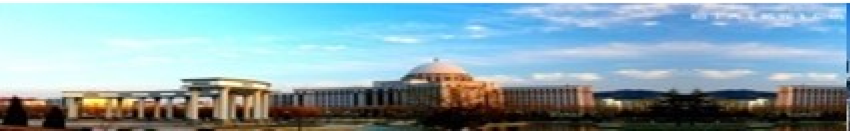
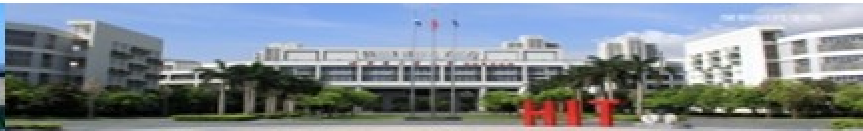
EDA技术高级应用

哈尔滨工业大学（威海）





目 录

CONTENTS

第5章

浮点数运算设计

群名称:EDA技术高级应用
群　号:1044047663

5.1 浮点数

在 20世纪80年代以前，每个计算机厂商都设计了自己表示浮点数的规则，以及对浮点数执行运算的细节，这对于应用程序在不同机器上的移植造成了巨大的困难。而在这之后，也就是1985年左右，**IEEE标准**产生了，这是一个仔细制定的表示浮点数及其运算的标准，**现在的计算机浮点数也都是采用这个标准。**

浮点数不仅仅是为了让数值的表示更加精确，也是为了表示一些整数无法达到的数字，比如一些**接近于0**的数字，或者一些**非常大**的数值。

因此**浮点数对于计算机的意义是非常大**的。

5.2 IEEE 754 标准

IEEE，电气和电子工程师协会(Institute of Electrical and Electronics Engineers)是一个国际性的电子技术与信息科学工程师的协会，是目前全球最大的非营利性专业技术学会。

IEEE754标准是**IEEE二进位浮点数算术标准**（IEEE Standard for Floating-Point Arithmetic）的标准编号。

IEEE 浮点标准表示：

$$V = (-1)^s * M * 2^E$$ 。

①s是符号位，为0时表示正，为1时表示负；

②M为尾数，是一个二进制小数，它的范围是0至1-ε，或者1至2-ε（ε的值一般是2^{-k}次方，其中设k ＞ 0）；

③E为阶码，可正可负，作用是给尾数加权。

将十进制数 25.125 转换为浮点数，转换过程就是这样的（D代表十进制，B代表二进制）：

- **整数部分：** 25(D) = 11001(B)
- **小数部分：** 0.125(D) = 0.001(B)
- **用二进制科学计数法表示：**

25.125(D) = 11001.001(B) = 1.1001001 * 2^4(B)

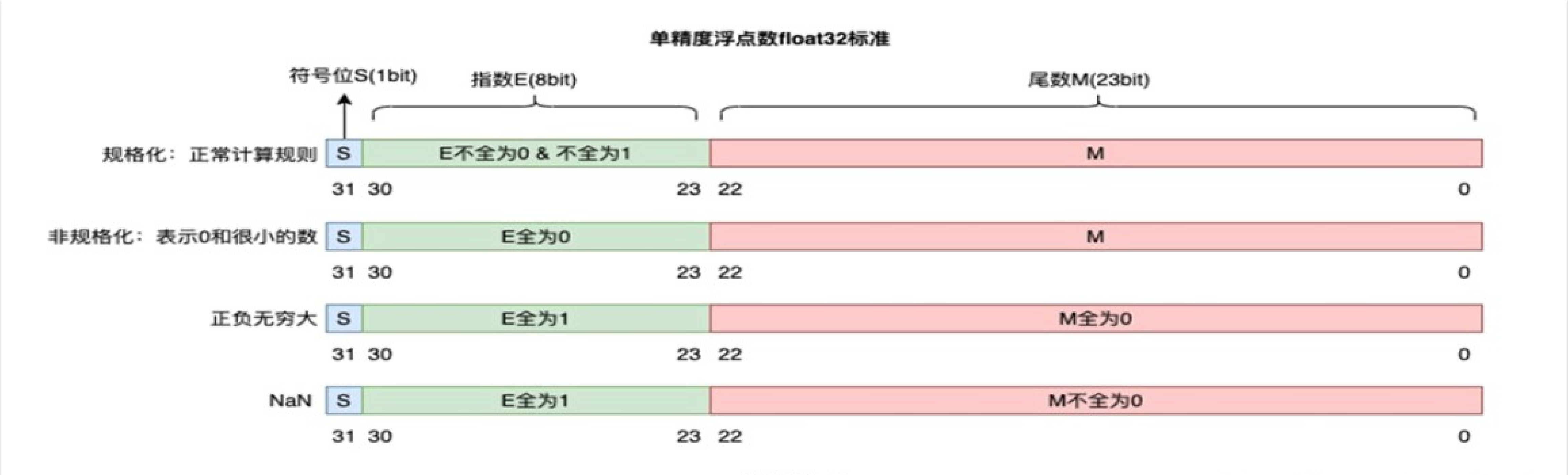
所以**符号位** S = 0，**尾数** M = 1.001001(B)，**指数** E = 4(D) = 100(B)。

