计算机学报
CHINESE JOURNAL OF COMPUTERS

面向 E 量级超算的并行循环压缩浮点乘加校验结构

高剑刚　刘骁　郑方　唐勇

(国家并行计算机工程技术中心, 北京　100190)

摘　要　E 量级超算面临超十亿浮点融合乘加(FMA, Fused Multiply-Add)部件同时运行的严峻挑战，单个 FMA 检错率的少量变化可引起系统可用性的较大变动。E 级超算核心的高运行频率、实时校验需求对校验逻辑时序提出了更高的要求。同时，E 级超算需要控制系统规模，同芯片面积下集成的核心数目更多，片上资源较为紧张。因此，FMA 校验设计需要在保证错误检测能力的前提下，对校验逻辑的时序、面积开销进行控制。本文提出了并行循环 4:2 压缩结构。余数系统模数增大后，并行循环 4:2 压缩结构能在降低余数生成逻辑的时序、面积开销的同时，提升余数系统的检错能力。本文还对余数域中的 FMA 尾数运算进行研究，提出了取反符号扩展操作、乘法尾数、加法尾数的余数域加速变换。实验结果表明，本文提出的并行循环 4:2 混合压缩余数生成逻辑较模加器树余数生成逻辑、CSA(Carry Saved Adder)3:2 压缩余数生成逻辑最多可取得 19.64%、6.75%的时序优化和 71%、18.18%的面积优化。基于并行循环 4:2 压缩树的模 63 余数校验在面积开销、检错率、系统可用性上均优于 IBM 采用的模 15 浮点 FMA 校验设计，面积开销、检错率优化效果分别能达到 67.61%、5%，系统可用性优化最多可达 49.6%。

关键词　浮点融合乘加；可用性；浮点校验；模加器；并行循环压缩

中图法分类号　TP302

Exascale Supercomputer Oriented Parallel Cyclic Compression Based Checking Structure for Floating-Point Fused Multiply-Add Unit

Gao Jiangang　Liu Xiao　Zheng Fang　Tang Yong

(National Research Center of Parallel Computer Engineering and Technology, Beijing 100190)

Abstract　Simultaneously operating of billions of floating-point FMA (Fused Multiply-Add) units has raised severe availability challenges for the exascale supercomputer. To ensure sustainable and efficient operation of the exascale supercomputer, processors must adopt more efficient fault-tolerance mechanisms on FMA. In the exascale supercomputer, the real-time check on high frequency processor and limited resources on chip challenge the design of FMA checker. The design of FMA checker must take timing overhead and hardware overhead into consideration under the premise of getting better error detection coverage. Floating-point FMA adopts a fusion design and has to deal with multiple special operations in IEEE 754 standard, such as mantissa align shift, normalization, round; as a result, the widely-used residue domain transformation is not able to effectively accelerate the residue encoding in FMA units. In this paper, we propose a parallel cyclic 4:2 compressor-based residue generation technique, which reduces the number of logic gates on the critical path when the modulus is increasing. By adopting cyclic carry processing for the highest bit of each partition, cyclic 4:2 compressors abate the logical dependency in carry chains and reduce the overhead caused by carry correction. When improving error detection coverage, the cyclic 4:2 compressor can reduce the timing cost and hardware overhead of residue generation. We also study the mantissa calculation in residue domain and propose the residue domain compression technology for negative sign extension of mantissa, mantissa multiplication and mantissa addition based on mathematical transformations. These

高剑刚（**通信作者**），硕士，正高级工程师，主要研究领域为计算机体系结构、高性能互联网络.E-mail: 13701512205@139.com. 刘骁，硕士，工程师，主要研究领域为高性能计算和处理器结构.E-mail: ustc_m@163.com. 郑方，博士，副研究员，主要研究领域为高性能计算和处理器结构.E-mail: zhengfangwww_2000@163.com.唐勇，硕士，副研究员，主要研究领域为高性能计算和处理器结构.E-mail: donrong@139.com.

techniques reduce the input data width of the residue generator and limit the alignment range by dividing and transforming the mantissa fusion operation. For the reverse sign extension of mantissa in residue domain, this paper decreases the overhead by transforming the negative sign extension operation to the combined operations of residue generation and modular subtraction. For the mantissa multiplication in residue domain, this paper utilizes mathematical transformations to separate mantissa multiplication result from the mantissa fusion operation on FMA main path. Then multiplication distribution rate in residue domain is allowed to reduce the overhead of residue calculation of mantissa multiplication. For the mantissa addition in residue domain, this paper avoids large shift range caused by alignment by utilizing modular shift and modular subtraction. By using these techniques, the area overhead of shift logic and multiplication logic can be reduced by 10 times on average. Timing is also improved by utilizing these transformations in mathematic. Experimental results show that both timing cost and area cost of residue generation are optimized by utilizing the parallel cyclic compression structure. Compared with modular adder-based residue generator and carry-saved adder-based residue generator, the parallel cyclic 4:2 compressor-based residue generator shows up to 19.64%, 6.75% timing optimization and 71%, 18.18% area reduction respectively. The residue system proposed in this paper outperforms conventional design in terms of area overhead and error detection coverage. Compared with the moduli 15 residue check system for FMA proposed by IBM, the moduli 63 FMA checker based on parallel cyclic 4:2 compressor reduces area by 67.61%, yields 5% error coverage improvement and yields up to 49.6% exascale supercomputer’s availability improvement.

Key words floating-point fused multiply-add; availability; residue check; modular adder; parallel cyclic compression

1 引言

当前，高性能计算已经迈入后E级时代，由于后E级超算系统结构复杂、规模庞大，系统保存全局检查点的时间正逼近系统的平均无故障时间[1]。可靠性问题已成为制约后E级高性能计算发展的瓶颈之一[2]。

浮点融合乘加(FMA, Fused Multiply-Add)部件作为处理器芯片的核心部件，是高性能计算通用算力的重要支撑部分。后 E 级超算系统集成的 FMA 规模将超十亿量级，系统可靠性将面临十亿级 FMA 同时运行带来的严峻挑战。若 E 级超算系统需要保持分钟级的平均失效前时间(Mean Time To Failure, MTTF)，单个 FMA 部件的 MTTF 至少要达到万年量级。浮点融合乘加部件的可靠运行能力对全芯片影响较大，其运算结果不仅影响程序中的数据正确性，而且还会对程序执行的轨迹造成影响[3]。性能方面，当计算错误发生后，系统若不能及时检错，程序的收敛时间将持续恶化、甚至不可收敛。系统将出现大量无意义的后续计算，频繁软错误下系统的运行效率将难以得到保障。能耗方面，E 级超算系统功耗将超过 20MW[2][3]，程序的错误执行将大幅增加系统总能耗，导致实际性能功耗比的下降。因此，研究面向 E 量级的 FMA 容错技术具有较为重要的意义。

运算部件中通常采用多模冗余或校验编码校验技术，以实时校验的方式尽早定位运算过程中的软错误，进而通过软硬件协同的方式完成容错操作。由于检错所需次数服从参数为检错率的几何分布，保持实时校验能力、提升软错误检测率对减少检错所需次数、优化平均修复时间(Mean Time To Repair, MTTR)、提升系统可用性、优化系统实际能效等方面具有重要意义。

E 级超算系统上的校验逻辑面临着较为严峻的挑战。时序方面，E 级超算核心运行频率较高，实时校验对校验逻辑的时序提出了更高的要求。面积方面，E 级超算需要控制系统规模，片上资源紧张，需要对校验逻辑的复杂度和开销进行控制，从而在更多的运算部件上实现实时校验功能。电路规模的增大、时序的紧张都将增大电路的软错误率，不利于电路的可靠运行[4-6]。因此，需要在保证错误检测能力的前提下，针对检错逻辑的时序、面积进行优化，从而为芯片及系统提供高效的容错支撑。

由于浮点运算部件是芯片面积的重要消耗部分之一[7]，多模冗余的检错方式将带来较大的面积开销。部分冗余校验技术牺牲了检测精度，仅对浮点运算部分逻辑、部分数据进行冗余检测，能在较小的开销下对浮点乘法、加法、除法运算进行检错[7-9]。以上设计虽然硬件开销优于多模冗余设计，但

错误检测范围、错误检测率均有所降低。增加冗余精度可以提升错误检测能力，但同时也会明显增加硬件开销[10]。

运算部件校验编码的研究主要集中于整数部件，面向浮点部件的校验研究较少。余数码是整数运算通路上应用广泛的校验码，其理论检错率会随着模数的增大而提升。IBM 面向 FMA 运算设计了余数校验部件，受限于时序、面积开销，采用模 3、模 15 的浮点校验逻辑对浮点 FMA 运算进行检错[11][12]。IBM 的分布式余数校验[12]的加数余数生成依赖扩展移位后的结果，以上处理不利于面积开销的优化。同时，余数生成逻辑易成为校验设计的关键时序路径，不利于校验部件的稳定运行。

由于浮点运算通路较整数运算通路运算更为复杂，整数余数校验中常用的优化方式难以有效加速 FMA 操作的余数码计算。因此，浮点 FMA 校验部件的设计需要面向硬件开销，探讨浮点乘加操作在余数域上的变换。

本文的主要贡献如下：

（1）本文利用单位门模型对基于压缩树的余数生成逻辑进行了时序、面积评估。评估表明，基于并行循环 4:2 压缩树的余数生成逻辑能在提升余数检错率的同时保持面积开销基本不变，同时对逻辑的时序开销进行优化。

（2）本文面向浮点 FMA 操作设计了高检错率、低开销的余数校验部件。设计中针对取反符号扩展、尾数乘法、加数尾数等部分提出了余数域加速变换方法。本文综合利用以上方法对浮点 FMA 检错逻辑的时序、面积开销进行了优化。

（3）本文对不同结构下的双精度尾数余数生成逻辑进行了实验评估，实验选择基于模加器树、CSA(Carry-Saved Adder)3:2 压缩树、并行循环 4:2 混合压缩树结构，对不同检错能力下的余数生成逻辑进行综合、对比。实验结果表明并行循环结构在模数增大时，时序和面积开销方面均能得到优化，优化效果和单位门模型推导一致。

（4）本文完成了浮点 FMA 余数校验部件的设计空间探索。实验中对比了不同模数下基于模加器树、CSA3:2 压缩树、并行循环 4:2 混合压缩树的 FMA 校验部件的面积开销，并给出了各参数下校验部件各部分逻辑的面积占比，为 FMA 余数校验的进一步优化提供了依据。

（5）本文给出了单 FMA 检错能力对 E 级超算系统可用性的理论评估。评估表明单 FMA 检错能力的较小变化可能对系统可用性造成巨大影响。

本文组织结构如下，第 2 节介绍相关工作；第 3 节介绍余数域运算的基本原理；第 4 节首先对余数并行循环压缩生成算法进行介绍并给出理论推导，其次结合浮点乘加运算的特点，提出了取反扩展尾数、尾数乘法、加数尾数等余数域加速变换，对 FMA 校验的时序和面积开销进行了优化；第 5 节基于 DCG(Design Compiler Graphical)综合工具进行实验，并对实验结果进行比较和分析；第 6 节对论文进行总结。

