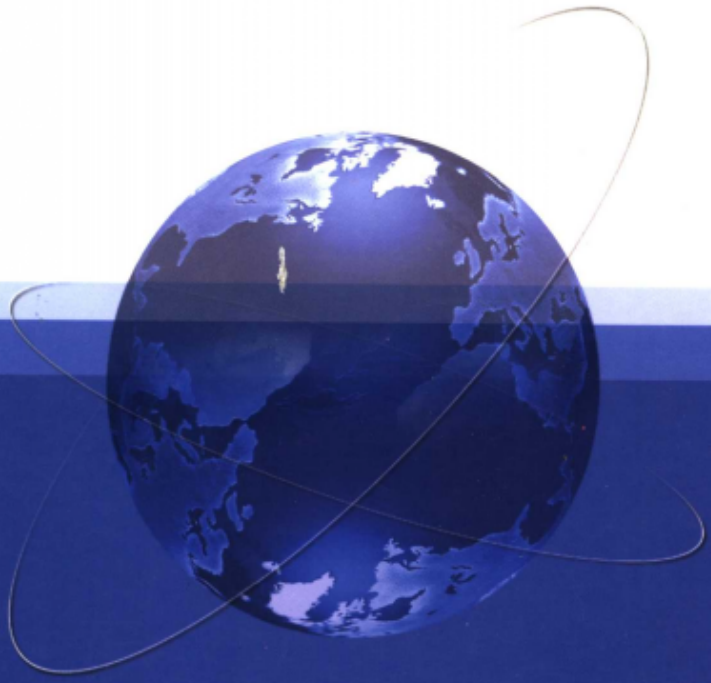
21世纪高职高专规划教材

数字逻辑电路



朱强 主编

机械工业出版社
CHINA MACHINE PRESS

ISBN 7-111-16460-1/TN·438(课)
策划编辑：余茂祚
封面设计：饶　薇

ISBN 7-111-16460-1
9 787111 164609 >

定价：20.00元

地址：北京市百万庄大街22号　邮政编码：100037
联系电话：(010) 68326294　网址：http://www.cmpbook.com
E-mail:online@cmpbook.com

21世纪高职高专规划教材

数字逻辑电路

主　编　朱　强
副主编　陈桂兰　欧阳乔
参　编　白俊平　卫秀峰　李福军　鲁波涌
主　审　沈兵虎

机械工业出版社

本书主要介绍了数字逻辑电路基础、组合逻辑电路、ROM 与可编程逻辑器件、触发器、时序逻辑电路、脉冲波形的产生和变换、数/模和模/数转换电路、现代数字系统设计初步。系统地介绍数字逻辑电路的最新发展方向，即由固定的硬件向数字硬件编程技术的转变，突出基础理论和现代技术应用。

本书可作为 2 年制和 3 年制高职高专计算机专业、电子类专业教材，亦适用于应用性本科、电大、职大等院校作为教材采用。

图书在版编目（CIP）数据

数字逻辑电路/朱强主编. —北京：机械工业出版社，2005.4

21 世纪高职高专规划教材

ISBN 7-111-16460-1

Ⅰ.数… Ⅱ.朱… Ⅲ.数字电路：逻辑电路-高等学校：技术学校-教材 Ⅳ.TN79

中国版本图书馆 CIP 数据核字（2005）第 033198 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策划编辑：余茂祚

责任编辑：余茂祚 版式设计：霍永明 责任校对：姚培新

封面设计：饶 薇 责任印制：洪汉军

北京京丰印刷厂印刷·新华书店北京发行所发行

2005 年 6 月第 1 版·第 1 次印刷

787mm×1092mm 1/16· 13.25 印张 · 324 千字

定价：20.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话（010）68326294

封面无防伪标均为盗版

21世纪高职高专规划教材
编委会名单

前　言

本书是根据教育部教高［2002］2号文件的精神，由中国机械工业教育协会和机械工业出版社组织全国80多所高职高专院校编写的21世纪高职高专规划教材之一。

随着计算机等相关技术的飞速发展，数字逻辑电路也发生了很大的变革。因此本书在基础知识和基本技能方面增加了相应的新内容，如可编程逻辑器件和现代数字逻辑电路设计的内容等。

