On the Portability of the OpenCL Dwarfs on Fixed and Reconfigurable Parallel Platforms

Konstantinos Krommydas*, Muhsen Owaida†, Christos D. Antonopoulos†, Nikolaos Bellas† and Wu-chun Feng*
*Department of Computer Science, Virginia Tech
†Department of Electrical and Computer Engineering, University of Thessaly
E-mails: {kokrommy, wfeng}@vt.edu, {mowaida, cda, nbellas}@uth.gr

***Abstract*—The proliferation of heterogeneous computing systems presents the parallel computing community with the challenge of porting legacy and emerging applications to multiple processors with diverse programming abstractions. OpenCL is a vendor-agnostic and industry-supported programming model that offers code portability on heterogeneous platforms, allowing applications to be developed once and deployed "anywhere." In this paper, we use the OpenCL implementation of the OpenDwarfs, a benchmark suite that captures patterns of computation and communication common to classes of important applications, as delineated by Berkeley's Dwarfs. We evaluate portability across multicore CPU, GPU, APU (CPUs+GPUs on a die), the Intel Xeon Phi co-processor, and the FPGA. To realize FPGA portability, we exploit SOpenCL (Silicon OpenCL), a CAD tool that automatically converts OpenCL kernels to customizable hardware accelerators. We show that a single, unmodified OpenCL code base, i.e., OpenDwarfs, can be effectively used to target multiple, architecturally diverse platforms.**

***Keywords*-OpenCL, CPU, GPU, APU, Xeon Phi, FPGA, dwarfs, portability**

I. Introduction

Modern high-performance computing systems have become increasingly heterogeneous in nature. This trend mandates that heterogeneous components, such as CPUs, GPUs (discrete or fused with CPUs on the same die), co-processors, such as Intel Xeon Phi of the Many Integrated Cores (MIC) architecture, or FPGAs, be used concurrently and efficiently. As a result, a new challenge for software developers has arisen in that each of these components generally uses its own distinct programming model, possesses its own architectural peculiarities, and introduces complex memory hierarchies and multiple address spaces within a single compute node. Further expertise, typically outside the realm of software developers is also required to design and exploit FPGAs and ASICs.

OpenCL [1] is a vendor-agnostic and industry-supported programming model, which aspires to serve as a unified development framework for heterogeneous computing systems, allowing applications to be developed once and deployed "anywhere." While OpenCL programmers may wish to resort to platform-specific optimizations in order to achieve performance portability, this approach is labor-intensive, limits the overall code portability and cannot be applied if the target platform is not known *a priori*.

In this paper, we use OpenDwarfs [2], a collection of OpenCL codes that correspond to Berkeley's Dwarfs [3], to evaluate the practicality and effectiveness of executing applications on architecturally diverse substrates, using a single, unmodified code base in the form of OpenCL kernels. We experiment with a modern CPU, GPU, APU, the Intel Xeon Phi co-processor, and the FPGA. For FPGAs, we use SOpenCL [4] to transform OpenCL kernels to equivalent synthesizable Verilog hardware descriptions, thus facilitating the exploitation of FPGAs as hardware accelerators without the overhead of additional development cost and expertise. This is one of the first research efforts to use and evaluate OpenCL as a language to describe hardware. It is also one of the first works to evaluate Intel Xeon Phi using OpenCL across a diverse set of applications.

II. Background

Here we provide a brief overview of FPGA technology and the SOpenCL tool used for automating translation of OpenCL code to Verilog for hardware generation.

A. FPGA Technology

Reconfigurable devices (FPGAs) are high-density arrays of uncommitted logic blocks that are configured post-fabrication. The functionality of FPGAs is determined through configuration bits that specify the functionality of the configurable logic blocks and the routing channels between them. Since reconfigurable logic is more efficient in implementing specific applications than multicore CPUs, it enjoys higher power efficiency than any general-purpose computing substrate. One of FPGAs' main drawbacks, with respect to programmability, is that they are traditionally programmed using Hardware Description Languages (VHDL or Verilog), a time-consuming and labor-intensive task, which requires deep knowledge of low-level hardware details.

B. Silicon OpenCL (SOpenCL)

We use the SOpenCL tool [4] to automatically generate hardware accelerators for the OpenDwarf kernels, thus dramatically minimizing development time and increasing productivity. SOpenCL converts OpenCL kernels to equivalent synthesizable Verilog, which can in turn be used to configure an FPGA. The SOpenCL front end is a source-to-source compiler that adjusts parallelism granularity of the OpenCL kernel to better match the hardware capabilities of the FPGA. The output of this stage is semantically equivalent C code at the work-group granularity. SOpenCL back-end flow, which is based on the LLVM compiler infrastructure [5], generates the synthesizable Verilog of the accelerator.

