第 1 章　数字逻辑基础

本章主要介绍计数体制、常用编码、二极管及三极管的开关特性和逻辑代数基础。这些内容是学习其他有关章节的基础，是研究逻辑电路的重要数学工具。下面分别进行介绍。

1.1　计数体制

在日常生活中人们习惯于使用十进制数，而在数字系统中常采用二进制数。本节首先从人们最熟悉的十进制数开始分析，进而引出各种不同的进位计数制。

1.1.1　十进制数

一个十进制数具有两个特点，一是用 10 个不同的数字符号 0，1，2，3，4，5，6，7，8，9 来表示，通常把这 10 个数字符号称为数码；二是它逢“十”进位。因此，同一个数码在一个数中处在不同的位置（或数位）代表的数值是不同的。例如，6666.66 这个数中，小数点左边的第 1 位代表个位，它的权值为 10^0，就是它本身的数值 6（或 6×10^0）；小数点左边第 2 位代表十位，它的数值为 6×10^1；小数点左边第 3 位代表百位，它的数值为 6×10^2；小数点左边第 4 位代表千位，它的数值为 6×10^3；而小数点右边第 1 位的权值为 10^{-1}，它的数值为 6×10^{-1}；而小数点右边第 2 位的权值为 10^{-2}，它的数值为 6×10^{-2}。因此，这个数可以写成：

$$6666.66=6\times10^3+6\times10^2+6\times10^1+6\times10^0+6\times10^{-1}+6\times10^{-2}$$

式中，6，6，6，6，6，6 这些数码均称为系数；10^3，10^2，10^1，10^0，10^{-1}，10^{-2} 是每位数对应的权，这里 10 称为十进制数的基数，权乘以系数称为加权系数，所以一个十进制数的数值就是以 10 为基数的加权系数之和。任意一个十进制数 M_{10} 都可以表示为

$$\begin{aligned}M_{10}&=a_{n-1}\times10^{n-1}+a_{n-2}\times10^{n-2}+\cdots+a_1\times10^1+a_0\times10^0\\&\quad+a_{-1}\times10^{-1}+a_{-2}\times10^{-2}+\cdots+a_{-m}\times10^{-m}\\&=\sum_{i=-m}^{n-1}a_i\times10^i\end{aligned}$$

式中，i 表示数中的第 i 位；a_i 表示第 i 位的数码（系数），它可以是 0～9 这 10 个数码中的任意一个；n 和 m 为正整数，n 为小数点左边的位数，m 为小数点右边的位数；10 为计数制的基数；M 的下标为 10，表示 M 是一个十进制数。基数和 M 的下标是一致的。如果 M 是 R 进制数，则写成 M_R。以 R 为基数的 n 位整数、m 位小数的 R 进制数，其按权展开式可写为

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

1.1.2　二进制数

与十进制数类似，二进制数也有两个主要特点：一是用两个不同的数字符号 0 和 1 来表示；二是它逢“二”进位，当 1＋1 时，本位为 0，向高位进 1（1＋1＝10）。因此，同一个数码在不

同的数位所代表的值也是不同的。例如

$$(1001)_2=1\times2^3+0\times2^2+0\times2^1+1\times2^0=(8+0+0+1)_{10}=(9)_{10}$$

$$(11011.101)_2=1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$
$$=(27.625)_{10}$$

任意一个二进制数 M_2 都可表示为

$$M_2=a_{n-1}\times2^{n-1}+a_{n-2}\times2^{n-2}+\cdots+a_1\times2^1+a_0\times2^0$$
$$+a_{-1}\times2^{-1}+a_{-2}\times2^{-2}+\cdots+a_{-m}\times2^{-m}$$
$$=\sum_{i=-m}^{n-1}a_i\times2^i$$

式中，a_i 只能是 0 或 1；n 和 m 为正整数，n 为小数点左面的位数，m 为小数点右面的位数；2 是进位制的基数，故称二进制数。

在数字系统中采用二进制数是比较方便的，由于二进制数只有两个数码 0 和 1，因此，它的每一位数都可以用某些元件所具有的两种不同的稳定状态来表示，如三极管的饱和导通与截止。某些器件输出电压有低与高两种稳定状态，只要用其中一种状态表示 1，而用另一种状态表示 0，就可以表示二进制数了。

1.1.3 八进制数和十六进制数

1. 八进制数

八进制数有两个特点：一是用 8 个数码符号 0，1，2，3，4，5，6，7 来表示数值；二是逢“八”进位，即 7+1=10。

任意一个八进制数 M_8 都可以表示为

$$M_8=\sum_{i=-m}^{n-1}a_i\times8^i$$

式中，a_i 可取 0～7 这 8 个数码符号之中的任意一个；n 和 m 为正整数，n 为小数点左边的位数，m 为小数点右边的位数；8 为基数，故称八进制数。

2. 十六进制数

十六进制数也有两个特点：一是用 16 个数码符号 0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F 来表示数值；二是逢“十六”进位，即 F+1=10。它的表达式为

$$M_{16}=\sum_{i=-m}^{n-1}a_i\times16^i$$

式中，a_i 可取 0～F 这 16 个数码符号之中的任意一个；n 和 m 为正整数，n 为小数点左边的位数，m 为小数点右边的位数；16 为基数，故称十六进制数。

综上所述，4 种计数制的特点类似，可以概括如下：

(1) 每一种计数制都有一个固定的基数 R，它的每一位可取 R 个数码符号中的任意一个数码。

(2) 它们是逢 R 进位的。因此，它的每一个数位 i，对应一个固定的值 R^i，R^i 就是该位的“权”，小数点左边各位的权依次是基数 R 的正次幂；而小数点右边各位的权依次是基数 R 的

负次幂。显然，若将一个数中的小数点向左移一位，则等于将该数减小为 $1/R$；若将小数点向右移一位，则等于将该数增大 R 倍。

1.1.4 数制间的转换

1. 二进制数与十进制数之间的转换

(1) 二进制数转换成十进制数

通常的方法是用加权系数之和求得。例如

$$\begin{aligned} M_2 &= (11011.101)_2 \\ &= 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3} \\ &= (27.625)_{10} \end{aligned}$$

(2) 十进制数转换成二进制数

把十进制数 25.625 转换成二进制数，其方法是：把数的整数部分连续除以 2(直至商为 0)取余数作为二进制数整数；小数部分连续乘以 2(直至积为 1)取整数作为二进制数小数。例如

$$\begin{array}{r|l} 2 & 25 \\ 2 & 12 \quad \text{余 } 1 = a_0 \\ 2 & 6 \quad \text{余 } 0 = a_1 \\ 2 & 3 \quad \text{余 } 0 = a_2 \\ 2 & 1 \quad \text{余 } 1 = a_3 \\ & 0 \quad \text{余 } 1 = a_4 \end{array}$$

$$\begin{array}{rl} & 0.625 \\ \times & 2 \\ \hline & 1.250 \quad a_{-1} = 1 \\ & 0.250 \\ \times & 2 \\ \hline & 0.500 \quad a_{-2} = 0 \\ & 0.500 \\ \times & 2 \\ \hline & 1.000 \quad a_{-3} = 1 \end{array}$$

