高技术通讯 2026 年 第 36 卷 第 8 期:823 - 835

doi:10.3772/j.issn.1002 - 0470.2026.08.005

HyEA-Gen:超弹材料柔性结构计算加速器生成框架①

耿国帅②*** 李早霞***

(*中国科学院计算技术研究所智能计算机研究中心 北京 100190)

(**中国科学院大学 北京 100049)

(***安阳师范学院物理与电气工程学院 安阳 455000)

摘 要 刚性和柔性结构的混合使用是机器人设计中的一个重要趋势,而柔性结构模型的计算复杂度带来的控制实时性问题限制了刚柔混合机器人的发展。本工作针对刚柔混合机器人中柔性结构模型的计算加速需求,设计了专用加速器自动生成框架 HyEA-Gen,兼顾了计算模型灵活性和计算性能。该框架由两部分构成:其一为高层次综合器,用于从高层次语言(如 Matlab,C ++ 等)生成硬件加速器设计;其二为领域知识优化器,用于利用领域知识对电路结构进行优化,加速性能。通过覆盖广泛应用领域的多种柔性材料模型的实验测试,验证了设计方法的正确性和先进性。相比通用中央处理器(central processing unit,CPU)平台和 Vitis HLS(high-level synthesis)生成的现场可编程门阵列(field-programmable gate array,FPGA)实现,HyEA-Gen 分别实现了 208.8 × 和 14.1 × 的加速比。

关键词 刚柔机器人;本构模型;加速器生成框架;领域知识优化

在刚性机器人中引入柔性结构已经成为机器人设计的重要趋势。柔性关节的引入可以提高机器人的运动能力[1-2],柔性连杆的引入可以提高机器人的操控能力[3-4]。柔性结构的引入扩展了机器人的任务能力和应用场景[5-6]。

柔性结构的精确控制需要用本构模型进行计算,相比刚体结构的运动模型极大地增加了计算量。本构模型用来描述柔性单元的受力和形变之间的关系,常表现出强烈的非线性特征[7]。柔性结构的材料存在多样性,包括橡胶、硅胶等有机弹性体,形状记忆合金,介电弹性体,液晶弹性体等[8]。不同的柔性材料需要采用不同的本构模型用来描述其多种多样的非线性应力应变关系。随着新材料的不断发展和创新,材料的计算模型也在不断更新。

为保证柔性结构的灵活性,柔性材料通常选择超弹性材料,例如采用具有大形变能力的软硅胶以保证柔性手指的操控能力[4]。超弹材料具有非线性的应力应变关系,其本构模型具有强烈的非线性特征。计算模型的非线性导致了计算复杂性的提升,使柔性结构的计算延时非常高。例如,文献[9]中仿生爬虫的一个步态周期计算,在 Intel i9-11900K 上运行,需要耗时 30 min。柔性结构的计算实时性差,严重限制了机器人中柔性结构的开发和应用[10]。

本工作针对通用中央处理器(central processing unit,CPU)平台上柔性材料本构模型的计算能力不足的问题开展了研究工作。针对计算过程中大量存在的非线性计算,例如幂函数算子、对数算子等,进行了非线性算子专用加速设计,并结合领域知识进行电路优化,实现了专用加速器生成框架,并实验验证了计算性能的提升。本工作具有以下 3 个贡献点。

① 北京市科技计划(Z241100004224015)资助项目。

② 男,1991 年生,博士生;研究方向:领域专用加速器自动生成;联系人,E-mail:gengguoshuai19b@ict.ac.cn。

(收稿日期:2025 - 03 - 06)

(1) 非线性算子并行实现设计。通过对非线性算子对浮点表示格式中符号位、指数位和尾数位的作用分析,设计了高效地展开式并行计算实现,有效提高了非线性计算负载的计算性能。

(2) 领域专用加速器生成框架。用户通过高级语言进行编程,通过该框架快速生成加速器实现。面对材料模型不断更新的趋势,提高了专用加速器的开发效率。

(3) 基于领域知识的优化。生成框架中利用领域知识进行电路的稀疏性和并行性优化,实现了通用生成工具难以达到的计算性能。

1 研究背景

1.1 刚柔混合机器人

刚柔混合机器人已经成为机器人设计的重要发展趋势。合理的混合结构设计可以兼顾刚性材料和柔性材料的优势,显著提升机器人的运动灵活性和适应性[1,3]。

利用柔性结构可以提高机器人的灵活性。首先,柔性结构执行动作时整个柔性连续体中的所有微观部分都会产生形变,这不同于刚性结构执行时运动只能发生于宏观的连杆单元之间,因此柔性结构可以满足任务执行时更复杂的形状变化需求。其次,柔性材料通常比刚性材料更轻,可以降低结构重量,增加结构移动速度。例如,利用串联弹性单元构成的柔性爪,可以实现复杂形状物体的灵活抓取,其最大负载达到柔性爪自身重量的 260 倍[3]。而且,柔性材料具有储能特性,通过施加预形变进行预充能,可以提高柔性结构的瞬时输出功率,有效提升机器人的运动能力。例如,通过在多节刚性腿中添加被动柔性结构,可以实现更好地行走、跳跃和飞行起降性能,同时提高能源利用率[1]。

利用柔性结构可以提高机器人与人类的交互适应性。柔性材料的被动形变特点,常被用于人机交互机器人中被动柔顺结构的设计。例如,研究人员通过在亚毫米柔性光纤骨架上喷涂厚度仅为 100 μm 的多功能弹性皮肤,实现了微创手术机器人探头的主动运动控制,以支持对狭窄的病理区域(如肺末端支气管)的成像检测[11]。基于编织技术的柔性机器人可以按需设计柔性结构的各向异性力学特性,实现舒适轻便的可穿戴医疗康复设备[12]。

1.2 超弹材料本构模型

为保证机器人运动的灵活性和交互性,柔性结构通常选取超弹材料[13]。超弹材料的非线性应力-应变关系,通常用本构模型表示。本构模型是基于力学理论建立的计算模型,能够较好地反映材料形变的物理机制[7]。本构模型的建立需要深入考虑材料的微观物理特性(如晶格结构)和变形机制,通过合适的物理理论(如热激活理论和位错动力学理论)将材料的微观特性与宏观力学性能联系起来,推导出描述材料应力-应变关系的计算模型[14]。模型通常采用不同的应变能密度函数来描述超弹材料的力学行为。在模型计算中,基于应变能密度函数的应力更新和应变更新是主要计算任务。

本构模型的计算是柔性结构性能仿真评估的主要计算负载。有限元法是柔性结构高精度性能评估的主要方法。在有限元法的计算过程中,柔性结构被划分为有限个网格单元。为满足精度需求,网格单元通常数量极大。每个网格单元都需要执行本构模型计算,以迭代求解的形式更新单元状态[15]。

本构模型的计算通常具有强烈的非线性特征,导致计算复杂性的提升。首先,本构模型通常选取合适的应变不变量进行构造。应变不变量的计算本身涉及到非线性计算,例如幂函数计算、矩阵特征值计算等。其次,本构模型的计算流中存在大量的非线性算子,以描述材料的非线性应力-应变关系,例如对数、指数、幂函数的计算等。以描述极大应变材料特性的 Ogden 模型的应变能密度函数(式(1))的计算为例,一个六面体网格单元的一次计算至少需要 32 次幂函数计算。

由于本构模型建立时采用的物理理论不同,导致其计算流程差别很大。因此难以设计一种专用电路适合所有的本构模型计算加速。在机器人柔性结构设计时,本构模型需要根据应用需求进行选择,例如适用于小应变和中等应变场景的 Neo-Hookean 模型,适用于大应变场景的 Yeoh 模型,适用于极大应变场景的 Ogden 模型,适用于压缩性材料的 Blatz-

