第1章 数字逻辑基础

数字逻辑电路设计与芯片集成技术是EDA(电子设计)技术的专业基础,它把电路中的正负逻辑电平用二值数字逻辑“0”或“1”来表示,从而以逻辑代数的形式来描述电路工作的行为。也可以利用逻辑代数表达式构建数字逻辑电路。用“1”代表高电平,“0”表示低电平,称为正逻辑表示法;反之,称为负逻辑表示法。本书采用正逻辑表示法。

1.1 电路引入二进制、芯片及集成概念

电路分为模拟电路和数字电路两种。模拟电路中传送的信号为模拟信号,是随着时间连续变化的信号,例如速度、压力、温度、声音等,其采用时间函数表示幅值。模拟电路无法可控地表示电路的数据,由电路本身特性引发,无直观表示性,也容易受到外界环境的干扰。自二进制数和逻辑代数产生以来,人们设计出能够根据稳定的电路输入(条件)得出稳定电路逻辑输出的结果,这就是数字逻辑电路。数字电路中用若干个传送二值“0”和“1”的电路元件组合在一起来表示数据、外界在电路中存储或处理的信息(编码)等,从而把二进制数和逻辑代数与电路对等,形成了开发电路的基本方法。

根据数字电路输入和输出的条件和结果,为其设置具有二值性的变量,利用逻辑代数相关知识,求出输出与输入变量之间的逻辑表达式,并转换为电路。利用制作电路的设备工艺,将输入输出变量引出做成能够拔插、焊接的金属“插脚”或焊点,从电路集成后形成的具有一定形状的块中引出,电路块即数字电路芯片,“插脚”称为引脚,作为电路外部特性可以扩展设计更大规模的电路,如图1-1所示。

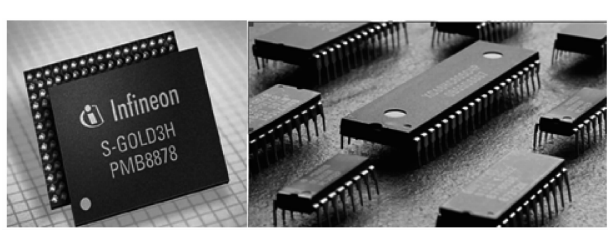

图1-1 具有不同形状和引脚的数字电路芯片

芯片根据需要可以在电子市场购买,根据不同的芯片可以设计出更加强大的电路,并开发为电路板,这称为电路集成。根据集成规模的不同,可分为超大规模、大规模和中小规模集成电路。一个计算机系统是以微处理器芯片为核心,辅以存储器、接口电路、外部设备芯

片，通过其他辅助电路设计构成的完整的系统。图 1-2 就是一个利用芯片开发设计实现的具有一定电路功能的电路板。

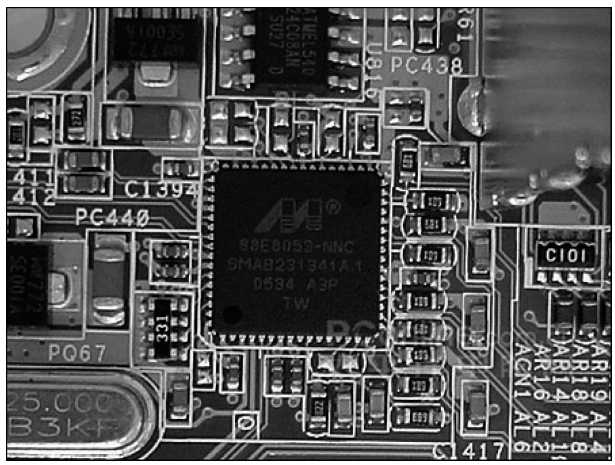

图 1-2　利用芯片设计的电路板

1.2　数制与数制转换

数制是一种表示数的方法，常用的计数制是十进制计数制，而在计算机中，因为数字电路芯片引脚表示数据具有二值性“0”和“1”，数字在计算机内部只能以 0、1 组成的二进制形式存储，但在电路程序设计中，可以用其他进制数来表示。为了说明二进制数，在这里把数制推广到任意的 $R(R\geqslant 2)$ 进制数。

与十进制数相比较，任意 R 进制数可以表示为多项式的形式：

$$(k_n k_{n-1}\cdots k_1 k_0 \cdot k_{-1}k_{-2}\cdots k_{-m+1}k_{-m})_R = \sum_{i=-m}^{n} k_i \times R^i$$
$$= k_n \times R^n + k_{n-1} \times R^{n-1} + \cdots + k_1 \times R^1 + k_0 \times R^0 + k_{-1} \times R^{-1}$$
$$+ k_{-2} \times R^{-2} + \cdots + k_{-m} \times R^{-m}$$

其中，k_i 称为数据的位或系数，其大小范围为 0～$R-1$，i 为该位对应的指数，i 为负数表示该位为小数位；R 为基数，R^i 称为 i 位对应的权，表示本位在基本单位 1 下的数据大小。$k_i \times R^i$ 表示 i 位的实际大小，因此这种表示数据的方法称为位权表示法，和十进制逢 10 进 1 一样，坚持逢 R 进 1 原则。下面看一下计算机中常用的进位计数制及表示。

1.2.1　计算机中常用进位计数制

1. 十进制数

十进制数基数 R 为 10，位 k_i 由 0～9 表示，采用位权表示法来表示数据，并且可以展开为多项式的形式，数据运算时坚持逢 10 进 1 原则。示例如下：

$$(267.109)_{10} = 2\times 10^2 + 6\times 10^1 + 7\times 10^0 + 1\times 10^{-1} + 0\times 10^{-2} + 9\times 10^{-3}$$

