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, …, m−1 位；小数部分，自左向右依次为第−1, −2, −3, …, $-n$ 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

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

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

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

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

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

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。

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

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。

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)。

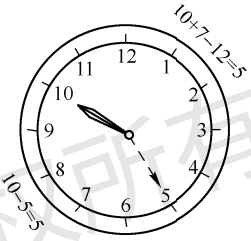


图1-1 说明补码运算原理的实例图

(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。

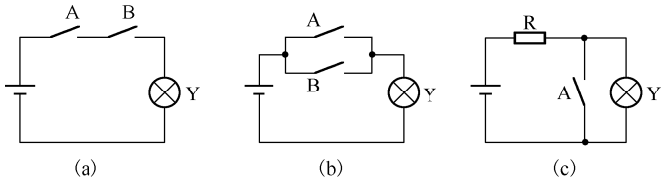


图 1-2　与、或、非逻辑的电路实例图

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 $Y = \overline{A}$。

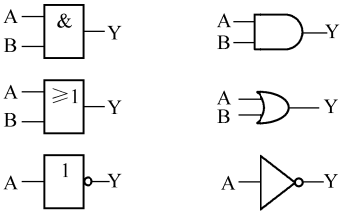
通常称 A 为原变量，$\overline{A}$ 为反变量，二者共同称为互补变量。

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

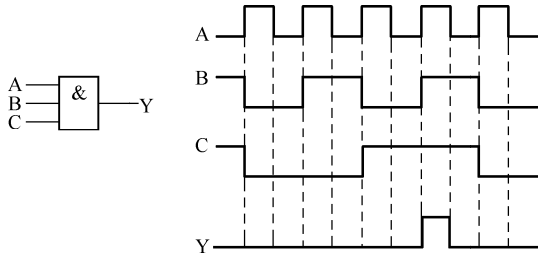


图 1-4　与门波形图

1.3.2　几种复合逻辑运算

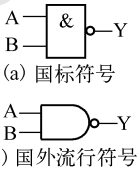
实际的逻辑问题往往要比与、或、非运算复杂得多，它们都可以用与、或、非运算复合而成。有些复合运算大量地、重复地出现，所以也可把它们视为一些基本的运算单元，其中主要有与非、或非、与或非、异或和同或等。

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

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

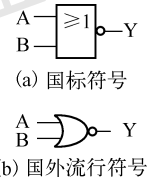
实现与非逻辑运算的门电路叫做与非门。其图形符号如图 1-5 所示，逻辑真值表如表 1-7 所示。



(a) 国标符号

(b) 国外流行符号

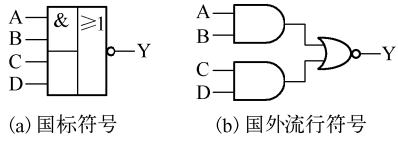
图 1-5　与非门图形符号



(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

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

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

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

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

$$Y=A\odot B=\bar{A}\bar{B}+AB$$

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

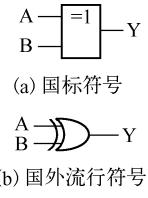
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

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

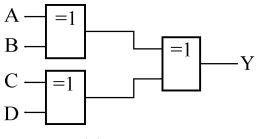
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1



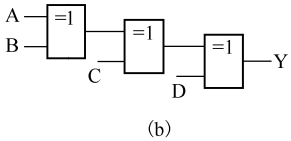
(a) 国标符号

(b) 国外流行符号

图 1-8　异或门图形符号

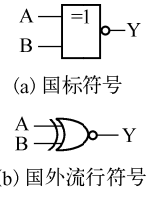


(a)



(b)

图 1-9　多变量异或的实现



(a) 国标符号

(b) 国外流行符号

图 1-10　同或门图形符号

由二变量同或及异或的逻辑真值表可以看出，同或和异或互为反函数。即

$$\overline{A \oplus B} = A \odot B \quad \overline{A \odot B} = A \oplus B$$

工业上一般只生产异或门，所以同或门用“异或非门”来代替。

1.4 逻辑代数的公式和规则

1.4.1 基本公式

基本公式反映了逻辑运算的一些基本规律，如表 1-12 所示。通过列真值表的方法，很容易证明这些基本公式的正确性。也可以通过直观的判断或简单的推理得出大部分公式的正确性。

表 1-12 基本公式

公式名称	序号	公式	序号	公式
0-1 律	1	$0 \cdot A = 0$	11	$1 + A = 1$
自等律	2	$1 \cdot A = A$	12	$0 + A = A$
重叠率	3	$A \cdot A = A$	13	$A + A = A$
互补律	4	$A \cdot \overline{A} = 0$	14	$A + \overline{A} = 1$
交换律	5	$A \cdot B = B \cdot A$	15	$A + B = B + A$
结合律	6	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	16	$A + (B + C) = (A + B) + C$
分配律	7	$A \cdot (B + C) = A \cdot B + A \cdot C$	17	$A + B \cdot C = (A + B) \cdot (A + C)$
反演律	8	$\overline{AB} = \overline{A} + \overline{B}$	18	$\overline{A + B} = \overline{A} \cdot \overline{B}$
还原律	9	$\overline{\overline{A}} = A$		
求反运算规则	10	$\overline{0} = 1; \overline{1} = 0$		

【例 14】 用真值表证明表 1-12 中公式 8 和公式 17 的正确性。

证明：将公式中全部变量的所有可能取值的组合，逐一代入公式的左边和右边，分别计算出相应的结果，即得到如表 1-13 所示的真值表。可见等式两边对应的真值表相同，故等式成立。

表 1-13 例 14 的真值表

公式 8 的真值表

A	B	AB	$\overline{AB}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

公式 17 的真值表

A	B	C	BC	A+B	A+C	A + B·C	(A + B)·(A + C)
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

公式 8 及公式 18 是反演律，也叫摩根定律，它们在逻辑代数中十分重要，能解决逻辑函数的求反问题和逻辑函数的变换问题。

公式 17 是“加”对“乘”的分配律。

通过这 18 个基本公式，可以推导出一些常用公式。

1.4.2 导出公式

几个常用的导出公式如表 1-14 所示。

吸收律 1 的证明：

$$AB + A\overline{B} = A \cdot (B + \overline{B}) = A \cdot 1 = A$$

吸收律 2 的证明：

$$A + AB = A \cdot (1 + B) = A \cdot 1 = A$$

吸收律 3 的证明：

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

吸收律在逻辑函数式的化简中十分有用。吸收律 1 可以将两项合并为一项，并消去一个变量；吸收律 2 可以消去相应的逻辑项；吸收律 3 可将某些项的部分因子消去。

表 1-14　导出公式

公式名称	序　号	公　　式
吸收律 1	1	$AB + A\overline{B} = A$
吸收律 2	2	$A + AB = A$
吸收律 3	3	$A + \overline{A}B = A + B$
多余项定律	4	$AB+\overline{A}C+BC=AB+\overline{A}C$

