# 数字系统逻辑设计

曲凤英　丁韵苓　　编著

北京邮电学院出版社

# 数字系统逻辑设计

丁韵苓　曲凤英　编著

北京邮电学院出版社

## 内 容 提 要

本书内容以中、小规模集成电路的逻辑分析和设计为主，同时介绍了大规模集成电路在逻辑设计中的应用，以及常用的各种逻辑部件及数模与模数转换，最后介绍了数字系统的一般设计方法。本书主要对象是工科院校电子类、计算机类和自控类有关专业的本科生，也可作为有关专业的工程技术人员的学习参考书。

数字系统逻辑设计

编　著　丁玲　曲凤英

责任编辑　王平

北京邮电学院出版社出版

新华书店北京发行所发行　各地新华书店经售

北京通县建新印刷厂印刷

*

787×1092毫米　1/16　印张 16.625 字数 411.8 千字

1988年 12 月第一版　1988年 12 月第一次印刷

印数：1—5000册

ISBN 7-5635-0015-4/TN·3　定价：3.35元

# 前　言

本书是编者在多年教学实践的基础上进行整理编写而成的。考虑到目前高等工科院校低年级的实际教学内容，本书没有把集合论作为数学工具，而是从实际应用出发，以逻辑代数作为数学基础，重点讨论了数字系统逻辑设计的基本概念、基本理论和基本方法。但考虑到数技术的迅速发展，目前中、大规模集成电路的应用已成为数字系统的主体，因此本书也以相当多的篇幅介绍常用的中规模集成电路及应用，因考虑到有关大规模集成电路的某些内容将在后续课程《计算机原理及其应用》中还要介绍，这里仅介绍了与逻辑设计有关的只读存贮器(ROM)和可编程序逻辑阵列(PLA)。

全书共分八章。第一、二两章作为逻辑设计的理论基础，介绍在数字系统中常用的几种数制、BCD 编码、逻辑代数及逻辑函数。第三章介绍组合逻辑电路，侧重组合电路的分析和设计方法，在介绍小规模集成电路传统设计方法的同时，也介绍了用中、大规模集成电路进行逻辑设计的特点和一般方法。在第四章中引出了时序电路的概念，介绍了作为时序电路的基本元件——触发器的逻辑功能及触发方式。第五章是时序电路的分析和设计，这里不仅介绍了作为典型时序电路的寄存器、计数器、序列信号发生器及脉冲分配器的分析和设计，而且还介绍了同步时序电路的一般分析和设计方法。第六章介绍以上几章中所使用的器件的内部结构和外部特性，这样，使初学者在此之前可以集中精力掌握逻辑分析和设计的方法。考虑到在实际应用中，合理地选择器件是非常重要的环节，为此，在这一章中介绍目前广泛使用的几类器件的结构和特点。第七章主要介绍数模及模数转换的基本概念、转换原理及基本转换方法。第八章是数字系统设计，这里以寄存器传输语言为工具，仅对同步数字系统进行逻辑设计的方法和步骤进行了探讨，并通过一定的例子加以说明。

本书主要章节后面均附有一定数量的习题，这将有助于读者对课程内容的理解和补充。

本书的第四、五章及第六章的五、六、七节由丁韵苓副教授编写，其余部分均由曲风英副教授编写。陈玉华同志参加了本书的绘图工作。由于编者水平有限，编写时间仓促，书中难免有欠妥或错误之处，恳请读者批评指正。

编　者

# 目　录

# 第一章　数制与BCD码

## 第一节　几种常用的数制及其相互转换

### 一、十进制和二进制

数制是以计数符号的个数(称为基数)来命名的。

日常生活中用的最多的数制是十进制。在十进制数制中，有0～9十个数码，即基数为10. 数码处于不同数位时，所表示的位值是不同的，位值又称权值或位权值。十进制数各位的位权值是以基数10为底的连续整数幂。例如，第i位的位权值为$10^i$，相邻位的位权值相差十倍，并从右到左递增。因此，对于任一个十进制数$N$按其位权值展开均可表示为：

$$\begin{aligned}(N)_{10}&=a_{n-1}\times 10^{n-1}+a_{n-2}\times 10^{n-2}+\cdots+a_1\times 10^1+a_0\times 10^0\\&\quad+a_{-1}\times 10^{-1}+a_{-2}\times 10^{-2}+\cdots+a_{-m}\times 10^{-m}\\&=\sum_{i=-m}^{n-1}a_i\times 10^i\end{aligned}\qquad(1\text{-}1\text{-}1)$$

