《集成电路设计实践》课程

结课调研报告

调研选题：“浮点数乘法器”

的设计与发展

姓名：施念

学号：1120161302

班号：05011609

专业：电子信息工程

摘要

在微处理系统中，数据类型分为定点数和浮点数。随着电路系统数值运算范围需求的不断扩大，数据的灵活性以及精确度要求不断提高，二者之中，浮点数地位越来越高，因此，对浮点数的研究在当下显得尤为重要。

要提高浮点数运算的性能，就必须解决浮点数运算单元硬件复杂、功耗大、延时的问题。在实际问题中，可以采用非精确计算等其他方法减少容错设备的动态及静态能量损耗，来提高运算的性能。但性能与精确度不能同时改进，本文就浮点数乘法器的算法与设计进行了分析与总结。

**关键词**：非精确 乘法器 浮点数 节约面积 节能

目录

[1 调研背景 1](#_Toc512376757)

[1.1浮点数表示方法 1](#_Toc512376758)

[1.2 精确浮点数乘法器 1](#_Toc512376759)

[2 调研内容 2](#_Toc512376760)

[2.1 非精确浮点数乘法器结构 2](#_Toc512376761)

[2.1.1 算法结构 2](#_Toc512376762)

[2.1.2 舍入单元 3](#_Toc512376763)

[2.2 节能型多格式浮点乘法器 3](#_Toc512376764)

[2.2.1 基数为16的乘法器 4](#_Toc512376765)

[2.2.2 多格式浮点数乘法器 4](#_Toc512376766)

[2.2.3 改进的多格式浮点数乘法器 4](#_Toc512376767)

[2.3 使用混合GPP加法的区域高效的32位浮点乘法器 5](#_Toc512376768)

[2.3.1 32位FP乘法器 6](#_Toc512376769)

[2.3.2 用于GPP积累的混合加法器 7](#_Toc512376770)

[3 总结 8](#_Toc512376771)

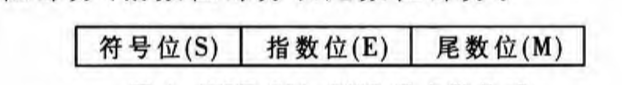
[参考文献： 9](#_Toc512376772)

# 1 调研背景

## 1.1浮点数表示方法

在算数运算中，数据的表示方法有两个，定点数和浮点数。定点数可以表示以0为中心的一定范围的正负整数，由于定点数小数点固定，不能表示太大动态范围的数，同时也不能表示过大或过小的数，此时浮点数的优势就体现了出来，它可以在不增加位数的前提下扩展数据的动态范围。【1】

在IEEE 754—2008标准中，浮点数是一个以3个量表示的二进制位串(如图1所示)，该位串分为3个部分：符号位部分、指数位部分和尾数位部分。



（图 1）

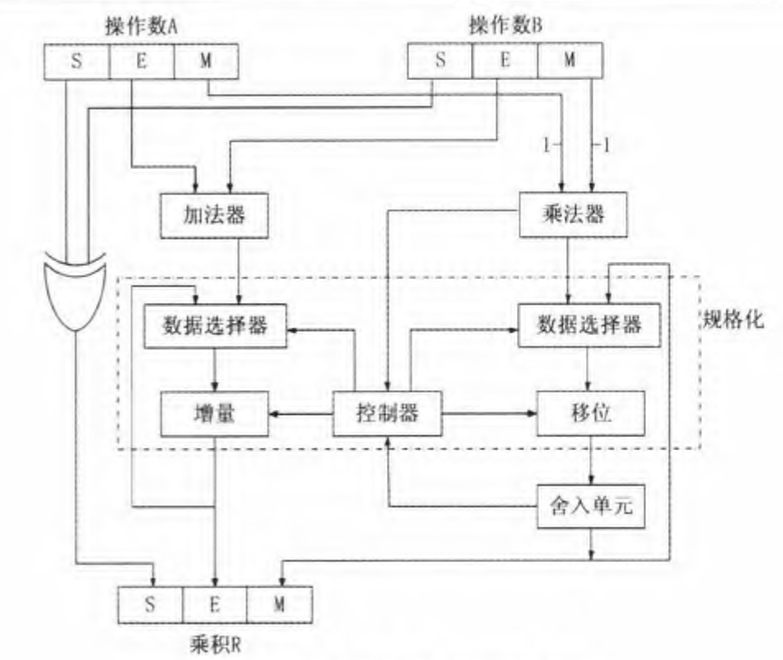
浮点数的标准表达式为：

FP 

其中，S为符号位，当S=0时表示该浮点数是一个正值，当S=1时表示该浮点数是一个负值；E为指数位，bias表示偏移量，对于单精度bias=127，双精度bias=1023，则E—bias既可为正数也可为负数；M为尾数位，对于规格化的尾数，存储时默认省去小数点前的l，则对于非零尾数，尾数值为1.M。【1】

