数码

数码是用来表示数值的基本**符号**

如

十六进制数码有 "0"、"1"、"2"、"3"、"4"、"5"、"6"、"7"、"8"、"9"、"A"、"B"、"C"、"D"、"E"、"F"

十进制数码有 "0"、"1"、"2"、"3"、"4"、"5"、"6"、"7"、"8"、"9"

八进制数码有 "0"、"1"、"2"、"3"、"4"、"5"、"6"、"7"

二进制数码有 "0"、"1"

数制

用一组固定的**符号**和统一的**规则**来表示数值的**方法**，称为**数制。**

如

二进制，"逢二进一"

十进制，"逢十进一"

任意进制的表示

任意进制的表达式为

$$(N)_R = \sum_{i=-\infty}^{\infty} K_i \times R^i$$

其中

N为若干数码的一种组合

R为进制数，又称作底数、基数。同一个数码在不同位置上的**权重**不同，第i个数码的**位权**为R^i

K_i为第i位数码的值

其中最为常用的进制有**十进制**、**二进制**、**八进制**、**十六进制。**

> 这四种进制每一种都是在特定场合下**最方便**的表示
>
> 十进制：人类最常用的进制，现实依据是人有10根手指。
>
> 二进制：计算机底层最常用的进制，现实依据是数字电路有2种电平。
>
> 八进制：和二进制做转换时最方便的进制。
>
> 十六进制：和二进制做转换时数码最少的进制。

上述的这四种进制每个都有特定的缩写

二进制：Binary 缩写作 Bin 或 B

八进制：Octal 缩写作 Oct 或 O

十进制：Decimal 缩写作 Dec 或 D

十六进制：Hexadecimal 缩写作 Hex 或 H

> 有一个冷笑话，程序员分不清圣诞节和万圣节，因为 Oct 31 = Dec 25
>
> ————尼斯湖水怪

数制转换

R进制转十进制

如下展开并求和

$$(N)_R = \sum_{i=-\infty}^{\infty} K_i \times R^i$$

例如

$$N = 1234 \quad R = 10$$

$$(1234)_{10} = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 = 1234$$

$$N = 1234 \quad R = 8$$

$$(1234)_8 = 1 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 = 668$$

其中**十进制**的括号与下标可以**省略**，在不加以标注的情况下一切数值都作**十进制**处理。

十进制转二进制

小数点前部分

牢记口诀"**除二取余、倒序读数**"

例如

将十进制数37转换为二进制数

$$\begin{aligned} 37 \div 2 = 18 \cdots\cdots 1 \\ 18 \div 2 = 9 \cdots\cdots 0 \\ 9 \div 2 = 4 \cdots\cdots 1 \\ 4 \div 2 = 2 \cdots\cdots 0 \\ 2 \div 2 = 1 \cdots\cdots 0 \\ 1 \div 2 = 0 \cdots\cdots 1 \end{aligned}$$

倒序读数，即对应的二进制数为$(100101)_2$

小数点后部分★★★

牢记口诀"**乘二取个、正序读数**"

例如

将十进制数0.6875转换为二进制数

$$
\begin{aligned}
0.6875 \times 2 = 0.375 \cdots\cdots 1 \\
0.375 \times 2 = 0.75 \cdots\cdots 0 \\
0.75 \times 2 = 0.5 \cdots\cdots 1 \\
0.5 \times 2 = 0 \cdots\cdots 1
\end{aligned}
$$

正序读数，即对应二进制为(0.1011)。

> 请理解我在乘法中使用**余数**的写法。
>
> ————尼斯湖水怪

进制误差与舍位准则★★★

部分十进制小数无法用**有限个**二进制小数来表示。

有一部分小数可以用有限个二进制数表示，例如 0.25；但大部分小数不能，例如 0.3 。

用有限个二进制小数来表示十进制小数通常会产生**误差**。

就像**十进制**的”**四舍五入**“，对于**二进制**而言就是”**零舍一入**“。

例如

将十进制数0.41转换为二进制数，保留5位小数

$$
\begin{aligned}
0.41 \times 2 = 0.82 \cdots\cdots 0 \\
0.82 \times 2 = 0.64 \cdots\cdots 1 \\
0.64 \times 2 = 0.28 \cdots\cdots 1 \\
0.28 \times 2 = 0.56 \cdots\cdots 0 \\
0.56 \times 2 = 0.12 \cdots\cdots 1
\end{aligned}
$$

正序读数，即对应二进制为(0.01101)