2 相关工作

多模冗余是提升芯片可用性的重要技术方向之一。NVIDIA 通过冗余核心来提高芯片可用性，A100、H100 系列芯片均仅能提供部分 SM(Streaming Multiprocessor)供程序员使用[13][14]。ARM、RISC-V 则主要采用更细粒度的多模冗余方式对运算部件进行错误检测[15][16]。针对多模冗余技术开销较大的问题，多个研究采用牺牲数据精度、减少检测范围的方式对检错逻辑的面积开销进行了优化。Maniatakos 等人[8]利用控制通路信息、浮点指数数据对浮点加法、乘法、除法逻辑进行错误检测，结合模 15 余数码浮点对尾数部分进行校验后能在 16.32%的 FPU 面积开销下达到 94.1%的错误覆盖率。Eibl 等人[9]提出了弱化精度技术，研究中通过牺牲数据精度检测范围对浮点加法检错逻辑的面积开销进行了优化。Seetharam 等人[17]主要采用牺牲输入数据精度的方式对检错逻辑开销进行优化，该研究给出了位数重叠部分对检错能力的影响。Kito 等人[18]针对精简输入数据检错能力损失过大的问题，在检错逻辑的设计中选择乘法华莱士树进行切分精简，较 Seetharam 等人[17]的设计能取得更高的检错覆盖率。以上两篇文献仅针对数据高位错误进行检测，对浮点乘法的弱化精度检错进行了研究。Zhang 等人[10]对浮点加法、乘法、除法、开方的弱化精度检错技术进行了研究。该研究提出了反向运算技术，能解决幅值相减操作时弱化精度在规格化移位后尾数高位和实际尾数高位不相等的问题。但反向运算技术需要主通路逻辑输出数据高位作为检错逻辑输入，不能做到实时校验。

余数码检错通过对比输入数据的余数域运算结果、主通路运算输出的余数域运算结果，能达到对运算主通路进行正确性验证的目的。余数校验采

用主通路运算结果的余数码作为正确性校验的特征值。余数校验特征值与运算主通路数据位相比，减少了位数。整数运算的余数校验中，通常利用余数域的整数乘法、整数加法等变换来优化输入数据余数域运算的时序、面积开销。

余数校验的研究主要集中于整数部件，面向浮点部件的余数校验研究较少。Lo[19]以硬件开销作为评价标准，在浮点运算部件中对 Berger 校验码和余数码技术进行了比较。实验中分别比较了浮点加法部件和浮点乘法部件在采用这两种校验码下的硬件开销。Lo[19]提供的实验表明，浮点加法部件中模 3、模 15 余数校验逻辑和 Berger 校验逻辑面积开销相近，浮点乘法部件中模 3、模 15 余数校验逻辑的面积开销较 Berger 校验逻辑的面积开销均能优化一个数量级。受限于时序、面积开销，国际上通常采用模 3、模 7、模 15 余数码对浮点乘加部件进行校验[11][20]。IBM 在 P 系列和 Z 系列处理器中利用冗余部件及余数码技术对浮点运算部件(FPU, Floating-point Process Unit)中的浮点乘加操作进行了容错处理。其中，POWER7 采用的模 15 的浮点校验逻辑占整个 FPU 面积的 18%[11]。但 Lipetz 等人[11]对浮点余数校验部件的设计结构介绍较为欠缺，不利于实验结果的复现[17]。IBM 的分布式余数校验[12]对浮点余数校验的门控功耗控制进行了研究，分布式的余数校验可分别对乘法和浮点加法分别进行校验及门控，但分布式的余数校验设计不利于面积开销的优化。IBM RBEFC[21]针对 FMA 部件指数域的运算进行了余数校验，但设计主要对规格化移位后的数据进行校验，对于舍入操作后的指数没有进行保护。IBM 的设计基于孙子定理，利用模 3、模 5 余数域结合独热编码器、移位器实现模 15 的余数生成。但由于小模数的余数生成逻辑层数较多，时序较为紧张。

3 余数域运算基本原理

本节对余数码涉及的基本原理进行介绍。为方便算法描述,本文中的变量定义如表 1 所示。

定义 1. 给定正整数 p，对于任意整数 n，可表示为 $n = k*p + r$，其中 $0\le r<p$。称 r 为 n 除以 p 的余数。给定 p，由 n 得到 r 的运算称为模 p 运算，记为

$$n \bmod p = r \quad 或 \quad |n|_p=r$$

定义 2（模 p 余数集）.给定正整数 p，对整数集 $\mathbb{Z}$中所有元素进行模 p 运算的结果组成的集合称为模 p 余数集，记为 $\mathbb{Z}_p$。模 p 余数集的元素为 0 到 p-1，因此

$$\mathbb{Z}_p= \{0,1,\cdots, p-1\}$$

模 p 运算具有如下性质[19]:

性质 1. 设 A 为大于 2 的整数，则对于 $M=A$-1，和任意正整数 k，有$\left|A^k\right|_M=1$

性质 2. 对于一个整数 X，记其 A 进制表示为 $(x_{n-1}x_{n-2}\cdots x_1x_0)_A$，则对于 $M=A$-1，有$|X|_M=\left|\sum_{i=0}^{n-1}x_i\right|_M$

性质 3. 对于任意整数 a、b，有 $|a+b|_p=\left||a|_p+|b|_p\right|_p$

性质 4. 对于任意整数 a、b，有$|a\times b|_p=\left||a|_p\times|b|_p\right|_p$

性质 5. 设 $M=2^m$-1，对于任意正整数 p，有 $|2^p|_M=\left|2^{|p|_m}\right|_M$

性质 6. 设 $M=2^m$-1，对于任意能被2^m整除的正整数 X 和满足 $p+q=m$ 的正整数 p、q，有$\left|\frac{X}{2^p}\right|_M=|X\times 2^q|_M$

表1. 变量定义表

符号	意义
M	模数
m	模数二进制位宽
M_X	浮点数的 X 尾数
H_X	浮点数的 X 尾数隐藏位
mx_i	浮点数的 X 尾数的第 i 位
E_X	浮点数 X 的指数移码
SUB	浮点数尾数幅值相减标记
Q	A×B尾数幅值大于 C 移位对齐后尾数幅值的标记
T_C	浮点数 C 因对齐移位丢弃的尾数
ST	浮点数 sticky 位
ASC	浮点数 C 的对齐移位数 ASC(Align Shift Count)
T_m	规格化/非规格化移位丢弃部分
tm_i	T_m 第 i 位
N_SHT	规格化/非规格化移位的位数
$addzero$	余数生成逻辑尾数低位补零位$_{m-\left\|3\right\|_m}$
μ	单 FMA 恢复率
λ	单 FMA 失效率

余数域运算相比主通路运算减少了位数，较运算主通路逻辑能有效减少面积开销。浮点 FMA 余数

校验涉及的关键运算组成部分包括余数生成、余数加法、余数乘法和余数移位运算。

余数生成可以基于除法运算进行，但这种方式实现运算开销太大。模数 M 取2^m时有快速求余算法，但模2^m仅对输入的低 m 位进行检查。本文模数 M 取2^m-1，这种设置下有快速的余数生成算法。根据性质 1、性质 2，对于位长为 $n=mk$ 位的整数 X，从最低位开始，将 X 划分为 k 份，每个划分a_i为 m 位，$|X|_M$等效于每个a_i相加后再取模的结果。但a_i求和的过程中会导致中间结果位数的不断增长，余数的计算需要经历多次重复的划分、求和过程。性质 3 对余数域加法变换进行了描述，利用先取模再模加的方式，能有效降低余数生成中求和结果位数增加带来的时序和面积开销。综合性质 1、性质 2、性质 3，$|X|_M$则可以等效于每个a_i分别取模后再模加的结果。不同于 X 模2^m余数校验中仅取低 m 位作为校验特征值，模数取2^m-1时 X 的所有位数均参与校验特征值运算。相比模2^m校验，模2^m-1校验对输入的检查更加全面，错误检测能力可达$\frac{2^m-2}{2^m-1}$[22]。

余数校验中通常利用余数域的整数乘法、整数移位等变换来优化输入数据余数域运算的时序、面积开销。由于乘法面积和输入数据位宽 n 的平方成正比，随着 n 的增加，乘法面积开销的增加较为明显。性质 4 对乘法各个输入数据采用先取模、再模乘的方式，可以将乘法输入数据的位宽降低至 m 位，从而优化面积开销。移位器的面积开销对数据数据位宽、移位范围较为敏感。对于最大移位位宽均为 n 的对数移位器，面积开销呈 O($n$log$n$) 趋势增长。性质 5、性质 6 分别针对循环左移、循环右移进行余数域变化，通过对移位范围的压缩可以优化相应逻辑的时序及面积。

4 余数域浮点乘加校验部件

浮点 FMA 部件可以支持对乘法和加法的融合运算，一轮执行可以完成 $Y=A\times B+C$ 形式的操作。浮点 FMA 运算流程如图 1 所示，bias 为浮点数的指数偏移常量。FMA 运算可分为符号域运算、指数域运算和尾数域运算三大部分。其中，符号域部分根据浮点数 A、B、C 原本的符号值和尾数结果进行确定；指数域部分由浮点数 C 指数 E_C 与浮点数 A、B 乘法结果对应指数 E_A+E_B-bias 决定，并结合尾数结果进行修正；尾数域部分可分为乘法结果部分和加数部分，乘法结果部分由浮点 A、B 的尾数 $M_A\times M_B$ 决定，加数部分由浮点 C 的尾数加法 M_C 进行位移等变换得到。

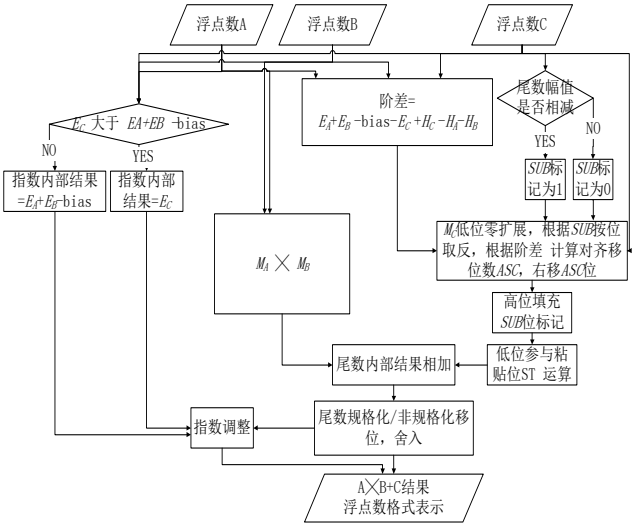

图1 FMA 运算流程图

浮点 FMA 部件是芯片面积的重要消耗部分之一，FMA 部件的浮点校验逻辑需要在保证检错率的前提下，面向面积和时序开销进行优化。由于浮点运算通路较整数运算通路运算更为复杂，涉及 IEEE 754 标准中 guard 位、sticky 位和规格化、舍入等多个特殊处理，整数余数校验中常用的余数域加法、余数域乘法变换等优化方式难以直接加速浮点乘加操作的余数码计算。因此，本文针对余数域 FMA 操作中时序、面积开销较大的部分，探讨浮点乘加操作在余数域上的变换。

