第 1 章 数字逻辑运算

学习重点 掌握数制转换及常用编码的方法；掌握逻辑代数的基本定律、恒等式和规则、卡诺图化简方法。

学习难点 不同形式的逻辑函数表达式之间的转换；带有无关项的卡诺图化简法熟练运用。

能力要求 能够熟练进行数制、码制及各数制之间的转换；熟悉常用逻辑代数的基本定律、恒等式和规则，熟练运用代数化简法和卡诺图化简法化简逻辑函数表达式。

思政育人 通过思政故事，分析编码在战争、加密、传输等不同场景下的应用，引出信息加密性的重要性、安全可靠性要求；名人成长经历、布尔代数、香农定理等逻辑思维方法解读；逻辑函数表达式化简结果的不唯一特点引申出“条条大道通罗马”“思考方式方法角度不同，结果可能不同”。

1.1 模拟信号和数字信号

在信号分析中，按照时间和幅值的连续性和离散性把信号分为 4 类：

(1) 时间连续、数值连续信号；

(2) 时间离散、数值连续信号；

(3) 时间离散、数值离散信号；

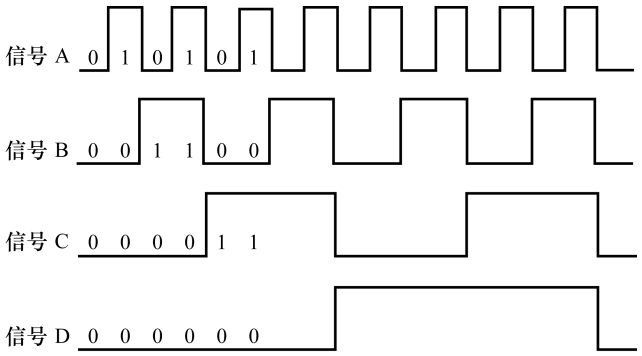
(4) 时间连续、数值离散信号。

按照信号与系统的定义，信号是传递有关一些现象的行为或属性的信息的函数，而这个函数通常自变量是时间或者是位置。随着时间连续变化的信号，定义为模拟信号(Analog signal)。模拟信号的特点是在时间上和幅值上均是连续的，在一定动态范围内可能取任意值。从宏观上看，我们周围的世界大多数物理量都是时间连续、数值连续的变量。而不是随着时间连续变化，也就是间隔一段时间变化的信号，其实就是离散时间信

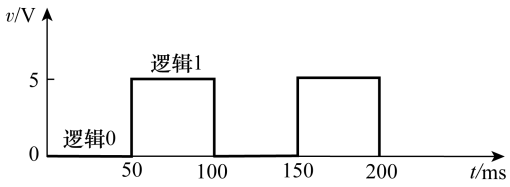
号，但如果离散时间信号只有有限个取值的就是数字信号，所以我们谈数字信号，实际上谈的是信号本身的一种属性或者数学上的特征。

从另外一方面来讲，信号是必须有载体的。例如声音信号可以传达语言、音乐或其他信息，图像信号可以传达人类视觉系统能够接受的图像信息。现实世界的物理现象如果以时间为自变量，那么绝大部分都可以表示成某种物理量的变化过程。而这种变化过程中，时间是连续变化的，物理量也是连续变化的。这显然是一种模拟信号，例如温度的变化、飞机速度的变化等。

数字系统(如微处理器系统)中运行的信号都是数字信号(Dagital signal)。从时间函数波形看，它们只存在高、低两种电平的相互转换，这两种电平分别代表了二元编码中的1和0。图1-1是一组数字信号的实例，用信号的角度来看，信号A、B、C、D的周期不同，有1和0两个逻辑状态。



(a)不同数字信号区别



(b)用逻辑电平描述的数字波形

图1-1　数字信号

既然数字信号的自变量和应变量都是离散的，那我们其实最直观想到的就是用一个有有限个幅度值的脉冲信号来表征它，比如1就是1 V，2就是2 V，3就是3 V。事实上在数模转换中，我们其实是把常规的数字信号转换成了这种类型的脉冲。但我们在真实的系统中并没有用这种方法来表征数字信号，而是采用了的一套二进制的数值系统来表征这个数字信号。这个二进制的数值系统是采用多个二值信号+权重的方式来表示数值的。

所谓的“数字信号处理”，实际上是对已经被某种数值系统表示出来的数字信号加以计算，本质上是在用“数值计算”的方法来处理。常规的“数值计算”自然是二进制的加减运算了。

数字电子技术已经广泛应用于电视、雷达、通信、电子计算机、自动控制、电子测量仪表、航空航天等领域，例如在测量仪表中，数字测量仪表比模拟仪表测量精度更高、测试功能更强，而且易于实现测试的自动化和智能化；在通信系统中，应用数字电子技术的数字通信系统，不仅比模拟通信系统抗干扰能力更强、保密性好，而且还能应用计算机进行信息处理和控制，形成以计算机和通信设备组成的自动交换通信网络。随着集成电路技术的发展，尤其是大规模和超大规模集成器件的发展，使得各种电子协同可靠性大大提高，设备的体积大大缩小，各种功能尤其是自动化、智能化和数字化呈现程度显著提高。

1.2　常用数制

1.2.1　进位进制数

计数是用数码表示事物数量的多少。数制也称为“计数制”，是用一组固定的符号和统一的规则来表示数值的方法。任何一个数制都包含两个基本要素：基数和位权。常用数制有十进制数 D(decimal)、二进制数 B(binary)、八进制数 O(octal)、十六进制数 H(hexadecimal)等。

基数是指数制所使用数码的个数，例如二进制的基数为 2；十进制的基数为 10。位权是指数制中某一位上的 1 所表示数值的大小，例如十进制的 123，1 的位权是 100，2 的位权是 10，3 的位权是 1。二进制中的 1011(一般从左向右开始)，第一个 1 的位权是 8(即 2^3)，0 的位权是 4(即 2^2)，第二个 1 的位权是 2(即 2^1)，第三个 1 的位权是 1(即 2^0)。

由于人类解剖学的特点，双手共有十根手指，故在人类自发采用的进位制中，十进制是使用最为普遍的一种。成语“屈指可数”某种意义上来说描述了一个简单计数的场景，而原始人类在需要计数的时候，首先想到的就是利用天然的算筹(手指)来进行计数。十进制的计数规则是逢十进一，例如

$$(1234.56)_{10}=1\times10^3+2\times10^2+3\times10^1+4\times10^0+5\times10^{-1}+6\times10^{-2}$$

计算机领域我们主要采用二进制进行计数，是因为二进制具有以下优点。

(1) 二进制数中只有两个数码 0 和 1，可用具有两个不同稳定状态的元器件来表示一位数码，例如电路中某一通路的电流的有无、某一节点电压的高低、晶体管的导通和截止等。

(2) 二进制数运算简单，大大简化了计算中运算部件的结构。计数规则是逢二进一，借一当二，例如

$$(1001.0101)_2=1\times2^3+0\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}$$

(3) 二进制天然兼容逻辑运算。

但是二进制计数在日常使用上有个不便之处，就是位数往往很长，读写不便，例如把

十进制的(100000)D 写成二进制就是(11000011010100000)B，所以计算机领域我们实际采用的是十六进制。

十六进制是人们在计算机指令代码和数据的书写中经常使用的数制。在十六进制中，数用 0～9 和 A(10)、B(11)、C(12)、D(13)、E(14)、F(15)(或 a，b，…，f)16 个符号来描述。计数规则是逢十六进一。例如十进制的 100000 写成十六进制就是$(186A0)_{H}$。

由于二进制数据的基数 R 较小，因此二进制数据的书写和阅读不方便，为此在小型机中引入了八进制。八进制中，数用 0～7 来表示，计数规则是逢八进一。八进制用下标 8 或数据后面加 O 表示，例如

$$(1234.56)_8=(1234.56)_O=1\times8^3+2\times8^2+3\times8^1+4\times8^0+5\times8^1+6\times8^{-2}$$

1.2.2 不同进制数的转化

1. 十进制转换为二进制数

十进制转换为二进制数可分为整数部分和小数部分的转换。灵活运用转换方法，如 47.5=32+15+1/2，可以直接转换为二进制数为 101111.1。这时可以使用 $2^{10}=1024$，$2^9=512$，$2^8=256$，$2^7=128$，$2^6=64$，$2^5=32$，$2^4=16$ 去接近。

例 1-1　将十进制数 47 化为二进制数。可按照除 2 求余数，转换进制数注意是余数的反序排列，从而得到$(47)_{10}=(101111)_2$。

2	47	……	余 1
2	23	……	余 1
2	11	……	余 1
2	5	……	余 1
2	2	……	余 0
2	1	……	余 1
	0		

	0.8125		取整	基数
×	2			
	1.6250	……	1	k_{-1}
	0.6250			
×	2			
	1.2500	……	1	k_{-2}
	0.2500			
×	2			
	0.5000	……	0	k_{-3}
	0.5000			
×	2			
	1.0000	……	1	k_{-4}

例如将$(0.8125)_{10}$化为二进制小数时可乘 2 求整数进行，转换进制数注意是余数的正序排列。所以$(0.8125)_{10}=(0.1101)_2$。

例 1-2　将十进制数 1000 转化为二进制数。

方法 1：可以按照除 2 求余数 0001 0111 11，得到 1000=11 1110 1000。

方法 2：可以采用我们即将在后续课程中使用的称重时采用的逐次逼近法。

$$1000=512+256+128+64+32+8=11\ 1110\ 1000$$

方法 3：可以采用求差法

$$1000=1023-23=11\ 1111\ 1111-1\ 0111=11\ 1110\ 1000$$

2. 二、八、十六进制转换为十进制数

按照“按权对位展开相加”的方法即可，如表 1-1 所示。

表 1-1　常用不同进制数对照表

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

3. 十进制到八进制、十六进制转换方法与十进制-二进制转换方法一样，可按小数点为界，分整数和小数部分分别进行转换，再求和。

4. 二进制与八进制数转换

由于 $8=2^3$，因此一位八进制数可用 3 位二进制数表示，从 000，001，010，011，100，101，110，111 共计 8 种状态。转换时，由小数点开始，整数部分自右向左，小数部分自左向右，3 位一组，不够三位的添零补齐。

