

OpenACC to FPGA: A Directive-Based High-Level Programming Framework for High-Performance Reconfigurable Computing

Seyong Lee
Oak Ridge National Laboratory
lees2@ornl.gov

Jacob Lambert
University of Oregon
jlambert@cs.uoregon.edu

Jungwon Kim
Oak Ridge National Laboratory
kimj@ornl.gov

Jeffrey S. Vetter
Oak Ridge National Laboratory
vetter@computer.org

Allen D. Malony
University of Oregon
malony@cs.uoregon.edu

ABSTRACT

Accelerator-based heterogeneous computing has become a popular solution for energy-efficient high performance computing (HPC). Along these lines, Field Programmable Gate Arrays (FPGAs) have offered more advantages in terms of performance and energy efficiency for specific workloads than other accelerators. Nevertheless, FPGAs have traditionally suffered from several disadvantages limiting their deployment in HPC systems, mostly due to challenges of programmability and portability. We present a directive-based, high-level programming framework for high-performance reconfigurable computing. It takes a standard, portable OpenACC C program as input and generates a hardware configuration file for execution on FPGAs. We implemented this prototype system in our open-source OpenARC compiler, which uses the Altera OpenCL compiler as its backend. Preliminary evaluation of the proposed framework on an Intel Stratix V with five OpenACC benchmarks demonstrates that our proposed FPGA-specific compiler optimizations and novel OpenACC pragma extensions assist the compiler in generating more efficient FPGA programs.

CCS CONCEPTS

• **Hardware** → **Hardware accelerators**; *Emerging languages and compilers*; • **Software and its engineering** → **Source code generation**; • **Computing methodologies** → *Concurrent programming languages*; • **Computer systems organization** → *Pipeline computing*;

KEYWORDS

OpenACC, FPGA, OpenCL, directive-based programming, reconfigurable computing, OpenARC, sliding window

1 INTRODUCTION

Heterogeneous computing using accelerators such as GPUs has become a popular solution to the challenges of current and upcoming high-performance computing (HPC) systems, such as issues of performance, energy efficiency, reliability, and cost. Along these lines,

reconfigurable architectures like Field Programmable Gate Arrays (FPGAs) have received renewed interest due to their unique combination of performance and energy efficiency through hardware customizations. The reconfigurable nature of FPGAs allows for generation of hardware customized for applications, which can achieve much higher energy efficiencies compared to conventional CPUs and GPUs, resulting in increasing adoption in various industry and research platforms [5, 8, 13, 15].

However, FPGAs have not been widely adopted for HPC, mainly because of the difficulties in programming and optimizing FPGAs. Traditionally, FPGAs have been programmed using low-level hardware description languages (HDLs), which expose complex hardware designs to users. To alleviate FPGA programming complexities, several high-level synthesis (HLS) programming models have been proposed [2–4, 6, 12, 14]. Among these, OpenCL is the first standard programming model that has been adopted by all major FPGA vendors (e.g., Intel [1] and Xilinx [3]) and is functionally portable across diverse heterogeneous architectures.

Despite offering the potential for portability to GPUs and Xeon Phi, programming FPGAs with OpenCL has two contradicting problems: (1) the semantic gap between the OpenCL abstraction and the low-level hardware design is too wide, and (2) the programming abstraction offered by OpenCL is still considered too low for typical domain science programmers.

When programming FPGAs with OpenCL, the efficiency of the OpenCL compiler is critical to performance because it synthesizes all hardware logic for the input program. However, there exist several practical limits mainly caused by the semantic gap, such that existing FPGA OpenCL compilers tend to be very sensitive to specific code patterns [16]. Although lowering the programming abstraction level offered by OpenCL could reduce the compiler's sensitivity caused by the semantic gap, this would negatively affect programmability.

To address these issues, we propose a directive-based, high-level programming and optimization framework for efficient FPGA computing, built on top of the directive-based, Open Accelerator Research Compiler (OpenARC) [11], which performs source-to-source translation and optimization of an input OpenACC code to an output OpenCL code. The directive-based, high-level programming model offers better programmability, and the semantic information offered by user directives can better inform the OpenARC compiler for efficient OpenACC-to-OpenCL translation. Our goal is to generate specific OpenCL code patterns such that the underlying OpenCL compiler can correctly infer the intended semantics.

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor, or affiliate of the United States government. As such, the United States government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for government purposes only.