本书主要读者对象为高职高专计算机、电子类专业学生。在内容的安排上，注意循序渐进，结构紧凑、理论严谨、难易合理，注重理论联系实际，突出基础知识的应用和实践能力的培养，突出如何引导学生利用所学知识去分析和解决数字逻辑电路方面的问题。

本书行文简洁明确，逻辑严密细致，力求突出基本概念和基本理论，强调分析和设计的思路以及必须掌握的分析方法，突出基础理论和现代技术应用，为后续课程的学习和今后实际应用打下基础。

每章前列出学习重点，每节后有小结，章后附有复习思考题。

本书第1、2章由浙江传媒学院的朱强编写，第3章由金华职业技术学院的陈桂兰编写，第4章由河北机电职业技术学院的白俊平编写，第5章由辽宁机电职业技术学院的李福军编写，第6章由太原理工大学长治学院的卫秀峰编写，第7章由湖南机电职业技术学院的鲁波涌编写，第8章由无锡商业职业技术学院的欧阳乔编写。本书由朱强任主编，负责全书的组织与统稿。浙江传媒学院沈兵虎副院长对于本书的编写给予了热情的指导，并对全书进行了细致的审阅，谨在此表示感谢。

由于编者的水平有限，书中的不足和错误之处在所难免，请各位读者批评指正。

编　者

目　录

第 1 章　数字逻辑电路基础

学 习 要 点

二进制、二进制与十进制的相互转换。

逻辑代数的公式与定理、逻辑函数化简。

基本逻辑门电路的逻辑功能。

1.1　数字逻辑电路概述

1.1.1　数字信号与数字电路

1. 模拟信号　在时间上和数值上连续的信号，如图 1-1 所示。对模拟信号进行传输、处理的电子线路称为模拟电路。

2. 数字信号　在时间上和数值上不连续的（即离散的）信号。对数字信号进行传输、处理的电子线路称为数字逻辑电路。

1.1.2　数字电路的特点与分类

1. 数字电路的特点

1）工作信号是二进制的数字信号，在时间上和数值上是离散的（不连续），反映在电路上就是低电平和高电平两种状态（即 0 和 1 两个逻辑值）。

2）在数字电路中，研究的主要问题是电路的逻辑功能，即输入信号的状态和输出信号的状态之间的关系。

3）对组成数字电路的元器件的精度要求不高，只要在工作时能够可靠地区分 0 和 1 两种状态即可。

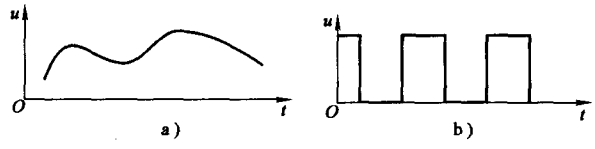


图 1-1　模拟信号与数字信号示意图

a）模拟信号波形　b）数字信号波形

2. 数字电路的分类

1）按集成度的不同分类：数字电路可分为小规模（SSI，每片数十元器件）、中规模（MSI，每片数百元器件）、大规模（LSI，每片数千元器件）和超大规模（VLSI，每片元器件数目大于 1 万）数字集成电路。集成电路从应用的角度又可分为通用型和专用型两大类型。

2）按所用元器件制作工艺的不同分类：数字电路可分为双极型（TTL 型）和单极型（MOS 型）两类。

3）按照电路的结构和工作原理的不同分类：数字电路可分为组合逻辑电路和时序逻辑电路两类。组合逻辑电路没有记忆功能，其输出信号只与当时的输入信号有关，而与电路以前的状态无关。时序逻辑电路具有记忆功能，其输出信号不仅和当时的输入信号有关，而且与电路以前的状态有关。