最后证明多余项定律如下：

$$AB+\overline{A}C+BC=AB+\overline{A}C+BC(A+\overline{A})=AB+\overline{A}C+ABC+\overline{A}BC$$

$$=AB(1+C)+\overline{A}C(1+B)=AB+\overline{A}C$$

注意上述公式均是公式的基本形式，在应用中要学会推广。例如多余项定律可推广为

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

这个推广公式留给读者自己证明。

1.4.3　基本规则

逻辑代数中有 3 个重要的规则，即代入规则、反演规则和对偶规则。它们在逻辑函数式的化简和变换中是十分有用的。

1. 代入规则

将逻辑等式中的某一个逻辑变量全部都用同一个逻辑表达式代替，则等式仍然成立。这就是所谓的代入规则。

这是因为任何一个逻辑表达式和逻辑变量一样，只有 0 和 1 两种取值，所以用逻辑表达式代替逻辑变量后，逻辑等式肯定成立。使用代入规则，可以很容易地证明许多逻辑等式，从而扩大基本公式和导出公式的使用范围。

【例 15】 用代入规则证明摩根定律也适用于 3 变量的情况。

证明： 已知 2 变量的摩根定律为

$$\overline{AB}=\overline{A}+\overline{B} \text{ 及 } \overline{A+B}=\overline{A}\cdot\overline{B}$$

若将左边等式中的 B 以 $B \cdot C$ 代入，同时将右边等式中的 B 以 (B+C) 代入，则有

$$\overline{A(B\cdot C)}=\overline{A}+\overline{(B\cdot C)}=\overline{A}+\overline{B}+\overline{C}\text{，即 }\overline{ABC}=\overline{A}+\overline{B}+\overline{C}$$

$$\overline{A+(B+C)}=\overline{A}\cdot\overline{B+C}=\overline{A}\cdot\overline{B}\cdot\overline{C}\text{，即 }\overline{A+B+C}=\overline{A}\cdot\overline{B}\cdot\overline{C}$$

当对一个乘积项 (即逻辑与表达式) 或逻辑式求反时，应合理地加括号。此外在运算时，仍须遵守像普通代数那样的运算优先顺序，即先算括号里的内容，其次是非运算，接下来是与运算，最后是或运算。

2. 反演规则

由原函数 Y 求其反函数 $\overline{Y}$，称为求反或反演。摩根定律是进行反演的重要工具。但是反复利用摩根定律，求反过程将变得十分烦琐。为此人们提出了重要的反演规则。

对任意一个逻辑函数式 Y，若将其中所有的“·”换成“+”，“+”换成“·”，0 换成 1，1 换成 0，原变量换成反变量，反变量换成原变量，长非符号保持不变，所得到的逻辑函数式就是 $\overline{Y}$。这就是反演规则。

【例 16】 求逻辑函数式 $Y=\overline{A\overline{B}+\overline{C}+D+\overline{C}D}$ 的反函数。

解：根据反演规则可以写出

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

值得注意的是，在求反过程中，应合理地添加括号，以保持原来的运算优先顺序，否则会出错。

3. 对偶规则

对于任意一个逻辑表达式 Y，若将其中所有的“·”换成“+”，“+”换成“·”，0 换成 1，1 换成 0，变量不变，长非符号不变，并保持原来的运算优先级，所得到的逻辑函数式就是 Y^*。且 Y 和 Y^* 互为对偶式。这就是对偶规则。

对偶规则指出，若两个逻辑式相等，则它们的对偶式也相等。

【例 17】 试证明表 1-12 中的基本公式 16，即 $A+(B+C)=(A+B)+C$。

证明：首先写出等式两边的对偶式分别为 $A\cdot(B\cdot C)$ 和 $(A\cdot B)\cdot C$

因为等式 $A\cdot(B\cdot C)=(A\cdot B)\cdot C$ 成立，由对偶规则可知，$A+(B+C)=(A+B)+C$，也成立。

如果仔细研究一下表 1-12 就能发现，其中基本公式（1）和(11)、（2）和(12)、（3）和(13)、（4）和(14)、（5）和(15)、（6）和(16)、（7）和(17)、（8）和(18)皆互为对偶式。因此只要证明公式（1）～（8）成立，则可直接利用对偶规则得出公式(11)～(18)的正确性。

1.5 逻辑关系的表示形式

逻辑关系可用逻辑函数式来表示，除此之外，还有逻辑真值表、逻辑图、波形图和卡诺图等。卡诺图的表示形式将在后面介绍。

1.5.1 逻辑关系的表示形式

（1）逻辑函数式

将输出与输入之间的逻辑关系写成与、或、非等运算的组合形式，就得到了逻辑函数式。

（2）逻辑真值表

根据逻辑关系，将输入变量所有取值所对应的输出值找出来，列成表格，就得到了逻辑真值表。

（3）逻辑图

将逻辑函数式中各变量之间的与、或、非等运算及其组合形式，用相应的图形符号来替代，就可以画出表示该函数关系的逻辑图。

（4）波形图

如果将逻辑函数式中输入变量的每一种可能出现的取值与对应的输出值按照时间顺序依次排列起来，就得到了表示该逻辑函数的波形图。这种波形图也称为时序图。在逻辑分析仪和一些仿真工具中经常以这种波形图的形式给出分析结果。此外，也可以通过实验观察这些波形图，以检验实际逻辑电路的功能是否正确。

1.5.2 各种表示方法之间的相互转换

既然同一种逻辑关系可以用不同的方法描述，那么这些表示方法之间必然能相互转换。这

里重点介绍以下转换。

1. 真值表与逻辑函数式的相互转换

如果已知逻辑函数式，则可很容易列出与之对应的真值表。在前面用真值表证明逻辑代数的基本公式时已经用过列真值表的方法。也就是把输入变量所有取值的组合逐一代入逻辑函数式进行运算，求出输出变量的取值，之后列表，就能得到与之对应的真值表。

反之，如果已知真值表可按如下步骤写出与之对应的逻辑函数式。

（1）从真值表找出所有使输出变量值等于 1 的输入变量的取值的组合。

（2）在上述每组变量的取值下，都会使得一个乘积项(即输入变量的逻辑与表达式)的值为 1。乘积项的写法是，取值为 1 的写其原变量，取值为 0 的写其反变量。

（3）将上述的乘积项相加，即得逻辑函数式。

表 1-15　例 18 的真值表

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

【例 18】 已知真值表如表 1-15 所示，试写出它所对应的逻辑函数式。

解：由真值表可见，当 ABC 取值为 001 时，使得 $\bar{A}\bar{B}C=1$；当 ABC 取值为 010 时，使得 $\bar{A}B\bar{C}=1$；当 ABC 取值为 100 时，使得 $A\bar{B}\bar{C}=1$；当 ABC 取值为 111 时，使得 $ABC=1$。这四个乘积项中的任何一个的取值等于 1 时 Y 都为 1，所以 Y 应为这四个乘积项的和，即

$$Y=\bar{A}\bar{B}C+\bar{A}B\bar{C}+A\bar{B}\bar{C}+ABC$$

2. 逻辑函数式与逻辑图的相互转换

如果已知逻辑函数式，则只要用图形符号代替函数式中的逻辑运算符号，并依照逻辑函数式中的运算优先顺序，将这些图形符号连接起来，就得到了逻辑图。

反之，如果给定了逻辑图，则只要从输入端逐级向输出端写出每个图形符号所表示的逻辑表达式，就可得到与之对应的逻辑函数式了。

1.5.3　逻辑函数式形式的变换

逻辑函数式的形式是多种多样的，同一个逻辑问题可以用不同形式的逻辑函数式来表示。逻辑函数式通常可分为五种形式：与或表达式、与非-与非表达式、与或非表达式、或与表达式及或非-或非表达式。

