计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(6): 1581−1593, 2025

一种可重构的单精度近似浮点乘法器设计

李鹏程[1,2] 黄立波[1,2] 陈 刚[3] 赖明澈[1,2] 邓 林[1,2] 刘 威[1,2] 杨乾明[1,2] 王永文[1,2]

[1](国防科技大学计算机学院 长沙 410073)

[2](先进微处理器芯片与系统重点实验室(国防科技大学) 长沙 410073)

[3](中山大学计算机学院 广州 510275)

(lipengcheng@nudt.edu.cn)

A Reconfigurable Single-Precision Approximate Floating-Point Multiplier Design

Li Pengcheng[1,2], Huang Libo[1,2], Chen Gang[3], Lai Mingche[1,2], Deng Lin[1,2], Liu Wei[1,2], Yang Qianming[1,2], and Wang Yongwen[1,2]

[1](*College of Computer Science and Technology*, *National University of Defense Technology*, *Changsha* 410073)

[2](*Key Laboratory of Advanced Microprocessor Chips and Systems* (*National University of Defense Technology*), *Changsha* 410073)

[3](*School of Computer Science and Engineering*, *Sun Yat-sen University*, *Guangzhou* 510275)

Abstract The rapid development of artificial intelligence (AI) and Internet of things (IoT) technologies has increased demands for the computing energy efficiency, and terminal devices are also facing great challenges in terms of hardware resource overhead. In order to solve the problem of energy efficiency, new designs of approximate computing units with low power have been widely studied. In application scenarios such as digital signal processing and image processing, there are a large number of floating-point operations. These applications consume a lot of hardware resources. However, they have certain fault tolerance characteristics and do not have to perform accurate calculations. In this paper, we propose a 32-bit single-precision reconfigurable approximate floating-point multiplier design method based on multiplication to shift addition (MTA) and asymmetric truncation. Firstly, we utilize an efficient and low-power approximate algorithm that transforms multiplication into shift-addition for partial operands. Secondly, to ensure a balance between accuracy and cost, we propose asymmetric truncation processing for the most significant bits (MSBs) and perform precise calculations on the truncated retained parts. The design achieves high precision and low power. A wide design space is generated by combining the MTA algorithm with different bit widths and varying the number of rows in the partial product array after truncation. This allows for various trade-offs between accuracy and cost. Compared with the exact floating-point multiplier, MTA5T5 has an accuracy loss of mean relative error distance (MRED) of only about 0.32%, a power reduction of 85.80%, and an area reduction of about 79.53%. The accuracy loss of MTA3T3 with lower accuracy is about 1.92%, while the power and area decrease to 90.55% and 85.80%, respectively. Ultimately, we test the application of FIR filtering and image processing, and the results show that the proposed design has significant advantages in terms of accuracy.

Key words approximate floating-point multiplier; MTA; asymmetric truncation; accuracy; cost

摘 要 人工智能(AI)和物联网(IoT)技术的迅速发展,对计算能效提出了更高的要求,终端设备在硬

收稿日期: 2025-02-26; **修回日期**: 2025-04-15

基金项目: 国家自然科学基金项目(62102433, 62272475); 湖南省自然科学基金项目(2022JJ10064)

This work was supported by the National Natural Science Foundation of China (62102433, 62272475) and the Hunan Provincial Natural Science Foundation (2022JJ10064).

通信作者: 黄立波(libohuang@nudt.edu.cn)

件资源开销方面同样面临巨大挑战. 为了应对能效问题, 新型低功耗近似计算单元的设计得到了广泛研究. 在数字信号处理和图像处理等应用场景中, 存在大量的浮点运算. 这些应用消耗了大量的硬件资源, 但它们具有一定的容错性, 没有必要进行完全精确的计算. 据此, 提出了一种基于移位近似算法 MTA (multiplication to shift addition) 和非对称截断的单精度可重构近似浮点乘法器设计方法. 首先, 采用了一种低功耗的近似算法 MTA, 将部分操作数的乘法运算转换为移位加法. 其次, 为了在精度和成本之间取得平衡, 设计了针对操作数高有效位的非对称截断处理, 并对截断后保留的部分进行精确计算. 通过采用不同位宽的 MTA 近似计算和改变截断后部分积阵列的行数, 生成了广阔的设计空间, 从而可以在精度和成本之间进行多种权衡调整. 与精确浮点乘法器相比, 所提出设计 MTA5T5 的精度损失 (MRED) 仅约为 0.32%, 功耗降低了 85.80%, 面积减少了 79.53%. 对于精度较低的 MTA3T3, 其精度损失约为 1.92%, 而功耗和面积分别降低了 90.55% 和 85.80%. 最后, 进行了 FIR 滤波和图像处理的应用测试, 结果表明所提出的设计在精度和开销方面具有显著优势.

关键词 近似浮点乘法器; MTA; 非对称截断; 精度; 开销

中图法分类号 TP331

DOI: 10.7544/issn1000-1239.202550116 **CSTR:** 32373.14.issn1000-1239.202550116

高性能计算、多媒体和人工智能大模型等新一代信息技术的迅速发展, 正以颠覆性创新姿态重塑全球科技图景, 从推动科学研究跃迁到赋能千行百业智能化转型, 这些技术突破不仅释放了巨大的社会生产力, 更催生出全新的产业生态. 然而, 技术跃升的背后潜藏着严峻的挑战: 计算机系统的总能耗正以惊人的速度增长, 以处理不断增加的信息量, 提高计算能效已成为解决能耗持续增长的关键[1-3]. 然而随着登纳德缩放定律几乎失效[4], 摩尔定律逐渐走到尽头[5], 晶体管的尺寸正接近其物理极限, 导致“暗硅”[6] 等许多关键问题产生, 通过缩小晶体管尺寸来大幅降低功耗、面积等资源开销的方式已经难以为继. 为了应对日益增长的性能和功耗挑战, 研究人员正在积极探索一种新的计算范式——近似计算[7]. 近似计算通过牺牲一定的精确度, 降低了电路的逻辑复杂度, 简化了硬件结构, 从而大幅降低了功耗和面积等开销.

在数字信号处理和图像处理等计算密集型应用中, 乘法是最典型且出现频次最高的操作[8], 完全精确的乘法计算占用了极大的功耗和延迟. 同时, 这类应用具有一定的容错性, 因而近似乘法器在近似计算领域得到越来越多研究人员的关注[9-10]. 尽管浮点运算单元在当前系统中成本较高且运行缓慢, 但乘法器中仍然存在大量浮点格式设计[11-13].

近似浮点乘法器经过多年来的发展已经取得了阶段性成果, 但仍然存在一些亟待解决的问题. 一方面, 设计的近似浮点乘法器可重构性不足以及精度和开销固定, 难以适用于不同要求的多种应用场景; 另一方面, 虽然存在一些精度可调节的近似设计, 但是难以实现精度的精细调节, 往往都是粗粒度的精度和开销控制. 此外, 精度调节单元的加入也大大增加了近似电路设计的复杂性, 成本迅速增加. 针对近似浮点乘法器的研究需求, 本文提出了一种可重构的近似浮点乘法器设计方案. 在设计阶段, 策略性地运用多种截断和 MTA 算法相结合, 充分探索整个设计空间. MTA 是一种将乘法运算转换为移位加法运算的近似计算方法, 它使用前导“1”检测和移位电路来实现运算, 但不会显著增加开销. 所采用的截断方法不同于常见的部分积列截断, 而是采用部分积列截断和行截断相结合的非对称截断方式, 能够与 MTA 相互配合实现良好的精度和开销平衡.

实验结果表明, 与精确乘法器相比, 高精度设计 MTA5T5 的精度损失仅约为 0.32%, 功耗延迟积(power-delay product, PDP)降低了 85.92%, 低精度设计 MTA-3T3 的精度损失达 1.9%, 但 PDP 降至 90.69%. 与当前先进的近似乘法器相比, 所提出的设计在精度和资源效率之间取得了更有力的平衡.

本文的主要贡献有 3 点:

1)实现了一种通过前导“1”检测和移位相加进行近似乘法计算的结构, 避免了过多的门级电路, 减少了硬件资源需求和延迟;

2)根据尾数乘法的特点, 通过对部分积阵列进行不同的截断, 并与 MTA 算法相结合, 广泛探索了近似浮点乘法器的设计空间;

3)利用仿真和综合工具验证了可重构近似浮点乘法器在精度和开销上的优势, 并进一步通过应用

测试验证了所提出设计的良好效果.