在表示数据时，也可以直接写成 267.109，或者在末尾加上字母 D，表示为 267.109D。

2. 二进制数

从前面的知识我们知道，计算机中的数据都是以二进制数的形式存储的。二进制数基数为 2，因此它的位只能由 0、1 组成。相邻位的进位规则为“逢 2 进 1”。通常在末尾加字母 B 或下标 2 来表示数据并区别于其他数制。例如，11011.101B 和$(11011.101)_2$都表示二进制数 11011.101，把该数展开为多项式的表示形式：

$$11011.101B = 1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$

3. 十六进制数

在表示二进制数据时，由于位数较多，在输入、输出、书写和阅读时均不方便，因此引进十六进制数。十六进制数基数为 16，由于 10～15 无法单独表示，它的位由 0～9、A(10)、B(11)、C(12)、D(13)、E(14)、F(15)组成。相邻位的进位规则为“逢 16 进 1”。通常在末尾加字母 H 或下标 16 来表示数据并区别于其他数值。例如，9AB.6FH 和$(9AB.6F)_{16}$都表示十六进制数 9AB.6F，把该数展开为多项式的表示形式：

$$9AB.6FH = 9\times16^2+10\times16^1+11\times16^0+6\times16^{-1}+15\times16^{-2}$$

4. 八进制数

八进制数基数为 8，它的位由数字 0～7 组成。相邻位的进位规则为“逢 8 进 1”。通常在末尾加字母 O 或下标 8 来表示数据并区别于其他数值。例如，275.361O 和$(275.361)_8$都表示八进制数 275.361，把该数展开为多项式的表示形式：

$$275.361O = 2\times8^2+7\times8^1+5\times8^0+3\times8^{-1}+6\times8^{-2}+1\times8^{-3}$$

1.2.2　数制转换

计算机中处理和存储的数据为二进制数，但是在输入、输出、书写时也使用十、八、十六进制数，这就需要掌握二进制数与其他进制数之间的数据转换问题。

1. 二进制数与十进制数之间的相互转换

1）二进制数转换为十进制数

根据二进制数展开的多项式，直接按照十进制数的进位原则计算，得出的结果即十进制数。任意的 R 进制数转换为十进制数都可采用这种方法。把 11011.101B 转换为十进制数：

$$\begin{aligned}11011.101B &= 1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}\\&= 1\times16+1\times8+0\times4+1\times2+1\times1+1\times0.5+0\times0.25+1\times0.125\\&= 16+8+0+2+1+0.5+0+0.125 = 27.625\end{aligned}$$

2）十进制数转换为二进制数

十进制数是可以转换为任意 R 进制数的，先把十进制数转化为 R 进制数多项式表示形式，可以发现整数部分和小数部分具有不同的转换规律。

(1) 整数部分。

十进制数整数 N 可以表示为 R 进制的多项式：

$$(N)_{10}=(k_nk_{n-1}\cdots k_1k_0)_R=k_n\times R^n+k_{n-1}\times R^{n-1}+\cdots+k_1\times R^1+k_0\times R^0$$

式中，k_n、k_{n-1}、…、k_1、k_0 是转换后 R 进制数各位的数字。将等式两边分别除以 R，得到下式：

$$\frac{1}{R}(N)_{10}=k_n\times R^{n-1}+k_{n-1}\times R^{n-2}+\cdots+k_1\times R^0+\frac{k_0}{R}$$

其中，余数为 k_0，为转换的 R 进制数的最低位，商为 $k_n\times R^{n-1}+k_{n-1}\times R^{n-2}+\cdots+k_1\times R^0$，把商再除以 R，可以得到余数 k_1 和新的商，每次给新商除以一次 R，都可以得到一位新的余数 k_i，直到得到最高位为 k_n，商为 0 为止。就可以得到由所有余数组成的 R 进制数。根据这个规则，把十进制数 19 转换为二进制数：

$$\begin{array}{r|l}
2 & 19 \quad \cdots\cdots \text{余}1 \cdots\cdots k_0 \\
2 & 9 \quad \cdots\cdots \text{余}1 \cdots\cdots k_1 \\
2 & 4 \quad \cdots\cdots \text{余}0 \cdots\cdots k_2 \\
2 & 2 \quad \cdots\cdots \text{余}0 \cdots\cdots k_3 \\
2 & 1 \quad \cdots\cdots \text{余}1 \cdots\cdots k_4 \\
\multicolumn{2}{r}{0}
\end{array}$$

根据计算得出 $19=(10011)_2$。

当十进制数较大时，可将其转换为权之和，有权的位为 1，否则为 0。

(2) 小数部分。

十进制数小数 M 可以表示为 R 进制的多项式：

$$(M)_{10}=(k_{-1}k_{-2}\cdots k_{-m+1}k_{-m})_R=k_{-1}\times R^{-1}+k_{-2}\times R^{-2}+\cdots+k_{-m+1}\times R^{-m+1}+k_{-m}\times R^{-m}$$

将上式两边乘以 R，得到下式：

$$R(M)_{10}=k_{-1}+k_{-2}\times R^{-1}+\cdots+k_{-m+1}\times R^{-m+2}+k_{-m}\times R^{-m+1}$$

乘以 R 后，第 1 位小数 k_{-1} 变为整数，即求出第一位小数，k_{-2} 变为第一位小数，把剩下的小数再乘以 R，求出 k_{-2}，依次求出需要求出的所有小数位为止。根据这个规则，把十进制数 0.375 转换为二进制数：

$$\begin{array}{r}
0.375 \\
\times \quad 2 \\
\hline
0.75 \cdots\cdots\cdots\cdots\cdots\cdots \text{取整得 } 0 \cdots\cdots\cdots\cdots k_{-1} \\
0.75 \\
\times \quad 2 \\
\hline
1.5 \cdots\cdots\cdots\cdots\cdots\cdots \text{取整得 } 1 \cdots\cdots\cdots\cdots k_{-2} \\
0.5 \\
\times \quad 2 \\
\hline
1.0 \cdots\cdots\cdots\cdots\cdots\cdots \text{取整得 } 1 \cdots\cdots\cdots\cdots k_{-3} \\
0.0
\end{array}$$

