计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(5): 1224−1237, 2026

高效可扩展的3维快速傅里叶变换异构计算架构

邓子为[1] 郭 巍[1] 徐亚明[1] 王 青[2] 张德闪[2] 卢圣才[2] 刘 弢[3]
[1](浪潮(北京)电子信息产业有限公司 北京 100085)
[2](浪潮电子信息产业股份有限公司 济南 250101)
[3](齐鲁工业大学计算机科学与技术学部 济南 250014)
(dengziwei@ieisystem.com)

Efficient and Scalable 3D-FFT Heterogeneous Computing Architecture

Deng Ziwei[1], Guo Wei[1], Xu Yaming[1], Wang Qing[2], Zhang Deshan[2], Lu Shengcai[2], and Liu Tao[3]
[1](*Inspur* (*Beijing*) *Electronic Information Industry Co., Ltd.*, *Beijing* 100085)
[2](*Inspur Electronic Information Industry Co., Ltd.*, *Jinan* 250101)
[3](*Faculty of Computer Science and Technology*, *Qilu University of Technology*, *Jinan* 250014)

Abstract The three-dimensional fast Fourier transform (3D-FFT) algorithm serves as a fundamental computing kernel for numerous high performance computing (HPC) applications, where its efficient implementation critically determines the overall system performance. We propose a heterogeneous computing architecture leveraging CPU-FPGA co-processing to address the performance bottlenecks in conventional 3D-FFT implementations, including inefficient memory access patterns, procedural redundancy, and limited parallelization capabilities. Three key architectural innovations are introduced: 1) a hierarchical data management strategy enabling multi-channel data transmission with optimized bandwidth utilization; 2) a pipelined synchronization mechanism that overlaps matrix transpose operations with FFT computation; 3) a scalable parallel computation architecture supporting up to 128 concurrent FFT processing units. Experimental evaluations demonstrate significant performance improvements of 3D-FFT for common scientific computing scenarios involving 64^3 and 128^3 matrices. Compared with CPU-only solutions, the proposed architecture achieves computation time reductions of 62.7% and 56.6% for 64^3 and 128^3 matrices, respectively. When compared with existing FPGA-accelerated 3D-FFT implementations, the proposed design exhibits performance enhancements of 32.6% and 35.3% for the corresponding matrix sizes. These results validate our architecture's effectiveness in optimizing memory utilization, improving task synchronization efficiency and enhancing computational parallelism. The solution provides a scalable and energy-efficient acceleration architecture for HPC systems requiring large-scale 3D-FFT computations, particularly in scenarios demanding high throughput and low latency.

Key words high performance computing; three-dimensional fast Fourier transform; FPGA accelerator; heterogeneous computing architecture; synchronous data transpose

摘 要 3维快速傅里叶变换(three-dimensional fast Fourier transform, 3D-FFT)作为高性能计算领域的核心算法,其加速优化对提升科学计算应用性能具有重要价值。针对现有3D-FFT实现方案在数据访存效率、同步处理机制和并行计算能力方面的性能瓶颈,设计并实现了一种基于CPU+FPGA异构计算架构的

收稿日期: 2025-06-04; **修回日期**: 2026-01-12
基金项目: 山东省自然科学基金项目(ZR2024LZH009, ZR2022LZH016)
This work was supported by Shandong Provincial Natural Science Foundation (ZR2024LZH009, ZR2022LZH016).
通信作者: 刘弢(liutao@sdas.org)

优化方法。通过建立3维数据分块传输模型，构建流水线式同步数据转置单元以及开发层次化多核并行计算架构，系统性地改善了3D-FFT算法执行效率。实验验证所提出的异构计算架构其存储器数据传输带宽达到202 GBps，实现了128个FFT核的并行计算且计算规模可扩展。实验结果表明，在处理典型科学计算场景中64^3和128^3这2种3维矩阵的3D-FFT运算时，相较于传统CPU实现方案，该架构分别取得了62.7%和56.6%的性能提升；横向对比现有FPGA加速方案，计算效率分别提高32.6%和35.3%；在大规模矩阵处理中与GPU加速方案的性能相当。

关键词 高性能计算；3维快速傅里叶变换；FPGA加速器；异构计算架构；同步数据转置

中图法分类号 TP302

DOI: 10.7544/issn1000-1239.202550393 **CSTR:** 32373.14.issn1000-1239.202550393

自20世纪以来，高性能计算(high performance computing, HPC)在物理、化学及生物现象仿真领域发挥着日益重要的作用。以分子动力学仿真为例，该技术通过数值模拟方法研究物质微观结构与动态行为，其中3维快速傅里叶变换(three-dimensional fast Fourier transform, 3D-FFT)作为核心算法之一用于计算长程静电力，在建立粒子间相互作用模型方面发挥着关键作用[1]。3D-FFT作为众多HPC应用的基础算法内核[2-3]，其计算效率的优化已成为提升相关领域研究效能的重要突破口。

为了提升3D-FFT的计算效率，其实现策略经历了显著的技术演进。在CPU平台方面，FFTW(fastest Fourier transform in the west)作为一种基于C语言开发的快速傅里叶变换库，具备良好的可移植性，可适配多种CPU架构及操作系统[4]。通过多线程并行化及多处理器协同计算，该方案显著提升了3D-FFT的计算效率。然而，受限于通用处理器架构的并行处理能力瓶颈，基于GPU的加速方案逐渐成为研究热点。主流技术路径包括利用硬件厂商提供的专用计算库，例如NVIDIA的cuFFT库、AMD的rocFFT库、Intel的oneMKL库等完成3D-FFT计算任务[5]。此外，现场可编程门阵列(field-programmable gate array, FPGA)凭借其能效优势展现出独特的潜力。相较于GPU方案，FPGA在保持相当计算效率的同时，可实现更优的能耗控制指标[6-7]。

当前国际学术界对基于FPGA的3D-FFT架构的研究已取得阶段性成果，但国内在该领域的系统性研究仍显不足。综合分析现有技术方案，主要存在3方面局限：1)频繁的数据访存操作易受存储带宽限制，导致整体计算效率衰减；2)跨维度数据转置需独立执行，产生显著额外时空开销；3)并行计算单元间的数据交互引发物理实现时的布线拥塞，严重制约时钟频率提升与并行规模扩展。

本研究针对上述技术瓶颈开展创新性探索，主要贡献如下：

1)针对存储带宽受限导致的性能下降问题，创新性提出了本地内存数据缓存机制和基于高带宽存储器(high bandwidth memory, HBM)的多通道传输策略，全局内存与本地内存之间的数据传输带宽达202 GBps，多通道较单通道传输性能最大提升近31倍；

2)突破传统独立转置处理模式，构建流水线式同步转置单元，通过FFT计算与跨维度转置的流水线执行，在减少50%存储资源占用的同时使转置时间开销完全隐藏于计算周期内；

3)攻克高扇出信号引发的物理实现难题，采用分组控制技术与分级互连架构有效缓解布线拥塞，成功实现128个并行计算核的系统集成，较现有方案将并行规模提升100%。

1 相关研究

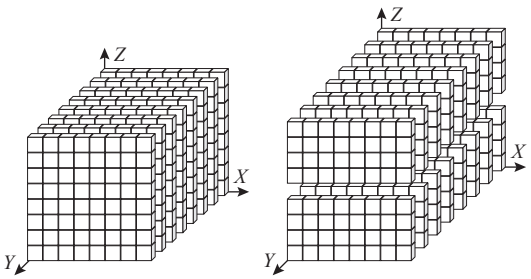
3D-FFT计算通常分解为沿X, Y, Z这3个维度连续执行的1维FFT[8]，其定义式为：

$$X(k_0,k_1,k_2)=\sum_{n_0=0}^{N_0-1}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_0,n_1,n_2)w_{N_2}^{n_2k_2}w_{N_1}^{n_1k_1}w_{N_0}^{n_0k_0},$$

其中，N_0, N_1, N_2为3维矩阵3个维度的长度，x为1维傅里叶变换序列，w为序列对应的旋转因子。

为了提升3D-FFT计算速度，普遍采用并行计算策略对每个维度的多个序列进行并行处理，以此缩短各个维度的计算耗时[9]。针对于$N\times N\times N$这样的3维矩阵，计算过程实现分为图1所示的1维分解和2维分解2种策略[10]。其中，1维分解的最高并行度为N，而2维分解的并行度最高可达N^2。从图1可以看出，当N=8时，采用1维分解的极限并行配置为8，每个FFT核需独立处理由8个长度为8的1维序列构成的数据块。而采用2维分解则可以突破1维分解

