CMST SI(1) 63-69 (2010) DOI:10.12921/cmst.2010.SI.01.63-69

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **Special Issue 2010,** 63-69

Utilization of FPGA Architectures for High Performance Computations

Agnieszka Dąbrowska-Boruch[1,2], Ernest Jamro[1,2], Marcin Janiszewski[2], Dawid Kuna[2], Krzysztof Machaczek[2], Paweł Russek[1,2], Kazimierz Wiatr[1,2], Maciej Wielgosz[1,2]

[1]*Department of Electronics AGH, al. Mickiewicza 30, 30-059 Kraków, Poland*
[2]*ACC Cyfronet AGH, Nawojka 11, 30-950 Kraków, Poland*
e-mail: {russek/jamro/wiatr/wielgosz}@agh.edu.pl; {Dawid.Kuna/Machaczek}@cyfronet.kraków.pl

(Received: 20 August 2010; revised: 20 October 2010; published online: 23 November 2010)

Abstract: The primary intention of this paper is to present the results of several cases where the FPGA technology was used for high performance calculations. We gathered applications that had been developed over the last couple of years. Over this period of time we observed that there had been a rapid growth of interest in the utilization of FPGA for HPC. Basing on our expertise we give selected metrics, results and conclusions which, in our opinion, are important for anyone who is interested in FPGA as an alternative for faster computations. A brief description of the characteristics of FPGA and FPGA usage for acceleration are also included for novices on the subject.

Key words: custom computing, dedicated architectures, hardware acceleration, reconfigurable computing

I. INTRODUCTION

The FPGA technology combines semiconductor technology with design technology. In principle, FPGA circuits were conceived to allow electronics designers to implement such digital circuits as processors, communication devices, peripherals, etc. Each digital device consists of logic gates and memory elements which are connected by wires. The architecture of FPGA offers logic and memory together with programmable connections. Programmability is an essential feature of FPGA technology. It has several distinctive features. The most important are on-site programmability and reconfiguration ability. On-site programmability means that the structure of an FPGA chip can be programmed by an end user. This is usually done by the downloading of a proper configuration file. Reconfiguration means that a programmed FPGA chip can be erased and programmed with a different configuration. Thanks to FPGA, we achieve the ability to implement highly integrated, custom designed digital devices with no need to employ Full Custom semiconductor technology. Compared to FPGA, Full Custom technologies are extremely expensive in terms of NRE (Non Return Engineering) costs. An NRE cost is the cost of the very first semiconductor device. The production of the following devices is very cheap in Full Custom technology. This is not the case for FPGA where the cost per chip is stable despite the number of released circuits. The cost decreases for Full Custom chips as the number of devices increases. The technology chosen for CPU processors is Full Custom because the volume of production is usually high enough to make up for the NRE cost. There is the penalty paid for the reconfiguration ability of FPGA that suffers from lower performance and gate capacity when compared to the corresponding Full Custom chips.

A hardware accelerator is a co-processor which is capable of performing a certain task better than a CPU does. Better means faster (performance), cheaper (power) or more precisely (computation accuracy). Graphics processors may be common examples. It is also worth noting that some hardware accelerators have been integrated into CPU: math co-processor and SSE feature. Despite the above-mentioned standard hardware accelerators, it is also plausible to use custom designed accelerators to leverage tasks that are characteristic for a particular user. Unfortunately, it would be very expensive to differentiate a co-processor according to individual user needs. The above cost could be reduced if FPGA is used to implement the necessary hardware accelerator.

What distinguishes a hardware processor (accelerator) from the CPU? In general, its internal structure fits the task it was designed to perform. The principal advantages of custom architectures are:

- The size of registers fit the algorithm's data representation.
- A data path is designed to exploit algorithm parallelism.
- The number and the type of execution units are optimized.

As a result, we achieve compact architecture which is fully exploited during algorithm execution, i.e. the number of futile clock cycles for implemented operators is minimized. If performance is the principle goal, the compaction of the architecture also leads to more execution units that are implemented to execute algorithms in parallel.

In HPC, hardware acceleration is also present. The MDGRAPE project is a good example [1]. MDGRAPE is the name of a PC acceleration board designed for efficient Molecular Dynamics calculation. It is also the name of an ultra-high performance petascale supercomputer system that was developed by the RIKEN research institute in Japan. It utilizes the MDGRAPE accelerator boards. In this case, the accelerator chips were manufactured in 130 nm Full Custom technology.