【例 19】 将与或表达式 $Y=AB+\bar{A}C$ 转换成其他几种形式。

解：（1）转换成与非-与非式：将与或式两次取反，利用摩根定律可得

$$Y=\overline{\overline{AB+\bar{A}C}}=\overline{\overline{AB}\cdot\overline{\bar{A}C}}$$

（2）转换成或与式：首先利用对偶规则求出 Y 的对偶式 Y^*

$$Y^*=(A+B)(\bar{A}+C)$$

然后利用分配律和多余项定律将 Y^* 化简为

$$Y^*=(A+B)(\bar{A}+C)=AC+\bar{A}B+BC=AC+\bar{A}B$$

最后再求 Y^* 的对偶式 Y，即为或与式

$$Y=(A+C)(\bar{A}+B)$$

（3）转换成或非-或非式：方法是，对或与式两次取反，利用摩根定律可得

$$Y=\overline{\overline{(A+C)(\overline{A}+B)}}=\overline{\overline{A+C}+\overline{\overline{A}+B}}$$

（4）转换成与或非式：方法是将或非-或非式继续应用摩根定律可得

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

由此可见，不管以何种形式给出逻辑函数式，总能利用一些方法将其转换为我们需要的形式，并选择相应的逻辑门电路去实现。此外，由于“与或式”的形式与真值表对应明确，并且对与或式形式给出的公式比较熟悉，因此，逻辑函数通常都以“与或式”的形式给出。

1.6 逻辑函数的公式化简法

由实际问题抽象出来的逻辑函数式，通常不是最简式。如果直接采用门电路来实现，既不经济又不可靠。为此，需要对逻辑函数式进行化简，以便用最少的门实现之。例如有两个逻辑功能相同的与或逻辑函数式 $ABC+\overline{B}C+ACD$ 和 $AC+\overline{B}C$，可以看出，前者需要 2 个三输入与门、1 个二输入与门及 1 个或门，而后者只需要 2 个二输入与门及 1 个或门即可。后者显然要比前者实现起来简单得多。

经化简后的与或逻辑函数式，如果其中包含的乘积项最少(所用的与门数最少)，并且每个乘积项中的因子也最少(每个门的输入端数量最少)，则称其为最简与或式。所以化简的目的就是要消去多余的乘积项和每个乘积项中多余的因子。常用的化简方法有公式化简法和下一节要讲的卡诺图化简法。

公式化简法指的是反复利用基本公式和导出公式将给定的逻辑函数式化简，其实并无固定的规律和步骤可循。现将常用的方法归纳如下。

1. 使用吸收律 1

利用表 1-14 中的公式 $AB+A\overline{B}=A$，可以将两项合并为一项，从中消去 B 和 $\overline{B}$ 这一对相反的变量因子，留下共同的因子 A。而且，根据代入规则可知，A 和 B 均可以是任何一个复杂的逻辑式。

【例 20】 试用吸收律 1 化简下列逻辑函数

$$Y_1=A\overline{BC\overline{D}}+ABC\overline{D}$$

$$Y_2=\overline{A}\overline{B}+A\overline{B}+ACD+A\overline{C}D$$

解：

$$Y_1=A(\overline{BC\overline{D}}+BC\overline{D})=A$$

$$Y_2=(\overline{A}+A)\overline{B}+AD(C+\overline{C})=\overline{B}+AD$$

2. 使用吸收律 2

利用表 1-14 中的公式 $A+AB=A$，可以将 AB 项消去。A 和 B 均可以是任何一个复杂的逻辑式。

【例 21】 试用吸收律 2 化简下列逻辑函数

$$Y_1=(\overline{A\overline{B}}+C)ABD+AD$$

$$Y_2=\overline{A}B+\overline{A}B\overline{C}+\overline{A}B\overline{D}$$

解：

$$Y_1=(\overline{A\overline{B}}+C)B\cdot AD+AD=AD$$

$$Y_2=\overline{A}B+\overline{A}B(\overline{C}+\overline{D})=\overline{A}B$$

3. 使用吸收律 3

利用表 1-14 中的公式 $A+\overline{A}B=A+B$，可以消去多余的因子。A 和 B 均可以是任何一个复杂的逻辑式。

【例 22】 试用吸收律 3 化简下列逻辑函数

$$Y_1=\overline{B}+ABC$$

$$Y_2=AC+\overline{A}D+\overline{C}D$$

解：

$$Y_1=\overline{B}+AC$$

$$Y_2=AC+(\overline{A}+\overline{C})D=AC+\overline{ACD}=AC+D$$

4. 使用多余项定律

利用表 1-14 中的公式 $AB+\overline{A}C+BC=AB+\overline{A}C$，可以消去多余的乘积项。A、B 和 C 均可以是任何一个复杂的逻辑式。

【例 23】 试用多余项定律化简下列逻辑函数

$$Y_1=AC+A\overline{B}+\overline{\overline{B}+\overline{C}}$$

$$Y_2=A\overline{B}C\overline{D}+\overline{A\overline{B}}E+C\overline{D}E$$

解：

$$Y_1=AC+A\overline{B}+BC=A\overline{B}+BC$$

$$Y_2=A\overline{B}C\overline{D}+\overline{A\overline{B}}E+C\overline{D}E=A\overline{B}C\overline{D}+\overline{A\overline{B}}E$$

5. 添项法

（1）根据表 1-12 中的基本公式 $A+A=A$，在与或式中重复写入某一项，有时能获得更加简单的化简结果。

【例 24】 试化简逻辑函数 $Y=ABC+\overline{A}BC+A\overline{B}C$。

解：若在逻辑函数式中重复写入 $\overline{A}BC$，则可得到

$$\begin{aligned}Y&=(ABC+\overline{A}BC)+(\overline{A}B\overline{C}+A\overline{B}C)\\&=(A+\overline{A})BC+\overline{A}B(\overline{C}+C)\\&=BC+\overline{A}B\end{aligned}$$