## 1.2 精确浮点数乘法器

精确的浮点数乘法器如图2所示，默认这里两个操作数都已规格化，如若没有，则首先需将输入的操作数规格化。然后从两操作数中分别提取每个操作数的符号位、指数位及尾数位，并将尾数位补充省略的第一位1。浮点数的乘法只要包括两个运算：指数的加法和尾数的 乘法，这两运算结束后就是将结果进行规格化，生成符合IEEE 754—2008标准的结果。



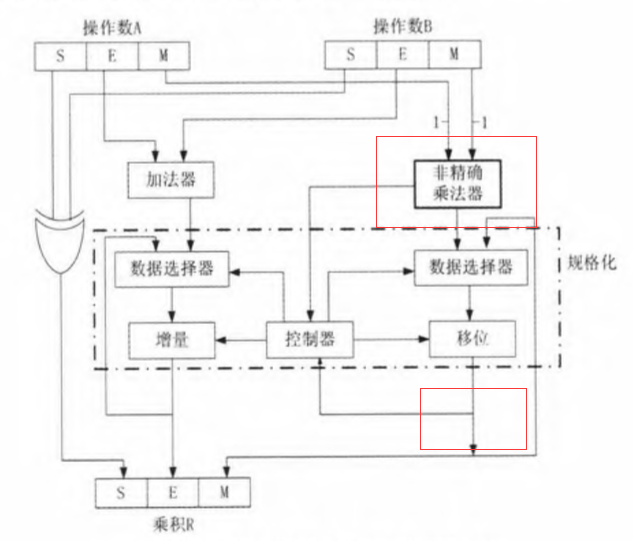
（图 2）

# 2 调研内容

## 2.1 非精确浮点数乘法器结构

### 2.1.1 算法结构

非精度乘法器的设计是将非精度定点数乘法器运用到浮点数尾数乘法器设计中，同时考虑到尾数乘积已是非精确结果，舍入单元及规格化也进行了简化。非精确浮点数乘法器具体结构如图3所示，从图中可以看出发生变化的部分（红色框内为与精确浮点数乘法器不同的地方）。【1】



（图 3）

### 2.1.2 舍入单元

观察图2中的电路，舍入单元被舍去。舍入单元的作用是在结果被返回放回浮点格式时，将多出来的位舍弃，使得有效数据的位数保持在固定的位数范围内。在精确浮点数乘法器中，IEEE列出了4种不同的舍入方法：RTF（舍入到最近）、RTPI（朝正无穷大舍入）、RTNI（朝负无穷大舍入）和RTZ（朝0舍入）。在非精度设计中，乘积结果已是非精确的，在非精度乘法器中，舍入单元不在考虑。【1】

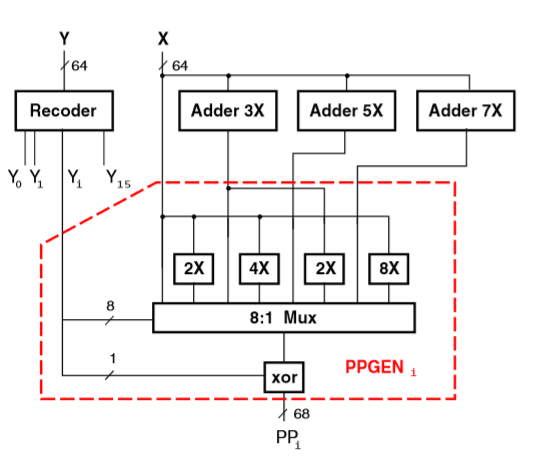
## 2.2 节能型多格式浮点乘法器

在这个节能型多格式乘法器中，主要是一个基数为16的乘法器，它支持*binary32*（单精度）和*binary64*（双精度）的整数和FP（浮点）乘法。在这个乘法器中，它可以并行执行两个*二进制32*乘法（双通道）。我们选择了基数为16来限制PP积累树的深度并节省电力。

