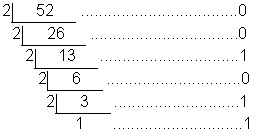
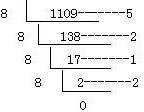
：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽**一、进制转换**

**1十进制转其他进制**

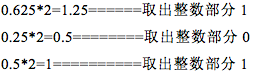
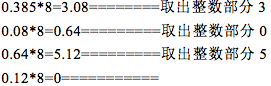
整数：除以进制数，并取余数的倒序

例：(52)10 = (110100)2、(1109)10 =(2125)8

 ：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽

小数：乘以进制数，取进位的正序：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽：码an'ma﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽

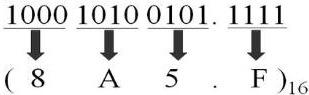
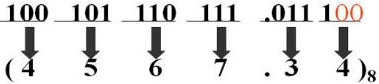
例：(0.625)10 = (0.001)2、(0.385)10 =(0.305)8

**2其他进制转十进制**

(101.1011)2 = 1\*22 + 0\*21 + 1\*20 + 1\*2-1 + 0\*2-2 + 1\*2-3 + 1\*2-4

**3二进制转其他进制**

**二、计算机二进制所承载的信息**

指令信息：用于CPU可识别的操作(片选信号)

地址信息：字节对应的物理地址(64位片选信号)

数据信息：数值、字符

**三、二进制数据信息的编码**

**1非数值数据**

ASCII码：用八位二进制描述英文、数字和符号的一个字符

UTF-8码：用做多3字节描述Unicode字符集

**2数值数据**

BCD码：用四位二进制表示一位十进制

原码：最高位为符号(0正1负)，数值跟随其后

(13)10 = [01101]原 (-13)10 = [11101]原

反码：最高位为符号(0正1负)，正数同原码，负数逐位取反

(13)10 = [01101]反 (-13)10 = [10010]反

补码：最高位为符号(0正1负)，正数同原码，负数为反码+1

(13)10 = [01101]反 (-13)10 = [10011]反

余2m-1码：每个数都用自身和2m-1的和表示，范围-2m-1 ~ (2m-1 - 1)

(13)10 = 01101 + 25-1 = [11101]余16码

(-13)10 = -01101 + 25-1 = [00011]余16码

**四、二进制数据的运算**

**1加减运算**

补码加减法：[X + Y]补 = [X]补 + [Y]补

例：X=0.1010 Y= -0.0101

[X+Y]补 = 0.1011 + 1.1011 = 10.0110 = 0.0110 模2舍去

例：X= -0.1010 Y= -0.0101

[X+Y]补 = 1.0110 + 1.1011 = 11.0001 = 1.0001 模2舍去(若符号取反则溢出)

溢出：异号相加无溢出，同号相加符号取反溢出，通常计算机采用双符号位识别溢出

例：X= -0.1101 Y= -0.1011，用双符号位计算

[X+Y]补 = 11.0011 + 11.0101 = 10.1000 溢出(双符号位不同)

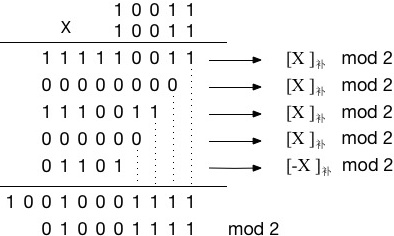
**2乘法运算**

补码的右移

[x]补 = X0.X1X2...Xn 则右移一位有 [x/2]补 = X0.X0X1X2...Xn

补码一位乘法：若[Y]补 = Y0.Y1Y2Y3…Yn 那么[X·Y]补 = [X]补·(-Y0 + 0.Y1Y2Y3…Yn)

例：X=-0.1101 Y=-0.1011 有[X]补=1.0011 [Y]补=1.0101



直接补码乘法：该乘法器有如下公式 [X]补·[Y]补= [X·Y]真

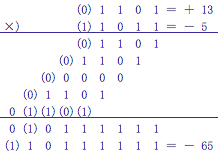
若 [X]补=(X4)X3X2X1X0 [Y]补=(Y4)Y3Y2Y1Y0

有 X = X4(-24) + X3(23) + X2(22) + X1(21) + X0(20)

Y = Y4(-24) + Y3(23) + Y2(22) + Y1(21) + Y0(20)

