doi:10.3969/j.issn.1001-4616.2024.03.014

弱耦合协处理器设计方法研究

——以人工智能应用为例

严忻恺[1,2],陈芳园[3]

(1.浙江大学工程师学院,浙江 杭州 310000)
(2.江苏信息职业技术学院,江苏 无锡 214153)
(3.江南计算技术研究所,江苏 无锡 214000)

[**摘要**] 近些年随着人工智能、大数据、元宇宙等应用的蓬勃发展和半导体工艺进步的放缓,软件应用与硬件性能之间出现了巨大的算力鸿沟,通过软硬件协同设计的特定领域架构作为应对方案得到了学术界和工业界的广泛关注和认可. 所以针对特定领域应用的核心需求设计专用协处理器,研究专用协处理器的设计方法,对于提高软件应用性能和效率,提升硬件设计效率等问题具有重大意义. 本文分析了不同耦合度和不同负载需求的协处理器设计空间,重点研究了弱耦合协处理器的设计方法,包括基于 RISC-V 定制指令设计协处理器指令架构、弱耦合协处理器在不同应用场景下的控制交互接口、访存接口和设计框架;同时归纳总结了人工智能应用的共性需求和人工智能协处理器研究现状;并给出了两种面向不同人工智能应用场景的弱耦合协处理器设计实例,为提高协处理器设计效率提供了有效支撑.

[**关键词**] 协处理器,领域特定架构,弱耦合,RISC-V,人工智能
[**中图分类号**]TP332　[**文献标志码**]A　[**文章编号**]1001-4616(2024)03-0112-10

Research on Design Method of Weakly-Coupled Coprocessor: A Case Study of Artificial Intelligence Application

Yan Xinkai[1,2], Chen Fangyuan[3]

(1.Polytechnic Institute, Zhejiang University, Hangzhou 310000, China)
(2.Jiangsu Vocational College of Information Technology, Wuxi 214153, China)
(3.Jiangnan Institute of Computing Technology, Wuxi 214000, China)

Abstract: Recent years, with the rapid development of artificial intelligence, big data, meta-universe and other applications as well as the slowing down of semiconductor technology progress, there has been a huge computing gap between software application and hardware performance. As a solution, domain-specific architecture through software and hardware co-design has been widely concerned and recognized by academia and industry. Therefore, it is of great significance to design specific-coprocessor and research the design method of coprocessor for the core requirements of specific applications in order to improve the performance and efficiency of software application and improve the efficiency of hardware design. This paper analyzes the coprocessor design space of different coupling degree and different load requirements, and focuses on the design method of weakly coupled coprocessor, including the design of coprocessor instruction architecture based on RISC-V custom instruction, the control interaction interface, access interface and design framework of weakly coupled coprocessor in different application scenarios. The general requirements of artificial intelligence applications and the research status of artificial intelligence coprocessor are summarized, finally two design cases of weakly-coupled coprocessor for different AI application scenarios are described, which provides an effective support for improving the design efficiency of coprocessor.

Key words: coprocessor, DSA, weak-coupled, RISC-V, AI

收稿日期:2023-05-22.
基金项目:江苏省高等学校基础科学(自然科学)研究重大项目(24KJA510005).
通讯作者:严忻恺,博士,讲师,研究方向:计算机体系结构和芯片设计等. E-mail:yanxinkai@zju.edu.cn

近些年来,人工智能、大数据、元宇宙等新兴应用的发展正在掀起新一轮的技术革命,但随之而来的是上层丰富的应用场景对底层硬件算力的需求呈爆炸式增长. 以人工智能应用为例,据估算当前人工智能算力需求每隔 3~4 个月翻一倍,未来十年人工智能应用的算力需求将增长 500 倍以上. 根据 Amdahl 定律,软件性能加速比由程序的可并行部分决定,而当前通用处理器一方面设计越发复杂且由于工艺进步放缓越来越难提升核心的性能频率,另一方面通用处理器仍对部分软件应用中的典型算子不能很好加速,难以满足存储、计算以及功耗等方面的需求,所以软件应用与硬件性能之间出现了巨大的算力鸿沟.

为了解决以上问题,Hennessy 与 Patterson[1] 提出通过软硬件协同设计的领域特定架构(domain specific architecture,DSA)作为应对方案. DSA 是一种更加倾向以硬件为中心的方法,是针对特定领域设计定制的体系结构,并为该领域提供显著的性能(和能效)增益. DSA 和通用处理器相比更贴近应用程序的需求,它在加速特定应用程序时可以实现更好的性能;与此同时 DSA 具有一定的可编程性,所以它和专用集成电路(application specific integrated circuit,ASIC)相比具有更好的灵活性和功能覆盖范围. 而协处理器作为分担通用处理器特定工作负载的主要部件,承担着专用算力支持,是 DSA 设计的重要实现途径,也是当前硬件加速结构的研究热点. 与此同时领域专用协处理器需要针对领域应用的功能需求设计体系结构,并定制相关应用专用硬件加速架构和配套软件栈,这会带来较长的设计周期和较高的资本开销. 所以研究协处理器设计方法从而缩短设计周期、降低设计成本开销、提高设计效率,已成为当前硬件架构设计的重要研究方向.

本文的主要贡献包括 4 个方面:

(1)探索了基于不同耦合度和不同应用需求的人工智能协处理器设计空间;

(2)研究了基于 RISC-V 定制指令的弱耦合协处理器指令架构,包括控制指令和访存指令;

(3)研究了不同应用场景下弱耦合协处理器控制接口、访存接口和设计框架的设计方法;

(4)提出了两种面向不同人工智能应用场景的弱耦合协处理器设计实例.

1 协处理器设计空间

本节主要介绍基于不同耦合度协处理器的设计空间及相关工作.

