# ParaFPGA 2017: Enlarging the scope of parallel programming with FPGAs

Erik H. D'HOLLANDER[a,1] and Abdellah TOUHAFI[b]
[a] *Ghent University, B-9000 Ghent, Belgium*
[b] *Vrije Universiteit Brussel, B-1050 Brussels, Belgium*

**Abstract.** The biennial mini-symposium "*Parallel computing with FPGAs*" brings together research on applications and tools fostering the use of field programmable gate arrays. Key aspects are the efficiency, programmability, scalability and portability of high-level synthesis languages and tools. In particular this year's contributions present productivity and programmability results of using HLS languages OpenCL, OmpSs, MATLAB/Octave, OpenSPL and Vivado HLS. The current state and future challenges of HLS within the FPGA landscape is covered in a special keynote on bridging the gap between software and hardware designers.

**Keywords.** High-level synthesis languages, field-programmable gate arrays, high-performance computing, hardware/software codesign.

## 1. Introduction

The biennial mini-symposium "parallel computing with FPGAs" focuses on applications and tools fostering the use of field programmable gate arrays. Since the last ParaFPGA event in 2015 we have seen a number of milestones which characterize the mainstream adoption of this technology in high-performance computing. First, the merger of Altera with Intel promises a much larger integration of the programmable logic within the conventional processor die. In particular the support for a uniformly shared address space, cache coherency and a fast low latency processor interconnect will stimulate the development of applications on and beyond the streaming paradigm [1]. Second, the rapid development cycle and the maintenance of the development platform will be enhanced by cloud services such as Amazon EC2 and others [2]. Third, after the design and exploration of a myriad of a high-level synthesis systems and tools, there is a growing support for a common language such as OpenCL both in academia and industry. While this bridges the language gap between programming a GPU or an FPGA accelerator, performance optimizations for both accelerators are vastly different [3].

## 2. Exploring HLS languages

Two contributions of this mini-symposium use OpenCL for programming FPGAs. Paulino et al. [4] converted the domain specific language, MATLAB/Octave, into

[1] Corresponding Author.

OpenCL and applied five optimizations to create or improve pipelining, vectorization, function inlining and memory partitioning. Hosseinabady et al. [5] show how to optimize a memory-bound application, the calculation of an image histogram, by systematically reducing the initiation interval, a performance metric of pipelined loops. The authors also illustrate that irregular memory access and the ensuing load distribution favors the use of FPGAs over GPUs.

Depending on the hardware environment or the application, other software programming approaches may become useful and productive. A case in point arises when an application is targeted for a GPU, an FPGA or a combination of both accelerators. The programming environment OmpSs supports both GPU and FPGA kernels. In [6] a blocked matrix multiplication is targeted both for GPUs and FPGAs using the OpenMP pragma “target device”. The kernel of the algorithm is generated for the FPGA using the Vivado HLS compiler and for the GPU using CUDA cores. The advantage of this approach is portability and scalability with minimal software modification.

In the I/O data path as well as in the core of an application low latency access to the data is of paramount importance for the performance. Most HLS languages present pragmas for the organization of parallel memory channels and a single cycle based streaming access of multidimensional arrays. For less common access patterns, the parallel data access may be automated by using the spatial programming paradigm of OpenSPL [7], a language which separates the control and data paths of the application. In Janson et al. [8], access to the data stencil of the Wilson Dirac operator has been described using the data flow operators of OpenSPL. The result compares favorably with existing implementations based on OpenCL.

## 3. Conclusion

The contributions of the mini-symposium reflect the state of the art in parallel programming with FPGAs. Despite the manifest progress, several stumbling blocks but also exciting challenges lie ahead on the road to this maturing field of research. These involve not only improving the programmability and resource usage of today's FPGAs, but also taking into account the implications of emerging technologies. Christian Pilato presents a comprehensive overview of the present status, open challenges and future research topics in the keynote paper “Bridging the Gap between Software and Hardware Designers using High-Level Synthesis” [9]. It is clear that the raised abstraction level in high-level synthesis should not obfuscate the hardware parameters which are essential to optimize the performance of the program being synthesized. Therefore the challenge remains to maintain a balance between managing the growing complexity of the memory hierarchy, the algorithmic core and the HLS productivity and performance.

## 4. Acknowledgement

The organizers of ParaFPGA 2017 thank the members of the program committee who provided timely and elaborated reviews helping to maintain the scope and quality of this symposium.

## References

[1] Young-kyu Choi, Jason Cong, Zhenman Fang, Yuchen Hao, Glenn Reinman, and Peng Wei, “A Quantitative Analysis on Microarchitectures of Modern CPU-FPGA Platforms,” in *Proceedings of the 53rd Annual Design Automation Conference*, Austin, TX, 2016, vol. Article 109, 6 pages.

[2] N. Tarafdar, N. Eskandari, T. Lin, and P. Chow, “Designing for FPGAs in the Cloud,” *IEEE Design Test*, p. accepted for publication, 6 pages, Sep. 2017.

[3] Z. Wang, B. He, W. Zhang, and S. Jiang, “A performance analysis framework for optimizing OpenCL applications on FPGAs,” in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016, pp. 114–125.

[4] Nuno Paulino, Luís Reis, and João M.P. Cardoso, “On Coding Techniques for Targeting FPGAs via OpenCL,” in *Proceedings of the conference Parallel Computing 2017*, Bologna, 2017.

[5] Mohammad Hosseinabady and Jose Nunez-Yanez, “Pipelined Streaming Computation of Histogram in FPGA OpenCL,” in *Proceedings of the conference Parallel Computing 2017*, Bologna, 2017.

[6] Ying Hao Xu Lin *et al.*, “Implementation of the K-means algorithm on heterogeneous devices: a use case based on an industrial dataset,” in *Proceedings of the conference Parallel Computing 2017*, Bologna, 2017.

[7] T. Becker, O. Mencer, and G. Gaydadjiev, “Spatial Programming with OpenSPL,” in *FPGAs for Software Programmers*, D. Koch, F. Hannig, and D. Ziener, Eds. Cham: Springer International Publishing, 2016, pp. 81–95.

[8] Thomas Janson and Udo Kebschull, “Highly Parallel Lattice QCD Wilson Dirac Operator with FPGAs,” in *Proceedings of the conference Parallel Computing 2017*, Bologna, 2017.

[9] Christian Pilato, “Bridging the Gap between Software and Hardware Designers,” in *Proceedings of the conference Parallel Computing 2017*, Bologna, 2017.