则 $(25.625)_{10} = (11001.101)_2$。

2. 二进制数与八进制数之间的转换

(1) 二进制数转换成八进制数

把二进制数 101011011.110101110 转换成八进制数要分别对整数和小数进行转换。整数的转换可从最低位(小数点左边第一位)开始，每 3 位分为一组，每组用 1 位等价八进制数来替代；小数的转换可从小数点右边第一位开始，每 3 位分为一组，最后不足 3 位的补零，然后顺序写出对应的八进制数即可。例如

$$\begin{array}{ccccccccc} 101 & & 011 & & 011 & . & 110 & & 101 & & 110 \\ 5 & & 3 & & 3 & . & 6 & & 5 & & 6 \end{array}$$

则 $(101011011.110101110)_2 = (533.656)_8$。

(2) 八进制数转换成二进制数

八进制数转换成二进制数，只要将每位八进制数用等价的 3 位二进制数表示即可。

例如，$(564.321)_8 = (101110100.011010001)_2$。

3. 二进制数与十六进制数之间的转换

(1) 二进制数转换成十六进制数

二进制数转换成十六进制数，其方法是：将二进制数的整数部分由小数点向左，每 4 位分一组，最后不足 4 位的前面补零；小数部分由小数点向右，每 4 位分一组，最后不足 4 位的后面

补零，然后把每 4 位二进制数用等价的十六进制数来代替，即可转换为十六进制数。例如，$(1101110.1101110)_2$ 转换成十六进制数：

0110　1110　.　1101　1100

6　　E　　.　　D　　C

则 $(1101110.1101110)_2=(6E.DC)_{16}$。

(2) 十六进制数转换成二进制数

转换方法与上述过程相反，每位十六进制数用 4 位二进制数替换即可。例如，$(1BE3.97)_{16}$ 转换成二进制数，其转换过程如下：

1　　B　　E　　3　　.　　9　　7

0001　1011　1110　0011　.　1001　0111

则 $(1BE3.97)_{16}=(1101111100011.10010111)_2$。

1.2　常用编码

什么是编码？一般来说，用文字、符号或者数码来表示某种信息(数值、语言、操作命令、状态)的过程称为编码。在数字系统或计算机中是用多位二进制码按照一定规律来表示某种信息的。这些多位二进制码称为代码，编码后的代码都具有一定的含义。因为二进制代码只有 0 和 1 两个数字，所以电路上实现起来最容易。

1.2.1　二-十进制编码(BCD 码)

十进制数是用 0～9 这 10 个数字符号组成的，为此可用 4 位二进制码的 16 种组合作为代码，取其中 10 种组合来表示 0～9 这 10 个数字符号。通常，把用 4 位二进制数码来表示一位十进制数称为二-十进制编码，也称为 BCD 码。取哪 10 种组合来表示 10 个数字符号有很多种方案，这就形成了各种不同的 BCD 码，常用的几种 BCD 码列于表 1-1 中。

表 1-1　常用的几种 BCD 码

编码种类 / 十进制数	8421 码	余 3 码	2421 码 (A)码	2421 码 (B)码	5421 码	余 3 循环码
0	0000	0011	0000	0000	0000	0010
1	0001	0100	0001	0001	0001	0110
2	0010	0101	0010	0010	0010	0111
3	0011	0110	0011	0011	0011	0101
4	0100	0111	0100	0100	0100	0100
5	0101	1000	0101	1011	1000	1100
6	0110	1001	0110	1100	1001	1101
7	0111	1010	0111	1101	1010	1111
8	1000	1011	1110	1110	1011	1110
9	1001	1100	1111	1111	1100	1010
权	8421		2421	2421	5421	

1. 8421 码

8421 码是使用最多的一种 BCD 码，它是一个有权码，其各位的权分别是(从最高有效位开始至最低有效位)8，4，2，1。如果把每一个代码都看成是一个 4 位二进制数，这个代码的数值恰好等于它所代表的十进制数的大小。

2. 余 3 码

因为每一个余 3 码所对应的 4 位二进制数的数值要比它所表示的十进制数恰好多 3，所以这种编码称为余 3 码。从编码表中可以看到：0 和 9，1 和 8，2 和 7，3 和 6，4 和 5，这 5 对代码是互补的。例如，2 中的 0 变 1，1 变 0 就可得到 7；7 中的 0 变 1，1 变 0 就可得到 2。这种互补性有利于进行减法运算，在此不进行讨论。

1.2.2 循环码

4 位循环码如表 1-2 所示。从表中可以看到，相邻两组代码间只有一位取值不同，而其他位均相同。再有，每一位代码从上到下的排列顺序都是以固定的周期进行循环的，右起第 1 位的循环周期是 0110、第 2 位的循环周期是 00111100、第 3 位的循环周期是 0000111111110000 等。从表中还可以看到，以 16 种组合的中间为轴，最高位由 0 变 1，其余 3 位均以轴为对称组合。这种编码是一种无权码，又称为反射码或格雷码。

表 1-2 4 位循环码

十进制数	循环码
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	0 1 1 1
6	0 1 0 1
7	0 1 0 0
8	1 1 0 0
9	1 1 0 1
10	1 1 1 1
11	1 1 1 0
12	1 0 1 0
13	1 0 1 1
14	1 0 0 1
15	1 0 0 0

1.2.3 ASCII 码

ASCII 码是美国信息交换标准代码的简称(American Standard Code for Information Interchange)。它的编码表如表 1-3所示，它是一组 7 位代码，用来表示十进制数、英文字母及专用符号。

表 1-3 ASCII 码表

$b_7b_6b_5$ \ $b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0 0 0 0	NUL	DLE	SP	0	@	P	\	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y

续表

$b_7b_6b_5$ \ $b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	!
1 1 0 1	CR	GS	−	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	↑	n	~
1 1 1 1	SI	US	/	?	O	↓	o	DEL

1.3 二极管和三极管的开关特性

1.3.1 二极管的开关特性

在数字电路中，二极管经常工作在开关状态（导通和截止状态交替工作）下，下面简述二极管在开关状态下的工作特点。

1. 二极管导通条件及导通时的特点

在硅二极管两端外加正向电压 V_F，并把 $V_F \geqslant 0.7V$ 作为硅二极管的导通条件，近似认为二极管一旦导通，它的正向管压降 V_D 保持在 0.7V 不变，相当于一个开关闭合后有一个0.7V的压降（锗管为 0.2V）。这就是把二极管导通看作开关闭合时的特点。

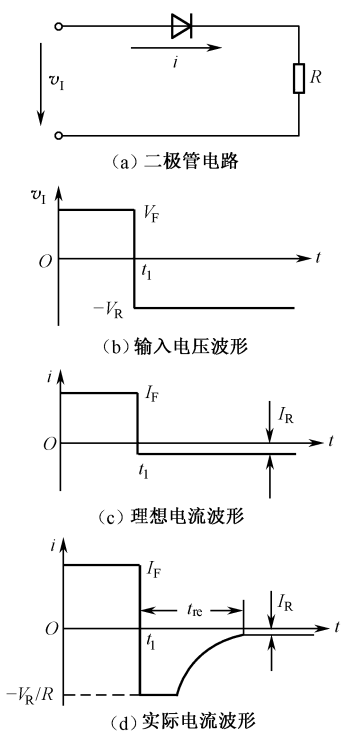


图 1-1 二极管开关特性

2. 二极管截止条件及截止时的特点

当硅二极管两端外加的正向电压 V_F 小于死区电压 0.5V（或外加反向电压 V_R）时，二极管电流 I_D 很小（$I_D \approx 0$），相当于硅二极管截止，所以把 $V_F \leqslant 0.5V$ 作为硅二极管的截止条件（锗管 $V_F \leqslant 0.1V$）。硅二极管处于截止状态时，其电流 I_D 可看成零，相当于开关断开。这就是把硅二极管截止看作开关断开时的特点。