本 节 小 结

数字信号的数值相对于时间的变化过程是跳变的、间断性的。对数字信号进行传输、处理的电子线路称为数字电路。模拟信号通过模/数转换后变成数字信号，即可用数字电路进

行传输、处理。

1.2 数制与编码

1.2.1 数制

(1) 进位制：表示数时，仅用一位数码往往不够用，必须用进位计数的方法组成多位数码。多位数码每一位的构成以及从低位到高位的进位规则称为进位计数制，简称进位制。

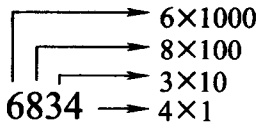
(2) 基数：进位制的基数，就是在该进位制中可能用到的数码个数。

(3) 位权（位的权数）：在某一进位制的数中，每一位的大小都对应着该位上的数码乘上一个固定的数，这个固定的数就是这一位的权数。权数是一个幂。

1. 十进制　十进制数是日常生活中使用最广的计数制。组成十进制数的符号有 0、1、2、3、4、5、6、7、8、9 等共 10 个符号，称这些符号为数码。

在十进制中，每一位有 0～9 共 10 个数码，所以计数的基数为 10。超过 9 就必须用多位数来表示。十进制数的运算遵循加法时“逢十进一”，减法时“借一当十”。

十进制数中，数码的位置不同，所表示的值就不相同。如



式中，每个对应的数码分别有一个系数 1000、100、10、1 与之相对应，这个系数就叫做权或位权。

对于任意一个十进制数可表示为

$$N_{10} = a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} + \cdots + \times a_1 \times 10^1 + a_0 \times 10^0 + a_{-1} \times 10^{-1} + a_{-2} \times 10^{-2} + \cdots + a_{-m} \times 10^{-m}$$

$$= \sum_{-m}^{n-1} a_i \times 10^i$$

式中，a_i 为 0～9 中的任一数码；10 为进制的基数；10 的 i 次幂为第 i 位的权；m、n 为正整数，n 为整数部分的位数，m 为小数部分的位数。

2. 二进制　与十进制相似，二进制数也遵循两个规则：仅有两个不同的数码，即 0 和 1；进、借位规则为“逢二进一，借一当二”。

对于任意一个二进制数均可表示为

$$N_2 = \sum_{-m}^{n-1} b_i \times 2^i$$

由于二进制数仅由 0 和 1 两个数码组成，所以其运算规则比较简单，以下列出了二进制数进行加法和乘法运算的规则：

加法：		乘法：	
	0 + 0 = 0		0 × 0 = 0
	0 + 1 = 1		0 × 1 = 0
	1 + 0 = 1		1 × 0 = 0
	1 + 1 = **10**		1 × 1 = 1

上式中 1 + 1 = **10** 中的**黑体1**为进位位。

3. 十六进制　在计算机系统中处理二进制数很方便，但当位数较多时，就难以记忆及书写。为了减少位数，通常将二进制数用十六进制数表示。

十六进制是计算机系统中除二进制之外使用较多的进制，其遵循的两个规则为：由 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 等共 16 个数码，分别对应于十进制数的 0～15。

十六进制数加、减法的进、借位规则为："借一当十六，逢十六进一"。十六进制数同二进制数及十进制数一样，也可以写成展开式的形式。

表 1-1 列出了十进制数 0～16 对应的二进制数和十六进制数。

表 1-1　二、十、十六进制对照表

十　进　制	二　进　制	十六进制	十　进　制	二　进　制	十六进制
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

在数制使用时，常将各种数制用简码来表示，如十进制数用 D 表示或省略；二进制数用 B 来表示；十六进制数用 H 来表示。

如十进制数 123 表示为 123D 或者 123；二进制数 1011 表示为 1011B；十六进制数 3A4 表示为 3A4H。

在计算机中除上面讲到的二进制、十进制、十六进制外，常常还会讲到八进制数，这里就不讨论了。

1.2.2　数制转换

二进制数和十六进制数广泛用于计算机内部的运算及表示，但人们通常是与十进制数打交道，这样在计算机的输入端就必须将十进制数转换为二进制数或十六进制数来让计算机进行处理，计算机必须将二进制数或十六进制数的处理结果转换为十进制数，否则人们无法看懂。

