High Level Programming of FPGAs for HPC and Data Centric Applications

Oren Segal, Nasibeh Nasiri, Martin Margala
Department of Electrical and Computer Engineering
University of Massachusetts Lowell
Lowell, USA
Oren_Segal/Nasibeh_Nasiri@ student.uml.edu
Martin_Margala@uml.edu

Wim Vanderbauwhede
School of Computing Science
University of Glasgow
Glasgow, UK
Wim.Vanderbauwhede@glasgow.ac.uk

***Abstract*—Heterogeneous computing offers a promising solution for high performance and energy efficient computing. Until recently the high performance heterogeneous computing arena was dominated by discrete GPUs but in recent years, new solutions based on devices such as APUs and FPGAs have emerged. These new solutions show promise for further improvements in energy efficiency. FPGA based heterogeneous computing is an especially promising direction since it allows for the creation of custom hardware solutions for data centric parallel applications. One of the main issues delaying wide spread adoption of FPGAs as main stream high performance computing devices is the difficulty in programming them. Altera's OpenCL implementation for FPGAs provides a high level of abstraction and increased ease of programmability of FPGAs. Two high performance computing applications (Lava Molecular Dynamics and Nearest-Neighbours) and a data centric application (Document Classification) were compiled using Altera's OpenCL compiler and programmed on a Nallatech FPGA board. Hardware utilization, kernel execution time and total execution time are reported. Up to 5.3x, 4.3x and 1.3x speed up over the Dual Xeon processor implementations was achieved respectively for LavaMD, Nearest-Neighbours and Document Classification.**

Keywords—heterogeneous computing; FPGA; OpenCL; high performance computing (HPC);

I. INTRODUCTION

Today's top ten energy efficient super computers [1] are all heterogeneous computing systems. While these systems currently employ GPUs as heterogeneous accelerators, recent research is suggesting that FPGAs [2] can be used as more efficient accelerators in some areas of high performance computing. One main drawback to using FPGAs is the difficulty in programming them [3]. The traditional way to program FPGAs has been through the use of hardware description languages (HDLs) such as Verilog and VHDL; Languages which require technical abilities and know-how not found in the average computer programmer [6]. A number of higher-level solutions, commonly known as "C-to-Gates", have been proposed and commercialized [10]. However, these approaches suffer from being proprietary and vendor-specific, as well as generally lacking in an architectural abstraction framework.

The emerging open programming standard for heterogeneous computing is OpenCL [4]. OpenCL offers a unified C programming model for any device that adheres to its standards. In OpenCL a C based kernel is replicated and run in parallel on multiple hardware compute units. Each parallel run is assigned an ID which allows the kernel to work on a subset of the data that is associated with it. Recently Altera Corp released an OpenCL SDK for FPGAs [5] allowing FPGAs to be treated as OpenCL based accelerators. The hardware design is extracted automatically from the OpenCL kernel by Altera's OpenCL compiler. A prominent feature of Altera's OpenCL compiler is the pipeline parallelism. Each pipeline can be fed from external memories. Local memory accesses are connected through a specialized interconnect structure to on-chip M9K RAMs. This compiler can create the required interfaces to the external and internal memories. Thus when designing a system targeting FPGAs one can benefit from the high abstraction offered by Altera's OpenCL compiler and avoid dealing with the low level complexity of hardware [7]. FPGAs have great potential for high performance computing applications and data centric parallel applications since they are low power devices compared to the multi core CPUs and GPUs commonly used in HPC applications, they offer a massive amount of fine grained parallelism and a significant amount of internal memory bandwidth. The rest of the paper is organized as follows. In section II we discuss the applications that were implemented on FPGA. In section III we elaborate on experiment design. Section IV discusses the results, section V discusses preliminary experiences with OpenCL on FPGAs and finally section VI and VII offer conclusions and future directions.

II. APPLICATIONS

A. High Performance Computing

Two algorithms from the Rodinia benchmark suite for heterogeneous computing [9] were implemented on FPGA using Altera's OpenCL compiler: the Nearest Neighbor algorithm from the Data Mining domain and the Lava Molecular Dynamic algorithm from the Molecular Dynamics domain.

The Nearest Neighbor application (NN) finds the k-nearest neighbors from an unstructured data set. The sequential NN algorithm reads in one record at a time, calculates the Euclidean distance from the target latitude and longitude, and evaluates the k nearest neighbors. The parallel versions read in many records at a time, execute the distance calculation on

978-1-4799-6233-4/14/$31.00

multiple threads, and the master thread updates the list of nearest neighbors [8].