III. Experimental Study and Discussion

This section presents our experiences with OpenDwarfs across five types of parallel computing platforms, that span a variety of architectures: CPU (AMD Opteron 6272), GPU (AMD Radeon HD 7970), APU (AMD Llano A8-3850 — a heterogeneous system that combines an AMD A8-3850 CPU with an AMD Radeon HD 6550D integrated GPU on the same die), Intel MIC co-processor (Intel Xeon Phi P1750), and FPGA (Xilinx Virtex-6 LX760).

Using OpenCL as a common programming model, we were able to quickly compile (or synthesize in the FPGA case) and run all benchmarks on five widely different architectures, without modifying code, thus dramatically increasing productivity. Below, we discuss our observations on performance of this unmodified, unoptimized code across our test platforms.[1]

OpenCL vendor toolchains were able to exploit multi- and many-core architectures, providing scalable speedups for applications whose working set can be easily partitioned. GPUs and Xeon Phi offer much higher single-precision FP performance compared to both CPUs and FPGAs, which helps to explain their superior performance for compute-bound applications that have a high computation/communication ratio. A large number of applications spanning scientific computing, linear algebra, and bioinformatics are characterized by independent, operation-heavy computations and are prime targets for GPU acceleration.

Interestingly, but not unexpectedly, state-of-art GPUs can attain good performance on applications that are not data parallel. Similar to CPUs, these architectures include cache hierarchies to reduce memory access latency, thus alleviating lack of memory-level parallelism and runtime-dependent accesses. Erratic branching, the realm of conditional-heavy integer code, seems to be the only case in which GPUs are outperformed by CPUs in our evaluation.

Data transfers are, as expected, costly on accelerator platforms that communicate with main memory through a PCIe interface. Another interesting observation concerns the APU and the standalone HD7970 GPU. Although the Radeon HD 7970 is significantly more powerful in terms of computational performance, Llano's integrated GPU enjoys tighter coupling with the system's main memory and the CPU cache hierarchy, resulting in lower data transfer times. Therefore, it outperforms the discrete GPU when the data transfer overhead is a significant portion of total execution time, or whenever the characteristics of the application (e.g., data dependencies and low computation/communication ratio) prohibit the Radeon HD 7970 from exploiting its peak computational capability.

The FPGA results are negatively biased by the fact that all massively parallel OpenDwarfs are based on floating-point (FP) computations for which GPUs have a distinct advantage. Conversion from FP to fixed-point arithmetic, whenever that conversion can be tolerated by the application, would offset this disadvantage. However, we decided to retain bit-exactness and use IEEE-754 FP arithmetic. FPGAs outperformed all other accelerators in integer applications. Finally, careful placement of the input working set into on-chip memories to exploit the huge internal FPGA bandwidth is of paramount importance in data-parallel applications.

Finally, while Xeon Phi features 2 TFLOPS (SP) of theoretical peak performance, our experimental results reveal that initial functional code portability is trivial and speed enhancements are observed, depending on the application's nature. However, careful application of platform-specific optimizations is necessary for exploiting Xeon Phi's full capabilities.

[1] For details on the OpenDwarfs applications mentioned in this section, refer to [2]

IV. Conclusions

In this work we presented an evaluation of the functional portability of OpenCL by executing an unmodified code base of the OpenDwarfs benchmark suite on five architecturally diverse parallel platforms: CPU, GPU, APU, FPGA and the Xeon Phi co-processor. Our experimental study demonstrates that OpenCL can be realistically used as a hardware description language (HDL) by employing tools like SOpenCL that automatically translate unmodified OpenCL code to Verilog and eliminate the burden of HDL programming. Such tools facilitate the proliferation of OpenCL as a "universal" programming language and, according to our studies, offer FPGA implementations whose performance typically lies between that of multi-core CPUs and GPUs or even surpasses it in specific applications. Further research towards making such tools even more efficient will help bridge the performance gap.

Acknowledgments

This work was supported in part by the Institute for Critical Technology and Applied Science (ICTAS) at Virginia Tech and co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program. The authors would also like to thank the OpenDwarfs project, supported by the NSF Center for High-Performance Reconfigurable Computing (CHREC).

References

[1] A. Munshi, Editor, *The OpenCL Specification. Version: 1.2*, Khronos OpenCL Working Group, November 2012.

[2] W.-C. Feng, H. Lin, T. Scogland, and J. Zhang, "OpenCL and the 13 Dwarfs: A Work In Progress," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE)*, Boston, MA, April 2012, pp. 291–294.

[3] K. Asanovic et al., "The Landscape of Parallel Computing Research: A View from Berkeley," Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Tech. Rep. UCB/EECS-2006-183, December 2006.

[4] M. Owaida, N. Bellas, K. Daloukas, and C. D. Antonopoulos, "Synthesis of Platform Architectures from OpenCL Programs," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Salt Lake City, UT, May 2011.

[5] C. Lattner and V. Adve, "LLVM: A Compilation Framework for Lifelong Program Analysis Transformation," in *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*, Palo Alto, CA, March 2004, pp. 75–86.