即 补码转化为真值时，符号位取负权值，其余位取正权值

例：[X]补=01101 [Y]补=11011



注 带()的位表示负权值，与正权值相加时，实际为相减

结果通过位权调整0(1) = (1)1或1(1) = (0)1将负权值调整到符号位

直接补码阵列乘法器

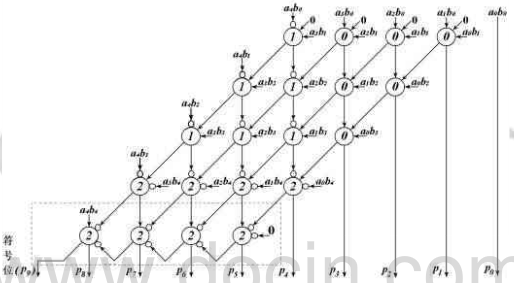
A阵列加法器逻辑单元：乘法器包含3类全加器，见下图

0类全加逻辑单元 输入输出均为正权值

1类全加逻辑单元 一个输入负权，一个输出负权

2类全加逻辑单元 两个输入负权，一个输出负权

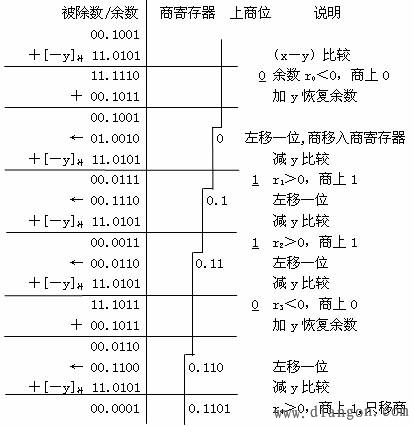
B阵列乘法器



**3除法运算**

原码恢复余数法：除法通过减法实现，试商为1时，若余数得负表除不尽，应恢复余数

例：[X]原 = 0.1001 [Y]原 = 0.1011 [-Y]补 = 1.0101 求[X]原 / [Y]原



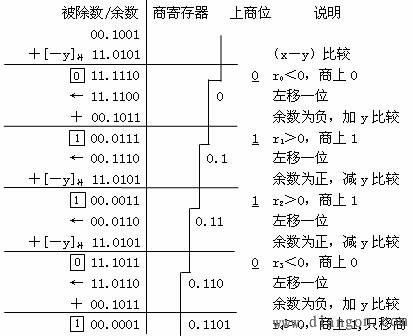
原码加减交替法：通过加减除数Y实现除法

若 余数Ri = 2Ri-1 - Y

当 Ri < 0时

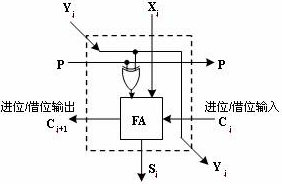
有 余数Ri+1 = 2(Ri + Y) - Y = 2Ri + Y

例：[X]原 = 0.1001 [Y]原 = 0.1011 [-Y]补 = 1.0101 求[X]原 / [Y]原

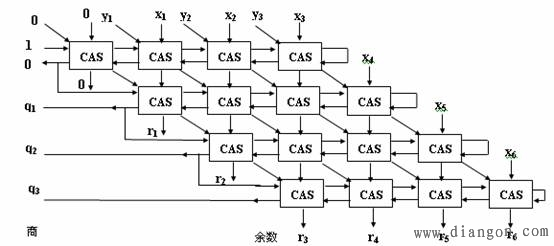


原码加减交替阵列除法器

A阵列除法器逻辑单元



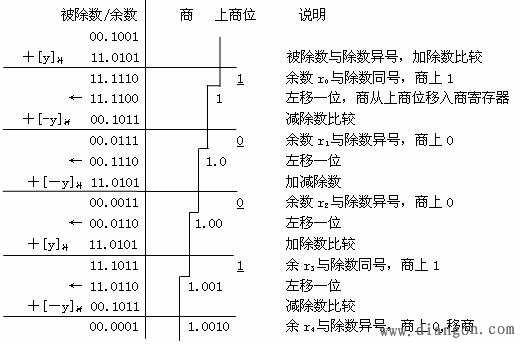
B阵列除法器



补码加减交替除法：

A被除数与除数同号，被除数减去除数；被除数与除数异号，被除数加上除数。   
 B余数与除数同号，商上1，余数左移一位减去除数；余数与除数异号，商上0，余数左移一位加上除数。（注意：余数左移加上或减去除数后就得到了新余数。）   
 C重复B，直到商的位数满足要求为止。

例：[X]补=0.1001 [Y]补=1.0101 [-Y]补=0.1011 求 [X/Y]补



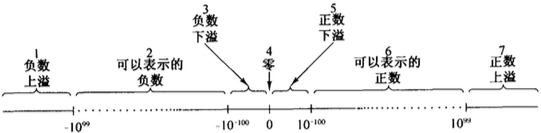
**五、浮点数**

**1浮点数的原理**

定义：类似科学计数法把精度和表示范围分开

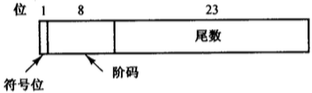
浮点数 = M·2E (M为尾数 E为阶码)