的并行度极限, 如配置为 16, 则理论上耗时能缩短一半。2 维分解虽具有更高的并行潜力, 但 FFT 核之间的数据交互复杂度提升, 从而导致通信开销呈数量级增长[11]。

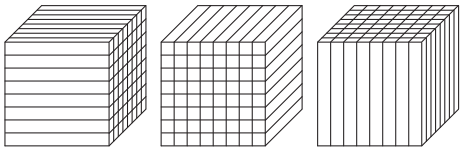


(a) 1维分解 (b) 2维分解

Fig. 1 One-dimensional and two-dimensional decompositions of 3D matrix

图 1 3 维矩阵的 1 维分解和 2 维分解

图 2 展示了不同维度 FFT 计算时的数据组织形式。从图 2 中可以看出, 不同维度 FFT 核需要处理不同空间方向的数据序列。为确保后续维度计算时的连续内存访问特性, 必须对前一个维度的计算结果实施矩阵转置操作。这种数据重组策略可有效提升存储系统访问效率, 降低整体计算延迟[12-13]。



(a) *X*维FFT (b) *Y*维FFT (c) *Z*维FFT

Fig. 2 One-dimensional FFT data in each dimension

图 2 各个维度的 1 维 FFT 数据

仍以 N=8 为例, 1 个 XY 平面的数据编号如图 3 所示。如果使用 N 个 FFT 核进行并行计算, 需为每个 FFT 核配备 1 个本地内存, 以完全发挥多核并行计算的优势[14]。对于 1 次 X 维 FFT, 每个 FFT 核输入的初始数据为 n-1 到 n-8 序列(n 为 FFT 核编号)。而在执行 Y 维 FFT 时, 每个 FFT 核输入的初始数据变为跨核分布的 1-n 到 8-n 序列。这种数据分布特性要求系统在完成 X 维 FFT 后, 必须通过全局转置操作将数据重新映射至 FFT 核的本地存储, 从而保证后续计算的数据连续性和访存效率。

1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8
2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8
3-1	3-2	3-3	3-4	3-5	3-6	3-7	3-8
4-1	4-2	4-3	4-4	4-5	4-6	4-7	4-8
5-1	5-2	5-3	5-4	5-5	5-6	5-7	5-8
6-1	6-2	6-3	6-4	6-5	6-6	6-7	6-8
7-1	7-2	7-3	7-4	7-5	7-6	7-7	7-8
8-1	8-2	8-3	8-4	8-5	8-6	8-7	8-8

Fig. 3 Data number of a *XY* plane

图 3 *XY* 平面数据编号

在 FPGA 硬件实现方面, Humphries 等人[15] 基于 Xilinx 官方提供的 1 维 FFT IP 构建了 3D-FFT 加速架构。该架构采用多 FFT IP 核并行计算模式, FFT IP 的输入输出数据格式为单精度浮点, 采用流水线模式进行 FFT 计算。每个 IP 配备独立 RAM 存储数据, 通过 2 层连接 RAM 的互连通道实现数据的转置和逆转置。在该加速架构设计中, 最多可支持 64 个 FFT IP 协同完成 64^3 矩阵的 3D-FFT 计算。同时, 研究结果也表明配置的 FFT IP 越多, 出现的时序违例越严重。

Sheng 等人[16] 基于微软的 Catapult II 云平台实现了多节点 3D-FFT 计算。在该平台中使用 Intel 提供的 1 维 FFT IP 实现 3D-FFT, IP 配置为流水线输入输出, 数据格式为单精度浮点。在云平台下最多使用了 32 个计算节点, 每个计算节点配置 4 个 FFT IP。各维度计算间的转置操作通过 2 套 RAM 存储体系和转置单元实现, 从主储存器读取数据进行转置处理后写入辅助存储器。这种双存储器机制势必会增大存储资源消耗, 同时延长计算周期。

Stewart 等人[17] 基于 Intel Stratix FPGA 芯片, 采用 OpenCL(open computing language)框架实现了 3D-FFT 优化架构。在该设计中, 使用了 Intel 提供的用于 1 维 FFT 的 OpenCL 计算内核, 使用 2 个这样的计算内核组成 1 个复合计算单元, 并在 2 个 1 维 FFT 流水线之间插入专用转置单元。在该架构中使用了 HBM 作为存储单元, 在 FPGA 平台上实现不同数量的流水线完成 3D-FFT 计算, 最多支持 8 条流水线并行处理 128^3 矩阵。该架构针对分子动力学仿真进行优化, 将 2 个维度的 FFT 计算进行耦合设计, 在一定程度上牺牲了架构的通用性。

2 算法分析

2.1 数据传输

在 3D-FFT 计算过程中, 数据访存效率对整体性能具有决定性影响。传统 FPGA 架构通常采用 DDR(double data rate)存储器作为全局内存[18-19], 然而其固有的访问延迟和带宽限制难以满足大规模 3D-FFT 的计算需求。本文提出的架构基于高端 FPGA 集成的 HBM 作为全局内存, 利用其高带宽特性有效解决了这个问题。为实现计算效率的系统性优化, 本文提出的架构采用分级存储方案: 全局内存基于 HBM

实现，同时为每个 FFT 核配置基于 BRAM（block RAM）或者 URAM（ultra RAM）的本地内存。通过建立层次化的数据访问机制，将核心计算过程中的数据访存操作约束在本地内存层级，有效规避了频繁访问全局内存产生的性能损耗[20-21]。

针对大规模矩阵场景下（如 128^3）全局内存与本地内存的数据交互瓶颈，本文提出多通道并行传输机制：首先将 N 个本地内存划分为 K 个内存组（K 为启用的 HBM 存储区数量），每个内存组通过专用 HBM 通道建立点对点数据传输通路；其次采用负载均衡策略实现通道内多存储单元的时分复用传输。该方案使理论上整体数据吞吐量达到单通道传输的 K 倍，启用 HBM 多通道数据传输前后数据交互过程对比如图 4 所示。

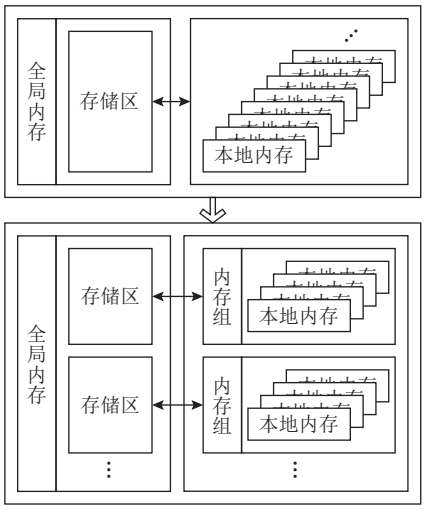


Fig. 4 Enable HBM multi-channel parallel data transmission

图 4 启用 HBM 多通道并行数据传输

本文架构构建分级存储体系结构支持高效数据交互，其中全局内存遵循 AXI（advanced extensible interface）接口规范，本地内存则采用标准 RAM 接口。鉴于全局内存单个存储区需对接多个本地内存的拓扑结构，特别设计了具备双重复用功能的数据加载单元：其一是实现 AXI 总线协议与 RAM 接口之间的信号转换，其二是完成存储通道数的动态适配。数据加载单元由与启用的 HBM 存储区相同数量的数据加载模块构成，可达成最优数据吞吐效率。

以 128^3 矩阵处理为例，每个 XZ 平面包含 128^2 个 64 b 复数（实部与虚部均为单精度浮点数），共计 128 个垂直平面构成完整数据立方体。当启用 16 个 HBM 的存储区时，采用等容量划分策略将 128 个平面数据分配至 16 个存储区。HBM 接口的数据位宽为 256 b，每个时钟周期传输 4 个序列的 1 个数据，通过设置突发传输长度为序列长度，可实现单次数据帧包含 4 个完整序列。在此配置下，每个数据加载模块仅需处理 2 个传输帧即可完成单个 XY 平面的数据传输，显著提升了传输效率。各个模块的连接关系如图 5 所示。其中数据适配模块负责接口协议转换（如 AXI3 与 AXI4 协议转换）及跨时钟域同步等功能。