II. METHODS

The FPGA technology can also be applied to develop hardware accelerators for HPC. This idea has been particularly popular for the last few years. FPGA circuits have been used as accelerators since the first chips were offered. Recently, their capacity (number of transistors) allowed them to face the challenges of HPC. What distinguishes the HPC solutions from other applications is the need to compete against state-of-the-art CPUs. In this race performance is the key measure. What also matters is the performance/power factor. As FPGAs became popular, providers of the HPC systems integrated reconfigurable chips into their computational solutions. One of the first available was the Cray XD. Another distinct solution is the SGI RASC RC100 [2]

The method of design and design automation tools is an important element of all design technology. Accelerating co-processors implemented in FPGAs are a hardware subsystem. That is why the hardware's design methods must be applied. Currently, Hardware Description Languages (HDLs) are the most popular method of describing digital hardware systems. HDLs allow for modeling, verification and automated synthesis of the digital designs. VHDL and Verilog are the most popular HDLs. Unfortunately, the process of hardware planning takes a long time and is laborious. An alternative way to obtain a hardware accelerator is to take advantages of the High Level Languages (HLLs). This method significantly speeds up the creation of the accelerators because it takes care of the low level details of the hardware architecture. These details are, for example, the data representation, the signal timings, the communication protocols, etc. Mitrion-C [3] and Impulse-C [4] are examples of the most popular HLLs which are used for the HPC acceleration. Unfortunately, the high level of design abstraction leads to the lower performance of the final design.

In this paper we present the acceleration results when the various FPGA based hardware accelerators were used. We studied the algorithms executed in the different fields of the HPC. We compared the achieved speed-up against the factors which in our opinion had the strongest influence on the successful use of the FPGAs. All the presented solutions run on the SGI's Altix 4700 system that is equipped with the RASC platform. This machine is publicly available at ACC Cyfronet AGH at rasc.cyf-kr.edu.pl.

We designed in both VHDL and Mitrion-C. Our algorithms required different data representation. The researched problems were also of different computational complexity. The type of data dependency in the algorithms was different as well.

The choice between an HDL and an HLL is a tradeoff between the time required to develop a prototype and the performance of the accelerator. In our opinion an HLL is a quick start solution. It can be used in the prototyping phase of software to the hardware migration. When good results are achieved for a HLL, a consecutive step should be made the HDL of the accelerator. Production solutions should always originate in HDL.

The type of data representation of variables is the key to successful hardware acceleration. As CPUs have a typically 64-bit floating-point data representation, it is possible to gain some advantages if it is possible to reduce the size and the complexity of used data. For example, integer type adders and multipliers require less logical resources than their floating-point counterparts.

Transferring data to and from the accelerator is an important issue. From this point of view, a very important issue is the computational complexity. For algorithms with complexities lower than $O(N^2)$ it may occur that the speed-up of an accelerator can not be exploited because of an insufficient amount of delivered data. In this case the co-processor waits instead of processing. The low complexity algorithms are referred to as bandwidth limited. To accelerate hardware, we looked for computationally limited problems.

The type of data dependency limits parallel execution of the algorithms. If there is a lot of data dependency, neither spatial nor temporal parallelism is possible. In this case we lose the main source of acceleration that could be introduced.

III. RESULTS

The following algorithms were implemented in hardware (the names of the accelerators are given in parentheses).

1. Gaussian-Type Orbitals calculations (GTO) [5],
2. Modular Exponentiation function (MONT) [6],
3. Bloom Filtering (BLOOM) [7],
4. Merge Sorting algorithm (MERGE) [8],
5. DGEMM function (DGEMM) [9].

Detailed descriptions of the co-processors are given elsewhere and the appropriate references are included. We only put the abridged characteristics necessary for the purpose of comparison here.

The Gaussian-Type Orbitals function is used for modeling electron orbitals in the quantum chemistry calculation. It is expressed by the formula (1).

$$\chi_{klm}(\mathbf{r}) = r_x^k r_y^l r_z^m \sum_i C_i e^{-\alpha_i \mathbf{r}^2} \tag{1}$$

Its role in computational methods can be found in [10]. Modular Exponentiation is an operation widely used in cryptography. It is suited to security applications thanks to a one way property. This means that the modular exponentiation is easy to compute but the reverse operation (logarithm) is very computationally exhaustive. This property is exploited in cryptography, for example in the RSA algorithm. The modular exponentiation is presented by Equation 2.

$$\mathrm{MontExp}(a) = a^{\theta} \bmod m \tag{2}$$

A detailed description of the Montgomery Exponentiation is presented in [11].

Bloom filtering is a method for a fast word search. It is very popular in algorithms where a high efficiency match operation is necessary. This method was conceived by Bloom [12] and it has been utilized in many applications so far.