根据计算得出 $0.375=(0.011)_2$。

如果小数部分位数是无穷的，要根据要求获取适当的位数。

2. 二进制数与八、十六进制数之间的转换

因为八进制数每位用3位二进制数表示，十六进制数用4位二进制数表示，因此八进制数转化为二进制数只要把每一位数表示为3位二进制数即可，十六进制数转换为二进制数则把每一位表示为4位二进制数。反之，则要以小数点为中心，向两边每取3位转换为八进制数，每取4位转换为十六进制数，不够位需要补位。

1）八进制数转换为二进制数

$$756.23O=111\ 101\ 110.010\ 011B$$

2）二进制数转换为八进制数

$$101\ 110\ 100.011\ 10B=101\ 110\ 100.011\ 100B=564.34O$$

3）十六进制数转换为二进制数

$$DF16.5H=1101\ 1111\ 0001\ 0110.0101B$$

4）二进制数转换为十六进制数

$$101\ 1011\ 1100\ 1001.1010\ 100B=0101\ 1011\ 1100\ 1001.1010\ 1000B=5BC9.A8H$$

1.2.3　二进制算术运算

在计算机中，运算的数据是以多个电路输出的0和1排列成二进制数据来表示的，计算机运算是通过二进制电路来计算。数据在计算机中的表示有无符号二进制数和有符号二进制数两种。算术运算也分为无符号二进制运算和有符号二进制运算两种。二进制加法运算规则为“逢2进1”。

1. 无符号二进制数的算术运算

1）无符号二进制数的加法

加法规则为：**0+0=0，0+1=1，1+1=10**

求10110和10100的和：

$$\begin{array}{r} 10110 \\ +10100 \\ \hline 101010 \end{array}$$

2）无符号二进制数的减法

减法规则为：**1−0=1，1−1=0，0−0=0，10−1=1**，其中被减数要大于减数。

求1010和0101的差：

$$\begin{array}{r} 1010 \\ -0101 \\ \hline 0101 \end{array}$$

3）无符号二进制数的乘法运算

计算1011和1010的积：

```
      1011
    ×1010
  ------
      0000
     1011
    0000
   1011
  --------
  1101110
```

由上述运算可以看出，乘法运算是由左移被乘数和加法运算组成的。

4）无符号二进制数的除法运算

计算 1010 和 111 的商：

```
          1.011
      ------
111 ) 1010
        111
      -----
        1100
         111
       -----
         1010
          111
        -----
          011
```

由上述运算过程可见，除法运算由右移被除数和减法运算组成。

2. 有符号二进制数在计算机中的表示和算术运算

数据在计算机中一般是以有符号数来存储的，现代计算机一般是补码计算机，符号位同样参加到运算中。在计算机中，一般把最高位作为符号位，“0”表示为正号，“1”表示为负号。学习补码，需要了解真值、原码、反码的概念。

真值是指书写中带符号的二进制数，例如－1110.11B，＋10111.101B。

1）原码表示法

用二进制数“0”和“1”分别表示真值 X 中的“＋”和“－”，就得到数据真值对应的原码。计算机中一般用 8 位二进制数作为表示一个数据的最小单位，称为字节(Byte)。两个字节数据称为一个字(Word)，两个字称为双字(DWord)。

如果机器数为 8 位定点整数，X＝＋1011110B，则$[X]_{原}$＝01011110B；如果 X＝－1011110B，则$[X]_{原}$＝11011110B。

如果机器数为 8 位定点小数，X＝＋0.1011110B，则$[X]_{原}$＝0.1011110B；如果 X＝－0.1011110B，则$[X]_{原}$＝1.1011110B。

