第1章 数字电路与系统基本概念

一、数字电路中的数制

(1)十进制数

有0,1,2，…，9等十个数码元素，任何一个大小的数字都由这十个元素组成。

$575.6=5\times10^2+7\times10^1+5\times10^0+6\times10^{-1}$

10^2 ——百位的“权”

10^1 ——拾位的“权”

10^0 ——个位的“权”

10^{-1} ——拾分之一位的“权”

“权”表示价值

任意进制数的通式:

$$(N)_r = \sum_{i=-m}^{n-1} K_i \cdot (r)^i = K_{n-1} \cdot r^{n-1} + K_{n-2} \cdot r^{n-2} + \cdots + K_2 \cdot r^2$$

$$+ K_1 \cdot r^1 + K_0 \cdot r^0 + K_{-1} \cdot r^{-1} + K_{-2} \cdot r^{-2} + \cdots + K_{-m} \cdot r^{-m}$$

r ——任意进制数的基数

K_i ——某数中第i位的数码元素

n ——该数整数部分的位数

m ——小数部分的位数

(2) **二进制数**(Binary number)

基数r=2，逢二进一，只有0和1二个数码元素

$$(1101.001)_2=1\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0 +0\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}$$

从高位至低位的“位权”依次是：

2^{n-1}、2^{n-2}、…、2^0、2^{-1}、…、2^{-m}。

(3)八进制数(Octal number)

基数r=8，逢八进一

八个数码元素为0、1、…7

$(357.61)_8$或

$(357.61)_O=3\times8^2+5\times8^1+7\times8^0+6\times8^{-1}+1\times8^{-2}$

从高位至低位的“位权”依次是：

8^{n-1}、8^{n-2}、…、8^0、8^{-1}、…、8^{-m}

(4) **十六进制数**(Hexadecimal number)

基数r=16，逢十六进一

十六个数码元素为　0、1、…9、

A、B、C、D、E、F。

$(A8D.C6)_{16}$或

$(A8D.C6)_H = A \times 16^2 + 8 \times 16^1 + D \times 16^0$

$+ C \times 16^{-1} + \ldots$

表：几种常见数制间的关系

十进制 r=10	二进制 r=2	八进制 r=8	十六进制 r=16
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

二、各种进制数间的相互转换

(1) 整数部分

设一个任意十进制整数可用二进制数表示如下：

$$(N)_r=K_{n-1}\times 2^{n-1}+K_{n-2}\times 2^{n-2}+\ldots+K_2\times 2^2+K_1\times 2^1+K_0\times 2^0$$

以十进制数转换成二进制数为例：

LSB—Least Siginificant Bit

$$\begin{aligned}(N)_{10}&=K_{n-1}\times 2^{n-1}+K_{n-2}\times 2^{n-2}+\ldots\\&\quad+K_2\times 2^2+K_1\times 2^1+K_0\times 2^0\\&=2\{K_{n-1}\times 2^{n-2}+K_{n-2}\times 2^{n-3}+\ldots\\&\quad+K_2\times 2^1+K_1\times 2^0\}+K_0\\&=2\{2[K_{n-1}\times 2^{n-3}+K_{n-2}\times 2^{n-3}+\ldots\\&\quad+K_3\times 2^1+K_2\times 2^0]+K_1\}+K_0\\&=2\{2[\ldots 2(0)+K_{n-1}\ldots]+K_1\}+K_0\end{aligned}$$