式中$a_i$为0～9中任一个数码，$n$和$m$为正整数，$n$为整数部分的位数，$m$为小数部分的位数。

当用十进制计数时，执行“逢十进一”及“借一当十”的原则。

通常对十进制数的表示，可以在数字的右下角标注10（或$D$.或省略角标），如$(48)_{10}$，$(48)_D$或48。

由于二进制计数制简单，容易采用数字电路实现数字系统并且工作可靠，故目前数字系统中普遍采用二进制计数制。在二进制中，只有两个数码0和1，故基数为2。用二进制计数时执行“逢二进一”和“借一当二”的原则。不同数位的数码表示的值不同。各位的位权值是以2为底的连续整数幂。因此，任一个二进制数$N$按其位权值展开，可以表示为：

$$\begin{aligned}(N)_2&=a_{n-1}\times 2^{n-1}+a_{n-2}\times 2^{n-2}+\cdots+a_1\times 2^1+a_0\times 2^0\\&\quad+a_{-1}\times 2^{-1}+a_{-2}\times 2^{-2}+\cdots+a_{-m}\times 2^{-m}\\&=\sum_{i=-m}^{n-1}a_i\times 2^i\end{aligned}\qquad(1\text{-}1\text{-}2)$$

式（1-1-2）中$a_i$只能是数码0或1，$n$和$m$为正整数，$n$为整数部分的位数，$m$为小数部分的位数，$2^i$为第$i$位的位权值，相邻位的位权值相差2倍，并从右向左递增。

通常对二进制数的表示，可以在数字右下角标注2，如$(1101)_2=(13)_{10}$。

### 二、八进制和十六进制

用二进制表示一个数，所用的位数要比用十进制用的位数多很多，使人们读、写二进制数感到很不方便。为解决此矛盾，常采用八进制或十六进制数。

八进制数的基数是8，采用的数码是0、1、2、3、4、5、6、7。计数原则是“逢八进一”

和“借一当八”。八进制数按位权值展开的表达式为：

$$(N)_8=\sum_{i=-m}^{n-1}a_i\times 8^i \qquad (1\text{-}1\text{-}3)$$

由于八进制的数码和十进制前八个数码相同，为了便于区分，我们规定，凡是标有下标8的数为八进制数。如$(15)_8=(13)_{10}$

十六进制数的基数为16，采用的数码是0、1、2、3、4、5、6、7、8、9、*A*、*B*、*C*、*D*、*E*、*F*。符号*A*～*F*分别表示十进制数的10～15。凡是用十六进制表示的数，规定下标标注16或*H* (*hexadecimal*)。

十六进制数的计数原则是“逢十六进一”和“借一当十六”。按位权值展开的表达式为：

$$(N)_H=\sum_{i=-m}^{n-1}a_i\times 16^i \qquad (1\text{-}1\text{-}4)$$

表1-1-1列出了在十进制数0～16之间与上述数制的对应关系。

表 1-1-1

| 十进制 | 二进制 | 八进制 | 十六进制 |
|---|---|---|---|
| 0 | 0000 | 0 | 0 |
| 1 | 0001 | 1 | 1 |
| 2 | 0010 | 2 | 2 |
| 3 | 0011 | 3 | 3 |
| 4 | 0100 | 4 | 4 |
| 5 | 0101 | 5 | 5 |
| 6 | 0110 | 6 | 6 |
| 7 | 0111 | 7 | 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 |

