《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

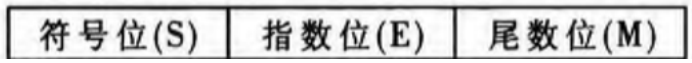
目录

1 调研背景

1.1 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）

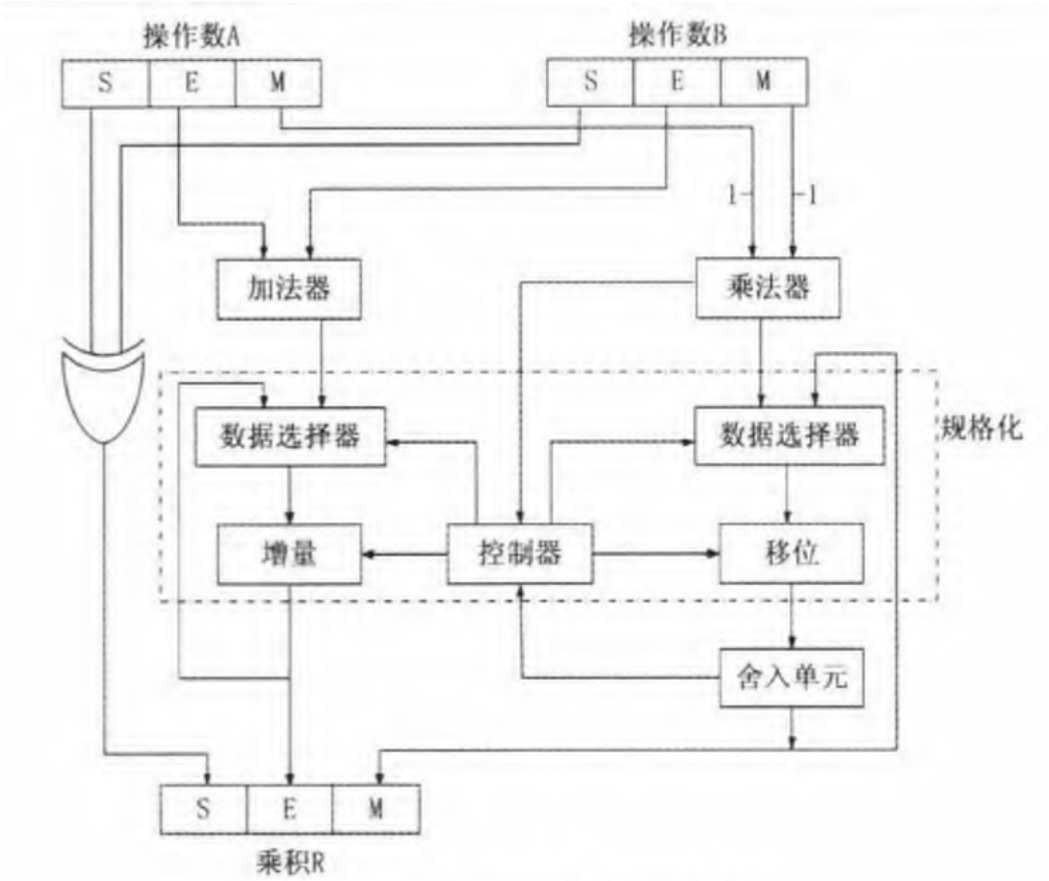
浮点数的标准表达式为：

FP $No = (-1)^S \times 2^{E-bias} \times (1.M)$

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 1，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。