Merge Sorting together with other sorting algorithms is frequently utilized in solutions where high efficiency of sorting is necessary. Its simplicity makes it a very good candidate for hardware and software implementations. The algorithm has been described in [13]. It belongs to a data mining class of algorithms.

The DGEMM is a function which is a part of the BLAS library. It performs matrix multiplications of tables with double precision coefficients' representation.

The results of the implementations of the above algorithms in hardware are summarized in Table 1. We included parameters that we believe are most important from the point of view of the hardware acceleration technique. We took into account such factors as the design tool that was used for the hardware creation, the type of data representation utilized by an algorithm, and the existence of data dependency in an algorithm's flow. The result achieved is the performance of an accelerator.

The data type is the size and the representation of input, output and temporal variables used in the algorithm. It decides on the width of operator units and its complexity. Data dependency gives us information about the corresponding algorithm and if it has been performed in a full pipeline manner. If there is no data dependency in the algorithm and the series of input data is processed, it is possible to build hardware architecture that has the ability to perform all consecutive operations in parallel. This is also called temporal parallelism.

Given in Table 1, performance is a measurement of computational power of a processor. In our case, it gives the number of operations per second that are performed during an algorithm execution. It is not peak but rather a sustained performance measure as we calculated the number of useful operations per second (OPS) performed by the hardware processor (not the number of operator units implemented in the architecture). To achieve an even better objectivity, we normalized the performed operations to 64-bits, i.e. an 8 bit addition is treated only as an eighth of a performed operation. Operations such as additions, multiplications, Look-up Tables, logical operations and shift operations were taken into account. The performance of the hardware processors was sometimes limited by the Bandwidth(BW) limitations which were caused by the real hardware. Sometimes a processor can not achieve its full computational power due to bandwidth limitations. However, in the case of our projects the performance is not BW affected. This is thanks to the multi-buffering mode that is available on the RASC platform.

Table 1. Comparison of performances using different types of algorithms

Name of accelerator	Design tool	Data type	Data dependency	FPGA utilization	Clock Frequency	Performance
MONT	VHDL	INT1024bit	No	35%	200MHz	25 GOPS
BLOOM	VHDL	BYTE	No	60%	100 MHz	7,2 GOPS
DGEMM	VHDL	DOUBLE	Yes	50%	100 MHZ	2,4 GOPS
GTO	VHDL	DOUBLE	Yes	30%	100 MHz	1,7 GOPS
MERGE_HDL	VHDL	CHAR [16]	Yes	20%	100 MHz	200 MOPS
MERGE	Mitrion-C	CHAR [16]	Yes	20%	100 MHz	20 MOPS

The clock frequencies for the RASC designs could be 50, 100 or 200 MHz.

The highest performance was achieved on the MONT hardware processor. In order to provide sufficient security, all arguments in the above equation must have a sufficient bit length. Typically the numbers presented in Equation 2 i.e. *a*, *e*, and *m* consist of thousands of bits. For RSA cryptography the recommended key sizes are 1024 bits for corporate use, and 2048 bits for extremely valuable data. This gives the advantage to dedicated co-processors over the CPUs. In custom solutions operators can fit exact data representation. A processor with limited register sizes must split its work which introduces additional overhead. On the other hand, FPGA is capable of processing all bits simultaneously, which is an obvious advantage in this case.

A modular exponentiation operation is carried out by sequential modular multiplications. For example, for 1024 bits exponent *e* length 1024 modular squaring and 512 multiplications must be performed on average with respect to the same modulus *m*. The most efficient way of performing this task is by using the Montgomery Multiplication algorithm, and the entire operation is called the Montgomery Exponentiation. Our implementation includes specific hardware optimizations which allow area optimizations and high clock frequency [6].

Very good results were achieved for the BLOOM hardware processor. This was possible because the Bloom filter has an ideal algorithm for acceleration using FPGA due to the data type representation, a large number of logical operations and trivial parallelism, which are necessary to perform the Bloom algorithm. Random logic operations are cumbersome for the CPUs and require several clock cycles. In contrast, for the FPGAs any logic function is the easiest operation to perform. Parallelism introduced here allows the FPGA's designers to efficiently use 60% of resources.

Implementation of the GTO function in FPGA is an argument against the opinion that the usage of FPGA produces no acceleration when the double-precision calculations are considered. We decomposed the double precision to integer calculations, and in this particular example it was possible to implement the exp() function which performs twice as well as the CPU implementation for the double precision argument. Additionally, the double precision adders, multipliers and MAC were implemented to perform the final function. It was also possible to neutralize the data dependency in the necessary series summation. This was possible thanks to the reduction of the full MAC operation to a single clock cycle.