3. 二极管反向恢复时间 t_{re}

如图 1-1(a) 所示二极管电路，其输入电压 v_I 为如图 1-1(b)所示的矩形波。在 $0 \sim t_1$ 的时间内输入电压为 $+V_F$，二极管正向导通，有正向电流 I_F 流过。在 t_1 时刻输入电压由 $+V_F$ 跳变到 $-V_R$，在理想情况下，二极管应立刻截止，只有很小的反向饱和电流 I_R，电流 i 的波形如图 1-1(c)所示。两种状态的相互转换不需要时间。但实际情况是，二极管并不会立刻截止，而是仍然导通的，在 t_1 时

刻产生一个很大的反向电流，只有经过时间 t_{re} 后二极管才恢复到截止状态。电流 i 的实际波形如图 1-1(d)所示。时间 t_{re} 称为反向恢复时间。

为什么会出现很大的反向电流呢？原因是当二极管正向导通时，多数载流子不断向对方区域扩散，在 PN 结的两侧存储大量扩散过去的载流子，如图 1-2 所示。P 区中的多数载流子空穴扩散到 N 区后，成为 N 区中的少数载流子；N 区中的多数载流子电子扩散到 P 区后，成为 P 区中的少数载流子。因此，一旦外加电压反向时，它们就会形成较大的反向漂移电流 V_R/R，如图 1-1(d)所示。只有经过一段反向恢复时间，在 PN 结的两侧存储的载流子消散后，二极管才进入截止状态。

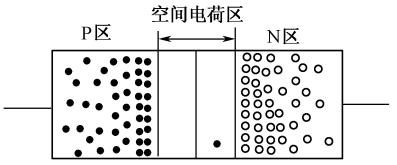


图 1-2　P 区和 N 区少数载流子存储情况

反向恢复时间 t_{re} 一般为纳秒(ns)数量级，它是开关二极管特有的参数，用来衡量开关速度的快慢。t_{re} 值越小，开关速度越快，允许的信号频率越高。

1.3.2　三极管的开关特性

在数字电路中，三极管经常工作在截止状态(相当于开关断开)和饱和状态(相当于开关闭合)下，并且经常在这两个状态之间进行快速转换。我们把这种工作情况称为三极管工作在开关状态下。

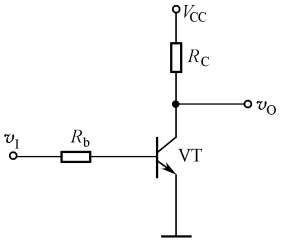


图 1-3　三极管开关电路

1. 截止、饱和的条件

三极管开关电路如图 1-3 所示。设电路的输入电压 $v_I = V_{IL}$ 时，$V_{BE} < 0$V(一般 $V_{BE} < 0.5$V 时三极管就截止)，$I_B = 0$，$I_C = 0$，$v_O = V_{OH} = V_{CE} = V_{CC}$，三极管截止，相当于开关断开。因此，把 $V_{BE} < 0$V(或 $V_{BE} < 0.5$V)作为三极管截止的条件。设电路的输入 $v_I = V_{IH}$ 使三极管饱和导通。当 $V_{CE} = V_{BE}$ 时，三极管为临界饱和导通，则 I_C 用 I_{CS} 表示，$I_{CS} = (V_{CC} - 0.7)/R_C \approx V_{CC}/R_C$ 称为集电极临界饱和电流，$I_{BS} = I_{CS}/\beta \approx V_{CC}/(\beta R_C)$ 称为基极临界饱和电流。当 $I_B > I_{BS}$ 时，三极管工作在饱和状态下，一般 $V_{CES} = 0.1 \sim 0.3$V，由于三极管的 C，E 之间电压很小，相当于开关闭合，因此，把 $I_B > I_{BS}$ 作为三极管饱和的条件。

2. 三极管的开关时间

三极管的开关时间如图 1-4 所示。三极管截止与饱和两种状态相互转换的过程，也需要一定时间，该时间就是三极管中电荷的建立与消散过程所需的时间。所以集电极电流 i_C 和输出电压 v_O 的变化总是滞后于输入电压 v_I 的变化。

三极管由截止到饱和导通所需要的时间，称为开启时间，用 t_{on} 表示，$t_{on} = t_d + t_r$。t_d 称为延迟时间，它是从 v_I 的正跳变至 i_C 上升到 $0.1I_{CS}$ 所需的时间；t_r 称为上升时间，它是 i_C 从 $0.1I_{CS}$ 上升到 $0.9I_{CS}$ 所需的时间。所以 t_{on} 是三极管发射结由宽变窄以及基区建立电荷所需要的时间。

三极管由饱和导通到截止所需要的时间，称为关闭时间，用 t_{off} 表示，$t_{off} = t_s + t_f$。t_s 称为

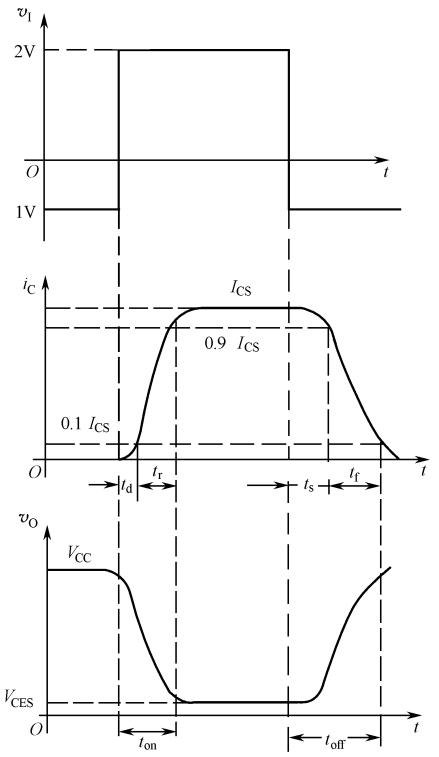


图 1-4　三极管的开关时间

存储时间，它是从 v_I 的负跳变至 i_C 下降到0.9I_{CS}所需的时间；t_f 称为下降时间，它是 i_C 从0.9I_{CS}下降到0.1I_{CS}所需的时间。t_{off} 主要是清除三极管内存储电荷的时间。

三极管的开关时间一般为纳秒(ns)数量级，并且 $t_{off}>t_{on}$，$t_s>t_f$，因此 t_s 的大小是决定三极管开关速度的最主要的参数。

1.4　逻辑代数基础

逻辑代数是 1847 年英国数学家乔治·布尔(George Boole)首先创立的，所以有时把逻辑代数称为布尔代数。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量大小之间的关系，而是表示逻辑变量之间的逻辑关系。它是分析和设计数字电路的基本数学工具。