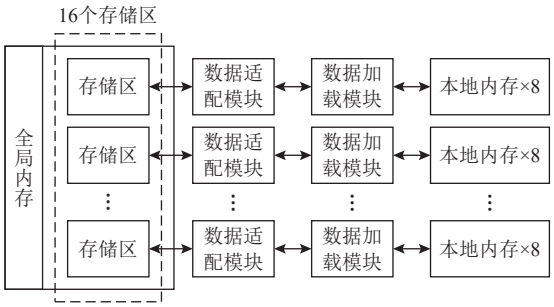


Fig. 5 Two-level memory structure and its interconnection

图 5 2 级存储结构及其互连

2.2 数据转置

在 3D-FFT 计算架构设计中，数据转置操作是影响计算效率的关键环节。文献 [16] 提出的解决方案采用双存储结构实现数据转置，其实现机制如图 6 所示。首先，FFT 单元从主存储区读取原始数据进行 X 维 FFT 运算，计算结果回写至原存储空间；其次，专用转置单元对主存储区中的 X 维 FFT 结果执行矩阵转置操作，并将重构数据存入辅助存储区；随后，FFT 单元从辅助存储区获取数据进行 Y 维 FFT 运算，计算结果仍保留在辅助存储区；后续操作以此类推。这种计算—转置交替执行的双存储区架构虽然实现了转置功能，但在实际应用中暴露出 2 个显著缺陷：其一，双存储区配置导致硬件资源需求倍增，显著增加了系统的面积成本；其二，频繁的数据访存操作不仅造成存储器带宽瓶颈，还会引入额外时延，最终影响 3D-FFT 的整体计算性能。

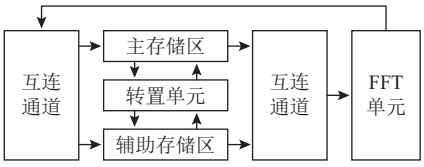


Fig. 6 Data transpose method based on two sets of memory

图 6 基于 2 套存储器的数据转置方法

文献 [15] 提出了一种基于双层互连通道的数据转置架构，如图 7 所示。该架构采用双层互连策略，通过控制单元动态配置数据传输路径：第 1 层互连通道从分布式本地内存中读数据并执行矩阵转置操作，将预处理后的数据分发至多路并行 FFT 核进行

处理；计算结果经第 2 层互连通道实施逆转置后回存至本地内存。研究表明，双层互连通道架构在 FFT 核数量扩展时面临严峻挑战：互连复杂度的指数级增长导致物理实现阶段面临严重的时序收敛问题。这种结构性缺陷使得 FFT 核数量和系统运行频率受限于互连通道的物理约束，成为制约计算性能的关键性瓶颈。

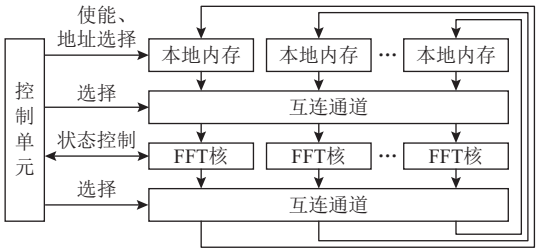


Fig. 7 Data transpose method based on two-layer crossbar

图 7 基于双层互连通道的数据转置方法

针对上述问题，本文提出了一种基于流水线的单层同步数据转置方法，其架构如图 8 所示。在本文提出的异构计算架构中，FFT 核的数量可根据硬件资源进行灵活配置，实现了 FFT 核数量在一定范围内的动态调整，在处理 N^3 矩阵时形成 3 种典型的配置模式：1)标准配置。启用 N 个 FFT 核，同步配置 N 个本地内存。2)受限配置。在硬件资源不足时，FFT 核的数量配置为少于 N，本地内存数量仍然为 N。3)富余配置。在硬件资源充足时，FFT 核的数量可以配置为多于 N，本地内存数量与 FFT 核数量相同。计算规模可扩展使设计的架构灵活应用于各类 FPGA。

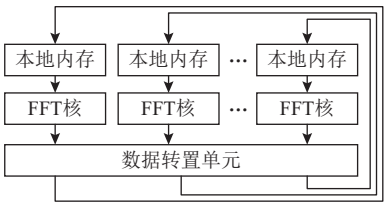


Fig. 8 Data transpose method based on pipeline

图 8 基于流水线的数据转置方法

为了缓解访存压力，提出的架构采用分布式数据存储结构。对于 N^3 矩阵，在标准配置和受限配置下，将 3 维矩阵数据分割为 N 个 $N\times N$ 的 2 维矩阵块，并将数据分别存储到 N 个本地内存中。在标准配置下，N 个 FFT 核以独占的方式分别访问这 N 个本地内存，而在受限配置下，可以使用少于 N 个 FFT 核轮询处理多个本地内存的数据。在富余配置下，可对上述 $N\times N$ 的 2 维矩阵块进一步分割，如将 3 维矩阵分割为 $2N$ 个 $N\times N/2$ 的 2 维矩阵，并将数据存储到 $2N$ 个本地内存中，启用 $2N$ 个 FFT 核以独占的方式分别访问这 $2N$ 个本地内存。

标准配置和受限配置本质上对应于数据 1 维分解范式，而富余配置则对应于数据 2 维分解范式。因此在标准配置和受限配置下执行 3D-FFT 时仅需执行单次数据转置操作，而在富余配置下则需进行 2 次数据转置。需要特别指出的是，对于富余配置的具体实施，首先需完成 Y 维度的全局分解，随后根据硬件资源约束条件对 Z 维度实施 2 次分解，其具体实现方式如图 1 所示。

依靠上述架构，本文对 3D-FFT 计算、转置、存储流程进行了系统性优化。在 X 维 FFT 阶段，得益于原始数据在本地内存中的连续存储特性，各 FFT 核可直接从本地内存连续地址空间读取数据进行并行计算。如图 2 所示，当进行 Y 维 FFT 时，初始数据分布呈现跨存储区特性。为解决该问题，本文设计了数据转置单元，通过重构数据存储拓扑使各 FFT 核能够继续从其本地内存中连续获取 Y 维 FFT 的初始数据。对于 Z 维 FFT，在标准配置或受限配置情况下，得益于配置了与序列长度匹配的本地内存数量，各 FFT 核仅需操作其本地内存的数据，所以不需要进行数据转置和重新分配。如果是采用富余配置，则仍需通过数据转置单元重构数据存储拓扑。不管是何种配置，对于 Z 维 FFT，数据访问模式将转变为固定地址间隔的非连续访问。

为实现高效数据重组，本文架构创新性地设计了基于延时机制的数据转置单元，以 X 维和 Y 维之间的数据转置为例，其架构如图 9 所示，Y 维和 Z 维之间的数据转置同理。数据转置单元采用流水线处理架构，每个 FFT 核输出端连接专用先进先出(first input first output, FIFO)缓存器，通过优化的单层互连通道完成实时数据转置与重组。与传统的双存储器架构和双层互连架构相比，该设计具有 3 重优势：首先，计算与转置的流水化协同消除了对双倍存储空间的依赖；其次，硬件级路由优化使转置时间开销完全隐藏于计算周期内；最后，单层互连结构能够有效缓解大规模并行计算中的布线拥塞问题。

上述数据转置单元执行数据转置的具体过程(以 N=8 为例)如图 10 所示。8 个 FFT 核的 X 维 FFT 结果分别输入到数据转置单元的 8 个 FIFO 中，通过周期性时序控制策略，以单时钟周期为间隔轮询读取各 FIFO 数据，如间隔 1 个时钟周期从 8 个 FIFO 分别取出 1-1, 2-1, 3-1, 4-1, 5-1, 6-1, 7-1, 8-1 数据。经互连通道重组后形成 8 组符合 Y 维 FFT 需求的新数据

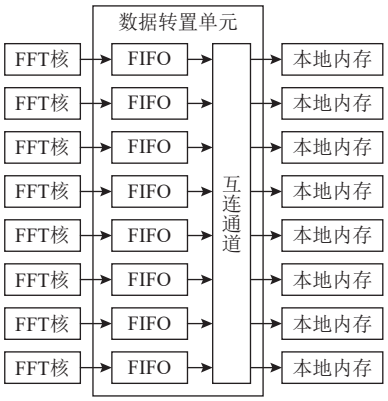


Fig. 9 Data transpose unit based on delay mechanism

图 9 基于延时机制的数据转置单元

序列，并存入对应的本地内存中，存数据时也是 1 个时钟周期存 1 个数据。数据转置单元有效解决了 3D-FFT 计算过程中的非连续访存问题，确保各个 FFT 核在后续维度计算时仍能从对应本地内存的连续地址空间获取数据，从而提升数据访存效率，进而提升 3D-FFT 计算效率。由于数据转置与 FFT 计算同步进行，节省了单独进行数据转置需要的额外存储空间和转置时间。