如 $(10110.011)_2=(\underline{0}10\ 110.011)_2=(26.3)_8$

$(365.12)_8=(011\ 110\ 101.001\ 010)_2=(11110101.00101)_2$

5. 二进制与十六进制数转换

由于 $16=2^4$，一位十六进制数可用 4 位二进制数表示，从 0000，0001，0010，0011，…，1111 共计 16 种状态。转换时，由小数点开始，整数部分自右向左，小数部分自左向右，4 位一组，不够三位的添零补齐。

如 $(110111.011101)_2=(\underline{00}11\ 0111.0111\ 0\underline{100})_2=(37.74)_{16}$

$(1A.C8)_{16}=(0001\ 1010.1100\ 1000)_2=(11010.11001)_2$

1.2.3 二进制数的运算

二进制数的加法和乘法基本运算法则各有四条，即

$$0+0=0,\ 0+1=1,\ 1+0=1,\ 1+1=10$$

$$0\times0=0,\ 0\times1=0,\ 1\times0=0,\ 1\times1=1$$

例 1-3　计算两个二进制数 1010 和 0101 的积，和我们后续课程“信号与系统”中的卷积尺计算方法相似。

```
   1 0 1 0
 + 0 1 0 1
 -------
   1 1 1 1
```

（a）两个二进制数1010和0101的和

```
   1 0 1 0
 - 0 1 0 1
 -------
   0 1 0 1
```

（b）两个二进制数1010和0101的差

```
       1 0 1 0
     × 0 1 0 1
     -------
       1 0 1 0
     0 0 0 0
   1 0 1 0
 0 0 0 0
 -----------
   1 1 0 0 1 0
```

（c）两个二进制数1010和0101的乘积

```
          1. 0 1 1
       ------------
111 ) 1 0 1 0
        1 1 1
      -------
          1 1 0 0
            1 1 1
          -------
            1 0 1 0
              1 1 1
            -------
              1 1 ……余数
```

（d）两个二进制数1010和0111之商

图 1-2　二进制数的运算

计算机系统中存储一个数使用的是有符号的二进制数，最高位为符号位，其余位为数值位。其中反码形式：正数的反码为原码本身，负数反码符号位不变，剩余的数字位取反。补码形式：正数的补码为原码本身，负数的补码为反码+1。

如整数部分 X_n，X_{n-1}，X_{n-2}，…，X_1，X_0，其中最高位为 0 时表示正数，数值位与真值或原码相同；最高位为 1 时表示负数，数值位在反码最低位加 1。

如小数部分 X_{-1}，X_{-2}，$X_{-3,\cdots}$，X_{-n+1}，X_{-n}，其中最高位为 0 时表示正数，数值位与真值或原码相同；最高位为 1 时表示负数，数值位在反码最低位加 1。

$$[X]_{补}=\begin{cases}X & 0\leqslant X\leqslant 2^n\\ 2^{n+1}+X & -2^n\leqslant X\leqslant 0\end{cases}$$

$$X=+0.1011\rightarrow[X]_{补}=0.1011$$

$$X=-0.1011\rightarrow[X]_{补}=1.0101$$

4 位二进制数的原码、反码、补码形式如表 1-2 所示。

表 1-2　4 位二进制数的原码、反码、补码形式

十进制数	原码	反码	补码	十进制数	原码	反码	补码
-8	-	-	1000	0	0 0 0 0	0 0 0 0	0 0 0 0
-7	1 1 1 1	1 0 0 0	1 0 0 1	+ 1	0 0 0 1	0 0 0 1	0 0 0 1
-6	1 1 1 0	1 0 0 1	1 0 1 0	+ 2	0 0 1 0	0 0 1 0	0 0 1 0
-5	1 1 0 1	1 0 1 0	1 0 1 1	+ 3	0 0 1 1	0 0 1 1	0 0 1 1
-4	1 1 0 0	1 0 1 1	1 1 0 0	+ 4	0 1 0 0	0 1 0 0	0 1 0 0
-3	1 0 1 1	1 1 0 0	1 1 0 1	+ 5	0 1 0 1	0 1 0 1	0 1 0 1
-2	1 0 1 0	1 1 0 1	1 1 1 0	+ 6	0 1 1 0	0 1 1 0	0 1 1 0
-1	1 0 0 1	1 1 1 0	1 1 1 1	+ 7	0 1 1 1	0 1 1 1	0 1 1 1

n 位二进制数的原码、反码表示范围都为$-(2^{n-1}-1)\sim+(2^{n-1}-1)$，补码表示范围为$-2^{n-1}\sim+(2^{n-1}-1)$。

由于计算机只有加法没有减法，因此 1-1 被设计成 1 +(-1)，使用补码使得两个相反数的补码相加为 0，同时符号位也参与运算之中。减法运算的原理：减去一个正数相当于加上一个负数 A-B=A+(-B)，对(-B)求补码，然后进行加法运算。

例 1-4　用补码形式求 5+7。

解　(5+7)补=(5)补+(7)补=0 101 + 0 111=0 1100

此时我们对数字位进行了位扩展，由原来的三位变为四位。

例 1-5　用二进制补码计算 5-2。

解　因为(5-2)补=(5)补+(-2)补=0101+1110=0011，所以 5-2=3。

例 1-6　数 A 的补码为 11111001，数 B 的补码为 11010101，求-A-B，并讨论其是否溢出。

首先求-A 的补码可分为两步：

(1)A 的数据位取反；

(2)A 的数值位取反加 1。

所以-A 的补码为 00000111；同理-B 的补码为 00101011，所以求补码运算

$$-A-B=00000111+00101011=00110010$$

溢出判定有两个原则：

(1) 如果两个符号相异的数相加，则不会溢出；

(2) 如果两个同符号的数相加，其最高有效位的进位输出和进位输入不等，则溢出。

题中为两个同符号的数相加，但是最高有效位的进位输出和进位输入都为 0，故无溢出发生。

1.3 常用码制

数码不仅可以表示数量的大小，还可以表示不同的信号和符号，这时的数码没有大小，只是一种代号，称为代码，如学生学号 119100101、身份证号 371428198204295517、教室门牌号 YFJ201 等，一般用十进制和字母、特殊符号表示。在数字电路系统中，用二进制数形式书写的代码表示各种符号、文字等信息的过程，称为编码，编制代码需要遵循收发双方约定的编码规则称为码制。

1.3.1 BCD 码

BCD 码(Binary Code Decimal)，常见的 BCD 码又称 8421 码，是十进制代码中最常用的一种，由于从左到右每一位的 1 分别表示 8、4、2、1，因此把这种代码称为 8421BCD 码。

BCD 码都是用四位二进制数表示十进制中的 0～9 十个数，每一位的 1 代表的十进制数称为这一位的权，且每一位的权是固定不变的。

从 4 位二进制数 16 种代码中，选择 10 种来表示 0～9 个数码的方案有很多种。每种方案产生一种 BCD 码，如表 1-3 所示。

表 1-3 几种常见的十进制代码

十进制数	8421 码	5211 码	2421 码	余 3 码	余 3 循环码	格雷码
0	0000	0000	0000	0011	0010	0000
1	0001	0001	0001	0100	0110	0001
2	0010	0100	0010	0101	0111	0011
3	0011	0101	0011	0110	0101	0010
4	0100	0111	0100	0111	0100	0110
5	0101	1000	1011	1000	1100	0111
6	0110	1001	1100	1001	1101	0101
7	0111	1100	1101	1010	1111	0100
8	1000	1101	1110	1011	1110	1100
9	1001	1111	1111	1100	1010	1000
特点	十进制有权、恒权码			无权码	变权码	变权码

用 4 位二进制数字可以表示 0～15，但是 8421BCD 码只有 0～9 十个数字，所以在 8421BCD 编码规则中 1010～1111 这 6 个数是无效的。

8421BCD 编码在时钟、温度计、计数器中应用比较广泛。

例 1-7　$(0001\ 0010)_{8421BCD}=(\qquad\qquad\qquad\qquad)_2$

解　$(0001\ 0010)_{8421BCD}=(12)_{10}=(1100)_2$

例 1-8　$(463.5)_{10}=(0100\ 0110\ 0011.0101)_{8421BCD码}$

注意：每位数字上的 0 不可省略。

2421 码是一种恒权代码，它的 0 和 9，1 和 8，2 和 7，3 和 6，4 和 5 也互为反码，这个特点和余 3 码相仿。

5211 码是另一种恒权代码。如果按 8421 码接成十进制计数器，则连续输入计数脉冲时，4 个触发器输出脉冲对于计数脉冲的分频比从低位到高位依次为 5∶2∶1∶1。可见，5211 码每一位的权正好与 8421 码十进制计数器 4 个触发器输出脉冲的分频比相对应。这种对应关系在构成某些数字系统时很有用。

在采用的无权码的一些方案中，早期用的比较多的是余 3 码(Excess-3 code)，余 3 码的编码规则与 8421 码不同，如果把每一个余 3 码看作 4 位二进制数，则它的数值要比它所表示的十进制数码多 3，故而将这种代码称为余 3 码。它的主要优点是执行十进制数位相加时，能正确地产生进位信号，而且还给减法运算带来了方便。

余 3 循环码是一种变权码，每一位的 1 在不同代码中并不代表固定的数值。它的特点是相邻的两个代码之间仅有一位的状态不同。

自然界存在很多干扰，二进制数据在被处理、传输、存储的过程中受干扰易发生错误。数据是被处理、传输和存储的直接对象，当直接对象出现错误时若不能检测，高层检测将会付出更大的代价，因此我们引入可靠性编码，常用的可靠性编码有格雷码、奇偶校验码等。

1.3.2　格雷码

在一组数的编码中，若任意两个相邻的代码只有一位二进制数不同，则称这种编码为格雷码(Gray Code)，另外由于最大数与最小数之间也仅一位数不同，即“首尾相连”，因此又称循环码或反射码。在数字系统中，常要求代码按一定顺序变化。若采用 8421 码，则数 0111 变到 1000 时四位均要变化，而在实际电路中，4 位的变化不可能绝对同时发生，则计数中可能出现短暂的其他代码。

如表 1-4 所示，格雷码属于可靠性编码，是一种错误率最小的编码方式，另外由于这种编码相邻的两个码组之间只有一位不同，因而在用于方向的转角位移量-数字量的转换中，当方向的转角位移量发生微小变化，而可能引起数字量发生变化时，格雷码仅改变一位，这样与其他编码同时改变两位或多位的情况相比更为可靠，即可减少出错的可能性。