## 三、十进制数和二进制数的转换

十进制数是我们所熟悉的，而二进制数又是数字系统中所使用的，因此经常需要在两者之间进行转换。

1、二进制数转换成十进制数

二进制转换成十进制的方法很多，其中最常用的方法有两种，一是按权展开求和法，二是连乘除法。下面分别进行介绍。

(1) 按位权值展开求和法

这种方法是将二进制表示的数按（1-1-2）式展开，求和，其结果即为十进制所表示的相应数。

**例 1** 将$(101001)_2$转换成十进制数。

**解** $(101001)_2=1\times 2^5+1\times 2^3+1\times 2^0$

$=32+8+1=(41)_{10}$

**例 2** 将$(11010.011)_2$转换成十进制数。

**解** $(11010.011)_2=1\times 2^4+1\times 2^3+1\times 2^1+1\times 2^{-2}+1\times 2^{-3}$

$=16+8+2+0.25+0.125$

$=(26.375)_{10}$

(2) 连乘除法

二进制数的按位权展开式也可以改写成如下连乘连除的形式：

$$(N)_2=\{[(a_{n-1}\times 2+a_{n-2})\times 2+\cdots]\times 2+a_1\}\times 2+a_0$$
$$+\{[(a_{-m}\times 2^{-1}+a_{-(m-1)})\times 2^{-1}+\cdots]\times 2^{-1}+a_{-1}\}\times 2^{-1} \qquad (1\text{-}1\text{-}5)$$

由(1-1-5)式可以看出，用连乘除法把二进制数转换成十进制数时，其整数和小数部

分的转换方法不完全相同。

若要把二进制小数部分转换成十进制小数时，是从最低位（LSB）开始，将最低位小数除以2，而后将所得结果同次低位小数相加再除以2，这样重复地做下去，直到加上小数部分的最高位（MSB）后，再除以2，就得到变换后的十进制小数部分。整数部分的转换是从最高位（MSB）开始，将最高位数乘以2，而后将所得结果与次高位数相加，这样重复地做下去，直到加上最低位数为止，即得到变换后的十进制整数部分。转换时，可以将整数部分和小数部分分别转换成十进制数，然后将这两部分结合起来。

**例 3** 用连乘除法将二进制数 $(1101101.1011)_2$ 转换成十进制数。

**解** 分别转换二进制数的整数部分和小数部分，然后把两部分加起来。

整数部分：从最高位开始　　1(MSB)

乘2加下一位，　$1\times2+1=3$　　1

所得结果乘2，　$3\times2+0=6$　　0

再加下一位。　$6\times2+1=13$　　1

重复上述操作　$13\times2+1=27$　　1

直至最低位。　$27\times2+0=54$　　0

$54\times2+1=109$　　1(LSB)

即整数部分 $(1101101)_2=(109)_{10}$

小数部分：从最低位开始，　$1\times\frac{1}{2}=0.5$　　1(LSB)

除以2(乘$\frac{1}{2}$)，　$(0.5+1)\times\frac{1}{2}=0.75$　　1

将所得结果加　$(0.75+0)\times\frac{1}{2}=0.375$　　0

上一位除以2。　$(0.375+1)\times\frac{1}{2}=0.6875$　　1(MSB)

即小数部分 $(0.1011)_2=(0.6875)_{10}$

故　　$(1101101.1011)_2=(109.6875)_{10}$

2、十进制数转换成二进制数

十进制数转换成二进制数最常用的方法是用基数乘除法。按这种方法转换，也是把十进制数的整数部分和小数部分分别进行转换，然后将结果相加。

整数部分采用"除2取余"法转换。即把十进制整数除以2，取出余数1或0作为相应二进制数的最低位（LSB），把得到的商再除以2，再取余数1或0作为二进制数的次低位，依次类推，继续上面的过程，直至商为0时，所得余数为最高位（MSB）。

**例 4** 把十进制数 $(53)_{10}$ 转换成二进制数。

```
2 | 5 3
2 | 2 6 ····· 1   (LSB)
2 | 1 3 ····· 0     ↑
  2 | 6 ····· 1     |
  2 | 3 ····· 0     |
  2 | 1 ····· 1     |
    0 ····· 1   (MSB)
```

由此得到：$(53)_{10}=(110101)_2$

小数部分可采用"乘2取整"法。即先将十进制小数乘以2，取其整数部分1或0，作为二进制小数的最高位；而后将前一步结果的小数部分再乘以2，再取整数，作为次高位。重复以上过程，直到小数部分为0或已达到所要求的精度为止。