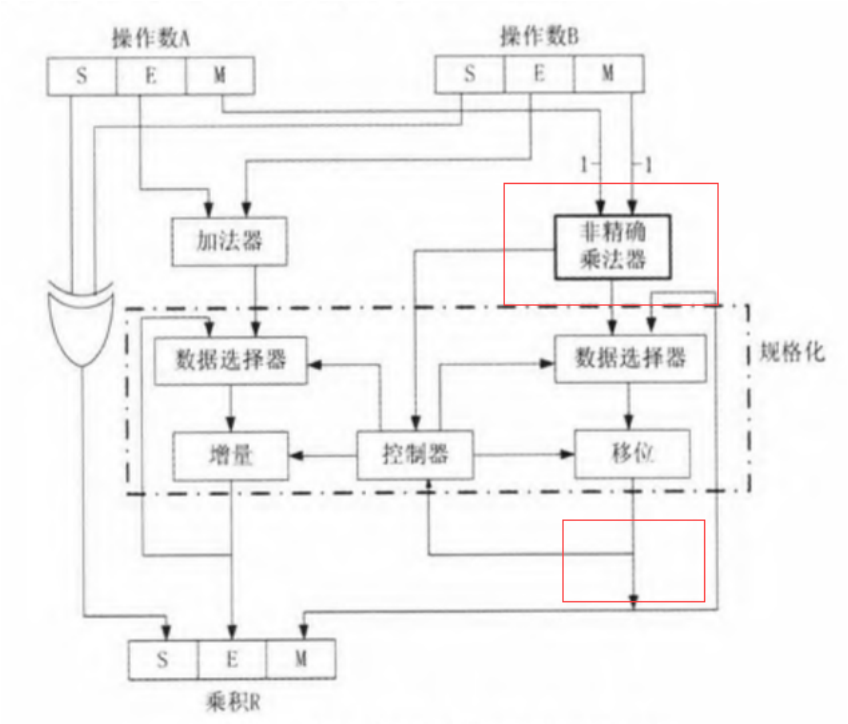
（图 2）

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]

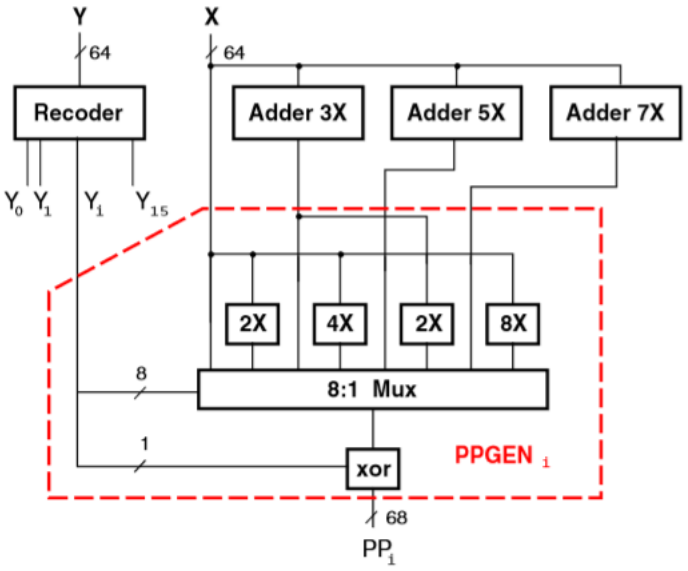
2.2 节能型多格式浮点乘法器

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。

Format	Power $[mW]$ 100 MHz	880 MHz	throughput [GFLOPS]	Power eff. [GFLOPS/W]
int64	8.90	78.32	0.88	11.24
binary64	7.20	63.36	0.88	13.89
binary32 (dual)	5.17	45.50	1.76	38.68
binary32 (single)	3.77	33.18	0.88	26.53

（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。

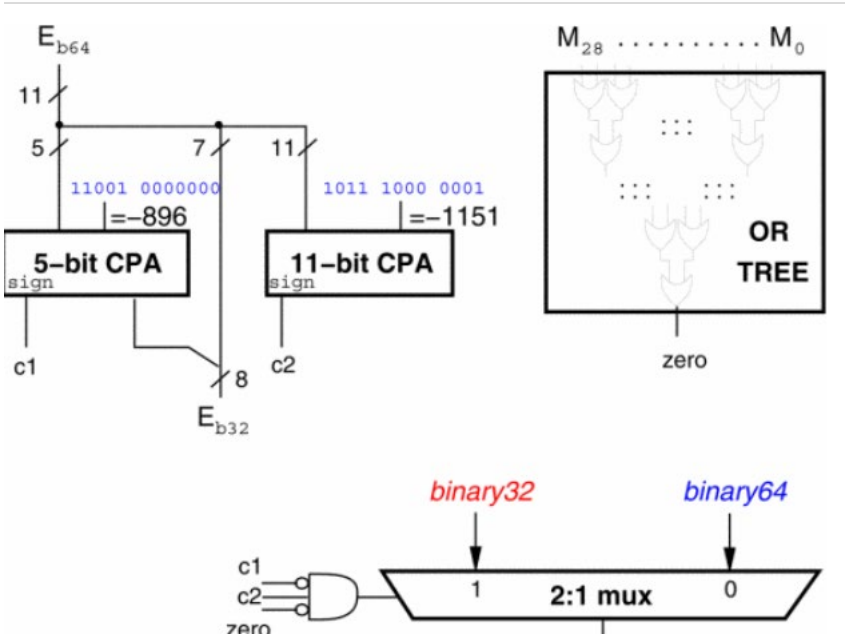
/* range checking (exponent) */
$E_{b32} = E_{b64} - B_{b64} + B_{b32} = E_{b64} - 896$ // must be positive