### 2.2.1 基数为16的乘法器

在图4中，我们将X表示乘数操作数，Y表示乘数。首先，对乘数操作数Y进行重新编码，这种重新编码完成了传输位是四位组的最高有效位。由于基数为16的重新编码，对于64位Y，部分PP的数量为16。但是，对于一般的 n位乘法器操作，由于是用基数为16的数字传输数字，得到的基数为16的数字是 [(n+1)/4]。因此，当n= 64，PP的数量是17。【2】

通过根据基数为16位的值选择X的倍数来生成PPs。图4中显示出了PP生成的可能实现。当记录的乘数数字为负时，一组XOR门补充了PP的位。【2】



（图 4）

### 2.2.2 多格式浮点数乘法器

构建多格式FP乘法器（MFmult）的起点是基数为16的 64×64的乘法器。MFmult应该支持以下格式：

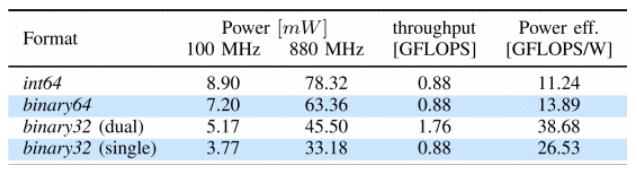
* int64：两个64位无符号整数的乘法产生一个128位的数。这是第二部分单元实施的操作。
* fp64：乘以两个浮点二进制64位数字（以前称为双精度），产生二进制64结果。
* fp32：两个浮点二进制32位数字（单精度）的两次乘法运算，产生两个二进制32结果。【2】

上述binary64 FP格式需要存储64个比特：1比特作为标志位，11比特作为指数位，52个比特用于有效数的小数部分。如果偏置指数大于零，则整数位为'1'，结果的有效位数为53位。因此，通过将操作数X和Y与64×64乘法的最低有效位对齐，可以容易地在图4的64×64乘法器中容纳53×53的有效数乘法。【2】

一旦结果被计算出来，结果可能不会被标准化。对于归一化数字的乘法，'1'可以位于bit 105或bit 104（bit 0是LSB）。因此，如果'1'在bit 104中，则需要2：1多路复用器将产品移动到左侧。之后，我们需要在52位处加上'1'来完成舍入。通过在位置53截断结果，我们获得53位（P105... P53）的归一化和圆形的有效数。【2】

### 2.2.3 改进的多格式浮点数乘法器

如果应用程序允许降低精度，图5的功效结果建议使用binary32 FP格式。这可能是许多应用的情况，例如，小整数或小部分的乘法。

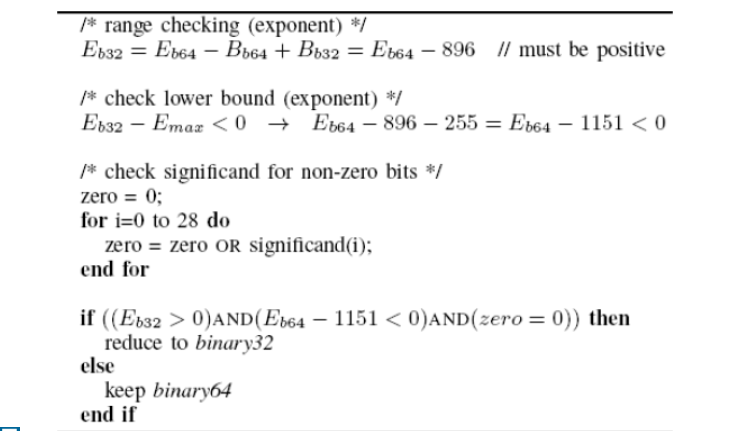


（图 5）

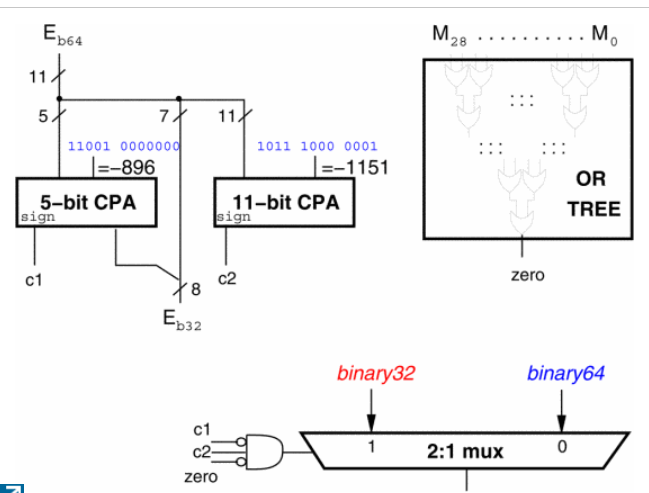
由于即使是单个binary32比binary64功耗更高，通过降低操作数的精度，我们总是可以节省功耗。【2】