表 1-4　二进制码与格雷码的转换

二进制码	1	0	1	1
格雷码	1	1	1	0
变换规则	最高位相同	0+1	1+0	1+1(舍去进位位 1)

格雷码是一种变权码，每一位码都不固定，因此不可以直接进行算术运算，需要经过一次变换成自然二进制码。

二进制到格雷码的转换规则：

(1) 格雷码的最高位(最左边)与二进制码的最高位相同；

(2) 从左到右，逐一将二进制码相邻的两位相加(舍去进位)，作为格雷码的下一位。

设二进制码为 $B=B_{n-1}B_{n-2}\cdots B_{i+1}B_i\cdots B_1B_0$，对应格雷码为 $G=G_{n-1}G_{n-2}\cdots G_{i+1}G_i\cdots G_1G_0$，则 $G_{n-1}=B_{n-1}$，$G_i=B_{i+1}\oplus B_i(0\leqslant i\leqslant n-2)$。其中⊕为异或符号，相同为 0，不同为 1。

1.3.3 检错码

检错码是一种编码，指在传输过程中发生错误后，在接收端能自动检查并发现错误的编码。目前常用的检错码有奇偶校验码、恒比码等。为提高信息传输的有效性和可靠性，根据香农信息理论，必须对信源消息实施信源编码和信道编码。

检错码的两大类别：奇偶校验编码和循环冗余编码。

奇偶校验码是在原信息码元后面附加一个监督元，使码组中 1 或 0 的个数为奇数或偶数，为奇数者称为奇数校验码；为偶数者称为偶数校验码。如将十进制 5 的 8421BCD 码 0101 增加校验位后，奇校验码是 10101，偶校验码是 00101，其中最高位分别为奇校验位 1 和偶校验位 0。

循环校验码(CRC 码)是数据通信领域中最常用的一种差错校验码，其特征是信息字段和校验字段的长度可以任意选定。

1.3.4 ASCII 码

ASCII 码是美国国家标准化协会指定的一种信息代码，它的研究与制定起始于 20 世纪 50 年代后期，在 1967 年定案，后被国际标准化组织 ISO 认定为国际标准，称为 ISO646 标准，广泛应用于计算机和通信领域中，最典型的就是键盘信号的输入。

ASCII 码使用指定的 7 位或 8 位二进制数组合来表示 128 或 256 种可能的字符。0～31 号及 127 号(共 33 个)是控制字符或通信专用字符，32～126 号(共 95 个)是字符(32 是空格)，其中 48～57 号为 0～9 十个阿拉伯数字，65～90 号为 26 个大写英文字母，97～122 号为 26 个小写英文字母，其余为一些标点符号、运算符号等。

同时还要注意，在标准 ASCII 中，其最高位(b7)用作奇偶校验位。所谓奇偶校验，是指在代码传送过程中用来检验是否出现错误的一种方法，一般分奇校验和偶校验两种。后

128 个称为扩展 ASCII 码。许多基于 X86 的系统都支持使用扩展或“高”ASCII。

表 1-5　ASCII 码表

ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符
0	NUT	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	”	66	B	98	b
3	ETX	35	#	67	C	99	
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	1
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	0	111	o
16	DLE	48	0	80	P	112	p
17	DCI	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	X	115	s
20	DC4	52	5	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	PS	60	<	92	/	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	—	127	DEL

具体代码含义可查阅网址 https：//www. runoob. com/tags/html-ascii. html。

1.3.5 GB 2312 简体中文编码表

GB 2312—1980(GB0)标准共收录 6 763 个汉字(一级、二级)及 682 个全角字符拉丁字母、希腊字母、日文平假名及片假名字母、俄语字母等；GB2312 依然保留 26 个英文字母和一些特殊符号的编码，覆盖 ASCII 中的相同部分，同时将 ASCII 中前 32 个控制字符则继续沿用，即十六进制 20H。因此区位码要加上 20H 才能得到国标码。在国标码的基础上，将字节的最高位设为 1，因为 ASCII 中使用 7 位，最高位为 0。

点阵字体(位图字体)把每个字分成 $M \times N$ 个点(16×16、24×24)，每个字形都以一组二维像素信息表示。样就区分开了 ASCII 和 GB 2312。存储 1 024 个 24 * 24 点阵的汉字字型码需要的字节数是 72B。位图字体难以缩放，强行放大会导致失真字形，产生马赛克式的锯齿边缘。

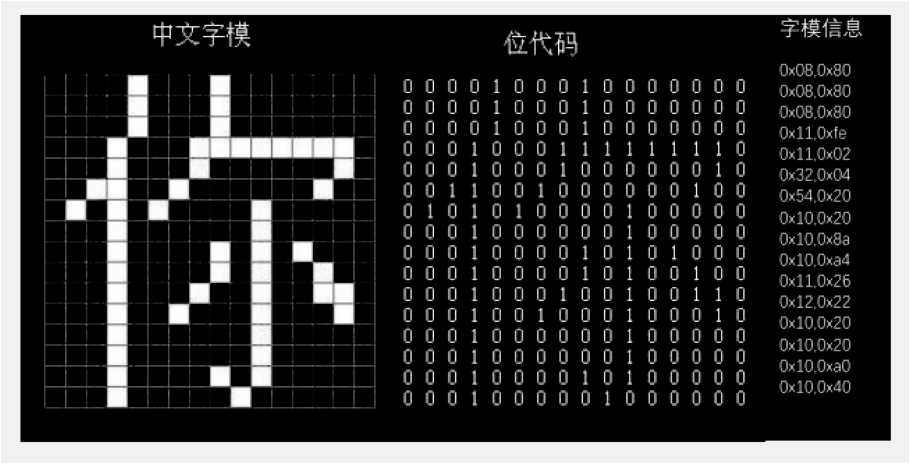


图 1-3 “你”的点阵字体

1.4 逻辑代数

中国人早就注意到二元现象的普遍性，可以追溯到先秦时期的二元辨证思辨，如高低、胖瘦、美丑、正反等，并发展出富有东方色彩的哲学体系。但遗憾的是，中国人未用数学的观点去刻划这种二元现象，将其视为阴阳玄学，而非精确的科学。

逻辑代数(Logic algebra)是英国数学家乔治·布尔于 1849 年首先提出的，所以又称为布尔代数。后来，布尔代数被广泛地应用于解决开关电路的分析与设计上，所以也把布尔代数叫作开关代数。

逻辑代数是按照一定的逻辑规律进行运算的代数，其中的变量称为逻辑变量，用字母 A，B，C，…表示，但它和普通代数中变量的含义却有着本质的区别。逻辑变量只有两个

值，即 0(逻辑 0)和 1(逻辑 1)，而没有中间值。0 和 1 并不表示数量的大小，只代表两种不同的逻辑状态。

乔治·布尔发明的工具，叫作“集合论”(Set theory)，如图 1-4 所示。他认为，逻辑思维的基础是一个个集合(Set)，每一个命题表达的都是集合之间的关系。

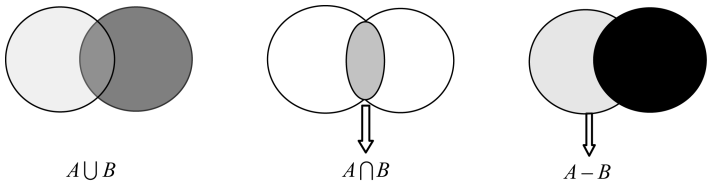


图 1-4　集合论

集合论可以帮助我们得到直觉无法得到的结论，保证推理过程正确，比文字推导更可靠。20 世纪初，英国科学家香农指出：布尔代数可以用来描述电路，或者说电路可以模拟布尔代数。香农 1938 年发表的论文《继电器与开关电路的逻辑分析》中，将纯数学布尔代数用于电路设计。布尔代数是信息论的基础。信息论讲的是通信的理论，而布尔代数解决的是计算的问题。只有将所有的电信号处理成离散的 0 和 1，才能进行数字化通信。

1.5　基本逻辑运算

数字电路是用来表达输入、输出之间的逻辑关系的电路，在逻辑代数中，有与、或、非三种基本逻辑运算。

所谓逻辑，就是事件的发生条件与结果之间所要遵循的规律。一般来说，事件的发生条件与产生的结果均为有限个状态，每一个和结果有关的条件都有满足或者不满足的可能，在逻辑中可以用“1”和“0”来表示。

逻辑关系中的“1”和“0”不表示数字，仅表示状态。当用“1”表示高电平，“0”表示低电平时，称为正逻辑关系，反之称为负逻辑关系。例如开关接通用逻辑 1 表示，开关断开用逻辑 0 表示，就是正逻辑。

1.5.1　与逻辑

图 1-5(a)表示一个简单与逻辑的电路，电压 V 通过开关 A 和 B 向灯泡 L 供电，只有 A 和 B 同时接通时，灯泡 L 才亮。A 和 B 中只要有一个不接通或二者均不接通时，则灯泡 L 不亮，其状态如表 1-6 所示。因此该电路可分析出输入输出关系为与逻辑。

功能描述：只有当一件事情(灯 L 亮)的几个条件(开关 A 与 B 都接通)全部具备之后，这件事情才会发生，这种关系称为与运算。

逻辑表达式：$L=A\cdot B$，式中小圆点“·”表示 A、B 的与运算，又称逻辑乘。在不致

引起混淆的前提下，乘号“·”可省略；也有用符号 ∧、∩表示与运算的。

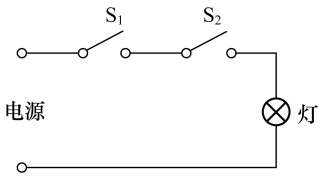


图 1-5　与逻辑关系电路图

表 1-6　与的逻辑功能表

开关 S_1	开关 S_2	灯 L
断开	断开	灭
断开	闭合	灭
闭合	断开	灭
闭合	闭合	亮

真值表：用 0、1 表示的与逻辑真值表如表 1-7 所示。

逻辑规则：0&0=0，0&1=0，1&0=0，1&1=1；有 0 出 0，全 1 出 1。

逻辑符号：与运算的逻辑符如图 1-6 所示，其中 A、B 表示输入，L 表示输出。

表 1-7　与的逻辑真值表

A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

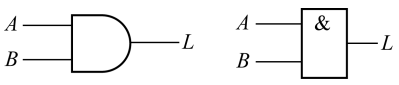


图 1-6　与的两种逻辑符号

1.5.2　或逻辑

图 1-7 为或逻辑电路，电源通过开关 A 或 B 向灯泡供电。只要开关 A 或 B 接通或二者均接通，则灯 L 亮；而当 A 和 B 均不通时，则灯 L 不亮，其状态如表 1-8 所示，可分析出输入输出关系为或逻辑。

