Rosetta: A Realistic Benchmark Suite for Software Programmable FPGAs

Udit Gupta, Steve Dai, Zhiru Zhang
Computer Systems Laboratory, Electrical and Computer Engineering, Cornell University, Ithaca, NY
ug28@cornell.edu, stevedai@csl.cornell.edu, zhiruz@cornell.edu

1. Introduction

Extreme-scale integration of modern system-on-chip (SoC) and escalating design complexity of emerging applications reiterate the importance of designing at a higher level of abstraction and call for an ever-improving suite of high-level synthesis (HLS) tools to enable optimization opportunities that are otherwise infeasible at the register-transfer level (RTL) [1]. The need for software programmability is especially pertinent to FPGAs as they emerge from logic devices to computing devices by combining a number of hardened blocks with the programmable fabric [4]. While leveraging the productivity and performance benefits of this new design automation paradigm relies on a line of novel HLS algorithms to exploit the reconfigurability, massive fine-grained parallelism, and performance per watt advantage of FPGAs, there lacks a set of benchmarks to provide a practical evaluation of the quality of results (QoR) generated by these algorithms.

We present Rosetta: a suite of software applications from a range of domains with hardware constraints targeting heterogeneous FPGAs. Unlike previous efforts, Rosetta provides breath by allowing benchmarking of parametrizable applications beyond the kernel level and incorporates applications from emerging domains. The initial set of applications are implemented in C/C++ and OpenCL to accommodate both the sequential and parallel programming models commonly supported by HLS. Rosetta also emphasizes depth by specifying realistic design constraints for each application. Our multi-dimensional approach aims to merge research and development among emerging HLS algorithms, software applications, and heterogeneous reconfigurable accelerators.

2. Benchmark Suite

HLS benchmarking poses several unique challenges. Unlike software design where performance is traditionally the first-order objective, hardware design must explore the complex trade-off among metrics such as performance, area, and timing. As such, an HLS benchmark must predictably reflect the influence of hardware-specific optimizations and constraints and outline the resulting QoR at both the kernel and application levels. Given the evolving nature of HLS, it is also necessary to support and provide a fair comparison between different programming models. Motivated by the observation that existing HLS benchmarks such as CHStone [3] and MachSuite [5] fall short in addressing these challenges, Rosetta couples a range of realistic applications with real-world constraints to benchmark HLS algorithms with various hardware optimizations under different programming models.

2.1 Realistic Applications

Existing HLS benchmark suites provide a myriad of small kernels prevalent in literature. For instance, CHStone [3] has 12 kernels from only three main application domains, and thus fails to characterize a diverse set of workloads. MachSuite [5] surveys a wider range of domains but falls short of integrating kernels into realistic applications commonly implemented in the field. Both suites contain very few kernels with over 1000 lines of code and lack the complexity representative of designs implemented with state-of-the-arts HLS tools in academic and commercial environments. As seen in Table 1, Rosetta is comprised of a set of applications selected from domains where FPGAs have demonstrated effectiveness as a computing platform and can achieve orders of magnitude improvement in performance compared to general-purpose CPUs or GPUs. Each application has been used in large scale HLS-focused academic studies or commercial solutions, and is composed of key kernels found in existing benchmark suites.

2.2 Realistic Constraints

In addition to realistic applications, Rosetta includes a set of enforceable realistic system-level design constraints based on each application's real world specifications to model realistic use cases for FPGA-based hardware accelerators. Unlike traditional software profiling where execution time constitutes the key requirement, FPGA-based hardware designs must be synthesized to meet certain throughput, latency, clock frequency, and area constraints. By enforcing strict system-level constraints, Rosetta allows a standardized approach to study the effectiveness of various HLS optimizations in synthesizing realistic design points. For instance, Rosetta