4.1 基于并行循环压缩结构的余数生成逻辑

本小节将具体阐述并行循环 4:2 压缩结构，并基于并行循环 4:2 压缩结构设计余数生成逻辑。本小节基于单位门模型，分别对利用模加器、CSA3:2 压缩器、循环 4:2 压缩器构建的余数生成逻辑进行理论评估。单位门模型常用于评估逻辑的面积与时序，该模型中两输入的“与”门、“或”门、“与非”门和“或非”门可视为单位门；两输入“异或”和“同或”门的面积、延时均为单位门的两倍[23]。

4.1.1 常规余数生成逻辑

余数码的检错电路包括余数码生成模块，余数运算模块和比较模块，余数生成是余数码运算的重要组成部分。基于性质 1、性质 2、性质 3，可以采用模加器树、CSA3:2 压缩树对余数生成逻辑进行构建，分组后按层进行模加，最终求得模 M 值。

基于模加器树的余数生成是划分求余运算的直接映射，但该结构面积、时序开销较大。模加器的逻辑功能如式(1)所示，由于需要同时对 $X+Y+1$ 及 $X+Y$ 部分进行运算，与同位宽的普通二进制加法器

相比，模加器需要忍受更高的延时及更大的面积开销[24]。Patel 等人[25]采用进位修正的方法减少了模加器中部分加法逻辑的复制，在现有研究中具有较优的时序开销及面积开销，单位门模型下的模加器时序开销为$2\log m+5$单位门延时。

$$Z=|X+Y|_{2^m-1}=\begin{cases}|X+Y+1|_{2^m} & \text{if } (X+Y)>2^m-1\\ X+Y & \text{otherwise}\end{cases} \quad (1)$$

在现有研究中，模数取2^m-1时，基于 CSA3:2 压缩树的余数生成具有最少的时序、面积开销[26][27]。基于 CSA3:2 压缩树的余数生成逻辑首先按 m 位周期对输入数据进行分组划分，然后对权值相同部分进行 CSA3:2 压缩处理，每层压缩后按 m 位周期重新进行划分后，继续进行对权值相同部分进行 CSA3:2 压缩处理。当压缩至 2 个 m 比特的伪余数时，进行模加操作，最终得到输入数据对应的模 M 值。

4.1.2 循环 4:2 压缩器

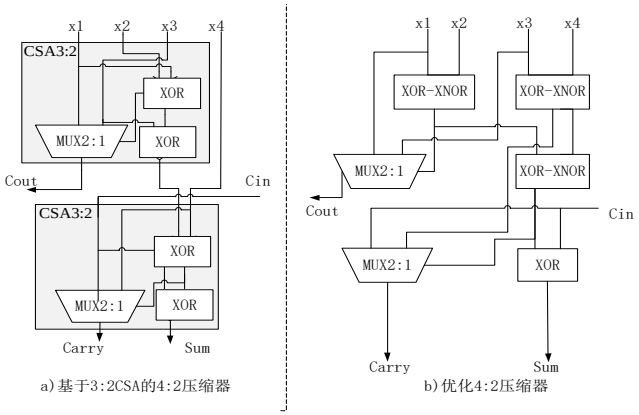

图2 4:2 压缩器门级电路实现[28]

本文基于 4:2 压缩器构建基于并行循环 4:2 压缩的余数生成逻辑。4:2 压缩器可以将 5 个 1 比特输入压缩为 3 比特输出，4:2 压缩器的主要特点是能打破进位链之间的逻辑依赖关系[29]。 4:2 压缩器可以基于多种门级电路进行优化实现，图 2 为不同实现下的 4:2 压缩器，图 2 a)直接采用 CSA3:2 实现 4:2 压缩器，时序开销与面积开销均较差；图 2 b)的 4:2 压缩器基于 XOR-XNOR 进行实现，XOR-XNOR 可同时生成 XOR 与 XNOR 结果，该 4:2 压缩器具有较优的时序开销与面积开销,单位门模型下的延迟为 6 个单位门延迟[28]。

单比特 4:2 压缩器的逻辑表达式如式(2)、(3)、(4)所示[30]，其中⊕为 XOR 操作，·为 AND 操作，|为 OR 操作。m 比特的 4:2 压缩器的运算可以表示为图 3 a)中的形式，该结构将每一单比特 4:2 压缩器的c_{out}输出作为下一单比特 4:2 压缩器的c_{in}输入，当$c_{in}[0]$和$c_{out}[m-1]$均为 0 时，该 4:2 压缩器可以将 4 个 m 比特数压缩为 2 个 m 比特数。图 3 a)中中x_1、x_2、x_3、x_4分别表示输入 4:2 压缩器的 m 比特二进制数；$c_{in}[0]$ 表示最低位的单比特进位输入；sum表示 4:2 压缩器的和输出；carry表示 4:2 压缩器的进位输出；cout$[m-1]$表示 4:2 压缩器的单比特输出。

$$sum[i]=f_1(x_1[i],x_2[i],x_3[i],x_4[i],c_{in}[i])=x_1[i]\oplus x_2[i]\oplus x_3[i]\oplus x_4[i]\oplus c_{in}[i] \quad (2)$$

$$c_{out}[i]=(x_1[i]\oplus x_2[i])\cdot x_3[i]\,|\,\overline{(x_1[i]\oplus x_2[i])}\cdot x_1[i] \quad (3)$$

$$carry[i]=f_2(x_1[i],x_2[i],x_3[i],x_4[i],c_{in}[i])=(x_1[i]\oplus x_2[i]\oplus x_3[i]\oplus x_4[i])\cdot c_{in}[i]\,|\,\overline{(x_1[i]\oplus x_2[i]\oplus x_3[i]\oplus x_4[i])}\cdot x_4[i] \quad (4)$$

由于 m 位 4:2 压缩运算的输出$c_{out}[m\text{-}1]$和carry$[m\text{-}1]$的权值为2^m，最终在求和、取模时该部分一定需要进行修正，修正处理不利于和逻辑时序的优化。而由定理 1 可知，余数域中 m 位 4:2 压缩运算的输出$c_{out}[m\text{-}1]$和carry$[m\text{-}1]$的权值为 1，即余数域运算中可以对 m 位 4:2 压缩器的输出采用循环进位处理。

由于 m 位 4:2 压缩运算的输出$c_{out}[m\text{-}1]$和carry$[m\text{-}1]$的权值为2^m，在求和、取模时该部分一定需要进行修正，该修正处理不利于和逻辑时序的优化。本文基于定理 1，通过循环进位对需要修正部分进行提前处理，优化了求和后取模时修正逻辑的时序。由定理 1 可知，余数域中 m 位 4:2 压缩运算的输出$c_{out}[m\text{-}1]$和carry$[m\text{-}1]$的权值为 1，即余数域运算中可以对 m 位 4:2 压缩器的输出采用循环进位处理。

定理 1. 设 $M=2^m-1$，对于$\sum_{j=1}^{j=4}\sum_{i=0}^{i=m-1}x_j[i]\times 2^i+c_{in}[0]=\sum_0^{m-1}sum[i]\times 2^i+\sum_0^{m-1}carry[i]\times 2^{i+1}+2^m c_{out}[m\text{-}1]$，等式的余数域运算有

$$\left|\sum_{j=1}^{j=4}\sum_{i=0}^{i=m-1}x_j[i]\times 2^i+c_{in}[0]\right|_M \quad (5)$$
$$=\left|sum[m\text{-}1{:}0]+2carry[m\text{-}2{:}0]+carry[m\text{-}1]+c_{out}[m\text{-}1]\right|_M$$

证明.

由性质 1、性质 4，有$|carry[m-1]\times 2^m|_M=|carry[m-1]|_M$，$|c_{out}[m-1]\times 2^m|_M=|c_{out}[m-1]|_M$，则

$$\left|\sum_{j=1}^{j=4}\sum_{i=0}^{i=m-1}x_j[i]\times 2^i+c_{in}[0]\right|_M$$
$$=\left|\sum_0^{m-1}2^i sum[i]+2^{i+1}carry[i]+2^m c_{out}[m-1]\right|_M$$
$$=\left|\sum_0^{m-1}2^i sum[i]+\sum_0^{m-1}2^{i+1}carry[i]+c_{out}[m-1]\right|_M$$
$$=\left|sum[m\text{-}1{:}0]+2carry[m\text{-}2{:}0]+2^m carry[m-1]+c_{out}[m-1]\right|_M$$
$$=\left|sum[m\text{-}1{:}0]+2carry[m\text{-}2{:}0]+carry[m-1]+c_{out}[m-1]\right|_M \quad (6)$$

证毕.

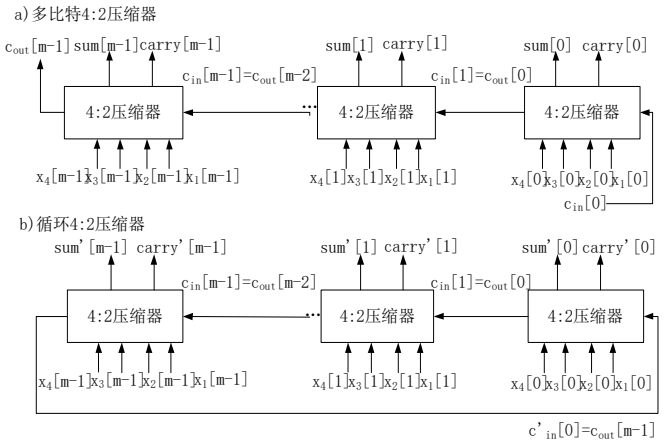

图3 多比特 4:2 压缩器及循环 4:2 压缩器

模数 M 取2^m-1的余数生成中，每个划分a_i为 m 位，每个划分之间无需进行进位计算。当模数 M 取2^m-1时，4:2 压缩器的输入可以表示为 $x_1[m\text{-}1{:}0]+x_2[m\text{-}1{:}0]+x_3[m\text{-}1{:}0]+x_4[m\text{-}1{:}0]$。循环 4:2 压缩器结构如图 3 b)所示，其中 m 比特和输出记为sum'[m-1:0]，m 比特进位输出为 carry'[m-1:0]。与 4:2 压缩器相比，循环 4:2 压缩没有将最高位 $c_{out}[m-1]$作为输出，而是将 $c_{out}[m-1]$作为 $c_{in}[0]$的输入。由定理 1，有

$|x_1[m\text{-}1{:}0]+x_2[m\text{-}1{:}0]+x_3[m\text{-}1{:}0]+x_4[m\text{-}1{:}0]+0|_M$
$=|sum[m\text{-}1{:}0]+2carry[m\text{-}2{:}0]+carry[m\text{-}1]+c_{out}[m\text{-}1]|_M$，由式(2)、(4)，有

$$sum[0]=f_1(x_1[0],x_2[0],x_3[0],x_4[0],0) \quad (7)$$
$$carry[0]=f_2(x_1[0],x_2[0],x_3[0],x_4[0],0) \quad (8)$$