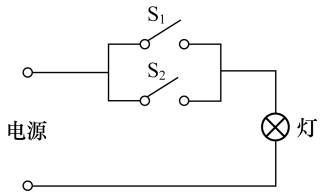


图 1-7　或逻辑关系电路图

表 1-8　或逻辑功能表

开关 S_1	开关 S_2	灯 L
断开	断开	灭
断开	闭合	亮
闭合	断开	亮
闭合	闭合	亮

功能描述：当一件事情(灯 L 亮)的几个条件(开关 A、B 接通)中只要有一个条件得到满足，这件事就会发生，这种关系称为或运算。

逻辑表达式：$L=A+B$，式中符号“+”表示 A、B 或运算，又称逻辑加，也有用符号 ∨、

∪来表示或运算。

真值表：同与运算一样，用 0、1 表示的或逻辑真值表如表 1-9 所示。

逻辑规则：0∣0=0，0∣1=1，1∣0=1，1∣1=1；有 1 出 1，全 0 出 0。

逻辑符号：或运算的逻辑符号如图 1-8 所示，A、B 表示输入变量，L 表示输出变量。

表 1-9　或逻辑真值表

A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

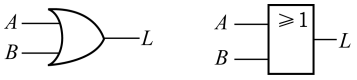


图 1-8　或的两种逻辑符号

1.5.3 非逻辑

如图 1-9 所示，电源通过 R 电阻向灯泡供电，当开关 A 闭合时，L 灯灭；而当开关 A 断开时，L 灯亮，其真值表如表 1-10 所示，分析得出为非逻辑关系。

功能描述：一件事情(灯亮)的发生是以其相反的条件为依据，这种逻辑关系为非运算。

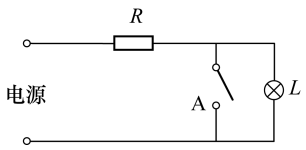


图 1-9　非逻辑关系电路图

表 1-10　非的逻辑真值表

A	L
0	1
1	0

逻辑表达式：$L=\overline{A}$

逻辑规则：~1=0，~0=1，~(10001)=01110

逻辑符号：非逻辑运算的逻辑符号如图 1-10 所示。

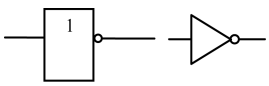


图 1-10　非的两种逻辑符号

1.5.4 复合逻辑运算

与非的逻辑运算如图 1-11、逻辑真值表如表 1-11 所示。

与非逻辑表达式：$L=\overline{A\cdot B}=\overline{AB}$

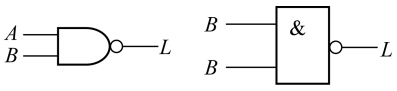


图 1-11　与非的两种逻辑符号

表 1-11　与非的逻辑真值表

A	B	L
0	0	1
0	1	1
1	0	1
1	1	0

或非的逻辑运算如图 1-12、逻辑真值表如表 1-12 所示。

或非逻辑表达式：$L=\overline{A+B}$

图 1-12　或非的逻辑符号

表 1-12　或非的逻辑真值表

A	B	L
0	0	1
0	1	0
1	0	0
1	1	0

异或的逻辑运算如图 1-13、逻辑真值表如表 1-13 所示。

异或逻辑表达式：$L = A \oplus B = \overline{A}B + A\overline{B}$

图 1-13　异或的两种逻辑符号

表 1-13　异或的逻辑真值表

A	B	L
0	0	0
0	1	1
1	0	1
1	1	0

参加异或运算的两个变量，如果两个相应位逻辑状态不同，则异或结果为 1，否则为 0。逻辑运算规则：0^0 = 0；0^1 = 1；1^0 = 1；1^1 = 0。

同或的逻辑运算如图 1-14、逻辑真值表如表 1-14 所示。

图 1-14　同或的逻辑符号

表 1-14　同或的逻辑真值表

A	B	L
0	0	1
0	1	0
1	0	0
1	1	1

同或逻辑表达式：$L = A \odot B = AB + \overline{A}\,\overline{B} = \overline{A \oplus B}$

此处两变量的异或与同或互为反函数。

与或非逻辑表达式：$L=\overline{AB+CD}$，其逻辑符号如图 1-15 所示。

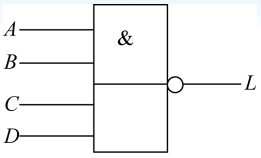


图 1-15　与或非的逻辑符号

例 1-9　试写出下图所示电路的逻辑表达式。

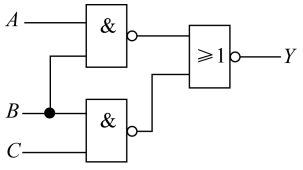


图 1-16　例题 1-9 电路图

从输入变量 A、B、C 开始逐个写出每个逻辑符号输出端的逻辑表达式为

$$Y=\overline{\overline{AB}+\overline{BC}}=AB\cdot BC=ABC$$

1.6　逻辑代数的基本定理和规则

1.6.1　逻辑代数的基本定律和恒等式

我们已经发现表 1-15 所示的逻辑代数的基本定律和恒等式。

表 1-15　常用逻辑代数定律和恒等式

定律名称	恒　等　式	定律名称	恒　等　式
0、1 律	$0+0=0$，$0+1=1$，$1+1=1$，$\overline{1}=0$， $0\cdot 0=0$，$1\cdot 0=0$，$1\cdot 1=1$，$\overline{0}=1$ $A+0=A$　$A+1=1$， $A\cdot 0=0$　$A\cdot 1=A$	反演律	$\overline{AB}=\overline{A}+\overline{B}$（与之非，等于非之或） $\overline{A+B}=A\cdot B$（或之非，等于非之与）
互补律	$A\cdot\overline{A}=0$，$A+\overline{A}=1$，$\overline{\overline{A}}=A$		
重叠律	$A+A=A$，$A\cdot A=A$	吸收律	$A+AB=A$，$A(A+B)=A$ $(A+B)(A+\overline{B})=A$，$AB+A\overline{B}=A$， $A+\overline{A}B=A+B$，$A(\overline{A}+B)=AB$ $AB+\overline{A}C+BC=AB+\overline{A}C$ $AB+\overline{A}C+BC(D+E)=AB+\overline{A}C$ $(A+B)(\overline{A}+C)(B+C)=(A+B)(\overline{A}+C)$
交换律	$A+B=B+A$，$A\cdot B=B\cdot A$		
结合律	$(A+B)+C=A+(B+C)$ $(A\cdot B)C=A(B\cdot C)$		
分配律	$A+BC=(A+B)(A+C)=$ $A+AB+AC+BC=A+BC$		

1.6.2 逻辑代数中的基本定理

1. 代入定理

所谓代入定理，就是在任何一个包含变量 A 的等式中，若以另外一个逻辑式代入式中所有 A 的位置，则等式仍然成立。代入规则可以扩展所有基本公式或定律的应用范围。

例 1-10　已知 $B(A+C)=BA+BC$，用 $A+E$ 代替 A 得

$$B[(A+E)+C]=B(A+E)+BC=BA+BE+BC$$

2. 对偶定理

所谓对偶定理，就是若两个逻辑式相等，则它们的对偶式也相等。所谓对偶式，就是对于任何一个逻辑式 Y，若将其中的“·”换成“+”，“+”换成“·”，0 换成 1，1 换成 0，则得到 L 的对偶式 L'。

在求对偶式时，需要注意要保持原式中的运算顺序不变，而且不属于单个变量上的反号应保留不变。

例 1-11　试求逻辑表达式 $L=(\overline{A}+B)(\overline{A}+C)$ 的对偶式。

解　根据对偶定理得 $L'=\overline{A}B+\overline{A}C$

3. 反演定理

对于任意一个逻辑表达式 L，若将其中所有的与(·)换成或(+)，或(+)换成与(·)；原变量换为反变量，反变量换为原变量；将 1 换成 0，0 换成 1；则得到的结果就是原函数的反函数。

利用反演定理，可以比较容易地求出原函数的反函数。但在使用反演定理时，还需注意遵守以下两条规则：

(1) 在变换时要保持原式中的运算顺序不变；

(2) 不属于单个变量上的反号应保留不变。

例 1-12　试求逻辑表达式 $L=A(\overline{B}+C)+\overline{C}D$ 的反函数。

解　根据反演定理得　$\overline{L}=\overline{A(\overline{B}+C)+\overline{C}D}=(\overline{A}+B\overline{C})(C+\overline{D})$

例 1-13　试求逻辑表达式 $L=\overline{(A+C)}\,\overline{(\overline{BC}+D)}(B+C)+AD$ 的反函数。

解　根据反演定理得　$\overline{L}=(\overline{AC})+\overline{(\overline{B}+\overline{CD})}+(\overline{B}\,\overline{C})(\overline{A}+\overline{D})$

也可以将 L 表达式左边看作一个整体考虑，即

$$\overline{L}=\overline{\overline{(A+C)}\,\overline{(\overline{BC}+D)}(B+C)+AD}=[(A+C)+(\overline{BC}+D)+\overline{B+C}](\overline{AD})$$

从中我们可以看到求解反函数的表达式形式不唯一，可能有多种形式。

在以上部分，我们依次学习了常见逻辑函数表达式的恒等式和三大定理，如需对逻辑

函数表达式进行证明，一般采用如下方法：

（1）真值表法。列出等式两边逻辑表达式的真值表，若真值表相同，则等式成立；

（2）对偶定理法。等式两边的对偶式仍相等；

（3）代入定理法。等式两边的同一变量若用同一函数代替，等式仍成立；

（4）公式法。利用逻辑代数中的定理和规则，将等式两边化为相同的形式，则等式成立。

例 1-14　$AB+\overline{A}C+BCD=AB+\overline{A}C$（多余项定理），可以通过公式推导证明等式成立。

$$\begin{aligned} AB+\overline{A}C+BCD &= AB+\overline{A}C+BCD(A+\overline{A}) \\ &= AB+\overline{A}C+ABCD+ABCD \\ &= AB(1+CD)+\overline{A}C(1+BD) \\ &= AB+\overline{A}C \end{aligned}$$