1.4.1　逻辑变量和逻辑函数

“逻辑”一词始于逻辑学。逻辑学研究的是逻辑思维与逻辑推理的规律。数字电路也是研究逻辑的，即研究数字电路的输入、输出的因果关系，也就是研究输入和输出间的逻辑关系。为了对输入和输出间的逻辑关系进行数学表达和演算，提出了逻辑变量和逻辑函数两个术语。一个逻辑电路框图如图 1-5 所示，A，B 为输入，F 为输出，输出和输入之间的逻辑关系可表示为 F＝f(A，B)。这种具有逻辑属性的变量称为逻辑变量。A，B 称为逻辑自变量，F 是逻辑因变量。当 A，B 的逻辑取值确定后，则 F 的逻辑值也就唯一地被确定下来，通常称 F 是 A，B 的逻辑函数。所以输出变量 F 又称为逻辑函数。F＝f(A，B)称为逻辑函数表达式。逻辑变量(自变量)和逻辑函数(因变量)的逻辑取值，只取 0，1 两个值，通常称为逻辑 0 和逻辑 1，以区别于数字 0 和 1。逻辑 0 或 1 表示两种对立的状态，表示信号的无或有，电平的低或高，电路的截止或导通，开关的断开或接通，一件事情的是或非，真或假等。

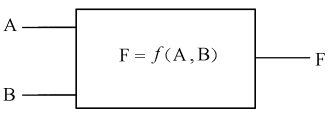


图 1-5　逻辑电路框图

在逻辑电路中，逻辑 0 和逻辑 1 是用电平的高和低来表示的。如果用高电平表示逻辑 1 而用低电平表示逻辑 0，则称为正逻辑体制(简称正逻辑)。如果用低电平表示逻辑 1 而用高电平表示逻辑 0，则称为负逻辑体制(简称负逻辑)。本书中如无特殊说明，一律采用正逻辑体制。

在逻辑电路中，电位常用电平这一术语来描述。高、低电平表示的是两种不同的状态，它们表示的都是一定的电压范围，而不是一个固定不变的值。例如在 TTL 电路中，常规定高电平的额定值为 3V，低电平的额定值为 0.2V，而从 0V 到 0.8V 都算作低电平，从 2V 到 5V 都算作高电平。

为了加深对逻辑关系的理解，下面通过图 1-6 所示开关控制电路的例子加以说明。

图 1-6 中，A，B 为单刀双掷开关，F 为电灯，开关 A，B 的上合或下合与灯 F 的亮与灭有何因果关系呢？设 A，B 向上合为 1，向下合为 0；灯 F 亮为 1，灭为 0。F 与 A，B 间的逻辑关系如表 1-4所示。此表描述了灯 F 和开关 A，B 的状态之间真实的逻辑关系，这个表又称为真值表。从表中变量和函数的取值可以看出只有 0 和 1 两个逻辑值。当 A 和 B 的取值相同（全向上合或全向下合）时，F 才亮；当 A 和 B 的取值不相同（A 向上合，B 向下合；A 向下合，B 向上合）时，F 就灭。这种逻辑关系写成函数表达式为 $F=f(A,B)=A\cdot B+\overline{A}\cdot\overline{B}$。表 1-4 中 1 用原变量 A，B 及原函数 F 表示，0 用反变量 $\overline{A}$（A 的反）、$\overline{B}$（B 的反）表示。函数表达式中“A · B”表示 1 · 1（全向上合），是 A，B 相与（ · ）关系；“$\overline{A}\cdot\overline{B}$”表示 0 · 0（全向下合），是 $\overline{A}$，$\overline{B}$ 相与（ · ）关系；符号“＋”表示 A，B“全向上合”或“全向下合”两种情况中有一种情况存在 F 就亮，“＋”表示或的关系；“$\overline{A}$”，“$\overline{B}$”分别表示 A 的非和 B 的非，为非的关系。

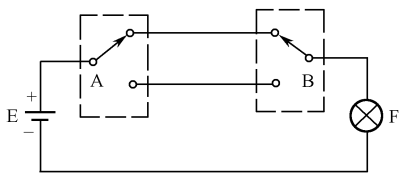


图 1-6　开关控制电路

表 1-4　F 与 A，B 间的逻辑关系

输入自变量 A　B	输出函数 F	说　明
0　0	1	A 与 B 均向下，F 亮
0　1	0	A 向下 B 向上，F 灭
1　0	0	A 向上 B 向下，F 灭
1　1	1	A 与 B 均向上，F 亮

此例的逻辑函数表达式 $F=A\cdot B+\overline{A}\cdot\overline{B}$，说明了逻辑代数有与、或、非三种基本逻辑运算。式中的“ · ”为与运算符号，“＋”为或运算符号，“－”为非运算符号。下面分别说明三种基本逻辑运算的含义及如何实现这三种基本运算。

1.4.2　基本逻辑运算及基本逻辑门

逻辑与、逻辑或、逻辑非是逻辑代数中的三种基本逻辑运算。实现这三种逻辑运算的电路，分别称为与门、或门、非门，这三种门是基本逻辑门。下面用三种指示灯的控制开关电路来说明三种基本逻辑运算。开关的闭合或断开作为条件，灯的亮灭作为事件，开关和灯之间的因果关系为逻辑关系。设开关 A，B 为输入变量，开关闭合用逻辑值 1 表示，开关断开用逻辑值 0 表示；灯 F 为逻辑函数，灯亮用逻辑值 1 表示，灯灭用逻辑值 0 表示。

1. 与运算（逻辑与、逻辑乘）

指示灯的控制开关电路如图 1-7(a)所示。灯 F 亮作为事件发生，开关 A，B 的闭合作为事件发生的条件。从图 1-7(a)可以看出，只有开关 A，B 同时闭合，灯 F 才会亮。故逻辑与定义为：只有当决定一个事件的条件全部具备之后，这个事件才会发生，我们把这种因果关系称为逻辑与。

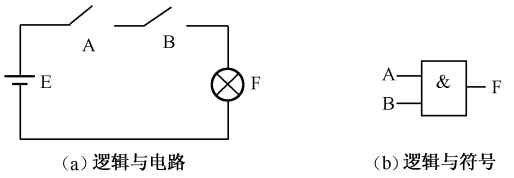
开关电路图 1-7 中，F 和 A，B 之间的逻辑关系也可以用列真值表的方式表示，如表 1-5 所示。由于每一个变量只有两种取值（0 或 1），因此 n 个变量有 2^n 种取值组合。图 1-7(a)有两个变量，则有 $2^2=4$ 种取值组合（00，01，10，11）。A，B 变量的 4 种取值组合与相应函数 F 值列于表 1-5中，这个表称为真值表。由真值表可见，只有当逻辑变量 A，B 全为 1 时，逻辑函数 F 才为 1。这种关系和算术中乘法相似，所以逻辑与又称为逻辑乘，其表达式为

$$F=A\cdot B=AB$$

式中，与运算符号“·”在逻辑函数表达式中也可以略去。与运算规则如下

$$0\cdot 0=0,\quad 0\cdot 1=0,\quad 1\cdot 0=0,\quad 1\cdot 1=1$$

实现与运算的逻辑电路称为与门，与门的逻辑符号如图 1-7(b)所示。



(a) 逻辑与电路　　(b) 逻辑与符号

图 1-7　逻辑与电路及逻辑与符号

表 1-5　逻辑与真值表

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

2. 或运算(逻辑或、逻辑加)