例 5　把 $(0.375)_{10}$转换成二进制数。

解

| | 整数部分 | |
|---|---|---|
| $0.375\times2=0.750$ | 0 | MSB |
| $0.750\times2=1.500$ | 1 | ↓ |
| $0.500\times2=1.000$ | 1 | LSB |

其结果为：$(0.375)_{10}=(0.011)_2$

### 四、八进制、十六进制数与二进制数的转换

由于八进制数的基数是8($8=2^3$)，十六进制数的基数为16($16=2^4$)，因此，将二进制数转换成八进制（或十六进制）数是相当方便的。其转换方法为：从小数点开始，分别向左、右按三位（转换成八进制）或四位（转换成十六进制）分组，最后不满三位或四位的加0补位。将每组以对应的八进制数或十六进制数代替之，即为等值的八进制数或十六进制数。

例 6　将$(11110110111.1011)_2$分别转换成八进制数和十六进制数。

解　转换成八进制数：按三位分组，不足三位数以0补位。

$$\begin{array}{ccccccccc} 011 & \vdots & 110 & \vdots & 110 & \vdots & 111.101 & \vdots & 100 \\ 3 & & 6 & & 6 & & 7.5 & & 4 \end{array}$$

即　$(11110110111.1011)_2=(3667.54)_8$

转换成十六进制数，按四位分组

$$\begin{array}{ccccc} 0111 & \vdots & 1011 & \vdots & 0111.1011 \\ 7 & & B & & 7\quad .B \end{array}$$

即　$(11110110111.1011)_2=(7B7.B)_H$

若需要把八进制数或十六进制数转换成二进制数时，只要按上述方法的逆过程即可。

至于八进制、十六进制和十进制之间的相互转换，也可用按位权值展开求和法或连乘除法及基数乘除法，只是，对八进制数基数是8，十六进制数基数是16。

## 第二节　BCD 码

数字系统处理的是二进制数码，而人们习惯于十进制数码，所以在数字系统的输入（又称写入）或输出（又称读出）时，仍常用十进制数。这就需要将十进制数转换成数字系统所能接收的二进制码。除了上节所述的转换方法之外，还有一种常用的方法，即用四位二进制数码表示一位十进制数，这就是二进制编码的十进制数，简称二-十进制码（*Binary Coded Decimal Codes*），即BCD码。

BCD码具有二进制码的形式（四位二进制码）和十进制的特点（基数为10），即它属于十进制，但数字是用二进制码表示的。四位二进制代码可以组成16（$2^4=16$）种不同的状态，而十进制数只需要十种状态分别用来表示0～9十个数码，若用四位二进码表示十进制数

时，就有六种状态不用。所以，根据数码所选状态的不同，可以有多种表示方法。表1-2-1列出了几种常用的BCD码及其对应的十进制数。

表 1-2-1

| 十进制数码 \ BCD码 | 有权码 | | | 无权码 | | |
|---|---|---|---|---|---|---|
| | 8421码 | 2421码 | 5121码 | 余3码 | 余3循环码 | 移存码 |
| 0 | 0000 | 0000 | 0000 | 0011 | 0010 | 0001 |
| 1 | 0001 | 0001 | 0001 | 0100 | 0110 | 0010 |
| 2 | 0010 | 0010 | 0010 | 0101 | 0111 | 0100 |
| 3 | 0011 | 0011 | 0011 | 0110 | 0101 | 1001 |
| 4 | 0100 | 0100 | 0111 | 0111 | 0100 | 0011 |
| 5 | 0101 | 1011 | 1000 | 1000 | 1100 | 0111 |
| 6 | 0110 | 1100 | 1001 | 1001 | 1101 | 1111 |
| 7 | 0111 | 1101 | 1010 | 1010 | 1111 | 1110 |
| 8 | 1000 | 1110 | 1011 | 1011 | 1110 | 1100 |
| 9 | 1001 | 1111 | 1111 | 1100 | 1010 | 1000 |

BCD码可以分为有权码和无权码。所谓有权码即是每一位都有固定位权值的码。有权码中用得最多的是8421BCD码，该码的位权值从左至右分别为8、4、2、1。显然，它和普通的四位二进制码相应位的位权值一样，但是在8421BCD码中，不允许出现1010～1111六种状态。