1.1 协处理器

协处理器负责协助通用处理器做一些主处理器无法执行或者执行效率不佳的任务,比如浮点、信号、音频处理等. 协处理器一般通过扩展指令集或提供配置寄存器来扩展通用处理器的处理功能,硬件架构设计人员既可以在标准指令集中添加专门的指令来扩展指令集,也可以通过专用指令来控制协处理器,还可以设计一套定制的协处理器专用指令来搭建异构系统. 所以从指令架构和硬件设计目标看,协处理器属于广义的 DSA 范畴. 图 1 给出了基于耦合度的协处理器设计类别,耦合度由高到低协处理器可以分为完全耦合、紧耦合、弱耦合、松耦合和解耦合 5 种类型,表 1 列出了 5 种设计类型的主要特点对比.

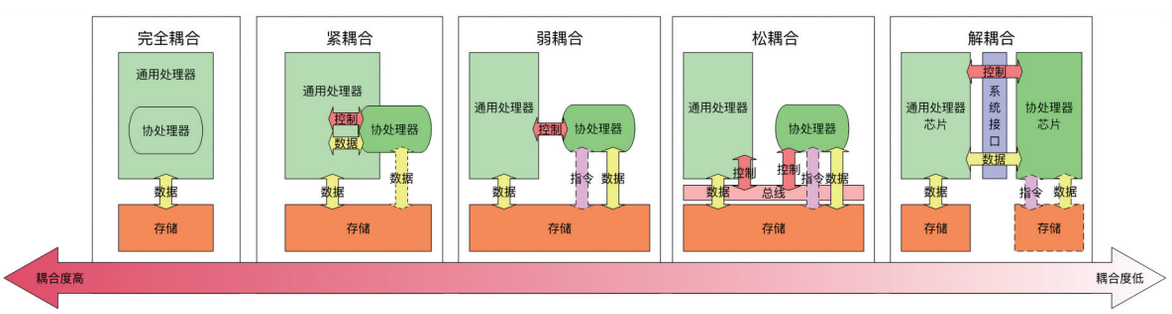

图 1 基于耦合度的协处理器设计分类示意图

Fig. 1 Illustration of coprocessors design classification based on coupling degree

1.2 完全耦合协处理器

完全耦合协处理器可以看做是通用处理器的一部分,负责执行特定负载,例如 ARM 架构的 SVE 和 SME 扩展、X86 架构的 AVX 扩展是针对浮点和并行计算任务的协处理器,可以用于深度学习中矩阵乘、卷积等算子的加速. 例如 AMD Zen2 通用处理器[2] 中的浮点/向量执行协处理器支持 X86 指令架构的 AVX-256

扩展指令集,协处理器与通用处理器中的基本核心(整数)共享前端逻辑(取指、分支预测、译码)和访存通路(LD/ST 逻辑和数据缓存),拥有自己独立的数据通路(调度、发射、寄存器文件). 此外通常在完全协处理器空闲时,通用处理器会关闭协处理器的时钟以减少功耗. 但由于完全耦合协处理器的耦合特性,一般作为通用处理器的通用算力补充,并不适合作为领域应用的任务卸载.

表 1　基于耦合度的协处理器设计特点对比

Table 1　Features comparison of the coprocessor based on coupling degree

类型\特点	时钟频率(基于通用处理器频率)	指令架构(基于标准指令集)	控制接口	执行粒度	负载粒度	访存端口
完全耦合	同频	标准扩展指令集	处理器前端	指令	小	共享
紧耦合	同频/倍频	标准扩展指令集	处理器前端	指令/函数	小/中	共享/独立
弱耦合	同频/倍频/异频	标准扩展指令集/定制指令集	处理器前端	指令/函数	中/大	独立
松耦合	异频	独立指令集/定制命令集	片上网络	任务	大	独立
解耦合	异频	独立指令集	系统接口	任务	大	独立

1.3　松/解耦合协处理器

松耦合协处理器通常是基于片上系统(System on Chip,SoC)集成的协处理器,帮助通用处理核心处理领域工作负载,松协处理器只支持有限的固定操作,当遇到不支持的操作时需要退回到通用处理核心进行计算. 例如 ARM Ethos-U55 协处理器[3]支持卷积神经网络和递归神经网络(recurrent neural network,RNN)的加速,它和通用处理核心共用片上 SRAM 和 FLASH 空间. 通用处理核心通过 APB 总线接口设置 Ethos-U55 协处理器中央控制模块;协处理器通过自己的 DMA 引擎和 AXI 总线去对应 FLASH 地址空间读取离线编译的定制命令流和数据进行计算;当协处理器将结果写回到 SRAM 完成计算任务后,发起中断通知通用处理核心.

而解耦合协处理器通常是基于系统接口(如 PCIe 接口)实现与通用处理器的连接,从而构建异构系统. 解耦合协处理器通常也被认为是领域计算专用芯片,它们一般拥有自己独立且完备的指令架构和控制、计算、访存逻辑. 面向人工智能的张量处理器(tensor processing unit,TPU)[4]开启了领域专用协处理器设计浪潮,贾迅等[5]和 Li 等[6]分别研究了双精度浮点矩阵乘协处理器和矩阵-矩阵乘协处理器,通过 PCIe 接口和通用处理器相连,协助通用处理器提高矩阵运算能力.

1.4　紧/弱耦合协处理器

紧耦合协处理器和弱耦合协处理器的共性在于它们接收通用处理核心的控制指令,区别在于紧耦合协处理器仍处于通用处理核心的数据通路中,它根据控制指令完成对应计算并和通用处理核心交互数据;而弱耦合协处理器既可能使用控制指令作为计算指令,也可能拥有独立的取指令通路,并且弱耦合协处理器拥有自己的访存通路. 本文将紧耦合协处理器作为弱耦合协处理器的一种特殊案例进行统一的设计方法研究.