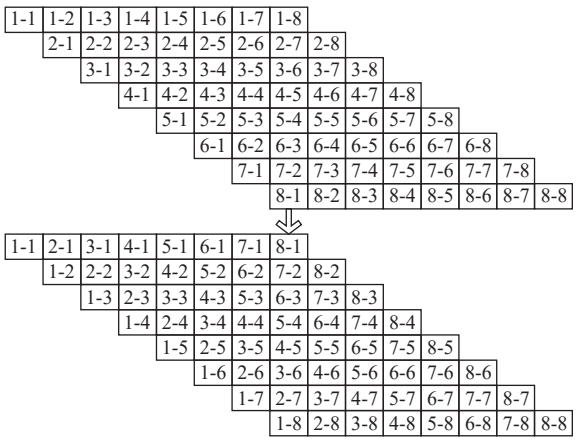


Fig. 10 Data transpose of single plane

图 10 单平面数据转置

2.3 互连分级

在 3D-FFT 计算中，提升计算效率最直接的策略是通过增加 FFT 核数量来提高计算并行度以缩短计算时间。然而由于 3D-FFT 计算过程中各 FFT 核之间存在必要的数据交互，当 FFT 核数量超过特定阈值时，硬件实现时会出现布局布线拥塞，这不仅限制了计算并行度的持续提升，还会导致系统时钟频率显著降低。文献 [15] 的研究表明，在 Xilinx VIRTEX-7 芯片上部署 64 个 FFT 核时，其系统运行时钟频率已经需要降至 61.35 MHz。本研究的实验也发现，当尝试将 FFT 核的数量扩展至 128 个时，同样出现了因布局布线拥塞而导致硬件实现失败的情况。

通过结构分析发现，制约并行度提升的关键因素包括高并行度下控制信号的高扇出特性导致时序收敛困难，以及数据转置单元中指数增长的互连分支引发布线竞争。为此，本研究提出在计算架构中采用分组设计降低控制信号扇出效应，同时在数据转置单元中实施分级转置策略以减少单级互连通道的分支密度。图 11 展示了采用分组和分级设计策略后的数据转置单元架构，此处以 N=8 为例，实际上该设计策略应用于 N 较大的情况，如 N 为 64 或 128 的情况。

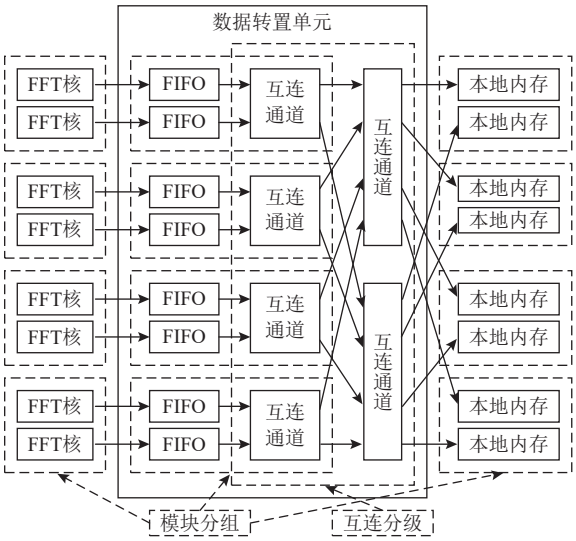


Fig. 11 Grouping and grading design strategy

图 11 分组与分级设计策略

如图 11 所示，在分组设计中，FFT 核、FIFO 与互连通道及本地内存被划分为 4 个功能组，通过组内控制逻辑共享与跨组控制逻辑解耦，有效提升了布局布线的灵活性和时序收敛的稳定性。相较于图 9 所示的数据转置单元，采用分级设计策略后，将原本 8 输入 8 输出的单级全互连通道改进为 2 级互连通道。在 2 级互连通道中最多只有 4 输入 4 输出的全互连通道，显著降低了单级互连通道的复杂度，从而有效降低了布线竞争概率，为实现更高并行度与时钟频率提供了可行路径。

在 2 级互连通道中，第 1 级互连通道实现并行化数据分段转置操作，如图 12 所示。具体实现方式是将 2 个 FFT 核的计算结果分为 4 个等长数据段，每个互连通道完成 4 个 2×2 维度的数据转置。第 2 级互连通道承担转置数据重组功能，如图 13 所示。通过构建分布式数据路由网络和建立跨通道数据整合机制实现数据重组。该级通道通过动态寻址，从第 1 级互

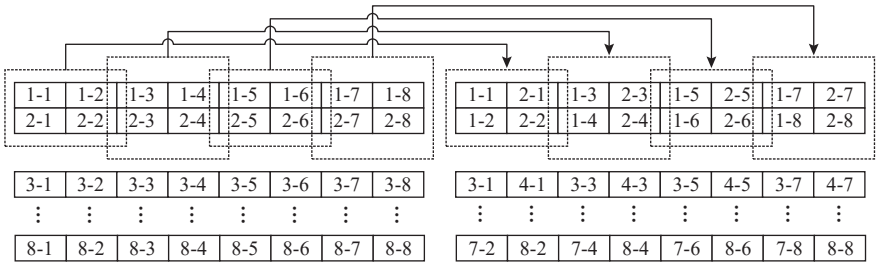


Fig. 12 Diagram of data segmentation transpose

图 12 数据分段转置图

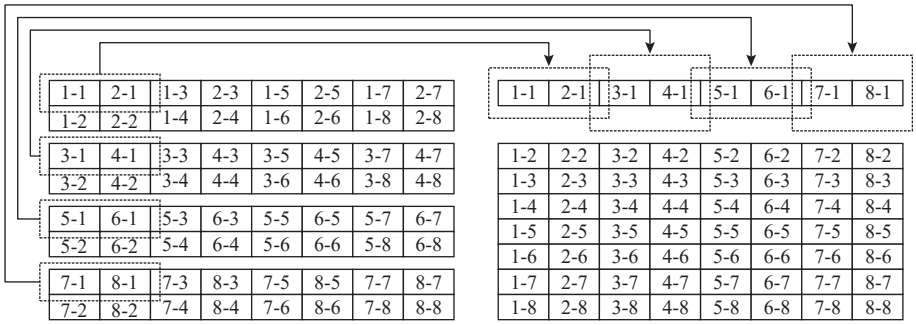


Fig. 13 Diagram of multi-segment transposed data reorganization

图 13 多段转置数据重组图

连通道提取转置后的数据段经过重组生成新序列。

3 架构实现

3.1 计算流程

本研究提出了基于 FPGA 的 3D-FFT 异构计算方案, 其异构计算过程如图 14 所示。整个计算过程分为 3 个阶段: 1)数据初始化阶段。由 CPU 通过 PCIe (peripheral component interconnect express)总线将 3D-FFT 的原始数据集传输至 FPGA 全局内存, 并发送任务启动指令。2)并行计算阶段。FPGA 获取全局内存数据并缓存至本地内存中, 随后由 FFT 核依次执行 3 维傅里叶正变换及逆变换(IFFT)运算, 运算结果写回全局内存, FPGA 中算法执行流程(1 维分解)如图 15 所示。3)数据读回阶段。FPGA 发送中断信号通知 CPU, CPU 通过内存映射接口获取全局内存中的最终计算结果, 完成闭环处理。

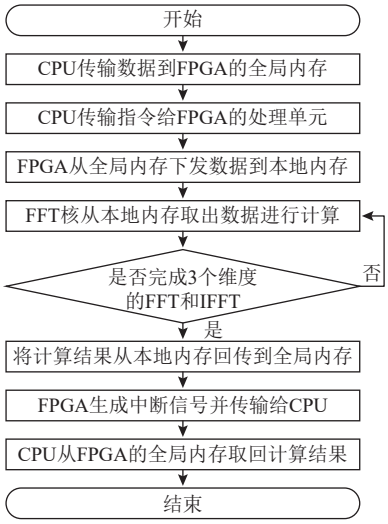


Fig. 14 Heterogeneous computing flow of 3D-FFT

图 14 3D-FFT 异构计算流程

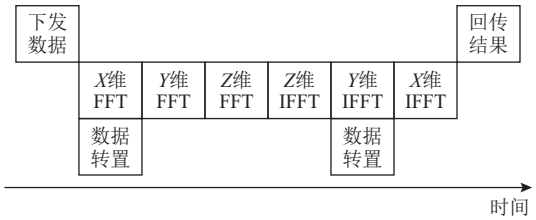


Fig. 15 Process of 3D-FFT executed by FPGA

图 15 FPGA 执行 3D-FFT 过程

3.2 计算架构

本文所提出的 3D-FFT 计算架构如图 16 所示。系统采用 CPU + FPGA 异构计算架构, 其中 FPGA 通过 PCIe 总线实现与主机的高速互连。在系统初始化阶段, FPGA 设备通过 PCIe 接口完成硬件识别与驱动加载, 建立稳定的通信链路。数据交互方面, 主机与 FPGA 之间的数据传输通过直接内存访问(direct memory access, DMA)引擎实现, 该引擎构建了主机内存与 FPGA 全局内存的双向传输通道。DMA 引擎

