本科毕业设计英文文献综述

课题名称：	基于 **RISC-V** 存算一体芯片的编译器关键技术研究

学员姓名：	简泽鑫	学号：	**202102001019**
培养类型：	无军籍学员	专业：	计算机科学与技术（计算机系统）
所属学院：	计算机学院	年级：	**2021** 级
校内导师：	曾坤	职称：	副研究员
校外导师：	王晨曦	职称：	副研究员
所属单位：	计算机学院微电子与微处理器研究所		

国防科技大学计算机学院制

目 录

ABSTRACT

With the rapid advancement of artificial intelligence (AI) and big data applications, computational demands have grown exponentially, posing significant challenges to traditional computing architectures. The von Neumann architecture, in particular, suffers from the "von Neumann bottleneck" due to frequent data transfers between memory and processing units, which limits overall system efficiency. To address this, the Compute-In-Memory (CIM) architecture has emerged, integrating computation directly into memory units to minimize data movement, thereby overcoming the bottleneck and improving system performance and energy efficiency.

Meanwhile, RISC-V, as an open, extensible instruction set architecture (ISA), has become the preferred choice for building CIM chips due to its open-source nature, modular design, and customizability. However, RISC-V-based CIM chips exhibit heterogeneity and fragmentation, forcing developers to create multiple application versions for different architectures, resulting in low development efficiency and deployment challenges. A critical solution to this "programming wall" lies in decoupling software operations (e.g., computation, data communication and so on) from hardware configurations (e.g., heterogeneous compute units, memory hierarchies and so on) to enable AI application development independent of CIM IP design.

Furthermore, the irregular evolution of AI applications and the heterogeneous, fragmented landscape of CIM chips necessitate novel dynamic compilation optimization methods. These methods must account for the dynamic nature of AI workloads while fully leveraging the architectural features of future CIM chips. By dynamically adjusting compilation strategies in real time, generated code can better adapt to hardware environments, enhancing computational efficiency and resource utilization.

To this end, this project focuses on modifying the LLVM compiler for RISC-V CIM chips to support in-memory computing instructions. By analyzing the characteristics of RISC-V CIM architectures and exploring LLVM’s framework, we aim to extend LLVM with RISC-V CIM support, including instruction set extensions, memory model adaptations, and optimization strategy adjustments. This will enable automatic mapping of AI operators to hardware and generate correct instruction streams, effectively coordinating computing components and exploiting on-chip parallelism. The outcome will provide robust compilation support for RISC-V CIM chips, bridging the gap between software agility and hardware specialization.

KEY WORDS:

AI Compiler, CIM, RISC-V, LLVM

Chapter 1 Global Research Landscape and Development Prospects

The construction of Compute-In-Memory (CIM) chips based on the RISC-V instruction set is increasingly becoming a mainstream approach for AI accelerators. On the one hand, the RISC-V ISA offers significant advantages in openness and standardization, making it ideal for domain-specific chip development and customization. On the other hand, leveraging the RISC-V international community's momentum, the promotion and standardization of RISC-V extensions are driving the formation of unified and efficient AI programming models and system software frameworks. Consequently, tech giants such as Google, Meta (Facebook), and Microsoft have adopted RISC-V to develop proprietary AI chips. In 2023, RISC-V-based System-on-Chip (SoC) solutions achieved a market penetration rate of 2.6%, with a market size of $6.1 billion, and this growth trajectory continues to rise.

However, the trend toward deep domain-specific customization in AI has led to heterogeneous and fragmented characteristics in RISC-V CIM chips. First, CIM chips inherently exhibit heterogeneity, integrating specialized components such as tensor cores for accelerating matrix operations and vector cores for vector computations. Second, while various organizations design CIM chips based on the RISC-V ISA, their architectures diverge significantly in aspects like internal interconnects, memory access mechanisms, and compute unit configurations. This fragmentation creates substantial challenges for user programming and code optimization, emerging as a critical international issue in the AI chip domain.