1.7　逻辑函数的表示方法

逻辑代数中用以描述逻辑关系的函数称为逻辑函数，前面叙述的与或非、与非、或非、异或等都是逻辑函数。逻辑函数通常有逻辑真值表、逻辑表达式、逻辑电路图、卡诺图、波形图等几种表示方法。

1.7.1　逻辑函数的建立

真值表法：采用一种表格来表示逻辑函数的运算关系，其中输入部分列出输入逻辑变量的所有可能组合，输出部分给出相应的输出逻辑变量值。

表达式法：采用逻辑表达式来描述输出变量和输入变量的逻辑关系。

逻辑图法：采用规定的图形符号，来构成逻辑函数运算关系的网络图形。

卡诺图法：卡诺图是一种几何图形，可以用来表示和简化逻辑函数表达式。

波形图法：采用数字信号波形描述每个时间段里输入变量与输出变量之间的取值(0/1)。

例 1-15　三个人表决一件事情，结果按照“少数服从多数”的原则决定，建立逻辑函数。

解　将实际问题中的逻辑关系表达为逻辑函数，需要以下 3 个步骤：

（1）定义输入变量、输出变量。将 3 个人的意见设置为输入变量 A、B、C，并规定只能有同意和不同意两种意见，将表决结果设置为输出变量 L，显然也只有通过和不通过两种情况。

（2）定义变量状态的逻辑值。对于输入变量 A、B、C，同意为逻辑 1、不同意为逻辑 0，对于输出变量 L，表决通过为逻辑 1、不通过为逻辑 0。

（3）按照以上逻辑规则，列出真值表(表 1-16)。

由真值表可以看出，输出变量 L 是由输入变量 A、B、C 的状态决定的。

$$L=f(A,\ B,\ C)$$

表 1-16　三人表决电路的逻辑真值表

A	B	C	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

逻辑函数的几种表示方法之间可以相互转换，例如由真值表⇌逻辑表达式。

(1) 找出真值表中使逻辑函数 $L=1$ 的那些输入变量取值的组合。

(2) 每组输入变量取值的组合对应一个乘积项，其中取值为 1 的取原变量，取值为 0 的取反变量。

(3) 将这些乘积项相加(或)，即得 L 的逻辑式。

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

用真值表表示逻辑函数的优点是直观、明了、唯一，可直接看出逻辑电路输出与输入之间的关系，但变量较多时较烦琐。

当然逻辑表达式→逻辑真值表的方法为，将输入变量取值的所有组合状态(枚举法)逐一带入逻辑表达式求出函数值，即得真值表。

其他几种表示方法之间的转换，我们将在随后逐一学习。

1.7.2 逻辑函数表达式的形式

一个逻辑函数可以有多种不同的逻辑表达式，如与或表达式、或与表达式、与非-与非表达式、或非-或非表达式以及与或非表达式等。例如

$$Y=AC+\overline{C}D\cdots\cdots\cdots\cdots \text{ 与或表达式}$$

$$Y=(A+\overline{C})(C+D)\cdots\cdots \text{ 或与表达式}$$

$$Y=\overline{\overline{AC}\cdot\overline{\overline{C}D}}\cdots\cdots\cdots\cdots \text{ 与非 - 与非表达式}$$

$$Y=\overline{\overline{(A+\overline{C})}+\overline{(C+D)}}\cdots \text{ 或非 - 或非表达式}$$

$$Y=\overline{\overline{AC}+\overline{C}D}\cdots\cdots\cdots\cdots \text{ 与或非表达式}$$

其中与或表达式是将若干与项进行或逻辑运算构成的表达式。或与表达式是将若干或项进行与逻辑运算构成的表达式。

与或表达式转换为与非-与非表达式是利用摩根定理(反演律)进行转换的。

或与表达式转换为或非-或非表达式也是利用反演律进行转换的。

1. 最小项表达式

在有 n 个变量的逻辑函数中，若 m 为包含 n 个因子的乘积项，而且这 n 个变量均以原变量或反变量的形式在 m 中出现一次，则称 m 为该组变量的最小项。即包含所有变量且每个变量只出现一次的乘积项为最小项。如三变量 A、B、C 有 8 个最小项，并用 m_i 来表示，其中 i 为 ABC 状态对应的进制值，如 ABC 状态组合为 111，即 m_7，$\overline{AB}C$ 状态组合为 101，即 m_5。

$$\begin{aligned}&\overline{A}\,\overline{B}\,\overline{C}+\overline{A}\,\overline{B}C+\overline{A}B\overline{C}+\overline{A}BC+A\,\overline{B}\,\overline{C}+A\overline{B}C+AB\overline{C}+ABC\\&=m_0+m_1+m_2+m_3+m_4+m_5+m_6+m_7\end{aligned}$$

最小项的性质：

① 对于 n 变量逻辑函数有 2^n 个最小项。

② 在输入变量的任一取值下，有且仅有一个最小项的值为 1。

③ 全体最小项之和为 1。

④ 对于输入变量的任何同一组取值，任意两个最小项的乘积为 0。

⑤ 相邻的两个最小项之间仅有一个变量不同，其余变量均相同。如 ABC 三变量的最小项 $\overline{A}B\overline{C}$ 和 $\overline{A}BC$ 为相邻的两个最小项，求和结果为

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

例 1-16　将 $L(A,B,C)=A\overline{B}+\overline{A}C$ 化成最小项表达式。

解　利用公式 $A+\overline{A}=1$，可写出逻辑函数的最小项表达式为

$$\begin{aligned}L(A,B,C)&=A\overline{B}+\overline{A}C\\&=A\overline{B}(C+\overline{C})+\overline{A}(B+\overline{B})C\\&=\overline{A}\,\overline{B}C+\overline{A}BC+A\,\overline{B}\,\overline{C}+A\overline{B}C\\&=m_1+m_3+m_4+m_5\\&=\sum m(1,3,4,5)\end{aligned}$$

2. 最大项表达式

在有 n 个变量的逻辑函数中，若 M 为 n 个变量之和，且每个变量均以原变量或反变量的形式在 M 中出现一次，则 M 为该组变量的最大项。

最大项的性质：

① 对于 n 变量逻辑函数有 2^n 个最大项。

② 在输入变量的任一取值下，有且仅有一个最大项的值为 0。

③ 全体最小项之积为 0。

④ 任意两个最大项之和为 1。

⑤ 相邻的两个最大项之积，结果为消去一对互补因子，只留下公共因子。

相邻：仅一个因子不同的最大项。如 $\overline{A}+B+C$ 和 $\overline{A}+B+\overline{C}$，这两个最大项之积为

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

⑥ 相同编号的最小项和最大项互为反函数，$m_i=\overline{M_i}$。

例 1-17　已知某逻辑电路的真值表如表 1-17 所示，写出最小项表达式和最大项表达式。

解　最小项表达式：将 $L=1$ 的各个最小项相加，即

$$\begin{aligned}L(A,B,C)&=m_3+m_5+m_6\\&=\sum m(3,5,6)\\&=\overline{A}\cdot B\cdot C+A\cdot\overline{B}\cdot C+A\cdot B\cdot\overline{C}\end{aligned}$$

表 1-17　逻辑电路的逻辑真值表

A	B	C	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

最大项表达式：将 $L=0$ 的各个最大项相乘，即

$$\begin{aligned}L(A,B,C)&=M_0\cdot M_1\cdot M_2\cdot M_4\cdot M_7\\&=\prod M(0,1,2,4,7)\\&=(A+B+C)\cdot(A+B+\overline{C})\cdot(A+\overline{B}+C)\cdot(\overline{A}+B+C)\cdot(\overline{A}+\overline{B}+\overline{C})\end{aligned}$$

1.7.3 逻辑表达式的代数化简法

化简的目的：降低电路实现的成本，以较少的门实现电路。

公式化简法：反复应用基本公式和常用公式，消去多余的与项和多余的因子。

利用代数法化简逻辑函数的方法主要如下：

（1）并项法。利用公式 $AB + A\overline{B} = A$ 将两个与项合并为一个，消去其中的一个变量。

（2）吸收法。利用公式 $A + AB = A$ 吸收多余的与项。

（3）消因子法。利用公式 $A + \overline{A}B = A + B$，消去与项多余的因子。

（4）消项法。利用公式 $AB + \overline{A}C = AB + \overline{A}C + BC$ 进行配项，以消去更多的与项。

（5）配项法。利用公式 $A + A = A$，$A + \overline{A} = 1$ 配项，简化表达式。

在实际化简时一般化简顺序如下：

（1）合并同类项。找公共变量。

（2）利用逻辑恒等式、多余项定理、反演规则等进行逻辑项的合并、化简。

（3）检查是否最简，不是就继续化简或者根据题目要求转换逻辑表达式形式。

一般将其中包含的与项数最少，且每个与项中变量数最少的与-或表达式称为最简与-或表达式。

例 1-18　利用配项法将逻辑函数 $L = AB + \overline{A}\,\overline{C} + B\overline{C}$ 化成最简与或式。

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

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

例 1-19　化简下列逻辑函数 $\begin{cases} L_1 = A\overline{B} + \overline{A}\,\overline{B} + ACD + \overline{A}CD \\ L_2 = AB(C + D) + D + \overline{D}(A + B)(\overline{B} + \overline{C}) \end{cases}$

解　$L_1 = A\overline{B} + \overline{A}\,\overline{B} + ACD + \overline{A}CD = (A + \overline{A})\overline{B} + (A + \overline{A})CD = \overline{B} + CD$

$L_2 = AB(C + D) + D + \overline{D}(A + B)(\overline{B} + \overline{C})$

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

$= ABC + \underline{ABD + D} + A\overline{B} + A\overline{C} + B\overline{C}$

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

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

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

例 1-20　通常在一片集成电路芯片中只有一种门电路，为了减少门电路的种类，需要对逻辑函数表达式进行变换。已知 $L = AB\overline{D} + \overline{A}\,\overline{BD} + ABD + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}CD$。