The Lava Molecular Dynamic application (LavaMD) calculates particle potential and relocation due to mutual forces between particles within a large 3D space. This space is divided into cubes, or large boxes, that are allocated to individual cluster nodes. The large box at each node is further divided into cubes, called boxes. 26 neighbor boxes surround each box. Home boxes at the boundaries of the particle space have fewer neighbors. Particles only interact with those other particles that are within a cutoff radius since ones at larger distances exert negligible forces [8].

B. Data Centric Applications

The two HPC applications are representative for a large class of scientific computations. As an exemplar of a data-centric application, we implemented a streaming document classification algorithm. This application classifies a stream of HTML (or email) documents in "relevant" or "not relevant", where relevance is determined by a profile of keywords provided by the user. The stream of HTML documents is parsed, compressed and scored on a document-by-document basis. We create bigrams and trigrams (groups of two and three words) to improve the accuracy of filtering.

III. Experiment Design

An HP DL180 G6 with dual Intel Xeon L5630 2.13GHz processors and 144GB DDR3, 1333MHz RAM, was chosen as the host system. The host system was equipped with a Nallatech PCIe-385N A7 FPGA board(Fig 7.) with 8GB RAM connected through a second generation PCIe connection to the host system. The system was running Linux Centos 6.4 and Altera SDK for OpenCL, Version 13.0sp1 Build 234 was installed on the system in order to be able to communicate and program the FPGA device.

IV. Results

A. Hardware Utilization

Hardware utilization is reported in Table I, the highest utilization occurs in LavaMD. All three algorithms were auto optimized by the Altera OpenCL compiler. In addition manual optimizations were done as well in order to extract better performance from the kernel when running on FPGA. Note that the Document Classification program has relatively low DSP utilization.

B. Speed Analysis

Kernel and total execution times of the applications are shown in Table II. Kernel speed results indicate an increase of up to 5.3X on Nearest Neighbor, up to 4.3X on LavaMD and about 1.3X on Document Classification when compared to the dual Xeon processor on the HP server. The total execution time of NN and LavaMD on the FPGA is 1.6X and 4.2X faster compared to that of running on the CPU. Relative speedup can be seen in Fig 1. The throughput of the Document Classification Application running on the CPU is 312MB/s while on the FPGA it is 324MB/s.

C. Power Analysis

Average power consumption while running in CPU mode is 211W (with the FPGA attached) compared to 190W when running in FPGA mode, but a more accurate power estimate should be done with a system with no FPGA attached. The inclusion of the FPGA device in the system adds an additional 24.7W; hence the true average power consumption on a fully utilized kernel without an FPGA device attached to the system is 186.3W. Fig 2 shows the kernel power savings and total power savings for each of the three applications.

TABLE I. Hardware utilization of the Nallatech FPGA board

	Document Classification	lavaMD	Nearest Neighbor
Logic utilization	68%	79%	74%
Dedicated logic reg.	33%	39%	32%
Memory blocks	75%	71%	60%
DSP blocks	13%	63%	50%

TABLE II. CPU Vs FPGA Execution Times

	Device	Kernel execution time in seconds	Total execution time in seconds
Nearest neighbour	CPU	0.000294	0.000712
Nearest neighbour	FPGA	0.000055	0.000444
Lava MD	CPU	849.552612	852.589172
Lava MD	FPGA	196.790756	201.447555
Document Classification	CPU	6.809791	7.547648
Document Classification	FPGA	5.212152	7.273355

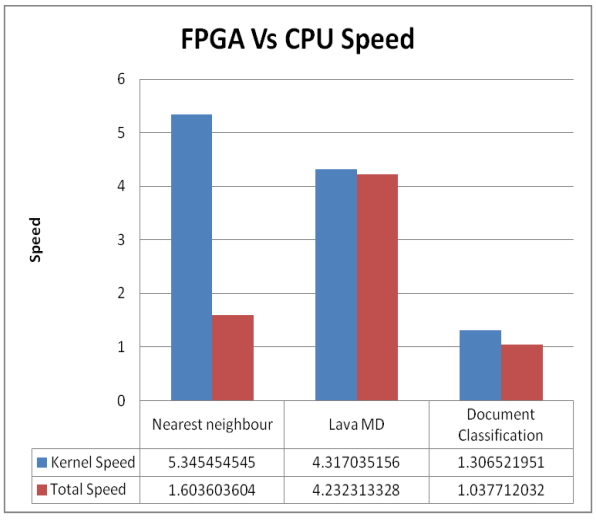

Fig.1. The relative speed of the FPGA implementation Vs the CPU implementation.