The lack of standardized hardware-software interfaces exacerbates these challenges, forcing developers to tailor applications to specific chip architectures, thereby hindering scalability and increasing deployment costs. Addressing this requires not only architectural innovations but also advancements in compiler technologies and programming frameworks to bridge the gap between heterogeneous hardware and dynamic AI workloads.

Chapter 2 Current status of research at home and abroad

2.1 TVM Compiler Framework for AI Accelerators

TVM[1] is a compiler framework tailored for heterogeneous AI accelerators, introduced in 2018 by researchers at the University of Washington. It imports models from frameworks like TensorFlow, PyTorch, or ONNX and compiles them into linkable object modules. The lightweight TVM Runtime dynamically loads models via C-language APIs and provides entry points for other languages such as Python and Rust. At the intermediate optimization level, TVM employs a two-layer intermediate representation (IR)—Relay IR and Tensor IR—to perform hardware-agnostic optimizations (e.g., constant folding, operator fusion) and hardware-aware optimizations (e.g., computational pattern recognition and accelerated instruction generation).

TVM automates code optimization for diverse hardware targets, supports end-to-end learning-based optimization, and offers a flexible compilation workflow. However, when adapting to novel accelerators, developers must manually extend TVM's IR to accommodate new chip instructions and design architecture-specific optimization strategies, which limits its extensibility. As shown in the following figure, while TVM excels in general-purpose hardware support, its reliance on manual customization for emerging architectures highlights the need for more automated and scalable compiler solutions to address the growing diversity of AI accelerators.

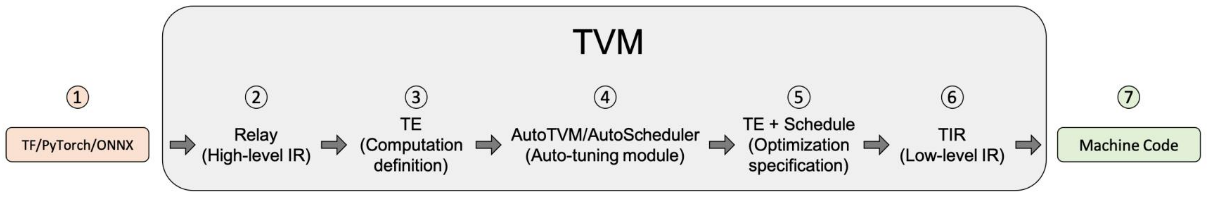

TVM
① TF/PyTorch/ONNX
② Relay (High-level IR)
③ TE (Computation definition)
④ AutoTVM/AutoScheduler (Auto-tuning module)
⑤ TE + Schedule (Optimization specification)
⑥ TIR (Low-level IR)
⑦ Machine Code

图 1　TVM 架构图

2.2 Triton Compiler for GPU Acceleration

Triton[2], released by OpenAI in 2021, is a compiler primarily designed to enhance the execution efficiency of deep learning applications on GPUs. It serves as a Python domain-specific language (DSL) for writing machine learning kernels and supports multiple hardware platforms, including CPUs, GPUs, and ASICs, by generating hardware-optimized kernels. At the intermediate optimization level, the Triton compiler employs block-level dataflow analysis techniques to automatically optimize the execution process of deep learning models.

While Triton demonstrates strong performance on NVIDIA and AMD GPUs, its support for RISC-V-based Compute-In-Memory (CIM) accelerators remains limited. This gap restricts its applicability in emerging heterogeneous environments where RISC-V CIM architectures are increasingly adopted for AI acceleration. As a result, developers targeting RISC-V CIM chips face challenges in leveraging Triton's optimization capabilities, necessitating further extensions to its compiler framework to accommodate the unique architectural features and instruction sets of CIM designs.

