5	余数	5
8	9	0		余数	2
8	1	1		余数	3
8	1			余数	1
	0				

转换结果，得到 $(725)_{10} = (1325)_8$。

类似地，一个十进制整数转换成十六进制数时，可按**除 16 取余**的方法进行。

【例 6】 $(725)_{10} = (?)_{16}$

解

16	7	2	5	余数 5
	16	4	5	余数 13
		16	2	余数 2
			0	

转换结果，得到$(725)_{10} = (2D5)_{16}$。

一个十进制小数转换成等值的八进制数时，可按**乘 8 取整**的方法进行。

【例 7】 $(0.7875)_{10} = (?)_{8}$

解

	0.7875	
×	8	
	6.3000	整数 6
	0.3000	
×	8	
	2.4000	整数 2
	0.4000	
×	8	
	3.2000	整数 3
	……	

注意，小数转换不一定能算尽，只能算到一定精度的位数为止，故要产生一些误差。不过当位数较多时，这个误差就很小了。因此转换结果，可得$(0.7875)_{10} \approx (0.623)_{8}$。

一个十进制小数转换成等值的十六进制小数时，可按**乘 16 取整**的方法进行，其步骤与转换成八进制小数的过程相类似，不再赘述。

如果一个十进制数既有整数部分又有小数部分，可将整数部分和小数部分分别进行八进制或十六进制数的等值转换，然后合并就可得到结果。

八进制数或十六进制数转换成等值的十进制数时，可按**权相加**的方法进行。

【例 8】 $(167)_8 = 1 \times 8^2 + 6 \times 8^1 + 7 \times 8^0 = 64 + 48 + 7 = (119)_{10}$

$(0.42)_8 = 4 \times 8^{-1} + 2 \times 8^{-2} = 0.5 + 0.03125 = (0.53125)_{10}$

$(1C4)_{16} = 1 \times 16^2 + 12 \times 16^1 + 4 \times 16^0 = 256 + 192 + 4 = (452)_{10}$

$(0.68)_{16} = 6 \times 16^{-1} + 8 \times 16^{-2} = 0.375 + 0.03125 = (0.40625)_{10}$

2. 八进制、十六进制与二进制数的转换

由于数 $2^3 = 8$，$2^4 = 16$，所以一位八进制数所能表示的数值恰好相当于三位二进制数能表示的数值，而一位十六进制数与四位二进制数能表示的数值正好相当，因此八进制、十六进制与二进制数之间的转换极为方便。例如：

【例 9】 $(67.731)_8 = (110111.111011001)_2$

$(3AB4)_{16} = (0011101010110100)_2$

反之，从二进制数转换成八进制数时，只要从小数点开始，分别向左右两边把3位二进制数码划为一组，最左和最右一组不足3位用0补充，然后每组用一个八进制数码代替即成。例如：

【例10】 $(11111101.01001111)_2 = (375.236)_8$

二进制数转换成十六进制数与此类似，只不过是四位二进制数码分为一组。例如：

【例11】 $(1111101.01001111)_2 = (7D.4F)_{16}$

由上可见，用八进制或十六进制书写要比用二进制书写简短，而且八进制或十六进制表示的数据信息很容易转换成二进制表示。这就是普遍使用八进制或十六进制的原因。鉴于如此，当十进制数转换成二进制数时，可采用八进制数或十六进制数作为中间过渡。

1.1.3 二进制编码

数字系统中的信息有两类，一类是数码信息，另一类是代码信息。数码信息的表示方法如前所述，以便在数字系统中进行运算、存储和传输。为了表示字符等一类被处理的信息，也需要用一定位数的二进制数码表示，这个特定的二进制码称为**代码**。注意，“代码”和“数码”的含义不尽相同，代码是不同信息的代号，不一定有数的含义。一般地一个码字是由若干信息位组成的，每位有0和1两种代码。n 位代码可以组合成 2^n 个不同的码字，即它们可以代表 2^n 种不同信息。