数制的转换可分为两类：十进制数与非十进制数之间的相互转换；非十进制数之间的相互转换。

1. 非十进制数转换成十进制数　由于任一数都可以按权展开为

$$N_R = \sum_{-m}^{n-1} a_i R^i$$

于是很容易将一个非十进制数转换为相应的十进制数。具体的步骤是：将一个非十进制数按权展开成一个多项式，每项是该位的数码与相应的权之积，把多项式按十进制数的规则进行计算及求和，所得结果即是该数对应的十进制数。

例 1　将二进制数1011.011B 转换为十进制数。

$$\begin{aligned} 1011.011_B &= 1\times2^3+0\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}+1\times2^{-3} \\ &= 8+2+1+0.25+0.125 \\ &= 11.375_D \end{aligned}$$

例 2 将十六进制数 4F.3AH 转换为十进制数。

$$4F.3A_H = 4\times16^1 + 15\times16^0 + 3\times16^{-1} + 10\times16^{-2}$$
$$= 64 + 15 + 0.1875 + 0.0390625$$
$$= 79.2265625_D$$

2. 十进制数转换成非十进制数　十进制数转换为非十进制数时，可将其分为整数部分和小数部分分别进行转换，最后将结果合并为目的数。

（1）整数部分的转换：整数部分的转换是采用除基取余法。所谓除基取余法就是用欲转换的数据的基数去除十进制数的整数部分，第一次除取得的余数为目的数的最低位，把得到的商再除以该基数，所得余数为目的数的次低位，依此类推，继续上面的过程，直到商为 0 时，所得余数为目的数的最高位。

例 3 将十进制数 53D 转换为二进制数。

53D = 110101B

2	53		
2	26	……1	低位
2	13	……0	
2	6	……1	
2	3	……0	
2	1	……1	
	0	……1	高位

（2）小数部分的转换：小数部分的转换是采用乘基取整法。所谓乘基取整法就是用该小数乘上目的数制的基数，第一次乘得结果的整数部分为目的数的小数部分的最高位，其小数部分再乘上基数，所得结果的整数部分为目的数的次高位，依此类推，继续上述的过程，直到小数部分为 0 或达到要求的精度为止。

例 4 将十进制数 71.34375D 转换为十六进制数。

16	71	余数
16	4	……7
	0	……4

0.34375	整数
× 16	……5
5.5000	
× 16	……8
8.0000	

例5 将十进制数 0.736D 转换为二进制数（结果最多保留 4 位小数）。

0.736	
× 2	整数
1.472	1
× 2	
0.944	0
× 2	
1.888	1
× 2	
1.776	1
× 2	
1.452	1
× 2	
0.904	0

从以上可以看出该数在转换为二进制数时，尽管已经经过了5次相乘，但其小数位还存在，由于题目要求保留小数后4位，故结果为

$$0.736D \approx 0.1011B$$

或

$$736D \approx 0.1100B$$

(3) 数制转换的误差问题：从上面的例子可以发现，十进制数的小数部分转换为非十进制数可能存在误差问题，这个问题是由两个方面的因素造成的：一方面是由于转换后的码元长度的限制，从上面可以看出码元长度越长其误差越小；另一方面是由于十进制与被转换的进制的基数之间存在非整数幂的关系，故即使转换的码元长度尽可能长也会存在误差，不过通常一些较小的误差并不影响计算的结果。

3. 二进制与十六进制数的相互转换 4位二进制数共有16种组合，而16种组合正好与十六进制的16种组合一致，故每4位二进制数对应于一位十六进制数，因此二进制数与十六进制数之间的转换非常简单。下面通过两个例子来加以说明。

例6 将二进制数11010110101.1100101B转换为十六进制数。

0010 1011 0101.1100 1010

2 B 5 · C A

$1010110101.11001010_B = 2B5.CA_H$

例7 将十六进制数B2C.4AH转换为二进制数。

B 2 C . 4 A

1011 0010 1100.0100 1010

$B2C.4A_H = 101100101100.0100101_B$

从以上例子可以总结出两种进制转换的方法：

(1) 二进制数转换为十六进制数时，只要将二进制数的整数部分自右向左每4位分为一组，最后不足4位的用0补足；小数部分则自左向右每4位分为一组，最后不足4位时在右边补0，最后把每4位二进制数对应的十六进制数写出来即可。