1 相关背景

1.1 单精度浮点乘法

浮点乘法运算中的尾数乘法与定点乘法几乎相同，但是浮点数的动态计算和表示范围更大，导致通常会比整数或定点计算消耗更多的资源. 根据浮点数IEEE 754标准[14]，单精度浮点数由1位符号位S(sign)、8位指数位E(exponent)和23位尾数位M(mantissa)共计32位组成，如图1所示.

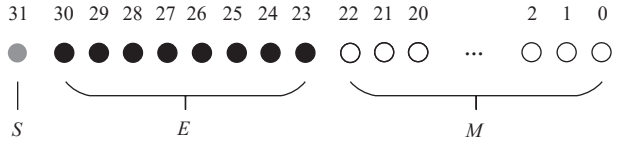


Fig. 1 Representation of 32-bit single-precision floating-point numbers

图1 32位单精度浮点数表示

S决定浮点数是正数(S=0)还是负数(S=1)；E是2的幂，作用是对浮点数加权. 由于指数可以为正数，也可以为负数，为了处理负指数的情况，实际的指数值需要加上偏置($Bias$)值作为保存在指数段中的值，在单精度浮点数中$Bias$=127. M是二进制小数，取值范围为0~1，加上隐藏位“1”构成完整的尾数部分1.M. 因此，一个规格化的单精度浮点数X可以用式(1)表示：

$$X=(-1)^S\times(1.M)\times 2^{E-127}. \tag{1}$$

2个单精度浮点数A、B相乘得到C的过程如图2所示，首先2个操作数的符号位通过异或操作得到结果的符号位，尾数部分通过相乘并舍入到1~2的范围内以产生规格化的表示，然后指数部分通过相加后减去偏置值和尾数计算的舍入情况来得到规格化的指数结果，最终将规格化的S、E、M三部分按照IEEE 754的标准拼接得到最终结果.

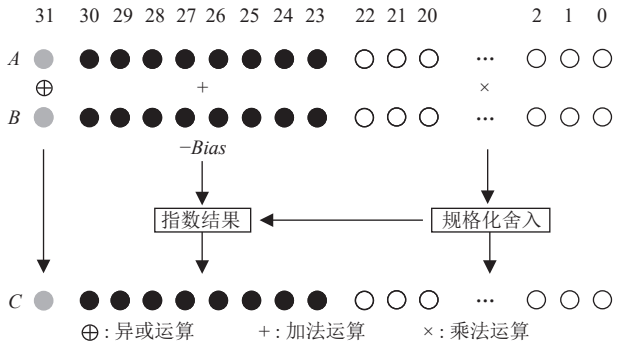


Fig. 2 Floating-point multiplication operation process

图2 浮点乘法运算过程

目前关于近似乘法器的研究可以分为定点和浮点2种设计类型，研究人员通常在乘法运算的部分积生成和累加2个阶段进行近似设计，主要的技术方法[15-19]包括直接简化乘法运算和改变乘法算法等. 在部分积生成阶段可以通过操作数分割、Booth编码、对数转化等引入近似，部分积累加阶段主要通过截断处理、调整精确和近似压缩的比例、设计使用近似单元等实现近似设计.

1.2 定点近似乘法器

在定点近似乘法设计中，Yao等人[20]通过设计预处理单元，将部分积阵列通过预处理调整为稀疏部分和稠密部分，在稀疏部分采用或门压缩以节省开销，稠密部分采用精确压缩以保证精度，但该方法针对部分积阵列行数较多的情况容易引起较大误差. Liu等人[17]设计了一种限制进位传播以实现并行计算的近似加法器，利用加法器树进行部分积的累加，使用近似加法器产生的误差向量来补偿输出中的错误，以实现更高的精度. Kumar等人[21]在部分积约简过程中加入误差校正模块，其根据输入模式生成“1”或“0”，将纠错输出作为部分积精确区域第1列的“随身行李”以提高计算精度. 然而，在上述设计中加入额外的纠错模块不可避免地增加了总体成本和电路的复杂性. 为了不使用额外的误差补偿模块，Park等人[22]提出通过在部分积生成和累加2个阶段强制生成彼此方向相反的误差来综合考虑，首先改进Booth编码使其在部分积生成阶段产生更大的单向误差，并降低硬件成本，然后在部分积累加阶段设计误差方向相反的近似压缩器以补偿来自Booth编码器的误差，从而在不使用复杂的误差恢复电路的情况下将内部误差自然平衡为零. Sabetzadeh等人[23]将部分积阵列的一半低有效位作为常数. 这种设计虽然实现了非常高的硬件效率，但也导致了较高的误差，特别是在输入数据较小的情况下. Haider等人[24]提出了一种称为解码器减少近似(decoder reduction approximation, DRA)的新近似方案，该方案使用较少的N/4解码器来代替传统Booth乘法器中的N/2解码器，从而减少了部分积行数的累加. 但由于该方案的技术类似于部分积阵列行的截断，因此在低位宽乘法器中会导致更大的精度损失. 此外，DRA方案的编码过滤逻辑相对复杂，导致资源开销增加.

1.3 浮点近似乘法器

相比于定点乘法器，浮点乘法由于其动态范围更广，在数字信号处理、图像处理和深度学习等应用场景中有广泛的应用[25-26]. 在浮点乘法器中，由于尾

数乘法的功耗、面积等开销占比巨大,且能较易容忍误差,因而有关浮点乘法器的近似设计主要体现在尾数乘法部分. Zhang 等人[27]通过设置精度控制单元以计算 Bfloat16 浮点乘积的指数部分,然后将其转换为 posit 编码格式,并生成掩码信号来控制尾数乘法器的近似程度. 在尾数乘法的部分积阵列中,如果掩码位为 1,则相应位置将使用压缩器累加部分积,否则将禁用该部分积列的累加计算,从而实现部分积阵列的截断处理. Saggese 等人[28]通过动态截断部分积阵列,并使用简单的舍入来减少误差,设计了一种精度可调的浮点乘法器. Saadat 等人[29]提出了一种误差可配置近似乘法器,通过在对数相加阶段中加入一个固定修正项来消除误差偏差并保持了较低的硬件复杂度. 类似地, Imani 等人[30]提出了另一种可配置的低开销近似浮点乘法器,该乘法器使用自适应选择器和调谐电路评估输入操作数,从而动态识别可能导致的误差,如果错误被认为是可接受的,自适应选择器电路更新指数值,并选择一个输入操作数作为输出尾数,否则切换到精确模式进行计算. Jha 等人[31]提出了与 Imani 等人不同的方法,不再需要单独的精确计算单元,只使用移位和加法运算,具有运行时的可配置性,但是该设计本质上是部分积阵列的行累加,精度和开销均无法达到较好的平衡. Towhidy 等人[32]提出了基于伪 Booth 编码的迭代近似浮点乘法器,该乘法器可以通过编码器基数、截断的位宽以及是否迭代调整精度,但随着精度的提高,开销急剧增加.

2 近似乘法器设计

2.1 动 机

从上述相关工作中可知,近似乘法器设计的主要技术思想可以总结为 2 点:一是确保高权重部分进行精确计算,以保证整体设计的准确性;二是尽可能在低权重部分进行近似计算,以降低硬件成本. 目前许多近似设计存在高权重位部分精确计算冗余或不足,以及低权重位近似计算成本过高的问题,难以实现精度与硬件资源之间的良好平衡. 为解决这一问题,本文提出了一种基于 MTA 和非对称截断的高精度低功耗近似浮点乘法器设计方案. 该方案主要对 2 个操作数的高有效位部分进行非对称截断,而其他位则采用低开销的 MTA 近似算法进行计算. 非对称截断解决了高有效位精确计算冗余的问题,而 MTA 近似算法解决了低有效位计算成本过高的问题. 通过 MTA 与截断处理相结合以及利用浮点乘法的特点等进行近似设计,在精度和资源开销之间实现了良好的平衡.

截断处理和 MTA 算法的多种不同组合使得近似浮点乘法器具有较强的可重构性,形成较大的设计空间,所提出近似乘法器的简化框图如图 3 所示. 由于浮点乘法的特性,我们在指数位采用精确计算,对尾数相乘采用近似计算. 当对 2 个操作数的尾数进行截断处理和 MTA 近似计算后,经过位宽扩展和加法器进行最后的规格化. 在 2.2 节中,我们将详细描述所提出的乘数器设计方法,并探讨其多种可重构设计的不同效果.