Ko 模型等[16-17]。而且随着先进力学性能的柔性材料的不断发展,本构模型也在不断更新和发展。因此,领域专用加速器生成器是提高机器人柔性结构开发效率的合适途径。

1.3 专用加速器

1.3.1 机器人专用加速器

机器人领域的快速发展和应用,使机器人计算加速器设计成为一个重要研究方向。根据机器人计算负载类型,相关研究工作可以分为以下几类:以 SPADE[17]、ParallelNN[18] 等为代表的 3D 点云语义感知加速器,以 Fusion-3D[19]、GauSPU[20] 等为代表的视觉建图与定位加速器,以 RTSA[21]、MOPED[22] 等为代表的运动规划加速器,和以 RoboShape[23]、Dadu-RBD[24] 等为代表的刚性关节运动控制加速器。本文的工作面向机器人中柔性结构运动计算负载的加速,而之前的工作仅面向机器人中刚性结构运动计算负载的加速。

1.3.2 加速器专用生成框架

通过领域专用生成框架来弥补高层次设计者和专用加速硬件之间的知识代沟,以促进专用硬件的敏捷开发,已经成为一种发展趋势。例如,Stellar[25]、Rubick[26] 等,专注于空间计算架构。ImaGen[27]、HeteroHalide[28] 等,专注于图像处理领域。相比之下,本文的工作专注于一个相对较新但重要的领域:机器人柔性结构力学计算模型的加速器生成。HyEA-Gen 通过分析领域特有的稀疏性和并行性知识,并将其应用于电路优化,是第一个关注了刚柔混合机器人高精度控制过程的硬件加速器工作。

2 研究动机

超弹性材料的力学分析,通常使用有限元法,即将整个材料分为有限个离散的小网格单元。每个小单元在进行分析时,又会调用对应的本构模型。这一计算过程十分耗时。例如,对于文献[9]中的仿生爬虫,采用有限元法计算一个运动步态周期过程所需的时间大于 30 min。为了缓解计算效率低的问题,本文首先根据实际任务进行了计算瓶颈的测试和分析;然后分析了现有加速器生成工具的不足,有针对性地提出了本研究的设计思路。

2.1 计算瓶颈测试与分析

本研究首先确定了本构模型是实际计算任务中的主要计算负载,然后确定了非线性计算是导致本构模型计算耗时增加的主要原因。

2.1.1 本构模型计算耗时占比

为了确定实际计算任务的计算瓶颈,本文对表 1 中 9 个柔性机器人计算任务进行了测试,并统计了计算过程中计算子任务的耗时占比。

表 1 测试的计算任务说明

#	任务描述	#	任务描述
T1	柔性气腔加压	T6	多腔体气动手指抓取
T2	柔性球落体弹跳	T7	柔性体极限拉拽分析
T3	柔性体一点施压	T8	柔性体夹取状态分析
T4	柔性杆动态碰撞	T9	分段多腔体气动爬行
T5	柔性体界面施压		

图 1 显示了在不同测试任务中,本构模型计算在整体计算任务中的耗时占比。统计数据显示,本构模型计算在测试任务整体计算时间中的比例在 50.0% ~70.0%之间,为耗时最严重的部分。因此,对超弹材料的本构模型计算进行加速对整体计算性能的提升最为重要。

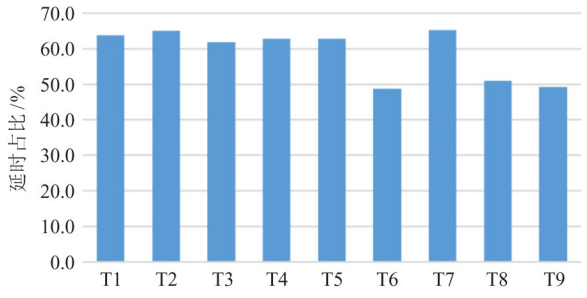

图 1 不同计算任务中本构模型的延时占比

2.1.2 非线性计算耗时占比

为了加速超弹材料本构模型的计算,我们进一步对多种本构模型的计算耗时进行了测试和分析。选取了覆盖不同应用场景的 9 种本构模型的应力计算作为测试任务。本构模型的类型编号和主要数学计算形式,如表 2 所示。

针对表 2 中计算任务的结构不规则性和非线性特征,本文对本构模型的计算延时进行了标量代数

算子级别的细粒度的统计。通过分析计算任务，将计算任务中将涉及到的算子分为线性和非线性 2 大类，如表 3 所示。在任务测试过程中，对两类算子计算造成的耗时分别进行了统计。

表 2　不同材料本构模型的编号及关键计算任务

材料	关键计算任务
M1	$\mu\left(\lambda^2-\frac{1}{\lambda}\right)$
M2	$2\left(\lambda^2-\frac{1}{\lambda}\right)\left[C_{10}+\frac{C_{01}}{\lambda}\right]$
M3	$2\left(\sum_{i=1}^{3} iC_{i0}(I_1-3)^{i-1}\right)\left(\lambda^2-\frac{1}{\lambda}\right)$
M4	$\mu\left(\lambda^2-\frac{1}{\lambda}\right)\frac{J_m}{J_m-\left(\lambda^2+\frac{2}{\lambda}-3\right)}$
M5	$\mu\left(1+\frac{b}{n}\left(\lambda^2+\frac{2}{\lambda}-3\right)\right)^{n-1}\left(\lambda^2-\frac{1}{\lambda}\right)$
M6	$2\left(\lambda^2-\frac{1}{\lambda}\right)C_1C_2\left[\mathrm{e}^{C_2(I_1-3)}\right]$
M7	$2\left(\lambda^2-\frac{1}{\lambda}\right)C_1C_2\left[\mathrm{e}^{C_2(I_1-3)}-\frac{1}{2\lambda}\right]$
M8	$\sum_{k=1}^{N}\frac{2\mu_k}{\alpha_k}\left[\lambda^{\alpha_k}-\left(\frac{1}{\sqrt{\lambda}}\right)^{\alpha_k}\right]$
M9	$\mu\lambda_{\max}^4\frac{\lambda^3-1}{(\lambda\lambda_{\max}^2-1)(\lambda_{\max}^2-\lambda^2)}$

表 3　本构模型中的标量代数算子分类

线性计算	$+$, $-$, $\times$, $\boldsymbol{AB}$, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, $\mathrm{tr}(\boldsymbol{A})$, $\det(\boldsymbol{A})$
非线性计算	$\frac{1}{a}$, $\frac{a}{b}$, $\ln(a)$, e^a, $\sqrt{a}$, a^b

*　a、b 为标量变量，$\boldsymbol{A}$、$\boldsymbol{B}$ 为矩阵变量。

** $\mathrm{tr}(\boldsymbol{A})$ 为矩阵的迹，$\det(\boldsymbol{A})$ 为矩阵的行列式。

测试结果如图 2 所示，计算任务中非线性算子的计算延时普遍在 50.0% 以上，最高可达 90.0%。且数据显示出非线性计算延时的增加是模型整体延时

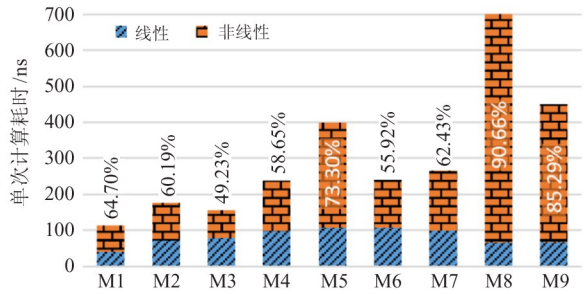

图 2　不同模型计算总耗时和非线性耗时占比

增加的主要因素。因此，对计算模型中的非线性算子进行加速对本构模型计算性能的提升尤为重要。

2.2　动机与思路

2.2.1　专用生成框架

因为材料模型多种多样，且不断发展更新，为某一种材料模型专门设计专用加速器，会导致可复用性差。需要自动生成框架，提高专用加速器的开发效率。

现有生成框架，生成的实现计算性能差。本文在通用 CPU 计算平台上测试了 Matlab 和 C ++2 种实现，并基于通用高层次综合工具 Vitis HLS 生成了现场可编程门阵列（field-programmable gate array，FPGA）的加速器实现，并对三者的计算性能进行了测试。测试结果如图 3 所示，在 CPU 计算平台上，C ++ 实现的计算性能优于 Matlab，而 Vitis HLS 生成的实现相比 C ++ 实现的优化效果有限。