给 2^n 种信息中的每个信息指定一个具体的码字去代表它，这一指定过程称为**编码**。由于指定的方法不是惟一的，故对一组信息存在着多种编码方案。

数字系统中常用的编码有两类，一类是二进制编码，另一类是二-十进制编码。

1. 二进制码

在二进制编码中，**自然二进制码**是最简单的一种。它的结构形式与二进制数完全相同。表1.1列出了四位自然二进制码，其中每位代码都有固定权值。这种代码称为**有权码**。自然二进制码是一种有权码，各信息位的权值为 2^i（i 是码元位序，$i = 0, 1, \cdots, n-1$）。

表1.1 两种4位二进制编码

十进制数	自然二进制码	循环二进制码	十进制数	自然二进制码	循环二进制码
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

另一种二进制编码是**循环二进制码**，简称**循环码**，其特性是任何相邻的两个码字中，仅有一位代码不同，其他位代码则相同。如表1.1所示，7和8是相邻的两个代码，7的代码是0100，8的代码是1100，仅有最高位代码不同。这种单位距离特性在某些设备中很有

用,因此循环码又叫**单位距离码**。循环码的编码方法不是惟一的,四位循环码就有许多种,表 1.1 中所示的是最基本的一种。

循环码是无权码,每一位都没有固定的权值。

2. 二-十进制码(BCD 码)

数字系统处理的是二进制数码,人机界面中常用十进制数进行输入和输出。为使数字系统能够传递、处理十进制数,必须把十进制数的各个数码用二进制代码的形式表示出来,这便是用二进制代码对十进制数进行编码,简称 **BCD 码**。BCD 码具有二进制码的形式(四位二进制码),又有十进制数的特点(每四位二进制码是一位十进制数)。

十进制数共有 10 个数码,需要用 4 位二进制代码来表示。4 位二进制码可以有 16 种组合,而表示十进制数只需要 10 种组合,因此用 4 位二进制码来表示十进制数有多种选取方式。表 1.2 列出了几种常用的 BCD 码与其相应的十进制数,它分为有权码和无权码两大类。

在采用有权码的一些方案中,用得最普遍的是 8421 码,即四个二进制位的位权从高向低分别为 8,4,2,1。其他的编码方法还有 2421 码,5421 码等等。其具体编码值分配如表 1.2 所示。

表 1.2 常用 BCD 码

十进制数	8421 码	2421 码	5211 码	余 3 码	格雷码
0	0000	0000	0000	0011	0000
1	0001	0001	0001	0100	0001
2	0010	0010	0011	0101	0011
3	0011	0011	0101	0110	0010
4	0100	0100	0111	0111	0110
5	0101	1011	1000	1000	1110
6	0110	1100	1010	1001	1010
7	0111	1101	1100	1010	1000
8	1000	1110	1110	1011	1100
9	1001	1111	1111	1100	0100

8421 码的编码值与字符 0 到 9 的 ASCII 码的低 4 位码相同,有利于简化输入输出过程中从字符到 BCD 或从 BCD 到字符的转换操作,是实现人机联系时比较好的中间表示。需要译码时,译码电路也比较简单。

把一个十进制数变成它的 8421 码数串,仅对十进制数的每一位单独进行。例如 1592 变为相应的 8421 码表示,结果为 0001 0101 1001 0010。相反转换过程也类似,例如 0110 1000 0100 0000 变为十进制数,结果应为 6840。

8421 码的主要缺点是实现加减运算的规则比较复杂,在某些情况下,需要对运算结果进行修正。

另外几种有权 BCD 码的共同特点是,任何两个这样的编码值相加等于 9 时,结果的四个二进制位一定为 1111。它能比较好地体现十进制的按 9 取补与二进制的按 1 取补的对应关系。