紧耦合协处理器的优点是对通用处理核心现有体系结构的其他部分没有影响,对微体系结构的影响最小,同时可以继承原有的编程模式,并具有一定的通用性. 例如苹果 M1 芯片的 AMX 矩阵协处理器和 IBM POWER10 的 MMA 矩阵协处理器为人工智能,特别是深度学习应用等工作负载提供高效的加速支持. 例如 POWER10 处理器[7]中的 MMA 协处理器包含一个 4×4 的矩阵外积计算单元和一组本地累加器(8 个 512 位宽寄存器),支持 FP64、FP32 和 BF16 等数据精度,与此同时在 POWER-ISA 3.1[8]中加入 VSX MMA 指令来控制 MMA 协处理器工作,包括 VSX 移动指令、VSX 整数外积指令、VSX 浮点外积指令等.

弱耦合协处理器的优点是在不影响通用处理核心的前端和主流水线设计的前提下,协处理器的控制、计算和访存更加灵活,更能满足领域应用高效计算的实际需求. DOJO[9]是特斯拉面向人工智能训练的高通量通用处理器的微架构,DOJO 处理节点中通用处理核心的指令宽度为 64b,而向量/矩阵协处理器的指令宽度为 64B,协处理器指令通过标量调度模块接口发送给协处理器的向量调度模块进行处理;通用处理核心的访存通路为 8Bx2 宽度,而向量/矩阵协处理器的访存通路为 64Bx3 宽度,通用处理核心和协处理器使用 SRAM 空间而不是数据缓存进行数据交互. 所以在 DOJO 中通用处理核心只负责通用的地址计算和逻辑计算,而工作负载高的计算任务则都由协处理器完成. Matteo 等[10]和 Schmidt 等[11]基于 RISC-V 通用处理核心实现了弱耦合的向量协处理器. Matteo 等通过通用核心中的加速器调度分派模块负责与协处理器进行交互:当通用处理核心在译码阶段判断指令为协处理器指令后,将指令发射到加速器调度分派模块

的指令队列中,该模块决定何时可以发送指令到协处理器,从而解耦协处理器指令与通用核心执行通路;而 Schmidt C 等实现的向量协处理器采用了另一种弱耦合协处理器的设计方法,通用处理核心通过一条特殊的定制指令 VF(vector fetch)来通知协处理器开始执行取指,VF 指令的参数就是协处理器取指的起始 PC 和代码块长度;当协处理器单次任务执行完成后,根据 VSTOP(vector stop)指令来表示执行暂停,等待下一轮 VF 指令的输入.

由于弱耦合协处理器与通用处理核心的控制、存储交互延迟相比于松耦合/完全耦合协处理器要短,且相比于完全耦合协处理器具有更灵活的设计空间,更适合特定领域应用场景,所以本文主要针对弱耦合协处理器的设计方法进行研究.

2 弱耦合协处理器设计

本节讨论弱耦合协处理器的控制架构设计方法,主要包括指令架构、控制和访存接口与整体硬件架构设计方法三个方面. 本文选取 RISC-V 指令架构中的协处理器定制指令格式作为实现弱耦合协处理器设计方法的实施示例,可支持拓展到任意 RISC 指令架构.

2.1 指令架构

指令是通用处理核心与协处理器之间命令和数据的交互逻辑接口,也是弱耦合协处理器设计的关键因素之一.

图 2 列出了 RISC-V 定制指令格式[12],其中功能码 opcode 位域中有 4 种固定指令编码格式可用于实现协处理器定制指令;rs2、rs1 和 rd 位域表示对应寄存器索引,根据 xs2、xs1 和 xsd 位域的值确定寄存器属于通用处理核心还是协处理器,当对应位域值为 1 时寄存器属于通用处理核心,当值为 0 时寄存器属于协处理器;同时根据通用处理核心中寄存器宽度的不同,可以选择 32 位、64 位的标量寄存器甚至 128 位/256 位的向量寄存器.

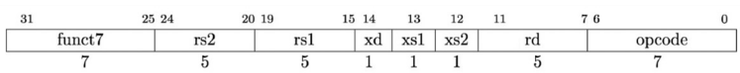

图 2 RISC-V 协处理器定制指令格式编码

Fig. 2 Illustration of RISC-V coprocessor custom instruction format

2.2 控制指令与接口

根据不同领域应用对象的控制场景,对弱耦合协处理器的控制指令设计方法进行研究,如表 2 所示.

表 2 协处理器控制指令

Table 2 Command instructions for coprocessor

指令名称	指令格式	来源编码域[xd xs1 xs2]	操作
寄存器读指令			
主-协寄存器直接读	cprd rd cps	100	Core[rd]← Copro[rs1]
主协-寄存器间接读	cprd rd rs1	110	Core[rd]← Copro[(rs1)]
寄存器写指令			
主-协寄存器直接写	cpwr cpd rs1	010	Core[rs1]→ Copro[rd]
协-主寄存器间接写	cpwr rs2 rs1	011	Copro[(rs1)]→ Core[rs2]
固定操作指令			
协独立操作	cpsop rd cps1 cps2	100	Core[rd]← Copro[rs1]op Copro[rs2]
主-协协同操作	cpcop rd rs1 cps2	110	Core[rd]← Core[rs1]op Copro[rs2]
主独立操作	cpcop rd rs1 rs2	111	Core[rd]← Core[rs1]op Core[rs2]
自定义操作指令			
自定义操作	cpcmd rs1,rs2	011	Copro[cmd]← Core[rs1+rs2]
控制状态寄存器(CSR)指令			
主-协 CSR 读	cprcsr rd cpcsr	100	Core[rd]← Copro CSR[rs1]
主-协 CSR 写	cpwcsr cpcsr rs1	010	Core[rs1]→ Copro CSR[rd]

表 2 中列出控制指令执行协处理器的寄存器读、寄存器写、固定操作、自定义操作、CSR 读写等功能,用于通用处理核心和协处理器的交互. 固定操作指令支持例如卷积/矩阵乘/乘累加等单次固定操作,自定义操作指令可支持超越函数查表、批量卷积/矩阵乘、信号处理等复杂融合操作.