指示灯的控制开关电路如图 1-8(a)所示。只要开关 A 或 B 任一闭合，灯 F 就会亮。故逻辑或定义为：对于决定一个事件(如灯亮)的几个条件(如开关 A 闭合、开关 B 闭合)，只要有一个条件得到满足，这个事件就会发生。

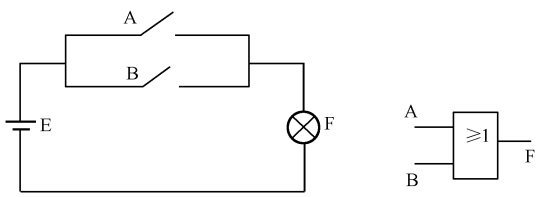
开关电路图 1-8(a)中，F 和 A，B 之间的逻辑关系也可以用真值表来表示，如表 1-6 所示。由表可见，逻辑变量 A，B 之中，只要有一个为 1 时，逻辑函数 F 就为 1。这种关系和算术中加法相似，因此又把逻辑或称为逻辑加，其表达式为

$$F=A+B$$

式中，“+”表示逻辑相加而不是算术相加。这里的 0，1 表示两种不同的逻辑状态，只是逻辑变量取值，没有数量的意思。1+1=1 的含义是 A 和 B 两个开关都闭合，灯 F 亮。或运算规则如下

$$0+0=0,\quad 0+1=1,\quad 1+0=1,\quad 1+1=1$$

实现或运算的逻辑电路称为或门，或门的逻辑符号如图1-8(b)所示。



(a) 逻辑或电路　　(b) 逻辑或符号

图 1-8　逻辑或电路及逻辑或符号

表 1-6　逻辑或真值表

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

3. 非运算(逻辑非)

指示灯的控制开关电路如图 1-9(a)所示。只要开关 A 断开(条件不具备)，灯 F 就会亮，若开关 A 闭合(条件具备)，则灯 F 不亮。由此可见，条件不具备时，事件发生，这种逻辑关系称为逻辑非。

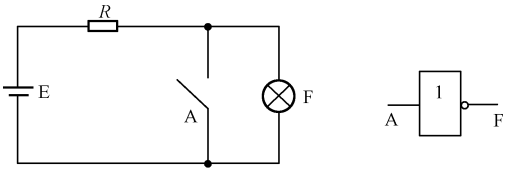
开关电路图 1-9(a)中，F 和 A 之间的逻辑关系也可以用真值表来表示，如表 1-7 所示。由表可见，逻辑变量 A 为 0 时，逻辑函数 F 为 1，其表达式为

$$F=\overline{A}$$

非运算规则为

$$\overline{0}=1,\quad \overline{1}=0$$

实现非运算的逻辑电路称为非门，非门的逻辑符号如图 1-9(b)所示。



(a) 逻辑非电路　　(b) 逻辑非符号

图 1-9　逻辑非电路及逻辑非符号

表 1-7　逻辑非真值表

A	F
0	1
1	0

4. 复合逻辑运算

复合逻辑运算是由与、或、非三种基本逻辑运算组合而成的，经常用到的有与非、或非、与或非、异或、同或等复合逻辑运算，逻辑符号如图 1-10 所示。其中(1)为国标符号，(2)为惯用符号，(3)为国外书刊中常用的符号。

① 与非运算。逻辑表达式为 $F=\overline{AB}$，逻辑符号如图 1-10(a)所示，图上的小圆圈表示非运算。它由与运算和非运算组合而成，先与后非。

② 或非运算。逻辑表达式为 $F=\overline{A+B}$，逻辑符号如图 1-10(b)所示。它由或运算和非运算组合而成，先或后非。

③ 与或非运算。逻辑表达式为 $F=\overline{AB+CD}$，逻辑符号如图 1-10(c)所示。它由与、或、非三种运算组成，先与后或再非。

④ 异或运算。两个输入变量 A，B 逻辑值相同时，输出函数 F 为 0；两个输入变量 A，B 逻辑值相异时，输出函数 F 为 1，这种逻辑关系称为异或。其真值表如表 1-8 所示。由真值表可写出其逻辑表达式为 $F=A\overline{B}+\overline{A}B=A\oplus B$，式中⊕为异或运算符号。逻辑异或符号如图 1-10(d)所示。

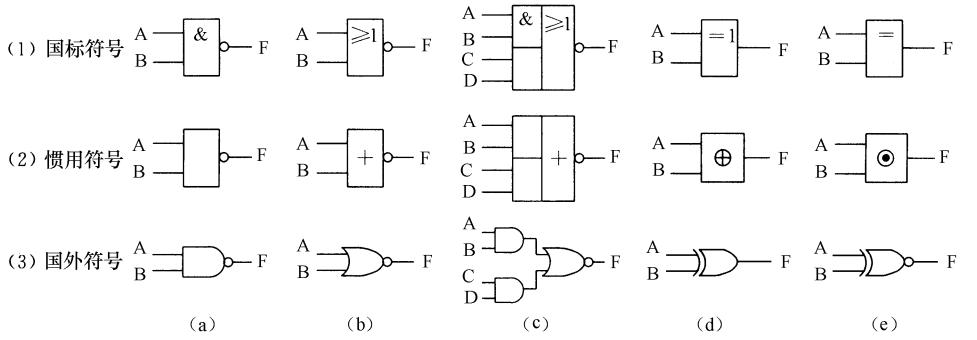


图 1-10　复合逻辑运算符号

⑤ 同或运算。若两个输入变量 A，B 逻辑值相同，则输出函数 F 为 1；若两个输入变量 A，B 逻辑值相异，则输出函数 F 为 0，这种逻辑关系称为同或。其真值表如表 1-9所示。逻辑表达式为 $F=AB+\overline{A}\,\overline{B}=A\odot B$，式中⊙为同或运算符号。逻辑同或符号如图 1-10(e)所示。

表 1-8　逻辑异或真值表

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

表 1-9　逻辑同或真值表

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

1.4.3 逻辑代数的基本公式和常用公式

1. 基本公式

0-1 律	$A \cdot 0 = 0$	$A + 1 = 1$
自等律	$A \cdot 1 = A$	$A + 0 = A$
重叠律	$A \cdot A = A$	$A + A = A$
互补律	$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
交换律	$A \cdot B = B \cdot A$	$A + B = B + A$
结合律	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
分配律	$A \cdot (B + C) = AB + AC$	$A + B \cdot C = (A + B)(A + C)$
吸收律	$A(A + B) = A$	$A + AB = A$
反演律	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$
双重否定律	$\overline{\overline{A}} = A$	

表 1-10 反演律证明

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

上述基本公式可用真值表进行证明。例如证明反演律 $\overline{A \cdot B} = \overline{A} + \overline{B}$，可将变量 A，B 的各种取值组合分别代入等式，其结果如表 1-10所示，等号两边的逻辑值完全对应相等，说明该公式成立。