通过AXI总线实现高速数据传输，同时采用AXIL（AXI-Lite）总线实现控制指令的精准投递。同时为保障系统协同性，DMA引擎集成中断处理机制，具备接收FPGA产生的中断请求（interrupt request, IRQ）的能力，并通过PCIe中断通道向主机发送任务完成通知，实现计算流程的同步控制。本文提出的异构计算架构通过硬件加速单元与通用处理器的协同，实现3D-FFT计算性能的成倍提升。

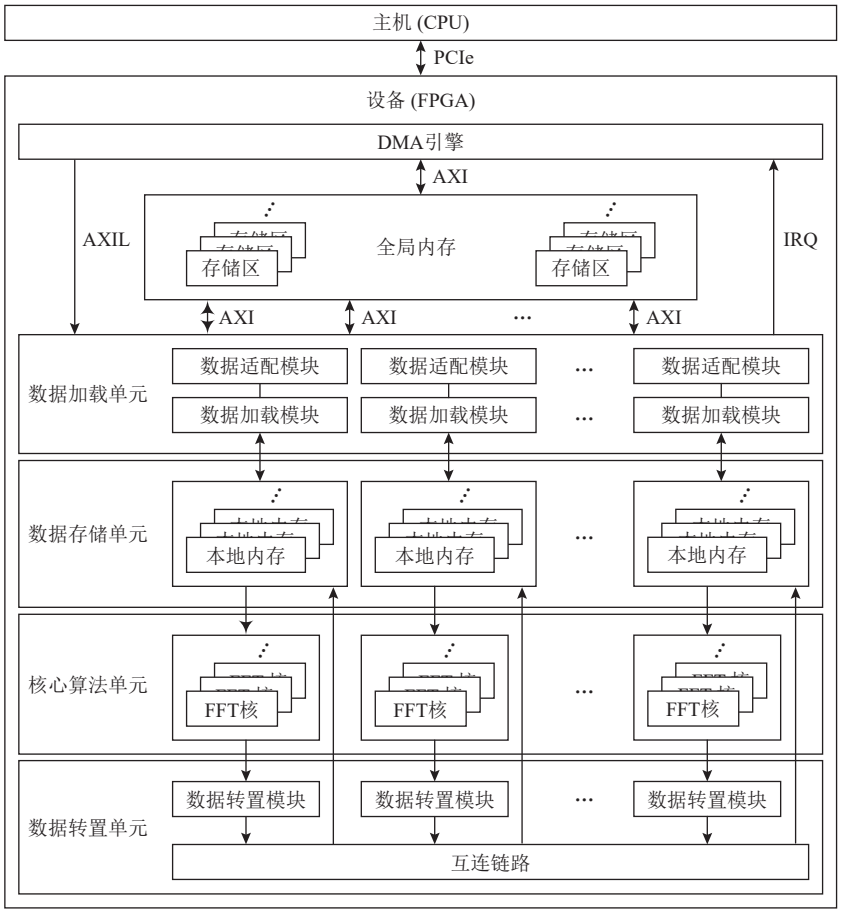


Fig. 16 Heterogeneous computing architecture of 3D-FFT

图16 3D-FFT异构计算架构

核心算法单元基于数量可扩展的FFT处理阵列实现，FFT核使用Xilinx FFT IP核，工作模式配置为流水线模式，数据精度采用IEEE-754单精度浮点标准，复数数据位宽为64 b，FFT核的数量在一定范围内灵活可配置。由于Xilinx FFT IP核仅能实现长度为2的幂次方序列的FFT计算，所以该架构当前仅能实现2的幂次方规模的3D-FFT计算。如使用支持非2的幂次方序列的FFT核替换Xilinx FFT IP核，则架构可实现非2的幂次方规模的3D-FFT计算。硬件实现结果表明，当FFT核数量扩展至128时，因控制信号扇出过大引发布局布线拥塞。为此，本架构提出分组式控制策略：将处理单元划分为多个互不干扰的独立计算簇，每个计算簇内部共享控制总线，通过层次化布线架构降低全局信号扇出，优化关键路径时序。经Vivado布局布线验证，该方案成功实现了128核并行计算。同时，本架构创新性地设计了流水线式同步数据转置单元，实现FFT计算与数据转置操作并行执行，消除传统实现中所需的显式转置操作环节。

本文所提出的异构计算架构采用分级存储结构设计，数据加载单元负责管理全局内存与本地内存间的数据交互，以支撑3D-FFT的高效运算。在全局存储层面，通过集成的高带宽、低时延HBM实现原始数据的分区存储与多通道并行传输。而在本地存储层面，提出了基于本地内存优化的数据访存加速方案，通过为每个FFT核配置专用本地内存实现数据的本地化存储。这种物理隔离的存储通道设计相较于传统的共享内存架构，能够有效提升数据访存效率。

在硬件实现层面采用BRAM或URAM构建双端

口存储体系结构, 如图 17 所示, 该双端口存储体系在数据交换与算法运算阶段实施差异化访存策略。在全局内存与本地内存数据交互阶段, 端口 *A* 专用于数据加载单元执行写入操作, 而端口 *B* 则支持数据加载单元完成读出操作。当系统进入 3D-FFT 计算阶段时, 存储端口进行动态重构: 端口 *B* 切换为核心算法单元的数据读取通道, 端口 *A* 被重新配置为数据转置单元的写入接口。

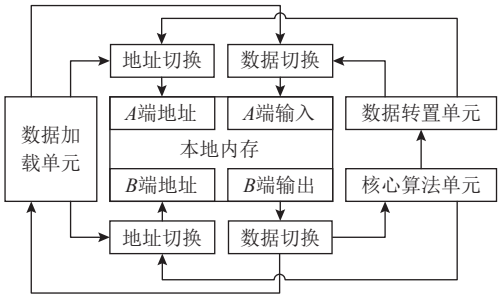


Fig. 17 Two-phase storage strategy of two-port local memory

图 17 双端口本地内存的双阶段存储策略

4 性能评估

4.1 实验环境

本文提出的架构采用 Xilinx Vivado v2020 开发工具进行全流程开发, 在浪潮 F37X FPGA 加速器上进行性能测试, 并与 Intel CPU 和 NVIDIA GPU 的性能进行比较, 实验环境配置如表 1 所示。

Table 1 Experimental Environment Configuration

表 1 实验环境配置

对比参数	CPU 工作站	GPU 显卡	FPGA 加速器
芯片型号	Intel Xeon Gold 5418Y	NVIDIA RTX A4500	Xilinx XCVU37P
算力资源	24 核 48 线程	7 168 个 CUDA 核	9 024 个 DSP, 1 303 680 个 LUT
存储资源	128 GB DDR5	20 GB GDDR6	8 GB HBM2, 24 GB DDR4
峰值算力/TFLOPS	5.84	23.7	4.38
访存带宽/GBps	38.4	640	460

Intel Xeon Gold 5418Y 的主频最大为 3.8 GHz, 每个时钟周期的单精度运算次数最多为 64, 则对于 24 核的 CPU, 其峰值算力为 5.84 TFLOPS。CPU 工作站 ThinkStation PX 的内存为 DDR5, 数据传输速率为 4 800 MT/s, 通道位宽为 8 B, 其访存速度为 38.4 GBps。Xilinx XCVU37P FPGA 具备 9 024 个 DSP 和 1 303 680 个 LUT, 执行单精度浮点运算约消耗 2 个 DSP 和 133 个 LUT, 或仅 619 个 LUT, 最大时钟频率为 775 MHz, 则 DSP 和 LUT 能够提供的单精度浮点算力分别为 3 497 GFLOPS 和 880 GFLOPS, 所以峰值算力为 4.38 TFLOPS。F37X 集成 HBM2, 最大时钟频率为 450 MHz, 共 32 个数据通道, 每个通道位宽为 32 B, 其访存带宽为 460 GBps。NVIDIA RTX A4500 数据手册给出其单精度性能为 23.7 TFLOPS, 访存带宽为 640 GBps。

图 18 所示的异构计算架构测试系统实现了 3 模态计算性能对比: 基于 Intel MKL 数学核心库的 CPU 基准方案、基于 NVIDIA cuFFT 库的 GPU 加速方案, 以及基于本文架构的 FPGA 加速方案。