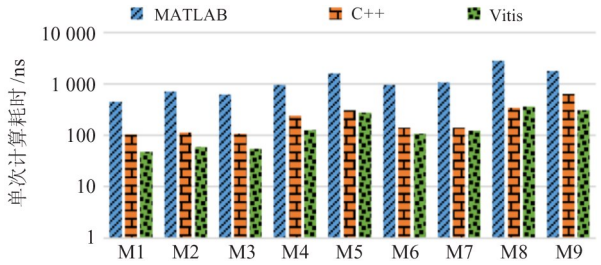

图 3　Vitis HLS 加速效果

2.2.2　领域知识的利用

本构模型计算任务中的领域知识分为稀疏性与并行性 2 类。稀疏性知识分为材料特性导致的数据稀疏性和计算稀疏性，以及算法导致的数据稀疏性。并行性知识来源于有限元方法算法中各计算单元之间计算的并行性，以及针对非线性负载特性的并行性加速方法。稀疏性意味着有效计算量少于实际计算量，并行性意味着不同计算可以同时发生，两者都可以显著提高加速器的性能。因此，在电路生成过程中，利用领域知识去提取稀疏性和并行性，并将二者应用来进行电路优化可以显著提高计算性能。下文将具体描述如何对稀疏性、并行性进行提取和利用。

领域稀疏性。材料模型的物理特性，例如方向性、可压缩性等，包含了本构模型中数据结构的稀疏

性知识。模型中的基本数据结构的元素由基本力学参数构成。通用材料建模需要 21 个非 0 的独立力学参数。然而,对于正交各向异性材料,例如复合材料[29],由于材料特性,某些方向上的运动自由度被限制为 0,在 21 个独立力学参数中,只有 9 个是非 0 的,这会导致基本数据结构中的稀疏性。对于不可压缩材料,例如橡胶,本构模型计算过程中的中间变量:形变梯度的行列式 $J = \det(\boldsymbol{F})$ 和第三应变不变量 $I_3 = \det(\boldsymbol{C})$ 的值恒等于 1,可以对相关计算过程进行剪枝,提高计算性能。通用生成工具如 Vitis HLS 等,在生成过程中未利用这些领域知识,难以达到最优计算性能。

领域并行性。柔性结构的计算,存在 2 种不同级别的并行性。第一,有限元算法中的并行性。这种并行性通过在 Vitis 中添加编译优化指令可以利用。例如,单元网格的每个顶点都有相应的本构模型计算任务,网格类型的不同会导致不同的计算并行性。通用生成工具可以通过添加 pragma 的形式利用这种粗粒度的并行性。第二,本构模型计算中的并行性。这种并行性无法在 Vitis 中被利用到,对于离散单元中的细粒度并行和非线性负载难以提供足够的优化。本构模型的非线性是计算性能的主要限制因素,利用专用的展开式并行化加速方法,可以有效提升计算性能。通用生成工具在生成过程中未利用这些领域知识,难以达到最优计算性能。

3 生成框架概述

为快速开发面向超弹材料本构模型计算的专用加速器,本文开发了专用加速器生成框架 HyEA-Gen,以兼顾开发过程的可编程性和硬件加速器的计算性能。HyEA-Gen 的开发流程如图 4 所示。

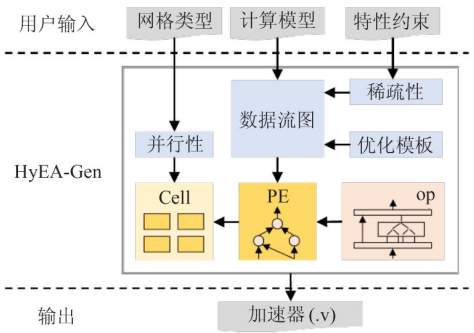

图 4 生成框架流程示意图

用户输入为计算模型文件、材料特性文件和网格类型文件。在有限元计算中,每个离散网格单元拥有多个网格顶点。在一次网格单元状态更新中,每个网格顶点都需要进行本构模型的计算。模型文件用于描述应力和应变之间的计算关系。模型计算单元(processing element,PE)用于执行模型文件描述的计算模型。网格类型文件包含了每个网格单元的顶点个数,揭示了每个网格单元一次状态更新中的最大 PE 并行度。因此,网格类型文件决定了每个网格计算单元(Cell)中的 PE 数量。材料特性文件用于描述材料特性的各向异性和体积压缩性等,这些特性包含了计算模型的稀疏性信息,可以用于电路优化。

图 4 中,每个 PE 包含一个或者多个基础算子(op),每个 Cell 又包含一个或者多个 PE,三者呈一个层次化的关系。

生成工具首先根据模型文件生成数据流图,根据材料特性进行数据流图的稀疏性优化。HyEA-Gen 提供了非线性计算优化模板用于计算流图的优化;针对计算负载的非线性特性,设计了非线性算子并行加速器(op),作为数据流图调用的基础算子;为提高 PE 的吞吐量,设计了基于关键非线性算子的自动流水线设计流程。针对网格顶点之间的计算并行性,在 Cell 单元中进行了 PE 的并行部署。针对有限元方法离散网格之间的计算并行性,在最终生成的专用加速器中包含多个并行执行的 Cell 单元。

4 计算负载分析

超弹材料的本构模型是柔性结构计算任务的主要负载,如图 1 中的数据所示。本节对计算特征进行详细分析,为专用加速方法的设计提供指引。

模型计算的非线性是最为显著的特征,这是由其表达非线性应力-应变关系的需求决定的。本构模型通常以其应变能密度函数 $\Psi(\boldsymbol{U})$ 来表示,其中 $\boldsymbol{U}$ 包括材料参数和应变不变量。以 Ogden 模型的应变能密度函数为例,其表达式为

$$\Psi(\boldsymbol{U}) = \sum_{k=1}^{N} \frac{2\mu_k}{\alpha_k^2}\left((\lambda_1^*)^{\alpha_k} + (\lambda_2^*)^{\alpha_k} + (\lambda_3^*)^{\alpha_k} - 3\right) + \sum_{k=1}^{N} \frac{1}{D_k}(J-1)^{2k} \tag{1}$$

式中：$\boldsymbol{U}=[\boldsymbol{\lambda}^*, J, \boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{D}]$ 为模型输入；$\boldsymbol{\lambda}^*=[\lambda_1^*, \lambda_2^*, \lambda_3^*]$ 和 J 为应变不变量；$\boldsymbol{\alpha}$、$\boldsymbol{\mu}$ 和 $\boldsymbol{D}$ 为材料参数；N 为模型参数。式(1)明显显示出了计算结构中强烈的非线性。

此外，输入的应变不变量的计算也具有非线性特征。以 $\boldsymbol{\lambda}^*$ 为例，$\boldsymbol{\lambda}^*$ 的计算如式(2)所示。

$$\boldsymbol{\lambda}^* = \det(\boldsymbol{F})^{-\frac{1}{3}}\mathrm{eig}(\boldsymbol{F}) \tag{2}$$