根据以上控制指令,通用处理核心和弱耦合协处理器的控制接口设计主要由三类接口组成:寄存器请求/响应接口、CSR 读/写接口、自定义功能接口,如图 3 所示. 设计方法可根据对象的任务粒度、负载粒度等需求进行接口设计的排列组合,同时这三种接口可以复用硬件连线,通过控制信息区分,从而减少接口宽度.

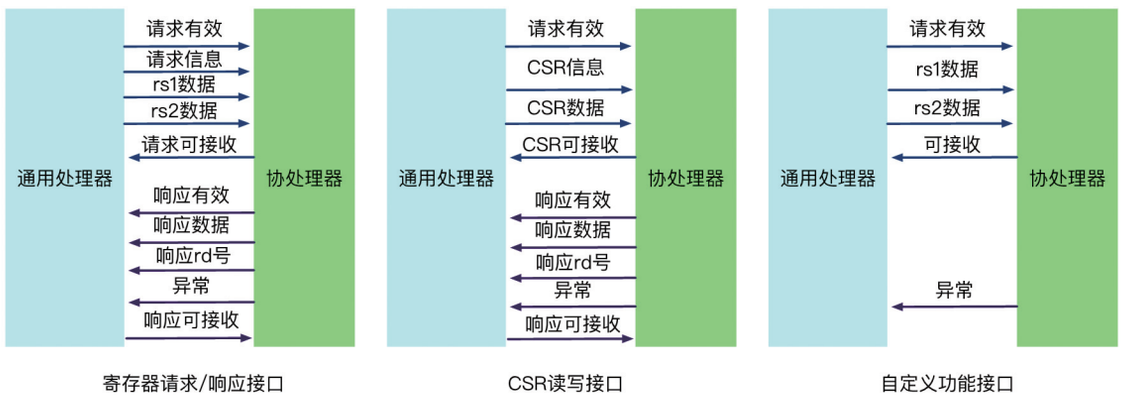

图 3　控制接口示意图

Fig. 3　Illustration of command interface

表 3 中列出了本文指令接口、定制协处理器接口[12](rocket custom coprocessor interface,RoCC)和蜂鸟协处理指令扩展接口[13](nuclei instruction Co-unit extension,NICE)的控制接口性能对比,本文提出的控制接口具有更好的灵活性和可扩展性.

表 3　控制接口性能对比

Table 3　The performance contrast of control interface

	本文	RoCC	NICE
寄存器读写	支持	支持	支持
固定功能	支持	支持	支持
64 位模式	支持	支持	不支持
CSR 读写	支持	不支持	不支持
自定义扩展功能	支持	不支持	不支持
控制接口宽度	67~229	133/229	133

2.3　访存指令与接口

根据不同领域应用对象的访存场景,对弱耦合协处理器的访存指令设计方法进行研究,如表 4 所示.

表 4　协处理器访存指令

Table 4　Memory instructions for coprocessor

指令名称	指令格式	来源编码域[xd xs1 xs2]	操作
寄存器访存指令			
寄存器直接加载	cprld cpd[rs1]	010	Copro[rd]← Mem[(rs1)]
寄存器间接加载	cprld rs2[rs1]	011	Copro[(rs2)]← Mem[(rs1)]
寄存器直接存储	cprst cps2[rs1]	010	Copro[cps2]→ Mem[(rs1)]
寄存器间接存储	cprst rs2[rs1]	011	Copro[(rs2)]→ Mem[(rs1)]
缓冲访存指令			
缓冲加载	cpbld rs2[rs1]	011	Copro_Buf[(rs2)]← Mem[(rs1)]
缓冲存储	cpbst rs2[rs1]	011	Copro_Buf[(rs2)]→ Mem[(rs1)]
批量访存指令			
批量加载/存储	cpdma rs1,rs2	—	Copro_DMA(rs1+rs2) Copro_Buf[cmd]⟷ Mem[cmd]

表 4 中的访存指令执行协处理器的加载、存储和批量直接读写等功能,用于协处理器和存储空间的交互. 然后根据以上访存指令,对弱耦合协处理器的访存接口设计方法进行讨论. 弱耦合协处理器的访存接口主要由访存请求接口和访存响应接口两种类型组成;根据目标存储空间是否支持缓存一致性,接口类型可分为可缓存空间访存接口或不可缓存空间访存接口,如图 4 所示.

表 5 中列出了本文、RoCC 和 NICE 的访存接口性能对比,本文提出的访存接口具有更好的可扩展性.

此外,如果弱耦合协处理器和通用处理核心共享便签存储空间,那么程序员可以通过对便签存储空间中固定地址的写读来实现通用处理核心与协处理器的交互:

(1)通用处理核心将非约定值存储到便签存储空间特定地址空间执行初始清操作;

(2)协处理器完成处理任务后写便签存储空间特定地址约定值;

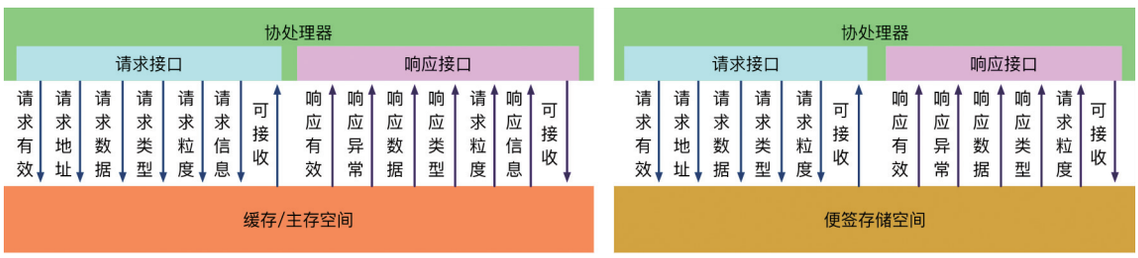

图 4　访存接口示意图

Fig. 4　Illustration of memory access interface

(3)通用处理器核心加载便签存储空间特定地址数据,判断是否为约定值.