0 是一个特殊实例。

$$[+0000000]_{原}=00000000B$$

$$[-0000000]_{原}=10000000B$$

可以看出，0 的原码表示有两种，即有重码。

2）反码表示法

（1）正数的反码。

正数的反码与原码相同。即对于真值 X，$[X]_{反}=[X]_{原}$。

（2）负数的反码。

负数的反码是原码符号位不变，数据位取反。如果 X＝－1011110B，$[X]_{原}=11011110B$，则$[X]_{反}=10100001B$。

如果 X＝－0.1011110B，$[X]_{原}=1.1011110B$，则$[X]_{反}=1.0100001B$。

3）补码表示法

如果基数为 R，位数为 n 的带“＋”或“－”的真值 N，其补码为：

$$(N)_{补}=R^n+N$$

例如，如果原码位数为 8 位，求 X＝－37 的补码。

解：37＝00100101B，则－37 ＝－00100101B，$[X]_{补}=100000000-00100101=11011011B$。

根据规律，得出补码直接转换过程：

（1）正数的补码。

正数的补码与原码相同。即对于真值 X，$[X]_{补}=[X]_{原}$。

（2）负数的补码。

负数的补码则在其反码的最后一位加 1。

例如，$[-37]_{原}=10100101B$，$[-37]_{反}=11011010B$，在其末尾加 1，得到$[X]_{补}=11011011B$。

4）n 位二进制整数不同码表示的数值范围

原码：$-(2^{n-1}-1)\sim+(2^{n-1}-1)$

反码：$-(2^{n-1}-1)\sim+(2^{n-1}-1)$

补码：$-2^{n-1}\sim+(2^{n-1}-1)$

5）二进制补码的加减运算

采用补码形式，符号位参与运算，可以和无符号数一样进行加减乘除的运算，最终都可以采用加法实现运算过程。假设 X、Y 为两个二进制数的真值，则可得到以下表达式：

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}$$

$$[X-Y]_{补}=[X+(-Y)]_{补}=[X]_{补}+[-Y]_{补}$$

例如，设 X＝56，Y＝31，求$[X-Y]_{补}$。

解：56＝＋0111000B，31＝＋0011111B，$[X]_{补}=[56]_{补}=00111000B$，$[-Y]_{补}=[-31]_{补}=11100001B$，$[X-Y]_{补}=[X]_{补}+[-Y]_{补}=00111000+11100001=1\ 00011001B$，数据只有 8 位，最高位 1 自动丢弃。

最终结果：$[X-Y]_{补}=[X]_{补}+[-Y]_{补}=00111000+11100001=00011001B$。

6）补码运算的溢出和判别

根据数据运算规律，两个符号相反的数相加产生结果的位数是不会超过两个加数对应位数的，即不产生溢出。但两个同符号数相加，则符号位可能产生进位。如果运算时最高两位产生的进位不相同，说明符号位不统一，则产生了溢出。

例如，试计算－76 和－56 表示的 8 位补码数据相加后是否溢出。

(1) 直接相加判别。

8 位负数补码最小数据为－128，因此，小于－128 会产生溢出，－76＋(－56)＝－132＜－128，所以产生溢出。

(2) 根据公式计算后判别。

$[-76]_{补}=10110100B$，$[-56]_{补}=11001000B$，$[-76]_{补}+[-56]_{补}=10110100+11001000=101111100B$，最高两位进位为 10，不相同，所以产生溢出。

1.3 计算机中常用编码

计算机与外界进行数据输入输出，主要是通过键盘、外部设备获得信息，这些信息并不能直接通过计算机识别，必须在计算机中给予每个信息一个二进制组合，用来作为在计算机中识别它们的标志，而同类信息具有相同的二进制组合规则。这种以一定的编制规则表示数值、字母、符号等外部信息的二进制组合称为二进制编码；而将编码在计算机内再翻译为原信息的过程称为译码。若同类信息为 N 个，则对它们编码的二进制位数 n 满足以下条件：

$$2^n \geqslant N$$

1.3.1 二-十进制编码

二-十进制编码就是用 4 位二进制数来表示一位十进制数中的 0～9 这 10 个数码，简称 BCD(Binary Coded Decimal)码。4 位二进制数共有 16 种组合，可以从中选择 10 个编码来表示十进制数的 10 个数码。表 1-1 为几种常用的 BCD 码。

表 1-1 几种常用的 BCD 码

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

用 4 位自然二进制码中的前 10 个码字来表示十进制数码，因各位的权值依次为 8、4、2、1，故称为 8421 BCD 码。可以用它在计算机中表示十进制的数据。

2421 码的权值依次为 2、4、2、1；5421 码的权值依次为 5、4、2、1；余 3 码由 8421 BCD 码加 0011 得到；余 3 循环码的特点是任何相邻的两个码字中仅有一位二进制位不同，其他位相同。余 3 码和余 3 循环码是无权码，代表一定含义，每一位并不表示本位的权值。

1.3.2　格雷码

格雷码是电路设计中卡诺图化简采用的编码规则，是一种无权码。它具有相邻性，即两个相邻代码之间仅有一位取值不同。相邻编码不同数字的位数称为码距，码距越小，避免错误数码出现的概率越低。格雷码编码规则如下：

设某二进制数为 $B_{n-1}B_{n-2}\cdots B_2B_1B_0$，其对应的格雷码为 $G_{n-1}G_{n-2}\cdots G_2G_1G_0$，其中，最高位保留即 $G_{n-1}=B_{n-1}$，其他各位利用异或运算（两个二进制位进行异或运算，数据不相同，结果输出为 1，否则，输出为 0，异或运算符为⊕）求得：$G_i=B_{i+1}\oplus B_i$（$i=0,1,2,\cdots,n-2$）。示例如下：

$B_3B_2B_1B_0=0110$，$G_3=B_3=0$，$G_2=B_3\oplus B_2=0\oplus 1=1$，$G_1=B_2\oplus B_1=1\oplus 1=0$，$G_0=B_1\oplus B_0=1\oplus 0=1$。所以 0110 的格雷码为 0101。

根据以上规则，求得 3 位二进制数对应的格雷码如表 1-2 所示。

表 1-2　3 位二进制数对应的格雷码

二进制码			格雷码			二进制码			格雷码		
B_2	B_1	B_0	G_2	G_1	G_0	B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0	1	0	0	1	1	0
0	0	1	0	0	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	0	1
0	1	1	0	1	0	1	1	1	1	0	0

通过表 1-2 可以发现，相邻格雷码只有一位二进制数是不同的。

1.3.3　ASCII 码

人们通过键盘上的字母、符号和数值向计算机发送数据和指令，所有键盘符号用 7 位二进制数来编码，表示 128 个十进制数、英文大小写字母、控制符、运算符及特殊符号，称为美国标准信息交换码（American Standard Code for Information Interchange），简称 ASCII 码，如表 1-3 所示，其中一些字符的含义如表 1-4 所示。

在用汇编语言、高级语言编程时，这些 ASCII 编码可以被访问。编码 30H～39H 代表数字字符“0”～“9”；编码 41H～5AH 代表大写字母“A”～“Z”；编码 61H～7AH 代表小写字母“a”～“z”；0AH 代表换行符；ODH 代表回车符等。对于表 1-4 所示的字符，在使用时可以查看表 1-3 对应的 ASCII 码。

