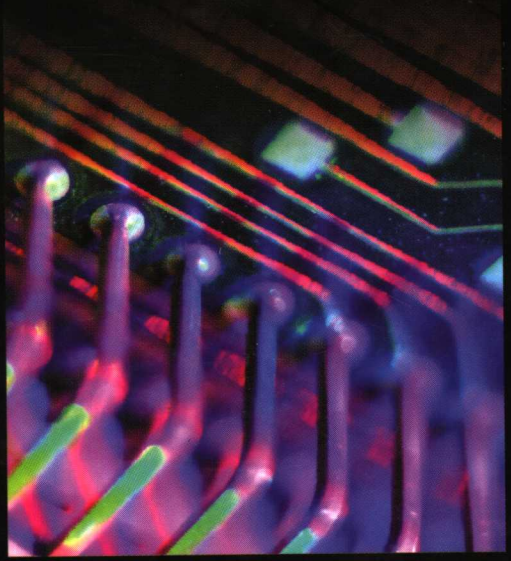
高等学校教材

电工电子技术

下册

数字与电气控制技术基础

太原理工大学电工基础教学部 编
渠云田 主编
王建平 李晓明 副主编



高等教育出版社

高等学校教材

电工电子技术

下　册

数字与电气控制技术基础

太原理工大学电工基础教学部编
渠云田　主编
王建平　李晓明　副主编

高等教育出版社

内容简介

本书是根据教育部面向21世纪电工电子技术课程教改方案，结合山西省教育厅21世纪初高等教育重点教改项目——理工科非电类专业电工电子课程模块教学改革的研究与实践而编写的第一模块教材。

本书的基本特点是传统理论内容精练、结构顺序合理，以EDA作为教学平台，淡化计算技巧，注重基本概念，强化电气应用，充分引用电工电子新技术，以加强学生电气技能与素质的培养。

本书的下册内容包括：数字电路基础、组合逻辑电路、触发器与时序逻辑电路、脉冲波形的产生与整形、数/模和模/数转换技术、存储器与可编程逻辑器件、电机与电气控制技术基础、可编程控制器，共8章。每章都配有基本概念题与EDA仿真题。

本书适用于高等学校理工科非电类专业和计算机专业的适用教材。也可作为高职、高专以及成人教育相应专业的选用教材，还可作为相关专业工程技术人员的参考书。

图书在版编目(CIP)数据

电工电子技术.下册，数字与电气控制技术基础/渠云田主编.—北京：高等教育出版社，2003.2(2004重印)
ISBN 7-04-011861-0

Ⅰ.电… Ⅱ.渠… Ⅲ.①电工技术-高等学校-教材②电子技术-高等学校-教材 Ⅳ.TM

中国版本图书馆CIP数据核字(2003)第000417号

出版发行	高等教育出版社	购书热线	010-64054588
社　　址	北京市西城区德外大街4号	免费咨询	800-810-0598
邮政编码	100011	网　　址	http://www.hep.edu.cn
总　　机	010-82028899		http://www.hep.com.cn
经　　销	新华书店北京发行所		
排　　版	高等教育出版社照排中心		
印　　刷	河北新华印刷一厂		
开　　本	787×1092　1/16	版　　次	2003年2月第1版
印　　张	17.25	印　　次	2004年3月第2次印刷
字　　数	420 000	定　　价	20.10元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

版权所有　侵权必究

目　录

试读结束，需要全本PDF请购买 www.ertongbook.com

第 9 章　数字电路基础

数字信号是时间上和数值上均离散的一种信号，对该种信号进行传递、处理、运算和存储的电路称为数字电路。这里的运算不仅有普通的算术运算而且有逻辑运算。

本章将介绍数字电路的基础知识：数制、编码以及逻辑代数。因为门电路是数字电路的基本单元，故门电路也在这里介绍。

9.1　数　　制

数制是人们对数量计算的一种统计规律。在日常生活中，我们最熟悉的是十进制，而在数字系统中广泛使用的是二进制、八进制和十六进制。

9.1.1　几种常用的进位计数制

1. 十进制

十进制的数码有 0,1,2,3,4,5,6,7,8,9 共十个，进位规律是“逢十进一”。

十进制数 378 4.25 可表示成多项式形式：

$$(378\ 4.25)_{10} = 3\times10^{3} + 7\times10^{2} + 8\times10^{1} + 4\times10^{0} + 2\times10^{-1} + 5\times10^{-2}$$