2. 逻辑代数的三条规则

(1) 代入规则

在任何一个逻辑等式中，如果将等式两边所有出现的变量都代之以一个逻辑函数，此等式仍然成立，这一规则称为代入规则。例如：等式 $\overline{A \cdot B} = \overline{A} + \overline{B}$，若用 $F = AC$ 去代替等式中的 A，则等式仍然成立：$\overline{AC \cdot B} = \overline{AC} + \overline{B} = \overline{A} + \overline{C} + \overline{B}$。这样，两个变量的等式可以变成三个变量的等式。

(2) 反演规则

求逻辑函数 F 的反函数 $\overline{F}$ 时，可用反演规则。它将逻辑函数 F 中的

·　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

则得到的新函数是原逻辑函数的反函数 $\overline{F}$。

在变换过程中应注意：两个以上变量的公用非号保持不变；运算的优先顺序是，先算括号内的运算，然后算逻辑乘，最后算逻辑加。

例如：求 $F = A + \overline{B + \overline{C} + \overline{D + \overline{E}}} + (G \cdot H)$ 的反函数。

$$\overline{F} = \overline{A} \cdot \overline{\overline{B} \cdot C \cdot \overline{\overline{D} \cdot E}} \cdot (\overline{G} + \overline{H})$$

(3) 对偶规则

将逻辑函数 F 中的

· 换成 +，+ 换成 ·

0 换成 1，1 换成 0

便得到逻辑函数 F 的对偶式 F′。若两个逻辑函数相等，则它们的对偶式也相等。若两个逻辑函数的对偶式相等，那么这两个逻辑函数也相等。

例如：求 $F=A\cdot B+\overline{A}\cdot C+B\cdot C$ 的对偶式。

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

3. 常用公式

公式 1 $AB+A\overline{B}=A$

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

公式 2 $A+\overline{A}B=A+B$

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

公式 3 $AB+\overline{A}C+BC=AB+\overline{A}C$

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

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

公式 4 $\overline{AB+\overline{A}C}=A\overline{B}+\overline{A}\,\overline{C}$

证明：$\overline{AB+\overline{A}C}=\overline{AB}\cdot\overline{\overline{A}C}=(\overline{A}+\overline{B})\cdot(A+\overline{C})=\overline{A}A+\overline{A}\,\overline{C}+A\overline{B}+\overline{B}\,\overline{C}$

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

公式 5 $\overline{A\overline{B}+\overline{A}B}=AB+\overline{A}\,\overline{B}\quad(\overline{A\oplus B}=A\odot B)$

证明：$\overline{A\overline{B}+\overline{A}B}=\overline{A\overline{B}}\cdot\overline{\overline{A}B}=(\overline{A}+B)\cdot(A+\overline{B})=AB+\overline{A}\,\overline{B}=A\odot B$

同理 $\overline{A\odot B}=A\oplus B$

公式 6 $xf(x,\overline{x},\cdots,z)=xf(1,0,\cdots,z)$

例如：$A[AB+\overline{A}C+(A+D)(\overline{A}+E)]=A[1\cdot B+0\cdot C+(1+D)(0+E)]$

$=A(B+0+1\cdot E)=A(B+E)$

公式 7 $f(x,\overline{x},\cdots,z)=xf(1,0\cdots,z)+\overline{x}f(0,1,\cdots,z)$

例如：$F=AB+\overline{A}C+(A+D)E+(\overline{A}+H)G$

$=A[1\cdot B+0\cdot C+(1+D)E+(0+H)G]+$

$\overline{A}[0\cdot B+1\cdot C+(0+D)E+(1+H)G]$

$=A(B+E+HG)+\overline{A}(C+DE+G)$

1.4.4 逻辑函数的表示方法

表示逻辑函数的方法有 4 种：真值表、表达式、逻辑图和卡诺图。

设一个逻辑电路有两个输入变量 A 和 B，一个输出函数 F。若 A，B 逻辑取值相同，则输出 F 为 1；若 A，B 逻辑取值不同，则输出 F 为 0。以此例说明逻辑函数的 4 种表示方法。

由逻辑电路的逻辑函数 F 和输入变量 A，B 之间的因果关系可列出真值表，如表 1-11

所示。

根据真值表写出该逻辑电路的逻辑函数表达式为 $F=\overline{A}\cdot\overline{B}+A\cdot B$。

根据逻辑函数表达式可画出逻辑电路图，如图 1-11 所示。

表 1-11　真值表

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

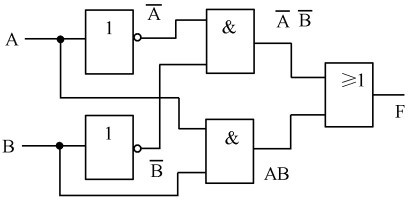


图 1-11　同或逻辑电路图

除上述三种表示方法外，逻辑函数还可用卡诺图表示。这将在逻辑函数的化简中专门讲述。

1.4.5　逻辑函数的化简

一般来说，如果逻辑函数表达式很简单，实现这个表达式的逻辑图就比较简单，所用的元件也就比较少，因而既节约了元件又可以提高可靠性。通常，从逻辑命题抽象出来的逻辑函数不一定是最简的，所以要求对逻辑函数进行化简，找到其最简单的表达式。此外，有时逻辑函数表达式是最简的形式，但是它不一定适合给定的逻辑门，这种实际情况又要求对已有的最简式进行适当的变换，才能用给定的逻辑门画出逻辑电路图。

一个逻辑函数可有多种不同的表达式，这些表达式可以互相转换，例如：

$F=AB+\overline{A}\,\overline{B}$　　与-或表达式

$=\overline{\overline{AB}\cdot\overline{\overline{A}\,\overline{B}}}$　　与非-与非表达式

$=\overline{(\overline{A}+\overline{B})\cdot(A+B)}$　　或-与非表达式

$=\overline{\overline{A}B+A\overline{B}}$　　与-或非表达式

$=\overline{(\overline{A}+\overline{B})}+\overline{(A+B)}$　　或非-或表达式

$=\overline{\overline{A}B}\cdot\overline{A\overline{B}}$　　与非-与表达式

$=(A+\overline{B})\cdot(\overline{A}+B)$　　或-与表达式

$=\overline{\overline{(A+\overline{B})}+\overline{(\overline{A}+B)}}$　　或非-或非表达式

与或表达式是最常用的一种逻辑表达式，最简与或表达式的标准是：式中含的与项最少；各与项中含的变量数最少。有了最简与或表达式，就很容易得到其他形式的最简表达式。这里只介绍两种与或表达式的化简方法。一种是公式化简法，另一种是卡诺图化简法。

1. 逻辑函数的公式化简法

公式化简法，就是利用基本公式和常用公式来化简逻辑函数，下面通过几个具体的例子来说明公式化简法。

(1) 吸收法

利用 $A+AB=A$ 公式，消去多余的乘积项。

例如：$F_1=A\overline{B}+A\overline{B}CD(E+F)=A\overline{B}[1+CD(E+F)]=A\overline{B}$

$$F_2=\overline{A}+\overline{A\cdot\overline{BC}}\cdot(B+\overline{AC+\overline{D}})+BC=\overline{A}+BC+(\overline{A}+BC)(B+\overline{AC+\overline{D}})$$

$$=(\overline{A}+BC)[1+(B+\overline{AC+\overline{D}})]=\overline{A}+BC$$

(2) 消去法

利用 $A+\overline{A}B=A+B$ 公式，消去多余因子。

例如：$F_1=\overline{A}+AB+\overline{B}E=\overline{A}+B+\overline{B}E=\overline{A}+B+E$

$$F_2=A\overline{B}+\overline{A}B+ABCD+\overline{A}\,\overline{B}CD=A\overline{B}+\overline{A}B+(AB+\overline{A}\,\overline{B})CD$$

$$=A\overline{B}+\overline{A}B+\overline{A\overline{B}+\overline{A}B}\,CD=A\overline{B}+\overline{A}B+CD$$

