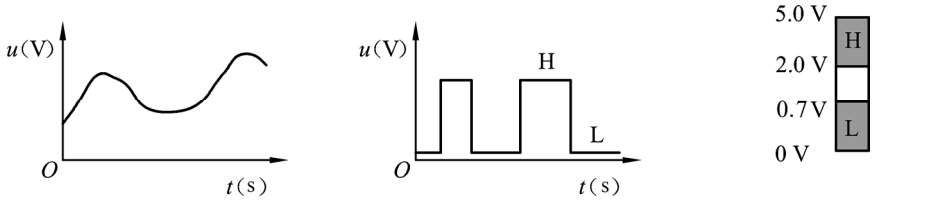
第 1 章　逻辑代数基础与信息表示

逻辑代数是由英国数学家 George Boole 于 1849 年创立的，因此逻辑代数亦叫布尔代数。这是一种建立在二元（抽象为 1 和 0）逻辑基础上的逻辑代数体系，具有与、或、非三种基本运算。逻辑代数广泛地应用于集合论、概率论和数理统计等领域，也是数字电路分析与设计中重要的数学工具。

1.1　概　述

1.1.1　模拟信号与数字信号

随着现代电子技术的发展，数字电子技术的应用越来越广泛，在许多方面取代模拟电子技术。在模拟电子技术中使用的是模拟信号（Analog Signal），如图 1.1.1（a）所示，电压随时间连续变化；在数字电子技术中使用的是数字信号（Digital Signal），如图 1.1.1（b）所示，电压在时间上的变化是不连续的。电压要么处于高电平状态（H），要么处于低电平状态（L）。



（a）模拟信号　　（b）数字信号　　（c）状态取值区间

图 1.1.1　模拟信号与数字信号

所谓的状态并非指某一固定的电压值，而是指一个允许的取值区间，或者说数字信号的高、低电平允许有一定的波动。如图 1.1.1（c）所示。

在数字电子技术中，定义的数字信号有正负逻辑之分，若将高电平状态定义为 1 状态，低电平状态定义为 0 状态，这样的数字信号称为正逻辑信号；若反过来将高电平状态定义为 0 状态，低电平状态定义为 1 状态，这样的数字信号称为负逻辑信号。这里的 1 和 0 是两种状态的抽象表示，没有大小之分。将传输数字信号的电路称为数字逻辑电路，简称数字电路。一般数字电路均采用正逻辑信号，除非特别指明是负逻辑信号。

【说明】电子计算机就是一个复杂的数字电路系统，在其内部处理、传输和存储的信号都是数字信号。32 位机的数据总线（Data BUS）是由 32 根单线并列组成的，每根单线的电平状态 1 和 0 定义为二进制数的 1 和 0，且各单线的权重由高到低依次定义为 2^{31}，2^{30}，…，

2^1，2^0。那么该数据总线就能传输 32 位的二进制数。

1.1.2 进制转换

二进制数学是德国著名数学家莱布尼兹最早提出的，1679 年莱布尼兹发表了论文《二进制算术》。因为二进制数易于电路实现且运算规则简单，例如电平的高低，开关元件的通断，电容器是否带电、人的性别等都可以用数字 1 和 0 表示，所以二进制数成为现代数字系统信息表示的基础。

1. 二进制与十六进制的定义

1）二进制数的定义

$N=k_n2^n+k_{n-1}2^{n-1}+\cdots+k_12^1+k_02^0+k_{-1}2^{-1}+\cdots+k_{-m}2^{-m}$，$k_i\in\{0，1\}$。请读者熟悉以下这些二进制数。