这种方法中，应该添加哪一项是个难题，下一节利用卡诺图进行化简的方法能很好地解决这一难题。

（2）根据表 1-12 中的基本公式 $A+\overline{A}=1$，在某个乘积项上乘以 $(A+\overline{A})$，然后拆成两项后再与其他项合并，有时能获得更加简单的化简结果。

【例 25】 试化简逻辑函数 $Y=\overline{B}C+A\overline{B}+\overline{A}B+B\overline{C}$。

解：对与或式中的第一个和第三个乘积项分别乘以 $(A+\overline{A})$ 和 $(C+\overline{C})$，则

$$\begin{aligned}Y&=(A+\overline{A})\overline{B}C+A\overline{B}+\overline{A}B(C+\overline{C})+B\overline{C}\\&=A\overline{B}C+\overline{A}\overline{B}C+A\overline{B}+\overline{A}BC+\overline{A}B\overline{C}+B\overline{C}\\&=(A\overline{B}C+A\overline{B})+(\overline{A}\overline{B}C+\overline{A}BC)+(\overline{A}B\overline{C}+B\overline{C})\\&=A\overline{B}+\overline{A}C+B\overline{C}\end{aligned}$$

在化简复杂的逻辑函数式时，往往需要灵活、交替地使用上述方法，才能得到最后的化简结果。下面举一个综合性的例子。

【例 26】 试化简逻辑函数 $Y = AC + \bar{B}C + B\bar{D} + C\bar{D} + A(B + \bar{C}) + \bar{A}BC\bar{D} + A\bar{B}DE$ 。

解：

$$\begin{aligned} Y &= AC + \bar{B}C + B\bar{D} + C\bar{D} + A(B + \bar{C}) + \bar{A}BC\bar{D} + A\bar{B}DE \\ &= AC + \bar{B}C + B\bar{D} + C\bar{D} + AB + A\bar{C} + \bar{A}BC\bar{D} + A\bar{B}DE \\ &= (AC + A\bar{C}) + \bar{B}C + B\bar{D} + (C\bar{D} + \bar{A}BC\bar{D}) + AB + A\bar{B}DE \\ &= A + \bar{B}C + B\bar{D} + C\bar{D} + AB + A\bar{B}DE \\ &= (A + AB + A\bar{B}DE) + (\bar{B}C + B\bar{D} + C\bar{D}) \\ &= A + \bar{B}C + B\bar{D} \end{aligned}$$

上述利用公式化简逻辑函数式的例子表明，公式化简法存在以下几个问题：

（1）需要记忆基本公式和导出公式；

（2）化简过程没有固定的模式和规律；

（3）难以判断化简结果是否最简。

为此，利用卡诺图进行逻辑函数式的化简的方法将能很好地解决上述问题。

1.7 逻辑函数的卡诺图及其化简法

在两个乘积项中，除了其中一个变量分别是原变量和反变量之外，其余变量都相同，则称这两个乘积项在逻辑上具有相邻性，或称相邻项。两个相邻项可以利用吸收律 1 进行合并，合并之后消去相异的变量，留下共同的变量。例如：

$$\begin{aligned} Y &= ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} \\ &= AB(C + \bar{C}) + A\bar{B}(C + \bar{C}) \\ &= AB + A\bar{B} = A(B + \bar{B}) = A \end{aligned}$$

对于一个三变量的乘积项，其相邻项有 3 个。例如：$\bar{A}BC$ 的相邻项是 $\bar{A}\bar{B}\bar{C}$ 、 $A\bar{B}C$ 和 $\bar{A}\bar{B}C$ 。

对于 n 个变量的乘积项，则可找到 n 个相邻项。为了方便地找出这些对应关系，我们首先介绍逻辑函数的一种标准形式，然后介绍一种比较直观的图形表示法，即卡诺图法。

1.7.1 逻辑函数式的标准形式

最小项之和是逻辑函数式的一种标准形式。

1. 最小项

所谓最小项是指：在该乘积项中含有逻辑问题的全部输入变量，每个变量均以原变量或反变量的形式出现且只出现一次。

对于两个变量 A、B 来讲，有 $4(=2^2)$个最小项 $\bar{A}\bar{B}$ 、 $\bar{A}B$ 、 $A\bar{B}$ 和 AB 。对于三个变量 A、B、C 来讲，有 $8(=2^3)$个最小项，即 $\bar{A}\bar{B}\bar{C}$ 、 $\bar{A}\bar{B}C$ 、 $\bar{A}B\bar{C}$ 、 $\bar{A}BC$ 、 $A\bar{B}\bar{C}$ 、 $A\bar{B}C$ 、 $AB\bar{C}$ 和 ABC 。由此可知，n 个变量共有 2^n 个最小项。为书写方便，常用“m_i”表示最小项，并按照如下规则确定下标 i 的值：把变量的每一种组合的取值都看成二进制数，与之相对应的十进制数就是 i。例如三变量 A、B、C 最小项的编号表如表 1-16 所示。

同样道理，我们把四个变量 A、B、C、D 的 16 个最小项记作 $m_0 \sim m_{15}$。

由表 1-16 可知最小项具有以下性质：

（1）对于任意一个最小项有且仅有一组变量的取值使它等于 1；

表 1-16　三变量最小项的编号表

			m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	对应的十进制数 i
A	B	C	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}B\bar{C}$	$\bar{A}BC$	$A\bar{B}\bar{C}$	$A\bar{B}C$	$AB\bar{C}$	ABC	
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	1
0	1	0	0	0	1	0	0	0	0	0	2
0	1	1	0	0	0	1	0	0	0	0	3
1	0	0	0	0	0	0	1	0	0	0	4
1	0	1	0	0	0	0	0	1	0	0	5
1	1	0	0	0	0	0	0	0	1	0	6
1	1	1	0	0	0	0	0	0	0	1	7

（2）任意两个最小项的乘积等于 0；

（3）全部最小项的和等于 1；

（4）任意两个相邻最小项之和可以进行合并，之后消去相异的变量留下相同的变量。

性质（4）是卡诺图化简法的重要依据。

2. 逻辑函数的标准形式——最小项之和

一个与或逻辑式中，若乘积项均是最小项，则称其为最小项之和。

任何一个逻辑函数都能展开成最小项之和的标准形式。方法是首先把逻辑函数转化为与或式，然后通过“添项法”（$A+\bar{A}=1$）将每个乘积项中缺少的变量补全，最后进行展开即可。这种标准形式在逻辑函数的化简以及计算机辅助分析和设计中得到了广泛的应用。

【例 27】 将逻辑函数 $Y = AC + \bar{B}C$ 化为最小项之和的标准形式。

解： $Y = AC + \bar{B}C = A(B + \bar{B})C + (A + \bar{A})\bar{B}C$

$$= ABC + A\bar{B}C + A\bar{B}C + \bar{A}\bar{B}C = ABC + A\bar{B}C + \bar{A}\bar{B}C$$