式中：$\boldsymbol{F}$ 为 3×3 的形变梯度矩阵，$\det(\boldsymbol{F})$ 为 $\boldsymbol{F}$ 的行列式，$\mathrm{eig}(\boldsymbol{F})$ 为 $\boldsymbol{F}$ 的特征值。此处，$\mathrm{eig}(\boldsymbol{F})$ 的计算可以采用解析法，具体地等效为一个三次方程求根问题的非线性计算过程实现。

根据应变能密度函数进行应力计算是本构模型的关键任务。应力计算过程中需要计算应变能密度函数对应变不变量的偏导数，如表 4 所示。这里偏导数的计算过程同样也具有强烈的非线性特征。

表 4　基于不同参数的计算模型示例

应变不变量	应力计算模型
$\boldsymbol{F}$	$\frac{1}{J}\frac{\partial\Psi(\boldsymbol{F})}{\partial\boldsymbol{F}}\boldsymbol{F}^T$
$\boldsymbol{C}$	$\frac{2}{J}\boldsymbol{F}\frac{\partial\Psi(\boldsymbol{C})}{\partial\boldsymbol{C}}\boldsymbol{F}^T$
I_1, I_2, I_3	$\frac{2}{J}\left[\left(\frac{\partial\Psi}{\partial I_1}+I_1\frac{\partial\Psi}{\partial I_2}\right)\boldsymbol{b}-\frac{\partial\Psi}{\partial I_2}\boldsymbol{b}^2+I_3\frac{\partial\Psi}{\partial I_3}\boldsymbol{I}\right]$
I_1, I_2, J	$\frac{2}{J}\left[\left(\frac{\partial\Psi}{\partial I_1}+I_1\frac{\partial\Psi}{\partial I_2}\right)\boldsymbol{b}-\frac{\partial\Psi}{\partial I_2}\boldsymbol{b}^2\right]+\frac{\partial\Psi}{\partial J}\boldsymbol{I}$
$\lambda_1, \lambda_2, \lambda_3$	$\sigma_i=\frac{\lambda_i}{J}\frac{\partial\Psi}{\partial\lambda_i}$

通用 CPU 平台上浮点数的非线性运算采用数值迭代的方式计算，导致计算周期长。另一方面，本构模型中的非线性算子不规则地分散在计算流图中，导致难以利用 CPU 硬件流水级结构的高吞吐率优势，因为这会在流水线中引入大量空泡。Vitis HLS 对此的优化效果也很有限。当通过 HLS C ++ 调用通用数学库进行综合时，仍会实现为数值迭代计算电路，导致计算周期长。即使调用 FPGA 内部专用硬件算子进行加速，也仍是多时钟周期算子，如表 5 所示，且对于多样化的本构模型而言非线性算子的支持不完整。因此，专用的非线性算子加速器的设计是本构模型加速器设计的重要环节之一。

表 5　非线性算子最优延时[30]

算子	div	log	exp	sqrt	rec
延时/时钟周期	29	23	21	29	30

5　非线性算子并行加速

2.1 节对本构模型的计算瓶颈进行了测试，结果显示对非线性计算的加速是整体性能提升的关键。

数字电路中的高精度计算通常采用浮点数表示，浮点数非线性算子的计算模式主要有数值迭代形式和展开式形式。在 CPU 平台上，非线性标量算子的计算，通常通过多周期数值迭代方式实现。在 Vitis HLS 平台上，直接通过原生 C ++ 代码进行高层次综合进行实现时，同样以多周期迭代方式实现。通过 pragma 指定硬件优化方案，可以进行一定程度的优化，但是仍然无法实现性能最优。表 5 所示为 Vitis HLS 平台上支持的非线性算子，以及达到性能最优时的延时。

非线性算子的数值迭代方法，每 2 次迭代之间存在数据依赖关系，难以实现迭代之间的并行化加速，导致计算延时受限于迭代次数。展开式方法本身适合进行并行化加速，然而对于浮点数的非线性计算而言，需要谨慎地处理展开方式和表示范围，以保证计算精度。下面介绍本工作的具体设计方法。

5.1　并行化方法

不同于定点数表示下函数计算可以直接使用展开式方法，浮点数计算使用展开式方法时，需要根据表示格式和算子类型进行设计。首先，需要根据非线性算子对浮点表示格式中各部分的作用效果设计合适的展开式目标函数 $f(\theta)$。然后，根据展开式形式设计展开式参数生成电路和计算电路。最后，根据算子类型设计对输出结果的格式处理电路。

以单精度浮点数的倒数算子为例进行设计流程说明。

步骤 1

浮点数的表示格式由符号位 s、指数位 K 和尾数位 m 3 部分组成，单精度浮点数 x 的表示格式如图 5 所示。

x	=	s	K	m
31:0		31	30:23	22:20

图 5　单精度浮点数表示格式

其数值大小由 3 部分共同构成,表示如下。

$$x = s \cdot 2^k \cdot (1 + m),\ m \in [0,1) \tag{3}$$

式中:真实的指数值 k 等于指数位表示值 K 减去偏移量 127,即 $k = K - 127$。因此,倒数算子 $rec(x) := \frac{1}{x}$ 的计算:

$$\begin{aligned} rec(x) &= rec(s,K,m) \\ &= s \cdot 2^{-k} \cdot \frac{1}{1+m},\ m \in [0,1) \end{aligned} \tag{4}$$

根据式(4),倒数算子的展开式目标函数 $f(\theta)$ 设计为

$$f(\theta) := f(m) = \frac{1}{1+m},\ m \in [0,1) \tag{5}$$

此时,$\theta = \Theta_{rec}(s,K,m) = m_{\circ}\Theta$ 表示从 (s,K,m) 到 θ 的某种映射,根据非线性算子类型决定。

步骤 2

以泰勒展开式形式对目标函数 $f(\theta)$ 进行展开为例,其表达式为

$$f(\theta) = \sum_{i=0}^{n} \frac{f^i(\theta_0)}{i!}(\theta - \theta_0)^i + R_n(\theta) \tag{6}$$

式中:θ_0 为展开点,n 为展开阶数,$R_n(\theta)$ 为展开式的拉格朗日余项,其表达式为

$$R_n(\theta) = \frac{f^{n+1}(\xi)}{(n+1)!}(\theta - \theta_0)^{n+1},\ \xi \in [\theta_0,\theta] \tag{7}$$

式中:n 指定了高阶项的保留项数,$(\theta - \theta_0)$ 的范围指定了分段精度。

结合式(5) 和式(6)可以得到倒数算子的泰勒展开式如下:

$$f(m) = \sum_{n\geqslant 0} \frac{(-1)^n}{(1+m_0)^{n+1}} \cdot (m - m_0)^n,\quad m_0 \in [0,1) \tag{8}$$

结合式(5) 和式(7)可以得到倒数算子的泰勒展开式的拉格朗日余项,其表达式为

$$|R_n(m)| = \left|\frac{f^{n+1}(\xi)}{(n+1)!}(m - m_0)^{n+1}\right| \leqslant \left|\frac{(m-m_0)^{n+1}}{(1+\xi)^{n+2}}\right|,\ \xi \in [m_0,m] \tag{9}$$

拉格朗日余项 $R_n(m)$ 以二进制表示时,最高位所在表示位数为

$$\log_2 |R_n(m)| \leqslant (n+1)\log_2(m - m_0) \tag{10}$$

当取 $n = 7, m - m_0 \leqslant \frac{1}{8}$ 时,$\log_2 |R_7(m)| \leqslant -24$。由于单精度浮点数的尾数位一共 23 位,此时,余项在单精度浮点表示下可忽略。

所以,将 m 在 $m_0 = \left\{0,\frac{1}{8},\frac{2}{8},\frac{3}{8},\frac{4}{8},\frac{5}{8},\frac{6}{8},\frac{7}{8}\right\}$ 这 8 个点处展开,保留到第 7 阶,即可保证单精度浮点数计算结果的精度。