对任意一个十进制数可表示为：

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

以上公式中 a_i 是第 i 位的系数，它可能是 0～9 中的任意数码，n 表示整数部分的位数，m 表示小数部分的位数，10^{i} 表示数码在不同位置的数值大小，称为位权。

2. 二进制

在数字电路中，电路的状态以数字来表示。找一个具有十种状态的电子器件比较难，而找一一个具有两种状态的器件很容易，故数字电路中广泛使用二进制。

二进制的数码只有二个，即 0 和 1。进位规律是“逢二进一”。

二进制数 1 101.11 可以表示成多项式形式：

$$(1\ 101.11)_{2} = 1\times2^{3} + 1\times2^{2} + 0\times2^{1} + 1\times2^{0} + 1\times2^{-1} + 1\times2^{-2}$$

对任意一个二进制数可表示为：

$$(N)_{2} = \sum_{i=-m}^{n-1} a_i \times 2^{i}$$

上式中 a_i 是第 i 位的系数，它可能是 0、1 中的任意数码，n 表示整数部分的位数，m 表示小数部分的位数，2^{i} 表示数码在不同位置的数值大小，称为位权。

3. 八进制和十六进制数

用二进制表示一个大数时，位数太多。在数字系统中采用八进制和十六进制作为二进制的缩写形式。

八进制数码有 8 个，即：0，1，2，3，4，5，6，7。进位规律是“逢八进一”。十六进制的数码是：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F。进位规律是“逢十六进一”。不管是八进制还是十六进制都可以像十进制和二进制那样，用多项式的形式来表示。

9.1.2 数制间的转换

计算机中存储数据和对数据进行运算采用的是二进制数，当把数据输入到计算机中，或者从计算机中输出数据时，要进行不同数制之间的转换。

1. 非十进制数到十进制数的转换

非十进制数转换成十进制数一般采用的方法是按权相加，这种方法是按照十进制数的运算规则，将非十进制数各位的数码乘以对应的权再累加起来。

例 9-1 将$(1\ 101.101)_2$转换成十进制数。

解：
$$\begin{aligned}(1\ 101.101)_2 &= (2^3+2^2+2^0+2^{-1}+2^{-3})_{10}\\ &= (8+4+1+0.5+0.125)_{10}\\ &= (13.625)_{10}\end{aligned}$$

在二进制数到十进制数的转换过程中，要频繁的计算 2 的整次幂。下面表 9-1 给出了常用的 2 的整次幂和十进制数的对应关系，记住这些值，对今后的学习是十分有益的。

表 9-1 常用的 2 的整次幂和十进制数的对应关系表

n	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10
2^n	0.062 5	0.125	0.25	0.5	1	2	4	8	16	32	64	128	256	512	1 024

2. 十进制数到非十进制数的转换

将十进制数转换成非十进制数时，整数部分的转换一般采用除基取余法，小数部分的转换一般采用乘基取整法。

(1) 十进制整数转换成非十进制整数

例 9-2 将$(41)_{10}$转换成二进制数。

解：
$$\begin{aligned}&41/2=20 \quad \text{余数为 } 1, \quad A_0=1\\ &20/2=10 \quad \text{余数为 } 0, \quad A_1=0\\ &10/2=5 \quad \text{余数为 } 0, \quad A_2=0\\ &5/2=2 \quad \text{余数为 } 1, \quad A_3=1\\ &2/2=1 \quad \text{余数为 } 0, \quad A_4=0\\ &1/2=0 \quad \text{余数为 } 1, \quad A_5=1\end{aligned}$$

所以，$(41)_{10}=(101\ 001)_2$

(2) 十进制小数转换成非十进制小数

例 9－3 将$(0.625)_{10}$转换成二进制数。

解： $0.625\times2=1+0.25$　　$A_{-1}=1$

$0.25\times2=0+0.5$　　$A_{-2}=0$

$0.5\times2=1+0$　　$A_{-3}=1$

所以，$(0.625)_{10}=(0.101)_2$

由于不是所有的十进制小数都能用有限位 R 进制小数来表示，因此，在转换过程中可根据精度要求取一定的位数即可。若要求误差小于 R^{-n}，则转换取小数点后 n 位就能满足要求。