则

$$\begin{aligned}&sum[m\text{-}1{:}0]+2carry[m\text{-}2{:}0]+carry[m\text{-}1]+c_{out}[m\text{-}1]\\&=f_1(x_1[0],x_2[0],x_3[0],x_4[0],0)\\&+2f_2(x_1[i],x_2[i],x_3[i],x_4[i],0)\\&+c_{out}[m\text{-}1]+\sum_1^{m-1}sum[i]\times 2^i\\&+\sum_1^{m-2}carry[i]\times 2^{i+1}+carry[m\text{-}1]\\&=f_1(x_1[0],x_2[0],x_3[0],x_4[0],c_{out}[m\text{-}1])\\&+2f_2(x_1[i],x_2[i],x_3[i],x_4[i],c_{out}[m\text{-}1])\\&+\sum_1^{m-1}sum[i]\times 2^i+\sum_1^{m-2}carry[i]\times 2^{i+1}+carry[m\text{-}1]\\&=sum'[0]+2carry'[0]+\sum_1^{m-1}sum[i]\times 2^i\\&+\sum_1^{m-2}carry[i]\times 2^{i+1}+carry[m\text{-}1]\end{aligned} \quad (9)$$

由式(3)，$c_{in}[i]=c_{out}[i-1](i\in[1,m-1])$仅和$x_1[i-1]$、$x_2[i-1]$、$x_3[i-1]$、$x_4[i-1]$有关。又由式(2)、(3)、(4)，$sum[i]$、$carry[i]$ ($i\in[1,m-1]$)仅和$x_1[i]$、$x_2[i]$、$x_3[i]$、$x_4[i]$、$x_1[i-1]$、$x_2[i-1]$、$x_3[i-1]$、$x_4[i-1]$有关，则

$$sum[i]=sum'[i] \qquad i\in[1,m\text{-}1] \quad (10)$$
$$carry[i]=carry'[i] \qquad i\in[1,m\text{-}1] \quad (11)$$

根据式(10)、(11)，式(9)的余数域运算可以变换为

$$\left|x_1[m\text{-}1{:}0]+x_2[m\text{-}1{:}0]+x_3[m\text{-}1{:}0]+x_4[m\text{-}1{:}0]+0\right|_M$$
$$=\left|sum'[m\text{-}1{:}0]+2carry'[m\text{-}2{:}0]+carry'[m\text{-}1]\right|_M \quad (12)$$

4.1.3 基于并行循环 4:2 压缩结构的余数生成逻辑

式(12)中的变换仅需要基于循环 4:2 压缩器进行实现，并行循环 4:2 压缩器的延时和 4:2 压缩器相同。基于以上推导，可以采用并行循环压缩器代替模加法器、CSA3:2 搭建余数生成逻辑。基于 CSA3:2 和并行循环 4:2 压缩器模 3 余数生成逻辑如图 4 所示，输入数据为 8 位数据 x[7:0]，图中 4:2 压缩器为单比特 4:2 压缩器。图 4a)中基于 CSA3:2

树的余数生成每过一层 CSA3:2 都需要按 m 位周期重新进行划分，共需要经过 2 层 CSA3:2。图 4a)中余数生成需要经过一级并行循环 4:2 压缩器，m=2 时并行循环 4:2 压缩器基于两个单比特 4:2 压缩器进行实现。由于两层 CSA3:2 和一层并行循环 4:2 的压缩效果一致，但一层并行循环 4:2 压缩延迟更小。所以，基于并行循环 4:2 压缩的余数生成逻辑可能具有较少的延时开销。

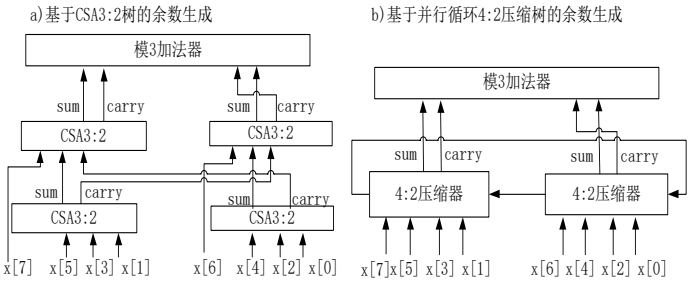

图4 基于 CSA3:2 和并行循环 4:2 压缩器余数生成逻辑

表 2 为单元门模型下的余数生成逻辑的时序及面积开销评估，路径总延迟为余数生成关键路径上各个构成单元延迟和层数相乘求和的结果，面积为余数生成逻辑各个构成单元的单位门面积和数量相乘求和的结果。表 2 中 n 为输入数据的位宽，m 为模数的位宽，τ为单位门延迟，α为单位门面积。

表 2 中模加器树、CSA3:2 压缩树、并行循环 4:2 压缩树分别指主要构成单元为模加器、CSA3:2 压缩器、循环 4:2 压缩器的余数生成逻辑。为了便于评估，并行循环 4:2 压缩树仅基于循环 4:2 压缩器、模加器进行搭建。实际实现时，并行循环 4:2 压缩树可利用 CSA3:2 压缩器对小于 4 输入的部分进行压缩，从而进一步优化面积和时序。

时序方面，当输入数据的位宽 n 和模数的位宽 m 相同时，基于三种结构的余数生成逻辑在关键路径上的主要构成单元层数均会减少，层数减少的比例基本一致。但模加器的延迟会随着模数位宽的增大而增加。因此，CSA3:2 压缩树、并行循环 4:2 较模加器树的余数生成逻辑能减少延迟开销，能达到 O($\log m$)数量级别的时序优化。由于并行循环 4:2 压缩和两层 3:2 压缩效果一致，但 4:2 压缩延迟更少，并行循环 4:2 压缩树较 CSA3:2 压缩树可减少 $1.2\ln\frac{n}{2m}$ 的单位门延迟。

面积方面，由于每个模加器的面积随模数位宽呈 O($m\log m$)的趋势增长。模加器树总面积随模数位宽呈 O($\log m$)的趋势增长，而 CSA3:2 压缩树、并行循环 4:2 压缩树中压缩器部分总面积能基本保持不变。因此，单位门模型下并行循环 4:2 较模加器树面积优化效果能达到 O($\log m$)数量级。并行循环 4:2 和 CSA3:2 压缩面积开销一致，但并行循环 4:2 压缩树延时更少，有更大的面积优化可能。

4.2 余数域浮点乘加校验算法与结构

浮点乘加余数校验总体结构如图 5 所示，浮点数乘加运算主通路、校验通路均由指数运算部分和尾数运算部分组成。FMA 运算主通路指数运算部分主要为加减运算，校验通路部分指数运算可在直接在余数域进行相应运算。

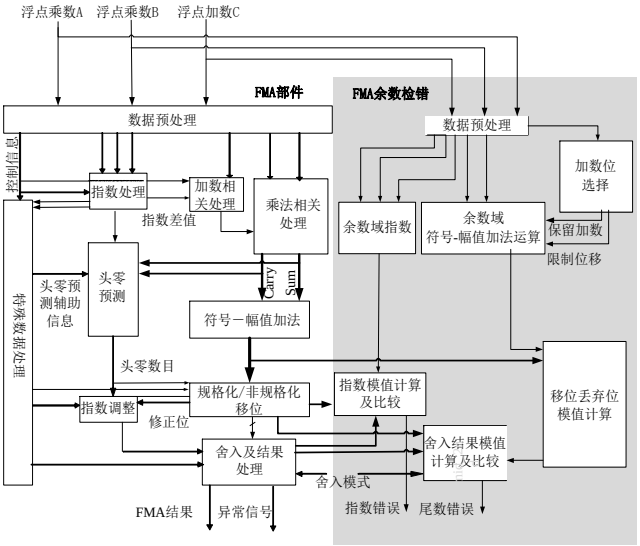

图5 浮点乘加余数校验总体结构示意图

表2. 余数生成逻辑的单元门模型延迟、面积评估

	模加器树	CSA3:2 压缩树	并行循环 4:2 压缩树
路径总延迟	$(2\log m+5)\left\lceil\log\frac{n}{m}\right\rceil\tau$	$\left(4\left\lceil\log_{1.5}\frac{n}{2m}\right\rceil+(2\log m+5)\right)\tau$	$\left(6\left\lceil\log\frac{n}{2m}\right\rceil+(2\log m+5)\right)\tau$
面积	$\left(11m+1.5m\log m+1.5\mathrm{m}\log(m-1)+2^{\log(m-1)-1}-2\right)\left(\frac{n}{m}-1\right)\alpha$	$\left(7\mathrm{m}\left(\frac{n}{m}-2\right)+11m+1.5m\log m+1.5\mathrm{m}\log(m-1)+2^{\log(m-1)-1}-2\right)\alpha$	$\left(14\mathrm{m}\left(\frac{n}{2m}-1\right)+11m+1.5m\log m+1.5\mathrm{m}\log(m-1)+2^{\log(m-1)-1}-2\right)\alpha$

$$\begin{cases}\left(\left(M_A\times M_B\times 2^3+SUB\right)+\left(M_C-T_C\right)\times 2^{111-ASC}\oplus\sum_{i=1}^{163}SUB\times 2^i+ST-\sum_{i=0}^{163}tm_i\times 2^i\right)\times 2^{N\_SHT} & \text{if } SUB\times Q=1\\ \left(\left(M_A\times M_B\times 2^3\right)\oplus\sum_{i=1}^{163}SUB\times 2^i+SUB\times 2+\left(M_C-T_C\right)\times 2^{111-ASC}+ST\oplus SUB-\sum_{i=0}^{163}tm_i\times 2^i\right)\times 2^{N\_SHT} & \text{otherwise}\end{cases} \tag{13}$$

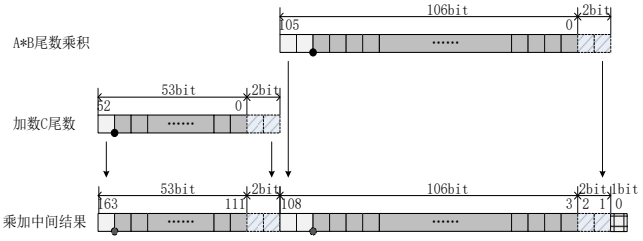

图6 乘加尾数中间结果格式

FMA 运算主通路中尾数运算部分由于位数占比高，运算操作复杂，是双精度浮点乘加运算中时序和面积的主要消耗部分。尾数乘法逻辑和移位逻辑在浮点 FMA 部件中面积占比较大，相应余数校验逻辑的设计需要针对乘法、移位操作进行开销优化。FMA 运算主通路中需要对乘数尾数、被乘数尾数和符号扩展后对阶移位的加数尾数进行运算。为了更好地优化主通路时序，通常在华莱士树中对乘法尾数部分积与加数尾数进行特定排布后再进行融合计算，并对融合运算结果进行规格化移位等操作。面向运算主通路时序的融合操作不利于余数部件的开销优化。

由于指数面积开销占比小，运算简单，本文余数运算逻辑主要介绍 FMA 尾数运算设计。针对余数域中 FMA 尾数运算的研究中，提出了取反符号扩展操作、乘法尾数、加法尾数的余数域加速变换。通过对尾数融合运算进行分割变换、缩减余数生成输入数据位宽、限定位移范围，对 FMA 校验逻辑的面积开销进行优化。