在采用无权码的一些方案中,用得比较多的是余 3 码和格雷码。

余3码是在8421码的基础上，把每个代码都加0011码而形成的。它的主要优点是执行十进制数相加时，能正确地产生进位信号，而且还给减法运算带来了方便。

格雷码的编码规则，是使任何两个相邻的代码只有一个二进制位的状态不同，其余三个二进制位必须有相同状态。这种编码方法的好处是，从某一编码变到下一个相邻编码时，只有一位的状态发生变化，有利于得到更好的译码波形。格雷码是一种循环码。

1.2 逻辑函数

1.2.1 逻辑函数的基本概念

从数学观点来讲，研究函数 $y = 5x^2 + 3x$ 时，我们对变量表示什么物理量并不感兴趣。

同样，在研究逻辑函数 $F = f(A, B)$ 时，我们可以赋给逻辑变量 A 和 B 以两个取值（二元常量**1**或**0**）中的一个，而这些逻辑变量表示什么并不重要。

数字电路是一种开关电路。开关的两种状态——“开通”与“关断”，常用电子器件的“导通”与“截止”来实现，并用二元常量**0**和**1**来表示。另一方面，数字电路的输入、输出量，一般用高、低电平来体现，高低电平又可用二元常量来表示。因此，就整体而言，数字电路的输入量和输出量之间的关系是一种因果关系，它可以用逻辑函数来描述。因此，数字电路又称**逻辑电路**。

设输入逻辑变量为 $A_1, A_2, \cdots, A_n$，输出逻辑变量为 F，当 $A_1, A_2, \cdots, A_n$ 的取值确定后，F 的值就被惟一地确定下来，则称 F 是 $A_1, A_2, \cdots, A_n$ 的**逻辑函数**，记为

$$F = f(A_1, A_2, \cdots, A_n) \tag{1.5}$$

逻辑变量和逻辑函数的取值只可能是0或1，没有其他中间值。

1.2.2 逻辑函数的表示方法

研究逻辑函数的一种数学工具叫布尔代数，它最早是由英国数学家布尔于1850年提出来的。但我们现在普遍使用的、适合于数字系统的布尔代数，是美国贝尔实验室香农于1938年提出的，它为分析、设计数字逻辑电路提供了坚强的理论基础。本书中我们仍采用**布尔代数**这一术语，不过它是指香农改进的布尔代数，不是原始的布尔代数。

布尔代数是按一定逻辑规律进行运算的代数。虽然它和普通代数一样也用字母表示变量，但是在两种代数中变量的含义完全不同。普通代数中的变量一般是连续量，而布尔代数中的变量称为**逻辑变量**，只有两种取值，即**0**和**1**。0和1并不表示数量的大小，而是表示两种对立的逻辑状态。

逻辑函数除了用布尔代数方法表述外，还常常采用另外几种工具来表述，它们是真值表法、逻辑图法、卡诺图法、波形图法、点阵图法和硬件设计语言法。

真值表是一种用表格表示逻辑函数的方法，它是由逻辑变量的所有可能取值组合及其对应的逻辑函数值所构成的表格。

逻辑图是用规定的图形符号来表示逻辑函数运算关系的网络图形。

卡诺图是一种几何图形，用来简化逻辑函数表达式，并将表达式化为最简形式的有用

工具。

波形图是用电平的高、低变化来动态表示逻辑变量值变化的图形。

点阵图是早期可编程逻辑器件中直观描述逻辑函数的一种方法。

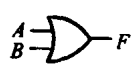
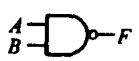
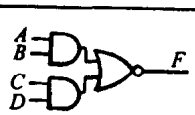
硬件设计语言法是采用计算机高级语言来描述逻辑函数并进行逻辑设计的方法，它应用于可编程逻辑器件中。目前应用最广的硬件设计语言有 ABLE-HDL、VHDL 等。