例 9－4 将$(0.7)_{10}$转换成二进制数，要求误差小于2^{-6}。

解： $0.7\times2=1+0.4$　　$A_{-1}=1$

$0.4\times2=0+0.8$　　$A_{-2}=0$

$0.8\times2=1+0.6$　　$A_{-3}=1$

$0.6\times2=1+0.2$　　$A_{-4}=1$

$0.2\times2=0+0.4$　　$A_{-5}=0$

$0.4\times2=0+0.8$　　$A_{-6}=0$

所以，$(0.7)_{10}=(0.101\ 100)_2$

由于最后剩下的未转换部分就是误差，它在转换过程中扩大了2^6，所以真正的误差应该是：0.8×2^{-6}，其值小于2^{-6}，所以满足精度要求。

3．非十进制数之间的转换

（1）二进制数和八进制数之间的转换

二进制数的基数是 2，八进制数的基数是 8，正好有$2^3=8$。因此，任意 1 位八进制数可以转换成 3 位二进制数。当要把一个八进制数转换成二进制数时，可以直接将每位八进制数转换成 3 位二进制数。而二进制数到八进制数的转换可按相反的过程进行，转换时，从小数点开始向两边分别将整数和小数每 3 位划分成一组，整数部分的最高一组不够三位时，在高位补 0，小数部分的最后一组不足 3 位时，在末位补 0，然后将每组的 3 位二进制数转换成 1 位八进制数即可。

例 9－5 将$(354.72)_8$转换成二进制数。

解：

3	5	4	.	7	2
↓	↓	↓		↓	↓
011	101	100	.	111	010

所以，$(354.72)_8=(011\ 101\ 100.111\ 010)_2$

例 9－6 将$(1\ 010\ 110.010\ 1)_2$转换成八进制数

解：

001	010	110	.	010	100
↓	↓	↓		↓	↓
1	2	6	.	2	4

所以，$(1\ 010\ 110.010\ 1)_2=(126.24)_8$

（2）二进制数和十六进制数之间的转换

二进制数的基数是 2，十六进制数的基数是 16，正好有$2^4=16$。因此，任意 1 位十六进制数

可以转换成4位二进制数。当要把一个十六进制数转换成二进制数时，可以直接将每位十六进制数转换成4位二进制数。对二进制数到十六进制数的转换可按相反的过程进行，转换时，从小数点开始向两边分别将整数和小数每4位划分成一组，整数部分的最高一组不够4位时，在高位补0，小数部分的最后一组不足4位时，在末位补0，然后将每组的4位二进制数转换成1位十六进制数即可。

例 9-7 将$(8E.4A)_{16}$转换成二进制数。

解：

8	E	.	4	A
↓	↓		↓	↓
1000	1110	.	0100	1010

所以，$(8E.4A)_{16} = (1000\ 1110.0100\ 1010)_2$

例 9-8 将$(101\ 1111.1011\ 01)_2$转换成十六进制数。

解：

0101	1111	.	1011	0100
↓	↓		↓	↓
5	F	.	B	4

所以，$(101\ 1111.1011\ 01)_2 = (5F.B4)_{16}$

(3) 八进制数和十六进制数之间的转换

八进制数和十六进制数之间的转换，直接进行比较困难，可用二进制数作为转换中介，即先转换成二进制数，再进行转换就比较容易了。

例 9-9 将$(345.27)_8$转换成十六进制数

解：

3	4	5	.	2	7	
↓	↓	↓		↓	↓	
011	100	101	.	010	111	先转换成二进制数
1110		0101	.	0101	1100	重新分组
↓		↓		↓	↓	
E		5	.	5	C	转换成十六进制数

所以，$(345.27)_8 = (E5.5C)_{16}$

例 9-10 将$(2B.A6)_{16}$转换成八进制数。

解：

2	B	.	A	6		
↓	↓		↓	↓		
0010	1011	.	1010	0110		先转换成二进制数
101	011	.	101	001	100	重新分组
↓	↓		↓	↓	↓	
5	3	.	5	1	4	转换成八进制数

所以，$(2B.A6)_{16} = (53.514)_8$

9.2 编　　码

用二进制数表示十进制数或其他特殊信息如字母、符号等的过程称为编码。编码在数字系统中经常使用，例如通过计算机键盘将命令、数据等输入后，首先将它们转换为二进制数，然后才能进行信息处理。