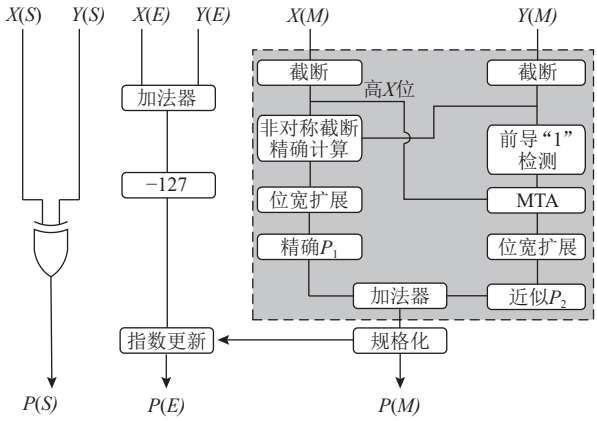


Fig. 3 Block diagram of simplified design for approximate multiplier

图 3 近似乘法器的简要设计框图

2.2 MTA 近似算法

MTA 近似算法是一种将乘法操作转变为移位加法操作的近似计算方法. 从单精度浮点数的表达式(式(1))可以看出,尾数乘法计算中的最高有效位始终为“1”,根据这一特征,提出 MTA 近似算法. 假设存在 2 个最高有效位均为“1”的 4 位二进制数 A 和 B,分别可以由式(2)和式(3)表示.

$$A = 1xxx = 1\times 2^3 + m, \tag{2}$$

$$B = 1xxx = 1\times 2^3 + n. \tag{3}$$

这 2 个二进制数相乘的结果 T 如式(4)所示.

$$\begin{aligned} T = A\times B &= (1\times 2^3 + m)\times(1\times 2^3 + n) = \\ & 1\times 2^6 + (m+n)\times 2^3 + m\times n. \end{aligned} \tag{4}$$

从式(2)~(4)可以看出,最高有效位为“1”的 2 个操作数在相乘的过程中可以转变为部分加法操作,因此,根据这一特点提出了将乘法操作近似为加法运算的 MTA 近似算法. 4 位 MTA 近似算法(MTA4)的具体计算过程如图 4 所示, A 与 B 相乘转变为 A 与 B 相加. 在 2 个操作数 A 和 B 相加的过程中,只有 m

和 n 参与求和运算，并得到和的低 3 位结果 *sum*. 中间结果的最高有效位始终为“1”，并不需要计算得到. 当 m 与 n 和的最高位不产生进位时，通过将中间结果左移 3 位，然后用 m 或 n 来填充低 3 位以获得最终结果. $A\times B$的最终结果被近似为式(5).

$$T_{\text{appro}}\approx A\times B\approx 1\times 2^6+(m+n)\times 2^3+m. \tag{5}$$

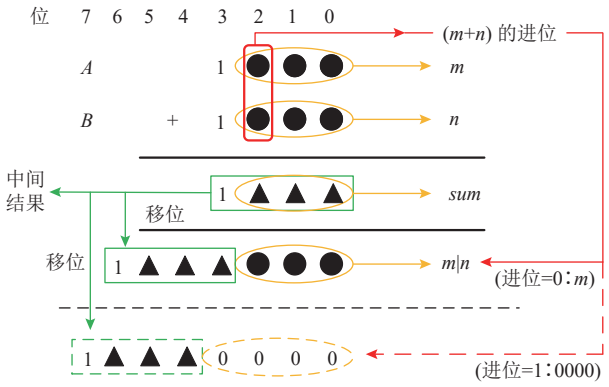


Fig. 4 Approximate computing procedure of MTA4

图 4 MTA4 近似计算过程

此时，MTA4 近似算法的误差值计算如式(6)所示.

$$Err=A\times B-T_{\text{appro}}=m\times n-m=(m-1)\times n. \tag{6}$$

当$m+n$的最高有效位产生进位“1”时，$A\times B$的最终结果被近似为$(m+n)\times 2^4$，即图 4 中的红色虚线部分所示，通过$m+n$获得中间结果，然后将中间结果向左移 4 位并用“0000”补充低 4 位以获得最终结果. 此时，MTA4 近似计算的误差如式(7)所示.

$$\begin{aligned}Err=&A\times B-T_{\text{appro}}=1\times 2^6+(m+n)\times 2^3+m\times n-\\&(m+n)\times 2^4=1\times 2^6-(m+n)\times 2^3+m\times n.\end{aligned} \tag{7}$$

MTA4 的电路设计较为精简，如图 5 所示，2 个 3 位的操作数 m 和 n 输入 4 位加法器中，由加法器分别输出$m+n$的进位以及低 3 位 *sum*，*sum* 通过位拼接形成 2 个多路选择器 MUX 的输入，进位信号通过 MUX 控制选择输出近似结果.

该方法有效地将复杂的乘法运算转化为简单的移位加法运算，降低了资源消耗，适合乘法中低权重部分的近似计算. 当操作数 A 和 B 的位宽发生变化时，上述近似算法依然适用. 在本文中，我们主要对 3~7 位 MTA 算法进行近似设计，从算法设计以及实际效果来看，3 位及以下的 MTA 近似算法具有较低的准确性，并且需要经历更多的前导“1”检测，延迟有所增加，位宽大于 6 的 MTA 近似算法提高精度的幅度较小，但是硬件资源开销增加相对明显，难以在精度和成本之间取得良好的平衡.

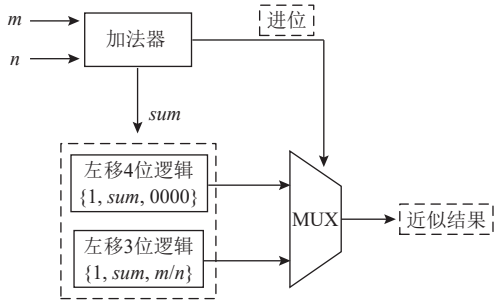


Fig. 5 Approximate module design of MTA4

图 5 MTA4 近似模块设计

2.3 非对称截断

部分积阵列的截断处理在近似设计中是一种常用的方法，大多数截断处理采用的是截断部分积阵列的行或者列的方法，但是为了能够与 MTA 近似算法相结合，实现良好的精度和开销平衡，我们采用了一种能够结合 MTA 的截断处理方法，选择在部分积阵列中的行和列同时展开截断操作，大幅降低功耗、面积等开销. 除此之外，相比定点乘法场景，非对称截断处理依然更加适合浮点乘法计算，这主要是由浮点乘法的特性决定，浮点尾数的隐藏位“1”导致高位截断误差呈指数放大，而定点乘法误差线性累积.

在非对称截断处理中，我们采用截断操作数的方式来实现截断处理，而非截断部分积阵列的方式，有效降低了部分积计算所带来的资源开销. 如图 6 所示，X、Y 两个操作数相乘，将操作数 X 截断并保留高 10 位，将操作数 Y 截断并保留高 5 位. 假设 X、Y 的最高有效位均为 1，当与最高有效位相乘时，可以不使用与门直接得到部分积的计算结果(图 6 中灰色阴影部分). 然后对截断的操作数在部分积生成过程中进一步截断得到 T5 部分积. 当 Y 被进一步截断，只保留高 3 位、高 4 位时，生成的部分积分别称为 T3 和 T4. 当部分积阵列为 T3 时，部分积采用全加器并行计算；当部分积阵列为 T4 时，部分积采用精确 4-2 压缩器并行计算；当部分积阵列为 T5 时，阴影上方的 4 行部

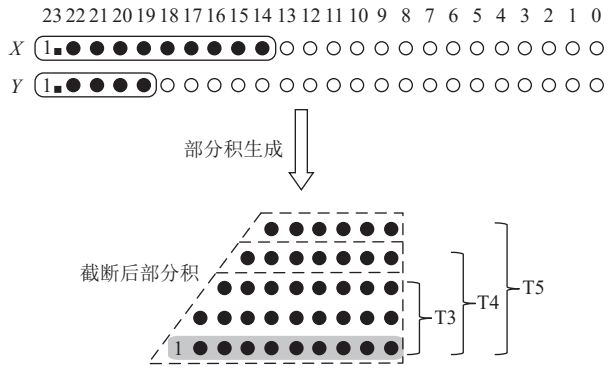


Fig. 6 Asymmetric truncation process

图 6 非对称截断处理

分积采用精确 4-2 压缩器并行计算，然后将结果与阴影部分积相加得到整个截断操作的结果.

通过非对称截断处理，生成的部分积有利于使用精确加法器或者 4-2 压缩器等标准压缩部件进行并行压缩，提高了计算性能，并大幅降低了功耗、延迟和面积等. 但是，过多的截断操作也带来了严重的精度损失，当配合 MTA 近似算法时，这种精度损失会得到明显降低，从而提高精度.