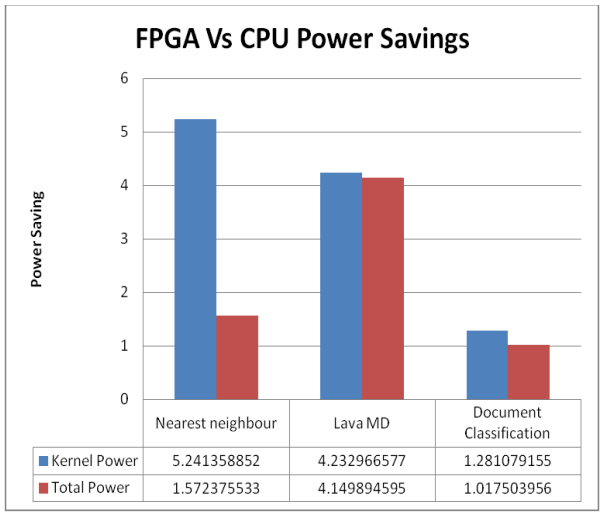

Fig.2. The Power Savings of the FPGA implementation relative to the CPU implementation.

V. OpenCL on FPGAs Perlimanary Discussion

The ability to compile arbitrary OpenCL code is important for portability and cross platform optimizations. The majority of the OpenCL code we ported to the FPGA platform compiles without any major issues, but in the case of the Document Classification application we planned to use a Bloom filter to reject terms not present in the profile. The Bloom filter could potentially improve the performance of the FPGA version since it is stored in the on-chip memory and determines the membership of a term in the profile. Only if the Bloom filter returns a hit, the term will be looked up in the profile memory residing in off-chip memory. The hit rate of the Bloom filter is proportional to the size of the profile. Unfortunately version 13 of the Altera OpenCL compiler could not successfully compile the bloom filter version that we created, but the issue was reported and is being addressed by Altera. We are told that future versions of the compiler will resolve this issue. The Nearest neighbor algorithm shows the best overall performance and power savings on FPGA and can be related to the fact that it is a relatively simple kernel with a large computation to data ratio that can be mapped very well to the FPGA fine grained parallelism.

VI. Conclusions

Our experiments show that FPGAs should be considered as high performance computing and datacenter hardware candidates in heterogeneous systems, capable to compete with CPUs and GPUs not only in terms of performance per Watt but also in ease of programmability and level of abstraction, but further research is required in order to better understand the strengths and weaknesses of FPGAs running such OpenCL based algorithms.

VII. Future Directions

We plan to continue to analyze the performance of OpenCL based algorithms on the FPGA platform and conduct comparative performance studies in order to find the areas of computation where FPGAs should be utilized in order to achieve higher performance and power savings.

VIII. Acknowledgments

We would like to thank HP, Nallatech and Altera for their generous hardware and software donations. We would also like to thank Sai Rahul Chalamalasetti and Mitch Wright from the HP Server Group for their contributions to this paper.

Fig 3. Nallatech PCIE-385n A7 Altera Stratix V board

References

[1] The Green500 List (11/2013). http://www.green500.org/news/green500-list-november-2013.

[2] L. Gan; H. Fu; Luk, W.; C. Yang; W. Xue; X. Huang; Y. Zhang; G. Yang, "Accelerating solvers for global atmospheric equations through mixed-precision data flow engine," Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on , vol., no., pp.1,6, 2-4 Sept. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] David F. Bacon, Rodric Rabbah, and Sunil Shukla. 2013. FPGA programming for the masses. Commun. ACM 56, 4 (April 2013), 56-63.

[4] Khronos OpenCL Working Group. "OpenCL-The open standard for parallel programming of heterogeneous systems." On line] http://www.khronos.org/opencl (2011).

[5] Altera SDK for OpenCL. http://www.altera.com/literature/lit-opencl-sdk.jsp

[6] TIOBE Index. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[7] Chen, D.; Singh, D., "Invited paper: Using OpenCL to evaluate the efficiency of CPUS, GPUS and FPGAS for information filtering," Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on , vol., no., pp.5,12, 29-31 Aug. 2012

[8] Shuai Che; Boyer, M.; Jiayuan Meng; Tarjan, D.; Sheaffer, J.W.; Sang-Ha Lee; Skadron, K., "Rodinia: A benchmark suite for heterogeneous computing," Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on , vol., no., pp.44,54, 4-6 Oct. 2009

[9] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron. Rodinia: A Benchmark Suite for Heterogeneous Computing. In Proceedings of the IEEE International Symposium on Workload Characterization (IISWC), pp. 44-54, Oct. 2009.

[10] C. Sullivan, A.Wilson, and S. Chappell, “Using C based logic synthesis to bridge the productivity gap,” in Proceedings of the 2004 conference on Asia South Pacific design automation: electronic design and solution fair. IEEE Press Piscataway, NJ, USA, 2004, pp. 349–354.