9.2.1 二－十进制码(BCD 码)

二－十进制码是用 4 位二进制数表示 1 位十进制数的代码(Binary Coded Decimals)，简称为 BCD 码。这种编码的方法很多，但常用的是 8421BCD 码、5421BCD 码和余 3 码等。

1. 8421BCD 码

8421BCD 码是最常用的一种二－十进制数编码，它是用 4 位二进制数 0000 到 1001 来表示十进制数的 0～9。该码的每一位都有固定的权，从左到右依次为：2^3，2^2，2^1，2^0，即 8，4，2，1。表 9－2 中给出了 8421BCD 码和十进制数之间的对应关系。

表 9－2　十进制数和 8421BCD 码之间的对应关系

十进制数	8421 码	十进制数	8421 码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

例 9－11　将十进制数 1 987.35 转换成 8421BCD 码。

解：　$(1\ 987.35)_{10}=(0001\quad 1001\quad 1000\quad 0111.0011\quad 0101)_{8421BCD}$

2. 余 3 码

余 3 码也是用 4 位二进制数表示 1 位十进制数，但对于同样的十进制数，比 8421BCD 码多 0011，所以叫余 3 码。余 3 码用 0011 到 1100 这十种编码表示十进制数的 0～9，和十进制数之间的对应关系如表 9－3 所示。

表 9－3　十进制数和余 3 码之间的对应关系

十进制数	余 3 码	十进制数	余 3 码
0	0011	5	1000
1	0100	6	1001
2	0101	7	1010
3	0110	8	1011
4	0111	9	1100

试读结束，需要全本PDF请购买 www.ertongbook.com

5421BCD 码最高位的权是 5,其他类似于 8421BCD 码,这里就不多讲了。

9.2.2 可靠性编码

表示信息的代码在形成、存储和传送过程中,由于某些原因可能会出现错误。为了提高信息的可靠性,需要采用可靠性编码。常用的可靠性编码有循环码、奇偶校验码等,下面分别介绍。

1. 循环码

循环码又称为格雷码(GRAY),具有多种编码形式,但都有一个共同的特点,就是任意两个相邻的循环码仅有 1 位不同。例如 4 位二进制数,在从 0101 变成 0110 时,最低两位都要发生变化。当两位不是同时变化时,如最低位先变,次低位后变,就会出现一个短暂的误码 0100。采用循环码表示时,因为只有 1 位发生变化,就可以避免出现这类错误。

循环码是一种无权码,每 1 位都按一定的规律循环。表 9-4 给出了一种 4 位循环码的编码方案。可以看出,任意两个相邻的编码仅有 1 位不同,而且存在一个对称轴(在 7 和 8 之间),对称轴上边和下边的编码,除最高位是互补外,其余各个数位都是以对称轴为中线镜像对称的。

表 9-4 4 位循环码

十进制数	二进制数	循环码
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

2. 奇偶校验码

奇偶校验码是目前使用最广泛、最简单的一种校验码。它是由有效和校验位组成,校验位可位于有效信息位的前面,也可以位于有效信息位的后面,奇偶校验码在编码时可根据有效信息位

中1的个数决定校验位是1还是0,从而使编码中1的个数是奇数或偶数。编码中1的个数是奇数的称为奇校验码,1的个数是偶数的称为偶校验码。表9-5给出了数字0到9的8421BCD码的奇校验码和偶校验码,校验位是添加在8421BCD码的后面。

在读出和接收到奇偶校验码时,检测编码中1的个数是否符合奇偶规律,如不符合则是有错。奇偶校验码可以发现错误,但不能纠正错误。当出现偶数个错误时,奇偶校验码也不能发现错误。

表9-5 数字0到9的8421BCD码的奇校验码和偶校验码

十进制数	8421BCD码	奇校验码	偶校验码
0	0000	00001	00000
1	0001	00010	00011
2	0010	00100	00101
3	0011	00111	00110
4	0100	01000	01001
5	0101	01011	01010
6	0110	01101	01100
7	0111	01110	01111
8	1000	10000	10001
9	1001	10011	10010

9.3 逻辑代数基础

逻辑代数是英国数学家乔治·布尔(George Boole)在19世纪中期(1848年)研究思维规律时首先提出来的,因此又称其为布尔代数。1938年布尔代数首次用于电话继电器开关电路的设计,所以又称它为开关代数。目前逻辑代数已成为数字系统分析和设计的重要工具。