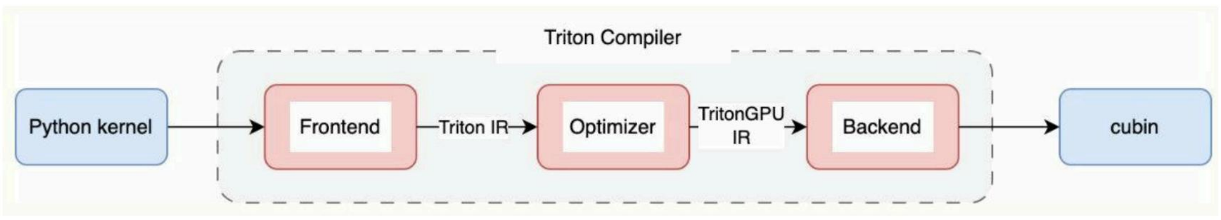
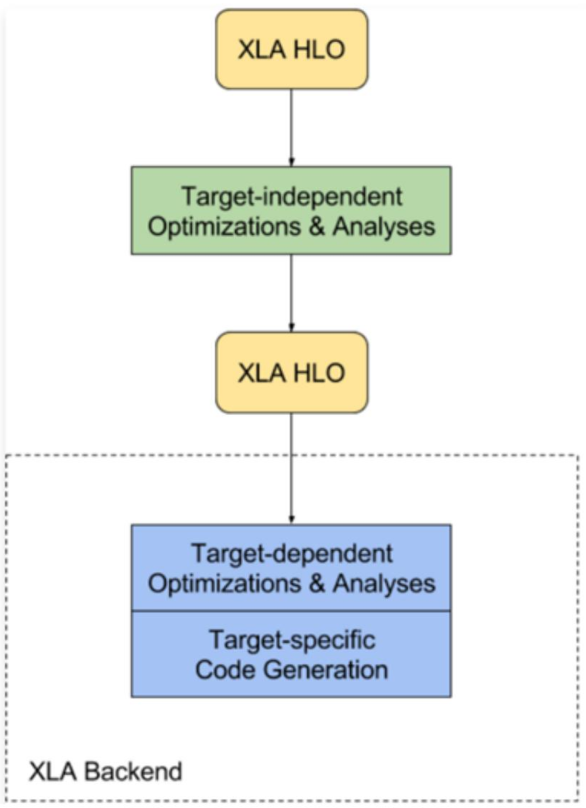


图 2　Triton 架构图

2.3 XLA Compiler for Deep Learning

XLA[3] (Accelerated Linear Algebra) is a domain-specific compiler for deep learning, initially developed by Google in 2017. It accepts models from machine learning frameworks such as PyTorch, TensorFlow, and JAX, performing mid-level optimizations that include simplifying algebraic expressions, optimizing memory data layouts, and enhancing execution scheduling. At the intermediate optimization level, XLA focuses on whole-model optimizations to improve computational efficiency and resource utilization.

However, XLA is primarily optimized for TensorFlow, requiring additional effort to support other frameworks effectively. Furthermore, its design targets conventional hardware like GPUs and Google's proprietary TPUs, with intermediate representations (IRs) abstracted at the level of deep learning operators. This high-level abstraction limits its adaptability to emerging RISC-V Compute-In-Memory (CIM) accelerators, which demand fine-grained control over memory-compute integration and specialized instruction sets. As a result, extending XLA to support RISC-V CIM architectures remains challenging, underscoring the need for compiler frameworks that better align with the heterogeneous and memory-centric nature of next-generation AI hardware.


XLA HLO
Target-independent Optimizations & Analyses
XLA HLO
Target-dependent Optimizations & Analyses
Target-specific Code Generation
XLA Backend
图 3　XLA 架构图

2.4 IREE Compiler and Runtime Framework

IREE[4] (Intermediate Representation Execution Environment), an open-source universal compilation and runtime framework, was released by Google in 2019. It takes high-level machine learning models as input and generates optimized executable code for diverse hardware targets. At the intermediate optimization level, IREE leverages MLIR (Multi-Level Intermediate

Representation) to perform multi-stage optimizations, ensuring efficient execution on target platforms. IREE provides high-performance compiler backends, and its hardware abstraction layer facilitates seamless integration of new hardware support.