逻辑函数的上述多种表达方式各有特点，相互关联，可按需选用。本书中我们将陆续介绍这些工具，并应用这些工具。

1.2.3 基本逻辑运算

在逻辑函数中，**与、或、非**运算是三种最基本的逻辑运算。在此基础上，三种运算可以进一步组合，形成更为复杂的逻辑运算关系。表 1.3 列出了几种基本的逻辑运算。其中，后面四种逻辑运算是与、或、非三种运算的组合形式。

表 1.3 基本的逻辑运算

表示方法 / 逻辑运算	逻辑图符	逻辑函数(上) ABEL-HDL(下)	真值表		
			A	*B*	*F*
与		$F=A\cdot B$ F=A&B	0 0 1 1	0 1 0 1	0 0 0 1
或		$F=A+B$ F=A#B	0 0 1 1	0 1 0 1	0 1 1 1
非		$F=\overline{A}$ F=!A	0 1		1 0
与非		$F=\overline{A\cdot B}$ F=!(A&B)	0 0 1 1	0 1 0 1	1 1 1 0
或非		$F=\overline{A+B}$ F=!(A#B)	0 0 1 1	0 1 0 1	1 0 0 0
异或		$F=A\oplus B$ F=A$B	0 0 1 1	0 1 0 1	0 1 1 0
与或非		$F=\overline{AB+CD}$ F=!(A&B#C&D)	略		

1. 与运算

假设甲乙二人同住一个房间，房门上并挂各自一把锁，两人约定同时打开各自一把锁时，他们才能进入房间。显然，甲乙二人单独想进房间时，由于另一把锁未打开，而无法进入房间。只有二人同时打开自己的锁时，房门才能打开。这是生活中进行逻辑与运算的一个例子。

与运算的逻辑关系是：只有逻辑变量 A 和 B 同时为 1 时，逻辑函数的输出才为 1。用布尔代数表达式来描述，可写为

$$F = A \cdot B = AB \tag{1.6}$$

式中的小圆点“·”表示逻辑变量 A 和 B 的**与运算**，又称**逻辑乘**。书写时小圆点常常省去。

工程应用中，与运算采用逻辑**与门**电路来实现，因此与运算的逻辑图符即采用逻辑与门符号。

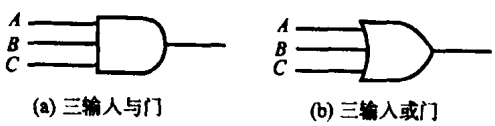
表 1.3 中右边两列表示了与运算的逻辑真值表。对 A，B 两个逻辑变量而言，输入有 $2^2 = 4$ 种(00,01,10,11)组合，故真值表中与运算的逻辑函数输出 F 也有四种情况。

与运算可以推广到任意多变量的情况。例如对三变量 A，B，C 而言，与运算的布尔代数表达式为

$$F = A \cdot B \cdot C = ABC \tag{1.7}$$

其逻辑图符如图 1.1(a)所示。显然，当输入变量 A，B，C 同时为逻辑 1 时，逻辑函数 F 的运算输出才为逻辑 1。

三变量与运算的真值表如表 1.4 所示。三变量有 $2^3 = 8$ 种组合(000 ~ 111)。只有最后一种组合(111)情况的逻辑函数 F 输出结果为 1。



(a) 三输入与门　　(b) 三输入或门

图 1.1　三输入变量的与门、或门逻辑图符

表 1.4　三变量与运算真值表

输　入			输　出
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

2. 或运算

前述例子中，假设甲乙二人在房门上共用一把锁，且各自带一把钥匙。那么任何时候，甲乙二人无论谁均可以单独进入房间，而不必等待另一人。这是生活中进行逻辑或运算的一个例子。