Although the data type was FPGA-friendly in the case of the MERGE processor, it performed worst when compared to the other researched solutions. This is because neither the nature of the algorithm nor the design entry tool served to better the performance results. In this solution only one fifth of the implemented comparators worked at a single clock cycle. This leads to inefficiency in resource usage. Additionally, HLL used to shorten the design time contributed to inefficient results. We assumed that it had originated from the same idea, architecture implemented in HDL (we call it MERGE_HDL in Table 1) would perform even ten times better. It must be stated that even for such "poor" numbers, the MERGE sort hardware processor twice outperformed a relative CPU that was manufactured in the same semiconductor technology. We compared our solution to Itanium2, and the speedup was 0,49.

We would also like to point out 'FPGA utilization' parameters. One could ask why not use more resources and get higher performance. In the case of FPGA, it is not that straightforward because the higher resource utilization leads to lower design clock frequencies. As a result, the performance remains the same. For example, it is possible to put two GTO accelerators into a single FPGA structure but, according to experiments performed, it would be necessary to reduce the RASC's clock frequency to 50 MHz and the performance results would remain the same.

IV. DISCUSSION

Recently, the interest in FPGA for HPC calculations has been falling rapidly. It can easily be seen by looking at System Providers' offers. The number of available FPGA solutions is not very abundant at the moment. This refers to both hardware (platforms) and software (tools) solutions in so called High Performance Reconfigurable Computing (HPRC). Obviously, the hope that FPGA could revolutionize the paradigm of common computer architecture was too far reaching. The FPGAs are the only choices available today. At present we have multi-core processors and Graphics Processor Units (GPUs) that perform very well in HPC. It should be noted from the given examples that achieving a speed-up with FPGAs is not easy. Unfortunately, the same applies to the other techniques available today. In the authors' opinion, there is a place for each of these three technologies in HPC, including reconfigurable hardware.

There are several reasons that limit the potential of FPGA technology. The first and most important is that there is a lack of matured design tools. It is commonly stated that it is impossible to design a hardware accelerator without the knowledge of low level details of the design. It is expected that proper tools should make the design process fully automated. We believe that full automation and high level designing is not possible at all. If we want to take advantage of the optimization possibilities offered by FPGAs that are not present in CPUs and GPUs, we must optimize at low level hardware. We would like to state here that the tools' relative problems are in their imperfections. Compilation time is very long, and a successful result is not obvious. The implementation process often finishes with a note that the designer requirements can not be fulfilled.

The other issue is that high level languages, compilers, simulators and technology fitters are quite expensive if compared to standards established in software programming, where good quality tools are available free of charge. The cost of FPGAs chips are also substantial and this should be taken into consideration when a choice between FPGAs and GPUs is taken as an example.

The costs are directly linked to the popularity of a solution. Higher popularity means a reduction in the cost for an individual user. It is easier to become popular when there are no other alternatives. Naturally, designers choose the solutions they are familiar with even if they are slightly worse than other possible approaches. The FPGA technology needs time to become more widely known. This can be stimulated mainly by the introduction of reconfigurable computing courses at universities and computer science related faculties.

Together with multi-core processors and GPUs, reconfigurable logic can still be helpful to build state-of-the-art solutions for HPC. The role of the designer is to choose the correct technology for a particular problem. Security, data mining and life science are applications where FPGAs are still in use, as they are most successful in these areas.

References

[1] R. Susukita et al., *Hardware accelerator for molecular dynamics: MDGRAPE-2.* Computer Physics Communications 155 (2), 115131 (2003).

[2] SGI Corp. Sgi rasc rc100 blade. http://www.sgi.com/pdfs/3920.pdf/

[3] Mitrionics AB. Mitrion Users' Guide. http://www.mitrion.com/.

[4] Impulse-C Homepage. http://www.impulseaccelerated.com/

[5] M. Wielgosz, E. Jamro, P. Russek, K. Wiatr, *Hardware implementation of the orbital function for quantum chemistry calculations.* Applied Reconfigurable Computing, Springer-Verlag, LNCS 5992, 337-342. ARC'2010,

[6] M. Janiszewski, P. Russek, E. Jamro, K. Wiatr, *Implementation of Montgomery Exponentiation in FPGA for Cryptographic Applications*. KU KDM 2010, Zakopane, March 18–19, 2010, Abstracts.

[7] K. Machaczek, P. Russek, E. Jamro, K. Wiatr, *Realizacja szybkiego wyszukiwania wzorców w układach FPGA.* Pomiary, Automatyka, Kontrola, 54 (8) 540–542, Abstr.