Prospposed Initial Benchmarks for Rosetta

Application	Domain	Constraints	Kernels / Algorithms
Voice Removal and Pitch Shifting	Audio Processing	Latency (Real-Time)	FFT, Inverse FFT, DSP Filters
DNA and Protein Sequencing	Bioinformatics	Throughput	Smith Waterman
Advanced Encryption Standard	Cryptography	Throughput	Matrix Subsitution and Permutations
Monte Carlo Option-Pricing	Financial Analysis	Throughput / Latency	Black Scholes, Mersenne Twister, Box Muller
Digit Recognition	Machine Learning	Throughput	Population-Count, K-NN, K-Means
Convolutional Neural Networks	Machine Learning	Throughput	Convolution, Soft Max and Max Pool Layers
Face Detection	Video Processing	Throughput	Viola Jones Algorithm
Lane Detection	Video Processing	Latency (Real-Time)	Edge Detection

Table 1: Multi-dimensional benchmarks with realistic design constraints to be supported by Rosetta.

allows its users to evaluate the ability of an algorithm in achieving a desired throughput and analyze the corresponding impact in area and latency. Furthermore, Rosetta allows parametrization of designs to fine tune functional behaviors and workload patterns, and provides an extensive software and hardware functional verification infrastructure to ensure the correctness of designs from pre-synthesis to post-place-and-route. The verification framework includes high-level software implementations of each design that produce golden outputs for given input datasets, and are used to validate results during software and hardware simulations.

2.3 Exploring HLS Optimizations

Much of the source code from existing benchmark suites is not HLS tool friendly in the sense that adding hardware-specific directives requires tremendous user overhead in terms of understanding and refactoring, possibly a large portion of, the code. For example, pipelining functions, unrolling loops, and enforcing latency constraints are often deemed fruitless as the design is not amenable to such optimizations without significant code modification. On the contrary, Rosetta supports HLS optimizations directives by providing multiple implementations of each design. Starting from a baseline software implementation, Rosetta guides the user in the process of porting each application to popular commercial and academic HLS tools and adding directives to implement key hardware optimizations necessary to achieve a realistic design. To enable a more productive design flow, Rosetta aggregates synthesis results of various design points, which allows the users to more efficiently compare the QoR resulting from various optimizations.

2.4 Programming Models and Architectures

CHStone [3] and MachSuite [5] are both designed for sequential C-based HLS flows. As a result, existing benchmark suites are language-specific and not portable to other programming models. As reconfigurable hardware architectures become increasingly heterogeneous and programming environments for hardware accelerators become increasingly versatile, a single benchmark that supports various programming models is needed. In anticipation of this trend, Rosetta will support both C/C++ and OpenCL - two prominent models that are currently supported by academic and commercial HLS tools. A different programming model introduces new device and tool-specific issues that need to be considered by the synthesis algorithm. In the case of OpenCL for FPGA, additional optimizations may be necessary to hide communication latency and efficiently pipeline kernels to achieve parallel execution of work items [2].

References

[1] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang. High-Level Synthesis for FPGAs: From Prototyping to Deployment. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 30(4):473–491, 2011.

[2] T. S. Czajkowski, D. Neto, M. Kinsner, U. Aydonat, J. Wong, D. Denisenko, P. Yiannacouras, J. Freeman, D. P. Singh, and S. D. Brown. OpenCL for FPGAs: Prototyping a Compiler. *Int'l Conf. on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, pages 3–12, 2012.

[3] Y. Hara, H. Tomiyama, S. Honda, H. Takada, and K. Ishii. CHStone: A Benchmark Program Suite for Practical C-Based High-Level Synthesis. *Int'l Symp. on Circuits and Systems (ISCAS)*, 2008.

[4] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, et al. A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services. *Int'l Symp. on Computer Architecture (ISCA)*, pages 13–24, 2014.

[5] B. Reagen, R. Adolf, Y. S. Shao, G.-Y. Wei, and D. Brooks. MachSuite: Benchmarks for Accelerator Design and Customized Architectures. *IEEE Int'l Symp. on Workload Characterization (IISWC)*, pages 110–119, 2014.