通用处理核心与协处理器通过存储空间进行交互相比于 2.2 中寄存器交互或 CSR 交互的优点是支持异步解耦执行并减少协处理器对通用处理核心架构的影响,缺点是交互延迟相对较长且容易增加分支跳转开销.

表 5　访存接口性能对比

Table 5　The performance contrast of memory interface

	本文	RoCC	NICE
寄存器-主存/可缓存/不可缓存读写	支持	支持	支持
缓冲-主存/可缓存/不可缓存读写	支持	支持	不支持
大粒度访存	支持	支持	不支持
批量访存	支持	不支持	不支持
访存接口宽度	165~245	161/241	105

2.4　整体架构

不同的算法应用可能在端、边缘、云等场景都有广泛的部署,不同场景应用在计算\访存负载粒度上存在较大的差别,因此需要组合不同指令和接口来进行硬件架构设计. 本文针对不同应用场景的需求提出了 4 种弱耦合协处理器架构,分别为细粒度小负载(FGSL)、细粒度中负载(FGML)、粗粒度中负载(CGML)和粗粒度大负载(CGBL)架构,分别使用第 2 章中所讨论的指令、控制和访存接口进行组合设计.

表 6 和表 7 分别列出 4 种协处理器架构设计(64 位架构)的性能和开销对比,其中 CGBL-C/E 为分别采用 CSR 接口和自定义接口的粗粒度大负载场景协处理器.

表 6　弱耦合协处理器设计对比

Table 6　The performance contrast of weakly-coupled coprocessor design

	FGSL	FGML	CGML	CGBL-C/E
指令发送延迟	1	1	>3	>3(-C) >访存延迟(-E)
结果返回延迟	1	1	3	3
流水线冲突	存在	存在	不存在	不存在
寄存器读写延迟	3	3	—	—
访存延迟	寄存器拷贝	访存通路传输+主存/缓存读写+本地读写	指令解析+访存通路传输+主存/缓存读写+本地读写	指令解析+DMA 拆包+访存通路传输+便签存储器读写+本地读写
访存带宽	—	最小值(指令发送带宽,访存通路带宽,主存/缓存读写带宽,寄存器允许飞行带宽)	最小值(指令发送带宽,访存通路带宽,主存/缓存读写带宽,本地存储读写带宽)	最小值(指令发送带宽,访存通路带宽,便签存储器读写带宽,本地存储读写带宽)
计算性能	最小值(寄存器供数能力,计算部件数量)	最小值(访存带宽,计算部件数量)	最小值(访存带宽,计算部件数量)	最小值(访存带宽,计算部件数量)

表 7　弱耦合协处理器设计开销对比

Table 7　The overhead contrast of weakly-coupled coprocessor design

	FGSL	FGML	CGML	CGBL-C/E
控制接口宽度	229	229	204	204(-C) 131(-E)
访存接口宽度	—	241	243	245
资源开销	寄存器+计算单元	寄存器+加载/存储管理单元+计算单元	指令队列+本地缓冲+缓冲管理单元+计算管理单元+计算单元	指令队列+批量访存引擎+本地缓冲+缓冲管理单元+计算管理单元+计算单元
功耗开销	主存-寄存器数据拷贝+寄存器-寄存器数据拷贝+计算+寄存器写回	主存-寄存器数据拷贝+计算+寄存器写回	指令存储解析+主存-本地缓冲数据拷贝+计算+CSR 写回	指令存储解析+DMA 拆包+便签-本地缓冲数据拷贝+计算+CSR 写回/便签写回

从表中可以得出:

(1)FGSL 架构资源开销少,但计算性能和功耗受寄存器供数影响大,此外由于寄存器容量有限,若计算操作需要频繁和通用处理核心寄存器交互,不但会受通用处理核心流水线结构冲突的影响还会增加数据移动延迟和功耗,适合例如单次 SIMD、卷积、矩阵乘或乘累加等计算操作.

(2)FGML 架构的计算性能和功耗受访存带宽影响较大,即受到协处理器中寄存器允许飞行的访存数量限制,同时也会受通用处理核心流水线结构冲突的影响,适合如单次张量卷积或小规模矩阵乘等计算操作.

(3)CGML 架构增加了指令队列开销以及本地缓冲和管理单元开销,但是访存带宽受限少,可以设计合适的计算单元数量来支持高效的计算访存比,适合批量计算任务,例如深度神经网络中一个卷积层的卷积计算或一次傅里叶变换计算.

(4)CGBL 架构拥有最少的控制接口线宽度,但它相比于 CGML 架构增加了批量直接访存(DMA)单元开销,同时采用便签存储器可以提高访存带宽,降低与通用核心的批量数据交互延迟,从而设计计算峰值性能高的协处理器(达到 T 级 ops),同时 CGBL 架构具有最优的灵活性和可扩展性.

综上所述,在弱耦合协处理器设计过程中需要根据硬件部署场景和应用需求来确定选择整体架构设计方法中的一种或多种,既可以对通用处理核心扩展单一类型协处理器接口与计算架构来增强特定领域计算性能,也可以对多通用处理核心扩展多领域计算部件来满足多种算力性能需求,但需要根据硬件设计复杂度与具体算力需求来确定使用单一类型或多类型的协处理器接口.

