HPVM2FPGA: Enabling True Hardware-Agnostic FPGA Programming

Adel Ejjeh, Leon Medvinsky, Aaron Councilman,
Hemang Nehra, Suraj Sharma, Vikram Adve
University of Illinois at Urbana-Champaign
USA

Luigi Nardi
Lund University and Stanford University
Sweden and USA

Eriko Nurvitadhi
Intel
USA

Rob A Rutenbar
University of Pittsburgh
USA

ABSTRACT

We present HPVM2FPGA, a framework that enables hardware-agnostic programming of FPGAs by coupling compiler optimization techniques with Design Space Exploration (DSE). By using a suitable compiler Intermediate Representation designed for heterogeneous parallel systems, called HPVM, suitable compiler optimizations, and a state-of-the-art DSE framework (HyperMapper), we created an extensible flow that automatically generates high-performing code for FPGAs. In its current state, HPVM2FPGA can achieve up to 33× speedup compared to unoptimized baselines, and can match hand-tuned code for some programs. With an optimization framework that can be easily extended with more compiler transformations, we expect HPVM2FPGA to match hand-tuned code for most programs as the system matures with more optimizations.

1 INTRODUCTION

FPGAs have become a more attractive hardware target for software teams due to their increased public availability. However, current FPGA programming paradigms are generally intractable for this category of developers. While higher-level paradigms, like High-Level Synthesis (HLS), have gained traction in the accelerator design community, they still require hardware-specific knowledge and tuning, which does not make them an acceptable solution for software teams that do not possess hardware-design expertise.

It is beyond doubt that true, general, hardware-agnostic programming of FPGAs remains a holy grail in our community. Achieving such a capability requires three main components: 1) a representation that can efficiently capture parallelism in an application and easily identify application components for acceleration, 2) a compiler and autotuner that can tune hardware-agnostic kernels by automatically selecting FPGA-specific optimizations, while also compiling the host code, and 3) a runtime system that can transparently interface the host and device code without any extra input from the programmer. This kind of end-to-end flow is largely missing in the FPGA design community.

While we acknowledge the difficulty of this problem, we do believe that it is attainable through a compiler-centric approach. We take one step closer to achieving general hardware-agnostic FPGA programming by presenting HPVM2FPGA, a novel compilation framework that leverages a suitable Intermediate Representation (IR) with automatic and sophisticated compiler optimizations, and performs Design Space Exploration (DSE) to automatically generate the most optimized kernels it can find, shifting this burden from programmer to compiler.

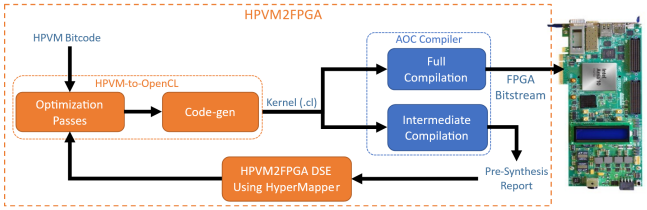


Figure 1: HPVM-to-FPGA back end.

2 HPVM2FPGA

HPVM2FPGA leverages the existing HPVM infrastructure [3] and adds to it: 1) an HPVM-to-FPGA back end, 2) runtime extensions, and 3) an extensible optimization framework which includes a number of compiler optimizations and DSE, with the ability to add more optimizations in the future fairly easily. We chose the HPVM IR for two main reasons: 1) Its hardware agnostic nature allows us to easily represent general, parallel, hardware-agnostic programs, and (in future) target them to systems containing a mix of FPGAs and other devices; and 2) Its ability to represent the different levels of parallelism, using a hierarchical dataflow graph (DFG), is crucial for optimizing hardware-agnostic codes for hardware-specific targets like FPGAs.