2.4 整体设计方案

在单精度浮点计算中，由于指数位的计算对精度影响过大，一般在尾数计算中引入近似计算. 在 23 位尾数中，高有效位对精度的影响更重要，当它们改变时会导致精度大幅波动，因此，在设计尾数近似计算时，尽可能保证高权重位执行精确计算，而低权重位则被近似处理.

在 $X\times Y$ 的过程中，首先固定 Y 的最高有效位数，这里选择 5 位最高有效位. 操作数 X 可以从第 23 位开始逐渐截断，这里选择截断低 14 位，X 和 Y 被截断后保留的部分执行精确乘法计算，在截断处理后，再运用 MTA 算法执行近似计算. 如图 7 所示，采用 MTA4 近似算法，选择 X 的高 4 位，其最高有效位固定为“1”. Y 的高 5 位已经执行精确计算，其余低位(18~3)采用前导“1”检测，当检测到第一个有效位“1”时，同时选择其后的 3 位，构成最高有效位为“1”的 4 位操作数. 当 18~3 位经过前导“1”检测均为 0 后，由于 2~0 位在整体所占重要性较低，因此，采用调优补偿的方式，将 MTA 的结果近似取值为 24’b011.

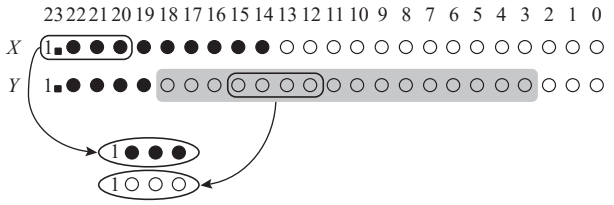


Fig. 7 Leading-one-detection of MTA4

图 7 MTA4 的前导“1”检测

以 MTA4T5 为例，整体计算过程如图 8 所示. 首先选择 X 的 23~14 位和 Y 的 23~19 位执行精确计算，生成精确结果 P_1(47~37 位). 然后，根据 MTA4 近似算法，选择 X 的 23~20 位高有效位，在 Y 的 18~3 位利用前导“1”检测选择最高有效位为“1”的 4 位操作数，并执行 MTA4 近似计算，得到近似结果 P_2. 图 8 中的 P_1 和 P_2 只扩展到 47~23 位，这是因为单精度浮点数的尾数部分只保留 23 位，在规格化的过程中会对部分积阵列的低权重位所产生的结果进行舍入，因此为了进一步降低开销，这里不再进行尾数结果低权重位的舍入，而是将计算过程中的 22~0 位直接截断，虽然是大比特的直接截断，但是在精度损失较低的前提下，大幅降低了硬件成本. 最后将 P_1 和 P_2 相加，得到尾数乘法的最终结果 P_3.

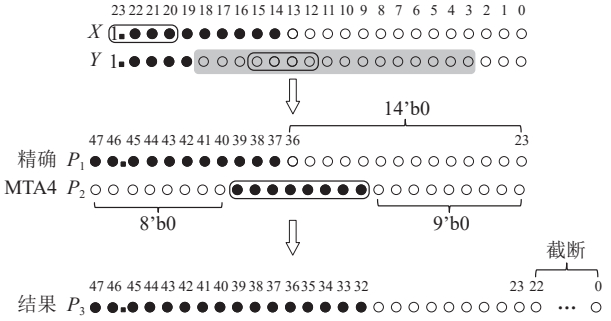


Fig. 8 Mantissa approximate calculation process of MTA4T5

图 8 MTA4T5 的尾数近似计算过程

2.5 设计空间探索

本文提出的 MTA 近似算法和非对称截断处理具有良好的可重构性，不同位宽的 MTA 近似算法和非对称截断处理相结合能形成广泛的设计空间，进而实现不同的精度和硬件资源开销之间的组合，探索不同的可重构设计有助于寻找最佳设计方案，也为不同精度和开销需求的应用提供了可选择性.

在 MTA 近似算法中，当位宽小于 3 时，所提供的精度较低，随着位宽的不断增加，精度逐渐提高，但硬件开销也相应增加. 当 MTA 的位宽大于 6 时，精度的提高相对缓慢，而功耗却仍持续线性增长. 此外，非对称截断也具备一定的可配置性，本文固定 X 的高 10 位，探索了 Y 的不同截断，也可一定程度上固定 Y 的位宽，调整 X 的截断作为变量进行研究，而两者的可操作性和设计形式类似，故不再单独作为空间探索的设计. 因此，在探索设计空间时，本文主要应用和测试了 MTA3, MTA4 和 MTA5 三种近似算法，并相应采用了 T3, T4 和 T5 三种不同的非对称截断处理，产生了 9 种不同的近似设计，分别为 MTA3T3, MTA3T4, MTA3T5, MTA4T3, MTA4T4, MTA4T5, MTA5T3, MTA5T4, MTA5T5. 同时为了进一步研究高位 MTA 近似算法对精度和开销的影响，统一在 T5 截断处理下设计了 MTA6T5 和 MTA7T5 两种高精度近似浮点乘法器.

3 性能评估与分析

本节从误差特性和硬件指标 2 个方面对提出的

近似浮点乘法器进行评估，包括提出的设计和现有近似方案，共实现了 20 种近似浮点乘法器.

3.1 误差特性

为了评估近似乘法器的有效性和量化误差，研究人员制定了多种方法和指标，Liang 等人[33]提出的平均相对误差距离(mean of relative error distance, *MRED*)和归一化平均误差距离(normalized mean error distance, *NMED*)已被广泛用于评估误差特征. 在 n 位近似乘法中，*MRED* 统计每组输出的近似结果与精确结果之间的相对偏差；*NMED* 常用于比较不同位宽或规模的近似设计的误差幅度. 以上精度评估指标可用式(8)(9)进行计算.

$$MRED=\frac{1}{2^{2n}}\sum_{i=1}^{2^{2n}}\frac{|T_i-T_i'|}{T_i}, \tag{8}$$

$$NMED=\frac{1}{2^{2n}}\sum_{i=1}^{2^{2n}}\frac{|T_i-T_i'|}{T_{\max}}, \tag{9}$$

其中，T_i 表示精确输出，T_i' 表示近似输出，$T_{\max}$ 表示最大精确输出.

所有的近似浮点乘法器用 Verilog 硬件描述语言进行编写，使用 Verilator 仿真平台进行仿真实验，验证正确性和得到近似结果，而后利用 C 语言程序计算所有近似乘法器的精确度. 由于单精度浮点数所表示的数据范围较大，因此采用随机生成 100 万组输入组合的方式计算精度评估指标，如表 1 所示.

从表 1 可以看出，与其他近似方案相比，所设计的 *MRED* 在 2.35×10^{-3}~1.92×10^{-2} 之间，具有较小的精度损失. 其中，MTA5T4, MTA4T5~MTA7T5 在各项精度指标上均具有显著优势.

Table 1 Error Comparison of Approximate Multipliers

表 1 近似乘法器误差对比

乘法器	*MRED*	*NMED*	乘法器	*MRED*	*NMED*
MTA3T3	19.175×10^{-3}	66.69×10^{-4}	MTA7T5	2.350×10^{-3}	7.73×10^{-4}
MTA4T3	11.429×10^{-3}	38.64×10^{-4}	HPC[19]	3.972×10^{-3}	13.01×10^{-4}
MTA5T3	7.767×10^{-3}	26.24×10^{-4}	LPC[19]	17.042×10^{-3}	50.32×10^{-4}
MTA3T4	8.776×10^{-3}	29.36×10^{-4}	CFPM[28]	7.898×10^{-3}	25.29×10^{-4}
MTA4T4	5.700×10^{-3}	18.60×10^{-4}	MBM[29]	30.720×10^{-3}	80.58×10^{-4}
MTA5T4	3.609×10^{-3}	11.47×10^{-4}	CFPU[30]	15.103×10^{-3}	45.62×10^{-4}
MTA3T5	5.607×10^{-3}	18.81×10^{-4}	PB4W16[32]	33.558×10^{-3}	85.16×10^{-4}
MTA4T5	3.865×10^{-3}	12.92×10^{-4}	IPB4W16[32]	20.004×10^{-3}	52.11×10^{-4}
MTA5T5	3.187×10^{-3}	10.56×10^{-4}	PB8W16[32]	10.501×10^{-3}	36.05×10^{-4}
MTA6T5	2.582×10^{-3}	8.53×10^{-4}	IPB8W16[32]	8.200×10^{-3}	30.65×10^{-4}

3.2 硬件开销分析

硬件资源开销分析与评估实验通过 Verilog 语言描述乘法器硬件结构，得到相应的 RTL 代码，然后统一利用 EDA 工具在 28 nm 典型工艺下进行逻辑综合. 首先，我们分析和评估了所提出设计的主要电路功能模块的功耗(Power)和面积(Area)开销，如表 2 所示，以 MTA4T5 为代表，将 32 位精确浮点乘法器(32-bit exact floating-point multiplier, EX-FP32)作为对照.