乘加操作尾数中间结果格式如图 6 所示，乘加尾数中间结果共 164 位。双精度浮点乘加操作中，对于浮点尾数乘法，其乘积结果为 106 位，其中小数点前有 2 位，小数点后有 104 位；对于加法操作，需要对齐尾数阶码。由于尾数乘积延迟较长，无论乘积指数域和加数指数域相对大小如何，可以采取加数向乘积对齐方式处理策略。其中，加数 C 尾数置于 A 与 B 尾数乘积左 56 位（以小数点所在位置衡量），确保若需移动时加数只需往同一个方向移动（即向右移）；加数 C 尾数与 A 与 B 尾数乘积之间额外添加 2 位，分别对应兼容 IEEE 754 标准中 guard 位和 round 位，确保对阶后加数尾数减 A 与 B 尾数乘积操作时结果尾数精度达到 53 位；同理，A 与 B 尾数乘积右边扩充 2 位，分别对应兼容 IEEE 754 标准中 guard 位和 round 位，确保 A 与 B 尾数乘积减对阶后加数尾数时结果尾数精度达到 53 位。最低位保留一位兼容 IEEE 754 标准中的 sticky 位。

规格化移位后的尾数计算内部结果如式(13)所示，记浮点数 A 的隐藏位为 H_A，浮点数 A 的尾数为$M_{\mathrm{A}}=H_{A\times 2^{53}}+\sum_{\mathrm{i}=0}^{52}ma_i\times 2^{\mathrm{i}}$，浮点数 B 的隐藏位为 H_B，浮点数 B 的尾数为$M_{\mathrm{B}}=H_{B\times 2^{53}}+\sum_{\mathrm{i}=0}^{52}mb_i\times 2^{\mathrm{i}}$，浮点数 C 的隐藏位为$H_C$，浮点数 C 的尾数为$M_C=H_C\times 2^{53}+\sum_{\mathrm{i}=0}^{52}mc_i\times 2^i$，浮点数 C 因对齐移位丢弃的尾数为$T_{\mathrm{C}}=\sum_{\mathrm{i}=0}^{53}\mathrm{tc}_{\mathrm{i}}\times 2^{\mathrm{i}}$，浮点数 C 的对齐移位数为 ASC，满足 $ASC\le 163$，幅值相减标记为 SUB，sticky 位为 ST。A × B尾数幅值大于 C 移位对齐后尾数幅值的标记为 Q，浮点尾数中间结果因规格化/非规格化移位丢弃部分为$T_m=\sum_{\mathrm{i}=0}^{162}tm_i\times 2^{\mathrm{i}}$，规格化/非规格化移位的位数为 N_SHT。下面将对浮点乘加尾数结果的余数码加速求解算法进行描述。

4.2.1取反符号扩展数的余数域加速变换

浮点乘加运算过程中会出现尾数幅值相减的情况，相应余数码生成可以基于取反的扩展尾数进行计算。余数生成逻辑的时序及面积开销与输入的位宽紧密相关，由于浮点乘加运算中涉及大量的位数扩展操作，扩展尾数取反结果的取模运算将带来较大的时序及面积开销。

由于位数扩展前的余数生成逻辑开销较小，本文利用将余数域中取反符号扩展操作变换为取模再模减的组合操作，从而对符号扩展操作的余数运算逻辑进行优化。具体推导如定理 2 所示，对于位数长度为 N 的二进制无符号数$X=\sum_0^{N-1}x_i\times 2^i$，$\overline{X}$ 的值等同于$2^N\text{-}1\text{-}X$。因此，尾数部分计算中$\overline{X}$的余数码$\left|\overline{X}\right|_M$值和$\left|\left|2^N\text{-}1\right|_M\text{-}\left|X\right|_M\right|_M$相同，后一种计算可以基于取模再模减的组合操作实现。取模再模减组合操作实现的余数码计算可基于扩展位数前的尾数，因此可以大幅减少余数计算输入的位数，从而优化计

算时间。所以当出现扩展尾数取反操作时，进行相应处理利用先取模再模减的组合运算进行求解。模减操作可以利用非门结合模加器进行实现。

定理 2 ： 设 $M=2^m-1$，对于位数长度为 N 的二进制无符号数X，有

$$\left|\bar{X}\right|_M=\left|\left|2^N-1\right|_M+\overline{\left|X\right|_M}\right|_M \tag{14}$$

证明.

$$\begin{aligned}&\left|\bar{X}\right|_M\\&=\left|2^N-1-X\right|_M\\&=\left|\left|2^N-1\right|_M-\left|X\right|_M\right|_M\\&=\left|\left|2^N-1\right|_M+\left|M\right|_M-\left|X\right|_M\right|_M\\&=\left|\left|2^N-1\right|_M+\left|\left|2^m-1\right|_M-\left|X\right|_M\right|_M\right|_M\\&=\left|\left|2^N-1\right|_M+\left|\overline{\left|X\right|_M}\right|_M\right|_M\\&=\left|\left|2^N-1\right|_M+\overline{\left|X\right|_M}\right|_M\end{aligned} \tag{15}$$

证毕.

以浮点乘法部分余数域运算为例，当 SUB=1、Q=0 时，$\left|\left((M_A\times M_B\times 2^3)\oplus\sum_{i=1}^{163}SUB\times 2^i\right)+SUB\times 2\right|_M$ 部分可以表示为 $\left|(M_A\times M_B\times 2^3)\oplus\sum_{i=1}^{163}2^{i+2}\right|_M$。根据定理 2，该部分可做如下式(16)的变换。余数生成逻辑最大位宽由 163 位优化为 53 位，该变换还可利用乘法尾数余数域压缩对乘法开销优化为 FMA 主通路对应部分的 2809/m^2 分之一。

$$\begin{aligned}&\left|\left(M_A\times M_B\times 2^3\right)\oplus\sum_{i=1}^{163}2^i+2\right|_M\\&=\left|2\times\left(2^{163}-1-\left(M_A\times M_B\times 2^2\right)\right)+2\right|_M\\&=\left|2^{164}-M_A\times M_B\times 2^3\right|_M\\&=\left|2^{|164|_m}-\left|\left(M_A\times M_B\times 2^3\right)\right|_M\right|_M\\&=\left|2^{|164|_m}+\sum_{i=0}^{m-1}2^i\oplus\left|\left(M_A\times M_B\times 2^3\right)\right|_M\right|_M\end{aligned} \tag{16}$$

对于浮点数尾数 C 运算，当 SUB=1、Q=1 时，$\left|(M_C-T_C)\times 2^{110\text{-}ASC}\oplus\sum_{i=1}^{163}SUB\times 2^i+ST+SUB\right|_M$部分可以表示为$\left|(M_C-T_C)\times 2^{111\text{-}ASC}\oplus\sum_{i=1}^{163}2^i+ST+1\right|_M$，根据定理 2，结合加数尾数余数域压缩变换，可以将余数生成逻辑最大位宽由 163 位优化为 53 位。同时避免了 163 位大范围移位带来的时序及面积开销，移位器的面积开销可优化为 FMA 主通路对应部分的$\frac{163}{m}\log\frac{163}{m}$分之一。

$$\begin{aligned}&\left|\left(M_C-T_C\right)\times 2^{111\text{-}ASC}\oplus\sum_{i=1}^{163}2^i+ST+1\right|_M\\&=\left|\left(2^{164}-1-\left(\left(M_C-T_C\right)\times 2^{111\text{-}ASC}+ST\oplus 1\right)\right)+1\right|_M\\&=\left|2^{164}-\left(\left(M_C-T_C\right)\times 2^{111\text{-ASC}}+ST\oplus 1\right)\right|_M\\&=\left|2^{|164|_m}-\left|\left(M_C-T_C\right)\times 2^{111\text{-}ASC}+\mathrm{ST}\oplus 1\right|_M\right|_M\\&=\left|2^{|164|_m}+\sum_{i=0}^{m-1}2^i\oplus\left|\left(M_C-T_C\right)\times 2^{111\text{-}ASC}+\mathrm{ST}\oplus 1\right|_M\right|_M\end{aligned} \tag{17}$$

4.2.2乘法尾数余数域加速变换及校验结构

基于华莱士树乘法的面积开销与乘数位宽的平方呈正比，FMA 运算主通路采用融合计算，不利于直接利用余数域乘法交换律进行时序及面积优化。不同于 FMA 运算主通路，为了有效利用余数域乘法分配率降低乘法操作硬件开销，本文简化乘法部分对应二进制码，将 SUB 指示位、ST 位的影响从乘法结果的余数码计算中剥离。

其中，乘法部分主要运算可以表示为 $M_A\times M_B\times 2^3$。本文中取$M=2^m-1$，根据性质 2，当 2^3的幂能被 m 整除时，乘法对应部分余数码等式成立

$$\begin{aligned}&\left|\left(M_A\times M_B\right)\times 2^3\right|_M\\&=\left|\left|M_A\right|_M\times\left|M_B\right|_M\right|_M\end{aligned} \tag{18}$$

这种情况下乘法对应部分余数码等于浮点数 A 的尾数取模 $\left|H_A\times 2^{53}+\sum_{i=0}^{52}ma_i\times 2^i\right|_M$和浮点数 B 的尾数取模$\left|H_B\times 2^{53}+\sum_{i=0}^{52}mb_i\times 2^i\right|_M$的模乘结果。此时，能利用式(18)大幅加速余数运算速度。针对乘法面积开销与乘数位宽的平方呈正比的特点，乘法尾数余数域压缩变换能对乘法面积开销优化为 FMA 运算主通路相应部分的$\frac{n^2}{m^2}$分之一。

更一般地，当2^3的幂不能被 m 整除时，可在同时在等式(18)两边同时乘以$2^{addzero}$，$addzero$取$m-|3|_m$，从而利用性质 1、性质 2 进行划分并快速求解余数。该处理同时需要对浮点 FMA 余数域运算的其他部分进行乘$2^{addzero}$操作。

$$\begin{aligned}&\left|\left(M_A\times M_B\right)\times 2^{3+addzero}\right|_M\\&=\left|\left|M_A\times M_B\right|_M+\left|0\right|_M\right|_M\\&=\left|\left|M_A\right|_M\times\left|M_B\right|_M\right|_M\end{aligned}\tag{19}$$

结合取反符号扩展数和乘法尾数余数域加速变换，可以构造 FMA 余数校验中尾数乘法部分余数运算逻辑，逻辑框图如图 7 虚线右端所示。图 7 左侧为 FMA 主通路乘法部分示意，图中白色虚线框部分是该功能等效需要忽略的部分，灰色虚线框是为了功能等效所增加的示意模块。依据式(19)，FMA 余数校验尾数乘法基础运算为 M_A、M_B 取模再模乘的结果。当 SUB=1、Q=0 时，根据式(16)的变换，对模乘结果取反，再复用加数模数域运算部分进行模加 $2^{|164+addzero|}$运算后可得余数域 FMA 乘法结果。其余情况下，余数域 FMA 乘法结果为 FMA 余数校验尾数乘法基础运算结果。