[8] ACK Cyfronet. The Computing Acceleration Group Homepage. http://www.cyf-kr.edu.pl/en/?a=zao/ReconfigurableComputing.

[9] P. Russek, K. Wiatr, *Dedicated architecture for double precision matrix multiplication in supercomputing environment.* Proceedings of the 2007 IEEE workshop on Design and Diagnostics of Electronic Circuits and Systems, April 11–13, 2007, Cracow, Poland.

[10] W. Kohn, L.J. Sham, *Self-Consistent Equations Including Exchange and Correlation Effects*. Phys Rev A, 140, 1133 (1965).

[11] P.L. Montgomery, *Modular Multiplication without Trivial Division.* Mathematics of Computation. 26, 27, 519-521 (1985).

[12] B.H. Bloom, *Space/time trade-offs in hash coding with allowable errors*. Communications of the ACM, 13, 422-426 (1970).

[13] T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company (1989).

Agnieszka Dąbrowska-Boruch received a PhD degree in Electrical Engineering from the AGH University of Science and Technology in Kraków in 2007. Her research interests concern custom computing architectures, reconfigurable computing accelerators and digital design methods and tools. As a lecturer in the Department of Electronics of AGH, she provides lectures and laboratories in "Digital Design". She is an author or co-author of papers in professional journals and conference proceedings which regards reconfigurable logic.

Ernest Jamro received his master's degree in electronics engineering from the AGH University of Science and Technology (UST), Kraków PoM. Phil. 1996; M.Phil. degree from the University of Huddersfield (UPh. D.in 1997; PhD degree from the UST in 2001. He is currently a professor assistant in the Department of Electronics UST. His research interests include reconfigurable hardware (esp. Field Programmable Gate Arrays – FPGAs), reconfigurable computing systems, System on Chip design, artificial intelligence.

Marcin Janiszewski graduated from the Faculty of Electrical Engineering, Automation and Electronics AGH Kraków, Poland (2007). Currently, ACC CYFRONET AGH staff as a technical aide. Research interests cover issues in the field of cryptography (public-key cryptography), systems and applications based on reprogrammable devices using hardware description languages VHDL, Verilog.

Dawid Kuna is a PhD student in Computer Science at the AGH University of Science and Technology in Kraków. His current interest lies in using Graphics Processing Units for scientific computing.

Krzysztof Machaczek received an engineer degree in Electrical Engineering from the AGH University of Science and Metallurgy in Kraków in 2007. His research interests digital design and reconfigurable computing accelerators. He works in Computing Acceleration Group in Academic Computing Center "Cyfronet" AGH.

PAWEŁ RUSSEK received a PhD degree in Electrical Engineering from the AGH University of Science and Technology in Kraków in 2003. His research interests concern custom computing architectures, reconfigurable computing accelerators and digital design methods and tools. He is a Manager of Computing Acceleration Group in Academic Computing Center "Cyfronet" AGH. As a lecturer in the Department of Electronics of AGH, he provides lectures and laboratories in Embedded Systems and Programmable Logic Devices. He is an author or co-author of papers in professional journals and conference proceedings which regards reconfigurable logic.

KAZIMIERZ WIATR received the master's and PhD degrees in electrical engineering from the AGH University of Science and Technology, Kraków, Poland, in 1980 and 1987, respectively, and the D.Hab. degree in electronics from the University of Technology of Łódź in 1999. He received the professor degree in 2001. His research interests include design and performance of dedicated hardware structures and reconfigurable processors employing FPGAs for acceleration computing. He received 9 research grants from Polish Committee of Science Research. These works resulted in above 200 publications, including 3 books, the recent one: Acceleration Computing in Video Processing Systems. He is also an author of 5 patents and 35 industrial implementations. He was the reviewer of: IEEE Expert Magazine: Intelligent Systems, IEE Computer and Digital Techniques, IEE Electronic Letters, International Journal Eng. App. of Artificial Intelligence, IEEE Transactions on Neural Networks, Journal Machine Graphics and Vision, Eurasip Journal on Applied Signal Processing. He currently is a director of Academic Computing Centre CYFORNET AGH, and is a head of PIONIER council – Polish Optical Internet. Last but not least, he is a member of the Polish parliament (Senate), and a head of the Senate Science and Education Committee. www.kazimierzwiatr.pl

MACIEJ WIELGOSZ received his master's and PhD degrees from the AGH University of Science and Technology (Kraków, Poland) in 2005 and 2010, respectively. His research interests include parallel computing on FPGAs, image processing, neural networks. He is currently developing an FPGA-based accelerator of quantum chemistry computations.