表 1-3　ASCII 码表

$b_3b_2b_1b_0$	$b_6b_5b_4$							
	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

续表

$b_3 b_2 b_1 b_0$	$b_6 b_5 b_4$							
	000	001	010	011	100	101	110	111
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

表 1-4 ASCII 码中各字符的含义

字符	含 义	字符	含 义
NUL	空符号	DC1	设备控制 1
SOH	标题开始	DC2	设备控制 2
STX	文本开始	DC3	设备控制 3
ETX	文本结束	DC4	设备控制 4（停止）
EOT	传输结束	NAK	回绝应答
ENQ	询问	SYN	同步空闲
ACK	应答	ETB	传输块结束
BEL	铃声(声音或报警信号)	CAN	取消
BS	退格	EM	中介结束
HT	水平表格	SUB	替换
LF	换行	ESC	脱离、取消
VT	垂直表格	FS	文件分隔符
FF	表格输入	GS	组分隔符
CR	回车	RS	记录分隔符
SO	移出	US	单元分隔符
SI	移入	SP	空格
DLE	数据连接变更	DEL	删除

1.4 逻辑运算与逻辑代数

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。

逻辑代数又称布尔代数，是由英国数学家乔治·布尔把数学的形式化方法应用到逻辑学领域而建立起来的一门“应用数学”，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。在逻辑代数中，只有 0 和 1 两种逻辑值，有与、或、非 3 种基本逻辑运算，还有与或、与非、与或非、异或几种导出逻辑运算。

逻辑代数中的变量称为逻辑变量，可以以字母开头，由字母和数字组成，例如 A、B、C、x、y、A_1 等。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。变量通过逻辑关系构成的表达式称为逻辑函数，例如 Y＝F(A,B,C,…)，F 为输出函数，A、B、C、…为输入逻辑变量。

1.4.1　3 种基本逻辑运算

逻辑关系相当于算术运算中的运算关系，逻辑运算符相当于算术运算中的算术运算符。逻辑代数的基本逻辑运算有逻辑与(乘)、逻辑或(加)、逻辑非，其他逻辑运算由这 3 种运算复合而成。

1. 与运算

与逻辑的定义：仅当决定事件(F)发生的所有条件(A,B,C,…)均满足时，事件(F)才能发生。表达式为：

$$F(A,B,C,\cdots) = A \cdot B \cdot C \cdot \cdots$$

式中，小圆点表示书写的与运算符，与运算符可省略，形式如下：

$$F(A,B,C,\cdots) = ABC\cdots$$

1) 与运算规则

$$0 \cdot 0=0;\quad 0 \cdot 1=1 \cdot 0=0;\quad 1 \cdot 1=1$$

2) 与运算的逻辑符号

任何逻辑运算最终要用一定的逻辑符号(或逻辑符号组成的逻辑电路)表示，逻辑关系才能得到分析和应用，逻辑符号也是开发电路原理图的组成部分。本书采用 Verilog HDL 语言开发设计环境中国际 IEEE 标准通用电路符号，不再采用国家标准的电路符号。对于与运算表达式 F＝A·B，其电路符号如图 1-3 所示。AND2 代表与门的符号名称，即二引脚输入的与门，inst 是符号的名称序号标志，是可以修改的。

2. 或运算

或逻辑的定义：当决定事件(F)发生的各种条件(A,B,C,…)中，只要有一个或多个条件具备，事件(F)就发生。表达式为：

$$F(A,B,C,\cdots) = A+B+C+\cdots$$

式中，加号表示书写用的或运算符。

1) 或运算规则

$$0+0=0;\quad 0+1=1+0=1;\quad 1+1=1$$

2）或运算的逻辑符号

对于或运算表达式 F＝A＋B，其电路符号如图 1-4 所示，在绘制原理图软件中的符号名称为 OR2，即二引脚输入的或门，inst 为自定义符号名字。

3. 非运算

非逻辑指的是逻辑的否定。当决定事件(F)发生的条件(A)满足时，事件不发生；条件不满足时，事件反而发生。表达式为：

$$F(A) = \overline{A}$$

在绘制电路原理图时，其符号如图 1-5 所示。NOT 为非门符号名称。

图 1-3 二输入变量的与门国际标准符号

图 1-4 二输入变量的或门国际标准符号

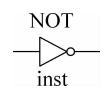

图 1-5 非门的国际标准符号

非运算的运算规则为：

$$\overline{0} = 1 \qquad \overline{1} = 0$$

4. 几种常见的逻辑运算

利用与、或、非 3 种基本运算可以构造出具有实际用途的复合运算。常用的有与非、或非、异或和同或运算。下面以二输入变量 A、B，输出变量 F 来说明这几种逻辑运算。