8421BCD码与十进制数之间的对应关系是按码组对应，即一个$n$位的十进制数，需要$n$组8421BCD码表示；反之，$n$组8421BCD码表示$n$位十进制数。例如8421BCD码中的1001对应十进制数9，0101对应十进制数5，故十进制数59的对应8421BCD码为01011001，即

$$(59)_{10}=(01011001)_{8421BCD}$$

反之，如果将$(1001001010000111)_{8421BCD}$转换成十进制数时，则从最低位开始，按四位分组；然后写出每组8421BCD码对应的十进制数。即

$$(1001001010000111)_{8421BCD}$$
$$=1001,0010,1000,0111=(9287)_{10}$$
$$9\quad 2\quad 8\quad 7$$

由表1-2-1可知，在8421BCD码中，凡是十进制数为奇数（1、3、5、7、9）时，它的二进制编码的最低位都是1；凡是十进制数为偶数（0、2、4、6、8）时，它的二进制编码的最低位都是0，故8421BCD码具有奇偶特性。

2421BCD码也是一种有权码，该码从左到右的位权值分别为2、4、2、1。2421BCD码有两种形式，表1-2-1中仅列出其中的一种。在这种编码中，十进制数0和9、1和8、2和7、3和6、4和5的对应位码一个是0时，另一个就是1，即互为反码。具有这种特性的代码称之为自补代码（*Self Complementing code*）。

BCD码可以直接参与十进制运算，在十进制加、减运算中，常需要求十进制数对9之补，即求9与该数之差（$9-d_i$），例如3对9之补是9−3＝6；2对9之补是9−2＝7，0对9之补是9−0＝9等。用2421BCD码能很方便地求得某数对9之补。即把该数的2421BCD码自身按位取反，就得到该数对9之补的2421BCD码。例如，十进制数3的2421BCD码为0011，3对9之补是6，6的2421BCD码为1100（即把0011自身按位取反）。

无权码是没有固定位权值的码。如表1-2-1中的余3码。这种代码比相应的8421BCD码多3（$3_{10}=0011_2$），故称余3码。如十进制数6的8421BCD码是0110，而余3码则应是0110＋0011＝1001。余3码也是一种自补代码，故采用余3码进行十进制加、减运算时比8421BCD码方便。

表1-2-1中还列出了余3格雷码。格雷码（Gray）又称循环码，有多种形式，但有一个共同的特点，就是任意两个相邻的数码之间，仅有一位二进制数码不同，其余各位数码均相同（包括首尾两个数码0和$2^n-1$在内）。这个特点在应用中，有实际意义。首先让我们看看没有这种特点代码存在的问题，例如两个相邻的十进制数5和6，它们的二进制代码分别为0101和0110，相互之间有两位码不同，当用二进制进行加法计数时，例如从5计到6，二进制码0101的最低两位都要改变。如果这两位码的改变不是同时发生的，那么，在计数过程中就可能出现瞬时其他代码，如0111（第二位先变化）或0100（第一位先变化），这种误码出现时间虽然短暂，有时却是不允许的。而格雷码就避免了这种错误，所以格雷码是一种可靠性代码。格雷码的值不能由码组的位权值决定，故格雷码也是一种无权码。

两位和三位格雷码分别如表1-2-2和表1-2-3所示。

表 1-2-2

| 十进制数 | $G_2$ $G_1$ |
|---|---|
| 0 | 0 0 |
| 1 | 0 1 |
| 2 | 1 1 |
| 3 | 1 0 |

表 1-2-3

| 十进制数 | $G_3$ $G_2$ $G_1$ |
|---|---|
| 0 | 0 0 0 |
| 1 | 0 0 1 |
| 2 | 0 1 1 |
| 3 | 0 1 0 |
| 4 | 1 1 0 |
| 5 | 1 1 1 |
| 6 | 1 0 1 |
| 7 | 1 0 0 |

由表1-2-2和表1-2-3可知，格雷码具有反射性，即若以最高位的0和1的交界处为轴（如表中的虚线），除了最高位，其他位的代码是上下以“轴”为对称的。利用这一特性可以很方便地由$n$位格雷码产生（$n+1$）位格雷码。

