A Unified AI Accelerator Interface for Scalable RISC-V Architectures

Fucong Qiu[1,2], Mingjin Gao[1,2*], Dan Tang[1,3], Yungang Bao[1,2] and Tao Xie[3,4*]

[1]State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Beijing Institute of Open-Source Chip
[4]School of Computer Science, Peking University

Abstract

Various general-purpose accelerator interfaces proposed by the industry struggle to meet AI workloads' demands. Thus specialized interface solutions have emerged to address AI-specific challenges but suffer from limited adaptability and expandability. To address the issue, in this paper, we propose a flexible, open AI accelerator interface for the RISC-V architecture, natively supporting AI-oriented extensions—including vector, matrix, and tensor operations. The interface integrates diverse data access mechanisms—L1 cache access, bus access, coprocessor extension access, and inter-coprocessor data movement, to accommodate various AI dataflow scenarios. In addition, the interface provides a robust virtual memory mechanism that enables accelerators to obtain physical addresses from the processor via private TLBs or direct MMU access for efficient address translation. By supporting multiple decoupled microarchitectural designs through flexible accelerator decoding schemes and various accelerator CSR management approaches, the interface enhances system scalability, versatility, and performance. Together, these features form a scalable and robust solution for future AI accelerator applications based on RISC-V architectures.

Introduction

As an open and scalable instruction set architecture, RISC-V has rapidly gained widespread acceptance across computing tasks. To enhance processor versatility, the industry has introduced various general-purpose accelerator interfaces. For instance, the Rocket Custom Coprocessor Interface (ROCC) [1] proposed by UC Berkeley employs a straightforward synchronous mechanism to coordinate processor and accelerator operations, while the OpenHW Group's Core-V eXtension Interface (CV-X-IF) [2] leverages a modern, high-performance design. While effective for general applications, these interfaces struggle to meet AI workloads' demands—specifically massive parallelism and high-throughput data processing.

Specialized interface solutions have emerged to address AI-specific challenges. The Open Vector Interface (OVI) [3], proposed by Semidynamic, is a notable example that extends vector accelerator capabilities. However, such specialized interfaces typically cater to specific accelerator classes, thereby suffering from limited adaptability and expandability.

To address the issue, in this paper, we propose a unified AI accelerator interface designed to address diverse AI workloads. The proposed interface features the following major enhancements:

1. **Unified AI Accelerator Interface.** Besides supporting vector acceleration, the interface also incorporates matrix and tensor modes, enabling broader applicability across a range of AI tasks.
2. **Diverse Data Access Paths.** Building on existing data access mechanisms, the interface integrates multiple pathways—L1 cache access, bus access, coprocessor extension access, and inter-coprocessor data movement—to support various AI dataflow patterns.
3. **Virtual Memory Interface.** The interface allows accelerators to acquire physical addresses from the processor by supporting both private TLBs and direct MMU access, ensuring flexible and efficient memory address translation.
4. **Decoupled Microarchitecture Design.** In contrast to traditional interfaces, which impose rigid microarchitectural constraints, our proposed design supports multiple decoupled architectures, including flexible instruction decoding schemes and versatile accelerator CSR management strategies, improving compatibility with diverse accelerator designs and system configurations.

* Corresponding authors: gaomingjin@bosc.ac.cn, taoxie@pku.edu.cn

Components

Considering the similarity among interconnection signals across various AI accelerators and the processor, we develop a unified AI accelerator interface composed of nine signal groups (shown in Fig. 1): Issue, Scalar, Commit, Completed, Translation, Mem_fast, Extension, Move, and Mem_far—each serving a specific role.

Instruction Issuance and Response

The interface segment of instruction issuance and response consists of four signal groups enabling instruction issuance, data transmission, and execution feedback.