K_0： LSB
（最低位）

K_{n-1}： MSB
（最位）

MSB—Most Siginificant Bit

整数部分的转换：除r取余

十进制数转换成二进制数：除二取余

十进制数转换成八进制数：除八取余

十进制数转换成十六进制数：除十六取余

- 将待转换的十进制数整数除以进制数（二、八、十六）取余数，不断地进行，直至余数为零

第一次的余数为转换后进制数的最低位（LSB）
最后的余数为转换后进制数的最高位（MSB）

将十进制数175转换成二进制，八进制和十六进制数

解：

2 | 175 …… $K_0=1$	8 | 175 …… $K_0=7$	16 | 175 …… $K_0=F$
2 | 87 …… $K_1=1$	8 | 21 …… $K_1=5$	16 | 10 …… $K_1=A$
2 | 43 …… $K_2=1$	8 | 2 …… $K_2=2$	0
2 | 21 …… $K_3=1$	0	
2 | 10 …… $K_4=0$		
2 | 5 …… $K_5=1$		
2 | 2 …… $K_6=0$		
2 | 1 $K_7=1$		
0		

结果 $(175)_{10}=(10101111)_2=(257)_8=(AF)_{16}$

(2)小数部分

一个任意十进制小数可用二进制数表示如下：

$(N)_r=K_{-1}\times 2^{-1}+K_{-2}\times 2^{-2}+\ldots+K_{-m+1}\times 2^{-m+1}+K_{-m}\times 2^{-m}$

小数部分的转换方法与整数部分转换基本相同，只是不断地将小数部分乘以进制数，并按其积的整数，依次确定K_{-1}，K_{-2}，K_{-3}……，一直进行到积的小数部分为零为止。若积的小数达不到零时，根据转换的精度来取位数。

第一次的整数为转换后进制数的最高位（MSB）

最后一次的整数为最低位（LSB）

例　将十进制数小数$(0.125)_{10}$转换成等值的二进制数、八进制数和十六进制数。

$$\begin{array}{r} 0.125 \\ \times \quad 2 \\ \hline 0.250 \end{array} \qquad \mathbf{K_0=0}$$

$$\begin{array}{r} \times \quad 2 \\ \hline 0.\ 50 \end{array} \qquad \mathbf{K_1=0}$$

$$\begin{array}{r} \times \quad 2 \\ \hline 1.\ 0 \end{array} \qquad \mathbf{K_2=1}$$

$$\begin{array}{r} 0.125 \\ \times \quad 8 \\ \hline 1.0 \end{array} \qquad \mathbf{K_0=1}$$

$$\begin{array}{r} 0.125 \\ \times \quad 16 \\ \hline 2.0 \end{array} \qquad \mathbf{K_0=2}$$

结果$(0.125)_{10}=(0.001)_2=(0.1)_8=(0.2)_{16}$

小数部分的转换：乘r取整

十进制数转换成二进制数：乘二取整

十进制数转换成八进制数：乘八取整

十进制数转换成十六进制数：乘十六取整

(3) 二进制数、八进制数及十六进制数转换成十进制数

“按权展开再相加”

(4) 二进制、八进制以及十六进制数之间的相互转换

用二进制数作为桥梁

一个八进制数码元素用一组三位二进制数表示

一个十六进制数码元素用一组四位二进制数表示

例：

$(0A)_{16}=(1010)_2=(12)_8$

$(22)_8=(010010)_2=(12)_{16}$

第1章 数字电路与系统基本概念

- 1.1 数字信号和数字电路
- 1.2 数字电路中的数制及转换
- **1.3 数字电路中的代码**
- 1.4 数字电路中的基本逻辑函数
- 1.5 逻辑代数

1.3 数字电路中的代码

一、数字系统中的常见代码

用一组十进制数代替一个特定对象的过程称为**编码**

用一组二进制数代表一个特定对象称**二进制数编码**

1、二——十进制编码(**BCD码**) (Binary Coded Decimal)

十进制数的0～9十个数字分别用一个四位的二进制编码表示，称**十进制数的二进制编码**，简称BCD码

四位二进制数有十六种不同组合，只要选出其中的十种分别代替0、1、…、9十个数码进行组合。不同的选取构成不同的编码方式。

十进制数	有权码 8421	5421	2421	2421 *	5211	无权码 余三码
0	0000	0000	0000	0000	0000	0011
1	0001	0001	0001	0001	0001	0100
2	0010	0010	0010	0010	0100	0101
3	0011	0011	0011	0011	0101	0110
4	0100	0100	0100	0100	0111	0111
5	0101	1000	0101	1011	1000	1000
6	0110	1001	0110	1100	1001	1001
7	0111	1010	0111	1101	1100	1010
8	1000	1011	1110	1110	1101	1011
9	1001	1100	1111	1111	1111	1100