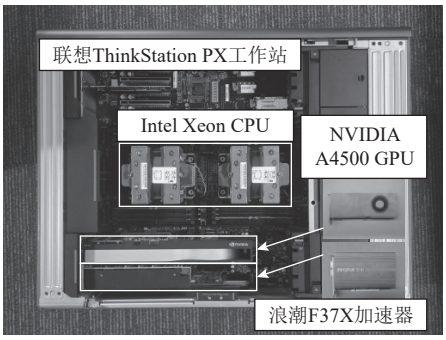


Fig. 18 Test system of 3D-FFT heterogeneous computing architecture

图 18 3D-FFT 异构计算架构测试系统

4.2 测试结果

FPGA 中执行全局内存与本地内存之间的数据同步、3 维快速傅里叶正变换和逆变换, 以及数据转置过程如图 19 所示。FPGA 加速器在接收到 CPU 发送的任务启动指令后, 数据加载模块从全局内存中读出原始数据并分配给各个本地内存。

在完成数据分配后, 各 FFT 核从其各自的本地内存读出数据依次进行 *X* 维、*Y* 维、*Z* 维 FFT 计算, 接着依次进行 *Z* 维、*Y* 维、*X* 维 IFFT 计算, 在进行 *X* 维 FFT 和 *Y* 维 IFFT 计算过程中同步进行数据转置操作(1 维分解), 从而实现数据转置与 FFT 计算的重叠, 去除单独进行转置的时间和空间开销。在所有维度的 FFT 完成后, 将最终的计算结果传回全局内存, 并产生中断信号。

提出的架构具备良好的通用性能, 通过参数配置可应用于各类 $N\times N\times N$ 的 3D-FFT 场景, 包括但不限于 32×32×32, 64×64×64, 128×128×128, 256×256×256 等。为与国外相关文献[15-17]直接进行性能比较, 且在 3D-FFT 的典型应用场景分子动力学仿真领域具备实际的应用价值, 本研究选取 64^3 和 128^3 这 2 种具有代表性的矩阵规模开展性能评估实验。本文对 3D-FFT

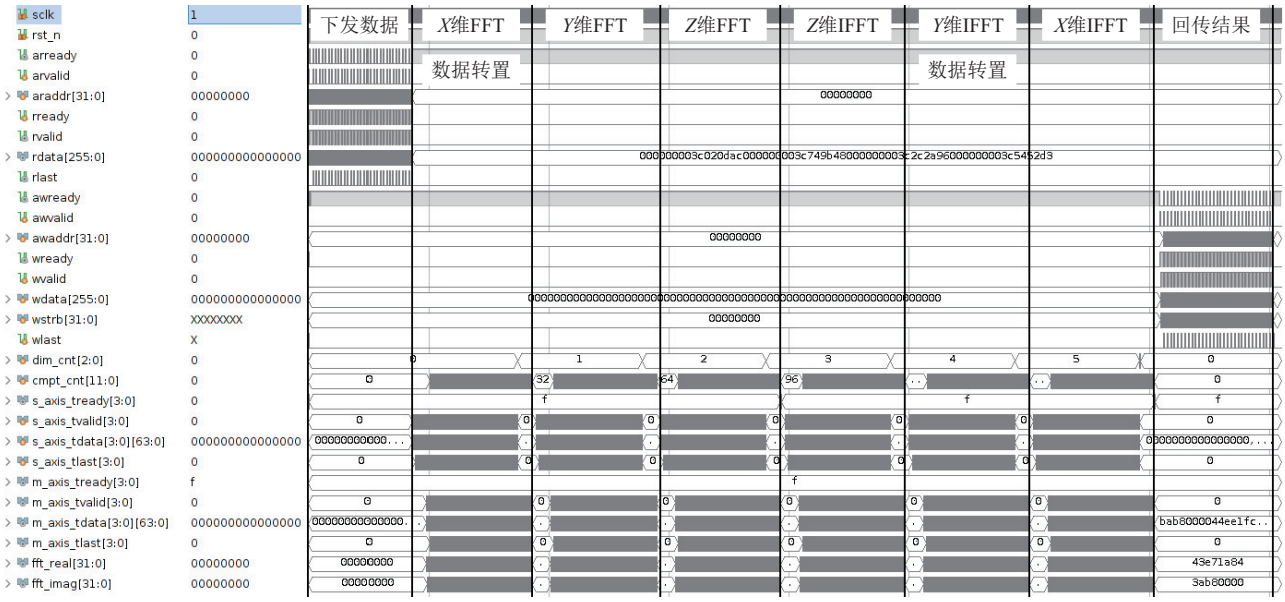


Fig. 19 Simulation results of 3D-FFT

图 19 3D-FFT 仿真结果

异构计算中的数据传输效率优化问题展开研究，通过对比分析不同存储架构对数据传输性能的影响，提出了基于 HBM 的多通道并行传输优化方案。

为保证实验结果具有可比性，实验测试统一在 250 MHz 时钟频率和 256 b 数据接口位宽条件下进行，采用 DDR 和 HBM 这 2 种全局内存架构开展对比研究。图 20 中的实验数据（纵轴采用对数坐标）给出了使用 DDR 和不同 HBM 通道数的情况下传输 64^3 和 128^3 矩阵需要的时钟周期数，根据下式计算带宽 *band width*：

$$band\ width = 8N^3/mT,$$

式中 N 为 1 维序列长度、T 为时钟周期、m 为时钟周期数。从图 20 可以看出，随着通道数量的增加，传输周期数呈线性下降趋势。在单通道传输模式下，DDR

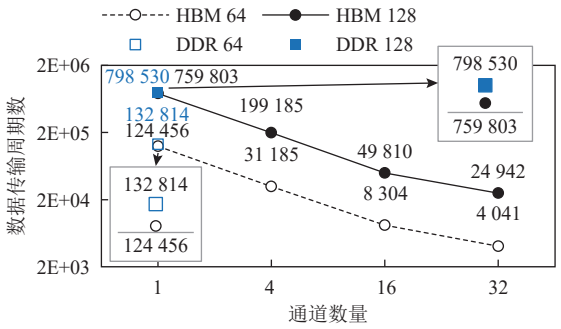


Fig. 20 Number of clock cycles for data exchange between the global memory and local memory under different conditions

图 20 不同条件下全局内存与本地内存数据交互时钟周期数

对于 64^3 和 128^3 矩阵的传输带宽分别为 3.95 GBps 和 5.25 GBps，而 HBM 对应的传输带宽则提升至 4.21 GBps 和 5.52 GBps。值得关注的是，当 HBM 采用 32 通道并行传输时，HBM 的传输带宽呈现量级提升，分别达到 129.7 GBps 和 168.2 GBps，较单通道传输模式带宽提升近 31 倍。通过提升时钟频率能够进一步提高数据传输带宽，如将时钟频率提升至 300 MHz，128^3 矩阵在 32 通道并行传输下带宽可达 202 GBps。

研究证实，在 3D-FFT 异构计算中采用 HBM 替代传统 DDR 架构，并辅以多通道并行传输策略，可有效提升全局内存与本地内存间的数据交互带宽。这种性能差异主要源于 HBM 的架构优势：由于 HBM 集成在 FPGA 芯片内部，距离处理单元更近，所以使用 HBM 完成数据传输所需的时钟周期数比使用 DDR 更少。在同为 HBM 作为全局内存的情况下，通过将原始数据分块存储于 HBM 的多个存储区，并建立多通道并行传输机制，可实现数据访问模式与存储架构的空间匹配，从而突破传统 DDR 架构的带宽限制。进一步分析表明，不同规模数据集的传输效率差异主要受突发传输长度影响。128^3 矩阵的突发传输长度（128）较 64^3 矩阵（64）增加 100%，使得更大规模数据能够更充分地利用内存总线带宽。

本文研究了计算效率与并行度之间的关系，在实验中分别部署了 16, 32, 64, 128 个 FFT 核完成 64^3 和 128^3 矩阵的 3D-FFT 计算，借此研究架构的可扩展性能。实验采集了各配置下完成 3D-FFT 计算所需的时钟周期数，如图 21 所示（纵轴采用对数坐标）。

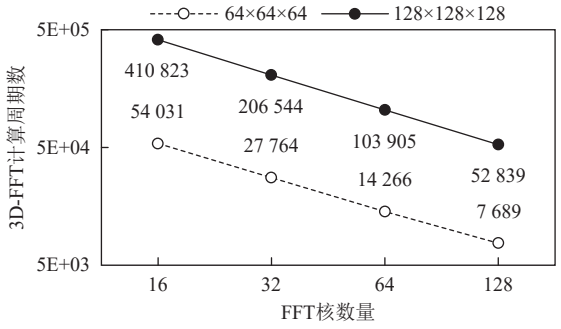


Fig. 21 Number of clock cycles required to complete 3D-FFT with different numbers of FFT kernels