从表 2 可以看出，所提出设计的 MTA 和前导“1”检测模块的功耗和面积均较小，在总开销中的占比接近 10%，主要的硬件开销来源于精确计算过程和尾数的规格化，实现了降低近似计算成本的目标. 与精确浮点乘法模块相比，尾数乘法模块的功耗下降了 91.06%，面积下降了 86.87%；规格化模块功耗下降了 55.89%，面积下降了 25.72%.

除了关注各模块的开销外，整体设计的硬件开销情况是评估的重点，为此，我们在同等条件下实现了包括所提出的设计及其他近似浮点乘法器在内的 20 种设计，对比了功耗、面积、关键路径延时(Delay)、功耗延时积(PDP)和面积延时积(area-delay product, ADP)等 5 项指标，如表 3 所示. PDP、ADP 能够分别

Table 2 Main Module Overhead of Approximate Multipliers

表 2 近似乘法器主要模块开销

模块	Power(μW)			Area(μm^2)		
	EX-FP32	MTA4T5	比率	EX-FP32	MTA4T5	比率
MTA4	n/a	22.2	n/a	n/a	25.9	n/a
前导检测	n/a	10.6	n/a	n/a	76.4	n/a
尾数乘法	3 111	278	91.06%↓	4 509.6	592.2	86.87%↓
规格化	180	79.4	55.89%↓	278	206.5	25.72%↓
精确 T5	n/a	72.5	n/a	n/a	176.1	n/a

注：“n/a”表示不适用对比，“↓”表示相比 EX-FP32 对应模块下降的百分比.

Table 3 Comparison of Hardware Parameters of Approximate Multipliers

表 3 近似乘法器硬件参数对比

乘法器	Power/mW	Area/μm^2	Delay/ns	PDP(pJ)	ADP/(ns×μm^2)	乘法器	Power/mW	Area/μm^2	Delay/ns	PDP/pJ	ADP/(ns×μm^2)
EX-FP32	3.469 5	4 968.43	4.77	16.549 5	23 699.4	MTA7T5	0.578 0	1 129.13	4.73	2.733 9	5 340.8
MTA3T3	0.327 7	705.43	4.70	1.540 2	3 315.5	HPC[19]	0.617 9	1 172.47	4.73	2.922 7	5 545.8
MTA4T3	0.385 3	879.98	4.74	1.826 3	4 171.1	LPC[19]	0.456 9	891.91	4.75	2.170 3	4 236.6
MTA5T3	0.430 7	970.87	4.73	2.037	4 592.2	CFPM[28]	0.502 6	981.29	4.72	2.372 3	4 631.7
MTA3T4	0.403 9	867.05	4.75	1.918 5	4 118.5	MBM[29]	0.408 5	1 098.33	4.73	1.932 2	5 195.1
MTA4T4	0.423 0	906.70	4.72	1.996 6	4 279.6	CFPU[29]	2.402 8	5 166.01	3.32	7.977 3	17 151.2
MTA5T4	0.459 3	962.47	4.72	2.167 9	4 542.9	PB4W16[32]	0.420 5	1 135.68	4.73	1.989 0	5 371.8
MTA3T5	0.430 8	928.37	4.76	2.050 6	4 419.0	IPB4W16[32]	0.803 7	1 846.49	4.74	3.809 5	8 752.4
MTA4T5	0.453 5	948.19	4.72	2.140 5	4 475.5	PB8W16[32]	0.633 6	1 360.80	4.73	2.996 9	6 436.6
MTA5T5	0.492 7	1 016.90	4.73	2.330 5	4 809.9	IPB8W16[32]	1.141 9	2 730.17	4.74	5.412 6	12 941.0
MTA6T5	0.537 4	1 077.89	4.73	2.541 9	5 098.4						

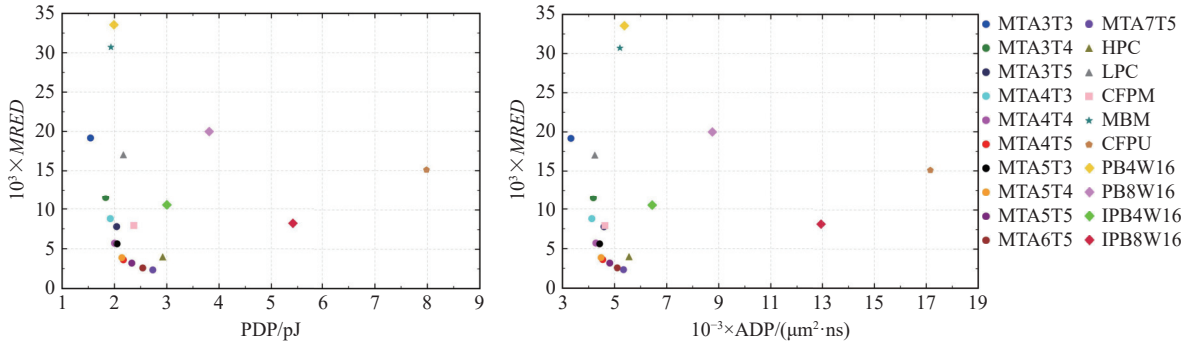
从功耗和面积与延时方面考虑近似乘法器的硬件资源开销情况，增加对比合理性.

从表 3 可以看出，与 EX-FP32 相比，MTA 系列乘法器功耗下降 83.34%~90.55%，面积下降 77.27%~85.80%，PDP 下降 83.48%~90.69%，ADP 下降 77.46%~86.01%. 与其他近似方案相比，MTA3T3，MTA4T3 和 MTA3T4 具有最低的功耗和面积开销. 在所提出的 MTA 系列乘法器设计中，MTA3T3 具有最小的开销，MTA7T5 具有最大的开销.

3.3 综合评估

综合考虑误差特性和硬件资源开销，分析 PDP 和 ADP 分别与 *MRED* 的关系，从精度与开销的权衡角度，进一步比较各类近似乘法器的优劣. 不同近似乘法器的硬件指标与 *MRED* 的关系如图 9 所示.

从图 9 中可以看出，在硬件开销几乎相同的情况下，MTA 系列乘法器具有更高的精度，如 MTA4T5 和 MTA5T4，硬件开销与 LPC[19] 相近，但精度损失却低 3~4 倍. 另外，本文所提出的大多数近似设计的值位于坐标图的左下角，相比其他近似设计更靠近原点，位于帕累托前沿，表明这些近似设计的精度更高且硬件开销更低. 其中，低位 MTA 算法相比其他设计虽然硬件开销较低，但是精度损失更大，远离帕累托前沿. 随着 MTA 算法位宽的增加，当位宽大于 6 时，精度的提升趋于平缓，综合来看 MTA4T5，MTA5T4 和 MTA5T5 在精度和成本之间实现了更好的折衷，使它们成为近似浮点乘法器设计的良好选择.



(a) *MRED* vs PDP (b) *MRED* vs ADP

Fig. 9 Relationship diagram of hardware parameters and *MRED* for different approximate multipliers

图 9 不同近似乘法器的硬件参数与 *MRED* 的关系图

4 应用测试研究

为了测试和验证本文所提出近似设计的效果，我们进行了有限冲激响应(finite impulse response, FIR)滤波和图像处理的应用实验，通过调用不同的近似浮点乘法器进行相应的仿真测试，并得到相应的测试结果.

4.1 FIR 滤波

FIR 滤波器是一种非常稳定的系统，其滤波精度高，可随时修改，不需要考虑阻抗等问题，可以实现特殊要求的幅频特性，在数字信号处理和信号系统等场景中有着广泛的应用.

实验设置中，本文使用等纹波设计法设计了一个 15 阶的低通滤波器. 设置采样频率 Fs 为 100 MHz，通带频率 Fpass 为 1 MHz，阻带频率 Fstop 为 15 MHz，滤波器的输入信号由 2 MHz 的正弦波信号和作为噪声的 30 MHz 的正弦波混合而成. 不同的近似乘法器分别被应用于 FIR 滤波操作，信噪比（signal-to-noise ratio, SNR）和均方误差（mean square error, MSE）作为评估滤波效果的指标.