接下来，通过一个简单的方法来转换binary32 FP编号中的无二进制64位 FP编号，当二进制64位有效位的非零位可以用二进制32位有效数表示（即非零位数≤23）时，范围是可表示的（即，无偏指数[-127,127]）。【2】

算法如图6所示，其硬件架构如图7所示。



（图 6）



（图 7）

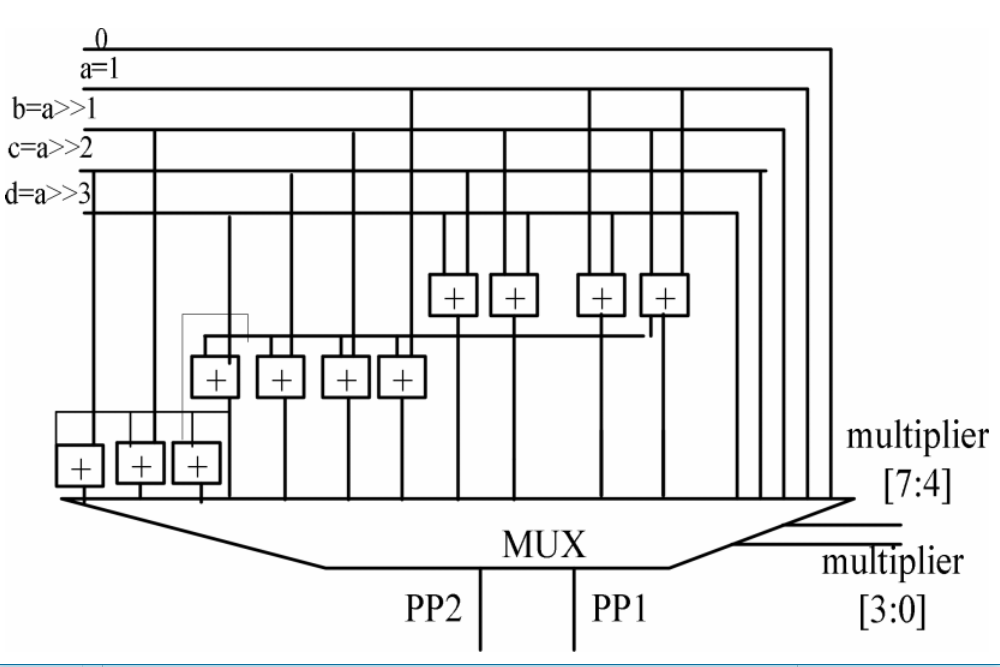
## 2.3 使用混合GPP加法的区域高效的32位浮点乘法器

通过结合传统的波纹进位加法器（RCA）和Wallace tree加法器来添加生成的部分产品（GPP），可以提高速度。应用于24×24尾数乘法器的Toom-3乘法方法，复杂度降低（n = 1.465）。对于N = 24位无符号操作数，预先确定的部分产品生成（3PG）方法将GPP的高度降低到（N / 3）/ 4。这与修改过的Booth编码（MBE）GPP减少高度N / 2相反。这种减少可以用来节省面积。与基于MBE的FP乘法器相比，该设计在TSMC 0.13μmCMOS上合成，面积减少62％。

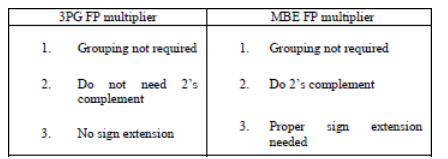
### 2.3.1 32位FP乘法器

如2.2.1中的16为基的乘法器，单精度FP具有*1*位符号，*8*位指数和23位尾数。FP乘法器需要3次计算：1.符号和指数计算；2.尾数乘法；3：标准化和舍入。FP乘法器中的瓶颈部分是尾数乘法，因此提出的设计主要集中在尾数乘法。