（1）求最简的与-或式，并画出相应的逻辑图；

（2）画出仅用与非门实现的电路。

解　（1）求最简的与-或式，并画出相应的逻辑图如图 1-17(a)所示。

$$L = AB\overline{D} + \overline{A}\,\overline{BD} + ABD + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}CD$$

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

$$= AB + \overline{A}\,\overline{BD} + \overline{A}\,\overline{B}D$$

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

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

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

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

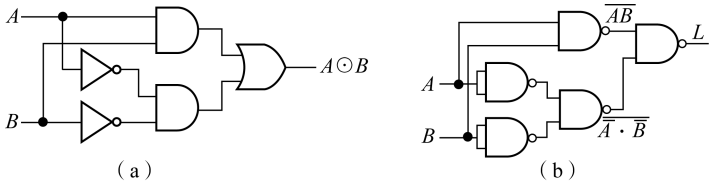


图 1-17　对应逻辑电路图

(2) 画出仅用与非门实现的电路，如图 1-17(b)所示。

代数化简方法技巧强，逻辑代数与普通代数的公式易混淆，如 $A + A = 2A$ 等，化简过程中要求学生对逻辑代数的基本定律和公式熟练掌握。同时对代数化简后得到的逻辑表达式是否是最简式判断有一定困难，所以我们优先采用卡诺图法，得到最简逻辑表达式和逻辑电路。

1.7.4 逻辑函数的卡诺图化简法

卡诺图是由美国工程师卡诺首先提出的。将 n 变量的全部最小项各用一个小方块表示，并使具有逻辑相邻性的最小项在几何位置上也相邻地排列起来，所得到的图形叫作 n 变量最小项的卡诺图(图 1-18)。

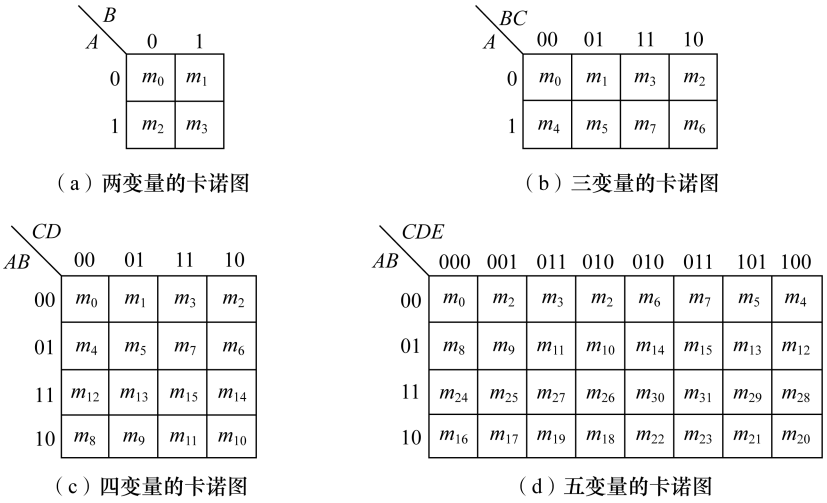


图 1-18　两变量到五变量的最小项构成的卡诺图

逻辑相邻项：仅有一个变量不同其余变量均相同的两个最小项称为逻辑相邻项。如左右相邻(m_0 和 m_1)，上下相邻(m_0 和 m_4)，同一行的最左边和最右边(m_0 和 m_2)，同一列的最上边和最下边(m_0 和 m_8)等。

1. 用卡诺图表示逻辑函数

方法一：先转化为最小项表达式。

① 把已知逻辑函数式化为最小项之和形式。

② 将函数式中包含的最小项在卡诺图对应的方格中填 1，其余方格中填 0(有时也可用空格表示)，就可以得到相应的卡诺图。

③ 任何逻辑函数都等于其卡诺图中为 1 的方格所对应的最小项之和。

方法二：根据函数表达式直接填卡诺图。

例 1-21　画出逻辑函数 $L(A, B, C, D) = \sum m(0, 1, 2, 3, 4, 8, 10, 11, 14, 15)$ 的卡诺图。

解　如图 1-19 所示。

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

图 1-19　例 1-21 卡诺图

例 1-22　已知逻辑函数的卡诺图如图 1-20 所示，写出该函数的逻辑表达式。

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

图 1-20　例 1-22 卡诺图

解　因为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，所以把卡诺图中填入 1 的那些方格所对应的最小项相加即可得到逻辑表达式为

$$Y = \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\overline{D} + AB\,\overline{C}\overline{D} + ABCD + A\,\overline{B}\,\overline{C}D + A\overline{B}C\overline{D}$$

2. 用卡诺图化简逻辑函数

化简依据：逻辑相邻性的最小项可以合并，并消去互补因子。

卡诺图化简法的步骤：

① 画出逻辑函数的卡诺图。

② 画圈，合并最小项；合并个数为 2^n 个(如 1、2、4、8 等)，圈尽可能大(乘积项中含因子数最少)；圈尽可能少(乘积项个数最少)；每个圈中至少有一个最小项仅被圈过一次，以免出现多余项(多画圈)。

③ 将包围圈所得的乘积项相加，就可得到简化后的与或表达式。

注意：卡诺图中所有的 1 都必须圈到，不能合并的 1 单独画圈；逻辑函数的化简结果可能不唯一。

例 1-23 用卡诺图化简法将逻辑函数 $Y = A\overline{C} + \overline{A}BC + B\overline{C} + ACD$ 化简为最简与或表达式。

解 首先填写卡诺图，然后按照合并最小项的规则合并最小项，如图 1-21 所示。

	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	0
10	1	1	1	0

图 1-21 例 1-23 卡诺图

化简后的逻辑函数为 $Y = \overline{A}B + A\overline{C} + AD$

3. 具有无关项的逻辑函数及其化简

实际中经常会遇到这样的问题，即输入变量的取值是任意的，就是在输入变量的某些取值下函数值是 1 还是 0 皆可，并不影响电路的功能。在这些变量取值下，其值等于 1 的那些最小项称为任意项。还有一种情况是输入变量的取值不是任意的，而是对输入变量的取值有所限制；对输入变量取值所加的限制称为约束。

约束项和任意项统称为逻辑函数式中的无关项。可以认为无关项包含在函数式中，也可以认为不包含在函数式中，那么在卡诺图中对应的位置上就既可以填入 1，也可以填入 0。为此，在卡诺图中用“×”表示无关项。

在含有无关项逻辑函数的卡诺图化简中，它的值可以取 0 或取 1，具体取什么值，可以根据使函数尽量得到简化而定。

无关项的运用时注意以下问题：

(1) 利用无关项在卡诺图中可以将函数表达式化简更简单；

(2) 无关项可能代表着不稳定因素(设计的可靠性验证)，需要大胆探索、小心求证。

例 1-24 设计一个逻辑电路，能够判断一位十进制数是奇数还是偶数。当十进制数为奇数时，电路输出为 1；当十进制数为偶数时，电路输出为 0。

解 (1)根据题意，设 1 位十进制数可用 4 位二进制数表示，如输入变量为 A、B、C、D，输出变量为 L，根据奇偶数判定，列出表 1-18 所示的真值表；

（2）画出卡诺图 1-22；

（3）用卡诺图化简：$L=D$。

例 1-25　化简下列函数：$Y = AC + \overline{A}\,\overline{B}C$，$\overline{B}\,\overline{C} = 0$ 为约束条件。

解　Y的卡诺图如图 1 - 23 所示，约束条件$\overline{B}\,\overline{C}=0$，包含两个最小项$A\,\overline{B}\,\overline{C}$、$\overline{A}\,\overline{B}\,\overline{C}$，在卡诺图中标“×”符号。

合并相邻最小项求函数的最简与或表达式，得到 $Y = AC + \overline{B}$。

表 1-18　逻辑真值表

A	B	C	D	L
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	×
1	0	1	1	×
1	1	0	0	×
1	1	0	1	×
1	1	1	0	×
1	1	1	1	×

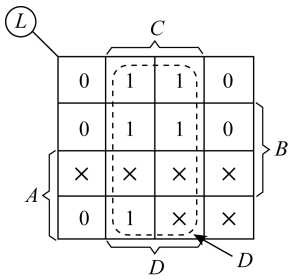


图 1-22　例 1-24 卡诺图化简

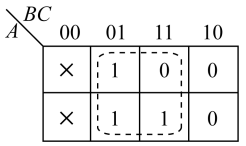


图 1-23　例 1-25 卡诺图化简

1.8　用 Multisim 软件化简逻辑函数表达式

利用 Mulitism 中自带的逻辑变换器，将所需化简的逻辑函数的真值关系输入进去，点击变换栏第三个按钮即可得到逻辑函数最简形式，此外点击变换栏第二个按钮可以得到逻辑函数的最小项表达式，点击最后一个按钮还可以得到相应的与非逻辑图。

例 1-26　以 $F = ABC + ABD + A\overline{C}D + \overline{C}\overline{D} + A\overline{B}C + AC\overline{D} + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BCD$ 为例，采用代数化简法、卡诺图化简法和 Mulitism 软件化简三种方法对逻辑函数表达式进行化简。

（1）代数化简法

从中可以看出，4 变量逻辑函数化简步骤非常复杂、化简过程可能也不一样，对于 4 变量以上逻辑函数的化简就更为复杂，而且对于化简之后的逻辑函数是否为最简形式的判断也很难一眼看出。

$$F = ABC + ABD + A\overline{C}D + \overline{CD} + A\overline{B}C + AC\overline{D} + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BCD$$

$$= AC(B + \overline{B}) + BD(A + \overline{A}C) + \overline{C}(AD + \overline{D}) + AC\overline{D} + \overline{A}\,\overline{B}\,\overline{C}$$

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

$$= AC + ABD + BCD + A\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + \overline{CD}$$

$$= A(C + \overline{C}) + ABD + BCD + \overline{A}\,\overline{B}\,\overline{C} + \overline{CD}$$

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

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

$$F = ABC + ABD + A\overline{C}D + \overline{CD} + A\overline{B}C + AC\overline{D} + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BCD$$

$$= BC(A + \overline{A}D) + BD(A + \overline{A}C) + \overline{C}(AD + \overline{D}) + AC(B + B) + \overline{A}\,\overline{B}\,\overline{C}$$

$$= ABC + BCD + ABD + BCD + A\overline{C} + \overline{CD} + AC + \overline{A}\,\overline{B}\,\overline{C}$$

$$= ABC + BCD + ABD + A(\overline{C} + \overline{C}) + \overline{CD} + \overline{A}\,\overline{B}\,\overline{C}$$

$$= ABC + BCD + ABD + A + \overline{CD} + \overline{A}\,\overline{B}\,\overline{C}$$