3 人工智能协处理器设计

3.1 需求分析

人工智能是当前发展最迅速的应用之一,尤其深度学习已经在计算机视觉、安防、智慧城市等行业中产生了巨大的影响和效果. 深度学习源于机器学习中的人工神经网络,它的实质是使用大量数据来构建具有统计特征的多层神经网络模型,与传统机器学习算法相比对硬件计算和访存性能需求更高,并且更适合硬件进行并行加速. 其中多层感知机 MLP、卷积神经网络 CNN 和 Transformer[14] 网络是当前使用最广泛的深度学习模型:

(1)多层感知机 MLP 主要由全连接层(fully connected layers,FC)和激活层(activation layers)组成,全连接层是网络权重最多的层,它的计算公式为:

$$O[n][k]=\sum_{c=0}^{C-1}\sum_{r=0}^{R-1}\sum_{s=0}^{S-1}\boldsymbol{F}[k][c][r][s]\times\boldsymbol{I}[n][c][r][s], \tag{1}$$

式中,$\boldsymbol{F}$ 为全连接层的权重(四维张量),$\boldsymbol{I}$ 为全连接层的图像(四维张量),N 为图像个数,C 为权重通道数量,H 为输入特征图高,W 为输入特征图宽,K 为输出通道,R 为权重高($R=H$),S 为权重宽($S=W$),当 $N=1$ 时,FC 层是一个 3 层循环的矩阵-向量乘计算,当 $N>1$ 时,FC 层是一个 3 层循环的矩阵-矩阵乘计算. 激活层由激活函数组成,常用的激活函数有 ReLU、PReLU、Softmax 和 tanh 等.

(2)卷积神经网络 CNN 主要由卷积层(convolution layers)、激活层、归一化层(batch normalization layers,BN)、池化层(pooling layers)和全连接层组成,其中卷积层是最重要的层,也是实现特征提取、预测等功能的来源,它的计算公式为:

$$O[n][m][x][y]+=\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\sum_{x=0}^{F-1}\sum_{y=0}^{E-1}\sum_{i=0}^{R-1}\sum_{j=0}^{S-1}\sum_{k=0}^{C-1}\boldsymbol{F}[m][k][i][j]\times\boldsymbol{I}[n][k][Ux+i][Uy+j], \tag{2}$$

式中,M 为输出图通道数量,卷积层是一个 7 层循环的矩阵乘累加计算. 卷积层和全连接层类似,但卷积核的权值数量要少于全连接层.

(3)Transformer 本质上也是一个编码器-解码器的结构,在 Transformer 的编码器中,数据经过自注意力模块(self-attention)得到一个加权之后的特征向量 $\boldsymbol{Z}$,自注意力模块的计算公式为:

$$\boldsymbol{Z}=\mathrm{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{3}$$

式中,$\boldsymbol{Q}$ 为查询(Query)矩阵,$\boldsymbol{K}$ 为键值(Key)矩阵,$\boldsymbol{V}$ 为值(Value)矩阵,d_k 是 $\boldsymbol{K}$,$\boldsymbol{Q}$ 矩阵的维度,需要进行

两次矩阵-矩阵乘计算;然后多头注意力(Multi-Head Attention)模块将 H 个自注意力模块的输出 $\boldsymbol{Z}$ 进行拼接处理和线性变换得到最终的 $\boldsymbol{Z}$ 矩阵;特征矩阵 $\boldsymbol{Z}$ 经过归一化残差(Add & Norm)层后再被送入前馈神经网络 FFN(Feed Forward Neural Network),FFN 是一个具有两层 FC 层的网络,计算公式为:

$$\mathrm{FFN}(\boldsymbol{Z})=\max(0,\boldsymbol{Z}\boldsymbol{W}_1+b_1)\boldsymbol{W}_2+b_2, \tag{4}$$

式中,$\boldsymbol{W}$ 为权值,需要进行两次矩阵-向量乘计算. 解码器和编码器结构类似,解码器输入真实值进行训练,它包含了两个多头注意力模块和一个 FFN,最终采用 Softmax 函数输出结果.

综上所述人工智能应用主要的算子需求有(1)卷积/反卷积;(2)矩阵-矩阵乘;(3)矩阵-向量乘;(4)池化/反池化;(5)逐点乘/加操作;(6)激活函数;(7)归一化. 部署在不同场景的不同人工智能应用,需要实现协处理器对不同的算子进行计算加速.

3.2 硬件平台

当前已经出现了许多面向人工智能应用的硬件加速平台,表 8 中列出了目前已公开的商用神经网络协处理器:

表 8 人工智能应用的硬件加速

Table 8 The hardware acceleration for AI

平台	加速结构	部署场景	工作任务	耦合类型
NVIDIA H100[15]	张量计算单元	数据中心	训练/推理	解耦合
TPUv4[16]	脉动阵列	数据中心	训练/推理	解耦合
TPUv4i[17]	脉动阵列	边缘	推理	解耦合
寒武纪 MLU370[18]	点积树	数据中心/边缘	训练/推理	解耦合
百度昆仑芯 2 代[19]	乘累加阵列	数据中心/边缘	训练/推理	解耦合
ARM Ethos-U55[3]	乘累加单元	边缘/端	推理	松耦合
嘉楠勘智 K510[20]	向量运算单元	边缘/端	推理	弱耦合
Tesla DOJO[7]	矩阵计算单元	数据中心	训练/推理	弱耦合
TachyumT16128[21]	向量计算单元	数据中心/边缘	训练/推理	紧耦合

从表中可以发现,面向训练的人工智能协处理器以解耦合协处理器为主,例如谷歌公司的 TPU 系列或 NVIDIA 公司的 GPU 100 系列,而面向推理的人工智能协处理器以松耦合协处理器为主;但同时可以发现基于 ARM 或 RISC-V 指令集的硬件平台也往往使用弱耦合或紧耦合协处理器来实现人工智能应用的硬件加速,这是由于 ARM 和 RISC-V 指令集都提供了相应的扩展加速指令. 所以本文第三章所讨论的弱耦合协处理器设计方法,可以较好地用于面向人工智能应用的协处理器设计.

4 设计实例

为了验证第三章中所提出的协处理器架构设计方法,本文针对第四章讨论的人工智能应用需求实现了两种弱耦合处理器,分别面向不同人工智能应用场景. 设计验证环境采用 16 nm 工艺库进行综合仿真.

