计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(1): 28−40, 2026

# 基于 PCIe 的链式 DMA 控制器设计与实现

王洪良[1] 郭振华[1] 牟 奇[1,3] 卢圣才[1] 徐亚明[2] 于泉泉[1]
[1]（浪潮电子信息产业股份有限公司 济南 250101）
[2]（浪潮（北京）电子信息产业有限公司 北京 100085）
[3]（华南理工大学计算机科学与工程学院 广州 510641）
（wanghl01@ieisystem.com）

# Design and Implementation of Chained DMA Controller Based on PCIe

Wang Hongliang[1], Guo Zhenhua[1], Mu Qi[1,3], Lu Shengcai[1], Xu Yaming[2], and Yu Quanquan[1]
[1]（*Inspur Electronic Information Industry Co., Ltd.*, *Jinan* 250101）
[2]（*Inspur* (*Beijing*) *Electronic Information Industry Co., Ltd.*, *Beijing* 100085）
[3]（*School of Computer Science and Engineering*, *South China University of Technology*, *Guangzhou* 510641）

**Abstract** In recent years, although Chinese domestic field programmable gate array (FPGA) manufacturers have developed rapidly, they still face challenges when deploying FPGA heterogeneous accelerators in data centers. Compared with international manufacturers such as Xilinx (now AMD) and Intel, domestic manufacturers generally lack solutions for high-speed transmission between PCIe devices and hosts, especially in the field of high-performance direct memory access (DMA) controller design, where there are obvious shortcomings. To solve this problem, we design and implement a PCIe-based multi-channel chained DMA controller. By using an independent descriptor controller to manage each channel, sharing data movers, and reducing the consumption of FPGA logic resources, this design improves resource efficiency. The adoption of a chain structure for descriptor management reduces CPU interrupt pressure while meeting the requirements for continuous high-speed transmission between hosts and devices. An innovative architecture for asynchronous internal information pre-processing is developed, enabling data stream processing that significantly improves bandwidth utilization and transmission performance. Testing results show that under PCIe Gen3x8, the DMA bandwidth between the hosts and the domestic FPGA accelerator reaches 6.91 GBps (86% utilization rate), supporting up to 16 channels with channel balancing implementation. This design effectively enables large-scale deployment of domestic FPGA heterogeneous accelerators in data center scenarios.

**Key words** PCIe; FPGA; heterogeneous acceleration; multi-channel; chained DMA

**摘 要** 近年来，国产可编程门阵列（field programmable gate array, FPGA）厂商虽发展迅速，但在数据中心部署 FPGA 异构加速器时仍面临挑战。相较于赛灵思、英特尔等国际厂商，国产厂商普遍缺乏 PCIe 设备与主机间高速传输的解决方案，尤其在高性能直接内存访问（direct memory access, DMA）控制器设计领域存在明显短板。为解决该问题，设计并实现了基于 PCIe 的多通道链式 DMA 控制器。采用独立描述符控制器管理各通道，共享数据搬移器，降低对 FPGA 逻辑资源的消耗。采用链式结构实现描述符管理，减少中断对 CPU 的压力，满足主机与设备连续高速传输的需求。创新性地构建内部信息异步与预处理的架构，实现数据流水化处理，显著提升带宽利用率以及传输性能。经测试，在 PCIe Gen3x8 下，主机与国产 FPGA 加速器之间的 DMA 带宽高达 6.91 GBps（利用率 86%），支持多达 16 通道且实现通道负载均衡，该设计有效支撑了国产 FPGA 异构加速器在数据中心场景下的规模化部署。

**收稿日期**：2024-10-31；**修回日期**：2025-06-16
**基金项目**：山东省自然科学基金项目（ZR2023LZH012）
This work was supported by the Shandong Provincial Natural Science Foundation (ZR2023LZH012).

**关键词** PCIe; FPGA; 异构加速; 多通道; 链式 DMA

**中图法分类号** TP303

**DOI:** 10.7544/issn1000-1239.202440833 **CSTR:** 32373.14.issn1000-1239.202440833

得益于强大的并行计算能力和灵活性，可编程门阵列（field programmable gate array, FPGA）异构加速器受到微软、腾讯、字节跳动、阿里等互联网巨头的青睐，并被广泛应用于数据中心的大规模部署。例如，微软利用 FPGA 加速搜索算法，计算速度较 CPU 提升了 40 倍，系统整体性能提升了 100%，同时服务器数量减少了 50%。字节跳动将 FPGA 加速器用于视频压缩和图像处理任务，显著降低了数据中心的存储和计算压力。腾讯利用 FPGA 进行网络数据卸载，致力于探索智能网卡的整体解决方案与硬件新形态。阿里则推出了 GPU 与 FPGA 的异构计算解决方案，为零售系统提供加速支持。清华大学基于 FPGA 开发大模型推理框架，显著提升了性价比与能效比。此外，科大讯飞、联捷科技、华大基金、百度等公司使用 FPGA 加速语音识别、图像处理、科学计算、自动驾驶、人工智能等业务。因此，FPGA 异构加速在学术界和工业界均展现出广阔的应用前景。

在科技贸易战的背景下，部署自主可控 FPGA 异构加速器的需求愈发迫切。数据中心部署的 FPGA 加速器普遍采用高性能、大容量芯片，而当前英特尔和赛灵思长期垄断市场，国产 FPGA 芯片在性能、容量及生态建设等方面存在显著差距，使得国产 FPGA 异构加速器的实际部署面临严峻挑战。本文调研了紫光同创、安路科技、高云、易灵思、智多晶、京微齐力 6 家国内民用厂商及其产品，发现目前国内 FPGA 产品的最高工艺水平为 12 nm，查找表（look up table, LUT）最大规模为 60 万，最高 Serdes 速率为 32 Gbps，最高 MAC 规格为 100 Gbps，PCIe 规格最高支持 Gen4x8。相较之下，国外产品已采用 7 nm 先进工艺，LUT 最大规模达 1 800 万，Serdes 速率达 112 Gbps，MAC 规格提升至 400 Gbps，PCIe 规格支持 Gen5x16。参数对比凸显国内产品全维度落后态势。自 2016 年起，国内企业开始基于英特尔 Arria10/Stratix10、赛灵思 VU37P 等国外平台开发加速器产品，而英特尔与赛灵思也相继推出 PAC 卡、U280 等自有解决方案。科技贸易战爆发后，国内企业开始尝试国产化替代，例如采用同创 PG2L100H 芯片研发 FPGA 加速器原型。FPGA 异构加速应用的核心难点在于通用高规格高性能直接内存访问（direct memory access, DMA）控制器，该技术是主机与 FPGA 间数据传输的关键，如赛灵思 XDMA、英特尔 SGDMA，而国产 FPGA 厂商尚未攻克相关技术，仅能提供 PCIe 硬核以及简易的演示程序，无法满足数据中心的应用要求，导致国产加速器难以部署。

当前 PCIe DMA 控制器设计领域存在 2 个参考设计方案：开源架构 RIFFA[1] 和赛灵思早期参考设计 XDMA[2]。然而，两者在规格性能指标上存在显著缺陷，主要表现为：带宽利用率低、缺乏链式传输支持、缺乏多通道支持或者调度不均衡、目的地址寻址机制受限等问题，无法满足复杂的应用场景。由于 PCIe 协议复杂且可借鉴的成熟设计稀缺，开发高性能高规格 DMA 控制器需要重点解决以下挑战：

1）链式传输机制。链式 DMA 机制相较于传统的块 DMA 以及分散聚合 DMA 更加复杂，在实现时整体架构设计难度也更高。在实际运行过程中，链式 DMA 可以连续不断搬运数据，当描述符执行完成时，如何优化中断上报机制成为关键问题。需要设计一套新的消息机制，既能防止因大量中断上报而导致的系统性能降低，又能及时有效地通知主机数据传输的完成情况，确保主机与 DMA 控制器之间的高效协作。