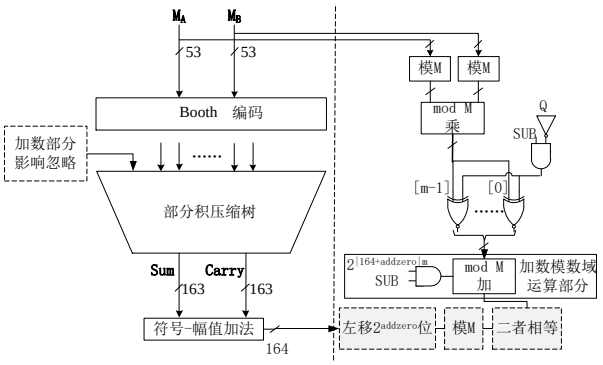

图7 尾数乘法部分余数校验结构示意图

4.2.3加数尾数余数域加速变换及校验结构

移位器是浮点 FMA 部件时序和面积开销的重要组成部分。对于输入位宽和最大移位位宽均为 n 的对数移位器，面积开销呈 O($n\log n$)趋势增长。加数 C 和乘法结果的尾数的对齐位移量为 ASC，ASC 最高可达 163 位。对阶移位还大幅增加了加数尾数部分的位宽，导致较大的余数生成逻辑开销。因此，需要利用余数域变换对移位器开销及余数生成开销进行优化。

双精度浮点数 C 尾数对应部分(M_C-T_C)×$2^{111-ASC}$，由性质 5、性质 6，C 尾数在余数域可有如下变换：

$$\begin{aligned}&\left|\left(M_C-T_C\right)\times 2^{111-ASC}\right|_M\\&=\left|\left|\left(M_C-T_C\right)\times 2^{111}\right|_M\times 2^{m-|ASC|_m}\right|_M\end{aligned}\tag{20}$$

由性质 4，有

$$\begin{aligned}&\left|\left(M_C-T_C\right)\times 2^{111}\right|_M\\&=\left|\left|M_C-T_C\right|_M\times 2^{|111|_m}\right|_M\end{aligned}\tag{21}$$

由式(21)，式(20)可以变换为

$$\begin{aligned}&\left|\left|\left(M_C-T_C\right)\times 2^{111}\right|_M\times 2^{m-|ASC|_m}\right|_M\\&=\left|\left|M_C-T_C\right|_M\times 2^{|111|_m}\times 2^{m-|ASC|_m}\right|_M\end{aligned}\tag{22}$$

依据式(22)，浮点数 C 尾数对应余数码的求解中，首先分别计算$\left|(H_C\text{-}tc_{53})\times 2^{53}+\sum_{i=0}^{52}(mc_i\text{-}tc_i)\times 2^i\right|_M$、$2^{|111|_m}$ 和$\left|2^{m-|ASC|_m}\right|_M$，再对这三部分做模乘运算即可。C 尾数保留位和等效左移位数(m-|ASC|$_m$)由加数 C 预处理模块生成，$2^{|111|_m}$ 为常量，可提前生成。以上变换可将移位器面积开销优化为 FMA 主通路相应部分的$\frac{163}{m}\log\frac{163}{m}$分之一，同时将余数生成逻辑输入位宽由 163 位缩减至 53 位。

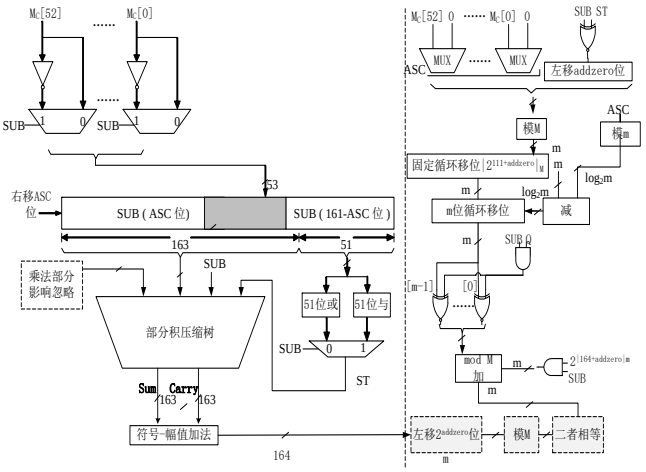

图8 尾数加数部分余数校验结构示意图

结合取反符号扩展数和乘法尾数余数域加速变换，可以构造 FMA 余数校验中尾数加法部分余数运算逻辑。图 8 为尾数加数部分余数校验结构示意，虚线左端为 FMA 主通路尾数加数运算部分，虚线右端为 FMA 余数校验尾数加数部分。图 8 中白色虚线框为该功能等效需要忽略的部分，灰色虚线框是为了功能等效所增加的示意模块。依据式(22)，FMA 余数校验尾数加数基础运算需要经过选择、拼接、取模、固定循环移位 $2^{|111+addzero|}$、左移 (m-|ASC|$_m$)操作。SUB=1、Q=1 时， 需要对 FMA 余数校验尾数加数基础运算结果取反，再与 $2^{|164+addzero|}$ 模加，$2^{|164+addzero|}$可以与 FMA 余数校验尾数加数基础运算并行计算。

4.2.4余数域浮点乘加舍入运算校验

浮点的舍入模式可以分为就近舍入(RN,Round to Nearest)、向无穷大舍入(RI, Round to Infinity)、截断舍入(RZ, Round to Zero)。RN、RI、RZ 三种舍入模式的说明如下[31]：

RN 模式下选取最接近真实“无限精度”结果最的可表示值作为舍入结果。若存在两个值与真实“无限精度”结果具有同等近似度，则选取 LSB(Least Significant Bit)为 0 的值作为舍入结果。

RI 舍入模式可以分为向正无穷大舍入(RPI, Round to Positive INFINITY)、向负无穷大舍入(RNI, Round to Negative INFINITY)。RPI 舍入时，则选取与真实“无限精度”结果靠近的两个浮点数中较大的值作为舍入结果；RNI 舍入时，则选取与真实“无限精度”结果靠近的两个浮点数中的较小值。RPI、RNI 舍入模式下，尾数部分只需要做正无穷大舍入或截断处理。

RZ 模式选取与真实“无限精度”结果靠近的两个可表示值中绝对值较小者作为舍入结果。尾数做截断处理即可。

表3. RN、RI、RZ 的理论正确值

K L R ST	RN	RPI+/RNI-	其他
X 0 00	*X* 0	X0	*X* 0
X 0 01	*X* 0	X1	*X* 0
X 0 10	*X* 0	X1	*X* 0
X 0 11	*X* 1	X1	*X* 0
X 1 00	*X* 1	X1	*X* 1
X 1 01	*X* 1	(X+1)0	*X* 1
X 1 10	(*X*+1)0	(X+1)0	*X* 1
X 1 11	(*X*+1)0	(X+1)0	*X* 1

RN、RI、RZ 舍入的组合情况如表 3 所示，其中，第一列 *K*、*L*、*R*、*ST* 表示尾数最低 2 位、舍入位和粘贴位在舍入操作前的值；第二列 RN 表示就近舍入时尾数最低 2 位的理论正确值；第三列 RPI+/RNI-表示 RPI 模式符号为正、RNI 符号为负时尾数最低 2 位的理论正确值；第四列表示 RZ 等其余情况时尾数最低 2 位的理论正确值。

综上所述，根据具体的舍入模式及尾数数值，舍入逻辑仅可能会对规格化移位后的尾数进行加 1、加 2 处理。由于浮点 FMA 余数域的运算中会利用等效位移的对余数域运算进行加速，导致余数生成部分会出现低位补零部分。舍入修正可复用等效位移低位补零部分的输入，当出现舍入修正时，可直接将舍入修正运算并入丢弃位等余数生成的计算中，通过复用余数生成逻辑优化时序及面积开销。

5 实验结果及分析

本文基于 Verilog-2005 进行硬件设计实现，利用 Synopsys 公司的 DCG(Design Compiler Graphical)综合方式在 28nm 工艺下进行综合，DCG 版本号为 0-2018.06-SP3。实验分别选择基于模加器树、CSA3:2 压缩树、并行循环 4:2 混合压缩树的一种具体余数生成逻辑对双精度 FMA 校验逻辑进行实现及评估，对不同结构下的 FMA 校验逻辑时序、检错率进行了对比和分析。其中，并行循环 4:2 混合压缩树为主体基于循环 4:2 压缩，使用少量 CSA3:2 和模加器的余数生成逻辑。

5.1 双精度尾数余数生成逻辑对比

由于尾数余数生成逻辑是 FMA 余数校验逻辑的重要组成部分，本文首先对这部分逻辑进行时序和面积的评估。实验结果表明，单位门模型、DCG 综合结果下，基于并行循环 4:2 混合压缩结构的余数生成逻辑均优于基于模加器树、CSA3:2 压缩结构的余数生成逻辑。

图 9 是单位门模型下双精度尾数余数生成逻辑的延时比较，纵坐标是延迟评估，单位为单位门延迟 τ；横坐标是模数位宽 *m* 的取值。模数位宽 *m* 相同时，并行循环 4:2 混合压缩结构的延迟均优于其他结构。模数位宽 *m* 增加后，由于并行循环压缩中压缩树层数不增，压缩树部分的延迟可以降低。*m*=6 的并行循环 4:2 混合压缩结构较 *m*=3 的并行循环 4:2 混合压缩结构延迟减少 19%，*m*=8 的并行循环 4:2 混合压缩结构较 *m*=4 的并行循环 4:2 混合压缩结构延迟减少 15%。

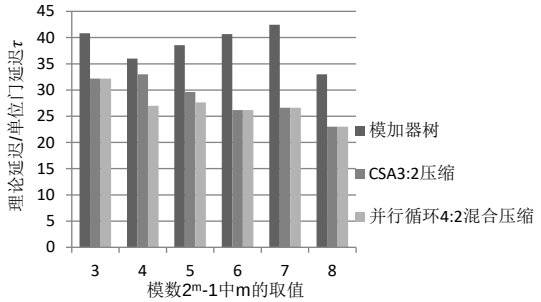

图9 各模数下 53 位尾数的余数生成逻辑理论延时比较

图 10 是 DCG 综合流程下各模数下双精度尾数的余数生成逻辑的延时比较，纵坐标是对应结构的延迟，单位为 ns；横坐标是相应模数位宽 *m* 的取值。所有模数下的尾数余数生成逻辑中 *m* 为 6 的并行循环 4:2 混合结构时序最优。模数位宽 *m* 相同时，三种结构中并行循环 4:2 混合结构的延迟较优，较模加器树结构平均优化 15%，m=6 时最多可取得