或运算的逻辑关系是：逻辑变量 A 或 B 任一为 1 时，逻辑函数的输出即为 1。用布尔代数表达式来描述，可写为

$$F = A + B \tag{1.8}$$

式中的“+”表示变量 A 和 B 的**或运算**，又称**逻辑加**。

工程应用中，或运算用逻辑**或门**电路来实现。因此，或运算的逻辑图符采用逻辑或门符号。

或运算的逻辑真值表列于表 1.3 中的右边两列。对 A，B 两个变量而言，输入也是 $2^2=4$种组合(00,01,10;11)，故真值表的输出也有 4 种情况。但运算结果值和与运算结果值不同。

或运算也可推广到任意多变量的情况。例如对三变量 A，B，C 而言，或运算的布尔代数表达式为

$$F = A + B + C \tag{1.9}$$

其逻辑图符如图 1.1(b)所示。显然，当输入变量 A，B，C 中任一为逻辑 1 时，逻辑函数 F 的运算输出即为逻辑 1。

三变量或运算的真值表读者依据表达式(1.9)自行导出，不再赘述。

3. 非运算

非运算是指某一逻辑函数的运算结果是逻辑变量的相反状态。用布尔代数表达式来描述，可写为

$$F = \overline{A} \tag{1.10}$$

式中，逻辑变量 A 上方的短线“-”表示非运算。

在逻辑图符中，用小圆圈“○”表示非运算。工程应用中，非运算用**非门**(反相器)电路来实现。

非运算的真值表非常简单，如表 1.3 所示。它只有两种组合，且输入变量 A 与逻辑函数 F 的输出总是处于相反的逻辑状态。

表 1.3 中的**与非**运算是与运算和非运算(先与后非)的组合，而**或非**运算是或运算和非运算的组合(先或后非)，它们的数学概念是不言自明的。相应地，在工程应用中与非运算用**与非门**电路来实现，或非运算用**或非门**电路来实现。

4. 异或运算

异或运算的逻辑函数表达式可写为

$$F = A \oplus B = \overline{A}B + A\overline{B} \tag{1.11}$$

式中，符号“⊕”表示逻辑的异或运算，它表示 AB 两个变量同为 1 或同为 0 时 F 为 1。

异或运算是非、与、或三种运算的组合，其真值表见表 1.3 所示。异或运算的规则是：变量 A 和 B 按二进制数加法法则进行**按位加**，不考虑进位，即 0+0=0,0+1=1,1+0=1,1+1=0。由于不考虑两个二进制数算术相加产生的进位(被丢弃)，故异或运算又称**模 2 加**。异或运算常用来设计二进制加法器。

工程应用中异或运算用**异或门**电路来实现，其逻辑图符如表 1.3 所示。

有时采用**异或非**(又称**同或**)运算，其逻辑表达式为

$$F = \overline{A \oplus B} = AB + \overline{A}\,\overline{B} = A \odot B \tag{1.12}$$

式中符号⊙表示逻辑的同或运算，它表示 AB 两个变量同为 1 或同为 0 时 F 为 1。

5. 与或非运算

以四变量为例，与或非运算的逻辑函数表达式可写为

$$F = \overline{AB + CD} \tag{1.13}$$

其中，A, B, C, D 为四个输入逻辑变量，它们实现**先与后或再非**的逻辑运算关系，是与、或、非三种运算的组合。只有当输入变量 $AB = 1$ 或 $CD = 1$ 时，函数 $F = 0$，在其他情况下函数 $F = 1$。四个变量有 16 种组合，读者可自行列出真值表进行验证。

在工程应用中，与或非运算由**与或非门**电路来实现，其逻辑图符示于表 1.3 中最下面一行。

1.2.4 正逻辑、负逻辑的概念

各种逻辑运算最终是通过相应的逻辑门电路来实现的，即通过门电路输入和输出的高低电平来表示逻辑变量值。

如果把门电路的输入、输出电压的高电平赋值为逻辑“1”，低电平赋值为逻辑“0”，这种关系称为**正逻辑**关系。

如果把门电路的输入、输出电压的高电平赋值为逻辑“0”，低电平赋值为逻辑“1”，这种关系称为**负逻辑**关系。

表 1.5 正负逻辑对应的门电路

正逻辑	负逻辑
或门	与门
与门	或门
与非门	或非门
或非门	与非门
异或门	同或门
同或门	异或门

按照上述定义，同一个逻辑门电路，如果在正逻辑下实现“与非”功能，那么在负逻辑下却实现“或非”功能。同理，在正逻辑定义下的“或非”门必定是在负逻辑定义下的“与非”门。表 1.5 列出了正、负逻辑定义下对应的门电路类型。读者可以用逻辑表达式或真值表进行验证。