或写作　$Y = m_1 + m_5 + m_7 = \sum m(1,5,7)$

【例 28】 将逻辑函数 $Y = A\bar{B}\bar{C}D + \bar{A}CD + AC$ 化为最小项之和的标准形式。

解：

$$\begin{aligned} Y &= A\bar{B}\bar{C}D + \bar{A}CD + AC \\ &= A\bar{B}\bar{C}D + \bar{A}(B + \bar{B})CD + A(B + \bar{B})C(D + \bar{D}) \\ &= A\bar{B}\bar{C}D + \bar{A}BCD + \bar{A}\bar{B}CD + ABCD + ABC\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} \\ &= m_9 + m_7 + m_3 + m_{15} + m_{14} + m_{11} + m_{10} \\ &= \sum m(3,7,9,10,11,14,15) \end{aligned}$$

1.7.2　逻辑函数的卡诺图化简法

1. 卡诺图

将 n 变量的全部最小项用 2^n 个小方格表示，并使具有逻辑相邻的最小项在几何位置上也相邻地排列成矩形，为此变量的标注采用格雷码，这样的图形就称为 n 变量的卡诺图。它把不直观的逻辑相邻变成了几何图形上的一目了然，使得寻找“可以合并的最小项”工作变得简单易行。

图 1-11 画出了二至五变量最小项的卡诺图。图形两侧标注的 0 和 1 表示使对应小方格内的最小项为 1 的变量取值。同时这些 0 和 1 组成的二进制数所对应的十进制数大小也就是对应的最小项的编号。

为了进一步掌握卡诺图的构造思想，下面将一些共性以及应该注意的地方再说明一下。

（1）n 个变量的卡诺图有 2^n 个小方格，表示 n 变量的 2^n 个最小项。

（2）在卡诺图中，任意相邻的小方格所表示的最小项都仅有一个因子不同，也即几何位置上相邻的最小项，逻辑上也具有相邻性。

（3）在寻找“可以合并的最小项”时，要注意把卡诺图看成是可以左右方向和上下方向闭合的图形。

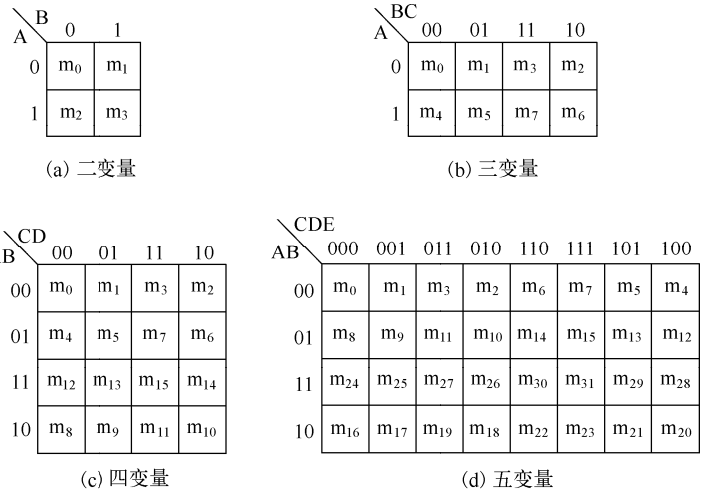


图 1-11　二至五变量最小项的卡诺图

2. 用卡诺图表示逻辑函数

既然任何一个逻辑函数都可以表示为若干个最小项之和的标准形式，那么自然也就可以用卡诺图来表示任何一个逻辑函数。具体方法是：首先将逻辑函数化为最小项之和的标准形式，然后在卡诺图上将标准形式中出现的最小项所对应的小方格内填写 1，在没有出现的最小项所对应的小方格内填写 0，就得到了表示该逻辑函数的卡诺图。也就是说，任何一个逻辑函数都等于它的卡诺图中填入 1 的那些最小项之和。这就是卡诺图与逻辑函数式的对应关系。

【例 29】 用卡诺图表示例 28 的逻辑函数 $Y = A\bar{B}\bar{C}D + \bar{A}CD + AC$。

解： 首先将逻辑函数式 Y 化为最小项之和的标准形式

$$Y = A\bar{B}\bar{C}D + \bar{A}CD + AC = \sum m(3,7,9,10,11,14,15)$$

再画出四变量最小项的卡诺图，最后在出现最小项的小方格内填入 1，其余位置填入 0，即可得到如图 1-12 所示的函数 Y 的卡诺图。

【例 30】 已知逻辑函数 Y 的卡诺图如图 1-13 所示，写出它所对应的逻辑函数式。

解： 因为逻辑函数 Y 等于卡诺图中填入 1 的那些最小项之和，所以有

$$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

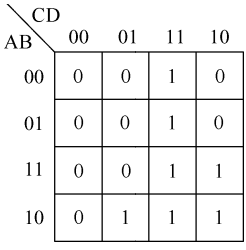


图 1-12　例 29 的卡诺图

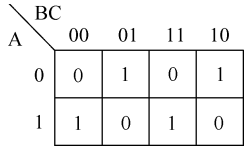


图 1-13　例 30 的卡诺图

3. 利用卡诺图化简逻辑函数

利用卡诺图来化简逻辑函数的方法称为卡诺图化简法。化简的依据是具有相邻性的最小项

可以合并，并消去不同的变量因子，留下共同的变量。由于卡诺图上几何位置相邻与逻辑上相邻是完全一致的，因而从卡诺图上可以非常直观地找出能够合并到一起的相邻最小项，从而达到化简的目的。

（1）合并最小项的原则

两个最小项相邻(即 2^1 个小方格排列成矩形)时，可以消去一个相异的变量；四个最小项相邻(即 2^2 个小方格排列成矩形)时，可以消去两个相异的变量；八个最小项相邻(即 2^3 个小方格排列成矩形)时，可以消去三个相异的变量；2^n 个最小项相邻(即 2^n 个小方格排列成矩形)时，可以消去 n 个相异的变量，n 为自然数。

（2）用卡诺图化简逻辑函数的步骤

① 用卡诺图表示逻辑函数；

② 找出可以合并的最小项，即画出合并圈；

③ 写出合并圈的乘积项，即读出合并圈；

④ 将所有乘积项相加，即得化简结果。

这里，化简的结果是与或式。由最简与或式的定义可知，利用卡诺图进行逻辑函数的化简时，应当使合并圈越大越好(对应的乘积项中的变量个数越少)，并且合并圈的个数越少越好(对应的乘积项的项数越少)。为了达到这样的目的，可以重复圈“1”(因为 A+A=A)，但每个合并圈至少要包含一个“尚未被圈过”的“1”。如果某一个“1”的周围没有可以和它合并成一个矩形合并圈时，则将这一个小方格独自圈出，直接写出它所对应的最小项即可。

此外，由于同一个卡诺图可能会出现几种不同的画出合并圈的方法，所以化简结果可能不唯一，但是化简结果的最简程度应该是一样的。

【例 31】 用卡诺图化简逻辑函数 $Y=\overline{A}\overline{B}\overline{C}D+ACD+\overline{A}B\overline{D}+A\overline{B}$。