步骤 3

针对步骤 2 设计的展开方式,需要设计合适的中间参数处理电路。以展开式计算为界限,之前的参数由输入处理电路产生,之后的参数由输出处理电路产生。图 6 展示了倒数算子的输入和输出处理电路。其中展开式目标函数 $f(\theta)$ 的计算由 Taylor 计算电路实现。

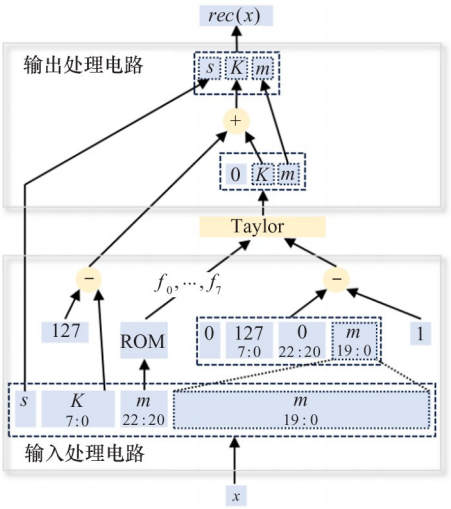

图 6　倒数算子电路示例

倒数算子的输出与输入的符号位相同,因此 x 的符号位直接传递到输出处理电路中作为最终输出的符号位。

$rec(x)$ 的指数位表示值为

$$K_{\frac{1}{x}} = k_{\frac{1}{x}} + 127 = -k_x + k_f + 127$$

$$= 127 - K_x + K_f \tag{11}$$

式中：K_x 为输入变量 x 的指数位表示值，K_f 为展开式目标函数 $f(\theta)$ 的计算结果的指数位表示值。图 6 所示电路中输入处理电路中计算式(11) 中的 $127 - K_x$ 部分，输出处理电路中计算剩余加法部分。

$rec(x)$ 的尾数部分等于展开式目标函数计算值的尾数部分，如图 6 输出处理电路所示。

由步骤 2 知，展开式的分段计算形式为

$$f(m) = \sum_{i=0}^{7} f_i \cdot \delta m^i \tag{12}$$

式中：$f_i = \dfrac{(-1)^i}{(1+m_0)^{i+1}}$，$\delta m = m - m_0$，$m_0 \in \left\{0, \dfrac{1}{8}, \dfrac{2}{8}, \dfrac{3}{8}, \dfrac{4}{8}, \dfrac{5}{8}, \dfrac{6}{8}, \dfrac{7}{8}\right\}$。如图 6 中输入处理电路中所示，$f_i$ 具体由尾数位最高 3 位作为地址读取 ROM 中的 f_i 常数值。δm 具体由 x 尾数位高 3 位置零、指数位置为 127、符号位置零构成的浮点数构成的浮点数减去 1 计算得到。

5.2　算子架构

以泰勒展开式为例，非线性算子加速架构如下图 7 所示，由 3 部分构成。(1)输入处理电路：根据算子类型进行预处理，获得展开式的输入参数；(2)展开式计算电路：在合适的范围内进行展开式的计算；(3)输出处理电路：调整展开式计算结果，结合预处理参数生成正确的结果表示形式。

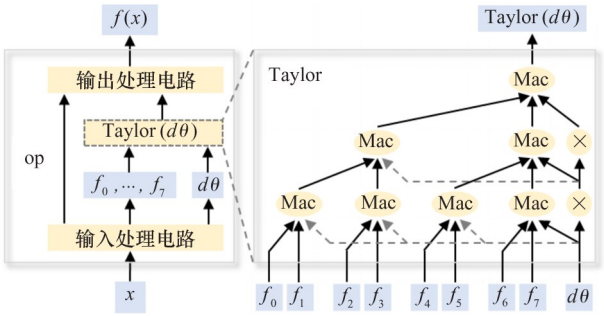

图 7　非线性算子加速架构示意图

在进行预处理的过程中，根据非线性算子对 s、K、m 的作用效果，设计合适的展开式目标函数，对 3 个部分处理后得到展开式目标函数 $f(\theta)$，其中处理过程中的映射 Θ：$(s,K,m) \mapsto \theta$ 根据 h 类型选择。

展开式计算电路通过并行化方法实现非线性计算的快速计算。此处以 Taylor 展开式形式为例，其基本计算形式为

$$\text{Taylor}(d\theta) = \sum_{i=0}^{n} f_i d\theta^i \tag{13}$$

针对该计算形式，基于奇偶分治的方法设计了树形并行计算结构，如图 7 中 Taylor 部分所示。

输出处理电路用于将展开式电路的计算结果调整为正确的算子表示结果。

6　稀疏性优化

本构模型具有丰富的力学性质表达能力，其代价是更多的计算复杂性。实际应用场景中，所选用的柔性材料往往具有某些方面的特殊性质，例如某些橡胶材料的不可压缩性、某些纤维材料的不可拉伸性等。这些材料特性对应了本构模型中特定的计算稀疏性，可以用来简化模型计算。此外，本文分析发现，计算过程中会产生中间变量的稀疏性，例如矩阵与其转置相乘结果的对称性，不变量 I_1 的计算仅依赖驱动数据的对角线元素等。本节介绍如何利用这些稀疏性简化本构模型的计算，提高计算性能。

6.1　材料特性导致的稀疏性

本构模型可以根据所选材料特性进行数据流图的优化，从而减少计算复杂度提高计算效率。

对于不可压缩材料，例如橡胶、生物软组织等，形变过程中体积不变，模型中的应变不变量保持不变 $J = I_3 = 1$。因此，模型计算过程中无需进行中间变量 $J = \det(\boldsymbol{F})$ 和 $I_3 = \det(\boldsymbol{C})$ 的矩阵行列式计算。相应地，I_i^*，λ_i^* 的计算也可以极大化简。例如，在式(1) 描述的通用 Ogden 模型的基础上，添加不可压缩特性后，可以得到计算复杂度更低的等价计算形式，如图 8 所示。

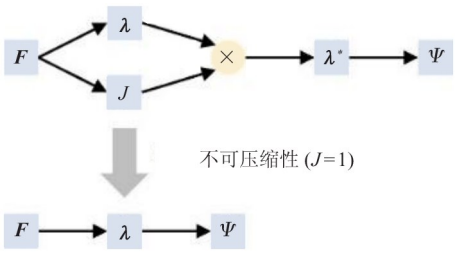

图 8　不可压缩性对 Ogden 模型的稀疏性简化

各向异性材料模型需要依赖 21 个独立参数。对于具有正交各向异性的材料,例如复合材料、木材等,仅需要 9 个独立参数。而各向同性材料,例如金属、玻璃等,形变过程中无需考虑方向依赖性,仅需 2 个材料参数。因此,利用材料的方向性特性,可以发现数据中的稀疏性,简化计算模型。

对于具有小形变约束的超弹材料,例如具有小范围形变的金属、陶瓷等,可以使用线性应变张量替代非线性应变张量,以简化计算模型。根据材料特性设置的不同,对相应参数进行稀疏性设置,并对计算模型进行剪枝,以提升计算性能。

6.2 算法导致的稀疏性

在本构模型的计算中,一些基本的变量及其计算过程具有稀疏性。例如,根据形变梯度矩阵 $\boldsymbol{F}$ 计算形变张量 $\boldsymbol{C}$ 的计算过程 $\boldsymbol{C}=\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ 显示出 $\boldsymbol{C}$ 的对称性。因此,$\boldsymbol{C}$ 中元素的计算和存储无需面向全部 9 个元素,只需面向 6 个元素。