① 2^0，2^1，…，2^{10}，2^{11}，2^{12}，2^{13} 这些十进制数是 1，2，…，1024，2048，4096，8192；

② $2^n=100\cdots0$B，其中 1 后面有 n 个 0，数据之后用 B 表示二进制数；

③ $2^n-1=11\cdots1$B，其中有 n 个 1；

④ $2^{-n}=0.00\cdots01$B，其中小数点之后有 $n-1$ 个 0；

⑤ $1-2^{-n}=0.11\cdots1$B，其中小数点之后有 n 个 1。

2）十六进制数的定义

$N=k_n16^n+k_{n-1}16^{n-1}+\cdots+k_116^1+k_016^0+k_{-1}16^{-1}+\cdots+k_{-m}16^{-m}$，$k_i\in\{0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F\}$。数据之后用 H 表示十六进制数。

例如：AC7H=1010 1100 0111B=$10\times16^2+12\times16+7=2759$

2. 十进制与二进制的互换

1）2 的整幂加减拼凑法

若一个十进制数接近 2^n，采用 2 的整幂加减拼凑法转换为二进制数简明快捷。后面介绍的“除权取商法”和“减权取 1 法”也可结合使用。例如：将十进制数 135 视为 128 (2^7)加 7，则其结果为 1 后面有 7 个 0 的二进制数再加上 111B，所以 135 = 10000111B。将十进制数 2034 视为 2047 ($2^{11}-1$) 减 13，则其结果为有 11 个 1 的二进制数再减去 1101B，所以 2034 = 11111110010B。

2）除权取商法

用十六进制数第 n 位的权重 16^n 去除十进制数，其商为十六进制数第 n 位上的数字；将其余数再用 16^{n-1} 去除，所得商为十六进制数第 $n-1$ 位上的数字；……；重复这样的运算步骤，直到容易看出某一步余数的二进制数为止。最后将每一次的商和最后一步的余数按权重拼成一个二进制数。

【例 1.1.1】将十进制数 78，915，3137 分别化为二进制数。

解：① 因为 78 除以 16 商 4 余 14，所以 78= 100 1110B。

② $915\div16^2=3\cdots\cdots147$ → 11B……147，前者为商，后者为余数，以下同。

147÷16 =9……3 → 1001B……0011B

所以 915 = 11 1001 0011B。

③ 3137÷16^2 =12……65 → 1100B……1000001B，余数 65 的二进制数用口算得到。

所以 3137= 1100 01000001B，注意二进制数 1000001B 前必须添一个 0，使其达到 8 位，因为前段 4 位数字 1100 的权重为 16^2（即 2^8）。

3）减权取 1 法

对于较大的十进制数化为二进制数可采用“减权取 1 法”。该方法是：用十进制数减去小于该数的最大的 2^i，将其差再减去小于此差数的最大的 2^j，…，重复这样的运算步骤，直到容易看出某一步差数的二进制数为止。最后将 2^i、2^j、…，以及最后这一步差数的二进制数按权重拼成一个二进制数。

【例 1.1.2】将十进制数 3145，10000 化为二进制数。

解：

3145	10000
−2048(2^{11})	− 8192 (2^{13})
1097	1808
−1024 (2^{10})	− 1024 (2^{10})
73	784
− 64 (2^6)	− 512 (2^9)
9 (1001B)	272
	− 256 (2^8)
	16 (2^4)
所以 3145 = 110001001001B	所以 10000 = 10011100010000B

4）二进制化为十进制 ——四位分段法

从二进制整数的最低位起，将二进制数视为十六进制数，即每 4 位分为一段，然后按十六进制数的权重展开求和。

【例 1.1.3】将下列二进制数化为十进制数。

解：1101 0110B =13×16 + 6 =214

110101.10111B =110 1011 0111B/2^5 = (6×16^2+11×16+7)/32=1719/32=53.71875

3. 二进制应用举例

1）两个古典数学问题

（1）相传古代印度国王舍汗要褒奖国际象棋发明者达依尔，问他需要什么。达依尔回答说：“国王只要在国际象棋棋盘（8×8 格）的第一格上放 1 粒小麦，第二格上放 2 粒小麦，第三格上放 4 粒小麦，第四格上放 8 粒小麦，按此规律一直放满整个棋盘，我心足矣。”国王居然答应了。