解：首先画出表示该逻辑函数 Y 的卡诺图，如图 1-14(a)所示。

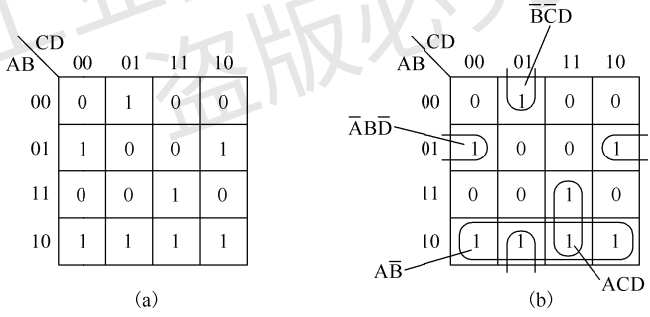


图 1-14　例 31 的卡诺图

其次找出可以合并的最小项，将能够合并到一起的最小项画圈勾出，即画出合并圈，如图 1-14(b)所示。

最后正确读出合并圈的结果，即写出合并圈所对应的乘积项。

将乘积项相加，即得化简最终结果为

$$Y=\overline{B}\overline{C}D+ACD+\overline{A}B\overline{D}+A\overline{B}$$

【例 32】 用卡诺图化简逻辑函数 $Y=A\overline{C}+\overline{A}C+\overline{B}C+B\overline{C}$。

解：第一种画合并圈的方法如图 1-15(a)所示。化简结果为 $Y=\overline{A}C+A\overline{B}+B\overline{C}$。

第二种画合并圈的方法如图 1-15(b)所示。化简结果为 $Y=A\overline{C}+\overline{A}B+\overline{B}C$。

可以看出，虽然化简结果不同，但是最简程度是一样的。这就是逻辑函数化简的不唯一性。

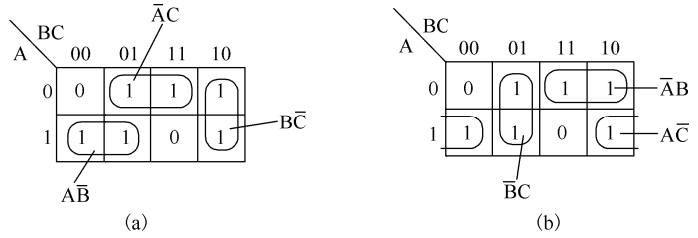


图 1-15　例 32 的卡诺图

【例 33】 用卡诺图化简逻辑函数 $Y=\sum m(0,2,4,5,6,7,8,10,12,13,14,15)$。

解：如图 1-16 所示，画出合并圈之后，得到最简与或式为 $Y=B+\bar{D}$。

补充说明一个问题。上述例子中，都是通过合并卡诺图中出现的“1”来求得逻辑函数 Y 的化简结果的。但是有时也可以通过合并卡诺图中的“0”先求得 $\bar{Y}$，然后再用反演规则求出 Y。依据的是最小项的性质 3，即全部最小项之和为 1。所以，若将全部最小项分成两部分，一部分(卡诺图中填写“1”的那些最小项)之和记作 Y，则根据 $Y+\bar{Y}=1$ 可知，剩余那部分(卡诺图中填写“0”的那些最小项)之和必为 $\bar{Y}$。

例如在图 1-16 所示的卡诺图中，若将“0”进行合并，则画出的合并圈如图 1-17 所示，可得 $\bar{Y}=\bar{B}D$，则 $Y=\overline{(\bar{Y})}=\overline{\bar{B}D}=B+\bar{D}$，与合并“1”得到的化简结果一致。

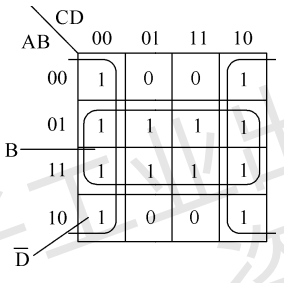


图 1-16　例 33 的卡诺图

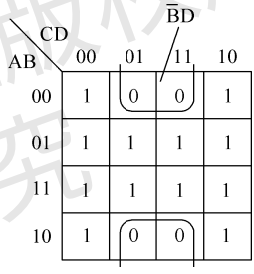


图 1-17　合并“0”得到 $\bar{Y}$

4. 其他函数形式的卡诺图化简

一个逻辑函数可以有许多不同的表达形式。前面介绍的都是最常见的“与或式”的化简，下面简略地介绍一下其他形式的化简方法。

若将逻辑函数化为最简或与式，首先采用合并“0”的方法求得反函数的最简与或式，然后利用反演规则即可求出最终的最简或与式。

【例 34】 将逻辑函数 $Y=\sum m(0,4,5,7,8,12,13,14,15)$ 化为最简或与式。

解：第一步，如图 1-18 所示，先在卡诺图上合并 0，得到 $\bar{Y}$ 的最简与或式为

$$\bar{Y}=\bar{B}D+\bar{B}C+\bar{A}C\bar{D}$$

第二步，再利用反演规则，将 $\bar{Y}$ 变换为 Y，即得最简或与式为

$$Y=(B+\bar{D})(B+\bar{C})(A+\bar{C}+D)$$

若将逻辑函数 Y 化为最简与或非式，仍需先合并“0”，求得反函数的最简与或式，然后直接写出 $Y=\bar{\bar{Y}}$，即可得最终的最简与或非式。

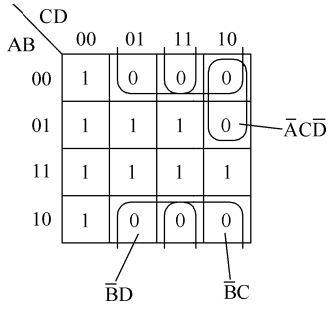


图 1-18　合并“0”得到 $\bar{Y}$

【例 35】 将例 34 的逻辑函数式化为最简与或非式。

解：如例 34 那样，首先在卡诺图上合并 0，得到 $\overline{Y}$ 的最简与或式为

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

然后直接写出 $Y=\overline{\overline{Y}}=\overline{\overline{B}D+\overline{B}C+\overline{A}C\overline{D}}$，即为最简与或非式。

表 1-17　完全描述

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

5. 具有约束条件的逻辑函数的化简

逻辑问题分为完全描述和非完全描述两种，对应于变量的每一组取值，函数都有定义，不是取“0”就是取“1”，如表 1-17 所示。即逻辑函数与每个最小项均有关，这类问题称为完全描述问题。

然而在实际的逻辑问题中，变量的某些取值组合并不允许出现，或者是变量之间具有一种制约关系。我们称这类问题为非完全描述，如表 1-18 所示。该逻辑函数只与部分最小项有关，而与另外一些最小项无关，我们把与逻辑函数无关的最小项称为无关项，有时也叫约束项或任意项。在真值表或卡诺图中常用“×”来表示。