表中，有权码下面的8-4-2-1、5-4-2-1、…，分别表示这种代码方案中高位至低位的“权”，即每一位的1代表的十进制数值。

无权码中的某一位代码就没有具体十制数值的意义

注意：编码与数制转换的区别！

如 $(359)_{10}=(001101011001)_{8421}$

$=(001110001100)_{5421}$

$=(001101011111)_{2421}$

$=(010110001111)_{5211}$

$=(011010001100)_{余三码}$

$=(101100111)_2=256+64+32+4+2+1$

2、循环码　又称格莱码(Gray code)

特征是：任何相邻二组代码之间只差一位数码不同，其余相同。

四位循环码

十进制数	4位循环码	十进制数	4位循环码
0	0000	8	1100
1	0001	9	1101
2	0011	A	1111
3	0010	B	1110
4	0110	C	1010
5	0111	D	1011
6	0101	E	1001
7	0100	F	1000

由表可知，这种代码具有下面几个优点：

①具有反射特性。如表中的7和8、6和9、5和10……之间，除了最高位，其余三位由反射得到。

②任何二组相邻代码之间，由于只差一位代码不同，其余相同。因此，用此代码的译码等电路不会产生竞争冒险，工作可靠。

③在信息的交换和传递过程中可以减少差错。

3 字符代码

ISO编码(International Standardization Organization)国际标准组织制定的八位二进制代码，主要用于信息交换，它包括十进制数的十个数码，二十六个英文字母，以及+、-、×、÷、……等二十个符号共56种特定对象。

ASCII码(American Standard Code for Information Interchange)是美国国家信息交换标准代码的简称，也是八位二进制代码，其中一位作奇偶校验位。

二、计算机中数据的存放和传输(正负数的表示)

数字系统中，一个数的最高位前设置一位符号位，并规定，符号位为“0”时，表示该数为正数，符号位为“1”是负数。

这种带符号位的数称为机器数，原正负数又称真值。

一个机器数的表示形式有三种：

原码，反码和补码

(1)原码(True form)

由符号位加原数的数值部分，即

【X】$_{原}$=符号位+原数值

如 x_1=+1001010　则【x_1】$_{原}$=01001010

x_2=-1001010　则【x_2】$_{原}$=11001010

特点：

原码表示简单，直观。在二进制乘法运算中用于决定积的符号也较容易(二个乘数符号位异或)。但减法运算的符号位较难求出。

(2) 反码(One's Complement)

正数的反码为符号位加上原数值部分，负数的反码为符号位加上原数值的反码(原数值按位求反)

【x】$_{反}$=符号位+原数值　(x为正数时)

=符号位+原数值反码　(x为负数时)

例　x_1=+1001010 则【x_1】$_{反}$=01001010

x_2=-1001010 则【x_2】$_{反}$=1 0110101

(3)补码(Two's complement)

【例】钟表校时的情况：

时钟停止在10：00上，要校准到7点，有二种方法：

a. 顺拨时钟9个小时，相当于10+9=12+7

b. 反拨时钟3个小时，相当于和10-3=7

对钟表走一圈为12的最大数而言，顺拨时的10+9和反拨的10-3是相等的。

数学上+9和-3就称为最大数12的互为补数，或称+9是-3对模12的补码(数学上最大数也称模)。

可见：

一个减法运算可以变换成加法求和了

一个n位的二进制数x的补码（不考虑符号）可用下式方法求取。

【x】$_{补}$=模-【x】=2^n-【x】

例如 $(1010)_2=2^4-1010=10000-1010=0110$

补码有两种求法:

（1）反码加1;

（2）从原数值的最低位开始，在遇到1之前(包括该1)原数码不变，其后数码按位求反。

机器数用补码表示时规定：

【x】$_{补}$=符号位+原数值 (x为正数)

=符号位+原数值的补码 (x为负数)

例如 x_1=+1001010的补码是［x_1］$_{补}$=01001010

x_2=-1001010的补码是［x_2］$_{补}$=10110110

补码的运算规则：$【x_1+x_2】_{补}=【x_1】_{补}+【x_2】_{补}$

$【【X】_{补}】_{补}=【X】_{原}$

所以有 $【x_1-x_2】_{补}=【x_1】_{补}+【-x_2】_{补}$

例如，$[1100-1001]_{补码}=[1100]_{补码}+[-1001]_{补码}$

$=01100+10111=100011$

其中，最高位为最大数，自然丢失(溢出)，次高位0为符号位，运算结果为+3。

又如 $【1001-1100】_{补}=【1001】_{补码}+【-1100】_{补码}$

$=01001+10100=11101$

结果是负数，再求补后得10011，所以是-3。

◆ 引入补码以后，在数字电子系统中的减法运算可用加法实现。

◆ 两个补码表示的带符号位的二进制数相加时，其相加结果为和的补码。

◆ 运算结果若符号位有进位产生，则应丢掉此进位，才能得到正确结果。

**两个补码表示的二进制数相加时的符号位讨论

例： 用二进制补码运算求出

12+9、12-9、9-12、-9-12

+12	0 01100	+12	0 01100
+ 9	0 01001	- 9	1 10111
+21	0 0 10101	+ 3	0 00011
+ 9	0 01001	-12	1 10100
- 12	1 10100	- 9	1 10111
- 3	1 11101	- 21	1 1 01011

◆进位位与和数的符号相反时，溢出出错

小数的反码和补码：(指 定点数)

小数反码定义：

【x】$_{反}$＝符号位＋原数值 (x为正数)

＝符号位＋原数值的反码 (x为负数)

小数补码定义：

【x】$_{补}$＝符号位＋原小数部分数值 (x为正数)

＝符号位＋原小数部分的补码 (x为负数)

几个数的真值、原码、反码、补码

x	$[x]_{原}$	$[x]_{反}$	$[x]_{补}$	x	$[x]_{原}$	$[x]_{反}$	$[x]_{补}$
+1010	01010	01010	01010	-1001	11001	10110	10111
+0100	00100	00100	00100	-0011	10011	11100	11101
+0.1001	0.1001	0.1001	0.1001	-0.1001	1.1001	1.0110	1.0111
+0.0000	0.0000	0.0000	0.0000	-0.0000	1.0000	1.1111	1.0000

第1章 数字电路与系统基本概念

- 1.1 数字信号和数字电路
- 1.2 数字电路中的数制及转换
- 1.3 数字电路中的代码
- **1.4 数字电路中的基本逻辑函数**
- 1.5 逻辑代数

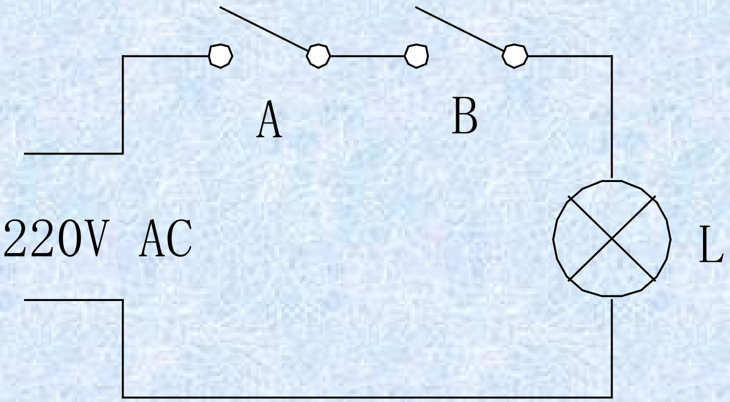
- 逻辑代数，又称布尔代数。由英国数学家乔治·布尔在1849提出，它用来描述客观事物中的逻辑关系。
- 用字母或符号表示变量，但是，该变量不代表具体数值大小，而只代表某种因果关系，或代表二种截然不同的状态，电平等。例如，开关的断开和闭合、晶体管的截止和饱和导电，灯的亮和暗，事件的是和非，真和假……等