SC '18, November 11–16, 2018, Dallas, Texas

© 2018 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

This work makes the following contributions:

- We design, implement, and evaluate directive-based, high-level FPGA-specific optimizations, exploiting known FPGA programming paradigms, such as shift registers and sliding windows, to reduce overall FPGA runtimes and resource usage.
- We compare the performance of OpenACC applications with FPGA-specific optimizations to manual OpenCL applications with hand-tuned FPGA-specific optimizations.
- We compare the runtime and energy usage of OpenACC FPGA executions against OpenACC GPU executions and OpenMP CPU executions.

2 OPENARC: BASELINE OPENACC-TO-FPGA TRANSLATION FRAMEWORK

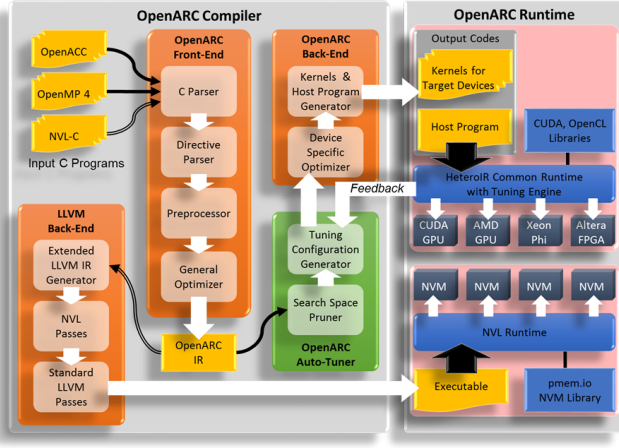


Figure 1: OpenARC System Architecture

The baseline OpenACC-to-FPGA translation framework is based on an open-source OpenACC compiler called OpenARC. As shown in Figure 1, OpenARC performs source-to-source translation of the input OpenACC or OpenMP program and generates either CUDA or OpenCL as an output programming model. To port OpenACC to FPGAs, we use OpenCL as the output model and the Altera Offline Compiler (AOCL) [7] as its backend compiler. Because OpenCL is architecture-independent, the same OpenCL code generated for GPUs can be executed on FPGAs, and is functionally portable. However, the unique architectural aspects of FPGAs necessitate customized translation and optimization strategies to achieve performance portability. The key benefit of using FPGAs is that they support wide, heterogeneous, and deeply pipelined parallelism customized for the input program. In other types of accelerators, such as GPUs, parallelism is achieved by replicating the same generic computational units multiple times, which limits the amount of work that the device can perform each clock cycle. Alternatively, FPGAs achieve parallelism by duplicating only the logic that the input program requires. In high-level FPGA programming with OpenCL, the OpenCL compiler synthesizes all the hardware logic for the input program, and thus the efficiency of the compiler is critical for performance. Therefore, we extended the OpenARC

passes to generate output OpenCL codes in a manner friendly to the underlying OpenCL compiler.

3 DIRECTIVE-BASED, HIGH-LEVEL FPGA-SPECIFIC OPTIMIZATIONS

The proposed framework consists of automatically applied optimizations within the OpenARC compiler, alternative FPGA-specific implementations of existing OpenACC parallelism directives, and directive extensions for optimizations not easily described with existing OpenACC directives.

We have developed these optimizations [9, 10] with the goal of abstracting away low-level FPGA-specific programming such as dynamic memory-transfer alignment for exploiting Direct Memory Access (DMA). Also, they allow programmers to use existing high-level parallelization approaches, such as OpenACC parallel, reduction, and collapse clauses, to efficiently program FPGAs. Finally, through directive extensions they allow programmers to utilize some patterns unique to FPGAs such as the FPGA-specific hardware mechanism of kernel pipelining and an FPGA-specific sliding window approach.

4 PRELIMINARY EVALUATION

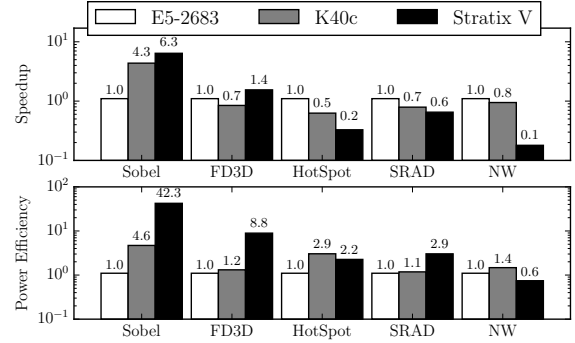


Figure 2: Comparison of OpenMP CPU executions, OpenACC GPU executions, and OpenACC FPGA executions.

To show that high-level FPGA programming can offer a competitive solution, we compare directive-based approaches across three devices. In Figure 2 we compare OpenACC programs executed on an FPGA (black) and a GPU (gray) with OpenMP programs executed on a CPU (white).