指数和尾数分配的位数不同，会产生以下情况：

（1）指数位越多，尾数位则越少，其表示的范围越大，但精度就会变差；**反之**，指数位越少，尾数位则越多，表示的范围越小，但精度就会变好；

（2）一个数字的浮点数格式，会因为定义的规则不同，得到的结果也不同，表示的范围和精度也有差异。

- 指数 E 非全 0 且非全 1：规格化数字，按上面的规则正常计算；
- 指数 E 全 0，尾数非 0：非规格化数，尾数隐藏位不再是 1，而是 0(M = 0.xxxxx)，表示 0 和很小的数；
- 指数 E 全 1，尾数全 0：正无穷大/负无穷大（正负取决于 S 符号位）；
- 指数 E 全 1，尾数非 0：NaN(Not a Number)。



把 25.125 转换为标准的 float 浮点数：

- **整数部分**：25(D) = 11001(B)
- **小数部分**：0.125(D) = 0.001(B)
- **用二进制科学计数法表示**：25.125(D) = 11001.001(B) = 1.1001001 * 2^4(B)

所以 S = 0，尾数 M = 1.001001 = 001001(去掉1，隐藏位)，指数 E = 4 + 127(中间数) = 135(D) = 1000111(B)。

现在的编译器都支持两种浮点格式，一种是**单精度**，一种是**双精度**。

单双精度分别对应于编程语言当中的**float**和double类型。

float是单精度的，采用**32位**二进制表示，其中1位符号位，8位阶码以及23位尾数。

double是双精度的，采用**64位**二进制表示，其中1位符号位，11位阶码以及52位尾数。

单精度

符号位 (31)	阶码 (30~23)	尾数 (22~0)

双精度

符号位 (63)	阶码 (62~52)	尾数 (51~0)

精 度

float 能表示的最小二进制数（精度）为0.0000…1（小数点后22个0，1个1），用十进制数表示就是1/2^23；能表示的最大二进制数为 +1.1.11111…1 * 2^127（小数点后23个1），而二进制 1.11111…1 ≈ 2，所以 float 能表示的最大数为 2^128 = 3.4 * 10^38，即 float 的表示范围为：-3.4 * 10^38 ~ 3.4 * 10 ^38;

double能表示的最小二进制数（精度）为0.0000…1(51个0，1个1)，用十进制表示就是 1/2^52；能表示的最大二进制数为 +1.111…111（小数点后52个1） * 2^1023 ≈ 2^1024 = 1.79 * 10^308，所以double的表示范围为：-1.79 * 10^308 ~ +1.79 * 10^308。

5.3 浮点数运算器设计

思考算法的实现过程

- 定点数到浮点数的转换：
- 加法：
- 减法：
- 乘法：
- 除法：

5.3.1 定点数到浮点数的转换

Fix_to_Float_Module.vhd

5.3.2 浮点数加法

算法思想：

设有两个浮点数 x 和 y ,它们分别为

$x = 2^{Ex} \cdot M_x$ $y = 2^{Ey} \cdot M_y$

$z = x \pm y = (M_x 2^{Ex-Ey} \pm M_y) 2^{Ey}, E_x \leq E_y$

5.3.2 浮点数加法

需要6步完成:

(1) 0 操作数的检查:

如果判知两个操作数 x 或 y 中有一个数为0，即可得知运算结果而没有必要再进行后续的一系列操作以节省运算时间。

0操作数检查步骤则用来完成这一功能。

5.3.2 浮点数加法

(2) 对阶操作：

比较两个浮点数的阶码值的大小，求△E=Ex-Ey。当其不等于零时，首先应使两个数取相同的阶码值。其实现方法是，将原来阶码小的数的尾数右移|△E|位，其阶码值加上|△E|，即每右移一次尾数要使阶码加1，则该浮点数的值不变(但精度变差了)。