图 21 不同 FFT 核数量下完成 3D-FFT 所需时钟周期数

实验结果表明, 当 FFT 核数量成倍增长时, 3D-FFT 计算周期数呈现近似线性的下降趋势。该现象验证了提高并行度是提升 3D-FFT 计算效率的有效途径。值得注意的是, 当 FFT 核规模扩展至 128 时, 出现因互连拥塞严重而无法完成物理实现的状况, 暴露出高并行度架构设计中的关键瓶颈——互连网络拥塞问题。本研究通过引入分组设计和分级转置策略, 有效降低对硬件资源的消耗并缓解互连资源竞争。

在对 64^3 矩阵进行 64 个 FFT 核并行 3D-FFT 时, 如果不采用分组和分级策略, 则单级转置单元消耗的 LUT(look-up table)资源数量为 105 102, 且有 4 716 个端点存在时序违例; 而如果将 64 个 FFT 核分为 4 个组, 转置时互连采用 2 级架构, 则 2 级转置单元总共消耗的 LUT 资源为 70 521, 节约 32.9% 的硬件资源, 并且此时不存在时序违例的端点。由此可见, 采取分组设计和分级转置策略有助于减少对硬件资源的消耗, 从而缓解布局布线拥塞问题, 通过实施该策略, 本文最终成功实现了 128 个 FFT 核的并行流水线架构。

本文提出的架构相较于其他基于 FPGA 的同类方案具有显著的优势。文献 [15] 和文献 [17] 分别采用 Xilinx XC7V2000T-1FLG1925 和 Intel Stratix 10 MX2100 实现 3D-FFT 计算架构, 而本文提出的架构应用于 Xilinx XCVU37P, 3 款芯片的关键硬件资源如表 2 所示。从表 2 中可以看出, XCVU37P 的 LUT 和 FF(flip-flop)资源与 XC7V2000T 基本持平, 但具备更丰富的 BRAM 和 DSP(digital signal processing)资源。与 MX2100 相比, XCVU37P 的 LUT 和 FF 资源相对较低, 但其 DSP 数量更多。3 款芯片的资源规模处于同一量级, 其上实现的 3D-FFT 计算架构的性能具有可比性。据我们所知, 文献 [15,17] 是已知方案中 2 个最具代表性的 FPGA 方案。文献 [15] 采用与本文相同的 Xilinx FFT IP 核进行 FFT 计算, 但本文在 3D-FFT 架构上进行了较大改进, 实现整体性能的显著提升; 文献 [17] 给出的 3D-FFT 计算性能是我们已知的最优性能, 与之比较可以体现本文架构的优越性。

Table 2 Number of Key Hardware Resources for the Three Types of Chips

表 2 3 款芯片的关键硬件资源数量

芯片型号	LUT	FF	BRAM	DSP
XC7V2000T	1 221 600	2 443 200	1 292	2 160
MX2100	2 073 000	2 810 880	6 812	3 960
XCVU37P	1 303 680	2 607 360	2 016	9 024

基于上述 3 款芯片完成 3D-FFT 计算所需时间如表 3 所示。在 64^3 矩阵处理中, 文献 [17] 使用了 8 条实现 FFT 计算的 OpenCL 流水线, 其运行时钟频率可达 295 MHz。文献 [15] 使用了 64 个 FFT 核进行并行计算, 因布局布线存在困难, 其运行时钟频率被迫降至 61.35 MHz。本文架构在实现 128 个 FFT 核并行计算的同时, 通过多维度布局优化策略使时钟频率提升至 300 MHz, 计算耗时显著降低。文献 [15] 中未实现 128^3 矩阵的 3D-FFT, 文献 [17] 仍使用 8 条流水线进行计算, 本文提出的架构使用了 128 个 FFT 核进行并行计算, 完成任务所需的周期数更少, 从而时间更短。相比于其他 FPGA 方案, 本文提出的架构性能更佳, 对 64^3 和 128^3 矩阵的 3D-FFT 性能分别提升了 32.6% 和 35.3%。对于 N^3 矩阵进行 3D-FFT 计算, 其浮点计算量为 $15N^3\text{lb}(N)$, 本文提出的架构对于 64^3 和 128^3 矩阵进行 3D-FFT 计算的算力分别达到 920.5 GFLOPS 和 1 250.4 GFLOPS。

Table 3 Performance Comparison of 3D-FFT with Different FPGAs

表 3 不同 FPGA 芯片下 3D-FFT 性能比较

规模	芯片型号	时钟频率/MHz	周期数	核数	耗时/μs
64^3	XC7V2000T	61.35	13 245	64	215.9
	MX2100	295	11 209	8	38
	XCVU37P	300	7 689	128	25.63
128^3	XC7V2000T				
	MX2100	327	88 947	8	272
	XCVU37P	300	52 839	128	176.1

使用 CPU, GPU 和 FPGA 完成相同规模的 3D-FFT 计算存在性能差距。在表 4 中给出了使用一颗 Intel Xeon CPU 基于 MKL 库和使用 NVIDIA A4500 GPU 基于 cuFFT 库对 64^3 和 128^3 矩阵进行 3D-FFT 所消耗的时间。本文提出的架构在性能上显著优于

CPU 的方案，对 64^3 和 128^3 矩阵的 3D-FFT 计算性能分别提升了 62.7% 和 56.6%。实验数据显示，当处理规模为 64^3 矩阵时，本文架构相较于 GPU 方案存在一定的性能差距。这种差异主要源于当前架构在较小计算规模下的并行度优化不足。值得注意的是，当处理规模扩展至 128^3 时二者性能相当。

Table 4 Performance Comparison of 3D-FFT Under Different Schemes

表 4 不同方案下 3D-FFT 性能比较

规模	加速设备	耗时/μs
64^3	Intel Xeon Gold 5418Y CPU	68.66
	NVIDIA RTX A4500 GPU	19.39
	Xilinx XCVU37P FPGA	25.63
128^3	Intel Xeon Gold 5418Y CPU	406.2
	NVIDIA RTX A4500 GPU	180.9
	Xilinx XCVU37P FPGA	176.1

表 5 展示了不同数据规模下计算并行度对硬件资源消耗的关联特征。实验数据分析表明，随着 FFT 计算核数量增加，各类硬件资源的消耗呈现显著增长趋势。当并行度提升至 128 时，LUT 资源消耗率突破 50%，同时在布局布线阶段出现时序违例和物理拥塞现象。实验结果表明，在此状况下继续增加并行度将导致设计规则违反而无法有效实现。因此，针对此类大规模计算场景，应当聚焦架构层面的优化改进，通过逻辑结构重组和时序路径优化等手段提升系统运行频率，从而突破现有计算效率瓶颈。

Table 5 Usage of Hardware Resources with Different Configurations

表 5 不同配置下硬件资源的使用情况

规模	核数	使用硬件资源占总资源的百分比/%				
		LUT	FF	BRAM	URAM	DSP
64^3	64	27	19	14	7	11
	128	52	38	25	13	37
128^3	64	38	23	6	53	17
	128	53	38	18	53	34

在异构计算系统中执行 3D-FFT 的整个计算过程如图 22 所示，具体实施过程包含 5 个关键步骤：

1）硬件资源检测与配置。系统启动时首先进行加速器硬件检测，当识别到可用 FPGA 加速器时建立连接。基于 PCIe 总线协议建立主机与加速器间的通信链路，完成双向数据传输通道的初始化配置。

2）数据传输与基准建立。通过上层应用生成数据集，同时执行标准数学库计算得到基准变换结果，并存储于系统中以供后续对比验证。主机端通过 DMA 引擎将原始数据传输至加速器全局内存。

```
[root@localhost float64-random]# ./execute.sh
### Computing three-dimensional FFT ###
Info: The Size of FFT is 64*64*64.
*** Stage 1: Detecting heterogeneous device ***
Info: Xilinx corporation device is detected.
Info: The device number is 903f.
*** Stage 2: Download the original data ***
Info: The data to be transferred is 2MB.
Info: Data download successfully.
*** Stage 3: Computing three-dimensional FFT ***
Info: Launch computing task.
Info: Computing task complete.
*** Stage 4: Upload the computing results ***
Info: Data upload successfully.
*** Stage 5: Verify the computing results ***
Info: Relative error distribution.
<0.01%  |  [0.01%, 0.1%)  |  [0.1%, 1%)  | >1%
261974  |       154       |       16     |  0
### Three-dimensional FFT compute successfully ###
```

Fig. 22 Heterogeneous computing flow of 3D-FFT

图 22 3D-FFT 异构计算流程

3）加速器任务调度。任务调度器发送启动指令至 FPGA 处理单元，触发 3D-FFT 计算流水线。加速器基于本文提出的 3D-FFT 异构计算架构实现数据的读取和并行化计算。