表 1-18　非完全描述

A	B	C	Y
0	0	0	0
0	0	1	×
0	1	0	1
0	1	1	×
1	0	0	1
1	0	1	×
1	1	0	0
1	1	1	1

在约束条件中所包含的最小项就是无关项，常用 d_i 来表示(与最小项之和的标准形式 $\sum m_i$ 写法一致)。如果合理地利用这些无关项，往往能将逻辑函数进一步化简。也就是说，当对化简有利时，将无关项的“×”视作“1”，否则视作“0”即可。

【例 36】 化简具有约束的逻辑函数 $Y=\overline{A}\overline{B}\overline{C}D+\overline{A}BCD+A\overline{B}\overline{C}\overline{D}$，给定的约束条件为

$$\overline{A}\overline{B}CD+\overline{A}B\overline{C}D+AB\overline{C}\overline{D}+A\overline{B}\overline{C}D+ABCD+AB\overline{C}D+A\overline{B}C\overline{D}=0\text{。}$$

解：在用最小项之和的形式表示上述具有约束的逻辑函数时，也可写成如下形式

$$Y(A,B,C,D)=\sum m(1,7,8)+\sum d(3,5,9,10,12,14,15)$$

化简过程如图 1-19 所示。若不利用约束条件，那么 Y 将不能进一步化简。但是适当加进一些无关项，可得到最简结果为 $Y=A\overline{D}+\overline{A}D$。

【例 37】 化简具有约束的逻辑函数 $Y(A,B,C,D)=\sum m(2,8,13)+\sum d(0,7,10)$。

解：化简过程如图 1-20 所示。化简结果为 $Y(A,B,C,D)=AB\overline{C}D+\overline{B}\overline{D}$。

【例 38】 化简具有约束的逻辑函数 $Y(A,B,C)=\sum m(0,5,6)+\sum d(2,4)$。

解：化简过程如图 1-21 所示。化简结果为 $Y(A,B,C)=A\overline{B}+\overline{C}$。

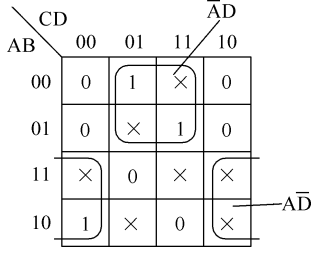


图 1-19　例 36 的卡诺图

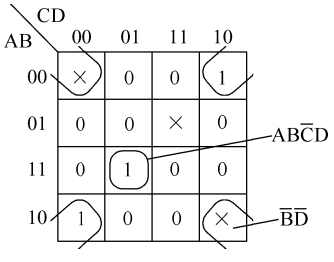


图 1-20　例 37 的卡诺图

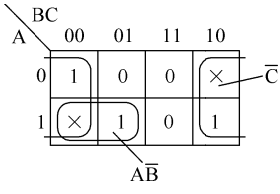


图 1-21　例 38 的卡诺图

*1.8　Multisim 仿真示例

本书每章最后一节所给出的仿真示例，仿真图中的门电路均采用的是特定外形的图形符号。使用时请注意辨别。

【例 39】 逻辑函数式的转换。

解：启动 Multisim 后，屏幕上将显示一个图形用户界面。电路设计工作区是空白的。用户在右侧的仪器仪表工具栏中可以找到“逻辑变换器”的按钮，单击后便随鼠标出现在电路设计工作区，可以看到一个逻辑变换器的图标“XLC1”。双击该图标，屏幕上弹出“逻辑变换器-XLC1”窗口。

在“逻辑变换器-XLC1”窗口中，可以实现的逻辑变换有“逻辑图→真值表”、“真值表→逻辑函数式”、“真值表→最简逻辑函数式”、“逻辑函数式→真值表”、“逻辑函数式→逻辑图”、“逻辑函数式→与非逻辑图”。

在逻辑函数表达式一栏中，键入逻辑函数式 A'B+AB'。变量的非用“'”表示。

（1）“逻辑函数式→真值表”、“逻辑函数式→逻辑图”、“逻辑函数式→与非逻辑图”的变换结果如图 1-22 所示。

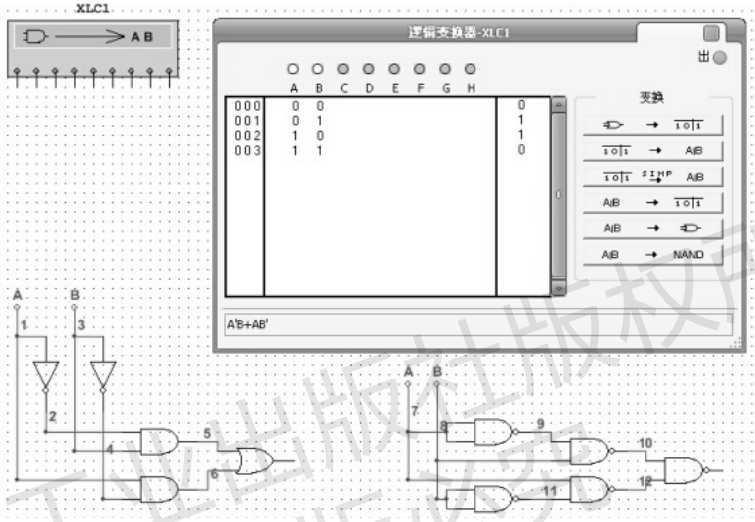


图 1-22　逻辑函数式转换成其他形式

（2）逻辑图与逻辑变换器-XLC1 如图 1-23 所示，之后双击 XLC1，单击按钮“逻辑图→真值表”，即可在窗口中得到该逻辑图所对应的真值表。

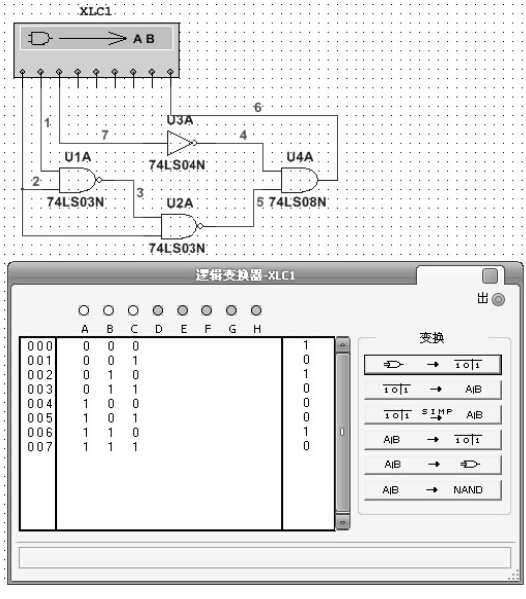


图 1-23　逻辑图转换成真值表

【例 40】 逻辑函数式的化简。

解：（1）在逻辑函数表达式一栏中，键入逻辑函数式 A'B + AC + BC。首先进行由“逻辑函数式→真值表”的转换，然后进行由“真值表→最简逻辑函数式”即可。如图 1-24 所示，即 A'B + AC + BC = A'B + AC。