我们随机生成了 1 024 组测试数据，并将这些数据作为输入激励，送入由 Verilator 搭建的仿真环境中. 通过更换不同的近似乘法器得到一系列的测试结果. 最后，对这些结果进行了统计分析，表 4 列出了不同近似乘法器的 SNR 和 MSE. 为了便于结合硬件资源开销进行分析，将不同近似乘法器的 PDP 和 ADP 分别与 SNR 的关系表示在图 10 中，从中可以看出所提出的大部分设计位于坐标图的左上角，靠近帕累托前沿，尤其是 MTA4T5, MTA5T4 和 MTA5T5 等相比其他设计更靠近左上角，即这些近似乘法器具有更优的精度和硬件开销之间的权衡.

Table 4 SNR and MSE of FIR Filtering

表 4 FIR 滤波的 SNR 和 MSE

乘法器	SNR/dB	MSE	乘法器	SNR/dB	MSE
MTA3T3	33.41	208.33×10^{-6}	MTA7T5	49.74	4.86×10^{-6}
MTA4T3	40.39	41.75×10^{-6}	HPC[19]	46.13	11.15×10^{-6}
MTA5T3	41.52	32.19×10^{-6}	LPC[19]	35.58	126.49×10^{-6}
MTA3T4	41.77	30.42×10^{-6}	CFPM[28]	41.40	33.13×10^{-6}
MTA4T4	44.96	14.60×10^{-6}	MBM[29]	30.17	398.62×10^{-6}
MTA5T4	47.50	8.14×10^{-6}	CFPU[30]	37.24	87.72×10^{-6}
MTA3T5	42.32	26.77×10^{-6}	PB4W16[32]	37.58	79.78×10^{-6}
MTA4T5	46.46	10.33×10^{-6}	IPB4W16[32]	36.86	94.16×10^{-6}
MTA5T5	47.39	8.33×10^{-6}	PB8W16[32]	42.24	26.85×10^{-6}
MTA6T5	48.76	6.08×10^{-6}	IPB8W16[32]	43.83	18.94×10^{-6}

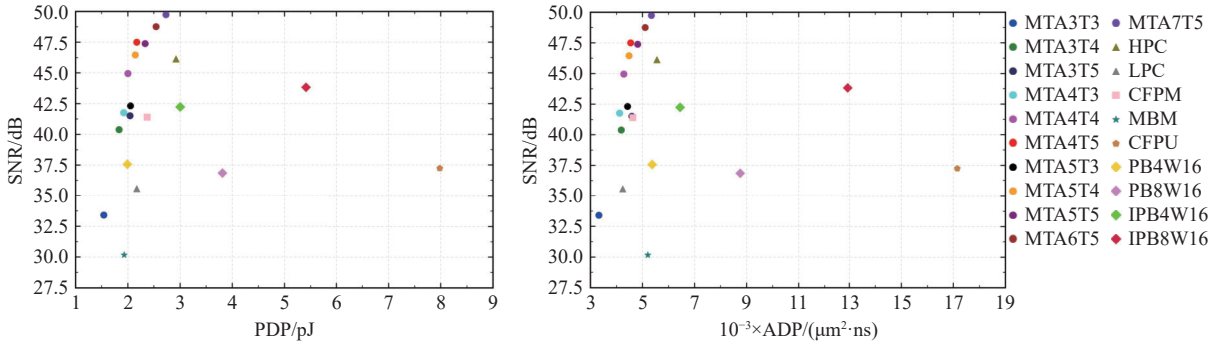


Fig. 10 Relationship diagram of hardware parameters and SNR for different approximate multipliers

图 10 不同近似乘法器的硬件参数与 SNR 的关系图

4.2 图像处理

图像处理有较广的动态存储和计算范围，需要用到大量的浮点运算，并且由于人眼难以识别图像的精细变化等原因，使得近似计算可以得到非常好的应用. 在图像处理测试中，峰值信噪比（peak signal-to-noise ratio, PSNR）和结构相似性（structural similarity, SSIM）是广泛用于评价图像质量和相似程度的 2 种客观指标，经过处理后图像的 PSNR 反映了处理程序是否令人满意. PSNR 接近 50 dB 代表处理后的图像仅有些许非常小的误差，大于 30 dB 代表图像质量高，低于 30 dB 人眼就可以察觉出图像的异常.

为了更清楚地了解各种近似方案在应用中的精度差异，我们分别选择图像自乘和图像融合进行应用测试. 图像自乘能够显著提高图像的对比度，放大近似计算引入的误差. 同时，为了防止单个图像的处理结果出现随机变化，在图像自乘中采用了 5 张不同的 RGB 彩色图像进行实验，如图 11 所示. 实验得到的 PSNR 和 SSIM 如表 5 所示. 为了便于分析精度和硬件开销之间的关系，图 12 展示了不同近似乘法器的硬件指标与 PSNR 的关系.



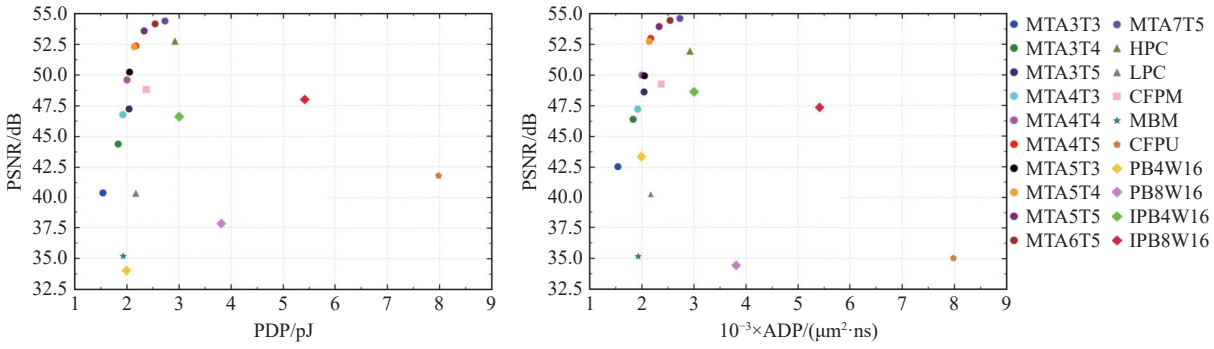
Fig. 11 Related test images of image self-multiplication

图 11 图像自乘的相关测试图像

Table 5 PSNR and SSIM of Image Self-Multiplication

表 5 图像自乘的 PSNR 和 SSIM

乘法器	Airplane		Baboon		Barbara		Female		Peppers		平均值	
	PSNR/dB	SSIM/%	PSNR/dB	SSIM/%	PSNR/dB	SSIM/%	PSNR/dB	SSIM/%	PSNR/dB	SSIM/%	PSNR/dB	SSIM/%
MTA3T3	37.127 8	99.89	41.309 8	99.93	42.141 1	99.92	40.370 2	99.89	41.134 7	99.92	40.416 7	99.91
MTA4T3	41.233 0	99.96	45.194 7	99.97	45.865 7	99.97	44.659 8	99.96	45.019 0	99.97	44.394 4	99.97
MTA5T3	44.286 5	99.98	48.097 3	99.99	48.700 1	99.98	47.194 4	99.97	47.971 3	99.98	47.249 9	99.98
MTA3T4	43.104 3	99.97	47.279 9	99.98	48.236 8	99.98	48.027 8	99.97	47.276 5	99.98	46.785 1	99.98
MTA4T4	45.705 7	99.98	49.989 8	99.99	50.922 5	99.99	51.358 2	99.99	50.069 8	99.99	49.609 2	99.99
MTA5T4	49.731 4	99.99	52.844 5	99.99	53.664 2	100.00	52.636 0	99.99	53.036 9	99.99	52.382 6	99.99
MTA3T5	47.016 6	99.99	50.569 2	99.99	51.498 0	99.99	51.446 2	99.99	50.710 0	99.99	50.248 0	99.99
MTA4T5	49.429 2	99.99	52.784 2	99.99	53.749 8	100.00	52.671 5	99.99	52.981 1	99.99	52.323 2	99.99
MTA5T5	50.815 9	99.99	54.388 2	100.00	55.064 2	100.00	53.297 5	99.99	54.456 3	100.00	53.604 4	100.00
MTA6T5	52.440 8	100.00	54.715 4	100.00	55.416 7	100.00	53.361 5	99.99	54.905 7	100.00	54.168 0	100.00
MTA7T5	52.511 0	100.00	54.963 2	100.00	55.635 9	100.00	53.792 9	99.99	55.082 9	100.00	54.397 2	100.00
HPC[19]	49.054 9	99.99	53.210 0	100.00	54.520 4	100.00	52.904 3	99.99	54.099 0	100.00	52.757 7	100.00
LPC[19]	35.041 8	99.81	40.184 8	99.91	42.137 3	99.92	43.312 8	99.90	41.214 3	99.92	40.378 2	99.89
CFPM[28]	45.323 1	99.98	48.967 5	99.99	50.460 1	99.99	50.204 5	99.98	49.247 0	99.99	48.840 4	99.99
MBM[29]	30.720 3	99.47	36.175 8	99.70	35.523 0	99.59	40.115 2	99.80	33.434 0	99.51	35.193 2	99.61
CFPU[30]	37.247 8	99.91	41.913 2	99.93	43.423 5	99.94	43.777 5	99.96	42.526 9	99.94	41.777 8	99.94
PB4W16[32]	29.038 7	99.42	34.583 0	99.66	34.561 0	99.53	39.463 0	99.77	32.506 9	99.40	34.030 5	99.56
IPB4W16[32]	34.323 9	99.64	37.654 6	99.83	38.488 1	99.81	42.373 1	99.86	36.333 5	99.76	37.834 6	99.78
PB8W16[32]	44.574 0	99.97	46.368 8	99.98	47.823 1	99.98	46.099 8	99.96	48.251 0	99.98	46.623 3	99.97
IPB8W16[32]	45.491 6	99.98	48.469 7	99.99	49.337 7	99.98	46.813 3	99.97	50.011 9	99.99	48.024 8	99.98