2）多通道资源优化与调度均衡。多通道设计面临双重技术问题。其一，硬件资源消耗与通道数量呈线性增长关系，以赛灵思 XDMA 为例，单通道消耗约 2 万 LUT，4 通道时消耗约 3.6 万 LUT，显著制约了通道扩展性。其二，多通道间需要设计调度机制，解决各个通道描述符执行不均衡现象，该问题影响系统多任务的公平性。

3）流水化架构设计。在 DMA 设计中，涉及大量的数据处理操作，如去帧头、组帧、数据预取、数据移位等，处理操作涉及控制信息计算、状态转移等，若采用传统的控制信息跟随数据的方式处理，将导致延迟增加、性能降低，因此，设计一套高效的流水化数据处理架构是提升性能的关键。

4）乱序接收重组机制。根据 PCIe 协议规范，非转发型事务需要返回完成报文，如果多个非转发型事务并发，返回的完成报文会产生乱序。因此，需要设计一套机制完成返回报文的缓存管理以及重新定序。虽然主机支持请求事务乱序可以提高处理器的性能，但这无疑为用户设计 DMA 控制器增加了难度。

为了解决上述问题，设计并实现了基于 PCIe 的

高规格、高性能 DMA 控制器，并在国产 FPGA 紫光同创平台进行测试验证，同时兼容英特尔、赛灵思、同创、安路等国内外一系列平台。实验结果表明，基于该 DMA 控制器的 FPGA 异构加速器设备与主机通信性能优越、功能完善，能够满足多种复杂应用场景的需求，填补了国内在该领域的空白。本文主要有 4 个贡献。

1)详细设计并定义链式 DMA 控制寄存器及描述符的格式，其中涵盖了控制寄存器与描述符的偏移量、寄存器名称及其具体功能说明。为了便于理解，本文以主机内存数据向 FPGA 内部地址空间的传输为例，详细阐述了 DMA 配置的具体流程，并提供了完整的操作步骤说明。

2)针对多通道系统的特点，对其实现架构进行了深入设计优化，在保证通道独立性的基本前提下，显著降低逻辑资源使用率。具体优化策略主要包括 2 个方面：首先，采用 FPGA 片上块 RAM 替代传统的寄存器组方案，有效解决了多通道场景下大量寄存器组导致的资源浪费问题。其次，基于状态机设计了各通道描述符轮询机制，结合共享式数据搬移器架构实现了通道间传输资源的时分复用。

3)针对 PCIe 协议中请求与响应独立处理的局限性，创新性地引入了一套控制信息管理机制。通过在系统架构中引入专门的控制信息处理模块，实现了对各类处理的异步调度。该机制能够提前解析并分发控制指令至各功能模块，在数据到达之前完成必要的预处理准备工作。这种改进显著优化了数据处理流程，使得各模块能够无缝衔接地开展工作，最终实现了高效的流水线式数据处理。

4)设计了一套基于 PCIe 的 Tag 管理与返回包乱序重组机制。该机制主要包括 Tag 的分发与回收、返回包数据缓存以及排序等功能模块。具体实现上，采用基于报文并发的 Tag 管理机制，为每个请求分配一个唯一的 Tag 标识，并预先为其分配对应的缓存空间。在缓存后端，实现返回包数据的乱序重组功能，当相关数据包接收完成后，完成 Tag 的释放和回收操作。

通过上述机制的设计与实现，成功提高了系统的带宽利用率，降低了系统延迟以及资源使用率。

## 1 相关工作

### 1.1 PCIe 与 AXI 简述

PCI Express 总线是一种基于点对点串行连接机制的高速互连技术，采用差分信号传输方式，显著提升了抗干扰性能和数据传输的完整性和可靠性。根据实际设计需求，PCIe 物理链路可灵活配置为 x1~x32 等多种链路宽度模式，以适应不同的带宽需求。此外，其传输速率可根据应用场景选择不同的代数标准(如 Gen1~Gen5)，从而实现更高的数据传输效率。

PCIe 层次结构如图 1 所示，主要包括事务传输层、数据链路层和物理层。数据报文在设备核心中生成后，依次通过事务传输层、数据链路层和物理层进行封装和处理，最终通过串行差分信号线发送至目标设备。接收端的数据则依次经过物理层、数据链路层和事务传输层的解封装和处理，最终将解析后的数据送入设备核心中，从而完成整个数据的高效传递[3]。

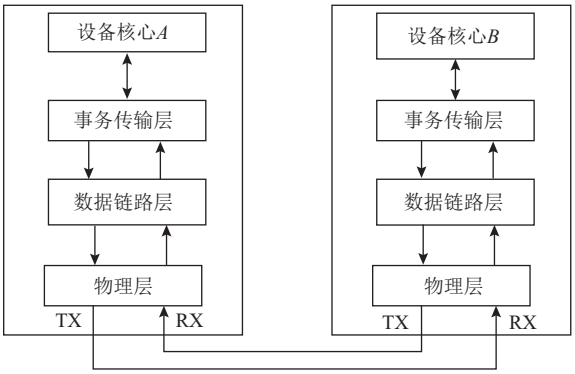


Fig. 1 PCIe hierarchical structure

图 1 PCIe 层次结构

事务层包(transaction layer packet, TLP)是用户层与事务层之间交互的主要机制，其数据头由 3 或 4 个双字(double word, DW)组成，如图 2 所示。其中，保留字段用 *R* 标识，其余字段的定义由具体的事务类型决定。根据 PCIe 协议的规定，*Fmt* 字段用于指示包的格式，*Type* 字段用于指示包的类型，通过 *Fmt* 与 *Type* 的不同组合可以定义多种类型的数据包[4]。本文设计的 DMA 控制器中，涉及的主要 TLP 类型包括存储器读(memory read, Mrd)、存储器写(memory write, Mwr)以及完成数据(completion with data, CplD)。具体而言，当 PCIe 主设备需要访问目标设备的存储器空间时，会使用 Non-Posted 总线事务向目标设备发送 Mrd 请求，目标设备接收到该请求后会通过 CplD 包将存储器中的数据返回给主设备。在 PCIe 总线中，存储器写操作采用 Posted 总线事务，主设备仅需发送 Mwr 包即可完成存储器写入操作，无需等待目标设备的响应报文。值得注意的是，Non-Posted 事务要求目标设备必须返回响应报文以确认数据已成功接

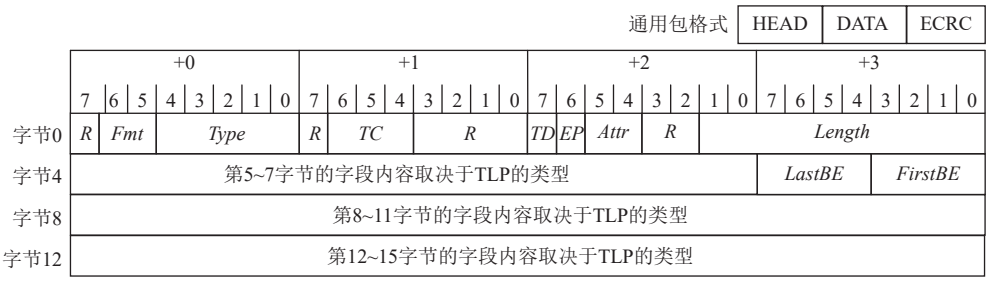


Fig. 2 PCIe packet header format

图 2 PCIe 包头格式

收，而 Posted 事务则不需要目标设备的显式响应[5]。

AXI 协议是 ARM 公司提出的 AMBA（advanced microcontroller bus architecture）协议的核心组成部分，代表了一种高性能片内总线架构。该协议以其高带宽、低延迟和高效能著称，已成为现代嵌入式系统设计中的重要技术。AXI 协议定义了 3 种接口类型：AXI-MM 接口专为高性能存储器映射操作设计，适用于需要地址指定的高速数据传输场景，如内存访问等。AXI-Lite 作为 AXI-MM 的精简版本，主要用于访问低速外设中的寄存器资源。AXI-Stream 接口则专注于高效的流数据传输，特别适用于主从设备之间的连续数据读写操作，其特点在于无需地址标识，从而实现了更高的传输效率。