19.64%的时序优化。并行循环 4:2 混合结构较 CSA3:2 压缩树结构平均优化 6%，m=5 时最多可优化 6.75%的时序。

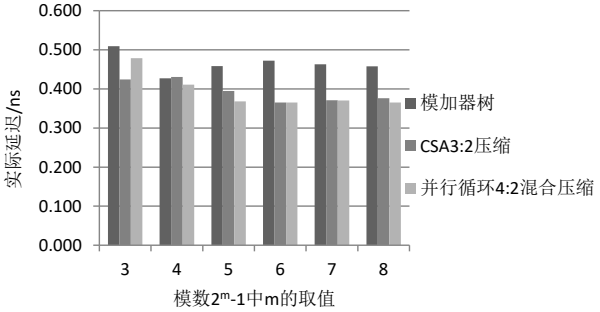

图10 各模数下 53 位尾数的余数生成逻辑实际延时比较

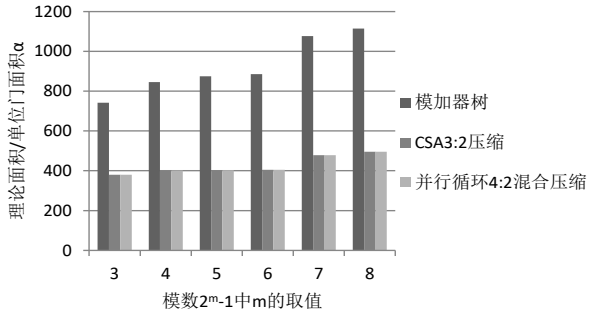

图11 各模数下 53 位尾数的余数生成逻辑理论面积

图 11 是单位门模型下双精度尾数在采用不同模数时余数生成逻辑的面积比较，纵坐标为相应结构的单位门面积评估，单位为单位门面积α；横坐标是相应模数位宽 *m* 的取值。*m*=3 时并行循环结构的面积开销最低。模数位宽 *m* 相同时，CSA3:2 压缩、并行循环 4:2 混合压缩结构的面积较优。*m* 增加时，模加器树中模加器个数会减少，但模加器面积呈 O(*mlogm*)的增加趋势，模加器树结构总面积呈上升趋势。当 *m* 增加时，由于并行循环结构压缩树部分的面积呈降低趋势，所以并行循环结构面积开销的增长幅度较模加器树要更为缓和。

图 12 是 2.5GHz 频率下双精度尾数在采用不同模数时余数生成逻辑的面积比较，图中纵坐标是对应结构的面积，单位为 um^2；横坐标是相应模数位宽 *m* 的取值。并行循环 4:2 混合压缩树结构在各模数下面积均较优，较模加器树结构面积平均优化 69%，m=6 时优化最高可达到 71%。并行循环 4:2 混合压缩较 CSA3:2 压缩树结构面积平均优化 5%，m=4 时最多可取得 18.18%的面积优化。面积变化趋势和单位门模型下的评估基本一致。

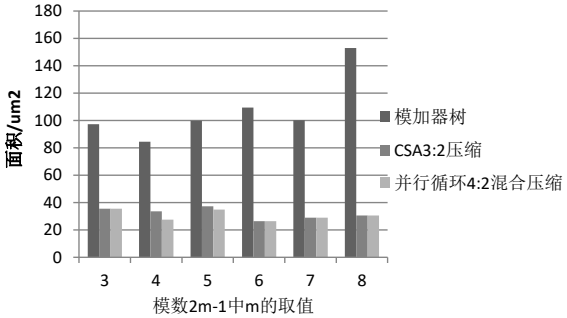

图12 各模数下 53 位尾数的余数生成逻辑实际面积

5.2 各模数FMA余数校验部件对比

本文还对各模数取值下浮点 FMA 校验部件的进行了测试对比。基于并行循环 4:2 混合压缩的 FMA 校验部件较在面积开销上均优于其他结构。

图 13 为各模数下浮点 FMA 校验部件各部分面积占比，FMA 余数校验逻辑根据功能可以分为余数生成逻辑、余数乘法逻辑、数据预处理及其他逻辑。

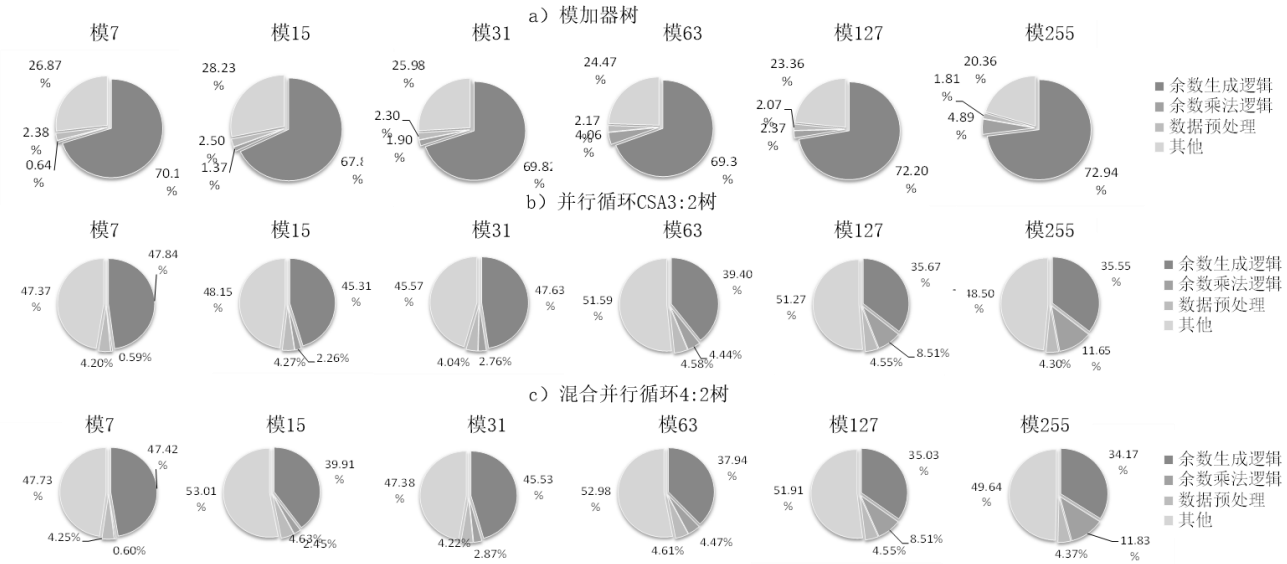

图13 各模数下浮点 FMA 校验部件各部分面积占比

表4. 各模数下 FMA 余数校验部件对比

模数 2^m-1	检错率	面积(um^2)			FMA 面积占比		
		模加器树	CSA3:2 压缩	并行循环 4:2 混合压缩	模加器树	CSA3:2 压缩	并行循环 4:2 混合压缩
7	87.50%	4453.48	2705.66	2675.56	15.32%	9.31%	9.20%
15	93.75%	4270.86	2665.80	2477.64	14.69%	9.17%	8.52%
31	96.88%	4582.67	2802.18	2697.64	15.77%	9.64%	9.28%
63	98.44%	5006.82	2503.40	2479.42	17.23%	8.61%	8.53%
127	99.22%	4817.83	2517.58	2517.07	16.57%	8.66%	8.66%
255	99.61%	5602.63	2648.47	2611.83	19.27%	9.11%	8.99%

三种结构下余数生成逻辑占比均较大，模加器树、CSA3:2 压缩、并行循环 4:2 混合压缩的余数生成逻辑平均占比分别达到 68.80%、41.90%、39.80%。随着模数位宽 *m* 的增加，三种结构的余数乘法逻辑占比均不断加大。

模加器树结构中，余数生成逻辑占比随 *m* 的增加不断增多，由 70.11%增长至 72.94%。余数乘法逻辑占比虽不断增加，但最多仍只占 4.89%的 FMA 校验部件面积。

由于并行循环 4:2 混合压缩结构和 CSA3:2 压缩结构采用压缩树替换了模加器树，2.5GHz 频率下面积反而随 *m* 的增加而下降。模数位宽 *m* 增加后，并行循环 4:2 混合压缩和 CSA3:2 压缩的余数生成逻辑面积占比呈降低趋势。

表 4 对各模数下 FMA 余数校验部件的检错率、面积大小、FMA 面积占比进行了对比。余数校验部件的检错率由模数 2^m-1 决定，检错率为$\frac{2^m-2}{2^m-1}$[22]。即模数越高时，检错率越高，本文评估参数中模数取 255 时检错率最高，为 99.61%。

面积方面，基于模加器树、CSA3:2 压缩、并行循环 4:2 混合压缩的 FMA 校验逻辑的 FMA 面积平均占比分别为 16.48%、9.08%、8.86%。虽然本文面向时序和面积开销，在余数域进针对浮点乘加运算进行了变换，但由于模加器树的面积开销对 *m* 敏感，随着模数位宽 *m* 的增加，基于模加器树的 FMA 余数校验的面积增长明显，面积占比由 15.63%增长至 19.65%。

虽然 CSA3:2 压缩树、并行循环 4:2 混合压缩树结构中余数生成部分随 *m* 的增加面积稍有降低，但余数乘法部分的面积随 *m* 的增加而增大，部分抵消了余数生成的面积优化，因此，CSA3:2 压缩树、并行循环 4:2 混合压缩树结构的 FMA 校验面积呈轻微波动态势。其中，基于并行循环 4:2 混合压缩树的模 15 余数校验面积最低，仅占 FMA 面积的 8.52%。

5.3 FMA检错能力和系统可用性分析

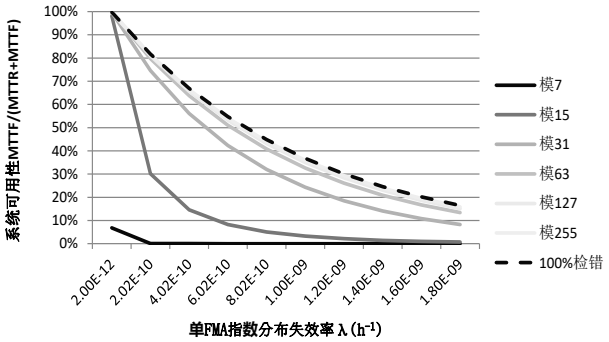

图14 单 FMA 检错率与系统可用性关系

假设单 FMA 失效、软件回卷恢复时间均符合指数分布，单 FMA 恢复率 μ=1000h^{-1}，对于包含 10^9 个 FMA 的 E 级超算系统，可以建立仅考虑 FMA 错误影响的非完全检错模型[32]。不同单 FMA 失效率 λ 下的系统可用性如图 14 所示，模 7 到模 255 为采用相应余数检错时系统的可用性，100%检错代表单 FMA 完全检错时系统的可用性。虽然模 7、模 15 的单 FMA 检错能力分别能达到 87.50%、97.35%，但相应系统的可用性较差。λ 取值范围在$2\times10^{-12}$$h^{-1}$到$1.8\times10^{-9}$$h^{-1}$时，模 7、模 15 对应系统较完全检错系统的可用性平均降低 46.92%、31.13%。在 λ 大于$2.02\times10^{-10}$$h^{-1}$后，模 31 系统与 FMA 完全检错系统的可用性差距变大。在当前系统的 FMA 规模设置下，模 63、模 127、模 255 校验对应系统与 FMA 完全检错系统的可用性基本一致。综合以上观察可以得知，提升 FMA 校验检错能力对提升系统可用