一、 三种基本逻辑函数

- “与”逻辑
- “或”逻辑
- “非”逻辑

“与”逻辑关系及运算

决定某一结果成立的各种条件都具备时，结果才成立，这种条件与结果之间的关系称为“与”逻辑。



条　件		结　果
A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

- 开关闭合:逻辑“1”
  断开: 逻辑“0”
- 灯亮暗结果
  亮:逻辑“1”
  暗:逻辑“0”
- 要使结果成立(L=“1”),二只串联的开关都必须闭合(A=“1”，B=“1”)。

$L=A\cdot B$

- “与”逻辑就是逻辑乘

$0 \cdot 0=0$，$0 \cdot 1=0$

$1 \cdot 0=0$，$1 \cdot 1=1$

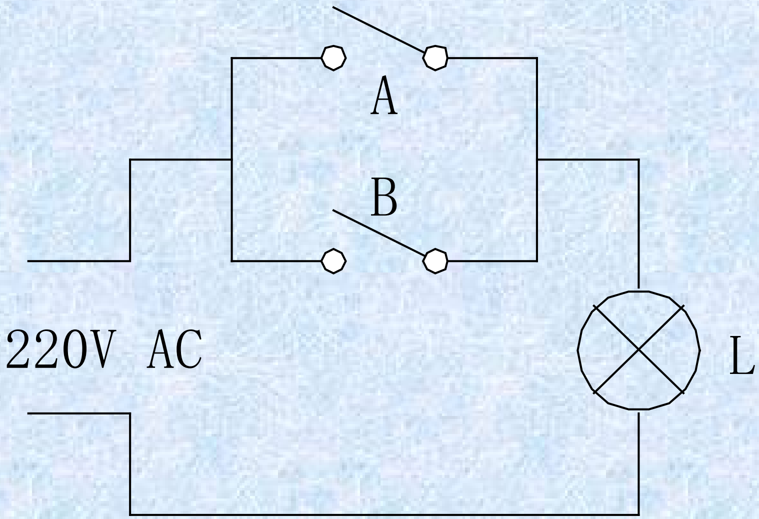
$$L = A \,\&\, B = A \bullet B = AB$$

- 能完成“与”逻辑功能的电路称为“与”门电路

“与”门逻辑符号

- “或”逻辑关系及运算

如果决定结果成立的条件中，只要有一个或一个以上的条件具备时，结果就能成立。



- 开关闭合:逻辑“1”
  断开:逻辑“0”
- 灯亮暗结果
  亮:逻辑“1”
  暗:逻辑“0”