通过3PG方法计算PPs 如图8所示。MBE是广泛接受的GPP高度降低方法。这种减少包括需要更多门数的Booth编码器。与MBE方法相比，3PG仅使用*30*％的门数。表9给出了MBE和3PG方法的*16×16*乘数门数比较的PP加数。



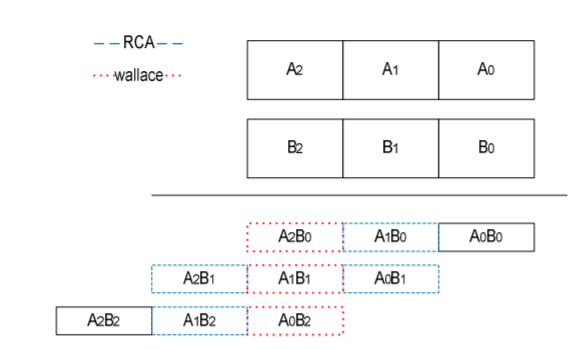
（图 8）



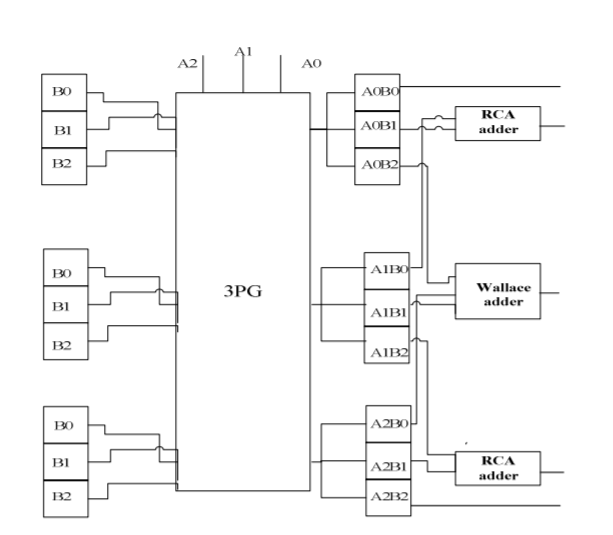
（图 9）

### 2.3.2 用于GPP积累的混合加法器

合加法器是传统波纹进位加法器和Wallace tree加法器的组合。在所有块乘法之后，如图10所示排列。整个乘法运算如图11所示。块分离之后，每个块都与适当的块相乘，并生成2n位宽度的结果。该块乘法使用进位选择加法器进行内部块乘法。在块乘法之后，2n位宽度结果与相同数据大小的相应乘法块相加。这个加法使用混合加法器。如果2n位加法有两个相乘块，则进行传统的RCA加法。如果有两个以上的块，将通过Wallace tree加法器。这种混合加法的选择用于补偿进位传播延迟。使用传统方法添加两个以上操作数需要（m-1）加法，（其中m是操作总次数）。传统方式的门延迟为O（mlogn）在Wallace中减少到O（log m \* log n），并且面积增加很少。通过CSA选择RCA将节省一些延迟时间。在3PG乘法器中，面积和延迟之间的性能折衷很好地平衡，并增加了混合GPP。



（图 10）



（图 11）

# 3 总结

在调研了诸多类型的乘法器后，可以看出，在设计电路时候，如果需要用到乘法器，可以考虑牺牲不必要的参数，换来更高的效率，当然，必要时候需要加一定的补偿。就像第一个非精确浮点数乘法器，虽然直接将舍入单元去除，但是在数据选择部分进行了优化，只牺牲了一部分精度，却极大程度的提高了效率。

除此之外，设计一种功能元件时，可以结合多种算法与电路，达到我们预期的效果，就像使用GPP的32位浮点乘法器一样，其将传统的RCA和Wallace tree加法器结合，根据数据的不同进行不同的运算。

# 参考文献：

[1] Yin Peipei．Design and analysis of inexact floating-point multiplier[J]．Application of Electronic Technique, 2016, 42(3):38-41, 46.

[2] Nannarelli, "A multi-format floating-point multiplier for power-efficient operations," 2017 30th IEEE International System-on-Chip Conference (SOCC), Munich, 2017, pp. 351-356.

[3] J.J.J.Nesam and S.Sivanantham, "An area-efficient 32-bit floating point multiplier using hybrid GPPs addition," 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS), Vellore, 2017, pp. 1-4.