However, the IREE framework primarily focuses on end-to-end optimization for deep learning models, lacking a unified programming model to harmonize heterogeneous compute units and memory hierarchies. This limitation becomes pronounced when targeting emerging architectures like RISC-V Compute-In-Memory (CIM) accelerators, where fine-grained coordination between computation and memory access patterns is critical. While IREE's modular design offers flexibility, the absence of architecture-specific abstractions for memory-centric computing hinders its ability to fully exploit the performance and energy efficiency advantages of CIM chips. Addressing this gap requires enhancing IREE's optimization pipeline to incorporate memory-compute co-design principles tailored to RISC-V CIM architectures.

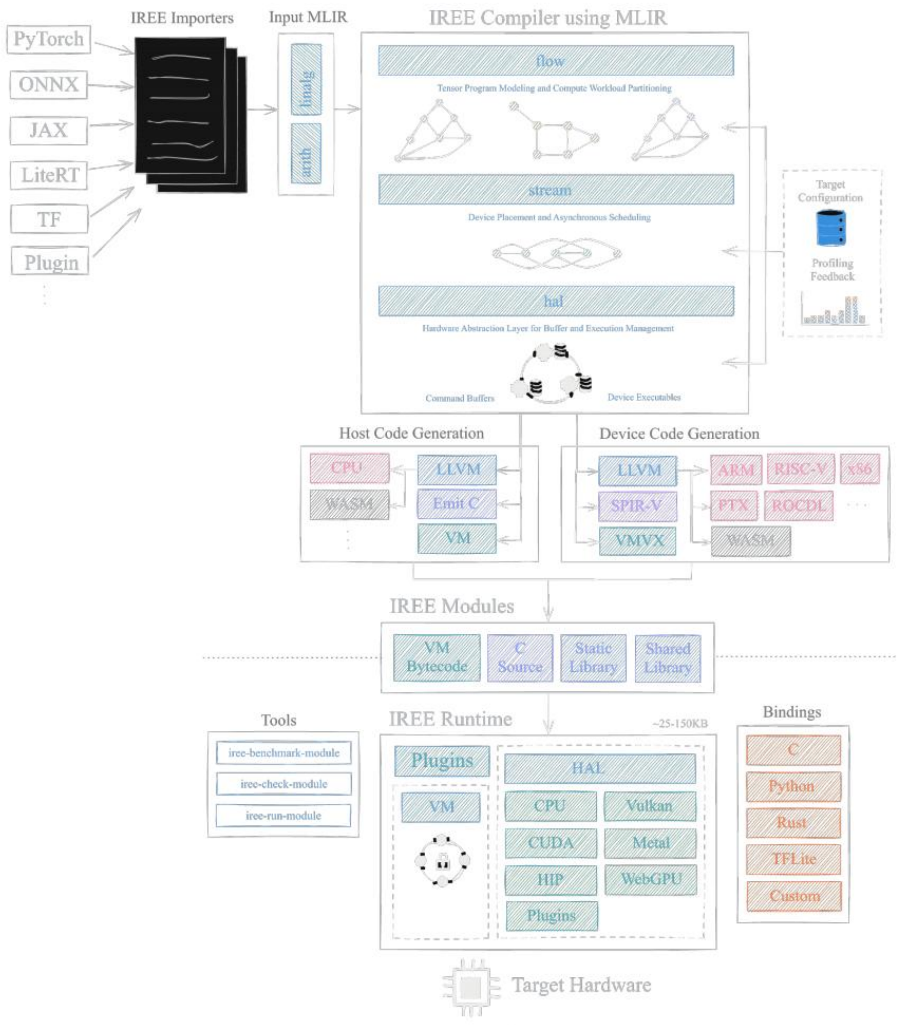


图 4 IREE 架构图

2.5 AKG Compiler Framework

AKG[5], a deep learning compiler framework led by Huawei, is designed to receive models from machine learning (ML) frameworks and generate hardware-optimized kernels for specific targets. At the intermediate optimization level, AKG employs automatic performance tuning tools to autonomously generate optimized kernels. The framework offers an automated tuning process that significantly enhances performance, making it particularly effective for Huawei's Ascend series AI accelerators and NVIDIA GPUs.