应变不变量是从形变张量中提取的标量,反映了材料形变的内在特性。第一应变不变量 I_1 的计算过程为 $I_1=\mathrm{tr}(\boldsymbol{C})$,如果本构模型中只使用不变量 I_1,则计算 $\boldsymbol{C}$ 时只需要计算 3 个对角线元素。如图 9 中虚线指示的计算流程所示,编程语言描述的中间计算过程 $\boldsymbol{C}=\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ 通过通用矩阵乘法实现了 $\boldsymbol{C}$ 的元素值计算。而通过由 $\boldsymbol{C}$ 计算 I_1 的算法可以知道只需对 $\boldsymbol{C}$ 的 9 个元素值中的 3 个进行求解即可,如图中实线指示的计算流程所示。因此,在电路实现中将 $\boldsymbol{C}$ 的元素值计算算子由一个矩阵乘替换为 3 个向量乘,计算量减少 66.6%。

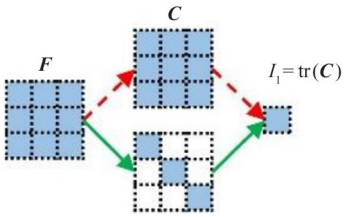

图 9 I_1 计算流程中的稀疏性示例

类似地,第二应变不变量 I_2 的计算 $I_2=\frac{1}{2}[\mathrm{tr}(\boldsymbol{C})^2+\mathrm{tr}(\boldsymbol{C}^2)]$,其中,$\mathrm{tr}(\boldsymbol{C})^2$ 的计算只需要 $\boldsymbol{C}$ 的对角线元素驱动,$\mathrm{tr}(\boldsymbol{C}^2)$ 只需要 $\boldsymbol{C}^2$ 的对角线元素驱动,无需计算整个 $\boldsymbol{C}^2$ 矩阵。第三应变不变量 I_3 的计算 $I_3=\det(\boldsymbol{C})=\det(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})=J^2$,因此,如果已经计算出 $J=\det(\boldsymbol{F})$,则 I_3 的计算仅需进行一次标量乘法,无需矩阵行列式计算。

7 加速器架构

在有限元方法中,一个状态更新周期内每个网格单元都会迭代地进行本构模型计算。每个网格单元的一步本构模型计算迭代,包含了该网格单元每个顶点的本构模型计算。为充分利用算法的并行性,HyEA-Gen 生成如图 10 所示架构的专用加速器。

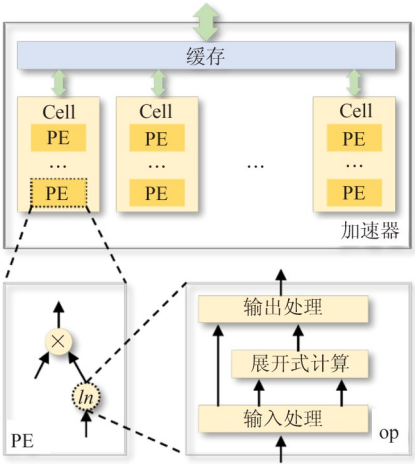

图 10 加速器架构

加速器中每个网格计算单元(Cell)的一次计算可以完成一个网格单元的一步本构模型计算迭代,每个模型计算单元(PE)的一次计算可以完成网格单元中一个网格顶点的一次本构模型计算。Cell 单元的数量决定了离散网格单元之间的最大计算并行度。PE 单元的个数由离散单元的网格类型决定,例如对于四面体和六面体网格,每个 Cell 单元中 PE 个数分别为 4 和 8。PE 采用数据流架构,其功能对应相应计算模型的数据流图。计算模型中包含的非线性运算,在生成过程中通过调用专用并行化加速算子实现。

PE 通过流水线设计提高吞吐率。流水线级数设计根据数据流图中延时最大的非线性算子决定。非线性算子电路中的输入处理电路和输出处理电路根据算子类型的不同而不同,而且可能存在算子的嵌套,因此具有不同的延时。根据最大的算子延时

作为流水线不同阶段的延时,可以在保证各个执行阶段的正确性的基础上实现最大吞吐率。

8 实验测试与分析

8.1 实验设置

本文对覆盖广泛应用场景的 9 种超弹材料的关键计算负载进行了测试,以验证设计方法对性能的提升效果。作为对照,我们设置了 6 组测试基准。测试基准分为 3 类,即通用 CPU(Intel Core i9-11900K)硬件平台上的 Matlab 和 C ++ 实现,图形处理单元(graphics processing unit,GPU)平台(NVIDIA GeForce RTX4060)上基于 CUDA(compute unified device architecture)的实现,以及 FPGA (xcvp1802-vsva5601)硬件平台上由 Vitis HLS 生成的实现。每类实现分为两组,其区别在于是否根据专家经验进行手动代码优化。使用的软件版本分别为 Matlab 2022b,g ++ 9.4.0 和 Vitis HLS 2022。测试基准的对比如表 6 所示。

表 6 测试基准

方案	领域知识	计算硬件
MATLAB-n	×	Intel Core i9-11900K
MATLAB-o	√	Intel Core i9-11900K
C ++ -n	×	Intel Core i9-11900K
C ++ -o	√	Intel Core i9-11900K
GPU-n	×	NVIDIA GeForce RTX 4060
GPU-o	√	NVIDIA GeForce RTX 4060
Vitis-n	×	xcvp1802-vsva5601
Vitis-o	√	xcvp1802-vsva5601

HyEA-Gen 生成的方案,根据是否引入领域知识优化和流水设计优化分为 4 个变体,如表 7 所示。HyEA-Gen 生成的方案同样在 FPGA (xcvp1802-vsva5601)硬件平台上运行。

表 7 HyEA-Gen 方案的变体

方案	领域知识优化	流水设计优化
HyEA-(Gen)-n	×	×
HyEA-(Gen)-n-p	×	√
HyEA-(Gen)-o	√	×
HyEA-(Gen)-o-p	√	√

虽然测试 HyEA-Gen 生成的方案时采用了 FPGA 的实现方式,但是生成的 Verilog 形式的加速器文件也可以被电路生成工具综合,生成专用集成电路。

8.2 测试结果分析

8.2.1 资 源

本文对 HyEA-Gen 生成方案与 Vitis HLS 生成方案的资源需求进行了比较。以 Ogden 材料模型的 PE 为例,测试数据如表 8 所示。

表 8 HyEA-Gen 与 Vitis HLS 方案的资源比较

方案	BRAM	DSP	FF	LUT
Vitis-n	30	393	171 368	106 793
Vitis-o	30	366	163 645	101 536
HyEA-Gen-n	30	238	3 712	227
HyEA-Gen-o	30	190	3 136	184

注:DSP 表示数字信号处理器(digital signal processor),FF 表示触发器(flip-flop),LUT 表示查找表(look-up table)

数据显示出,HyEA-Gen-x 方案的资源需求均优于 Vitis-x 方案。Vitis-o 的资源需求少于 Vitis-n 的原因是手动优化时对稀疏性知识的利用。而 HyEA-Gen-n 方案的资源需求少于 Vitis-o 的主要原因是其更加合理的时序优化。HyEA-Gen 通过更加合理的时序设计,实现了计算性能和资源的同时优化。

下面以 2 个易于理解的简单电路(Mac 和 Taylor)的实现为例进行说明。本文在 Vitis HLS 中通过设置不同的时钟周期,综合出不同的资源需求量。FPGA xcvp1802-vsva5601 的 DSP 最高工作频率约 800 MHz,对应时钟周期约 1.25 ns。因此,2 个电路的时钟周期均设置为 1.25 ns 作为 Vitis 对比基准。HyEA-Gen 对 2 个电路编译过程中得到的时钟周期分别为 4.00 ns 和 12.00 ns。因此,将这 2 个时钟周期设置到 Vitis 中进行综合,得到时序优化后的方案。4 个方案的电路综合结果如表 9 所示。

表 9 Mac 和 Taylor 电路的时序-资源优化比较

	时钟周期	Latency	DSP	FF	LUT
Mac	1.25 ns	5	1	7	8
Mac	4.00 ns	1	1	0	1
Taylor	1.25 ns	15	5	280	294
Taylor	12.00 ns	1	9	0	7