(3) 合并项法

利用 $A+\overline{A}=1$ 公式，两项合并为一项，消去一个变量。

例如：$F_1=ABC+\overline{A}BC+\overline{BC}=BC(A+\overline{A})+\overline{BC}=BC+\overline{BC}=1$

$$F_2=A(BC+\overline{B}\,\overline{C})+A(B\overline{C}+\overline{B}C)=ABC+A\overline{B}\,\overline{C}+AB\overline{C}+A\overline{B}C$$

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

(4) 配项法

为了达到化简目的，有时给某个与项乘以$(A+\overline{A})$，把一项变为两项再与其他项合并进行化简；有时也可以添加 $A\cdot\overline{A}$项进行化简。

例如：$F_1=A\overline{B}+B\overline{C}+\overline{B}C+\overline{A}B$

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

$$=A\overline{B}C+A\overline{B}\,\overline{C}+AB\overline{C}+\overline{A}B\overline{C}+\overline{B}C+\overline{A}B$$

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

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

$$F_2=AB\overline{C}+\overline{ABC}\cdot\overline{AB}=AB\overline{C}+\overline{ABC}\cdot\overline{AB}+AB\cdot\overline{AB}$$

$$=AB(\overline{C}+\overline{AB})+\overline{ABC}\;\overline{AB}=AB\,\overline{(\overline{C}+\overline{AB})}+\overline{AB}\;\overline{ABC}$$

$$=AB\,\overline{ABC}+\overline{AB}\;\overline{ABC}=\overline{ABC}$$

从上面几个例子可以看到，利用公式法化简逻辑函数，要求熟练掌握公式，而且需要一定的技巧，这对初学者来说比较困难。另外，还需说明一点，有的逻辑函数化简的结果不唯一。

2. 逻辑函数的卡诺图化简法

用卡诺图化简逻辑函数是一种简便直观、容易掌握、行之有效的方法。它在数字逻辑电路设计中已得到广泛应用。

一个逻辑函数的卡诺图就是将此函数最小项表达式中各个最小项相应地填入一个特定的方格图内，此方格图称为卡诺图。

(1) 最小项及最小项表达式

① 最小项及最小项性质

最小项是逻辑代数中的一个重要概念，卡诺图中每个小方格都表示了一个最小项。

在有 n 个逻辑变量的逻辑函数中，n 个变量（包含所有变量）的乘积项称为最小项。其特点是：n 个变量有 2^n 个最小项；每个最小项只有 n 个变量；每个变量只能出现一次，不是以原变量形式出现，就是以反变量形式出现。例如：两个变量 A，B，则有 $2^2=4$ 个最小项（$\overline{A}\,\overline{B}$，$\overline{A}B$，$A\overline{B}$，AB）。三个变量 A，B，C 共有 $2^3=8$ 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）。

为了读、写方便，通常将每个最小项编号，用 m_i 表示。编号是这样得到的：最小项中以原变量出现时用 1 表示，以反变量出现时用 0 表示，最小项的变量取值组合为二进制数值，将它转换为对应的十进制数值就是该最小项的编号。例如，最小项$\overline{A}BC$ 的变量取值为 011，所对应的十进制数为 3，所以$\overline{A}BC$ 的编号为 m_3。其余类推，$A\,\overline{B}\,\overline{C}$ 编号为 m_4，$A\,\overline{B}C$ 编号为 m_5 等。

要注意的是，提到最小项时，一定要说明变量的数目，否则最小项这一术语将失去意义。例如，乘积项 ABC 对三个变量是最小项，而对 4 个变量则不是最小项。

下面以三个变量最小项为例说明最小项的性质，最小项的真值表如表 1-12 所示。由此表可以看出最小项的性质。

表 1-12　三个变量最小项的真值表

变　量	最　小　项							
A B C	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$	$\overline{A}B\,\overline{C}$	$\overline{A}BC$	$A\,\overline{B}\,\overline{C}$	$A\,\overline{B}C$	$AB\,\overline{C}$	ABC
0 0 0	1	0	0	0	0	0	0	0
0 0 1	0	1	0	0	0	0	0	0
0 1 0	0	0	1	0	0	0	0	0
0 1 1	0	0	0	1	0	0	0	0
1 0 0	0	0	0	0	1	0	0	0
1 0 1	0	0	0	0	0	1	0	0
1 1 0	0	0	0	0	0	0	1	0
1 1 1	0	0	0	0	0	0	0	1
编　号	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7

i）每个最小项相对应的一组逻辑变量取值使它为 1，而在变量取其他值时，这个最小项都是 0。

ii）所有最小项的逻辑和为 1，即 $\sum(m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7)=1$。

iii）任意两个最小项的逻辑乘为 0，即 $m_i \cdot m_j=0(i\neq j)$。

iv）三个变量的最小项中，每个最小项都有三个相邻项。所谓相邻项，是指如果两个最小项中只有一个变量互为相反变量，其余变量均相同，则称这两个最小项在逻辑上是相邻的，又称逻辑上的相邻性。例如，ABC 和 $AB\,\overline{C}$两个最小项中除变量 C 互为相反变量外，A，B 两个变量均相同，所以 ABC 和 $AB\,\overline{C}$是相邻项。ABC 还有两个相邻项 $A\,\overline{B}C$ 和$\overline{A}BC$。n 个变量的每个最小项都有 n 个相邻项。

v）n 个变量有 2^n 个最小项（$m_0,m_1,\cdots,m_{2^n-1}$）。

② 最小项表达式

任何一个逻辑函数表达式都可以转换成最小项之和的形式。用最小项之和表示的逻辑函数表达式称为最小项表达式。例如，$F(A,B,C)=AB+\overline{A}C$，F 的表达式中每项只含有两个变量，需要把每项缺少的变量补入，使之成为最小项形式，而又不改变原有逻辑关系，则

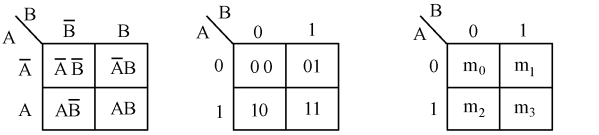
$$F(A,B,C)=AB\cdot(C+\overline{C})+\overline{A}C\cdot(B+\overline{B})=ABC+AB\,\overline{C}+\overline{A}BC+\overline{A}\,\overline{B}C=m_7+m_6+m_3+m_1$$