### 1.2 DMA 控制器相关研究

DMA 是一种高效的数据传输机制，它能够在外部设备与内存以及内存与内存之间实现快速的数据传输操作。与传统的 CPU 控制数据传输方式不同，DMA 传输过程完全独立于 CPU 运行，仅需 CPU 在传输开始和结束时进行少量的初始化和收尾处理。这种设计使得 CPU 在整个数据传输过程中能够专注于执行其他任务，从而实现了任务处理与数据交换的并行执行。当系统配置了 DMA 控制器后，CPU 只需向 DMA 控制器发送一条指令，并设置源地址、目的地址及数据长度等参数，DMA 控制器便会自主接管后续的数据传输任务。在数据传输过程中，DMA 控制器会高效地完成数据从源地址到目的地址的迁移工作。一旦数据传输完成，DMA 控制器会通过中断机制通知 CPU，以便系统能够及时进行后续的数据处理工作。这种机制显著降低了 CPU 的负载率，使 CPU 能够将更多资源投入到核心计算任务中。因此，DMA 技术在现代计算机系统中得到了广泛应用，并成为提升系统性能的重要手段之一。

DMA 根据工作模式可以分为块 DMA、分散聚合 DMA（scatter gather DMA, SGDMA）以及链式 DMA。块 DMA 在进行数据传输时，要求源和目标物理地址空间必须是连续的。然而，在实际应用中，系统分配的存储空间往往并非物理连续，因此块 DMA 方式需要将传输任务分割为多个独立操作。当数据分布于非连续物理地址空间时，块 DMA 模式需反复进行配置并频繁触发中断机制，这不仅降低了 CPU 的工作效率，也影响了数据传输的整体效率。SGDMA 通过构建一个链表来描述非连续物理存储空间。CPU 首先配置链表，然后将链表头地址写入 DMA 控制器寄存器。DMA 控制器会逐一解析并执行链表中的每个描述符，在完成每一块物理连续数据的传输后无需立即触发中断，而是根据链表继续传输下一块物理连续的数据。仅在所有数据传输完成后才统一触发中断。显然，SGDMA 模式相较于块 DMA 模式具有更高的效率。链式 DMA 作为 SGDMA 的扩展实现，在链表的基础上增加了指针，该指针位于链表末尾。当当前链表执行完毕后，DMA 控制器可以根据末尾指针指向下一个链表继续执行。这种机制虽然在控制器实现上更为复杂，但提升了数据传输的灵活性和效率。

在 DMA 控制器开发领域，由于设计复杂度高、研发周期长，国际厂商赛灵思在 2017 年前仅提供简易的 XAPP1052 参考设计方案。该方案采用传统中断机制，可对事务包进行基础处理，配套提供 FPGA 源码及 Linux 驱动程序。其开源性使得科研人员能够基于该参考设计开展功能扩展与优化研究，成为学术界主要研究对象。2017 年后，赛灵思推出升级版 XDMA 控制器，通过引入 MSI/MSI-X 中断机制、多通道架构及链式 DMA 传输技术显著完善功能规格。新方案提供图形化配置界面，支持参数快速设置与系统集成，大幅降低开发门槛。然而，XDMA 以加密源代码形式交付，用户无法像早期 XAPP1052 方案那样进行定制开发。同时，由于 XDMA 功能完备性已满足多数应用场景需求，用户端的定制化开发需求锐减，直接导致学术界相关研究方向热度下降，后续相关研究主要基于赛灵思 XDMA 及英特尔 SGDMA 等厂商提供的闭源解决方案开发应用。

在架构优化层面，研究者多基于赛灵思参考设计进行改良，如 Kavianipour 等人[6]通过简化用户接口实现 750~790 MBps 带宽，Rota 等人[7]通过 FPGA 内部存储描述符方式将块 DMA 升级为 SGDMA，在 Gen2x8 下将带宽提升至 3 461 MBps，Zazo 等人[8]则在官方参考设计的基础上扩展了对 SR-IOV 虚拟化的支持。在专用领域优化方面，文献 [9] 针对 10 Gbps 以太网控制器开发了支持平衡任务优先级的 SGDMA，Forencich 等人[10]在刚玉平台实现了 100 Gbps 网络定制 DMA，支持传输、接收、完成和事件队列，对于网络专用 DMA 设计具有参考意义，而文献 [11] 中则采用英特尔官方多 DMA 控制器并行的方式，突破了 400 Gbps 网络传输瓶颈。在异构计算领域，Kasai 等人[12]利用 XDMA 验证了 FPGA-GPU 直传可行性，但传输效率仍未达实用标准。此外，文献 [13] 中尝试用高层次综合语言开发 DMA 控制器，虽然解决了传统的开发效率问题，但是实际性能不足。

当前主流方案多依赖赛灵思 XDMA 或英特尔 SGDMA 直接开发应用[14-17]，自主架构设计研究明显不足。尤其在国内，同创、安路等厂商尚未形成成熟商用的 DMA 控制器产品，这一现状导致 2 个关键问题：其一，现有改良方案存在平台依赖性强、功能扩展性差等固有缺陷[18-22]；其二，国际技术垄断加剧了我国在该领域的技术风险。因此，研发具备全功能规格、性能高、跨平台兼容的自主化 DMA 控制器，已成为突破异构计算通信瓶颈和保障产业链安全的核心攻关方向之一。

## 2 DMA 控制器设计

### 2.1 DMA 控制寄存器与描述符设计

DMA 控制器的实现基础是控制寄存器与描述符设计，本文结合 PCIe 与通用 DMA 技术，自主设计了一套多通道链式 DMA 控制寄存器与描述符方案。控制寄存器分为读描述符控制寄存器和写描述符控制寄存器，分别对应 H2C(host to card)和 C2H(card to host)方向。读写描述符存储在 PCIe 系统内存中，每个描述符表最多可存储 128 个描述符，每个描述符由 8 个双字组成。描述符存储表采用状态表加描述符表的结构形式，其中状态表位于描述符表之前。状态表的起始地址为 H2C 描述符表基地址或 C2H 描述符表地址，描述符表的起始地址为基地址加固定偏移地址 0x200。DMA 开始寄存器用于触发 DMA 控制器来获取描述符表。因此，在配置 DMA 传输时，设置开始寄存器必须作为最后一步操作。

表 1 描述了 DMA 控制寄存器的配置。DMA 描述符控制寄存器位于 PCIe 的基地址寄存器(base address register, BAR)空间上。H2C 控制寄存器的偏移地址范围为 0x0000~0x01FF，共支持 16 个通道，每个通道占用 0x20 的地址空间。C2H 控制寄存器的偏移地址范围为 0x0200~0x03FF，同样支持 16 个通道，每个通道占用 0x20 的地址空间。

表 2 为 DMA 描述符格式，共 8 个描述符，每个描述符占用 32 b。H2C 和 C2H 各有一套描述符格式。每个描述符不仅记录了当前 DMA 传输所需的地址、数据量等信息，还在链表末尾包含下一个链表的起始位置及相关信息，从而实现链式 DMA 传输，减少了 CPU 对中断的频繁处理。

### 2.2 DMA 配置流程

本节示例展示了 3 个数据块从主机系统内存移动到 FPGA 总线地址空间的过程。图 3 详细说明了 PCIe 和 FPGA 总线地址空间中数据块的位置和大小分布，而图 4 为具体的描述符格式示意图。