尾数右移时，对原码形式的尾数，符号位不参加移位，尾数高位补0；对补码形式的尾数，符号位要参加右移并使自己保持不变。为减少误差，可用另外的线路，保留右移过程中丢掉的一到几位的高位值，供以后舍入操作使用。

(3)实现尾数的加(减)运算，对两个完成对阶后的浮点数执行求和(差)操作。

(4)规格化处理

若得到的结果不满足规格化规则，就必须把它变成规格化的数。

对双符号位的补码尾数来说，就必须是001××…×或110××…×的形式。

规格化处理规则是：

- 当结果尾数的两个符号位的值不同时，表明尾数运算结果溢出。此时应使结果尾数右移一位，并使阶码的值加1，这被称为向右规格化，简称**右规**。
- 当尾数的运算结果不溢出，但最高数值位与符号位同值，表明不满足规格化规则，此时应重复地使尾数左移、阶减减1，直到出现在最高数值位上的值与符号位的值不同为止，这是向左规格化的操作，简称**左规**。

(5)舍入操作:

在执行对阶或右规操作时，会使尾数低位上的一位或多位的数值被移掉，使数值的精度受到影响，可以把移掉的几个高位的值保存起来供舍入使用。**舍入的总的原则是要有舍有入，而且尽量使舍和入的机会均等，以防止误差积累。**常用的办法有“0”舍“1”入法，即移掉的最高位为1时，则在尾数末位加1；为0时则舍去移掉的数值。该方案的最大误差为 2-(n+1)。

这样做可能又使尾数溢出，此时就要再做一次右规。

另一种方法“置1”法，即右移时，丢掉移出的原低位上的值，并把结果的最低位置成1。该方案同样有使结果尾数**变大或变小两种可能**。即舍入前尾数最低位已为0，使其变1，对正数而言，其值变大，等于最低位入了个1。若尾数最低位已为1，则再对其置1无实际效用，等于舍掉了丢失的尾数低位值。

(6) 判断结果的正确性,即检查阶码是否溢出。

浮点数的溢出是以其阶码溢出表现出来的。

在加减运算真正结束前,要检查是否产生了溢出。

- 若阶码正常，加(减)运算正常结束；
- 若阶码下溢,要置运算结果为浮点形式的机器零；
- 若上溢，则置溢出标志。

5.3.3 浮点数乘法

◈ **算法思想:**

设有两个浮点数 x 和 y ,它们分别为

$$x = 2^{Ex} \cdot M_x \qquad y = 2^{Ey} \cdot M_y$$

$$z = x \times y = 2^{(E_x + E_y)} \cdot (M_x \times M_y)$$

$$z = x \div y = 2^{(E_x - E_y)} \cdot (M_x \div M_y)$$

步骤

Step1：0 操作数检查；

Step2：阶码加/减操作；

Step3：尾数乘/除操作；

Step4：结果规格化及舍入处理。

阶码的运算

- 阶码都是**补码**

$$[x+y]_{补}=[x]_{补}+[y]_{补}$$
$$[x-y]_{补}=[x]_{补}+[-y]_{补}$$

- 阶码都是**移码**

$$[x+y]_{移}=[x]_{移}+[y]_{补}$$
$$[x-y]_{移}=[x]_{移}+[-y]_{补}$$

尾数的运算

- **第1种方法:**无条件地丢掉正常尾数最低位之后的全部数值。这种办法被称为截断处理，好处是处理简单,缺点是影响结果的精度。
- **第2种方法:**运算过程中保留右移中移出的若干高位的值，最后再按某种规则用这些位上的值修正尾数。这种处理方法被称为舍入处理。

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

舍入处理

当尾数用原码表示时:

- 方法一是只要尾数的最低位为1，或移出的几位中有为1的数值位，就置最低位的值为1。
- 方法二是0舍1入法，即当丢失的最高位的值为1时，把这个1加到最低数值位上进行修正，否则舍去丢失的的各位的值。

哈尔滨工业大学
HARBIN INSTITUTE OF TECHNOLOGY

舍入处理

当尾数是用补码表示时

- 当丢失的各位均为0时，不必舍入；
- 当丢失的最高位为0时，以下各位不全为0时，或者丢失的最高位为1，以下各位均为0时，则舍去丢失位上的值；
- 当丢失的最高位为1，以下各位不全为0时，则执行在尾数最低位入1的修正操作。