工程应用中采用正逻辑还是负逻辑取决于个人的习惯。通常人们习惯于正逻辑，本书中也采用正逻辑。

系统中一旦采用了正逻辑，就要始终如一采用正逻辑。正、负逻辑不能混合使用。

1.3 布尔代数

1.3.1 布尔代数的基本定律

根据逻辑与、或、非三种最基本的运算法则，可导出布尔代数运算的一些基本定律和公式，如表 1.6 所示。这些基本定律在实际逻辑电路分析和设计中非常有用，需要读者熟记。

表 1.6 布尔代数定律

基本定律	$A+0=A$ $A+1=1$ $A+A=A$ $A+\overline{A}=1$	$A\cdot 0=0$ $A\cdot 1=A$ $A\cdot A=A$ $A\cdot\overline{A}=0$	$\overline{\overline{A}}=A$
结合律	$(A+B)+C=A+(B+C)$		$(AB)C=A(BC)$
交换律	$A+B=B+A$		$AB=BA$
分配律	$A(B+C)=AB+AC$		$A+BC=(A+B)(A+C)$
摩根定律	$\overline{A\cdot B\cdot C\cdots}=\overline{A}+\overline{B}+\overline{C}+\cdots$		$\overline{A+B+C+\cdots}=\overline{A}\cdot\overline{B}\cdot\overline{C}\cdots$
吸收律	$A+A\cdot B=A$ $A\cdot(A+B)=A$ $A+\overline{A}\cdot B=A+B$ $(A+B)\cdot(A+C)=A+BC$		

表 1.6 中所列的定律可以通过检验表达式两边的真值表来证明。

需要注意的是，上述基本公式只反映逻辑关系，而不是数量之间的关系，因此，初等代数中的移项规则不能使用。

1.3.2 布尔代数运算的基本规则

使用布尔代数时，可根据下面的规则从表 1.6 中得到更多的公式，从而扩充基本定律的使用范围。

1. 代入规则

任何一个含有变量 A 的等式，如果将所有出现 A 的位置都代之以一个逻辑函数，则等式仍成立。这个规则称为**代入规则**。因为任何一个逻辑函数，也和任何一个逻辑变量一样，只取二元常量 0 和 1，所以代入规则是正确的。

例如，在 $B(A+C)=BA+BC$ 中，将所有出现 A 的地方都代入函数 $A+D$，则等式仍成立，即

$$B[(A+D)+C]=B(A+D)+BC=BA+BD+BC$$

2. 反演规则

根据摩根定律，求一个逻辑函数 F 的非函数 $\overline{F}$ 时，可将 F 中的与($\cdot$)换成或($+$)，或($+$)换成与($\cdot$)；再将原变量换成非变量(如 B 换成 $\overline{B}$)，非变量换成原变量；并将 1 换成 0，0 换成 1，那么所得的逻辑函数式就是 $\overline{F}$。这个规则叫做**反演规则**。

利用反演规则，可以容易地求出一个函数的非函数。但是要注意变换时要保持原式中**先与后或**的顺序，否则容易出错。例如求 $F=\overline{A}\,\overline{B}+CD$ 的非函数时，按上述法则，可得 $\overline{F}=(A+B)\cdot(\overline{C}+\overline{D})$，而不能写成 $\overline{F}=A+B\cdot\overline{C}+\overline{D}$。