主机配置执行的步骤如下：

步骤 1.计算分配所需的内存。

① 状态表中的每个条目为 4 B，128 个条目需要

**Table 1 Control Registers**

**表 1 控制寄存器**

| 偏移 | 寄存器 | 描述 |
|---|---|---|
| 0x0000 | Descriptor L | 描述符表存储地址低 32 b。 |
| 0x0004 | Descriptor H | 描述符表的存储地址高 32 b。 |
| 0x0010 | LAST_PTR | 表示最后 1 个读描述符 ID（0~127）。 |
| 0x0014 | FIRST_PTR | 表示第 1 个读描述符 ID（0~127）。 |
| 0x0018 | CONTROL | [31:2] 保留。<br>[1]Ctrl。当设置为 1，本次 DMA 不上报中断；当设置为 0，在最后 1 个描述符完成后发送中断。<br>[0]Done。当设置为 1，每个完成描述符在状态表中 Done 位置写 1；当设置为 0，仅在 DMA_LAST_PTR 对应的 Done 位置写 1。 |
| 0x001C | DMA_START | 写该寄存器任意值来触发 DMA 开始。 |

**Table 2 Descriptor Format**

**表 2 描述符格式**

| 偏移 | 寄存器 | 描述 |
|---|---|---|
| 0x00 | LOW_SRC_ADDR | 源地址低 32 b，PCIe 系统内存地址。 |
| 0x04 | HIGH_SRC_ADDR | 源地址高 32 b。 |
| 0x08 | LOW_DEST_ADDR | 目的地址低 32 b，FPGA 内部总线地址。 |
| 0x0C | HIGH_DEST_ADDR | 目的地址高 32 b。 |
| 0x10 | LENGTH | [31] 1：指示该描述符含有下一个可执行链表信息；0：表示无。<br>[30:29] 为下一个链表信息的控制信号。<br>[30] Ctrl。当设置为 1，执行完链表不上报中断；当设置为 0，在最后 1 个描述符完成后上报中断。<br>[29]Done。当设置为 1，为每个完成的描述符在状态表中 Done 位置写 1；当设置为 0，仅在 DMA_ LAST_PTR 对应的 Done 位置写 1。<br>[28] 保留。<br>[27:0] 传输大小，单位为 B，不能为 0。 |
| 0x14 | DESC_ID | [31:16] magic 值，该数值用于验证驱动描述符是否配置正确。<br>[15:8] 下一个链表的 last_ptr。<br>[7] 保留。<br>[6:0] 指定描述符的 ID，最大为 127。 |
| 0x18 | Next Desc Addr L | 下一链表描述符存储低位地址 |
| 0x1C | Next Desc Addr H | 下一链表描述符存储高位地址 |