条　件		结　果
A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

$L=A+B$

- “或”逻辑就是逻辑加

  0+0=0,　0+1=1,

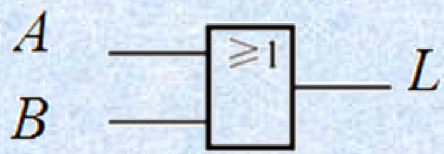
  1+0=1,　1+1=1

- 实现“或”逻辑功能的电路称为“或”门电路

$L=A+B$

“或”门逻辑符号

国际标准

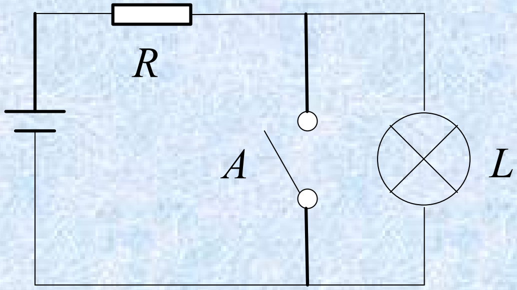


国家标准

➢ “非” 逻辑关系及运算

当条件具备时，结果不成立，

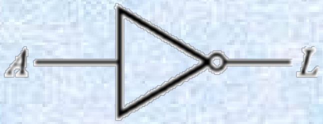
反之，结果成立



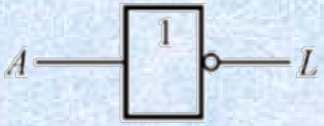
条　件	结　果
A	L
0	1
1	0

$$L = \overline{A} = A'$$

“非” 门逻辑符号



国际标准



国家标准

二、复杂逻辑关系

- “异或”逻辑关系

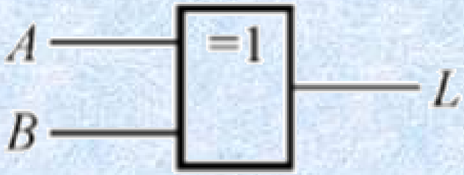
当决定结果的二个条件相异时，结果成立，二个条件相同时，结果不成立。

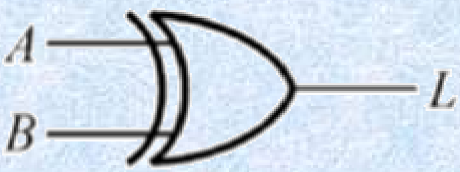
条件A B	结果 L
0 0	0
0 1	1
1 0	1
1 1	0

异或

$$L = f(A,B) = A \cdot \overline{B} + \overline{A} \cdot B$$

$$= A \oplus B$$





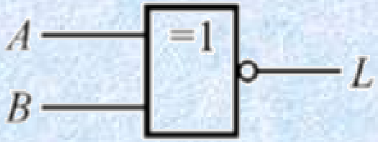
➢ “同或” 逻辑关系

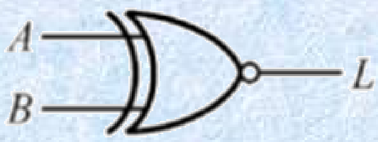
二个条件相同时，结果成立，二个条件相异时，结果不成立。

$$L = f(A,B) = \overline{A}\cdot\overline{B} + A\cdot B$$

$$= A \odot B$$

条件A B	结果 L
0 0	1
0 1	0
1 0	0
1 1	1



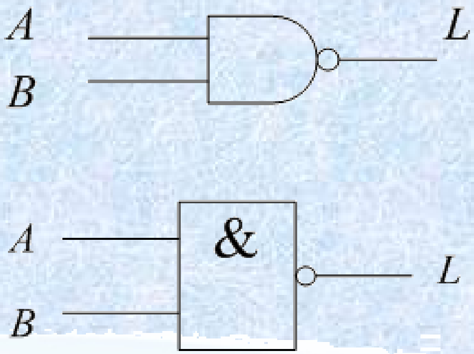


➢ 复合逻辑关系

它由“与”、“或”、“非”三种基本逻辑关系组合而成。

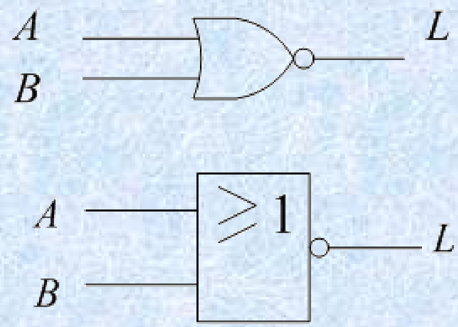
与非

$$L=\overline{AB}=(AB)'$$



或非

$$L=\overline{A+B}=(A+B)'$$



与或非

$$L=\overline{AB+CD}=(AB+CD)'$$