4）中断响应与数据回收。加速器完成 3D-FFT 计算流程后，系统生成中断信号。主机端通过内存映射接口读取计算结果，并进行数据格式转换与归一化处理，确保与基准变换结果具有可比性。

5）误差分析与验证。基于数值分析原理，系统执行逐点对比计算。

5 结 论

本研究提出了一种基于 FPGA 的 3D-FFT 异构计算架构，通过实施数据本地化存储与启用多通道传输机制、开发数据转置与傅里叶变换的流水线执行技术，以及构建多核并行计算体系 3 阶段优化策略实现 3D-FFT 算法加速。经统一测试数据集验证，相较于传统 CPU 实现方案，本架构在 64^3 和 128^3 规模数据集上的计算耗时分别降低 62.7% 和 56.6%。横向对比同类 FPGA 方案，其计算效率提升幅度达 32.6% 和 35.3%。与主流 GPU 方案相比，当前架构在计算效率方面存在改进空间，未来通过架构优化、计算规模扩展及系统运行频率提升，理论上可实现性能超越。

作者贡献声明：邓子为设计实现了 3 维快速傅

里叶变换异构计算架构，并完成论文的撰写；郭巍、徐亚明设计并完成实验测试，以及对论文提出修改建议；王青、张德闪完成软硬件协同测试；卢圣才完成工程仿真；刘弢提出论文撰写指导意见。

参 考 文 献

[1] Fang Bin, Deng Yuefan, Martyna G. Performance of the 3D-FFT on the 6D network torus QCDOC parallel supercomputer[J]. Computer Physics Communications, 2007, 176: 531−538

[2] Jung J, Kobayashi C, Imamura T, et al. Parallel implementation of 3D-FFT with volumetric decomposition schemes for efficient molecular dynamics simulations[J]. Computer Physics Communications, 2016, 200: 57−65

[3] Vay J L, Almgren A, Bell J, et al. Warp-X: A new exascale computing platform for beam–plasma simulations[J]. Nuclear Instruments and Methods in Physics Research Section A, 2018, 909: 476−479

[4] Frigo M, Johnson S G. The design and implementation of FFTW3[J]. Proceedings of the IEEE, 2005, 93(2): 216−231

[5] Tolmachev D. VkFFT-A performant, cross-platform and open-source GPU FFT library[J]. IEEE Access, 2023, 11: 12039−12058

[6] Quraishi M H, Tavakoli E B, Ren F. A survey of system architectures and techniques for FPGA virtualization[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(9): 2216−2230

[7] Jia Xun, Qian Lei, Wu Guiming, et al. Research advances and future challenges of FPGA-based high performance computing[J]. Computer Science, 2019, 46(11): 11−19 (in Chinese)
(贾迅, 钱磊, 邬贵明, 等. FPGA 应用于高性能计算的研究现状和未来挑战[J]. 计算机科学, 2019, 46(11): 11−19)

[8] Chen Tun, Jia Haipeng, Li Zhihao, et al. A transpose-free three-dimensional FFT algorithm on ARM CPUs[C] //Proc of the 23rd Int Conf on High Performance Computing and Communications. Piscataway, NJ: IEEE, 2021: 1−8

[9] Araujo G A d, Griebler D, Danelutto M, et al. Efficient NAS parallel benchmark kernels with CUDA[C] //Proc of the 28th Euromicro Int Conf on Parallel, Distributed and Network-Based Processing. Piscataway, NJ: IEEE, 2020: 9−16

[10] Pekurovsky D. P3D-FFT: A framework for parallel computations of Fourier transforms in three dimensions[J]. SIAM Journal on Scientific Computing, 2012, 34: 192−209

[11] Fang Wei, Sun Guangzhong, Wu Chao, et al. A parallel algorithm of three-dimensional fast Fourier transform[J]. Journal of Computer Research and Development, 2011, 48(3): 440−446 (in Chinese)
(方维, 孙广中, 吴超, 等. 一种三维快速傅里叶变换并行算法[J]. 计算机研究与发展, 2011, 48(3): 440−446)

[12] Bailey D H. FFTs in external or hierarchical memory[C] //Proc of the 1989 ACM/IEEE Conf on Supercomputing. Piscataway, NJ: IEEE, 1989: 234−242

[13] Wang Di, Shi Song, Wu Tiebin, et al. A high performance accelerator design for ultra-long point floating-point FFT[J]. Journal of Computer Research and Development, 2021, 58(6): 1192−1203 (in Chinese)
(王谛, 石嵩, 吴铁彬, 等. 一种高性能超长点数浮点 FFT 加速器设计[J]. 计算机研究与发展, 2021, 58(6): 1192−1203)

[14] Yang Chao, Chen Haiyan, Liu Sheng. An address parallel access method supporting four reformulated radix-2~4 FFT[J]. Journal of Computer Research and Development, 2017, 54(1): 134−141 (in Chinese)
(杨超, 陈海燕, 刘胜. 一种支持变形基 2~4 FFT 的 4 路并行访存方法[J]. 计算机研究与发展, 2017, 54(1): 134−141)

[15] Humphries B, Zhang Hansen, Sheng Jiayi, et al. 3D-FFTs on a single FPGA[C] //Proc of the 22nd Annual Int Symp on Field-Programmable Custom Computing Machines. Piscataway, NJ: IEEE, 2014: 68−71

[16] Sheng Jiayi, Yang Chen, Sanaullah A, et al. HPC on FPGA clouds: 3D-FFTs and implications for molecular dynamics[C] //Proc of the 27th Int Conf on Field Programmable Logic and Applications. Piscataway, NJ: IEEE, 2017: 1−4

[17] Stewart L C, Pascoe C, Sherman B W, et al. An OpenCL 3D-FFT for molecular dynamics distributed across multiple FPGAs[J]. arXiv preprint, arXiv: 2009.12617, 2020

[18] Seo S, Jo G, Lee J. Performance characterization of the NAS parallel benchmarks in OpenCL[C] //Proc of the 2011 IEEE Int Symp on Workload Characterization. Piscataway, NJ: IEEE, 2011: 137−148

[19] Li Yan, Zhang Yunquan. An automatic performance tuning framework for FFT on heterogenous platforms[J]. Journal of Computer Research and Development, 2014, 51(3): 637−649 (in Chinese)
(李焱, 张云泉. 异构平台上性能自适应 FFT 框架[J]. 计算机研究与发展, 2014, 51(3): 637−649)

[20] Lei Yuanwu, Chen Xiaowen, Peng Yuanxi. A high energy efficiency FFT accelerator on DSP chip[J]. Journal of Computer Research and Development, 2016, 53(7): 1438−1446 (in Chinese)
(雷元武, 陈小文, 彭元喜. DSP 芯片中的高能效 FFT 加速器[J]. 计算机研究与发展, 2016, 53(7): 1438−1446)

[21] Wang Zeke, Huang Hongjing, Zhang Jie, et al. Shuhai: Benchmarking high bandwidth memory on FPGAs[C] // Proc of the 28th Annual Int Symp on Field-Programmable Custom Computing Machines. Piscataway, NJ: IEEE , 2020: 111−119



Deng Ziwei, born in 1995. PhD, engineer. His main research interests include heterogeneous computing architecture and high performance computing.

邓子为, 1995 年生。博士, 工程师。主要研究方向为异构计算架构、高性能计算。



Guo Wei, born in 1976. Master, engineer. His main research interests include heterogeneous computing accelerator and high performance computing.

郭 巍, 1976 年生。硕士, 工程师。主要研究方向为异构计算加速器、高性能计算。



Xu Yaming, born in 1986. Bachelor, engineer. His main research interests include network technology and heterogeneous computing architecture.

徐亚明, 1986 年生。学士, 工程师。主要研究方向为网络技术、异构计算架构。



Wang Qing, born in 1992. Master, engineer. Her main research interest includes energy consumption optimization of servers and heterogeneous computing architecture.

王　青, 1992 年生。硕士, 工程师。主要研究方向为服务器及异构计算架构能耗优化。



Zhang Deshan, born in 1990. Master, engineer. His main research interests include virtualization of FPGA and heterogeneous computing architecture.

张德闪, 1990 年生。硕士, 工程师。主要研究方向为 FPGA 虚拟化、异构计算架构。



Lu Shengcai, born in 1986. Master, engineer. His main research interests include computer architecture and high performance computing.

卢圣才, 1986 年生。硕士, 工程师。主要研究方向为计算机架构、高性能计算。



Liu Tao, born in 1985. PhD, professor. His main research interests include FPGA design and implementation and high performance computing.

刘　弢, 1985 年生。博士, 研究员。主要研究方向为 FPGA 设计实现、高性能计算。