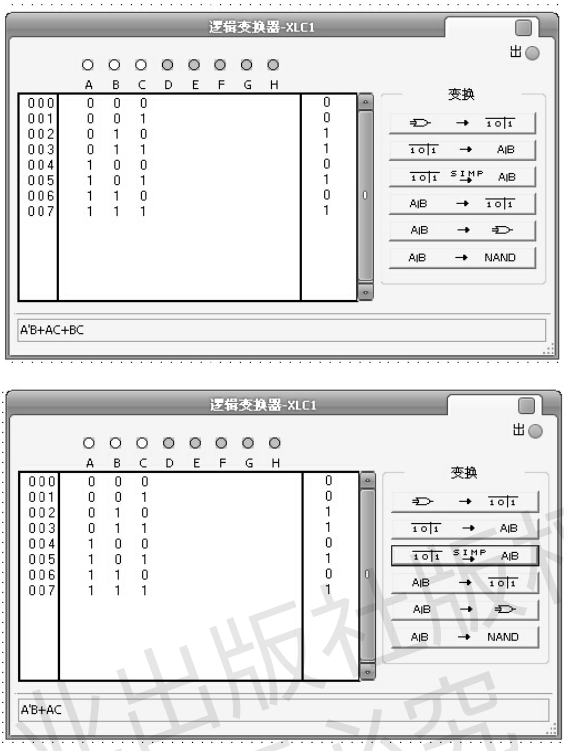


图 1-24　逻辑函数式的化简

（2）具有无关项的逻辑函数表达式为

$$Y(A,B,C,D)=\sum m(2,8,13)+\sum d(0,7,10)$$

首先在“逻辑变换器-XLC1”窗口中，列出四变量的真值表，然后在输出列，把出现的最小项通过鼠标点出“1”，出现的约束项点出“×”，其余的不做修改，即为“0”。

接下来单击“真值表→最简逻辑函数式”如图 1-25 所示，即可完成化简，化简结果为

$$Y(A,B,C,D)=\overline{B}\overline{D}+AB\overline{C}D$$

图 1-25　含有无关项的逻辑函数式的化简

习题

1-1　为了对 300 份文件进行顺序编号，若采用二进制代码，最少需要用几位？若采用八进制或十六进制代码，则最少各需要用几位？

1-2　将下列二进制数转换为等值的十进制数、八进制数和十六进制数。

（1）$(1011.01)_B$　（2）$(101100010.011)_B$

1-3　写出下列 BCD 码所代表的十进制数。

（1）$(100010010010.01111001)_{8421BCD}$　（2）$(11001010.01111001)_{余3码}$　（3）$(11001011.00111100)_{5421BCD}$

1-4　用二进制补码运算求下列各式的计算结果，假设机器字长为 8 位。

（1）+21−7　（2）−34+8　（3）106+21　（4）−106−21

1-5　写出下列二进制代码的奇校验位。

（1）1010101　（2）100100100　（3）1111110

1-6　证明下列逻辑等式。

（1）$A\oplus B\oplus C=A\odot B\odot C$　　（2）$ABC+\overline{A}\overline{B}\overline{C}=\overline{A\overline{B}+B\overline{C}+\overline{A}C}$

（3）$(AB+C)B=AB\overline{C}+\overline{A}BC+ABC$　　（4）$\overline{A}B+A\overline{B}=(\overline{A}+\overline{B})(A+B)$

（5）$ABC+\overline{A}+\overline{B}+\overline{C}=1$

1-7　写出下列逻辑函数的对偶式及反函数。

（1）$F=\overline{A}\overline{B}+CD$　　（2）$F=\overline{A+B+\overline{C}+\overline{D+E}}$

（3）$F=A\overline{BC}+(\overline{A}+\overline{B}\overline{C})\cdot(A+C)$　　（4）$F=(A+B+C)\overline{A}\overline{B}\overline{C}=0$

（5）$F=AB+\overline{C}\overline{D}+\overline{BC+\overline{D}+\overline{C}E+\overline{D+E}}$

1-8　已知逻辑函数 Y(A,B,C)的波形图如图 1-26 所示，试写出 Y 的逻辑函数式。

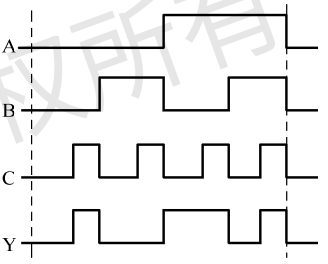


图 1-26　习题 1-8 的图

1-9　用公式化简法将下列逻辑函数化简成最简与或式。

（1）$F=ABC+\overline{A}+\overline{B}+\overline{C}$　　（2）$F=A\overline{B}+A\overline{C}+B\overline{C}+A\overline{B}\overline{C}+AB\overline{C}D$

（3）$F=AB+ABD+\overline{A}C+BCD$　　（4）$F=(A\oplus B)\overline{AB+\overline{A}\overline{B}}+AB$

（5）$F=A(\overline{A}+B)+B(B+C)+B$　　（6）$F=\overline{\overline{AB+\overline{A}\overline{B}BC+\overline{B}\overline{C}}}$

（7）$F=\overline{\overline{AC+\overline{B}\overline{C}}+B(A\overline{C}+\overline{A}C)}$　　（8）$F=A\overline{C}\overline{D}+BC+\overline{B}D+A\overline{B}+\overline{A}C+\overline{B}\overline{C}$

1-10　将下列逻辑函数化为最小项之和的标准形式。

（1）$F=\overline{A}BC+AC+\overline{B}C$　　（2）$F=A\overline{B}CD+A\overline{C}D+\overline{A}D$

（3）$F=A+BC+CD$　　（4）$F=AB+\overline{B\overline{C}}\,\overline{\overline{C}+\overline{D}}$

（5）$F=A\overline{B}+B\overline{C}+\overline{A}C$　　（6）$F=(A\oplus B)(C\odot D)$

1-11　用卡诺图将习题 1-9 中各题化简成最简与或式并用相应门实现。

1-12　将习题 1-10 中各题化简成最简“与或非”式、“与非”式、“或非”式。

1-13　用卡诺图化简法将下列具有无关项的逻辑函数化为最简与或式。

（1）$F(A,B,C)=\sum m(0,1,2,4)+\sum d(5,6)$　　（2）$F(A,B,C)=\sum m(1,2,4,7)+\sum d(3,6)$

（3）$F(A,B,C,D)=\sum m(3,5,6,7,10)+\sum d(0,1,2,4,8)$　（4）$F(A,B,C,D)=\sum m(2,3,7,8,11,14)+\sum d(0,5,10,15)$

（5）$F(A,B,C,D)=\sum m(2,3,4,5)+\sum d(10,11,12,13)$