The HPVM-to-FPGA back end, shown in Figure 1, comprises two main components: an HPVM-to-OpenCL back end, which is our contribution, and the Intel FPGA SDK for OpenCL (AOC compiler), which we leverage as is. The compilation process goes as follows: starting with an input program in an HPVM-compatible language (Hetero-C++, HPVM-C, or any other language that can be compiled to HPVM in the future), the compiler lowers the source code into a hardware-agnostic HPVM IR representation of the program. Next, an optimization step, described below, optimizes the HPVM leaf nodes, which will become FPGA kernels; the framework supports both inter-node and intra-node optimizations to achieve the best possible designs. Then, the HPVM-to-FPGA back end generates an OpenCL file containing the optimized kernels, and generates the required runtime code that launches and manages these kernels (i.e. creates OpenCL buffers, sets the arguments, copies the memory, etc.) into the host LLVM module. Finally, the OpenCL kernels get synthesized using AOC, and the host module gets compiled using LLVM's x86 back end to generate a binary. Note that Figure 1

only shows the kernel compilation process, omitting the host-code compilation for simplicity.

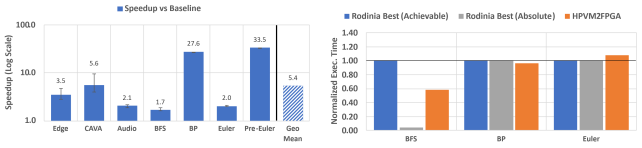
We currently have seven optimizations in HPVM2FPGA, implemented as either HPVM DFG or LLVM transformations. These optimizations are: automatic input buffering, guided argument privatization, automatic loop unrolling, greedy loop fusion, automatic node fusion, automatic task parallelism, and automatic `ivdep` insertion. All the optimizations have been parameterized for DSE, where the parameter is either a boolean turning it on or off, or an integer that sets a specific value (e.g. unroll factors). Our DSE framework, which uses HyperMapper (HM) [4], selects and tunes the optimization passes to obtain the best possible performance on the FPGA for a given application, by finding the best value for every parameter of every optimziation. The DSE objective function is calculated using a performance model that we have devised. This model uses information extracted from the AOC pre-synthesis report (specifically the loop initiation intervals, loop body latencies, and frequency), to estimate the execution time of a program. The model starts by estimating the execution time of each loop nest in an HPVM node (i.e. kernel), then of the node itself, then of the entire dataflow graph using a critical path analysis and graph traversal. Additionally, the resource utilization estimate for each design is extracted from the AOC reports, and used to determine whether or not the given design is "Valid" (i.e. fits on the FPGA). The objective and "validity" are then used by HM to traverse the parameter space.

HPVM2FPGA is designed to be a cornerstone for new and improved hardware-agnostic FPGA compilers by providing a fully extensible framework. Every component described above is modular and extensible, making it easy to add more powerful optimizations, add more advanced code-generation options to lower-level hardware IRs, or improve on the DSE performance modeling.

3 EXPERIMENTAL EVALUATION

We evaluate our framework on a selection of benchmarks, including a 3D spatial audio encoder (*Audio*) from the Illinois Extended Reality (ILLIXR) testbed [2], a camera vision pipeline (*CAVA*), an image processing edge detection pipeline (*Edge*) [3], and four multi-kernel benchmarks from the Rodinia benchmark suite (breadth-first search (BFS), backpropagation (BP), and two algorithms of computational fluid dynamics: euler (Euler) and euler with precomputed fluxes (Pre-Euler)) [1]. Three of these Rodinia benchmarks have been hand-tuned for the Arria 10 FPGA [6]. Each benchmark was ported to hardware-agnostic code in HPVM-C or Hetero-C++, and then compiled using HPVM2FPGA. Our evaluation setup uses an Arria 10 GX FPGA Development Kit FPGA with 2GB on-board memory, connected over PCIe to an Intel Xeon W-2775 host CPU with 256 GB main memory. For synthesis, we use the Intel FPGA SDK for OpenCL 19.3. Our results are presented in Figure 2.

