EXTENDED-ABSTRACT

An End-to-End Programming Model for AI Engine Architectures

MAKSIM LEVENTAL, The University of Chicago, Chicago, IL, United States

ARHAM KHAN, The University of Chicago, Chicago, IL, United States

RYAN CHARD, Argonne National Laboratory, Lemont, IL, United States

KYLE CHARD, The University of Chicago, Chicago, IL, United States

STEPHEN NEUENDORFFER, Advanced Micro Devices, Inc., Santa Clara, CA, United States

IAN FOSTER, The University of Chicago, Chicago, IL, United States

Open Access Support provided by:

Argonne National Laboratory

Advanced Micro Devices, Inc.

The University of Chicago

Published: 19 June 2024

HEART 2024: 14th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies
June 19 - 21, 2024
Porto, Portugal

https://doi.org/10.1145/3665283.3665294
ISBN: 9798400717277

An End-to-End Programming Model for AI Engine Architectures

Maksim Levental
mlevental@uchicago.edu
University of Chicago

Arham Khan
arham@uchicago.edu
University of Chicago

Ryan Chard
rchard@anl.gov
Argonne National Laboratory

Kyle Chard
chard@uchicago.edu
University of Chicago

Stephen Neuendorffer
stephen.neuendorffer@amd.com
AMD

Ian Foster
foster@uchicago.edu
University of Chicago

ABSTRACT

Coarse-Grained Reconfigurable Architectures (CGRAs) are becoming a promising alternative to conventional computing architectures such as CPUs, GPUs, and FPGAs when energy efficiency and high performance are required. Like CPUs and GPUs, CGRAs have processing elements (PEs) that can perform complex operations, such as vectorized arithmetic, and like FPGAs, they support a reconfigurable topology of components. Because of their coarser grain reconfigurability, they are less challenging to program than FPGAs but more challenging than CPUs and GPUs. This paper presents an end-to-end programming model for AMD AI Engine CGRAs, which enables programming simultaneously at a high level and a very implementation-specific level, all in the same language, all in the same "flow". Our programming model allows users to specify, implement, and test on-device, enabling the productive design of dataflow programs for streaming applications. The programming model is open source and includes a language frontend (Python eDSL), an MLIR based compiler, export paths to target codegen compilers, and runtime infrastructure. We show that our approach to language and compiler design enables users to program with much less friction and ceremony while preserving access to all features and device APIs necessary for achieving performance competitive with existing AI Engine programming models.

ACM Reference Format:
Maksim Levental, Arham Khan, Ryan Chard, Kyle Chard, Stephen Neuendorffer, and Ian Foster. 2024. An End-to-End Programming Model for AI Engine Architectures. In *14th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART'24)) (HEART '24), June 19–21, 2024, Porto, Portugal*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3665283.3665294

1 INTRODUCTION

Coarse-Grained Reconfigurable Architectures (CGRAs) are important due to their flexibility in adapting the instantiated configuration to the application; by being reconfigured, these architectures can selectively use or disuse various components, subsets of the architecture to gain much higher performance per Watt over a more diverse set of applications than conventional processors such as multi-core CPUs and GPUs [4]. In particular, owing to the reconfigurability of the device topology, CGRAs are well-suited for dataflow programs, i.e., specifications for the connectivity of functional units that explicitly represent and correspond to the flow of data in a program [3]. Archetypical in this class of programs are multi-layered Deep Neural Networks (DNNs), wherein individual layers potentially map to discrete subsets of the CGRA and with data flowing between them in the form of activations [4]. Finally, recently, as the necessary fabrication techniques have evolved to enable it, CGRAs have evolved to include processing elements (PEs) with functionality as rich as that of more conventional, standalone processors; today, the PEs found in CGRAs devices potentially have access to large local memories (data and program), scalar and vector ALUs, independent DMA controllers, and high-bandwidth streaming connections (both to other PEs and the host) [1]. Note the distinction between a CGRA and a GPU: while both provide access to an array of powerful PEs, only CGRAs (as of this writing) allow explicit specification and manipulation of connectivity between those PEs.