在用格雷码表示十进制数时（BCD码），为使0和9的代码也只有一位码不同，常采用余3格雷码。

## 习　　题

1、试将下列不同进制数表示为按位权值展开的和式。

(1) $(1754.329)_{10}$　　(2) $(11010.0101)_2$

(3) $(532.447)_8$　　(4) $(5A7.3ED)_{16}$

2、试将下列二进制数分别转换成等值的十进制数、八进制数和十六进制数。

(1) $(111010)_2$　　(2) $(10011.101)_2$

(3) $(0.110101)_2$

3、试将下列十进制数分别转换成等值的二进制数（取小数点后六位）、八进制数和十六进制数（取小数点后三位）。

(1) $(37)_{10}$　　(2) $(0.756)_{10}$

(3) $(42.186)_{10}$

4、试将下列十进制数用8421BCD码表示。

(1) $(15)_{10}$　　(2) $(0.207)_{10}$

(3) $(728.0123)_{10}$

5、将下列8421BCD码分别转换成等值的十进制数和二进制数。

(1) 101011100011　　(2) 000110110000011

(3) 01010110.10100100

6、试用余3码及余3格雷码分别表示下列各数。

(1) $(78)_{10}$　　(2) $(456)_{10}$

(3) $(1100110)_2$　　(4) $(1111001)_2$

7、完成下列二进制表达式的运算。

(1) 1101+0111　　(2) 10111+110.011

(3) 11001−10100　　(4) 1000−11.011

(5) 1001×101　　(6) 10.01×1.01

(7) 10110111÷111　　(8) 1001.0001÷11.101

# 第二章　逻辑代数及逻辑函数

逻辑代数（又称开关代数）是一种较为具体的布尔代数。

布尔代数是英国数学家乔治·布尔（*George Boole*，1815～1864）在十九世纪中叶首先提出的。他在《逻辑的数学分析》（1847年）和《思维规律》（1854年）中首先提出，所以叫布尔代数，但布尔代数本身是研究人类思维规律的。此后于1938年，克劳德·向农（*Claude E·Shannon*）把布尔代数应用到电话继电器开关电路的设计中，故应用于开关电路的布尔代数通常称为逻辑代数或开关代数。逻辑代数是研究数字电路的数学工具，并成为开关理论和逻辑设计的数学基础。

逻辑代数和普通代数相同之点是，它的变量也可以用英文字母*A*、*B*、*C*…*X*、*Y*、*Z*等表示，称之为逻辑变量，但逻辑变量只有两种取值1或0。这里的逻辑值0和1。不表示数值，通常只表示两种对立的状态，如开关的接通和断开；脉冲的有和无；命题的正确与不正确等。在逻辑代数中，对逻辑变量的运算不同于普通代数。它有三种最基本的运算—与、或、非，其运算规则是按逻辑规则定义的。

## 第一节　基本逻辑运算

### 一、与逻辑运算

所谓与逻辑运算，就是仅当决定一事件发生的所有条件均具备时，事件才发生。这种因果关系称之为与逻辑，与逻辑又叫逻辑乘，如图2-1-1所示，开关*A*和*B*串联控制灯*F*，可以看出，只有当开关*A*和*B*均合上时，灯*F*才亮，否则灯*F*不亮。故这是一种与逻辑关系。

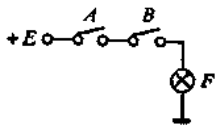


图2-1-1　与逻辑举例

作为开关，只有两种状态，合上或断开，我们用表2-1-1来描述图2-1-1电路中灯*F*与开关*A*、*B*的各种可能组合下的因果关系。

为了描述的方便，我们把开关合上定义为1，开关断开定义为0，灯亮定义为1，灯灭定义为0，那么表2-1-1就转变成表2-1-2所示的形式了。通常把表2-1-2称为真值表。

表2-1-1

| A | B | F |
|---|---|---|
| 断 | 断 | 灭 |
| 断 | 合 | 灭 |
| 合 | 断 | 灭 |
| 合 | 合 | 亮 |