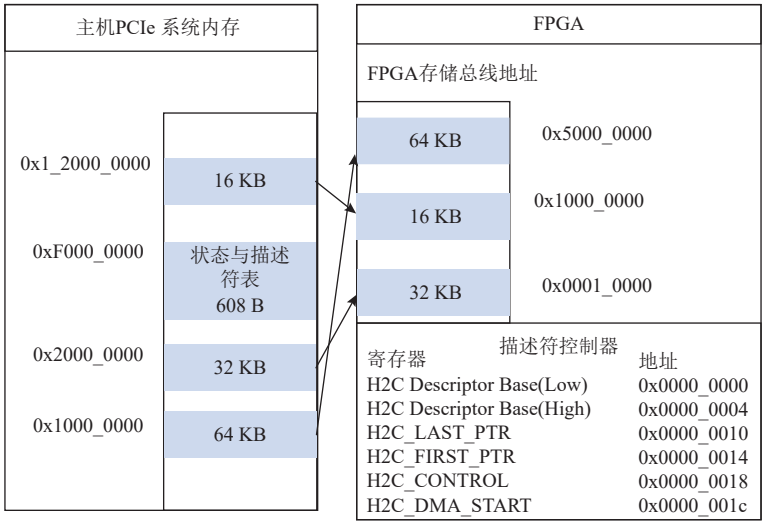


Fig. 3 Schematic diagram of system to FPGA internal bus address space

图 3 系统到 FPGA 内部总线地址空间示意图

512 B 的内存。

②每个描述符是 32 B。3 个描述符需要 96 B 的内存，状态和描述符表的总内存分配为 608 B。

步骤 2.在 PCIe 地址空间中分配 608 B 的内存。此示例中分配内存的起始地址是 0xf000_0000，将此地址写入 H2C 描述符控制寄存器中。

步骤 3.基地址为 0xF000_0000，那么第 1 个描述符的地址为 0xF000_0000+0x200。

①在 0xF000_0204 中写入高 32 b 源地址 0x0。

②在 0xF000_0200 中写入低 32 b 源地址 0x1000_0000。

③在 0xF000_0200c 中写入高 32 b 目的地址 0x0。

④在 0xF000_0208 中写入低 32 b 目的地址 0x5000_0000。

⑤在 0xF000_0210 中写入 0x0001_0000，长度 64 KB。

⑥在 0xF000_0214 中写入描述符 ID 0。

⑦在 0xF000_0218 中写入 magic 值 ad4b。

步骤 4.重复步骤 1~3，写入另外 2 个描述符，写完后如图 5 所示。

步骤 5.通过 BAR0 来配置 H2C 控制寄存器基地址，本例使用 64 b 地址。

①在 0x0000_0004 中写入 0x0000_0000，高 32 b 读描述符基地址。

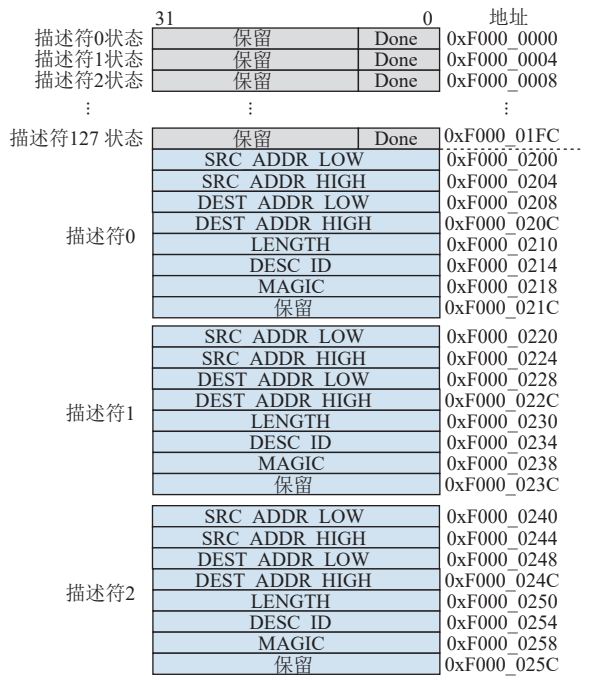


Fig. 4 Schematic diagram of descriptor configuration table format

图 4 描述符配置表格式示意图

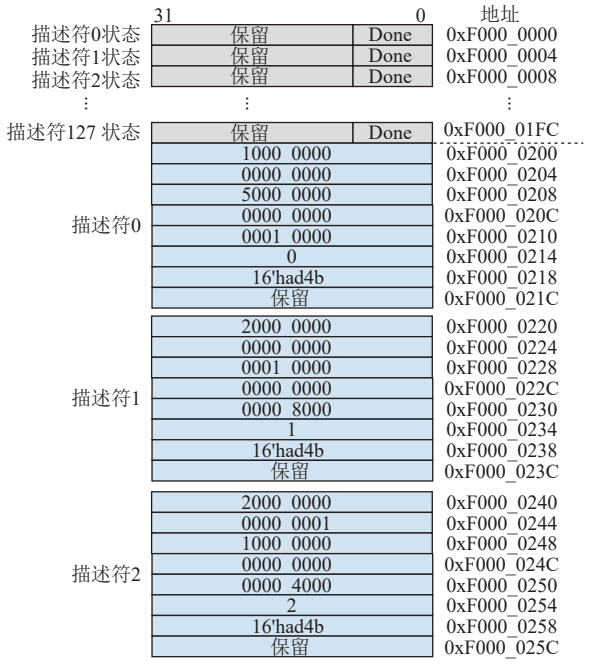


Fig. 5 Schematic diagram of the configured descriptor

图 5 配置后的描述符示意图

②在 0x0000_0000 中写入 0xF000_0000，低 32 b 读描述符基地址。

③描述符地址在此基础上加 0x200(无需驱动操作)。

步骤 6.写寄存器 H2C_LAST_PTR，在 0x0000_0010 写入 0x2。

步骤 7.写寄存器 H2C_FIRST_PTR，在 0x0000_0014 写入 0x0。

步骤 8.写寄存器 H2C_CONTROL，在 0x0000_0018 写入 0x0。

步骤 9.写寄存器 H2C_DMA_START，在 0x0000_001c 写任意值，启动 DMA 读。

## 3 DMA 控制器逻辑实现

PCIe 作为高速串行总线协议，其分层协议为外设互联提供了高带宽、低延迟的通信基础，而直接内存访问技术作为提升系统数据吞吐效率的核心机制，其实现方式并未在 PCIe 协议中做强制性规定，需结合具体应用场景进行设计与实现。本文根据自定义控制寄存器、描述符以及运行流程，实现了多通道链式 DMA 控制器，其中 DMA 控制器逻辑实现框架如图 6 所示。

PCIe 硬核模块实现物理层、数据链路层及事务传输层协议处理，DMA 通过主机请求、设备请求、设备返回、主机返回及中断请求等接口与硬核进行交互，接口遵循 AXI-Stream 协议规范。DMA 控制器核心组成包含描述符管理、数据搬移器、整帧缓存、状态上报、接口转换、中断管理、接口标准化等模块，各模块间通过流模式进行数据传输以保障传输效率。基地址寄存器管理模块负责将主机对 BAR 空间的访问转换为 APB 总线操作，其中 BAR0 用于 DMA 功能配置，BAR1~BAR5 负责用户业务参数配置。描述符管理模块通过 APB 总线与主机交互，实现描述符存储与调度管理功能。数据搬移器接收并解析描述符指令，除常规数据搬运外，主机端的描述符数据同样需通过数据搬移器传输至设备端。整帧缓存模块对 Mwr TLP 包实施至少单包缓存后再进行完整帧输出，该机制旨在避免流模式传输时帧内出现多拍无效间隔导致的 PCIe 链路错误。状态上报模块将每个描述符完成状态信息封装为 Mwr TLP 包回传主机，主机为每个描述符预先分配状态标志存储空间，当描述符处理完成后即向对应主机地址写入完成标志。由于数据搬移器的输入/输出数据流未处理突发传输及地址边界对齐等问题，接口标准化模块负责将流数据转换为符合 AXI-MM 标准的协议格式。

### 3.1 多通道与调度均衡

针对多通道的特点，本文在实现上做了如下优化，如图 7 所示。为节省逻辑资源，本文将一块片上 RAM 分为 16 份，分别存储 16 个通道的控制寄存器，

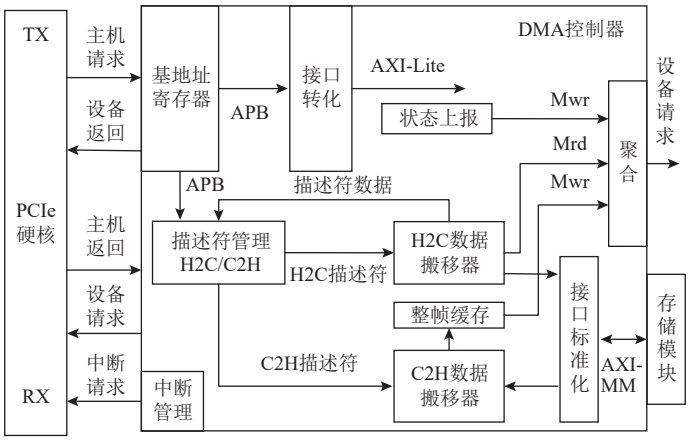


Fig. 6 DMA controller architecture diagram

图 6 DMA 控制器架构图

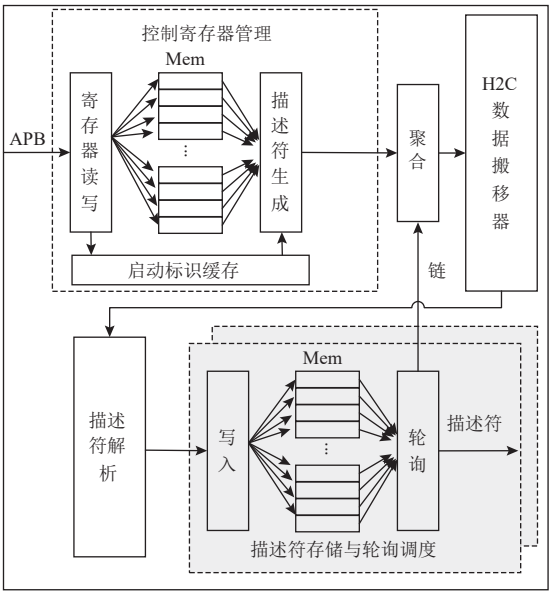


Fig. 7 Multi-channel and scheduling balance

图 7 多通道与调度均衡

同时，将 DMA 启动标识缓存到 FIFO 中，通过状态机将配置的控制寄存器逐个读出，生成 H2C 描述符，该描述符为特殊描述符，利用数据搬移器将主机中的描述符表中的数据搬移到设备中，这里与普通数据共用搬移器以节省资源。描述符数据在解析后，分别存储到深度为 16×128（通道×单次最大描述符个数）RAM 中，按照轮询调度算法调度每个通道。当每个通道最后一个描述符被执行后，需要检查是否有链式标志位，如果有，需要生成新的描述符，将下一个链表内的描述符搬运到设备中继续执行。

### 3.2 流水化架构设计

本文创新性地提出了一种基于流水化架构的数据搬移器设计方案，通过将预处理后的描述符信息进行异步分发，使各个数据关键处理单元能够并行执行。该架构采用多级流水线缓存结构，构建了描述符解析与数据处理相分离的协同工作机制，相较于传统串行处理模式中控制信息与数据流同步传递的方式，实现了处理单元间的零等待数据传递，这一架构突破性地解决了数据搬移过程中控制流与数据流耦合度过高的瓶颈问题，为高速数据传输控制系统提供了新的设计范式。

如图 8 所示，在 H2C 数据搬移器架构中，描述符预处理模块首先解析原始描述符，动态计算目标地址、数据长度等关键参数，并通过异步分发机制将控制信息广播至流水线各处理单元，同时基于 PCIe 协议（包括最大读请求大小限制与 4 KB 地址对齐规则），执行跨界拆分逻辑，将连续内存访问请求拆分为多个符合规范的 Mrd TLP。当 CplD 报文到达接收端后，处理流程依次包含：1）帧头剥离，提取有效负载数据。2）依据 Tag 标识符将分片数据路由至独立缓存分区，实现乱序到达分片的动态重组。3）根据描述符总长度将多个 Tag 携带数据组成一包数据以完成数据拼接，同时产生字节掩码。4）针对目的地址非 0 的情况，通过移位对齐，确保数据与地址对应。5）采用多级流水式位宽转换器，将重组数据按目标