Conceptually, CGRAs have been around since the 1980s, but have failed to see wide deployment [2]. Primarily, there are two reasons for this: firstly, prior to the "deep learning renaissance", there was not such a surfeit of programs that could, naturally, be represented as dataflow graphs (and, thus, the platform lacked a strong, compelling use-case); secondly, prior to the availability of prefabricated CGRAs, deployment required designing one "whole cloth", either on FPGA or in ASIC. Besides being the veritable antithesis of "coarse-grained", digital design at the RTL level is typically far outside the comfort zone of most software developers. This lack of robust and familiar programming models prevented software developers from productively utilizing CGRAs and potentially still impedes their broader adoption. Recently, deep learning has taken over the world and prefabricated CGRAs such as AMD's AI Engine (AIE), Cerebras CS-1, SambaNova's SN40L have become commercially available[1]. The software "stacks" entailed by these coarse-grained devices significantly reduce the barrier to use by raising the abstraction level of the programming model from RTL. That notwithstanding, they do not eliminate the requirement that software developers be familiar with and manipulate various hardware layers.

In this work, we develop an "end-to-end" programming model for AMD's AI Engine such that a developer can design a dataflow, program the individual PEs, configure the device, launch the program, and manage host-device communications (vis-a-vis memory buffers) all in one Python script (or Jupyter notebook). Our flow is open source and even available as a `pip install`-able package.

[1] Cause and effect?

HEART '24, June 19–21, 2024, Porto, Portugal

ACM ISBN 979-8-4007-1727-7/24/06
https://doi.org/10.1145/3665283.3665294

This paper extends previous work, which introduced the frontend language design techniques in a more target-agnostic context [5]. The paper's contributions can be summarized as follows:

(1) A programming language frontend (Python embedded domain-specific language) which can be used to represent/specify CGRA specific concepts such as data movement, streaming connections, DMA access patterns, as well as PE-specific concepts (scalar and vector arithmetic operations); in addition, our frontend supports metaprogramming designed to enable easy extension/reuse by users;
(2) An MLIR-based compiler with support for two stream router implementations (optimal and approximate), buffer placement/allocation, and auto-vectorization;
(3) Integrations with target codegen compilers, and various host-side runtime bindings that enable seamless host-device-host data passing using familiar (NumPy) APIs;
(4) An evaluation of our end-to-end programming model for GEMMs on the Ryzen AI platform (an edge-device deployment of the AI Engine architecture).

To our knowledge, our flow is the first implementation of such an end-to-end programming model for AIE devices.

REFERENCES

[1] 2022. *AI Engines and Their Applications*. Technical Report. Advanced Micro Devices, Inc.
[2] Jeronimo C. Penha, Lucas B. Silva, Jansen M. Silva, Kristtopher K Coelho, Hector P. Baranda, José Augusto M. Nacif, and Ricardo S. Ferreira. 2019. ADD: Accelerator Design and Deploy-A tool for FPGA high-performance dataflow computing. *Concurrency and Computation: Practice and Experience* 31, 18 (2019), e5096.
[3] George Charitopoulos and Dionisios N. Pnevmatikatos. 2020. A CGRA Definition Framework for Dataflow Applications. In *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, Fernando Rincón, Jesús Barba, Hayden K. H. So, Pedro Diniz, and Julián Caba (Eds.). Springer International Publishing, Cham, 271–287.
[4] I. Stephen Choi and Yang-Suk Kee. 2015. Energy Efficient Scale-In Clusters with In-Storage Processing for Big-Data Analytics. In *Proceedings of the 2015 International Symposium on Memory Systems* (Washington DC, DC, USA) *(MEMSYS '15)*. Association for Computing Machinery, New York, NY, USA, 265–273. https://doi.org/10.1145/2818950.2818983
[5] Maksim Levental, Alok Kamatar, Ryan Chard, Kyle Chard, and Ian Foster. 2023. nelli: a lightweight frontend for MLIR. arXiv:2307.16080 [cs.PL]