However, AKG exhibits limited support for RISC-V Compute-In-Memory (CIM) heterogeneous chips, which integrate memory and computation units to address the von Neumann bottleneck. While its automation capabilities excel in mainstream hardware ecosystems, the framework's current architecture lacks tailored optimizations for RISC-V CIM's unique memory-compute co-location paradigms and fragmented instruction sets. This limitation underscores the need for compiler frameworks that adapt to emerging memory-centric architectures, bridging the gap between generalized automation and domain-specific customization for next-generation AI accelerators.

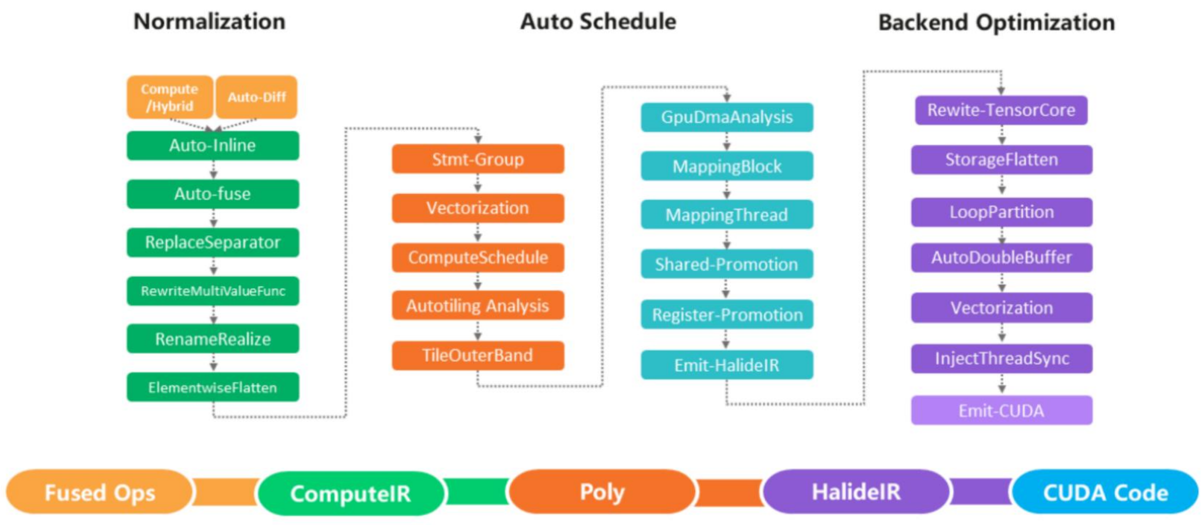


图 5　AKG 架构图

References

[1].CHEN T, MOREAU T, JIANG Z, 等. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning[C/OL]//13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)Carlsbad, CAUSENIX Association, 2018: 578-594. https://www.usenix.org/conference/osdi18/presentation/chen
[2].TILLET P, KUNG H T, COX D. Triton: an intermediate language and compiler for tiled neural network computations[C/OL]//Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming LanguagesPhoenix, AZ, USAAssociation for Computing Machinery, 2019: 10-19. https://doi.org/10.1145/3315508.3329973
[3].SNIDER D, LIANG R. Operator Fusion in XLA: Analysis and Evaluation[EB/OL] (2023). https://arxiv.org/abs/2301.13062
[4].iree[M]https://github.com/iree-org/iree
[5].ZHAO J, LI B, NIE W, 等. AKG: automatic kernel generation for neural processing units using polyhedral transformations[C]//PLDI 2021: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation2021