性方面具有重要意义。

6 总结

单 FMA 检错能力对 E 级超算的系统可用性影响较大。E 级超算芯片的错误检测设计需要在保证错误检测能力的前提下，针对检错逻辑的时序、面积进行优化，从而为芯片及系统提供高效的容错支撑。

本文提出了并行循环 4:2 混合压缩结构，该结构能在模数增大的情况下优化余数生成逻辑的时序及面积开销，并提升 FMA 余数校验检错率。本文还针对浮点乘加运算特点，提出了取反扩展尾数、尾数乘法、加数尾数的余数域加速变换，对 FMA 校验的时序和面积开销进行了优化。本文提出的 FMA 校验能支持多种浮点舍入模式的检查，通过复用余数生成的低位补零部分，能对规格化移位结果的舍入修正和余数求解进行并行操作。

实验结果表明，并行循环 4:2 混合压缩余数生成较模加器树余数生成、CSA(Carry Saved Adder)3:2 压缩余数生成最多可取得取得 19.64%、6.75%的时序优化和 71%、18.18%的面积优化。相比基于模加器树、CSA3:2 压缩树的 FMA 余数校验部件，基于并行循环 4:2 混合压缩结构的 FMA 余数校验部件在时序开销和面积开销上均能得到优化。综合考虑面积开销、检错率和系统可用性，基于并行循环 4:2 混合压缩树的模 63 余数校验部件较优，其 FMA 面积占比为 8.53%，检错率达到 98.44%，对系统可用性的支撑能力接近采用完全检错的校验逻辑。基于并行循环 4:2 混合压缩树的模 63 余数校验在面积开销、检错率、系统可用性上均优于 IBM 采用的模 15 余数校验。本文提出的 FMA 校验逻辑具备对 FMA 部件进行运算实时校验的能力，可以有效保障 FMA 部件及 E 级超算的可靠运行。目前，该设计已应用于新一代国产神威超级计算机。

作者贡献声明：高剑刚，刘骁二人对本文具有同等贡献，均为第一作者。

参 考 文 献

[1] Qian DePei, Wang Rui. Key issues in exascale computing. Scientia Sinica: Informationis, 2020, 50(09): 1303-1326 (in Chinese)
(钱德沛, 王锐. E 级计算的几个问题.中国科学:信息科学, 2020, 50(09): 1303-1326)

[2] Lucas R , Ang J , Bergman K , et al. Top ten exascale research challenges. Washington D.C., USA: U.S. Department of Energy Advanced Scientific Computing Advisory Subcommittee: technical report: 1222713, 2014.

[3] Gao JianGang，Gong Daoyong. Power management technology for exascale computing. Chinese Journal of Computers, 2022, 45(7): 1373-1383 (in Chinese)
（高剑刚，龚道勇. 面向 E 级计算的功耗管理技术. 计算机学报, 2022, 45(7): 1373-1383)

[4] Liu C , He X , Liang B , et al. Detailed placement for pulse quenching enhancement in anti-radiation combinational circuit design. Integration-the VLSI Journal, 2018, 62(6):182-189.

[5] Cai Shuo, Kuang Ji-Shun. Reliability estimation for soft error of sequential circuit based on error propagation probability matrix. Chinese Journal of Computers, 2015, 38(5): 923-931 (in Chinese)
(蔡烁，邝继顺. 基于差错传播概率矩阵的时序电路软错误可靠性评估. 计算机学报, 2015, 38(5): 923-931)

[6] Gong Rui, Guo Yu-Feng. Quantitative evaluation metric and methodology for microprocessor soft error tolerance design. Journal of National University of Defense Technology, 2017, 39(3): 64-68 (in Chinese)
(龚锐，郭御风. 微处理器容软错误设计量化评估指标及评估方法 国防科技大学学报, 2017, 39(3): 64-68)

[7] A. A. Wahba and H. A. H. Fahmy, Area efficient and fast combined binary/decimal floating point fused multiply add unit. IEEE Transactions on Computers, 2017, 66(2): 226-239

[8] Maniatakos M , Kudva P , Fleischer B M , et al. Low-cost concurrent error detection for floating-point unit controllers. IEEE Transactions on Computers, 2013, 62(7):1376-1388.

[9] Eibl P J , Cook A D , Sorin D J . Reduced precision checking for a floating point adder// Proceedings of the International Symposium on Defect & Fault Tolerance in VlSI Systems. Chicago, USA, 2009: 145-152

[10] Zhang Y, Nathan R, Sorin D J. Reduced precision checking to detect errors in floating point arithmetic. arXiv preprint arXiv:1510.01145, 2015

[11] D. Lipetz and E. Schwarz, Self checking in current floating-point units// Proceedings of the Symposium on Computer Arithmetic, Tuebingen, Germany, 2011: 73-76

[12] Dao S T, Haess J G, Kroener M K, et al. Distributed residue-checking of a floating point unit, USA, 2013.10.22

[13] Choquette J, Gandhi W, Giroux O, et al. NVIDIA A100 tensor core

GPU: Performance and innovation. IEEE Micro, 2021, 41(2): 29-35.

[14] NVIDIA. NVIDIA H100 tensor core GPU architecture. Santa Clara, USA: NVIDIA, technical report: V1.01, 2022

[15] ARM. Reliability, availability, and serviceability, for Armv8-A. Cambridge, UK: ARM, technical report: D.c, 2021

[16] Jaume Abella ,et al. Security, reliability and test aspects of the RISC-V ecosystem// Proceedings of the European Test Symposium (ETS). Bruges, Belgium, 2021: 1-10

[17] K. Seetharam, L. C. T. Keh, R. Nathan and D. J. Sorin, Applying reduced precision arithmetic to detect errors in floating point multiplication// Proceedings of the Pacific Rim International Symposium on Dependable Computing, Vancouver, BC, Canada, 2013: 232-235

[18] Kito N , Akimoto K , Takagi N . Floating-point multiplier with concurrent error detection capability by partial duplication. Ieice Transactions on Information & Systems, 2017, 100(3):531-536.

[19] Lo J C . Reliable Floating-point arithmetic algorithms for error-coded operands. IEEE Transactions on Computers, 1994, 43(4):400-412.

[20] Shuji Yamamura. A64FX: 52-core processor designed for the 442PetaFLOPS supercomputer Fugaku// Proceedings of the International Solid State Circuits Conference. San Francisco, USA, 2022: 352-354.

[21] Haess J , Kroener M K , Mueller S M , et al. Residue-based exponent flow checking, USA, 2013.12.19

[22] W. A. Chren, One-hot residue coding for high-speed non-uniform pseudo-random test pattern generation// Proceedings of the International Symposium on Circuits and Systems. Seattle, USA, 1995: 401-404

[23] Tukur Gupta, Shamim Akhter. Design and implementation of area-power efficient generic modular adder using flagged prefix addition Approach// Proceedings of the 7th International Conference on Signal Processing and Communication. Noida, India, 2021: 302-307

[24] Li L , Zhou L , Zhou W . An improved architecture for designing modulo (2n-2p+1) multipliers. IEICE Electronics Express, 2012, 9(14):1141-1146.

[25] R. A. Patel, M. Benaissa and S. Boussakta, Fast modulo 2^{n} - (2^{n - 2} + 1) addition: A new class of adder for RNS, IEEE Transactions on Computers, 2007, 56: 572-576

[26] E. Vassalos and D. Bakalis, Residue-to-binary converter for the new RNS moduli set {22n−2, 2n−1, 2n+1}// Proceedings of the Panhellenic Conference on Electronics & Telecommunications . Volos, Greece, 2019: 1-4

[27] P. Patronik and S. J. Piestrak, Design of residue generators with CLA/compressor trees and multi-bit EAC// Proceedings of the 8th Latin American Symposium on Circuits & Systems. Bariloche, Argentina, 2017: 1-4

[28] Chip-Hong Chang, Jiangmin Gu. Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits. IEEE Transactions On Circuits And Systems, 2004, 51(10): 1985-1997

[29] Wang M , Liu D , Liu M , et al. A two-item floating point fused dot-product unit with latency reduced. IEICE Electronics Express, 2016, 13(23):20160937-20160937.

[30] R. Abhilash, I. B. K. Raju, G. Chary and S. Dubey, Area-power efficient vedic multiplier using compressors// Proceedings of the International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO). Visakhapatnam, India, 2015: 1-5

[31] G. Even and P. Seidel, A comparison of three rounding algorithms for IEEE floating-point multiplication// Proceedings of the 14th IEEE Symposium on Computer Arithmetic. Adelaide, Australia, 1999: 225-232

[32] Rui Peng and Huadong Mo. Optimal structure of multi-state systems with multi-fault coverage. Reliability Engineering and System Safety, 2013, 119:18-25

Gao Jiangang, M.S., senior engineer. His main research interests include high performance computing and computer architecture

Liu Xiao, M.S., engineer. His research interests include high performance computing and computer architecture.

Zheng Fang, Ph.D., associate professor. His research interests include high performance computing and computer architecture.

Tang Yong, M.S., associate professor. His research interests include high performance computing and computer architecture.

Background

This work focuses on reliability improvement techniques for floating-point fused multiply-add unit in high performance exascale supercomputers.

With the continuous expansion of the system scale, reliability of exascale supercomputer is facing increasingly serious challenges. As the core unit of high-performance computing systems, the reliability of arithmetic logic units on chip plays a vital role for sustainable high performance of the computing system. There exist several previous works focus on fault tolerant techniques for arithmetic logic unit. For example, time redundant techniques, execution unit redundant techniques and arithmetic error detecting codes. Researches on arithmetic error detecting codes are widely studied, such as parity code, checksum code, parity-based linear code, Berger code and Residue code.

However, there are few researches focusing on error detection techniques in floating-point fused multiply-add unit. Reduced precision arithmetic-based checking, partial duplication and Moduli 15 residue system based on modular adder are techniques for floating-point fused multiply-add unit. The timing cost, error detection coverage and hardware overhead are disadvantages of previous works.

In this paper, we design a parallel cyclic compression-based error-detection residue code for floating-point fused multiply-add unit. In the design of residue code based on parallel cyclic compression, under the premise of getting better error detection coverage of FMA unit, timing cost and area cost are also taken into consideration. The parallel cyclic compression structure optimizes the timing overhead and area overhead caused by redundant result corrections of multi-layer modular adders. Compared with the residue generation logic based on modular adder tree, the residue generation logic based on parallel cyclic compression can achieve the optimization of $O(\log m)$ both in timing cost and area cost. Moreover, we propose the residue domain compression technology for mantissa multiplication, mantissa addition and inverse extended mantissa. By using these techniques, the area overhead of shift logic and multiplication logic can be reduced by 10 times on average. Compared with generally-used Moduli 15 residue system based on modular adder, the moduli 63 FMA checker based on parallel cyclic 4:2 compressor reduces area by 67.61%, yields 5% error coverage improvement and yields up to 49.6% exascale supercomputer's availability improvement.

Our group has been working on the high-performance computing and processors design. We have developed several designs for active and passive fault tolerant techniques on home-grown heterogeneous many-core processor architecture. Such as low-cost (72,64) RS code in memory that is able to correct four symbol errors and independent and coordinated lightweight error recovery technique.