表示范围：以3位尾数，2位阶码为例



**2 IEEE754标准浮点数**

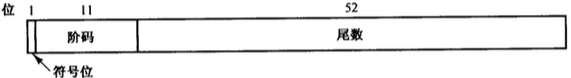
单精度浮点数32位



阶码：8位余127码

尾数：23位原码，缺省存储最高位的1(尾数规格化)，尾数实际表示范围24位

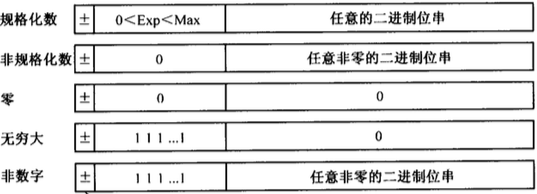
双精度浮点度64位



阶码：11位余1023码

尾数：52位原码，缺省存储最高位的1(尾数规格化)，尾数实际表示范围53位

规格化数：阶码不为0或全为1



**3浮点数的加减法运算**

对阶：使两数阶码相同。求两数阶码差△E，并右移阶码小的浮点数的尾数△E位

尾数加减：加减对阶后的尾数

规格化操作：使尾数规格化，即0.1xxx的格式，阶码做相应调整

舍入

检查阶码是否溢出

**4浮点数的乘除法运算**

阶码：乘法相加，除法相减

尾数：乘法相乘，除法相除

**六、数据校验**

**1奇偶校验码**

一个字节补充一位校验为，最高一位为校验位

**2海明校验码**

原理：数据中插入几个校验位，把数据位分组进行奇偶校验，其校验值放入校验位

若 校验位个数为r，即能表示2r个信息，其中一个标识正确信息

则 2r - 1个信息用于指出哪里错位，除去验证码自身错误为2r - 1 - r

即 数据位与验证码位应满足k = 2r - 1 - r

作用：通过组合奇偶校验，不但可发现错误，还可指出哪一位出错

编码规则：

·数据位与验证位关系满足k = 2r - 1 - r

·海明码中校验位Pi被分配在2i-1的位置

·海明码每位Hi参与多个校验位的奇偶校验，且满参与校验位的海明位号之和等于i

如海明位H5对应数据位D2，则其参与的校验位包括H1(即P1)和H4(即P3)

|  |  |  |  |
| --- | --- | --- | --- |
| 海明码位号 | 数据位/校验位 | 参与校验的校验位 | 海明位与校验位关系 |
| H1 | P1 | 1 | 1=1 |
| H2 | P2 | 2 | 2=2 |
| H3 | D1 | 1,2 | 3=1+2 |
| H4 | P3 | 4 | 4=4 |
| H5 | D2 | 1,4 | 5=1+4 |
| H6 | D3 | 2,4 | 6=2+4 |
| H7 | D4 | 1,2,4 | 7=1+2+4 |
| H8 | P4 | 8 | 8=8 |
| H9 | D5 | 1,8 | 9=1+8 |
| H10 | D6 | 2,8 | 10=2+8 |
| H11 | D7 | 1,2,8 | 11=1+2=8 |
| H12 | D8 | 4,8 | 12=4+8 |
| H13 | P5 | 13 | 13=13 |

生成验证码

P1 = D1⊕D2⊕D4⊕D5⊕D7 P2 = D1⊕D3⊕D4⊕D6⊕D7

P3 = D2⊕D3⊕D4⊕D8 P4 = D5⊕D6⊕D7⊕D8

P5 = D1⊕D2⊕D3⊕D4⊕D5 D6⊕D7⊕D8⊕P4⊕P3⊕P2⊕P1

验证海明码

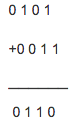
S1 = P1⊕D1⊕D2⊕D4⊕D5⊕D7 S2 = P2⊕D1⊕D3⊕D4⊕D6⊕D7

S3, S4依此类推，最终通过片选S1,S2,S3,S4指出并纠正发生错误的位

**3循环冗余校验(CRC)码**

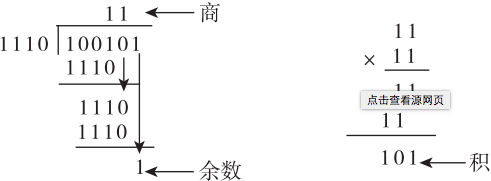
模2运算

加减法：按位半加，补进位，加法与减法值相同，可通过异或逻辑实现



乘法：按模2加法求部分积之和

除法：按模2减法求部分余数，部分余数首位1则上商1，否则0



被模2除的循环性：余数补0后继续模2除，余数将循环出现

CRC码的编码方法

·约定发送方与接收方的生成多项式(除数)

G(x) = G(X)=X3+X+1即1011

生成多项式的最高位与最低位必须是1

·将被编码数据左移r位（r = G(x)最高次幂数，上述为3）

·左移数据模2除以G(x)，所得余数即CRC码

·左移数据加CRC码，即得CRC编码数据

CRC的校验：CRC编码除G(x)余数为0时表示正确

见 [CRC验证](http://blog.163.com/yucheng_xiao/blog/static/76600192201393092918776/?COLLCC=1540649525&)

证 若数据为M(x)，CRC验证码为R(x)，商为Q(x) 则有

Q(x)·G(x) + R(x) = M(x)

M(x)·xr + R(x) = Q(x)·G(x) + R(x) + R(x) 模2相加

M(x)·xr + R(x) = Q(x)·G(x)

CRC的纠错：根据余数出错位表纠正错误位



CRC模2运算及LFSR电路实现