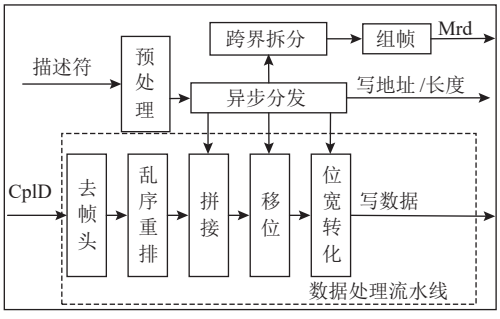


Fig. 8 H2C data mover

图 8 H2C 数据搬移器

存储器的位宽要求(如 64 b/128 b/256 b)进行位宽转化。整个数据通路通过握手协议实现背压控制与吞吐量优化,消除流水线间气泡,实现全流水处理。

如图 9 所示,在 C2H 数据搬移器架构中,描述符首先通过预处理模块进行参数解析,系统会同步计算出存储器读取操作的源地址/长度以及流水线处理模块所需的源地址/长度等配置,这些控制信息通过异步分发机制分别传输至存储器读控制管理单元和流水线处理模块。跨界拆分模块依据 PCIe 协议的最大载荷及 4 KB 地址边界约束条件执行跨界地址拆分算法。从存储器获取的原始数据流需经过多层处理:1)通过读地址与长度将描述符规定的所有数据读出。2)对读出的原始数据增加掩码标志,然后移位,将掩码标志为 0 的空数据移除。3)根据系统总线位宽(如 512 b)与存储总线位宽(如 256 b)的差异进行自适应位宽转换。4)经处理后的数据流暂存于高速 FIFO 缓冲队列,同时设计预缓存机制,缓存 2 拍数据,优化组帧模块取数据效率。5)根据拆分后的目的地址/长度等将缓存的数据按 Mwr TLP 协议规范进行组装。

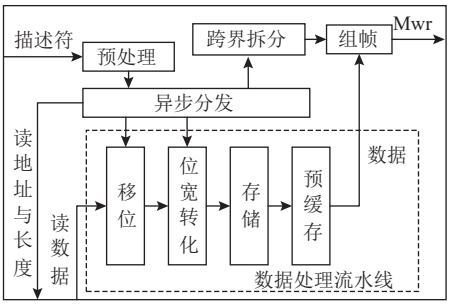


Fig. 9 C2H data mover

图 9 C2H 数据搬移器

### 3.3 Tag 管理与乱序重组

在 PCIe DMA 读事务中,设备端依据最大读请求大小与 4 KB 地址对齐规则对 DMA 描述符进行动态分片,生成符合规范的 Mrd TLP。主机端响应时,根据读完成边界(read completion boundary, RCB)参数对返回数据进行多次分片,形成多个 CplD 报文。由于 PCIe 协议允许不同的 CplD 乱序传输,Tag 作为 CplD 报文中的唯一标识符,成为解决乱序问题的关键要素。

如图 10 所示,本文采用基于累加计数器的 Tag 顺序分配策略,通过参数化配置 Tag 位宽动态调整 Tag 池容量以适配不同场景的并发请求需求。每个 Tag 关联独立分配的块 RAM 存储区,其深度由下式计算:深度=Tag 数×lb(最大读请求×8/数据位宽),确保不同 Tag 的数据分片物理隔离。每个存储区配置专属状态标志,实时跟踪分片到达状态。当某 Tag 所有分片到达后标志被置位,触发该 Tag 数据读取,按分片序号升序重构原始数据流,读取完毕后,Tag 管理模块计数器加 1,完成 Tag 回收。

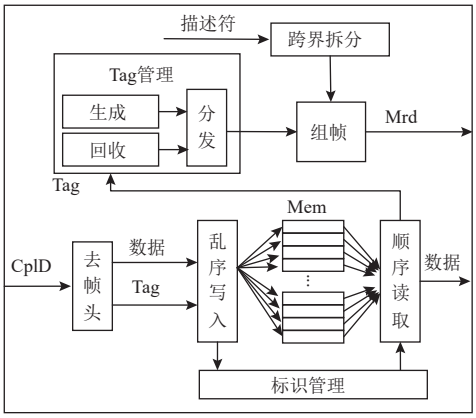


Fig. 10 Tag management and disorderly reorganization

图 10 Tag 管理与乱序重组

## 4 实验结果分析

本文设计并实现了一种基于 PCIe 的 DMA 控制器,并在自主研发的硬件板卡 FP600 加速器上完成了功能验证。如图 11 所示为硬件板卡的实物图。实验采用同创 Titan3 系列 FPGA 芯片 PG3T600P,该芯片理论上支持 PCIe Gen4x8 接口。由于当前 FPGA 厂商尚未提供适用于 PCIe Gen4x8 的硬核,本研究选用 Gen3x8 硬核进行测试及对比,理论最大传输速率为 8 GBps。除国产平台,本文 DMA 控制器也在赛灵思平台进行了测试,其测试性能与国产平台几乎一致。此外,本文还分别在英特尔 Arria 10 平台和赛灵思 Virtex7 平台上测试了英特尔 SGDMA 和赛灵思 XDMA 的性能表现,最终给出了 3 种 DMA 控制器的规格参数及性能指标对比表。