(a) PSNR vs PDP (b) PSNR vs ADP

Fig. 12 Relationship diagram of hardware parameters and PSNR for different approximate multipliers

图 12 不同近似乘法器的硬件参数与 PSNR 的关系图

图像融合实验是将 2 幅不同的灰度图融合并输出为一张图像，本文分别选取了“Walkbridge”×“House”，“Cameraman”×“Lake”和“Airplane”×“Clock”共 3 组图像进行融合测试，图像融合效果如图 13 所示，实验得到的 PSNR 和 SSIM 如表 6 所示. 同样，图 14 展示了不同近似乘法器在图像融合中的 PSNR 与硬件指标的关系.

实验结果表明，MTA5T4，MTA4T5~MTA7T5 的



Walkbridge×House Cameramen×Lake Airplane×Clock

Fig. 13 Fusion effect of three sets of images

图 13 3 组图像的融合效果

Table 6 PSNR and SSIM of Image Fusion

表 6 图像融合的 PSNR 和 SSIM

乘法器	Walkbridge×House		Cameraman×Lake		Airplane×Clock		平均值	
	PSNR/dB	SSIM/%	PSNR/dB	SSIM/%	PSNR/dB	SSIM/%	PSNR/dB	SSIM/%
MTA3T3	44.762 9	99.96	42.846 3	99.93	40.051 0	99.94	42.553 4	99.94
MTA4T3	48.518 3	99.98	46.809 5	99.97	43.877 2	99.97	46.401 7	99.97
MTA5T3	50.878 3	99.99	48.655 5	99.98	46.358 6	99.98	48.630 8	99.98
MTA3T4	46.548 8	99.97	49.596 1	99.98	45.532 8	99.98	47.225 9	99.98
MTA4T4	49.820 6	99.99	51.997 2	99.99	48.185 7	99.99	50.001 1	99.99
MTA5T4	52.926 2	99.99	54.627 4	99.99	51.420 4	99.99	52.991 3	99.99
MTA3T5	50.880 0	99.99	51.718 0	99.99	47.246 2	99.99	49.948 1	99.99
MTA4T5	53.460 9	99.99	54.217 6	99.99	50.568 3	99.99	52.748 9	99.99
MTA5T5	54.800 4	100.00	54.987 4	100.00	52.049 8	100.00	53.945 9	100.00
MTA6T5	55.208 2	100.00	55.506 6	100.00	52.628 2	100.00	54.447 7	100.00
MTA7T5	55.417 4	100.00	55.791 1	100.00	52.628 2	100.00	54.612 2	100.00
HPC[19]	52.536 6	99.99	53.334 7	99.99	49.953 0	99.99	51.941 4	99.99
LPC[19]	40.502 1	99.89	43.678 7	99.94	36.517 8	99.88	40.232 9	99.90
CFPM[28]	50.826 8	99.99	50.805 5	99.99	46.151 7	99.98	49.261 3	99.99
MBM[29]	38.251 6	99.71	34.802 6	99.50	32.386 1	99.49	35.146 8	99.57
CFPU[30]	45.267 7	99.97	44.758 8	99.95	40.056 8	99.94	43.361 1	99.95
PB4W16[32]	36.856 0	99.67	35.790 7	99.59	32.466 8	99.37	35.037 8	99.54
IPB4W16[32]	35.616 4	99.63	35.433 7	99.57	32.235 0	99.49	34.428 4	99.56
PB8W16[32]	50.842 7	99.99	49.238 3	99.98	45.874 0	99.97	48.651 7	99.98
IPB8W16[32]	40.006 1	99.98	47.694 3	99.97	45.432 9	99.97	47.377 8	99.97

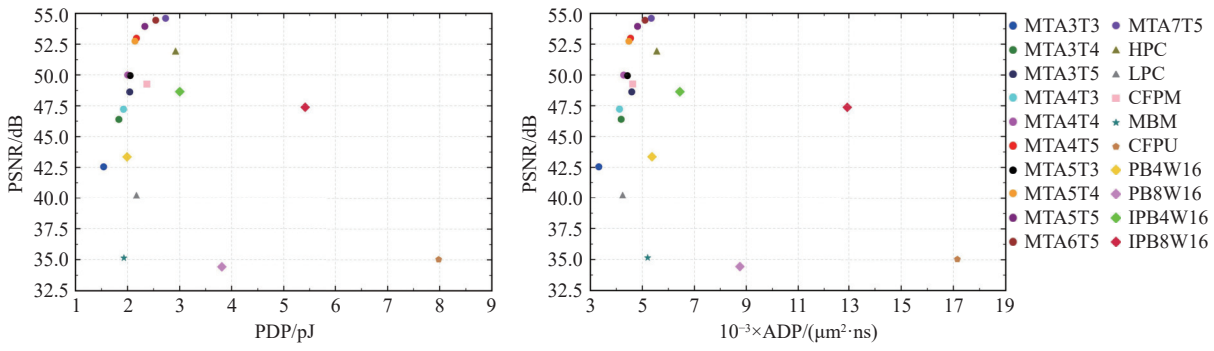


Fig. 14 Relationship diagram of hardware parameters and PSNR for different approximate multipliers

图 14 不同近似乘法器的硬件参数与 PSNR 的关系图

精度优于其他近似乘法器，在图像处理中具有最高的 PSNR 值和 SSIM 值. 在图 12 和图 14 中，可以看出所提出的大部分乘法器的值位于坐标关系图的左上角，相比其他乘法器更靠近帕累托前沿，这表明所提出的 MTA 乘法器在精度和硬件开销方面具有更好的权衡，进一步验证了所提出的近似浮点乘法器设计方案在图像处理应用中的有效性和实用性.

5 总 结

本文提出了一种可重构的 32 位单精度近似浮点乘法器设计方案. 为了进一步优化当前近似乘法器设计的精度和开销之间的平衡，我们提出了 MTA 的近似计算方法，通过将乘法运算近似为加法和移位

运算, 可以实现乘法低权重部分的近似计算. 在高有效位进行了多种不同的非对称截断处理并执行精确计算来保证精度和降低开销. 非对称截断配合 MTA 近似算法实现了近似设计的可重构性, 产生了较大的近似设计空间. 与精确浮点乘法器相比, 所提出的乘法器功耗下降 83.34%~90.55%, 面积下降 77.27%~85.80%, PDP 下降 83.48%~90.69%, ADP 下降 77.46%~86.01%. 与其他先进的近似设计相比, 提出的多种乘法器 MRED 在 0.4% 以内. 最后, 在 FIR 滤波和图像处理应用中验证了设计的优越性.

作者贡献声明: 李鹏程提出了设计优化思路、完成实验并撰写论文; 黄立波、陈刚和王永文对设计和实验提出指导意见并进行了论文的修改; 赖明澈、邓林、刘威和杨乾明修改了论文.