表2-1-2

| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

真值表是从命题过渡到逻辑代数的有力工具，它记录了在输入逻辑变量的所有可能取值

组合情况下，对应逻辑函数的取值。显然，对于具有$n$个输入变量的逻辑函数，其真值表应有$2^n$个函数值。

**与**逻辑运算的规则为：

$$0\cdot 0=0$$
$$0\cdot 1=0$$
$$1\cdot 0=0$$
$$1\cdot 1=1$$

因此**与**逻辑运算可以写成数学表达式：

$$F=A\cdot B \quad 或 \quad F=AB \tag{2-1-1}$$

式（2-1-1）又称为**与**逻辑函数表达式。在上面的例子中虽然仅定义了两个变量，但**与**运算的概念可以扩大到任意多个变量。

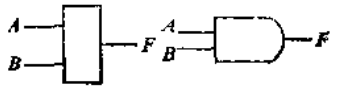

图 2-1-2　与门逻辑符号

在**数字电路**中，完成**与**运算的电路称为**与门**，**与门**有多个输入端和一个输出端，**与门**的逻辑符号如图2-1-2所示。

**与门**的逻辑功能可以概括为“只有全部输入均为1时，输出才为1”。

## 二、或逻辑运算

所谓**或**逻辑运算就是当决定一事件发生的各种条件中，只要有一个或一个以上条件具备时，事件便可以发生。这种因果关系称之为**或**逻辑，**或**逻辑又叫逻辑加。如图 2-1-3 所示，开关$A$和$B$并联控制灯$F$。可以看出，开关$A$和$B$中，只要有一个（或二个）闭合，灯 $F$ 即亮，故这是**或**逻辑关系。**或**逻辑可以表示为$F=A+B$，式中“+”读作**或**。

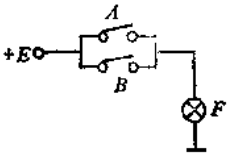

图 2-1-3　或逻辑举例

图2-1-3所示电路中，开关的全部可能状态及产生的结果如表2-1-3所示。表 2-1-4 是对应的**真值表**。

表 2-1-3

| A | B | F |
|---|---|---|
| 断 | 断 | 灭 |
| 断 | 合 | 亮 |
| 合 | 断 | 亮 |
| 合 | 合 | 亮 |

表 2-1-4

| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**或**逻辑运算的规则为：

$$0+0=0$$
$$0+1=1$$

$$1+0=1$$
$$1+1=1$$

值得注意的是，这里 1＋1＝1，它表示逻辑运算，不是数值运算。

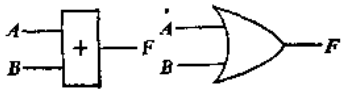


图 2-1-4　或门逻辑符号

完成或逻辑功能的电路叫**或门**，**或门**有多个输入端和一个输出端。**或门**的逻辑符号如图2-1-4所示。

**或门**的逻辑功能也可以概括为“只要输入有 1，输出即为1”。

## 三、非逻辑运算

所谓**非逻辑**反映两个互相矛盾的命题的判断，某事件$F$是否成立，和逻辑变量$A$相关联，若$A$**成立**，则$F$不成立；若$A$不成立，则$F$成立。我们记作$F=\bar{A}$．式中“—”表示逻辑**非**的运算符号，读作$F$等于$A$**非**。