综合结果证明了合理的时序设计,可以在提高计算实时性的同时,降低资源需求量。对于 Mac 实现,当时钟周期为 1.25 ns 时,计算延时 Δ = 1.25 ns × 5 = 6.25 ns;而时钟周期为 4.00 ns 时,计算延时为 Δ = 4.00 ns,计算延时减少 36.0%。同时由于时序优化之后避免了不同时钟周期之间的数据传递,所以避免了 FF 资源的需求,并避免了多余时钟周期中使用 LUT 对 FF 中数据进行处理的需求。因此,在 DSP 需求不变的情况下,FF 和 LUT 需求分别减少 100.0% 和 87.5%。Taylor 电路与之类似,时序优化后计算延时减少了 36.0%,FF 和 LUT 需求分别减少 100.0% 和 97.6%。但是其 DSP 需求增加了 80.0%,这是因为计算并行性的增加提高了计算资源需求量。

另外,HyEA-Gen-o 相比 HyEA-Gen-n 方案的 DSP 需求量减少 20.0%,这是因为充分利用了领域知识包含的稀疏性。

8.2.2 性 能

本文对不同材料的主要计算负载——应力计算负载,进行了性能测试,对不同实现方案的计算时间进行了统计。测试结果如图 11 所示,图中纵坐标显示了不同测试方案下单个网格一次本构模型的计算延时。

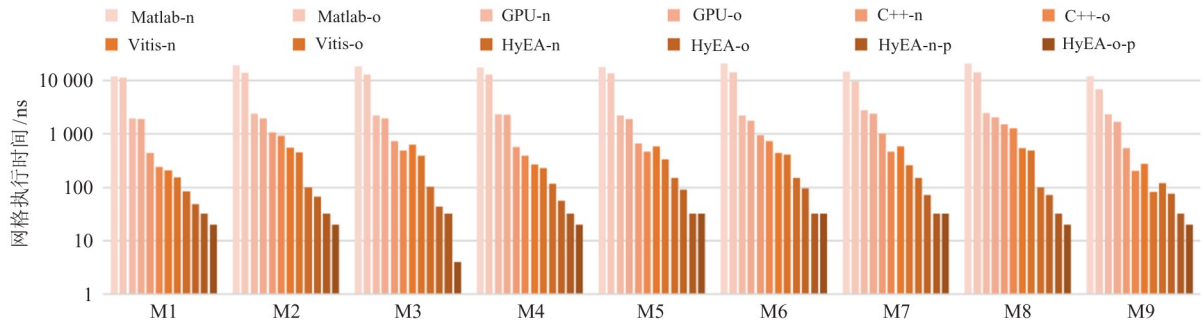

图 11 不同材料模型计算负载的执行时间

图 11 中 Matlab-n、GPU-n、C ++-n 和 Vitis-n 为原始计算模型,Matlab-o、GPU-o、C ++-o 和 Vitis-o 为手动调优的计算模型,HyEA-n 为原始生成模型,HyEA-o 为结合领域知识的优化模型,HyEA-x-p 为增加了流水线优化的模型。数据显示出,HyEA-Gen 生成的方案计算性能最优。

Matlab 方案性能最差。C ++ 方案为编译执行方案,相比 Matlab 的解释性语言执行方案获得了 10.0 ~ 25.0倍的性能提升。对于单个网格的计算性能,CPU 平台的 C ++ 方案优于 GPU 的 CUDA 方案。这主要是由于 CPU 核心相比 CUDA 核心计算能力更强。Vitis-n 方案得益于专用硬件 DSP 资源和并行执行能力,使其相比 CPU 上的 C ++ 方案获得了 2.0 ~ 3.0 倍的性能提升。Vitis-o 方案通过手动调优相比 Vitis-n 方案增加了计算并行性,实现了 1.1 ~ 3.3 倍的性能提升。HyEA-Gen 方案相比 Vitis-o 方案,增加了领域知识的优化,进一步获得了 4.0 ~ 97.0 倍的性能提升。对于大规模网格计算场景,GPU 和 FPGA 平台将得益于其远超 CPU 的硬件并行能力,表现出远超 CPU 的计算性能。由于 HyEA-Gen 方案生成的计算核心性能优于 CUDA 通用计算核心和 Vitis 生成的实现,因此,HyEA-Gen 可以实现最佳的计算性能。

图 11 中的数据显示出流水线优化方案 HyEA-x-p 相比 HyEA-x 方案带来了计算性能的提升。为了更加直观地显示提升效果,图 12 展示了对于不同材料模型,单周期实现HyEA-Gen-o和流水线实现 HyEA-Gen-o-p 2 种方案中 PE 执行周期的不同,显示出流水线优化对于 PE 吞吐率可以实现约 2.0 ~ 10.0倍的提升。

计算任务的规模主要由网格数量和网格类型决定。HyEA-Gen 生成加速器时根据网格数量配置 Cell 数量,根据网格类型配置 PE 数量,根据材料类型生成 PE 结构。因此,HyEA-Gen 可以适应不同规模的计算任务。对于同一材料类型不同规模的计算任务,PE 具有相同的计算性能,整体任务的计算性能由 Cell 数量决定。因此,当任务规模增大时,可以根据网格数量相应地增加 Cell 数量,维持整体的

计算性能。

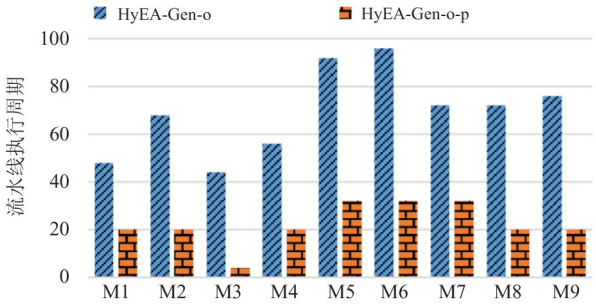

图 12 PE 的流水设计对执行周期的优化

8.2.3 能 耗

我们对 HyEA-Gen 方案与 Vitis-o 方案在不同材料的综合计算任务中的能耗进行了比较,数据如图 13所示。数据显示出 HyEA-Gen 的能耗在不同材料模型中均小于 Vitis-o 方案。这主要得益于利用领域知识进行稀疏性优化时,整体计算量的减少。

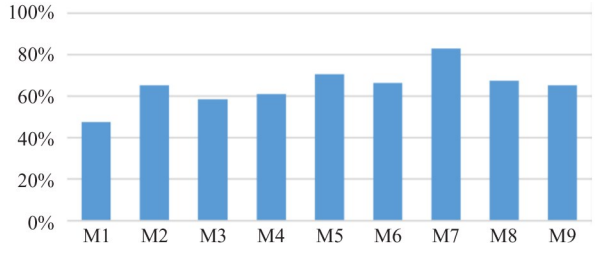

图 13 HyEA-Gen 与 Vitis HLS 方案的能耗比较

9 结 论

针对刚柔混合机器人中柔性结构运动计算复杂、延时高的问题,本工作通过计算任务测试分析,确定了计算负载的非线性计算瓶颈。针对计算负载的非线性特性,设计了并行加速的算子实现,实现了优于通用生成方案的非线性算子延时。针对柔性材料和计算模型快速发展的趋势,开发了超弹材料本构模型计算加速器生成框架 HyEA-Gen,以支持专用加速器的敏捷开发。在加速器生成过程中,利用领域知识进行电路优化,以提高加速器性能。实验测试表明,相比通用 CPU 硬件平台和通用生成框架 Vitis HLS 生成的 FPGA 实现本文提出的 HyEA-Gen 分别获得了平均 208.8 × 和 14.1 × 的性能提升。