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

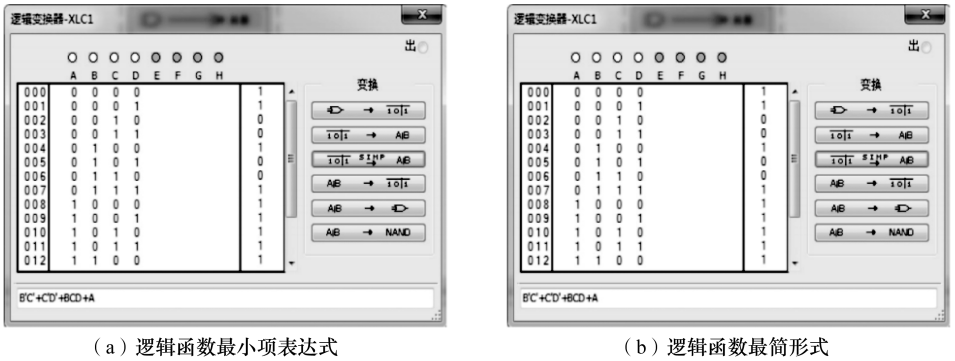
$$= A + \overline{B}\,\overline{C} + BCD + \overline{CD}$$

(2) 卡诺图化简法

利用卡诺图化简法，最简形式为 $F = A + \overline{B}\,\overline{C} + BCD + \overline{CD}$。可见，4 变量逻辑函数的卡诺图法化简相对也比较复杂，包围圈数比较多，容易遗漏导致化简错误。对于 4 变量以上的多变量逻辑函数的化简，如果继续用卡诺图法化简，那么卡诺图就变得庞大复杂，如果用到的包围圈也比较多，那就变得非常复杂，十分不利于化简，甚至化简出错。

(3) Mulitism 软件化简

将 4 变量逻辑函数的真值关系输入逻辑函数变换器中，得到图 1-24(a)所示逻辑函数最小项表达式，点击运行得到图 1-24(b)所示逻辑函数最简形式。显然，利用 Mulitism 14.0 软件法求逻辑函数的最简形式比公式法和卡诺图法简单快捷，而且不容易出错。



(a) 逻辑函数最小项表达式　　(b) 逻辑函数最简形式

图 1-24　例 1-26 的 Mulitism 软件化简

对于 4 变量以上的多变量逻辑函数的化简，如图 1-24 所示的 ABCDEFGH 共 8 个输入变量，Mulitism14.0 也能轻松做到化简；不仅如此，其还可以化简带有约束项的多变量逻辑函数。

例 1-27　用 Mulitism 软件化简 6 变量带有约束项的逻辑函数

$$F = \sum m(2, 4, 7, 9, 11, 14, 16, 28, 30, 33, 41, 43, 50, 51, 55, 60, 62, 63) + \sum d(3, 5, 6, 20, 31, 44, 53, 57)$$

解　此逻辑函数变量较多，最小项也非常多，无论是采用公式法还是卡诺图法都不好化简，而且对于 6 变量的逻辑函数对应的卡诺图也比较复杂(图 1-25)。

(a) 逻辑函数最小项表达式　　(b) 逻辑函数最简形式

图 1-25　例 1-27 的 Mulitism 软件化简

图 1-25(a)为 6 变量逻辑函数对应最小项，图 1-25(b)为化简之后的最简形式

$$F = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}E + \bar{A}\bar{B}DE\bar{F} + \bar{A}B\bar{C}\bar{E}\bar{F} + A\bar{B}\bar{D}\bar{E}F + \bar{B}C\bar{D}F + AB\bar{C}\bar{D}E + BCD\bar{F} + ABDEF$$

小结：

(1) 对于变量数小于 4 的逻辑函数的化简，公式法、卡诺图法、Mulitism14.0 软件法都能化简，且用这 3 种方法化简都比较简单；

(2) 对于变量数大于 4 的多变量逻辑函数的化简，公式法和卡诺图法变得十分复杂，难以化简，而 Mulitism14.0 软件法化简却十分简单，其软件法用于化简多变量带有约束条件的逻辑函数同样适用；

(3) 采用公式法和卡诺图法进行化简的逻辑函数需要对化简结果进行判定是否为最简形式，而采用 Mulitism14.0 软件法得出的结果不需进行判定，软件法化简结果即为最简形式。

1.9　课程思政育人案例

1.9.1　通信技术发展故事

为了尽快地把有用的信息传递到远方去，古代的中国在遥远的边塞通向京城的道路上

修建了许多烽火台，边境一有战事或其他紧急情况，就一站接着一站地点起烽火，把信息传到京城帝王那里。但是，烽火台造价很高，还需要昼夜派人驻守瞭望，又不能传达信息的具体内容，所以大量的信息还得靠人力、驿站传递。

公元前 490 年，希腊人在马拉松这个地方打败了波斯军队，赢得了保卫国土的胜利。为了让首都人民尽快地分享这一喜讯，在没有任何交通工具的情况下，希腊军队的将领就派了一个叫斐迪辟的士兵，徒步从马拉松平原一刻不停地跑到了当时希腊的首都雅典。当斐迪辟向首都人民报告了胜利的喜讯后，终于因极度疲劳而倒下牺牲了。为了永远纪念这位英雄，人们就把他所跑的全路路程(42 195 米)列为长跑比赛的一个项目，并命名为马拉松赛跑。在古代，人们传递信息是多么地困难啊。古代人们极力地寻找最快的传递信息的方法，然而只能在神话小说里创造出“千里眼”和“顺风耳”，以寄托自己的理想。

“顺风耳”的理想终于由一名美国画家实现了，他就是电报机的发明者——莫尔斯。19 世纪初期的一个秋天，在一艘航行的船上，一群旅客正围着一个名叫杰克逊的医生，听他讲述发明不久的电磁铁：一块马蹄形的、缠着导线的铁块，一通电就会产生吸引力；而电流一断，吸着的铁性物质便都掉了下来。大家都被这新鲜事吸引住了。当时莫尔斯也正好在场，他在感到好奇的同时，却比周围其他人想得更深、更远。他向杰克逊问了一个问题：电流在导线里流动的速度快不快(可见莫尔斯毫无电学知识)？当他知道电流的速度快得在几千千米长的电线里，一瞬间就能通过时，一个大胆而又新奇的想法，在他头脑中出现了。

海轮上的巧遇，改变了莫尔斯的生活道路。他放弃了自己心爱的绘画事业，开始了发明电报的艰苦研究工作。十多个春秋过去了，他终于获得了成功，利用电流一断一通的原理，发明了电报机和用点画表达信息的电码——“莫尔斯电码”，使通信变得便利了。

电报虽然能迅速地传递信息内容，但是，发报人先得把信息内容转换成符号，按一定的操作规律把这种符号发送到收报人那里。收报人收到这种符号后，再利用电码把它所代表的内容翻译出来，还是比较麻烦。如果能直接传送语言信号那该多好啊！人类是永远也不会满足的，发明了电报后，又在给自己出新的难题了。

第一个向这个难题宣战并获得胜利的是美国一位研究聋哑语的教师贝尔。贝尔开始研究这个难题时，对电学一窍不通。但是他在研究人的声带过程中想到：声音是靠声带的振动而产生的，能不能把这种振动通过电流的强弱变化送出去呢？能不能把物体的振动变成变化的电流，再把变化的电流还原成物体的振动发出声音来呢？这可真是个大难题。

为实现自己的理想，贝尔来到了千里之外的华盛顿，从头开始学习电学知识。经过 3 年的发奋努力，他在机械工匠沃特森的帮助下，终于在 1876 年制成了世界上第一套话筒和听筒。用电流传送声音的理想实现了。但是，当时的电话杂音太大，传送距离又太短，离实际应用还有一段距离。

1878 年，大发明家爱迪生对电话机作了较大的改进，使通话距离增长到 100 多千米。1915 年，贝尔又进一步解决了由于长距离通话给电话机带来的一系列技术性问题，终于在这一年的美国，架起了第一条长达 6 000 多千米的电话线路。

现在，电话已成了人们生活中不可缺少的通信工具。

1.9.2 莫尔斯电码

Morse Code 又译作摩斯电码，莫尔斯电码是将 ASCII 数据转换为二进制位的有效手段。我们假设“点”为零，“破折号”为一。使用查找表可以将 ASCII 字符转换为二进制值。其中 SOS 是国际使用的遇难信号，翻译成莫尔斯电码是三声短三声长再三声短，是救救我们的船(Save Our Ship)的三个单词首写字母。最早发出 SOS 信号的，是 1912 年在北大西洋与冰山相撞而沉没的豪华客船泰坦尼克号。

1.9.3 第二次世界大战中的密码破译

以钓鱼手段破解密码的手段，后来在第二次世界大战中被美军原样用回到日本头上。当时，美军从日军一系列电报中，发现频繁出现“AF”代码，估计应该是指中途岛。美军便用浅显的英语发了一份作为诱饵的电报，报告中途岛的淡水设备发生故障。果然不久美军截获的一份日军密码电报声称：AF 可能缺少淡水。至于美军截杀联合舰队司令山本五十六的行动，则应归功于新西兰海军从一艘被撞沉的日军潜艇上得到了日本海军最新版的密码本，日军此后更换的密码于是迅速被美军破译。

1.9.4 逻辑函数化简案例

人生中，很多时候我们总是面临无数次的激烈竞争，在竞争中也总会遇到各种各样的困难，甚至会出现在参与某种竞争中由于自身能力不足而被淘汰的情况，如果我们能从逻辑函数的化简法受到启发，就可以运用卡诺图化简和公式化简将不必要的、多余的因子消去。警惕的学生会不断充电，增加个人实力，提高自己的逻辑思维，通过努力学习增加自身核心竞争力，使自己在各行各业处于佼佼者的地位。从另一方面来看，卡诺图化简和公式化简可以使得逻辑函数表达式更加直观、简便，但这种化简都要遵守一定的规则和定律，否则有可能出现将式子越化越烦琐的情况，进而延伸到我们的生活必须要在一定的框架下，循规守矩，遵纪守法，人才能在社会上正常有序的生活，才会更加自由、舒适，从而培养学生严于律己、遵纪守法的意识。