4.1 点积协处理器

由于卷积或矩阵乘可以通过点积计算来实现,所以本文设计实现点积协处理器用于加速卷积/矩阵乘操作. 点积协处理器如图 6(a)所示,它面向端侧人工智能场景,采用图 5(a)的 FGSL 架构设计,适用于智能微处理器(MCU)或物联网(IoT)芯片的小粒度扩展. 由于 MCU 或 IoT 芯片通常采用 32 位寄存器结构,所以点积计算协处理器主要由触发器和四点积计算单元组成,可以在每个时钟周期内完成 8 位四点积运算(a0b0+a1b1+a2b2+a3b3),这是考虑到 32 位寄存器恰好可以保存 4 个 Int8 类型的操作数. 点积协处理器仅支持 cpdp 计算指令,由于点积计算协处理器没有独立访存接口,所以不支持访存指令. 点积协处理器完成一次四点积计算的指令执行:

- cpdp rd rs1 rs2

其中 cpdp 为协处理器点积计算指令,rs1 和 rs2 为通用处理核心源寄存器,rd 为通用处理核心目标寄存器,只需要一个时钟周期即可完成四点积计算和结果送回. 在 1.25 GHz 时钟频率约束的 16 nm 综合结果中,点积计算协处理器的逻辑单元数量为 750,峰值功耗小于 1 mW.

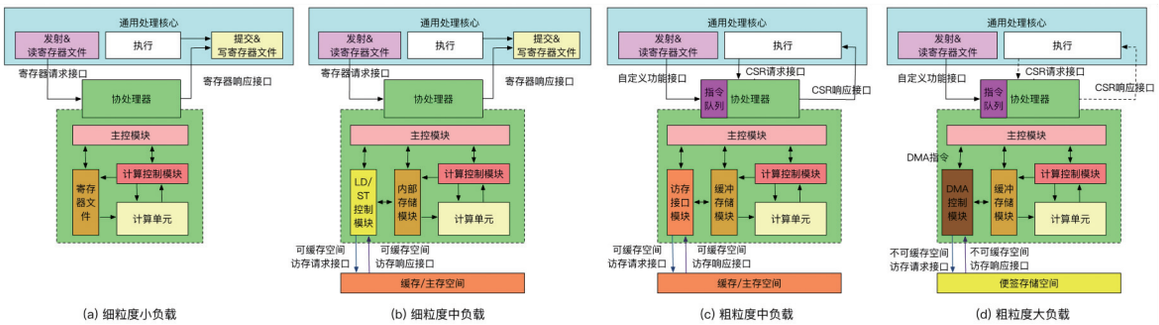

图 5　协处理器接口和架构示意图

Fig. 5　Illustration of the interface and architecture of coprocessor

4.2　推理协处理器

由于当前人工智能推理任务的核心操作由包括卷积、归一化、池化、逐点操作等,各层的计算负载与访存粒度较大且访存性能会直接影响推理任务的效率. 由于推理任务的对象常见为彩色高清图像或视频,所以实时推理任务的计算性能需要和数据量相匹配. 设备端或边缘端推理任务的峰值算力需求一般在 1~20 Tops 范围内,而云端推理任务则没有需求上限. 所以本文设计实现高性能推理协处理器用于加速云/边缘端的人工智能推理任务. 推理协处理器如图 6(b)所示,它采用图 5(d)的 CGBL-E 架构设计,适用于高性能芯片的大负载任务扩展. 推理协处理器主要由指令队列和调度模块、控制模块、批量访存模块、数据传输单元、乘累加计算阵列、其他计算单元和 Mem 访问单元组成. 推理协处理器采用自定义功能接口,通过灵活的自定义功能指令来支持卷积、池化、逐点加等推理应用所需算子和协处理器内部的数据传输与精度转换功能:

- cpcmd.conv rs1,rs2
- cpcmd.pool rs1,rs2
- cpcmd.elmadd rs1,rs2

其中 rs1 和 rs2 为通用处理核心源寄存器,支持在线/离线两种模式生成协处理器指令;此外推理协处理器通过自定义功能接口发送批量访存指令 cpdma rs1,rs2,经由批量访存模块完成协处理器与通用处理核心的数据共享传输和任务完成交互(根据地址区分便签存储空间或主存空间). 如果推理模型存在协处理器不支持的算子,需要经过数据传输由通用核心进行处理. 乘累加计算阵列提供 4Tops@ Int8 的推理算力,协处理器内 MEM 容量设计为 512 KB 以支持双缓冲执行从而增加计算效率. 在 1.25 GHz 时钟频率约束的 16 nm 综合结果中,推理协处理器的逻辑单元数量为 13 万,峰值功耗约为 2.6 W.