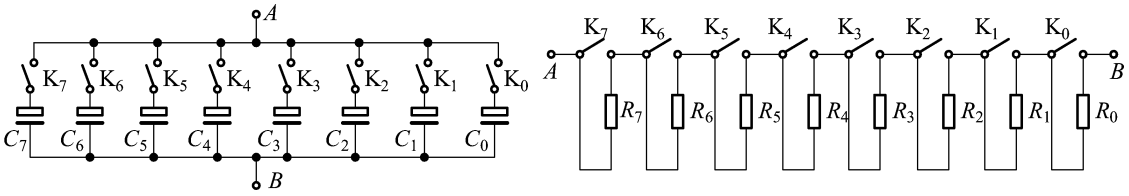
根据二进制数的定义，将棋盘上的小麦数用二进制数表示应为 64 个 1，那么小麦总数为 $2^{64}-1$ 粒。1 g 小麦不足 2^5 粒，1 t 小麦不足 2^{25} 粒。国王舍汗需要支付的小麦超过 $2^{64}/2^{25}=2^{39}$ 吨，这是一个惊人的天文数字啊！

（2）《庄子 · 天下篇》：“一尺之棰，日取其半，万世不竭。”

根据二进制数的定义，将前 n 日所得用二进制数表示之，该数应为小数点之后有 n 个 1，其总和为 $1-2^{-n}$。所以 $2^{-1}+2^{-2}+\cdots+2^{-n}=0.11\cdots1B<1$ 总是成立的。

2）二进制与含权开关量

图 1.1.2（a）是由 8 个开关和 8 个电容器组成的电路，图 1.1.2（b）是由 8 个开关和 8 个电阻组成的电路。其中电容器的取值为 $C_i=2^i\times1\ \mu F$（$i=0$，1，…，7），电阻的取值为 $R_j=2^j\times10\ \Omega$（$j=0$，1，…，7）。这是一种二进制含权电容或电阻电路，在图 1.1.2（a）中，用“1”表示开关 K_i 处于 ON 状态，用“0”表示开关 K_i 处于 OFF 状态；在图 1.1.2（b）中，用“1”表示开关 K_j 处于 OFF 状态，用“0”表示开关 K_j 处于 ON 状态。于是 8 个开关的每种组合状态对应于一个 8 位二进制数，所以电路取值为 0～255 之间的任一整数。



（a）含权电容电路　　（b）含权电阻电路

图 1.1.2　含权电容、电阻电路

【例 1.1.4】将电容 C_{AB} 的值设置为 168 μF，将电阻 R_{AB} 的值设置为 2500 Ω。

解：① 因为 168＝10101000B，所以在图 1.1.2（a）所示的电路中，将开关 K_7，K_5，K_3 闭合，其余 5 个开关断开。

② 因为 250＝11111010B，所以在图 1.1.2（b）所示的电路中，将开关 K_2，K_0 闭合，其余 6 个开关断开。

1.1.3　数字系统中信息的表示

凡是存储、传输或处理数字信号的电路系统简称数字系统，例如 PC 机、单片机、手机等电子设备。在数字系统中一般以字节（1 Byte=8 bit）为基本信息单位，其具体表示的信息是由程序员定义的。