参 考 文 献

[1] Al-Fuqaha A, Guizani M, Mohammadi M, et al. Internet of things: A survey on enabling technologies, protocols, and applications[J]. IEEE Communications Surveys & Tutorials, 2015, 17(4): 2347−2376

[2] Zhuo Cheng, Luo Shaoheng, Gan Houle, et al. Noise-aware DVFS for efficient transitions on battery-powered IoT devices[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(7): 1498−1510

[3] Ding Wenlong, Wang Chengning, Tong Wei. Energy efficient floating-point memristive in-memory processing system based on self-selective mantissa compaction[J]. Journal of Computer Research and Development, 2022, 59(3): 533−552 (in Chinese)
(丁文隆, 汪承宁, 童薇. 基于自选尾数压缩的高能效浮点忆阻存内处理系统[J]. 计算机研究与发展, 2022, 59(3): 533−552)

[4] Liu Weiqiang, Lombardi F, Shulte M. A retrospective and prospective view of approximate computing[J]. Proceedings of the IEEE, 2020, 108(3): 394−399

[5] Lundstrom M. Moore's law forever?[J]. Science, 2003, 299(5604): 210−211

[6] Shafique M, Hafiz R, Rehman S, et al. Invited: Cross-layer approximate computing: From logic to architectures[C]//Proc of the 53rd Annual Design Automation Conf. New York: ACM, 2016: 1−6

[7] Han Jie, Orshansky M. Approximate computing: An emerging paradigm for energy-efficient design[C]//Proc of IEEE European Test Symp (ETS). Piscataway, NJ: IEEE, 2013: 1−6

[8] Deng Jianing, Shi Zhiguo, Zhuo Cheng. Energy-efficient real-time UAV object detection on embedded platforms[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(10): 3123−3127

[9] Gupta V, Mohapatra D, Park S P, et al. IMPACT: Imprecise adders for low-power approximate computing[C]//Proc of Int Symp on Low Power Electronics and Design. Piscataway, NJ: IEEE, 2011: 409−414

[10] Venkataramani S, Chippa V K, Chakradhar S T, et al. Quality programmable vector processors for approximate computing[C]//Proc of the 46th Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2013: 1−12

[11] Liang Jian, Tessier R, Mencer O. Floating point unit generation and evaluation for FPGAs[C]//Proc of Annual IEEE Symp on Field-Programmable Custom Computing Machines. Piscataway, NJ: IEEE, 2003: 185−194

[12] Imani M, Masich M, Peroni D, et al. CANNA: Neural network acceleration using configurable approximation on GPGPU[C]//Proc of the 23rd Asia and South Pacific Design Automation Conf (ASP-DAC). Piscataway, NJ: IEEE, 2018: 682−689

[13] Imani M, Garcia R, Gupta S, et al. RMAC: Runtime configurable floating point multiplier for approximate computing[C]//Proc of the Int Symp on Low Power Electronics and Design. New York: ACM, 2018: 1−6

[14] Zuras D , Cowlishaw M , Aiken A , et al. IEEE Standard for Floating-Point Arithmetic[J]. IEEE Std 754−2008, 2008: 1−70

[15] Lin C H, Lin I C. High accuracy approximate multiplier with error correction[C]//Proc of Int Conf on Computer Design (ICCD). Piscataway, NJ: IEEE, 2013: 33−38

[16] Bhardwaj K, Mane P S, Henkel J. Power- and area-efficient approximate wallace tree multiplier for error- resilient systems[C]// Proc of 15th Int Symp on Quality Electronic Design. Piscataway, NJ: IEEE, 2014: 263−269

[17] Liu Cong, Han Jie, Lombardi F. A low-power, high performance approximate multiplier with configurable partial error recovery[C]// Proc of Design, Automation & Test in Europe Conf & Exhibition (DATE). Piscataway, NJ: IEEE, 2014: 1−4

[18] Narayanamoorthy S, Moghaddam H A, Liu Z, et al. Energy-efficient approximate multiplication for digital signal processing and classification applications[J]. IEEE Transactions on Very Large Scale Integration Systems, 2015, 23(6): 1180−1184

[19] Li Zhenhao, Lu Zhaojun, Jia Wei, et al. Efficient approximate floating-point multiplier with runtime reconfigurable frequency and precision[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2024, 71(7): 3533−3537

[20] Yao Shangshang, Zhang Liang, Wang Qiong, et al. An efficient hybrid parallel compression approximate multiplier[C]//Proc of Int Conf on Computer Design (ICCD). Piscataway, NJ: IEEE, 2021: 109−116

[21] Kumar U A, Bikki P, Veeramachaneni S, et al. Power efficient approximate multiplier architectures for error resilient applications[C]//Proc of IEEE 19th India Council Int Conf (INDICON). Piscataway, NJ: IEEE, 2022: 1−5

[22] Park G, Kung J, Lee Y. Simplified compressor and encoder designs for low-cost approximate radix-4 booth multiplier[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2023, 70(3): 1154−1158

[23] Sabetzadeh F, Moaiyeri M H, Ahmadinejad M. An ultra-efficient approximate multiplier with error compensation for error-resilient applications[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2023, 70(2): 776−780

[24] Haider M H, Zhang Hao, Ko S B. Decoder reduction approximation scheme for booth multipliers[J]. IEEE Transactions on Computers, 2024, 73(3): 735−746

[25] Tan Hongbing, Tong Gan, Huang Libo, et al. Multiple- mode-

supporting floating-point FMA unit for deep learning processors[J]. IEEE Transactions on Very Large Scale Integration Systems, 2023, 31(2): 253−266

[26] Tan Hongbing, Zhang Jing, Huang Libo, et al. A multi-level parallel integer/floating-point arithmetic architecture for deep learning instructions[C]//Proc of Euro-Par 2023: Parallel Processing. Berlin: Springer, 2023: 260−274

[27] Zhang Hao, Ko S B. Variable-precision approximate floating-point multiplier for efficient deep learning computation[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, 69(5): 2503−2507

[28] Saggese G, Napoli E, Strollo A G M. CFPM: Run-time configurable floating-point multiplier[C]//Proc of the 18th Conf on PhD Research in Microelectronics and Electronics (PRIME). Piscataway, NJ: IEEE, 2023: 173−176

[29] Saadat H, Bokhari H, Parameswaran S. Minimally biased multipliers for approximate integer and floating-point multiplication[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(11): 2623−2635

[30] Imani M, Peroni D, Rosing T. CFPU: Configurable floating point multiplier for energy-efficient computing[C]//Proc of the 54th Annual Design Automation Conf 2017. New York: ACM, 2017: 1−6

[31] Jha C K, Walia S, Kanojia G, et al. FPCAM: Floating point configurable approximate multiplier for error resilient applications[C]//Proc of Int Symp on Circuits and Systems (ISCAS). Piscataway, NJ: IEEE, 2021: 1−5

[32] Towhidy A, Omidi R, Mohammadi K. On the design of iterative approximate floating-point multipliers[J]. IEEE Transactions on Computers, 2023, 72(6): 1623−1635

[33] Liang Jinghang, Han Jie, Lombardi F. New metrics for the reliability of approximate and probabilistic adders[J]. IEEE Transactions on Computers, 2013, 62(9): 1760−1771



Li Pengcheng, born in 1995. Master candidate. His main research interest includes computer architecture.

李鹏程, 1995 年生. 硕士研究生. 主要研究方向为计算机体系结构.



Huang Libo, born in 1983. PhD, professor. Senior member of CCF. His main research interests include computer architecture and microprocessor design.

黄立波, 1983 年生. 博士, 研究员. CCF 高级会员. 主要研究方向为计算机体系结构、微处理器设计.



Chen Gang, born in 1985. PhD, professor. Senior member of CCF. His main research interests include artificial intelligence, accelerator design, and embedded systems.

陈　刚, 1985 年生. 博士, 教授. CCF 高级会员. 主要研究领域为人工智能、加速器设计、嵌入式系统.



Lai Mingche, born in 1982. PhD, professor. Member of CCF. His main research interest includes computer architecture.

赖明澈, 1982 年生. 博士, 教授. CCF 会员. 主要研究方向为计算机体系结构.



Deng Lin, born in 1980. PhD, associate professor. Member of CCF. His main research interests include computer architecture and high-performance processor design.

邓　林, 1980 年生. 博士, 副研究员. CCF 会员. 主要研究方向为计算机体系结构和高性能处理器设计.



Liu Wei, born in 1986. PhD, assistant professor. Member of CCF. His main research interest includes microprocessor design.

刘　威, 1986 年生. 博士, 助理研究员. CCF 会员. 主要研究方向为微处理器设计.



Yang Qianming, born in 1984. PhD, assistant professor. Member of CCF. His main research interest includes computer architecture.

杨乾明, 1984 年生. 博士, 助理研究员. CCF 会员. 主要研究方向为计算机体系结构.



Wang Yongwen, born in 1977. PhD, professor. Senior member of CCF. His main research interest includes microprocessor architecture.

王永文, 1977 年生. 博士, 研究员. CCF 高级会员. 主要研究方向为微处理器体系结构.