9.3.1 逻辑代数的特点和基本运算

逻辑代数是研究因果关系的一种代数,和普通代数类似,可以写成下面的表达形式

$$Y=F(A,B,C,D)$$

逻辑变量A,B,C和D称为自变量,Y称为因变量,描述因变量和自变量之间的关系称为逻辑函数。但它有与普通代数不同的两个特点:

第一,变量的值只有"0"和"1"两个,且这两个值不表示数值的大小,只表示事物的性质、状态等。

在逻辑电路中,通常规定1代表高电平,0代表低电平,为正逻辑。如果规定0代表高电平,1代表低电平,则称为负逻辑。在以后如不专门说明时,指的都是正逻辑。

第二,逻辑函数只有三种基本运算,它们是**与**运算、**或**运算和**非**运算。

以下分别介绍这三种基本运算

1. 与运算 $F = A \cdot B$

与运算的规则可用表 9－6 说明，该表称为真值表，它反映所有变量全部可能的组合和运算结果之间的关系。真值表在以后的逻辑电路分析和设计中是十分有用的。

与运算的例子在日常生活中经常会遇到，如图 9－1 所示的串联开关电路，灯 F 亮的条件是开关 A 和 B 都必须接通。如果开关闭合表示 1，开关断开表示 0；灯亮表示 1，灯灭表示 0。则灯和开关之间的逻辑关系可表示为 $F = A \cdot B$。

表 9－6　逻辑与运算

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

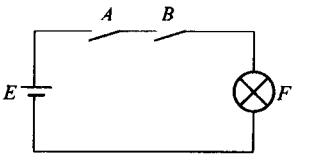


图 9－1　逻辑与的例子

2. 或运算 $F = A + B$

或运算的规则可用表 9－7 说明。或运算的例子在日常生活中也会经常遇到，如图 9－2 所示，灯 F 亮的条件是只要有一个开关或一个以上的开关接通就可以。灯和开关之间的逻辑关系可表示为 $F = A + B$。

表 9－7　逻辑或运算

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

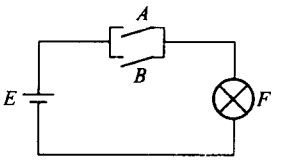


图 9－2　逻辑或的例子

3. 非运算 $F = \overline{A}$

非运算的真值表如表 9－8 所示，图 9－3 所示电路反映了灯 F 和开关 A 之间的非运算关系。如果闭合开关，灯则不亮；如果断开开关，灯则会亮。

表 9－8　逻辑非运算

A	F
0	1
1	0

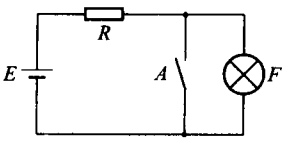


图 9－3　逻辑非的例子

9.3.2　逻辑代数的基本公式和规则

逻辑代数的基本公式对于逻辑函数的化简是非常有用的。大部分逻辑代数的基本公式的正确性是显见的，以下仅对不太直观的公式加以证明。

1. 基本公式

(1) 0－1 律　$A+1=1$　　$A\cdot 0=0$

(2) 自等律　$A+0=A$　　$A\cdot 1=A$

(3) 互补律　$A\cdot\overline{A}=0$　　$A+\overline{A}=1$

(4) 交换律　$A+B=B+A$　$A\cdot B=B\cdot A$

(5) 结合律　$A+(B+C)=(A+B)+C$　　$A\cdot(B\cdot C)=(A\cdot B)\cdot C$

(6) 分配律　$A+B\cdot C=(A+B)\cdot(A+C)$　$A\cdot(B+C)=A\cdot B+A\cdot C$

证明：$(A+B)\cdot(A+C)=A+AB+AC+BC=A+BC$

(7) 吸收律　$A+A\cdot B=A$　　$A\cdot(A+B)=A$

$A+\overline{A}\cdot B=A+B$　　$A\cdot(\overline{A}+B)=A\cdot B$

(8) 重迭律　$A+A=A$　　$A\cdot A=A$

(9) 反演律　$\overline{A\cdot B}=\overline{A}+\overline{B}$　　$\overline{A+B}=\overline{A}\cdot\overline{B}$