1）与非运算

与非运算是由与运算和非运算组合在一起的一种运算，其逻辑表达式为：

$$F(A,B) = \overline{AB}$$

其逻辑计算结果可以通过真值表来体现，真值表是一种能同时体现输入变量各种取值和输出变量计算结果情况的表格。与非运算的真值表如表 1-5 所示。

与非门的国际标准符号如图 1-6 所示。NAND2 是与非门符号名称。

表 1-5 与非运算的真值表

A	B	F(A,B)
0	0	1
0	1	1
1	0	1
1	1	0

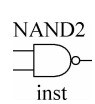

图 1-6 与非门的国际标准符号

2）或非运算

或非运算是由或运算和非运算组合在一起的一种运算。其逻辑表达式为：

$$F(A,B) = \overline{A+B}$$

或非运算的真值表如表 1-6 所示。

或非门的国际标准符号如图 1-7 所示。NOR2 是或非门符号名称。

表 1-6　或非运算的真值表

A	B	F(A,B)
0	0	1
0	1	0
1	0	0
1	1	0

图 1-7　或非门的国际标准符号

3）异或运算

当两个输入逻辑变量 A、B 不同时，输出 F 为 1；当两个输入逻辑变量 A、B 相同时，输出 F 为 0，这种逻辑运算称为异或运算。其逻辑表达式为：

$$F(A,B) = \overline{A}B + A\overline{B} = A \oplus B$$

⊕是异或运算符，异或运算的真值表如表 1-7 所示。

异或门的国际标准逻辑符号如图 1-8 所示。

表 1-7　异或运算的真值表

A	B	F(A,B)
0	0	0
0	1	1
1	0	1
1	1	0

图 1-8　异或门国际标准逻辑符号

XOR 是异或门符号名称。

4）同或运算

当两个输入逻辑变量 A、B 相同时，输出 F 为 1；当两个输入逻辑变量 A、B 不同时，输出 F 为 0，这种逻辑运算称为同或运算。其逻辑表达式为：

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

⊙是同或运算符，同或运算的真值表如表 1-8 所示。

同或门的国际标准逻辑符号如图 1-9 所示。

表 1-8　同或运算的真值表

A	B	F(A,B)
0	0	1
0	1	0
1	0	0
1	1	1

图 1-9　同或门国际标准逻辑符号

XNOR 是同或门符号名称。

1.4.2　逻辑函数及其表示方法

1. 逻辑函数

由 1.4.1 小节可以看到，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 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。

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：

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

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。

2. 逻辑函数的表示方法

一个逻辑函数是一个输入与输出之间的关系表达式，根据情况的不同，需要用不同的描述法来表示它。逻辑函数的描述方法有真值表、卡诺图、逻辑表达式、逻辑图、波形图等。卡诺图表示法在 1.5 节介绍。逻辑表达式、真值表描述法在前面已有所讲述，在这里只介绍逻辑图表示法和波形图表示法。

1）逻辑图表示法

在已知逻辑函数的基础上，用与、或、非、异或等逻辑运算符号表示逻辑函数中各变量之间的逻辑关系的图形称为逻辑图。已知逻辑表达式 $F(A,B)=AB+\overline{A}\,\overline{B}$，用与、或、非运算符号表示出的逻辑图如图 1-10 所示。

首先画出 A 的非符号，输出为 F_1，画出 B 的非符号，输出为 F_2，作为第一级；然后画出 AB 的与门符号，设输出为 F_3，同时画出 $\overline{A}\,\overline{B}$ 的与门符号，设输出为 F_4，F_3、F_4 作为第二级；最后画出 F_3+F_4 或门符号，作为第三级。

2）波形图的表示方法

用输入端在不同逻辑信号作用下所对应的输出信号的波形图表示电路的逻辑关系。图 1-11 是图 1-10 所示逻辑图的波形图。

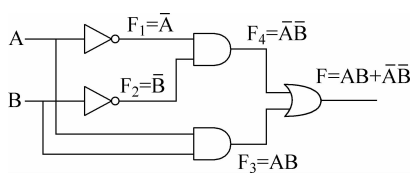

图 1-10　函数的逻辑图表示法

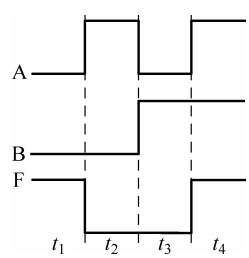

图 1-11　波形图描述法

图中，4 个 t 时刻 F 的输出值分别对应 A、B 输入的 4 种组合值。

1.4.3　逻辑代数

逻辑代数是 1854 年问世的，早年用于开关和继电器的分析、化简，随着半导体制造工艺和集成技术的发展，各种规模的集成电路不断出现并进入市场，而逻辑代数正是通过对芯片外特性和系统性能进行分析成为实现现代数字逻辑电路和计算机系统设计不可缺少的数学

工具。

逻辑代数具有一定的公式、定律和规则，用它们对逻辑表达式进行处理，可以完成对逻辑电路的化简、分析、设计，表达电路意图等。

1. 逻辑代数的基本公式、定律

利用与、或、非3种基本逻辑运算和恒等式代入规则可以证明下面公式或定律两边是成立的。

(1) **0-1律**，即数值0、1和变量的与、或运算：$A+0=A$；$A+1=1$；$A\cdot 0=0$；$A\cdot 1=A$。

(2) **重叠律**，即变量自身的与、或运算：$A+A=A$；$A\cdot A=A$。

(3) **互补律**，即变量与反变量的与、或运算：$A+\overline{A}=1$；$A\cdot\overline{A}=0$。

(4) **自补律**，$\overline{\overline{A}}=A$。

(5) **结合律**，即3个变量两两组合的与、或运算：$(A+B)+C=A+(B+C)$；$ABC=A(BC)$。

(6) **交换律**，即两个变量之间与、或运算位置互换：$A+B=B+A$；$AB=BA$。

(7) **分配律**，即3个变量与或混合运算重新分配：$A(B+C)=AB+AC$；$A+BC=(A+B)(A+C)$。

(8) **反演律**，即摩根定理：$\overline{A+B+C+\cdots}=\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\cdots$；$\overline{ABC\cdots}=\overline{A}+\overline{B}+\overline{C}+\cdots$。

(9) **吸收律**，可由以上定律推导：$A+A\cdot B=A$；$A\cdot(A+B)=A$；$A+\overline{A}B=A+B$。

(10) **冗余律**：$AB+\overline{A}C+BC=AB+\overline{A}C$；$(A+B)(\overline{A}+C)(B+C)=(A+B)(\overline{A}+C)$。