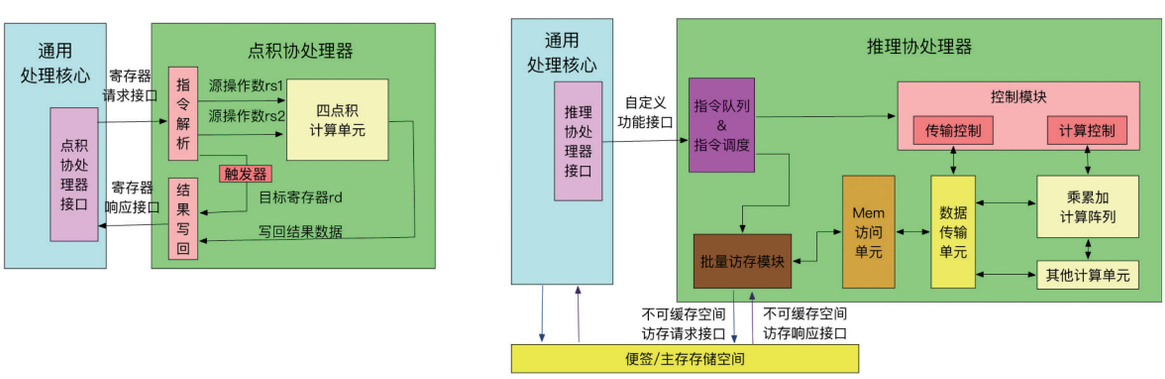

图 6　两种人工智能协处理器设计实例架构示意图

Fig. 6　Illustration of the architecture of two AI-coprocessor design cases

5　结论

本文通过归纳分析不同耦合度和不同应用需求的领域计算协处理器的相关工作,深入研究面向领域应用的弱耦合协处理器,基于 RISC-V 协处理器定制指令探索了弱耦合协处理器指令架构和控制接口、访存指令和访存接口的设计方法,研究提出了面向领域应用的弱耦合协处理器的整体框架设计方法,对比了

基于4种应用场景的弱耦合协处理器架构性能和开销;分析了人工智能应用的共性算法与人工智能协处理器的研究现状;最后给出了两种面向不同人工智能应用场景的弱耦合协处理器设计实例,为提升领域专用协处理器设计效率提供有效支撑.

[参考文献]

[1] HENNESSY J L,PATTERSON D A. A new golden age for computer architecture[J]. Communications of the ACM,2019,62(2):48-60.

[2] SUGGS D,SUBRAMONY M,DAN B. The AMD "Zen 2" processor[J]. IEEE micro,2020,40(2):45-52.

[3] SKILLMAN A,EDSO T. A technical overview of Cortex-M55 and Ethos-U55:Arm's most capable processors for endpoint AI[C]//2020 IEEE Hot Chips 32 Symposium(HCS). Palo Alto:IEEE,2020:1-20.

[4] JOUPPI N P,YOUNG C,PATIL N,et al. In-datacenter performance analysis of a tensor processing unit[J]. Computer architecture news,2017,45(2):1-12.

[5] 贾迅,邬贵明,谢向辉,等. 双精度浮点矩阵乘协处理器研究[J]. 计算机研究与发展,2019,56(2):410-420.

[6] LI Y,PEDRAM A. CATERPILLAR:Coarse grain reconfigurable architecture for accelerating the training of deep neural networks[C]//2017 IEEE 28th International Conference on Application-specific Systems,Architectures and Processors(ASAP). Seattle:IEEE,2017:1-10.

[7] STARKE W J,THOMPTO B W,STUECHELI J,et al. IBM's power10 processor[J]. IEEE micro,2021,41(2):7-14.

[8] IBM. Power instruction set architecture,version 3.1b[EB/OL]. [2023-05-28] https://openpowerfoundation.org/specifications/isa/.

[9] EMIL T,DOUGLAS W,SARMA D,et al. DOJO:The microarchitecture of Tesla's exa-scale computer[C]//2022 IEEE Hot Chips 34 Symposium(HCS).Cupertino:IEEE,2022:1-28.

[10] MATTEO P,MATHEUS C,WISTOFF N,et al. A"New Ara"for vector computing:An open source highly efficient RISC-V V 1.0 vector processor design[C]//2022 IEEE 33rd International Conference on Application-specific Systems,Architectures and Processors(ASAP). Gothenburg:IEEE,2022:43-51..

[11] SCHMIDT C,WRIGHT J,WANG Z,et al. 4.3 An eight-core 1.44 GHz RISC-V vector machine in 16 nm FinFET[C]//2021 IEEE International Solid-State Circuits Conference(ISSCC). SanFrancisco:IEEE,2021:58-60.

[12] ASANOVIC K,AVIZIENIS R,BACHRACH J,et al. The rocket chip generator,UCB/EECS-2016-17[R]. Berkeley,CA:EECS Department,University of California,2016.

[13] NUCLEI SYSTEM TECHNOLOGY. Hummingbirdv2 E203 Core and SoC v0.2.1[EB/OL]. [2023-05-28] https://doc.nucleisys.com/hbirdv2/core/core.html#nice.

[14] VASWANI A,SHAZEER N,PARMAR N,et al. Attention is all you need[J]. Neural information processing system,2011:5998-6008.

[15] NVIDIA. NVIDIA H100 tensor core GPU architecture[EB/OL]. [2023-05-28] https://nvdam.widen.net/s/9bz6dw7dqr/gtc22-whitepaper-hopper,2022.

[16] NORMAN P J,GEORGE K,SHEN G L,et al. TPU v4:An optically reconfigurable supercomputer for machine learning with hardware support for embeddings[J]. arXiv:2304.01433,2023.

[17] JOUPPI N P,YOON D H,ASHCRAFT M,et al. Ten lessons from three generations shaped Google's TPUv4i:Industrial product[C]//2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture(ISCA). Valencia:ACM,2021:1-14.

[18] CAMBRICON. Cambricon MLU370 chip[EB/OL]. [2023-05-28] https://www.cambricon.com/index.php? m=content&c=index&a=lists&catid=360,2023.

[19] OUYANG J,DU X,MA Y,et al. 3.3 Kunlun:A 14 nm high-performance AI processor for diversified workloads[C]//2021 IEEE International Solid-State Circuits Conference(ISSCC). SanFrancisco:IEEE,2021:50-51.

[20] AUFRANC J L. Kendryte K510 tri-core RISC-V AI processor deliver up to 3 TOPS[EB/OL]. [2023-05-28] https://www.cnx-software.com/2021/07/09/kendryte-k510-tri-core-risc-v-ai-processor-3-tops/,2021.

[21] SHILOV A. Tachyum teases 128-Core CPU:5.7 GHz,950W,16 DDR5 channels[EB/OL]. [2023-05-28] https://www.tomshardware.com/news/tachyum-teases-128-corecpu-57-ghz-950w-16-ddr5-channels,2022.

[责任编辑:黄　敏]