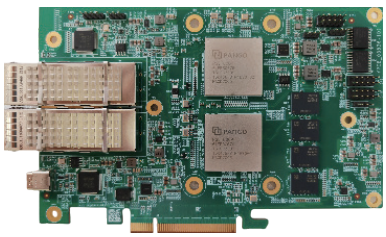

Fig. 11 Hardware board diagram

图 11 硬件板卡图

本文设计的 DMA 控制器最大支持 16 个通道传输。为了便于说明和验证性能,选择其中 4 个通道进行测试,多通道 DMA 控制器测试结果如图 12 所示。

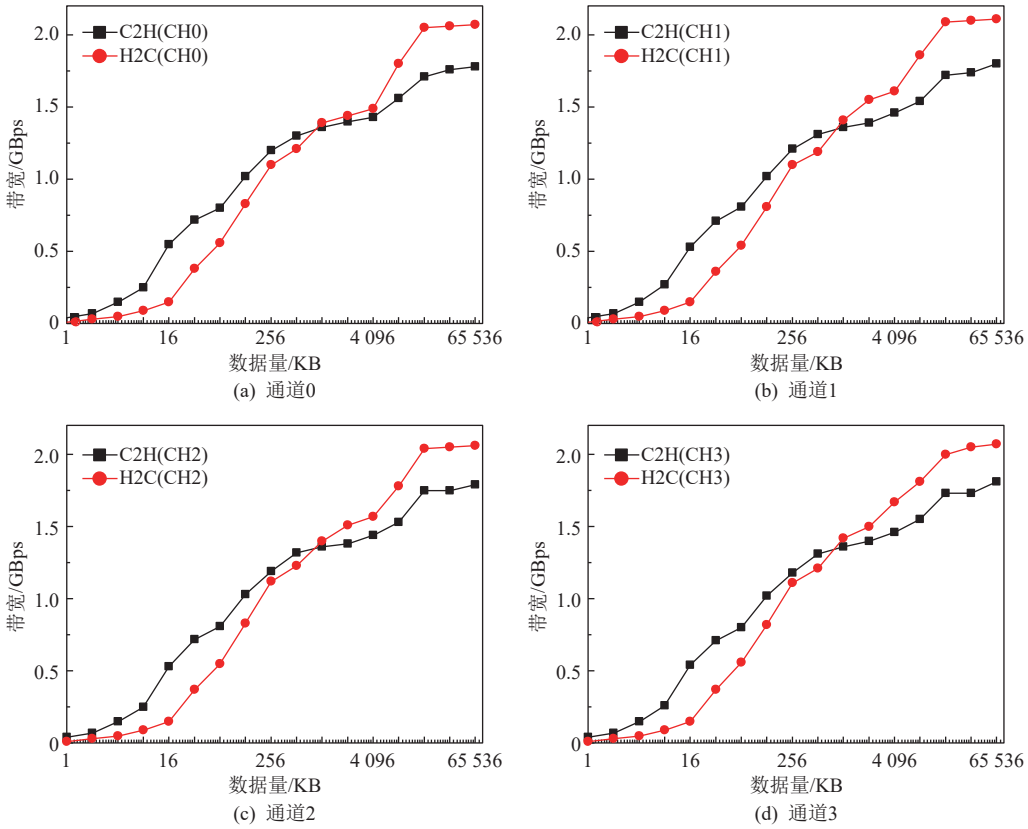


Fig. 12 DMA performance test chart under four channels

图 12 4 通道下 DMA 性能测试图

图中展示了 CH0~CH3 四个通道的 DMA 读写带宽实验结果，所有数据均在 4 个通道同时工作的情况下获取。

从图 12 可以看出，随着数据包长度的增加，带宽呈现显著增长趋势。这一现象表明，在数据传输过程中采用一次性执行 128 个描述符与链式机制的组合策略能够有效提升系统性能。当 4 个通道同时工作时，每个通道的带宽性能基本一致，能够有效共享 PCIe 总线带宽。这一现象与设计中各通道基于描述符轮询使用数据搬移器的机制完全吻合。具体而言，搬移器通过一种高效的轮询机制来管理多通道的数据传输任务。其核心工作原理是：搬移器会按照预设的顺序循环遍历每一个通道的描述符队列，逐一检查并处理其中待完成的数据搬移任务。这种机制确保了每个通道都能获得均等的处理机会，从而实现了多通道之间的公平带宽分配。

主机端的数据拷贝模式会影响 DMA 传输的执行效率，本文分别测试了零拷贝与非零拷贝模式下的 DMA 读写带宽性能，实验结果如图 13 所示。实验结果表明，得益于控制信息预处理异构分发机制和数据处理流水化架构设计，本文设计的 DMA 在性能上体现出显著优势。在非零拷贝下，H2C 方向的

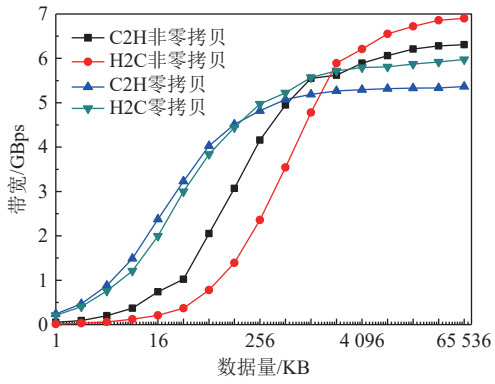


Fig. 13 Comparative diagram of DMA performance under different modes

图 13 不同模式下 DMA 性能对比图

最大带宽达到 6.91 GBps，约为理论带宽的 86%；C2H 方向的最大带宽为 6.31 GBps，约为理论带宽的 79%。在零拷贝模式下，H2C 方向的最大带宽为 5.67 GBps，C2H 方向的最大带宽为 5.37 GBps。测试过程中，自研加速器板卡与主机协商确定的最大有效负载为 256 B，最大读请求为 512 B。因此，H2C 方向的最大带宽高于 C2H 方向的结果符合预期。从实验曲线可以看出，在初始阶段，H2C 方向的带宽低于 C2H 方向的。这是因为 C2H 方向仅涉及设备向主机发送数据

的过程，而 H2C 方向需要设备发送读请求并处理主机返回的数据包，其流程更为复杂。因此，在初始阶段，C2H 方向的效率更高。

此外，图 13 还显示，在 1 MB 数据量以下的场景中，零拷贝模式展现出更高的性能优势；而当数据量超过 1 MB 时，非零拷贝模式则更具效率优势。从技术原理角度来看，零拷贝机制以部分灵活性和兼容性为代价，换取了显著的性能提升，特别适用于对吞吐量和延迟敏感的应用场景，以及需要实时处理的小文件传输需求。相比之下，非零拷贝机制则在通用性和开发便捷性方面表现更为突出，能够更好地满足高性能、大规模数据传输的应用需求。基于上述分析，建议根据具体的业务场景和性能要求，选择最合适的拷贝模式以实现最优效果。

英特尔的 SGDMA 控制器相对简单，缺乏链式传输和多通道支持等高级功能，本文未将其作为重点对比对象。在零拷贝模式下，本文设计的 IDMA 与赛灵思 XDMA 的性能对比数据如图 14 所示。可以看出，在数据量为 1 KB 至 64 MB 的范围内，IDMA 效率明显优于 XDMA，最大带宽高出约 16%。

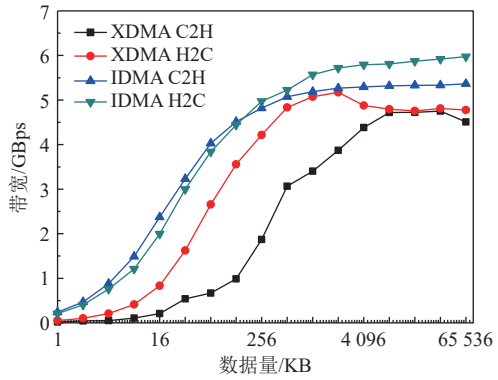


Fig. 14 Comparative diagram of performance for XDMA and IDMA

图 14 XDMA 与 IDMA 性能对比图

此外，本文对 DMA 控制器的规格和性能与赛灵思、英特尔的闭源产品进行了全面对比，具体结果见表 3。通过对比可以看出，IDMA 在规格和性能方面均达到了或超越了国外厂商的水平。特别是在多通道设计上，本文的设计明显优于赛灵思的方案，且资源占用更为节省。赛灵思的 XDMA 在单通道下占用 2 万 LUT 资源，4 通道下占用 3.6 万 LUT 资源。IDMA 在单通道下仅占用 1.6 万 LUT 资源，16 通道下也仅占用 1.7 万 LUT 资源，资源增长幅度仅为 0.1 万。相比之下，XDMA 最多仅支持 4 通道设计，并且通过大量占用逻辑资源来实现通道扩展。IDMA 通过优化多通道实现架构，成功避免了这一问题。

**Table 3 Performance Comparison for DMA Specification**

**表 3 DMA 规格性能对比**

| 规格 | SGDMA（英特尔） | XDMA（赛灵思） | IDMA（本文） |
|---|---|---|---|
| 测试平台 | 英特尔 Arria | 赛灵思 Virtex7 | 同创 Titan3 |
| SR-IOV 虚拟化 | 不支持 | 不支持 | 支持 |
| 虚拟设备数量 | | | 128 |
| 最大描述符数 | 128 | 64 | 128 |
| 链式 | 不支持 | 支持 | 支持 |
| 最大通道数 | 1 | 4 | 16 |
| 带宽利用率/% | 81.3 | 60 | 86 |
| 驱动零拷贝 | 不支持 | 支持 | 支持 |
| 非零拷贝 | 支持 | 不支持 | 支持 |
| 描述符轮询 | 不支持 | 支持 | 支持 |
| LUT 使用数量 | 1.2 万 | 单通道 2 万<br>4 通道 3.6 万 | 单通道 1.6 万<br>16 通道 1.7 万 |

## 5 总 结

本文在对 PCIe 协议和 AXI 协议总线进行深入理论研究的基础上，设计并实现了一种基于 FPGA 的多通道链式 DMA 控制器，并在自主研发的加速器 FP600 上成功验证。该设计有效解决了数据中心服务器与国产 FPGA 设备之间的高效数据传输及业务加速需求。在本设计中，对寄存器配置、描述符管理、系统架构等关键部分进行了详细介绍，并提供了具体的设计思路和使用流程步骤。此外，详细阐述了多通道的具体实现架构、通道均衡机制以及资源优化方案。通过设计 Tag 管理与返回包缓存机制，实现了数据包的乱序重组。同时，采用信息预处理和异步分发机制，实现了数据的流水化处理，显著提高了带宽利用率，使实际带宽基本接近理论极限。本文设计的 DMA 控制器支持最大 16 个通道链式传输，实验与分析表明，在 H2C 方向的最大带宽为 6.91 GBps，约为理论带宽的 86%；在 C2H 方向的最大带宽为 6.31 GBps，约为理论带宽的 79%。在系统性能与规格方面，本文设计的 DMA 控制器与英特尔的 SGDMA 和赛灵思的 XDMA 相比均具有竞争力，自主研发的 DMA 控制器不仅适用于数据中心 FPGA 加速器，还可广泛应用于云计算平台、智能网卡、人工智能等多种场景。