- The Issue group uniquely bridges the processor and accelerator. This group also coordinates communication with accelerator extension modules, managing instruction delivery and status feedback. It supports multiple accelerator-decoding schemes and omits scalar-operand read or write addresses if the first-stage decoder resides on the processor.
- The Scalar group transmits scalar operands and incorporates a flexible CSR management mechanism, relaying CSR data when centralized in the processor.
- The Commit group handles instruction cancellation during mispredicted speculative execution.
- The Completed group returns execution results to the processor and shares CSR status under centralized management.

These design choices promote microarchitectural decoupling, enhancing system flexibility and scalability.

Virtual Memory

The Translation group facilitates address translation for accelerators by enabling them to retrieve physical addresses from the processor. Compatible with both private TLB and direct MMU access modes, it supplies page table entries (PTEs) when a private TLB is present or returns physical addresses directly otherwise. Additionally, if memory-access exceptions require handling, the interface allows accelerators to communicate the exception cause and associated address back to the processor for resolution.

Data Access and Movement

This interface segment of data access and movement consists of four signal groups that support various data access methods: L1 cache, bus, coprocessor extension, and inter-coprocessor, addressing varied AI dataflow needs.

- The Mem_fast group enables rapid L1 cache access via address, data, mask, and exception signals, while the Mem_far group accesses remote L2/L3 caches or memory through bus transactions. Both groups employ ID signals for out-of-order memory accesses.
- The Extension group facilitates partial access to register data within the accelerator by conveying data and index information.
- The Move group, on the other hand, handles data movement between coprocessors by transmitting addresses, indices and synchronous signals.

Additionally, in line with existing interfaces, almost all signal groups in the proposed interface include both instruction ID and thread ID, to support accelerator out-of-order execution (OoO) and processor multi-hardware threading (Multi-HT).

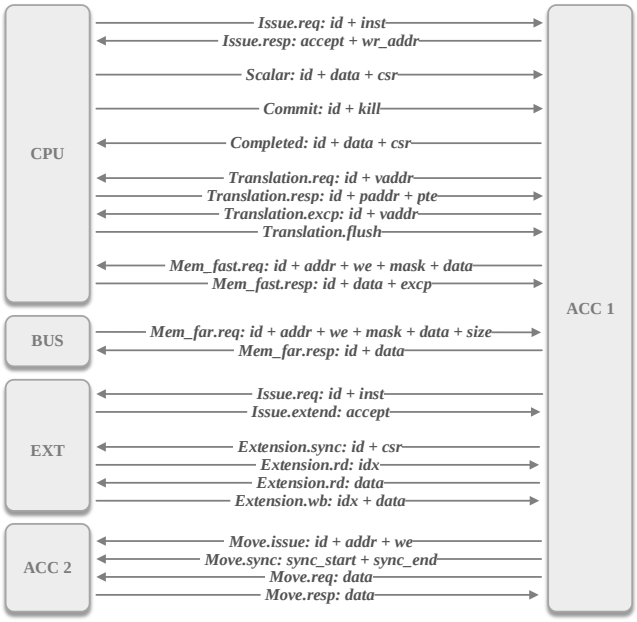

Figure 1: Overview of signals interfacing with accelerator 1.

Conclusion

In this paper, we have proposed a flexible, open AI accelerator interface that supports a variety of RISC-V extensions, diverse data access, virtual memory mechanisms, and decoupled microarchitectural designs via flexible decoding and CSR management. While the interface exhibits strong theoretical scalability, adaptability, and performance, practical validation is still required. Future efforts plan to prioritize design refinements and real-world testing to assess its efficacy.

References

[1] K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio et al. “The Rocket Chip Generator”. In: *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17 4* (Apr. 2016), pp. 6–2.

[2] OpenHW Group. *OpenHWGroup Specification: Core-V eXtension Interface (CV-X-IF).* OpenHW Group, 2024. [Online]. Available: https://docs.openhwgroup.org/projects/openhw-group-core-v-xif/en/v1.0.0/

[3] Semidynamics. *AVISPADO - VPU Interface*. GitHub, 2021. [Online]. Available: https://github.com/semidynamics/OpenVectorInterface