To study the performance that our framework can achieve, we compared the DSE-generated designs to a version compiled using our compiler without applying any optimizations (Figure 2(a)). Our framework was able to achieve speedups ranging from 1.7× to 33.5×, with applications that benefit more from the currently-supported optimizations getting higher speedups. Given that these benchmarks represent different workloads, with different characteristics, this shows that our framework is effective on a wide variety



(a) (b)

Figure 2: (a) HPVM2FPGA DSE vs unoptimized baselines. (b) HPVM2FPGA vs hand-tuned kernels.

of workload types, and we expect that as the framework matures with more optimizations speedups would also improve.

Next, to compare HPVM2FPGA-generated designs to hand-tuned FPGA kernels, we synthesized the versions of those Rodinia benchmarks that were hand-optimized by Zohouri [5, 6] and compared them to our DSE results. These benchmarks were BP, Euler, and BFS (Pre-Euler lacked an optimized version in the repository). Our results (Figure 2(b)) show that for BP and Euler, we are able to match the performance of the hand-tuned code. For BFS, we were able to outperform the version that was hand-tuned without using algorithmic changes because DSE explored more optimization options. However, we were unable to match the version that included algorithmic changes (i.e. "Absolute" best in the figure). In this case the "Absolute" best point uses NDRange kernels, whereas we always generate single work item kernels in our framework. We believe that as our framework matures to also support NDRange kernel generation as part of DSE we would be able to match the hand-tuned code for more applications.

4 CONCLUSION AND FUTURE DIRECTIONS

We presented HPVM2FPGA, a novel extensible framework designed to enable more powerful hardware-agnostic programming of FPGAs. Our framework uses a suitable compiler IR, HPVM, and powerful compiler optimizations, coupled with Design Space Exploration (DSE) that tunes the optimization parameters, to find the best combination of optimizations for a hardware-agnostic application and a given FPGA. Our goal is to pave the path for future hardware-agnostic FPGA programming research, by providing a modular and extensible framework that would keep on improving with more compiler optimizations, more advanced back end code-generation techniques, and faster and more accurate performance estimation models used in DSE.

With that, there are numerous possible directions to move forward within this research goal of providing hardware-agnostic programming of FPGAs. Some of these directions include incorporating more loop-level, memory-level, and data-movement optimizations that have been shown to improve performance in the context of HLS; extending the framework to support optimizing across multiple different devices in a heterogeneous system; adding automatic selection of nodes for device offloading; adding more options for backend code-generation (e.g. generating NDRange Kernels, supporting Xilinx FPGAs); and adding more advanced inter-kernel optimizations.

REFERENCES

[1] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W. Sheaffer, Sang-Ha Lee, and Kevin Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*. 44–54. https://doi.org/10.1109/IISWC.2009.5306797

[2] Muhammad Huzaifa, Rishi Desai, Samuel Grayson, Xutao Jiang, Ying Jing, Jae Lee, Fang Lu, Yihan Pang, Joseph Ravichandran, Finn Sinclair, Boyuan Tian, Hengzhi Yuan, Jeffrey Zhang, and Sarita V. Adve. 2021. Exploring Extended Reality with ILLIXR: A new Playground for Architecture Research. arXiv:2004.04643 [cs.DC]

[3] Maria Kotsifakou, Prakalp Srivastava, Matthew D. Sinclair, Rakesh Komuravelli, Vikram Adve, and Sarita Adve. 2018. HPVM: Heterogeneous Parallel Virtual Machine. In *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Vienna, Austria) *(PPoPP '18)*. ACM, New York, NY, USA, 68–80. https://doi.org/10.1145/3178487.3178493

[4] Luigi Nardi, David Koeplinger, and Kunle Olukotun. 2019. Practical design space exploration. In *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 347–358.

[5] Hamid Resa Zohouri. [n. d.]. fpga-opencl-benchmarks/rodinia_fpga. https://github.com/fpga-opencl-benchmarks/rodinia_fpga

[6] Hamid Reza Zohouri. 2018. *High performance computing with FPGAs and OpenCL*. Ph. D. Dissertation. Tokyo Institute of Technology, Tokyo, Japan, Tokyo, Japan.