1. 机器码

1）三种定点整数的定义

设机器字长 n+1 位，最高位是符号位，即正数的符号位为 0，负数的符号位为 1。

$$[X]_{原}=X\ (0\le X<2^n)\ /\ 2^n-X\ (-2^n<X\le0)$$

$$[X]_{反}=X\ (0\le X<2^n)\ /\ 2^{n+1}-1+X\ (-2^n<X\le0)$$

$$[X]_{补}=X\ (0\le X<2^n)\ /\ 2^{n+1}+X\ (-2^n\le X<0,\ \mathrm{mod}\ 2^{n+1})$$

2）操作

① 对于原码，不足 n 位的整数在符号位之后补 0；

② 正数的原码、反码和补码相同；

③ 负数的反码为其原码的尾数（除符号位的部分）按位取反；负数的补码为其反码末位加 1。

3）有关性质

① 补码加、减模其值不变；

② 0 的补码是唯一的；

③ $[[X]_{补}]_{补}=[X]_{原}$，$[[X]_{反}]_{反}=[X]_{原}$；

④ n 位定点整数的补码扩展为 $m+n$ 位补码，只需将符号位向高 m 位扩展。

4）n 位无符号数

在计算机中用于表示地址码。

5）机器码的值域

① $n+1$ 位原码与反码的值域：$-(2-1)\sim+(2^n-1)$；

② $n+1$ 位补码的值域：$-2^n\sim+(2^n-1)$；

③ n 位无符号数的值域：$0\sim2^n-1$。

【例 1.1.5】设机器字长为 8 位，求 $X=+57$，$Y=-57$ 的三种机器码。

解：$X=+57=+111001B$　　$Y=-57=-111001B$

$[X]_{原}=\underline{0}0111001B$（最高位为符号位）　　$[Y]_{原}=\underline{1}0111001B$

$[X]_{反}=\underline{0}0111001B$　　$[Y]_{反}=\underline{1}1000110B$

$[X]_{补}=\underline{0}0111001B$　　$[X]_{补}=\underline{1}1000111B$

2. 十进制数

十进制数的每一位由 0～9 十个数字组成，在数字系统中，1 位十进制数要用 4 位二进制数表示，即用二进制数对十进制数进行编码，这样的代码简称 BCD（Binary Coded Decimal）码。表 1.1.1 是几种常用的 BCD 码，其中 8421BCD 码是最常用的一种十进制编码。

表 1.1.1　常用 BCD 码

十进制数	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100
编码规则	权 8421	权 2421	8421 码＋$(11)_2$

4 位二进制数共有 16 个代码，除了对十进制数字 0～9 的编码外，还有 6 个伪码，不同的 BCD 码伴有不同的伪码。在计算机中，若以 BCD 码对十进制数进行运算，一旦结果产生伪码就要对其进行修正处理。例如，当用 8421BCD 码表示的两个十进制数进行相加，若某

位的和出现伪码（即大于 9）或者该位向高位产生了进位，则该位的和还要加 6 进行修正；当用 8421BCD 码表示的两个十进制数进行相减，若某位向高位产生了借位，则该位的差还要减 6 进行修正。

3. 字符

（1）西文（半角）字符的编码：ASCII 码（7 位）。存储在计算机中占 1 个字节，其标识位（即每个字节的最高位）为 0。

（2）汉字（全角）字符的编码：国标码（双 7 位）。存储在计算机中占 2 个字节，其标识位（即每个字节的最高位）为 1。

【例 1.1.6】已知存储信息 X=38H，Y=98H。求不同代码所对应的十进制真值。

解：X、Y 的原码、反码、补码、无符号数、8421BCD 码、ACSII 码如下表所示。

	原码	反码	补码（有符号数）	无符号数	8421BCD 码	ASCII 码
X=38H	+56	+56	+56	56	38	字符‘8’
Y=98H	－24	－103	－104	152	98	无定义

【说明】在数字系统中任何信息都是用二进制代码表示的，所以在数字系统中信息是可以量化的。通常用 b 表示位信息，B 表示字节信息。1 KB=2^{10} B，1 MB=2^{20} B，1 GB=2^{30} B，1 TB=2^{40} B。