(2) 十六进制数转换为二进制数的过程正好与此相反，只要将每位的十六进制数对应的4位二进制数写出来即可。

1.2.3 编码

数字系统只能识别0和1，怎样才能表示更多的数码、符号、字母呢？用编码可以解决此问题。用一定位数的二进制数来表示十进制数码、字母、符号等信息称为编码。用以表示十进制数码、字母、符号等信息的一定位数的二进制数称为代码。

1. 二-十进制代码 用4位二进制数 $b_3b_2b_1b_0$ 来表示十进制数中的0～9这10个数码。简称BCD码。它具有二进制数的形式以满足数字系统的要求，又具有十进制数的特点（只有10种有效状态）。在某些情况下，计算机也可以对这种形式的数直接进行运算。常见的BCD码表示有以下几种。

(1) 8421BCD编码：这是一种使用最广的BCD码，是一种有权码，其各位的权（从最有效高位开始到最低有效位）分别是8、4、2、1。其编码规则如表1-2所示。

表 1-2　常见 BCD 码编码表

十进制数	8421BCD 码	2421 BCD 码	余 3 码	十进制数	8421BCD 码	2421 BCD 码	余 3 码
0	0000	0000	0011	6	0110	1100	1001
1	0001	0001	0100	7	0111	1101	1010
2	0010	0010	0101	8	1000	1110	1011
3	0011	0011	0110	9	1001	1111	1100
4	0100	0100	0111	10	0001，0000	0001，0000	0100，0011
5	0101	1011	1000				

例 8　写出十进制数 563.97D 对应的 8421BCD 码。

563.97D = 0101 0110 0011.1001 0111 8421BCD

例 9　写出 8421BCD 码 1101001.010118421BCD 对应的十进制数。

1101001.010118421BCD = 0110 1001.0101 10008421BCD = 69.58D

在使用 8421BCD 码时一定要注意其有效的编码仅 10 个，即 0000 ~ 1001。4 位二进制数的其余 6 个编码 1010、1011、1100、1101、1110 和 1111 不是有效编码。

（2）2421BCD 编码：2421BCD 码也是一种有权码，其从高位到低位的权分别为 2、4、2、1，其也可以用 4 位二进制数来表示 1 位十进制数。其编码规则见表 1-2。

（3）余 3 码：余 3 码也是一种 BCD 码，但它是无权码。由于每一个码与对应的 8421BCD 码之间相差 3，故称为余 3 码，一般使用较少，故只须作一般性了解，具体的编码见表 1-2。

2. 格雷反射码（循环码）　格雷码是一种无权码，其特点是任意两个相邻的码之间只有一个数不同。另外，由于最大数与最小数之间也仅有一个数不同，故通常又叫格雷反射码或循环码，见表 1-3。

表 1-3　格雷码编码表

十进制数	二进制数	格雷码	十进制数	二进制数	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

本 节 小 结

人们通常在日常生活中使用十进制，但计算机中基本上使用二进制，有时也使用八进制或十六进制。利用权展开式可将任意进制数转换为十进制数。将十进制数转换为其他进制数时，整数部分采用基数除法，小数部分采用基数乘法。利用 1 位八进制数由 3 位二进制数构成、1 位十六进制数由 4 位二进制数构成的原理，可以实现二进制数与八进制数以及二进制数与十六进制数之间的相互转换。

二进制代码不仅可以表示数值，而且还可以表示符号及文字，使信息交换更加灵活、方便。BCD 码是用 4 位二进制代码代表 1 位十进制数的编码，有多种 BCD 码形式，最常用的是 8421BCD 码。

1.3 逻辑代数基础

逻辑代数是 1847 年由英国数学家乔治·布尔（George Bole）首先创立的，所以通常人们又称逻辑代数为布尔代数。逻辑代数与普通代数有着不同的概念，逻辑代数表示的不是数的大小之间的关系，而是逻辑的关系，即所处的状态。它是分析和设计数字系统的数学基础。