如图2-1-5所示，当开关$A$合上，灯不亮，而当开关$A$断开时，灯反而亮。这是一种**非**逻辑关系。其真值表如表2-1-5所示。

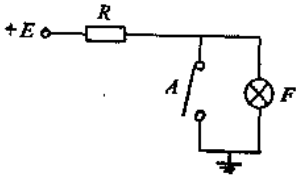


图 2-1-5　非逻辑举例

表 2-1-5

| A | F |
|---|---|
| 0 | 1 |
| 1 | 0 |

**非**逻辑又称为**反**逻辑或**补**逻辑。由真值表知：

$$\bar{0}=1$$
$$\bar{1}=0$$

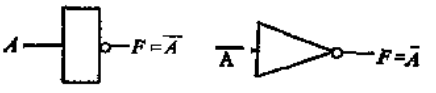


图 2-1-6　非门逻辑符号

实现**非**逻辑功能的电路称为**非门**．**非门**又称为**反相器**。**非门**的逻辑符号如图 2-1-6 所示。

## 四、正负逻辑

在逻辑电路中，常把电平的高低、信号的有无等赋予一定的逻辑值1或0，从而可以列出该逻辑电路的真值表，并得到一定的逻辑关系。

如果把高电平、有信号赋值为1；把低电平、无信号赋值为0，在这种赋值条件下，得到的逻辑关系通常称为正逻辑。反之，若把低电平、无信号赋值为1；把高电平，有信号赋值为0，在这种赋值条件下，得到的逻辑关系通常称为负逻辑。要强调的是电平的高低、信号的有无等是客观存在，而用0或1表示是人为的，电路功能并不会因为用不同定义的逻辑而有所改变，但是对同一个问题，如果采用不同定义的逻辑去分析，会得到不同的逻辑关系。

前面我们曾用图2-1-1为例，以开关合上为条件，灯亮为结果，得到**与**逻辑关系。对同一个电路，如果以开关断开为条件，灯灭为结果，那么可以得到**或**逻辑关系。由此可知**正逻辑与**就等于负逻辑**或**。同理，也可以导出正逻辑**或**等于负逻辑**与**的结论。

对于不同定义的逻辑关系，采用的逻辑符号也不同。图2-1-7列出了三种基本逻辑关系**与、或、非**的正负逻辑符号对应图。

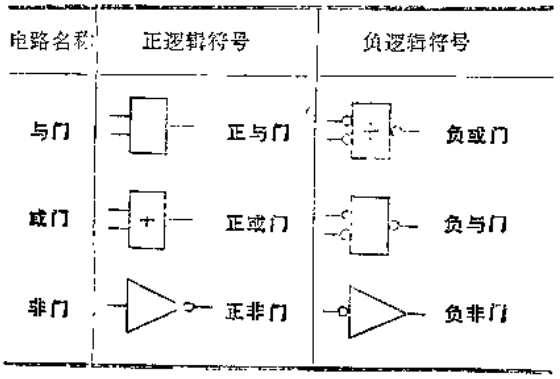


图 2-1-7　正负逻辑符号对应图

在本书中，没有特殊说明时，我们约定均以正逻辑讨论问题。

综上所述，**与、或、非**是多种逻辑运算中最基本的运算。它在形式上有和普通代数

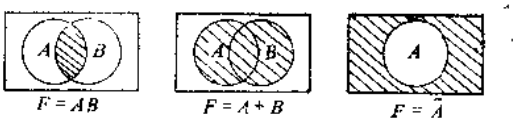


图 2-1-8　与、或、非维恩图

相似之处，但本质是截然不同的，其中运算符号“十”和“·”只表示逻辑运算不代表数值运算。这三种关系也可以用维恩图表示，如图2-1-8所示。

## 第二节　逻辑代数的基本定律和规则

逻辑代数既然是一门完整的代数学，便可以进行运算。那么，在运算中有哪些规则可依据呢？下面，我们根据**与、或、非**三种最基本的运算规则，可以推导出逻辑代数的基本定律，熟悉这些定律，对分析和综合数字电路是非常有用的。

### 一、逻辑函数的相等

逻辑函数和普通代数一样，有个相等的问题，如何判断两个函数相等？

设有两个函数$F_1=f_1(A_1, A_2, \cdots, A_n)$和$F_2=f_2(A_1, A_2, \cdots, A_n)$，如果对于$A_1, A_2, \cdots, A_n$的任何一组取值（共有$2^n$组），$F_1$和$F_2$的值都相等，则我们说函数$F_1$和函数$F_2$相等，记作$F_1=F_2$。换言之，如果$F_1=F_2$，则它们必然有相同的真值表；反之，如果两个函数的真值表相同，那么，这两个函数必然是相等的。

表 2-2-1

| A | B | C | $F_1$ | $F_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**例 1**　设有两个函数

$$F_1=A+BC$$

$$F_2=(A+B)(A+C)$$

求证：$F_1=F_2$。

**解**：这两个函数都具有三个自变量，可以有$2^3=8$组逻辑取值，这八种组合及其对应的函数$F_1$和$F_2$的值分别列在表2-2-1中。由表可见：对应于$A$、$B$、$C$的每一组取值，函数$F_1$的值都等于$F_2$的值，所以$F_1=F_2$。

（在逻辑代数中，**与**运算优先于**或**运算）。