二进制转八进制

将二进制数码，从**小数点位置**开始，向左向右每3位二进制数码为转换成1位八进制数码。

二进制转十六进制

将二进制数码，从**小数点位置**开始，向左向右每4位二进制数码为转换成1位十六进制数码。

二进制代码

以一定的规则编制代码，用以表示十进制数值、字母、符号等的过程称为**编码。**

将代码还原成所表示的十进制数、字母、符号等的过程称为**译码。**

若所需编码的信息有N项，则需要的二进制数码的位数n应满足如下关系：

$$2^n \geq N$$

二—十进制码

用4位二进制数来表示1位十进制数，即二进制编码的十进制码(Binary-Coded-Decimal，简称**BCD码**)。

BCD码有很多种，整体可以分为两大类，**有权码**和**无权码。**

8421BCD码是一种**有权码**，即四个数码权重分别为8、4、2、1，是最常用的一种**BCD码。**

8421BCD码是由自然二进制数**0000**(0)到**1111**(15)一共16种状态的前10种组合，即**0000**(0)到**1001**(9)，其余六种状态是**无效的。**

基本逻辑运算

1. **与运算**
2. **或运算**
3. **非运算**

元件符号

常用逻辑运算

1. **与非**
2. **或非**
3. **异或**
4. **同或**

元件符号

逻辑函数的表示方法

1. 真值表
2. 逻辑函数表达式
3. 逻辑图
4. 波形图

逻辑函数表示方法间的转换

通常**逻辑函数表达式**是多种表示方法之间转换的**中介**，知道**逻辑函数表达式**而转换成其他表示方法就很方便。

真值表到表达式

找**输出值**为1的行，看该行的**输入值**，若为1则填该**变量名**，若为0则填该**变量名的非**

一共有多少个变量输入，一项中就有多少个变量相”乘“，最后将每一项相”加“。

e.g.

逻辑图到表达式

从输入信号开始，一级一级得到中间表达式，最后得到输出表达式。

e.g.

波形图到表达式

找**输出值**为1的时段，看该时段的**输入值**，若为1则填该**变量名**，若为0则填该**变量名的非**

一共有多少个变量输入，一项中就有多少个变量相”乘“，最后将每一项相”加“。

e.g.

逻辑代数运算律★★★

基本运算律

交换律	$A + B = B + A$	$A \cdot B = B \cdot A$
结合律	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
分配律	$A \cdot (B + C) = A \cdot B + A \cdot C$	

特有运算律

0律	$A+0=A$	$A\cdot 0=0$
1律	$A+1=1$	$A\cdot 1=A$
★分配律	$A+BC=(A+B)\cdot(A+C)$	
吸收律	$A+AB=A+B$	$A\cdot(A+B)=A$
★反演律	$\overline{A+B}=\overline{A}\cdot\overline{B}$	$\overline{A\cdot B}=\overline{A}+\overline{B}$

常用恒等式

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

扩充恒等式

$$A\oplus 0=A \qquad A\odot 0=\overline{A}$$
$$A\oplus 1=\overline{A} \qquad A\odot 1=A$$
$$A\oplus A=0 \qquad A\odot A=1$$
$$A\oplus B=B\oplus A \qquad A\odot B=B\odot A$$
$$\overline{A\oplus B}=A\odot B$$

> 反演律实质上揭示了**逻辑与**和**逻辑或**之间的关系——**对偶**，这个关系是通过**逻辑非**来建立的。
>
> 通过**逻辑非**，可以将**逻辑与**和**逻辑或**相互转换，这一点在后面电路实现中有体现。
>
> ————尼斯湖水怪

逻辑函数表达式的形式

与或式

即**先与后或**，(Sum of Products，简称SOP)

例如

$$、L = AB + CD$$

或与式

即**先或后与**，(Products of Sum，简称POS)

例如

$$L = (A + B)(C + D)$$

最小项与最小项表达式

最小项的定义和性质

对于有**n个变量**的逻辑函数，若有一个乘积项包含了**全部**的n个变量，每个变量都以它的**原变量**或**非变量**的形式在乘积项中出现，且**仅出现一次**，则称该乘积项为**最小项。**

> 进一步理解
>
> 每一项包含了所有输入变量，且变量之间用**逻辑与**连接，这意味着**只有**所有输入变量的状态同时匹配表达式，那一项的结果才为1。
>
> 例如，某一项为 $A\,\overline{B}\,\overline{C}$ 那么唯有当输入为**100**时，该式才为1。
>
> 每一项之间用**逻辑或**连接，这意味着**只要**有一项为1，那最终的结果就为1。、
>
> 例如，$L = A\,\overline{B}\,\overline{C} + A\,B\,\overline{C} + \overline{A}\,B\,C$，当输入为**100**、**110**、**011**的时候，输出为1。