**作者贡献声明**：王洪良、郭振华提出研究思路，

设计系统整体架构并撰写论文；王洪良、牟奇、卢圣才负责工程设计开发、实验以及架构优化；徐亚明、于泉泉负责部分论文内容撰写并提出修改建议。

## 参 考 文 献


[1] Kastner. RIFFA: Reusable integration framework for FPGA accelerators [CP/OL]. San Francisco, CA: GitHub, (2015-05-04) [2025-05-08]. https://github.com/KastnerRG/riffa

[2] scryver. XAPP1052 [CP/OL]. San Francisco, CA: GitHub, (2019-01-08) [2025-05-08]. https://github.com/scryver/XAPP1052

[3] Wang Qi. Introduction to PCI Express Architecture [M]. Beijing: Machinery Industry Press, 2016 (in Chinese)
(王齐. PCI Express 体系结构导读[M]. 北京：机械工业出版社，2016)

[4] PCI-SIG. PCI Express Base Specification Revision 3.0[S/OL]. Beaverton, OR: PCI-SIG, 2010[2025-05-08]. https://www.pcisig.com

[5] Budruk R, Anderson D, Shanley T. PCI Express System Architecture[M]. Boston, MA: Addison-Wesley Professional, 2003

[6] Kavianipour H, Muschter S, Bohm C. High performance FPGA-based DMA interface for PCIe[J]. IEEE Transactions on Nuclear Science, 2014, 61(2): 745−749

[7] Rota L, Caselle M, Chilingaryan S, et al. A PCIe DMA architecture for multi-Gigabyte per second data transmission[J]. IEEE Transactions on Nuclear Science, 2015, 62(3): 972−976

[8] Zazo J F, Lopez-Buedo S, Audzevich Y, et al. A PCIe DMA engine to support the virtualization of 40 Gbps FPGA-accelerated network appliances[C]//Proc of the 2015 Int Conf on ReConFigurable Computing and FPGAs (ReConFig). Piscataway, NJ: IEEE, 2015: 1−6

[9] Wu Deze, Duan Guodong, Ren Minhua, et al. Research and design of descriptor-based DMA for 10 Gigabit Ethernet controller[C]//Proc of the 6th Int Conf on Electronic Engineering and Informatics (EEI). Piscataway, NJ: IEEE, 2024: 855−859

[10] Forencich A, Snoeren A C, Porter G, et al. Corundum: An open-source 100-Gbps Nic[C]//Proc of the 28th Annual Int Symp on Field-Programmable Custom Computing Machines (FCCM). Piscataway, NJ: IEEE, 2020: 38−46

[11] Kubálek J, Cabal J, Špinler M, et al. DMA Medusa: A vendor-independent FPGA-based architecture for 400 Gbps DMA transfers[C]//Proc of the 29th Annual Int Symp on Field-Programmable Custom Computing Machines (FCCM). Piscataway, NJ: IEEE, 2021: 258−258

[12] Kasai S, Osana Y. A driver-based approach for DMA transfer between FPGA-GPU[C]//Proc of the 10th Int Symp on Computing and Networking Workshops (CANDARW). Piscataway, NJ: IEEE, 2022: 103−108

[13] Zabołotny W M. Versatile DMA engine for high-energy physics data acquisition implemented with high-level synthesis[J]. Electronics, 2023, 12(4): 883

[14] Yokono T, Yamabe Y, Tanaka K, et al. FPGA-based accelerators system with low latency autonomous DMA engine[C]//Proc of the 30th Annual Int Symp on Field-Programmable Custom Computing Machines (FCCM). Piscataway, NJ: IEEE, 2022: 1−1

[15] Zhang Honglin, Yang Haibo, Liu Chengcheng, et al. Design of a common high-performance data acquisition system for physics experiments at HIAF[J]. IEEE Transactions on Nuclear Science, 2024, 72: 713−719

[16] Kobayashi R, Fujita N, Yamaguchi Y, et al. Multi-hybrid accelerated simulation by GPU and FPGA on radiative transfer simulation in astrophysics[J]. Journal of Information Processing, 2020, 28: 1073−1089

[17] Zhao Yingxiao, Liu Xin, Yang Jiong. A resource and timing optimized PCIe DMA architecture using FPGA internal data buffer[J]. IEICE Electronics Express, 2019, 16(1): 20180858

[18] Cheng Kun, Liao Shengkai, Shen Qi, et al. Design and implementation of high-throughput PCIe with DMA architecture between FPGA and powerPC[J]. arXiv preprint, arXiv: 1809.07702, 2018

[19] Zabołotny W M, Gumiński M, Kruszewski M, et al. Control and diagnostics system generator for complex FPGA-based measurement systems[J]. Sensors, 2021, 21(21): 7378

[20] Shanehsazzadeh F, Sadri M S. Area and performance evaluation of central DMA controller in Xilinx embedded FPGA designs[C]//Proc of the 2017 Iranian Conf on Electrical Engineering (ICEE). Piscataway, NJ: IEEE, 2017: 546−550

[21] Wang Yanwei, Li Rengang, Xu Ran, et al. Data center heterogeneous acceleration software-hardware system-level platform based on reconfigurable architecture[J]. Journal of Computer Research and Development, 2025, 62(4): 963−977 (in Chinese)
(王彦伟, 李仁刚, 徐冉, 等. 基于可重构架构的数据中心异构加速软硬件系统级平台[J]. 计算机研究与发展, 2025, 62(4): 963−977)

[22] Chandra K, Jagtap A P, Ranjan N, et al. Design of PCIe-DMA bridge interface for high speed ethernet applications[C]//Proc of the 2019 2nd Int Conf on Advanced Computational and Communication Paradigms (ICACCP). Piscataway, NJ: IEEE, 2019: 1−5



**Wang Hongliang**, born in 1991. Master, engineer. His main research interests include computer architecture, digital chip design, and high-performance computing.

**王洪良**, 1991 年生。硕士，工程师。主要研究方向为计算机体系结构、数字芯片设计、高性能计算。



**Guo Zhenhua**, born in 1988. PhD, senior engineer. His main research interests include computer architecture, distributed systems, and multimodal artificial intelligence.

**郭振华**, 1988 年生。博士，高级工程师。主要研究方向为计算机体系结构、分布式系统、多模态人工智能。

**Mu Qi**, born in 1989. Master, engineer. Member of CCF. His main research interests include computer architecture and high-performance computing.

**牟　奇**, 1989 年生。硕士，工程师。CCF 会员。主要研究方向为计算机体系结构、高性能计算。

**Lu Shengcai**, born in 1986. Master, engineer. His main research interests include computer architecture, digital chip design, 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体系结构、数字芯片设计、高性能计算。

**Xu Yaming**, born in 1986. Bachelor, engineer. His main research interests include computer architecture, computer network, and high-performance computing.

**徐亚明**, 1986 年生。学士，工程师。主要研究方向为计算机体系结构、计算机网络、高性能计算。

**Yu Quanquan**, born in 1987. Master, engineer. His main research interest includes system design and hardware development in the fields of cloud computing servers and storage servers.

**于泉泉**, 1987 年生。硕士，工程师。主要研究方向为云计算服务器、存储服务器领域的系统设计和硬件开发。