/* check lower bound (exponent) */
$E_{b32} - E_{max} < 0 \quad \rightarrow \quad E_{b64} - 896 - 255 = E_{b64} - 1151 < 0$

/* check significand for non-zero bits */
zero = 0;
for i=0 to 28 **do**
  zero = zero OR significand(i);
end for

if $((E_{b32} > 0) \text{AND} (E_{b64} - 1151 < 0) \text{AND} (zero = 0))$ **then**
  reduce to *binary32*
else
  keep *binary64*
end if

（图 6）



（图 7）

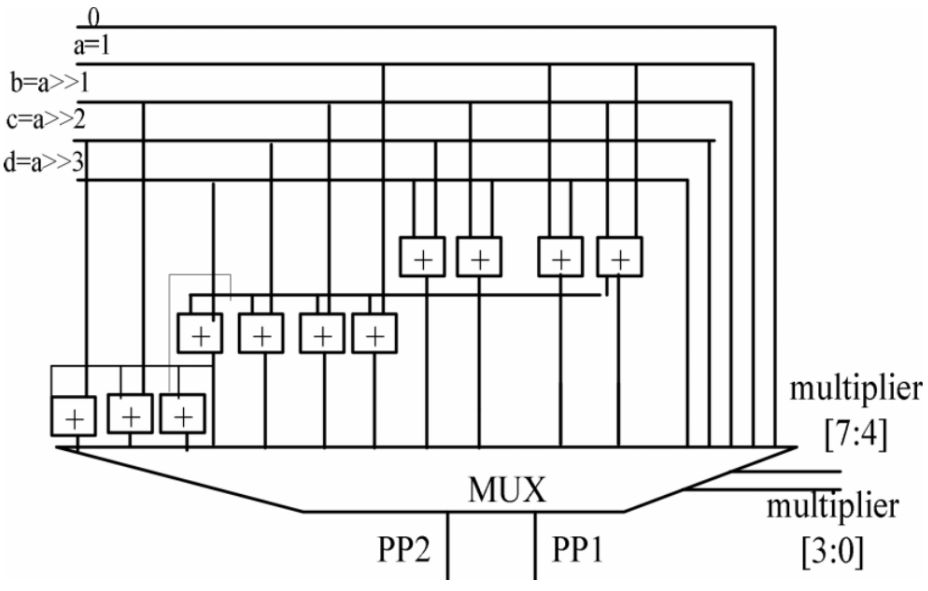
2.3 使用混合 GPP 加法的区域高效的 32 位浮点乘法器

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。



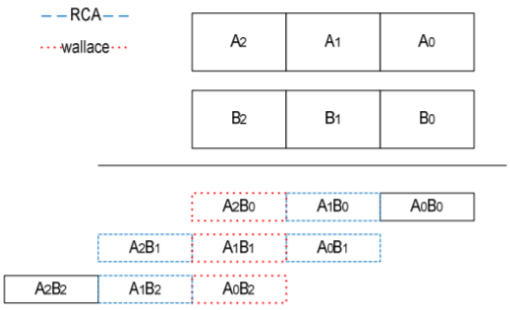
（图 8）

3PG FP multiplier	MBE FP multiplier
1. Grouping not required	1. Grouping not required
2. Do not need 2's complement	2. Do 2's complement
3. No sign extension	3. Proper sign extension needed

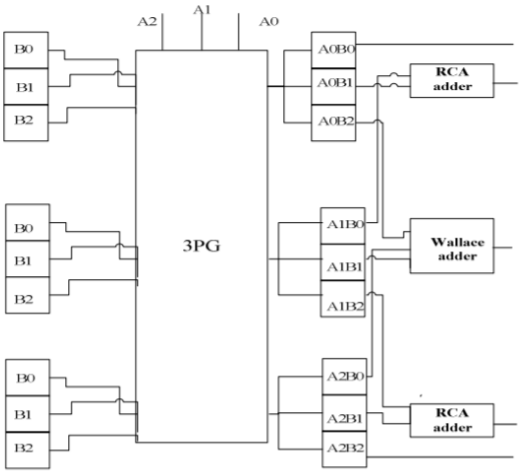
（图 9）

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.