逻辑是指事物的因果关系，或者说条件和结果的关系，这些因果关系可以用逻辑运算来表示，也就是用逻辑代数来描述。

事物往往存在两种对立的状态，在逻辑代数中可以抽象地表示为 0 和 1，称为逻辑 0 状态和逻辑 1 状态。逻辑代数中的变量称为逻辑变量，用大写字母表示。逻辑变量的取值只有两种，即逻辑 0 和逻辑 1，0 和 1 称为逻辑常量，并不表示数量的大小，而是表示两种对立的逻辑状态。

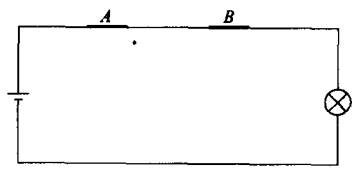
1.3.1 基本逻辑运算

逻辑代数是按一定的逻辑关系进行运算的代数，是分析和设计数字电路的数学工具。逻辑代数的运算规则也不同于普通的运算规则，在逻辑代数中，只有 0 和 1 两种逻辑值，有与、或、非三种基本逻辑运算，还有与或、与非、与或非、异或等几种导出逻辑运算。

1. 与逻辑（与运算）

与逻辑又叫做逻辑乘，下面通过开关的工作状态加以说明与逻辑的运算。

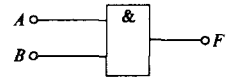
从图 1-2a 可以看出，当开关有一个断开时，灯泡处于灭的状态，仅当两个开关同时合上时，灯泡才会亮。于是可以将与逻辑的关系速记为“见 0 出 0，全 1 出 1”。



a)

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

b)



c)

图 1-2 与逻辑关系示意图

图 1-2b 列出了两个开关的所有组合，以及与灯泡状态的情况，用 0 表示开关处于断开状态，1 表示开关处于合上的状态；同时灯泡的状态用 0 表示灭，用 1 表示亮。

图 1-2c 给出了与逻辑关系的逻辑符号（Logic Symbol），该符号表示了两个输入的逻辑关系，& 在英文中是 *AND* 的速写，如果开关有 3 个，则符号的左边再加上一道线就行了。

逻辑与的关系还可以用表达式的形式表示为

$$F = A \cdot B$$

上式在不造成误解的情况下可简写为：$F = AB$。

从电路上可以看出，图 1-2a 所示的电路为一串联的电路形式，下面再来看一下并联的电路形式的逻辑关系如何。

2. 或逻辑(或运算) 图 1-3a 为一个并联直流电路,当两只开关都处于断开时,其灯泡不会亮;当 A,B 两个开关中有一个或两个一起合上时,其灯泡就会亮。如开关合上的状态用 1 表示,开关断开的状态用 0 表示;灯泡亮时的状态用 1 表示,不亮时用 0 表示,则可列出图 1-3b 所示的真值表。这种逻辑关系就是通常讲的"或逻辑"。从表中可看出,只要输入 A,B 两个中有一个为 1,则输出为 1,否则为 0。所以或逻辑可速记为:"见 1 出 1,全 0 出 0"。

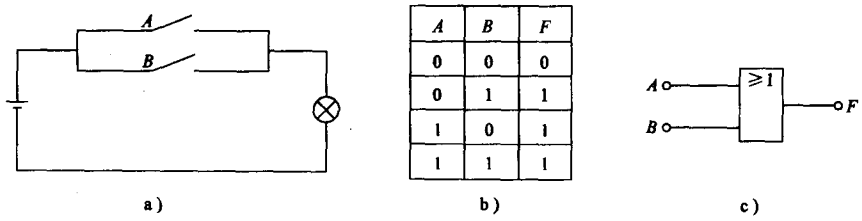


图 1-3 或逻辑关系示意图

图 1-3c 为或逻辑的逻辑符号,后面通常用该符号来表示或逻辑,其方块中的"≥1"表示输入中有一个及一个以上的 1,输出就为 1。

逻辑或的表示式为

$$F = A + B$$

3. 非逻辑 非逻辑又常称为反相运算(Inverters)。图 1-4a 所示的电路实现的逻辑功能就是非运算的功能,从图上可以看出当开关 A 合上时,灯泡反而灭;当开关断开时,灯泡才会亮,故其输出 F 的状态与输入 A 的状态正好相反。图 1-4c 给出了非逻辑的逻辑符号。非运算的逻辑表达式为