以上定律的证明可以采用真值表形式，把输入变量的取值组合一一列出，然后求出左右两边表达式的值，如果都相等，则定律成立，例如摩根定律。

一些定律可以利用其他定律来证明，试证明：

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

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

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

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

$=AB+\overline{A}C$

这个定律在以后化简逻辑函数中通过添加或吸收多余项来实现。

2. 逻辑代数的基本规则

(1) 代入规则：任何一个含有变量A的等式，如果将所有出现A的位置都用同一个逻辑函数代替，则等式仍然成立。这个规则称为代入规则。

例如，已知等式$\overline{AB}=\overline{A}+\overline{B}$，用函数$Y=AC$代替等式中的A，根据代入规则，等式仍然成立，即有：

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

代入规则可以扩展所有基本定律的应用范围。即把某变量变换为多个其他变量的表示并代入定律中，定律仍然成立。

(2) 反演规则：对于任何一个逻辑表达式Y，如果将表达式中的所有“·”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，那么所得

到的表达式就是函数 Y 的反函数 $\overline{Y}$(或称补函数)。这个规则称为反演规则。例如,对于下列表达式:

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

采用反演规则,Y 反变量为:

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

这里提供了一种求输出反变量的好方法。

(3) 对偶规则:对于任何一个逻辑式 Y,若将其中所有的"+"换成"·","·"换成"+",1 换成 0,0 换成 1,则得到的表达式称为 Y 的对偶式,记做 Y′。

如果 Y=A+AB,则其对偶式为 Y′=A·(A+B)。

若两逻辑式相等,则它们的对偶式也相等,这就是对偶规则。对于下面恒等式:

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

其对偶式为:

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

3. 逻辑代数的公式化简法

本书主要讲解数字逻辑电路设计、分析及逻辑电路应用。数字逻辑电路在设计过程中需要得出逻辑表达式,并转换为逻辑图或逻辑电路原理图。电路对应的逻辑表达式不一定是最简表达式,因此我们仍需要对逻辑表达式进行化简,利用化简后的逻辑表达式构成的逻辑电路使用的器件成本最低,提高了数字系统的可靠性。

最简表达式如与、或、非、与非、或非、与或非、或与非、异或等。

运用基本公式和常用公式来化简逻辑函数的方法主要有以下几种:

1) 并项法

利用公式 $A+\overline{A}=1$,将两项合并为一项,并消去一个变量,再利用分配率,化简 Y 表示的表达式:

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

2) 吸收法

(1) 利用公式 A+AB=A,消去多余的项,化简 Y 表示的表达式:

$$Y = \overline{A}B + \overline{A}BCD(E + F) = \overline{A}B$$

利用摩根定律和吸收法,化简 Y 表示的表达式:

$$\begin{aligned} Y &= A + \overline{\overline{B} + \overline{CD}} + \overline{\overline{AD}\,\overline{B}} = A + BCD + AD + B \\ &= (A + AD) + (B + BCD) = A + B \end{aligned}$$

(2) 利用公式 $A+\overline{A}B=A+B$,消去多余的变量,化简 Y 表示的表达式:

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

3）配项法

（1）利用公式 $A=A(B+\overline{B})$，为某一项配上其所缺的变量，以便用其他方法进行化简，化简 Y 表示的表达式：

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

（2）利用公式 $A+A=A$，为某项配上其所能合并的项，化简 Y 表示的表达式：

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

4）消去冗余项法

利用冗余律 $AB+\overline{A}C+BC=AB+\overline{A}C$，将冗余项 BC 消去，化简 Y 表示的表达式：

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

1.5　逻辑代数的卡诺图化简法

利用公式化简可以使逻辑函数变为较为简单的形式，但得到的逻辑表达式是否为最简很难判断，使用卡诺图化简法可以很好地解决这个问题。

1.5.1　最小项的定义及其性质

1. 最小项定义

如果一个函数的某个与运算项包含了函数的全部变量，其中每个变量都以原变量或反变量的形式出现，且仅出现一次，则这个乘积项称为该函数的一个标准积项，通常称为最小项。

3 个变量 A、B、C 可组成 8 个最小项。

（1）$\overline{A}\,\overline{B}\,\overline{C}$：把 ABC＝000 代入最小项，输出为 1，表示此项为逻辑函数有效项；反之，如果已知某最小项取值为 ABC＝000，其为逻辑函数有效项，则可在逻辑函数中用 $\overline{A}\,\overline{B}\,\overline{C}$ 表示。

（2）$\overline{A}\,\overline{B}C$：表示当 ABC＝001 时，此项为逻辑函数有效项，输出为 1。

（3）$\overline{A}\,B\,\overline{C}$：表示当 ABC＝010 时，此项为逻辑函数有效项，输出为 1。

（4）$\overline{A}BC$：表示当 ABC＝011 时，此项为逻辑函数有效项，输出为 1。

（5）$A\,\overline{B}\,\overline{C}$：表示当 ABC＝100 时，此项为逻辑函数有效项，输出为 1。

（6）$A\,\overline{B}C$：表示当 ABC＝101 时，此项为逻辑函数有效项，输出为 1。

(7) $AB\overline{C}$：表示当 ABC=110 时，此项为逻辑函数有效项，输出为 1。

(8) ABC：表示当 ABC=111 时，此项为逻辑函数有效项，输出为 1。

一般 n 个变量的最小项应有 2^n 个。

2. 最小项的表示方法

通常用符号 m_i 来表示最小项。下标 i 的确定方法：把最小项中的原变量记为 1，反变量记为 0，当变量顺序确定后，可以按顺序排列成一个二进制数，则与这个二进制数相对应的十进制数就是这个最小项的下标 i。