(2) 卡诺图的画法

卡诺图是根据最小项之间相邻项的关系画出来的方格图。每个小方格代表逻辑函数的一个最小项，下面以两个变量到 5 个变量为例来说明卡诺图的画法。

① 两个变量的卡诺图

两个变量 A，B 共有 4 个最小项$\overline{A}\,\overline{B}$，$\overline{A}B$，$A\,\overline{B}$，AB。用 4 个相邻的方格表示这 4 个最小项之间的相邻关系，如图 1-12 所示。画卡诺图时将变量分为两组，A 为一组，B 为一组。卡诺图的左边线用变量 A 的反变量$\overline{A}$和原变量 A 表示，即上边一行表示$\overline{A}$，下边一行表示 A。卡诺图的上边线用变量 B 的反变量$\overline{B}$和原变量 B 表示，即左边一列表示$\overline{B}$，右边一列表示 B。行和列相与就是最小项，记入行和列相交的小方格内，如图 1-12(a)所示。原变量用 1 表示，反变量用 0 表示，如图 1-12(b)所示。若每个最小项用编号表示，如图 1-12(c)所示。从卡诺图 1-12(a)中看出，每对相邻小方格表示的最小项都是相邻项。

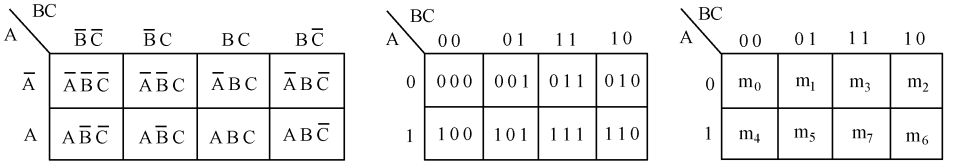


(a) 每格标最小项　(b) 每格标变量取值　(c) 每格标最小项编号

图 1-12　两个变量卡诺图

② 三个变量的卡诺图

三个变量 A，B，C 共有 8 个最小项，则用 8 个小方格分别表示各个最小项，图 1-13(a)是三个变量卡诺图的一种画法。A，B，C 三个变量分为两组，A 为一组，B，C 为一组，分别表示行和列。第 1 行表示$\overline{A}$，第 2 行表示 A，第 1 列表示$\overline{B}\,\overline{C}$，第 2 列表示$\overline{B}C$，第 3 列表示 BC，第 4 列表示 $B\,\overline{C}$。$\overline{A}$，A 标在卡诺图左边线外，$\overline{B}\,\overline{C}$，$\overline{B}C$，BC，$B\,\overline{C}$标在卡诺图上边线外。任意相邻两列都具有相邻性，两个边列也具有相邻性，相邻的两行显然具有相邻性，与上同理可以画出图 1-13(b)和(c)。例如：m_0 的相邻项有 m_1，m_2，m_4。



(a) 每格标最小项　(b) 每格标变量取值　(c) 每格标最小项编号

图 1-13　三个变量卡诺图

三个变量的卡诺图是在两个变量的卡诺图基础上画出来的。以两个变量的卡诺图右边线为对称轴线，作一个对称图形。卡诺图上边线变量 B，C 的标注：变量 C 在对称轴左边和两个变量卡诺图上边线变量标注相同，而轴线右边的变量 C 则与左边对称填写；变量 B 的标注，对称轴左边 B 均填写 0($\overline{B}$)，而右边 B 填写 1(B)。三个变量卡诺图左边线标注变量 $\overline{A}$ 和 A，和两个变量卡诺图标注相同。这样便构成了三个变量的卡诺图，如图 1-13 所示。

③ 4 个变量的卡诺图

4 个变量 A，B，C，D 共有 16 个最小项，则用 16 个小方格分别表示各个最小项，图 1-14 是

4 个变量的卡诺图。A,B,C,D 这 4 个变量分为两组,A,B 为一组,C,D 为一组,分别表示行和列。4 个变量的卡诺图也是在三个变量卡诺图基础上画出来的。以三个变量的卡诺图下边线为对称轴线,作一个对称图形。卡诺图左边线变量 A,B 的标注:变量 B 在对称轴上边和三个变量卡诺图左边线变量标注相同,而轴线下边的变量 B 则与上边对称填写;变量 A 的标注,对称轴上边 A 均填写 0,而下边 A 填写 1。4 个变量卡诺图上边线标注变量 C,D,和三个变量卡诺图标注相同。这样便构成了 4 个变量的卡诺图。

④ 5 个变量的卡诺图

5 个变量 A,B,C,D,E 共有 32 个最小项,则用 32 个小方格分别表示各个最小项,图 1-15 是 5 个变量的卡诺图,5 个变量分为两组,A,B 为一组,C,D,E 为一组,分别表示行和列,5 个变量的卡诺图是在 4 个变量的卡诺图基础上画出来的。以 4 个变量的卡诺图右边线为对称轴线,作一个对称图形,卡诺图上边线变量 C,D,E 的标注与上同理,轴线左面 D,E 和 4 个变量卡诺图 C,D 标注一样,轴线右边则与左边对称填写;变量 C 的标注:轴线左边 C 均填写 0,而右边 C 均填写 1。5 个变量卡诺图左边线标注变量 A,B,和 4 个变量卡诺图标注相同。这样便构成了 5 个变量的卡诺图,如图 1-15 所示。方格中标写的 0,1,2,…,31,是最小项编号的简写。

n 个变量的卡诺图是以 $n-1$ 个变量的卡诺图为基础画出来的。两个变量卡诺图是最基础的卡诺图。

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

图 1-14　4 个变量卡诺图

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

图 1-15　5 个变量卡诺图

(3) 用卡诺图表示逻辑函数

已知逻辑函数表达式,就可画出相应的卡诺图。如果逻辑函数是最小项表达式,则在相同变量的卡诺图中,与每个最小项相对应的小方格内填 1,其余填 0;若逻辑函数是一般式,则先把一般式变为最小项表达式后,再填卡诺图,或直接按逻辑函数一般式填卡诺图亦可。

如果已知逻辑函数真值表,对应于变量逻辑取值的每种组合,函数值为 1 或为 0,则在相同变量卡诺图的对应小方格内填 1 或填 0,就得到逻辑函数的卡诺图。

例如,用卡诺图表示逻辑函数 $F_1=AB+\overline{A}\,\overline{B}C+A\overline{B}\,\overline{C}$,此逻辑函数表达式为一般式。式中第 2,3 项是最小项,编号为 m_1,m_2,式中第一项只有两个变量 A 和 B,缺少变量 C,这一项不是最小项,将 AB 乘以 $(C+\overline{C})=1$,即 $AB(C+\overline{C})=ABC+AB\,\overline{C}$,于是逻辑函数 F 的最小项表达式为

$$F_1=ABC+AB\,\overline{C}+\overline{A}\,\overline{B}C+A\overline{B}\,\overline{C}=m_7+m_6+m_1+m_2=\sum(m_1,m_2,m_6,m_7)$$

将上式的最小项按其编号填入下面卡诺图中相对应的小方格内,标记为 1,其余的小方格填 0,如图 1-16(a)所示。此卡诺图表示了逻辑函数 F_1。也可直接将 AB 项填入卡诺图。先找出变量 A 为 1 的行,即第 2 行用虚线表示出来,再找出 B 为 1 的列,即第 3,4 列,也用虚线表示出来,则行和列虚线相交处的小方格 m_6,m_7($AB\,\overline{C}$,ABC)就是包含 AB 项的两个最小项,如