(10) 还原律　$\overline{\overline{A}}=A$

(11) 包含律

$$AB+\overline{A}C+BC=AB+\overline{A}C$$

证明：
$$\begin{aligned}AB+\overline{A}C+BC&=AB+\overline{A}C+BC(A+\overline{A})\\&=AB+\overline{A}C+ABC+\overline{A}BC\\&=AB+\overline{A}C\end{aligned}$$

2. 运算规则

逻辑代数有三个重要的运算规则，即代入规则、反演规则和对偶规则，这三个规则在逻辑函数的化简和变换中是十分有用的。

(1) 代入规则

代入规则是指：将逻辑等式中的一个逻辑变量用一个逻辑函数代替，则逻辑等式仍然成立。使用代入规则，可以容易地证明许多等式，扩大基本公式的应用范围。

(2) 反演规则

反演规则是指：如果将逻辑函数 F 的表达式中所有的“·”都换成“+”，“+”都换成“·”，“1”都换成“0”，“0”都换成“1”，原变量都换成反变量，反变量都换成原变量，所得到的逻辑函数就是 F 的反函数。

利用反演规则可以很容易地写出一个逻辑函数的反函数。

例 9－12　求逻辑函数 $F=AB+CD$ 的非。

解：　根据反演规则有：$\overline{F}=(\overline{A}+\overline{B})\cdot(\overline{C}+\overline{D})$。

(3) 对偶规则

将逻辑函数 F 的表达式中所有的“·”都换成“+”，“+”都换成“·”，“1”都换成“0”，“0”都换成“1”，所得到的逻辑函数就是 F 的对偶式，记为 F'。如果两个逻辑函数相等则对偶式也相等，即为对偶规则。利用对偶规则可以使逻辑函数的证明简单化。

9.3.3 最小项和最小项表达式

1. 最小项

如果一个具有 n 个变量的逻辑函数的"**与**项"包含全部 n 个变量，每个变量以原变量或反变量的形式出现，且仅出现一次，则这种"**与**项"被称为最小项。

对两个变量 A、B 来说，可以构成 4 个最小项：$\overline{A}\,\overline{B}$、$\overline{A}B$、$A\,\overline{B}$、$AB$；对三个变量 A、B、C 来说，可构成八个最小项：$\overline{A}\,\overline{B}\,\overline{C}$、$\overline{A}\,\overline{B}C$、$\overline{A}B\,\overline{C}$、、$\overline{A}BC$、$A\,\overline{B}\,\overline{C}$、$A\,\overline{B}C$、$AB\,\overline{C}$、$ABC$；同理，对 n 个变量来说，可以构成 2^n 个最小项。

为了叙述和书写方便，最小项通常用符号 m_i 表示，i 是最小项的编号，是一个十进制数。确定 i 的方法是：首先将最小项中的变量按顺序 A、B、C、D…排列好，然后将最小项中的原变量用 1 表示，反变量用 0 表示，这时最小项表示的二进制数对应的十进制数就是该最小项的编号。例如，对三变量的最小项来说，ABC 的编号是 7 的符号用 m_7 表示，$A\,\overline{B}C$ 的编号是 5 的符号用 m_5 表示。

2. 最小项表达式

如果一个逻辑函数表达式是由最小项构成的**与或**式，则这种表达式称为逻辑函数的最小项表达式，也称为标准与或式。例如

$$F = \overline{A}BC\,\overline{D} + ABC\,\overline{D} + ABCD$$

是一个四变量的最小项表达式。

对一个最小项表达式可以采用简写的方式，例如：

$$\begin{aligned} F(A,B,C) &= \overline{A}B\,\overline{C} + A\,\overline{B}C + ABC \\ &= m_2 + m_5 + m_7 \\ &= \sum m(2,5,7) \end{aligned}$$

要写出一个逻辑函数的最小项表达式，可以有多种方法，但最简单的方法是先给出逻辑函数的真值表，将真值表中能使逻辑函数取值为 1 的各个最小项相或就可以了。

例 9-13 已知三变量逻辑函数：$F = AB + BC + AC$，写出 F 的最小项表达式。

解： 首先列出 F 的真值表，如表 9-9 所示，将表中能使 F 为 1 的最小项相或可得下式：

$$\begin{aligned} F &= \overline{A}BC + A\,\overline{B}C + AB\,\overline{C} + ABC \\ &= \sum m(3,5,6,7) \end{aligned}$$