$$F = \overline{A}$$

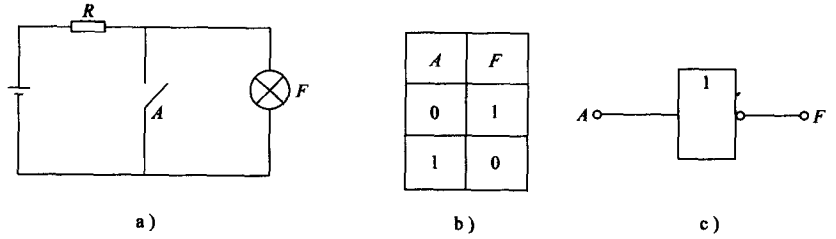


图 1-4 非逻辑关系示意图

4. 复合逻辑运算 在数字系统中,除了与运算、或运算、非运算之外,常常使用的逻辑运算还有一些是通过这三种运算派生出来的运算,这种运算通常称为复合运算,常见的复合运算有与非、或非、与或非、同或及异或等。

(1) 与非逻辑:与非逻辑是由与逻辑、非逻辑复合而成的。其逻辑可描述为"输入全部为 1 时,输出为 0;否则始终为 1"。图 1-5a 为与非运算的逻辑符号。

多输入的与非逻辑表达式可写为

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

(2) 或非逻辑:图 1-5b 为或非的逻辑符号,从与非的逻辑可以推出或非的逻辑关系:"输入中有一个及一个以上 1,则输出为 0,仅当输入全为 0 时输出为 1"。

或非逻辑的逻辑表达式可写为

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

（3）与或非逻辑：图 1-5c 为与或非的逻辑符号，A、B 相与后的输出作为或运算的输入，同时 C、D 相与后的输出作为或逻辑的输入，这两个输出再进行或运算后再进行非运算输出。与或非的逻辑表达式为

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

（4）异或逻辑：图 1-5d 为异或运算的逻辑符号，“=1”表示当两个输入中只有一个为 1 时，输出为 1；否则为 0。异或运算的逻辑表达式为

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

上式中，“⊕”表示异或运算。

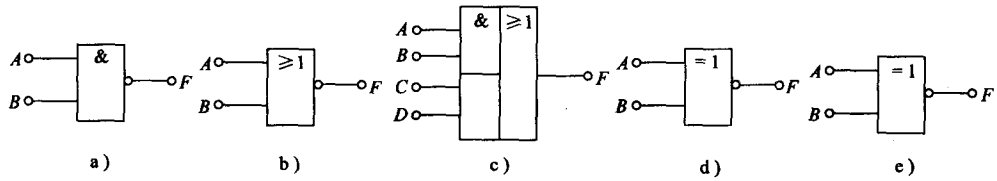


图 1-5　复合逻辑示意图

（5）同或逻辑：图 1-5e 为同或的逻辑关系，从图中上可以看出同或实际上是异或的非逻辑。

同或的逻辑表达式为

$$F=A\odot B=\overline{AB}+AB$$

上式中“⊙”是同或的逻辑运算符号。

5. 逻辑函数及其相等概念

（1）逻辑表达式：是由逻辑变量和与、或、非 3 种运算符连接起来所构成的式子。在逻辑表达式中，等式右边的字母 A、B、C、D 等称为输入逻辑变量，等式左边的字母 Y 称为输出逻辑变量，字母上面没有非运算符的叫做原变量，有非运算符的叫做反变量。