参考文献

[1] Shin W D, Phan H V, Daley M A, et al. Fast ground-to-air transition with avian-inspired multifunctional legs[J]. Nature, 2024,636(8041):86 – 91.
[2] Tang D, Zhang C, Pan C, et al. Bistable soft jumper capable of fast response and high takeoff velocity[J]. Science Robotics, 2024,9(93):eadm8484.
[3] Wang Z, Freris N M, Wei X. SpiRobs: logarithmic spiral-shaped robots for versatile grasping across scales[J]. Device, 2025,3(4):100646.
[4] Sinatra N R, Teeple C B, Vogt D M, et al. Ultragentle manipulation of delicate structures using a soft robotic gripper[J]. Science Robotics, 2019,4(33):eaax5425.
[5] Yao D R, Kim I, Yin S, et al. Multimodal soft robotic actuation and locomotion [J]. Advanced Materials, 2024,36(19):2308829.
[6] Guo X, Li W, Fang F, et al. Encoded sewing soft textile robots[J]. Science Advances, 2024,10(1):eadk3855.
[7] Bergström J. Mechanics of solid polymers[M]. San Diego/London/Waltham: William Andrew Publishing. 2015: 209 – 307.
[8] Ilami M, Bagheri H, Ahmed R, et al. Materials, actuators, and sensors for soft bioinspired robots [J]. Advanced Materials, 2021,33(19):2003139.
[9] Shepherd R F, Ilievski F, Choi W, et al. Multigait soft robot[J]. Proceedings of the National Academy of Sciences, 2011,108(51):20400 – 20403.
[10] Armanini C, Boyer F, Mathew A T, et al. Soft robots modeling: a structured overview[J]. IEEE Transactions on Robotics, 2023,39(3):1728 – 1748.
[11] Zhang T, Li G, Ren H, et al. Sub-millimeter fiber scopic robot with integrated maneuvering, imaging, and biomedical operation abilities [J]. Nature Communications, 2024,15(1):10874.
[12] Li H, Zhao Z, Yang M, et al. Yarn-grouping weaving soft robotics with directional inflation, bilateral bending, and self-sensing for healthcare[J]. Cell Reports Physical Science, 2024,5(8):102137.
[13] Marechal L, Balland P, Lindenroth L, et al. Toward a common framework and database of materials for soft robotics[J]. Soft Robotics, 2020,8(3):284 – 297.
[14] Wang J, Guo W G, Gao X, et al. The third-type of strain aging and the constitutive modeling of a Q235B steel over a wide range of temperatures and strain rates[J]. International Journal of Plasticity, 2015,65:85 – 107.
[15] Caasenbrood B J, Pogromsky A Y, Nijmeijer H. Sorotoki: a Matlab toolkit for design, modeling, and control of soft robots[J]. IEEE Access, 2024,12:17604 – 17638.
[16] Coyle S, Majidi C, Leduc P, et al. Bio-inspired soft robotics: material selection, actuation, and design[J]. Extreme Mechanics Letters, 2018,22:51 – 59.
[17] Lee M, Park S, Kim H, et al. SPADE: sparse pillar-based 3D object detection accelerator for autonomous driving[C] // Proceedings of the 2024 IEEE International Symposium on High-Performance Computer Architecture. Edinburgh, UK:IEEE, 2024:454 – 467.

[18] Chen F, Ying R, Xue J, et al. ParallelNN: a parallel octree-based nearest neighbor search accelerator for 3D point clouds[C]//Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture. Montreal, Canada:IEEE,2023:403 - 414.
[19] Li S, Zhao Y, Li C, et al. Fusion-3D: integrated acceleration for instant 3D reconstruction and real-time rendering[C]//Proceedings of the 2024 57th IEEE/ACM International Symposium on Micro-architecture. Austin, USA: IEEE, 2024:78 - 91.
[20] Wu L, Zhu H, He S, et al. GauSPU: 3D Gaussian splatting processor for real-time SLAM systems[C]//Proceedings of the 2024 57th IEEE/ACM International Symposium on Microarchitecture. Austin, USA: IEEE, 2024: 1562 - 1573.
[21] Sun J, Liu F, Zhang Y, et al. RTSA: an RRAM-TCAM based in-memory-search accelerator for sub-100 μs collision detection[C]//Proceedings of the 2024 Design, Automation and Test in Europe Conference and Exhibition. Valencia, Spain: IEEE, 2024:1 - 2.
[22] Huang L, Gong Y, Sui Y, et al. MOPED: efficient motion planning engine with flexible dimension support[C]//Proceedings of the 2024 IEEE International Symposium on High-Performance Computer Architecture. Edinburgh, UK: IEEE, 2024:483 - 497.
[23] Neuman S M, Ghosal R, Bourgeat T, et al. RoboShape: using topology patterns to scalably and flexibly deploy accelerators across robots[C]//Proceedings of the 50th Annual International Symposium on Computer Architecture. Orlando, USA: ACM, 2023: article 69.
[24] Yang Y, Chen X, Han Y. Dadu-RBD: robot rigid body dynamics accelerator with multifunctional pipelines[C]//Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture. Toronto, Canada: ACM, 2023:297 - 309.
[25] Genc H N, Kim H, Ganesh P, et al. Stellar: an automated design framework for dense and sparse spatial accelerators[C]//Proceedings of the 2024 57th IEEE/ACM International Symposium on Micro-architecture. Austin, USA:IEEE, 2024:409 - 422.
[26] Luo Z, Lu L, Zheng S, et al. Rubick: asynthesis framework for spatial architectures via dataflow decomposition[C]//Proceedings of the 2023 60th ACM/IEEE Design Automation Conference. San Francisco, USA:IEEE,2023:1 - 6.
[27] Ujjainkar N, Leng J, Zhu Y. ImaGen: a general framework for generating memory- and power-efficient image processing accelerators[C]//Proceedings of the 50th Annual International Symposium on Computer Architecture. Orlando, USA: ACM, 2023: article 41.
[28] Li J, Chi Y, Cong J. HeteroHalide: from image processing DSL to efficient FPGA acceleration[C]// Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Seaside, USA: ACM, 2020:51 - 57.
[29] Wu D, Yu Z, Xiong X, et al. A homogenized constitutive model for 2D woven composites under finite deformation:considering fiber reorientation[J]. Composite Structures, 2025,352:118649.
[30] Inc. AMD. Vitis high-level synthesis user guide (UG1399)[EB/OL]. (2024-02)[2025-02-26]. https://docs. amd. com/r/en-US/ug1399-vitis-hls/pragma-HLS-bind_op.

HyEA-Gen: generating accelerators for hyperelastic structure computing

Geng Guoshuai[*,**], Li Zaoxia[***]
([*]Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
([**]University of Chinese Academy of Sciences, Beijing 100049)
([***]College of Physics and Electrical Engineering, Anyang Normal University, Anyang 455000)

Abstract

The use of flexible structures in rigid robot is an important trend in robot design. However, the computational complexity of flexible structure poses challenges to real-time control, limiting the development of hybrid rigid-flexible robots. This work addresses this problem by proposed HyEA-Gen, a framework for dedicated accelerator generation, which balances the flexibility and performance of computational models. The framework consists of two parts: (1)A high-level synthesizer, which generates accelerator from high-level languages (such as Matlab, C++, etc.); (2)A domain optimizer, which leverages domain-specific knowledge to optimize circuit and enhance acceleration performance. Through experiments covering a variety of flexible material models widely used in applications, the correctness and advancement of HyEA-Gen have been verified. Compared with general-purpose central processing unit central processing unit (CPU) platforms and field-programmable gate array (FPGA) implementations generated by Vitis HLS (high-level synthesis), HyEA-Gen achieves speedup ratios of 208.8× and 14.1×, respectively.

Key words: hybrid rigid-flexible robots, constitutive model, accelerator generation framework, domain-knowledge based optimization