表 9-9 $F = AB + BC + AC$ 的真值表

A	B	C	$F = AB + BC + AC$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

9.3.4 逻辑函数的化简

逻辑函数的表达式和逻辑电路是一一对应的，表达式越简单，用逻辑电路去实现也越简单。

在传统的设计方法中，通常以**与或**表达式定义最简表达式，其标准是表达式中的项数最少，每项含的变量也最少。这样用逻辑电路去实现时，用的逻辑门最少，每个逻辑门的输入端也最少。另外还可提高逻辑电路的可靠性和速度。

在现代设计方法中，多采用可编程的逻辑器件进行逻辑电路的设计。设计并不一定要追求最简单的逻辑函数表达式，而是追求设计简单方便、可靠性好、效率高。但是，逻辑函数的化简仍是需要掌握的重要基础技能。

逻辑函数的化简方法有多种，最常用的方法是逻辑代数化简法和卡诺图化简法。

1. 逻辑代数化简法

逻辑代数化简法就是利用逻辑代数的基本公式和规则对给定的逻辑函数表达式进行化简。常用的逻辑代数化简法有吸收法、消去法、并项法、配项法。

(1) 利用公式：$A + AB = A$，吸收多余的**与**项进行化简。例如

$$F = \overline{A} + \overline{A}BC + \overline{A}BD + \overline{A}E = \overline{A} \cdot (1 + BC + BD + E) = \overline{A}$$

(2) 利用公式：$A + \overline{A}B = A + B$，消去**与**项中多余的因子进行化简。例如

$$\begin{aligned} F &= A + \overline{A}B + \overline{B}C + \overline{C}D = A + B + \overline{B}C + \overline{C}D \\ &= A + B + C + \overline{C}D = A + B + C + D \end{aligned}$$

(3) 利用公式：$A + \overline{A} = 1$，把两项并成一项进行化简。例如

$$\begin{aligned} F &= A\,\overline{BC} + AB + A \cdot (\overline{\overline{BC} + B}) \\ &= A \cdot (\overline{BC} + B + \overline{\overline{BC} + B}) = A \end{aligned}$$

(4) 利用公式：$A + \overline{A} = 1$，把一个**与**项变成两项再和其他项合并进行化简。例如

$$\begin{aligned} F &= \overline{A}B + \overline{B}C + B\,\overline{C} + A\,\overline{B} \\ &= \overline{A}B \cdot (C + \overline{C}) + \overline{B}C \cdot (A + \overline{A}) + B\,\overline{C} + A\,\overline{B} \\ &= \overline{A}BC + \overline{A}B\,\overline{C} + A\,\overline{B}C + \overline{A}\,\overline{B}C + B\,\overline{C} + A\,\overline{B} \\ &= A\,\overline{B} \cdot (C + 1) + \overline{A}C \cdot (B + \overline{B}) + B\,\overline{C} \cdot (\overline{A} + 1) \\ &= A\,\overline{B} + \overline{A}C + B\,\overline{C} \end{aligned}$$

有时对逻辑函数表达式进行化简，可以几种方法并用，综合考虑。例如

$$\begin{aligned} F &= \overline{A}BC + AB\,\overline{C} + A\,\overline{B}C + ABC \\ &= \overline{A}BC + ABC + AB\,\overline{C} + ABC + A\,\overline{B}C + ABC \\ &= AB \cdot (C + \overline{C}) + AC \cdot (B + \overline{B}) + BC \cdot (A + \overline{A}) = AB + AC + BC \end{aligned}$$

在这个例子中就使用了配项法和并项法两种方法。

2. 卡诺图化简法

采用逻辑代数法化简，不仅要求熟练掌握逻辑代数的公式，且需具有较强的化简技巧。卡诺图化简法简单、直观、有规律可循，当变量较少时，用来化简逻辑函数是十分方便的。

(1) 卡诺图

卡诺图其实质是真值表的一种特殊排列形式，二至四变量的卡诺图如图 9-4(a)至(c)图所示。n 个变量的逻辑函数有 2^n 个最小项，每个最小项对应一个小方格，所以，n 个变量的卡诺图

试读结束，需要全本PDF请购买 www.ertongbook.com