（2）逻辑函数：如果对应于输入逻辑变量 A、B、C、…的每一组确定值，输出逻辑变量 Y 就有唯一确定的值，则称 Y 是 A、B、C、…的逻辑函数。记为

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

注意：与普通代数不同的是，在逻辑代数中，不管是变量还是函数，其取值都只能是 0 或 1，并且这里的 0 和 1 只表示两种不同的状态，没有数量的含义。

（3）逻辑函数相等的概念：设有两个逻辑函数

$$Y_1=f(A,\ B,\ C,\ \cdots)\qquad Y_2=g(A,\ B,\ C,\ \cdots)$$

它们的变量都是 A、B、C、…，如果对应于变量 A、B、C、…的任何一组变量取值，Y_1 和 Y_2 的值都相同，则称 Y_1 和 Y_2 是相等的，记为 $Y_1=Y_2$。

若两个逻辑函数相等，则它们的真值表一定相同；反之，若两个函数的真值表完全相同，则这两个函数一定相等。因此，要证明两个逻辑函数是否相等，只要分别列出它们的真值表，看看它们的真值表是否相同即可。

1.3.2 逻辑代数的公式、定理和规则

1. 逻辑代数的公式和定理

(1) 常量之间的关系

与运算：$0\cdot0=0$　$0\cdot1=0$　$1\cdot0=0$　$1\cdot1=1$

或运算：$0+0=0$　$0+1=1$　$1+0=1$　$1+1=1$

非运算：$\overline{1}=0$　$\overline{0}=1$

(2) 基本公式

0-1 律：$\begin{cases}A+0=A\\A\cdot1=A\end{cases}$　$\begin{cases}A+1=1\\A\cdot0=0\end{cases}$

互补律：$A+\overline{A}=1$　$A\cdot\overline{A}=0$

等幂律：$A+A=A$　$A\cdot A=A$

双重否定律：$\overline{\overline{A}}=A$

分别令 $A=0$ 及 $A=1$ 代入这些公式，即可证明它们的正确性。

(3) 基本定理

结合律：$\begin{cases}(A\cdot B)\cdot C=A\cdot(B\cdot C)\\(A+B)+C=A+(B+C)\end{cases}$

分配律：$\begin{cases}A\cdot(B+C)=A\cdot B+A\cdot C\\A+B\cdot C=(A+B)\cdot(A+C)\end{cases}$

反演律(摩根定律)：$\begin{cases}\overline{A\cdot B}=\overline{A}+\overline{B}\\\overline{A+B}=\overline{A}\cdot\overline{B}\end{cases}$

利用真值表很容易证明这些公式的正确性。

(4) 常用公式

还原律：$\begin{cases}A\cdot B+A\cdot\overline{B}=A\\(A+B)\cdot(A+\overline{B})=A\end{cases}$

吸收率：$\begin{cases}A+A\cdot B=A\\A\cdot(A+B)=A\end{cases}$　$\begin{cases}A\cdot(\overline{A}+B)=A\cdot B\\A+\overline{A}\cdot B=A+B\end{cases}$

冗余律：$AB+\overline{A}C+BC=AB+\overline{A}C$

2. 逻辑代数运算的基本规则

(1) 代入规则：在任一逻辑等式中，如果将等式两边所有出现的某一变量都代之以一个逻辑函数，则此等式仍然成立，这一规则称之为代入规则。

例 10　将函数 $B=XY$ 代入等式 $\overline{AB}=\overline{A}+\overline{B}$ 中的 B，证明新的等式仍相等。

因为　左式 $=\overline{AB}=\overline{A(XY)}=\overline{A}+\overline{XY}=\overline{A}+\overline{X}+\overline{Y}$

　　右式 $=\overline{A}+\overline{B}=\overline{A}+\overline{XY}=\overline{A}+\overline{X}+\overline{Y}$

所以　等式成立

(2) 反演规则：已知一逻辑函数 F，求其反函数时，只要将原函数 F 中所有的原变量变为反变量，反变量变为原变量；“+”变为“·”，“·”变为“+”；“0”变为“1”；“1”变为“0”。这就是逻辑函数的反演规则。

(3) 对偶规则：已知一逻辑函数 F，只要将原函数 F 中所有的“+”变为“·”，“·”变为“+”；“0”变为“1”；“1”变为“0”，而变量保持不变、原函数的运算先后顺序保持