The upper half of the figure shows a runtime comparison across all three devices, while the lower half shows a comparison of energy consumption (Joules), both normalized to a CPU baseline of 1.

The preliminary results show that in some cases directive-based approaches for FPGA programming, when supported with FPGA-specific optimizations such as those developed in [9, 10], can compete with directive-based approaches for other devices like GPUs and CPUs when considering runtime and energy usage. However, in other cases more optimizations are needed.

REFERENCES

- [1] [n. d.]. *Intel FPGA SDK for OpenCL*. <https://www.altera.com/products/design-software/embedded-software-developers/opencl/overview.html>.

- [2] [n. d.]. *Mentor - DK Design Suite*. <https://www.mentor.com/products/fpga/handel-c/dk-design-suite/>.
- [3] [n. d.]. *Xilinx - SDSoC Development Environment*. <https://www.xilinx.com/products/design-tools/software-zone/sdsoc.html>.
- [4] S. R. Alam, P. K. Agarwal, M. C. Smith, J. S. Vetter, and D. Caliga. 2007. Using FPGA Devices to Accelerate Biomolecular Simulations. *IEEE Computer* 40, 2 (2007), 66–73.
- [5] Amazon. [n. d.]. Amazon EC2 F1 Instances. <https://aws.amazon.com/ec2/instance-types/f1/>
- [6] Matthew Aubury, Ian Page, Geoff Randall, Jonathan Saul, and Robin Watts. 1996. Handel-C Language Reference Guide. Oxford University Computing Laboratory.
- [7] T.S. Czajkowski, U. Aydonat, D. Denisenko, J. Freeman, M. Kinsner, D. Neto, J. Wong, P. Yiannacouras, and D.P. Singh. 2012. From OpenCL to High-Performance Hardware on FPGAs. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*. IEEE.
- [8] Intel. [n. d.]. Altera FPGA-Based Storage Reference Design. <https://newsroom.intel.com/news-releases/altera-fpga-based-storage-reference-design-doubles-life-nand-flash/>
- [9] Jacob Lambert, Seyong Lee, Jungwon Kim, Jeffrey S. Vetter, and Allen D. Malony. 2018. Directive-Based, High-Level Programming and Optimizations for High-Performance Computing with FPGAs. In *ACM International Conference on Supercomputing (ICS '18)*.
- [10] S. Lee, J. Kim, and J. S. Vetter. 2016. OpenACC to FPGA: A Framework for Directive-Based High-Performance Reconfigurable Computing. In *Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS '16)*. 544–554.
- [11] Seyong Lee and Jeffrey Vetter. 2014. OpenARC: Open Accelerator Research Compiler for Directive-Based, Efficient Heterogeneous Computing. In *HPDC '14: Proceedings of the ACM Symposium on High-Performance Parallel and Distributed Computing, Short Paper*.
- [12] Preeti Ranjan Panda. 2001. SystemC: A Modeling Platform Supporting Multiple Design Abstractions. In *Proceedings of the 14th International Symposium on Systems Synthesis (ISSS '01)*. 75–80.
- [13] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo-Young Kim, Sitaram Lanka, James Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. 2014. A Reconfigurable Fabric for Accelerating Large-scale Datacenter Services. In *Proceeding of the 41st Annual International Symposium on Computer Architecture (ISCA '14)*. IEEE Press, Piscataway, NJ, USA, 13–24. <http://dl.acm.org/citation.cfm?id=2665671.2665678>
- [14] M. C. Smith, J. S. Vetter, and X. Liang. 2005. Accelerating Scientific Applications with the SRC-6 Reconfigurable Computer: Methodologies and Analysis. In *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05)*.
- [15] Swagath Venkataramani, Ashish Ranjan, Subarno Banerjee, Dipankar Das, Sasikanth Avancha, Ashok Jagannathan, Ajaya Durg, Dheemanth Nagaraj, Bharat Kaul, Pradeep Dubey, and Anand Raghunathan. 2017. ScaleDeep: A Scalable Compute Architecture for Learning and Evaluating Deep Networks. *SIGARCH Comput. Archit. News* 45, 2 (June 2017), 13–26. <https://doi.org/10.1145/3140659.3080244>
- [16] Hamid Reza Zohouri, Naoya Maruyama, Aaron Smith, Motohiko Matsuda, and Satoshi Matsuoka. 2016. Evaluating and Optimizing OpenCL Kernels for High Performance Computing with FPGAs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16)*. 35:1–35:12.