1.2 逻辑代数中的基本运算及公式

1.2.1 逻辑函数

图 1.2.1 所示为某数字逻辑电路，输入信号 A_1，A_2，…，A_n 叫作逻辑变量，输出信号 Y 叫作逻辑函数，它们的取值为逻辑值 1 或 0。显然输出信号的变化是受输入信号的影响，或者说输出信号是关于输入信号的函数，即存在逻辑函数式

$$Y=f(A_1，A_2，\cdots，A_n) \qquad (1.2.1)$$

逻辑函数还可用逻辑真值表表示，见表 1.2.1。真值表的一行称为一个状态行，该行的内容是若干个变量的一组逻辑值和由此决定的函数值，n 个变量的逻辑真值表共有 2^n 个状态行。逻辑函数式和逻辑真值表可以相互转换。

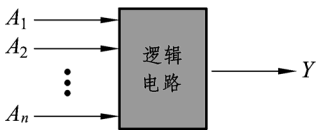


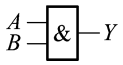
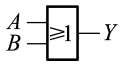
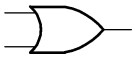
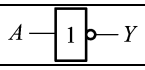
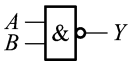
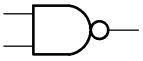
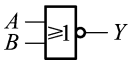
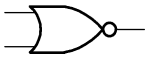
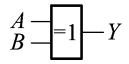
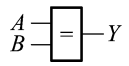
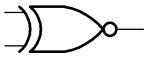
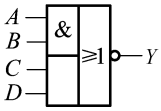
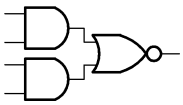
图 1.2.1 逻辑函数与变量的关系

1.2.2 基本运算及公式

在逻辑代数中，与、或、非是三种基本逻辑运算，其他逻辑运算都是这三种基本运算的复合。对两个二进制数实施逻辑运算不同于算术运算，其特点是按位运算，即不产生进借位。

表 1.2.1 是逻辑代数中的常见运算，其中列出了每种运算所对应的电路符号，即门电路。“ 何谓门，门者开关也 ”，所以早期文献将门电路直接叫作开关电路。表 1.2.2 是逻辑代数的基本公式，表 1.2.3 是逻辑代数的常用公式。

表 1.2.1　常见逻辑运算

逻辑运算	国际逻辑图符	国外流行图符	逻辑函数	逻辑真值表		
				A	B	Y
与			$Y=A\cdot B$	0 0 1 1	0 1 0 1	0 0 0 1
或			$Y=A+B$	0 0 1 1	0 1 0 1	0 1 1 1
非			$Y=\overline{A}$	0 1		1 0
与非			$Y=\overline{A\cdot B}$	0 0 1 1	0 1 0 1	1 1 1 0
或非			$Y=\overline{A+B}$	0 0 1 1	0 1 0 1	1 0 0 0
异或			$Y=A\oplus B$ $=\overline{A}B+A\overline{B}$	0 0 1 1	0 1 0 1	0 1 1 0
同或			$Y=A\odot B$ $=\overline{A}\overline{B}+AB$	0 0 1 1	0 1 0 1	1 0 0 1
与或非			$Y=\overline{AB+CD}$	略		

1. 与运算

与运算又称逻辑乘，类似于算术乘法，只是 $A\cdot A=A$。由表 1.2.2 的第 1、2 行左边可知 “ 信号 0 封锁与门，信号 1 开放与门 ”。这句话的意思是：一旦信号 0 打入与门，该与门的输出即为 0，其他输入信号就不起作用了，相当于这些信号被阻止了；信号 1 打入与门，该与门的输出取决于其他输入信号，相当于这些输入信号顺利地通过了与门。另外，与运算是多值运算，可以多个信号相与，即与门的输入端可以是多个变量。

2. 或运算

或运算又称逻辑加，类似于算术加法，只是 1+1=1，$A+A=A$。由表 1.2.2 的第 1、2 行右边可知 “ 信号 1 封锁或门，信号 0 开放或门 ”。或运算也是多值运算。