3 个变量 A、B、C 的 8 个最小项可以分别表示为：

$$m_0=\overline{A}\,\overline{B}\,\overline{C},\quad m_1=\overline{A}\,\overline{B}C,\quad m_2=\overline{A}B\overline{C},\quad m_3=\overline{A}BC$$

$$m_4=A\,\overline{B}\,\overline{C},\quad m_5=A\overline{B}C,\quad m_6=AB\overline{C},\quad m_7=ABC$$

3. 最小项的性质

为了分析最小项的性质，下面列出 3 个变量所有最小项的真值表，如表 1-9 所示。

表 1-9　3 个变量最小项真值表

A	B	C	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
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

观察该表，可得出最小项的下列性质：

(1) 对于任意一个最小项，只有一组变量取值使其值为 1。

(2) 对于输入变量的任意一组取值，任意两个不同的最小项的乘积必为 0。

(3) 对于输入变量的任意一组取值，全部最小项的和必为 1。

1.5.2　逻辑函数的最小项表达式

利用逻辑函数的基本公式与定律，可以把任意逻辑函数化成若干个最小项之和的形式，称为最小项表达式。

对于不是最小项表达式的与或表达式，可利用公式 $A+\overline{A}=1$ 和 $A(B+C)=AB+AC$ 来配项展开成最小项表达式。把 Y 表达式转换为最小项和的形式：

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

$$= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + ABC$$
$$= m_0 + m_1 + m_2 + m_3 + m_7$$
$$= \sum m(0,1,2,3,7)$$

$\sum m(0,1,2,3,7)$ 是 Y 对应的最小项之和的简化表示形式。

如果列出了函数的真值表，则只要将函数值为 1 的那些最小项相加，便是函数的最小项表达式，如表 1-10 所示。

根据真值表，列出 Y 对应的最小项表达式：

$$Y = m_1 + m_2 + m_3 + m_5 = \sum m(1,2,3,5)$$
$$= \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C$$

表 1-10　输出变量 *Y* 对应的表达式

A	B	C	Y	最小项
0	0	0	0	m_0
0	0	1	1	m_1
0	1	0	1	m_2
0	1	1	1	m_3
1	0	0	0	m_4
1	0	1	1	m_5
1	1	0	0	m_6
1	1	1	0	m_7

将真值表中函数值为 0 的那些最小项相加，便可得到反函数的最小项表达式。

1.5.3　用卡诺图表示逻辑函数

1. 卡诺图的构成

将逻辑函数或真值表中的最小项重新排列成矩阵形式，并且在矩阵的横方向和纵方向排列所有逻辑输入变量的取值，取值按照格雷码的顺序排列，横方向和纵方向交叉的单元格内填入最小项在函数中的输出值，这样构成的图形就是卡诺图。如图 1-12 所示的是三变量卡诺图。

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

图 1-12　三变量卡诺图

三变量卡诺图输入变量为 ABC，A 的取值在纵向按格雷码排列，保证相邻的项只有一位不同，BC 的取值在横向按格雷码排列，当 A＝0，B＝00，即 ABC＝000 时，在交叉处的方格应该填入 m_0 对应的逻辑输出值，真值表为 1 则填入 1，逻辑表达式存在的最小项填入 1，其他填入 0。

2. 卡诺图的特点

卡诺图的特点是任意两个相邻的最小项在图中是相邻的。

相邻项是指两个最小项只有一个因子互为反变量，其余因子均相同，又称为逻辑相邻项。可利用分配律和 $A+\overline{A}=1$ 消掉互为反变量的因子，就是两个相邻项消掉一个因子，这也是卡诺图化简的基本依据。

下面以四变量卡诺图说明相邻项的组合，如图 1-13 所示。

m_0 项的输入因子 AB 取值为 00，按照格雷码原则，它与 AB=01、AB=10 取值对应的 m_4、m_8 为相邻项；因子 CD 取值为 00，按照格雷码原则，它与 CD=01、CD=10 取值对应的 m_1、m_2 为相邻项，共有 4 个相邻项。可以按照图 1-13 标出相邻项。靠在一起的用椭圆圈在一起，例如 m_0 和 m_1，没有靠在一起的相邻项用相同的标志标出，例如 m_0 和 m_8，m_0 和 m_2。其他 m_i 项对应的相邻项用同样的方法找出。

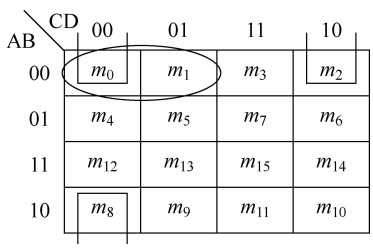

图 1-13 四变量卡诺图

3. 逻辑函数在卡诺图中的表示

将逻辑函数表达式化成最小项表达式，将表达式中出现的最小项按照编号在对应的卡诺图方格中填入“1”，其余填入“0”，就得到了逻辑函数的卡诺图形式。将 Y 表示的函数用卡诺图表示出来，如图 1-14 所示。

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

图 1-14 函数 Y 表示的卡诺图

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

如果要求函数 Y 的反函数 $\overline{Y}$，则对 Y 中所包含的各个最小项，在卡诺图相应方格内填入 0，其余方格内填入 1。也可以找出 $\overline{Y}$ 的最小项，剩下的就是 Y 对应的最小项。将 Y 表示的函数用卡诺图表示出来，如图 1-15 所示。

AB \ CD	00	01	11	10
00	0	1	1	1
01	1	1	1	0
11	1	0	0	1
10	1	1	1	0

图 1-15 利用反演律得到的卡诺图