1.9.5 名人轶事——布尔、香农

乔治·布尔(1815—1864 年)，生于英格兰的林肯(图 1-26)。乔治·布尔是皮匠的儿子，由于家境贫寒，布尔不得不在协助养家的同时为自己能受教育而奋斗。尽管他考虑过以牧师为业，但最终还是决定从教，1835 年他开办了自己的学校。在备课的时候，布尔不满意当时的数学课本，便决定阅读伟大数学家(如法国数学家拉格朗日)的论文。1847 年，布尔出版了《逻辑的数学分析》，这是它对符号逻辑诸多贡献中的第一次。1854 年，他出

版了《思维规律的研究》，这是他最著名的著作。在这本书中布尔介绍了现在以他的名字命名的布尔代数。布尔撰写了微分方程和差分方程的课本，这些课本在英国一直使用到19世纪末。

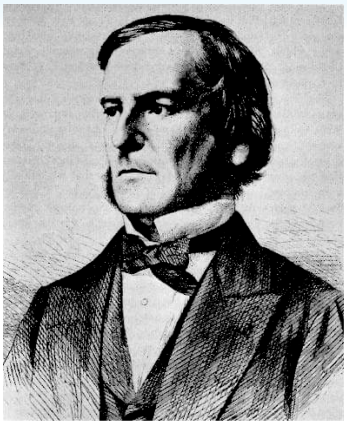


图 1-26　英国数学家乔治·布尔

克劳德·艾尔伍德·香农(1916—2001年)是美国数学家、信息论的创始人(图1-27)。他出生于美国密歇根州的佩托斯基，并且是爱迪生的远房亲戚。1936年毕业于密歇根大学并获得数学和电子工程学士学位。1940年获得麻省理工学院(MIT)数学博士学位和电子工程硕士学位。1941年他加入贝尔实验室数学部，工作到1972年。香农于1940年在普林斯顿高级研究所期间开始思考信息论与有效通信系统的问题。经过8年的努力，香农在1948年6月和10月在《贝尔系统技术杂志》上连载发表了具有深远影响的论文《通信的数学原理》。1949年，香农又在该杂志上发表了另一著名论文《噪声下的通信》。在这两篇论文中，香农阐明了通信的基本问题，给出了通信系统的模型，提出了信息量的数学表达式，并解决了信道容量、信源统计特性、信源编码、信道编码等一系列基本技术问题。两篇论文成为了信息论的奠基性著作。1938年香农在MIT获得电气工程硕士学位，硕士论文题目是《A Symbolic Analysis of Relay and Switching Circuits》(继电器与开关电路的符号分析)。当时他已经注意到电话交换电路与布尔代数之间的类似性，即把布尔代数的“真”与“假”和电路系统的“开”与“关”对应起来，并用1和0表示。于是他用布尔代数分析并优化开关电路，这就奠定了数字电路的理论基础。

图 1-27　信息论的创始人克劳德·香农

1.9.6 中国汉字编码

字符集编码是指对多个字符(通常在几十到几万个不等)进行整合封装成一个文件所使用的编码，外部程序通过这种编码就可以从字符集文件中调用指定的字符。我们常见的计

算机字体文件就使用了字符集编码，通过输入法输入文字或者浏览网页时都会通过指定的字符集编码从字体文件中调用字符。以下是常见的汉字字符集编码。

GB2312 编码：1981 年 5 月 1 日发布的简体中文汉字编码国家标准。GB2312 对汉字采用双字节编码，收录 7 445 个图形字符，其中包括 6 763 个汉字。

BIG5 编码：台湾地区繁体中文标准字符集，采用双字节编码，共收录 13 053 个中文字，1984 年实施。

GBK 编码：1995 年 12 月发布的汉字编码国家标准，是对 GB2312 编码的扩充，对汉字采用双字节编码。GBK 字符集共收录 21 003 个汉字，包含国家标准 GB13000—1 中的全部中日韩汉字，和 BIG5 编码中的所有汉字。

GB18030 编码：2000 年 3 月 17 日发布的汉字编码国家标准，是对 GBK 编码的扩充，覆盖中文、日文、朝鲜语和中国少数民族文字，其中收录 27 484 个汉字。GB18030 字符集采用单字节、双字节和四字节三种方式对字符编码。兼容 GBK 和 GB2312 字符集。

Unicode 编码：国际标准字符集，它将世界各种语言的每个字符定义一个唯一的编码，以满足跨语言、跨平台的文本信息转换。Unicode 采用四个字节为每个字符编码。

UTF-8 和 UTF-16 编码：Unicode 编码的转换格式，可变长编码，相对于 Unicode 更节省空间。UTF-16 的字节序有大端编码(big-endian)和小端编码(little-endian)之别。

本章小结

本章介绍了二进制数、十进制数、八进制数、十六进制数和 BCD 码，要准确地掌握各种数制和 BCD 码之间转换的方法。

逻辑代数是一种适用于逻辑推理，研究逻辑关系的主要数学工具。本章所介绍的逻辑代数的基本公式、定律和运算规则，在实际化简逻辑函数式中十分有用，应熟记这些公式、定律、规则，并能灵活运用。

逻辑函数通常有 5 种表示方法，即逻辑函数式、真值表、卡诺图、逻辑电路图和波形图。这 5 种之间可以任意地相互转换，在逻辑电路的分析和设计中经常会用到这些方法。依据具体的使用情况，可以选择最适当的一种方法表示逻辑函数。

逻辑函数的化简是分析和设计数字电路的重要环节，也是本章的重点。本章所介绍的两种化简方法：代数化简法和卡诺图化简法，各有所长，又各有不足，应熟练掌握。代数化简法适用于任何复杂的逻辑函数，它不受任何条件的限制。但由于这种方法没有固定的步骤可循，因此在化简较复杂的逻辑函数时，不仅需要熟练地运用多种公式和定律，而且需要有一定技巧和经验。卡诺图化简法比较直观、简单，也容易掌握，因为它有一定的化简步骤，初学者容易掌握。但在逻辑变量超过 5 个时没有太大的实用意义，将失去直观、简单的优点。

在实际逻辑问题中，输入变量之间常存在一定的制约关系，即存在约束。在逻辑函数化简时，充分利约束条件可使逻辑表达式更简化。

练 习 题

1-1　将下列十进制数转换为二进制数、八进制数、十六进制数。

(1) 22.75　　(2) 47.25　　(3) 16.8125

1-2　将下列十六进制数转换为二进制数。

(1) 7F.4　　(2) ABC.DE

1-3　将下列二进制数转换为八进制数、十进制数。

(1) 10011100　　(2) 1100.110101

1-4　求下列进制数对应的 8421BCD、余 3 码。

(1) $(79)_{10}$　　(2) $(723.14)_{10}$　　(3) $(1101101)_2$

1-5　求出下列进制数的原码、反码、补码。

(1) $(+1001)_2$　　(2) $(-1001)_2$

1-6　用 8 位二进制补码表示下列十进制数。

(1) $(+125)_{10}$　　(2) $(-120)_{10}$　　(3) $(-99)_{10}$

1-7　填空题。

(1) 一位 16 进制数可以用_____位二进制数来表示。

(2) $(A3)_{16}$、$(10100011)_2$、$(000101100011)_{8421BCD}$、$(203)_8$ 几个数据中，与十进制数 $(163)_{10}$ 不相等的是_____。

(3) 当逻辑函数有 n 个变量时，共有_____个变量取值组合。

(4) n 个变量可以构成_____个最小项。

(5) 逻辑函数的表示方法有_____、_____、_____、_____几种。

(6) 逻辑函数的化简方法主要有_____和_____。

(7) 2023 个 1 进行异或运算后的结果为_____。

(8) 函数 $F = \overline{A} + B + \overline{C}D$ 的反函数是_____，对偶式是_____。

(9) 已知函数的对偶式为 $\overline{A\overline{B}} + \overline{\overline{C}D} + BC$，则它的原函数为_____。

(10) 逻辑函数表达式 $F(A, B, C) = \sum m(0, 3, 5, 6)$ 和下列哪些逻辑表达式相等：□$F = A \oplus B \oplus C$，□$F = \overline{ABC}$，□$F = A \oplus B \oplus \overline{C}$，□$F = A \oplus B \oplus \overline{C}$

(11) 逻辑函数表达式 $F=\overline{A}B+\overline{CD}$ 为 1 的最小项表达式为_____。

1-9　求下列逻辑表达式的最小项表达式。

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

（2）$F = AB + \overline{A}\,\overline{C}$

（3）$F = AB + AC + BC$

1-10　某逻辑函数有三个输入端 ABC，当输入信号有奇数个 1 时，输出为 1，否则输出为 0，列出此逻辑电路的真值表，写出其逻辑函数 F 的逻辑表达式。

1-11　常用逻辑门电路的真值表如下表所示，分析 F_1、F_2、F_3、F_4 属于哪种常用逻辑运算。

A	B	C	F_1	F_2	F_3	F_4
0	0	0	0	1	0	1
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	1	0	1	1
1	0	1	0	1	1	1
1	1	0	0	1	1	1
1	1	1	1	0	1	0

1-12　利用代数法化简下列逻辑函数，必须写出化简过程。

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

（2）$Y = ABC + ABD + A\overline{C}D + \overline{CD} + A\overline{B}C + \overline{A}C\overline{D}$

（3）$Y = ABC\overline{D} + ABD + BC\overline{D} + ABCBD + B\overline{C}$

（4）$Y = AB + \overline{A}C + BC + BCDE$

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

1-13　用卡诺图化简下列逻辑函数，必须写出化简过程。

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

（2）$F(A, B, C, D) = \sum m(6, 7, 8, 9, 12, 13)$

（3）$F(A, B, C, D) = \sum m(6, 7, 8, 9, 10, 11, 13, 14, 15)$

（4）$F(A, B, C, D) = \sum m(0, 2, 8, 9, 10, 11) + \sum d(1, 5)$

（5）$F(A, B, C, D) = \sum m(0, 2, 5, 9, 15) + \sum d(6, 7, 8, 10, 12, 13)$

（6）$F(A, B, C, D) = \sum m(0, 13, 14, 15) + \sum d(1, 2, 3, 9, 10, 11)$