Hardware implementation of the exponent based computational core for an exchange-correlation potential matrix generation

Maciej Wielgosz , Ernest Jamro, Kazimierz Wiatr

AGH University of Science and Technology,
al. Mickiewicza 30, 30-059 Krakow
ACK Cyfronet AGH
ul. Nawojki 11, 30-950 Krakow
{wielgosz/jamro/wiatr}@agh.edu.pl

Abstract. This paper presents an FPGA implementation of a calculation module for a finite sum of the exponential products (orbital function). The module is composed of several specially designed floating-point modules which are, fully pipelined and optimized for high speed performance. The hardware implementation revealed significant speed-up for the finite sum of the exponential products calculation ranging from 2.5x to 20x in comparison to the CPU. Orbital function is a computationally critical part of the Hartree-Fock algorithm. The presented approach aims to increase the performance of the part of the quantum chemistry computational system by employing FPGA-based accelerator. Several issues are addressed such as an identification of proper code fragments, porting a part of the Hartree-Fock algorithm to FPGA, data precision adjustment and data transfer overheads. The authors' intention was also to make hardware application of the orbital function universal and easily attachable to different systems.

Key words: HPRC (High Performance Reconfigurable Computing), FPGA, quantum chemistry, floating-point

1 Introduction

High performance computing has been recognized for many years as an area dominated only by microprocessors. The rapid increase of the logic resources of modern FPGAs (Field Programmable Gate Arrays) gives a huge opportunity to adopt hardware accelerators for calculations which involve processing a large volume of data. Such a vital rise in the computational capabilities of FPGAs has stimulated investigations to construct a reconfigurable hardware accelerator [1].

The proposed hardware module aims to speed up HPC chemistry and physics calculations and provide compatibility with the standard floating point data format. The RASC [2] platform has been chosen as a hardware accelerator because of the computational capabilities it offers for SMP supercomputers. Since achievable computation speed-ups strictly depends on the selected part of the algorithm, the choice of it becomes a critical issue. Therefore the finite sum of exp() functions was found a proper candidate for hardware acceleration. This operation is predominant in scientific computations, therefore the described solution may be employed in many kinds of applications (e.g. Quantum Monte Carlo)[3, 4].

However, the implementation described here should be considered as a first step in hardware exchange-correlation potential implementation. The ultimate goal is to design a hardware unit capable of generating an exchange-correlation potential matrix, which is one of the most computationally intensive routines within the Hartree-Fock calculations.

2 Algorithm

The time-independent Schroedinger equation for a system of N particles interacting via the Coulomb interaction is:

$$\hat{H}\psi = E\psi \tag{1}$$

$$\hat{H} = \sum_{i=1}^{N}(-\frac{\hbar^2}{2m_i}\nabla_i^2) + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{Z_iZ_j}{4\pi\epsilon_0\,|r_i - r_j|} \tag{2}$$

where: ψ is an N-body wavefunction, r denotes spatial positions and Z represents the charges of the individual particles. E denotes the energy of either the ground or an excited state of the system.

Quantum chemistry issues consist of a number of interacting electrons and ions. The total number of particles, N, is usually sufficiently large that an exact solution cannot be found. Controlled and well understood approximations are sought to reduce the complexity to a manageable level. Once the equations are solved, a large number of properties may be calculated from the wavefunction. Errors of approximations made in obtaining the wavefunction will be manifested in any property derived from the wave function. Where high accuracy is required, considerable attention must be paid to the derivation of the wavefunction and any approximations made. Therefore there has been careful investigation into what level of precision should be adopted in calculations to ensure satisfactory approximation.

The general procedure of solving the Hartree-Fock equations is to make the orbitals self-consistent with the potential field they generate. It is achieved through an iterative trial-and-error computational process, thus the entire procedure is called the self-consistent field (SCF) method.

The SCF procedure of solving the Hartree-Fock equation leads to the following set of eigenvalue equations in matrix formulation:

$$FC - SCE = 0 \tag{3}$$

F is the Fock-operator, C the matrix of the unknown coefficients, S the overlap matrix and E is the energy eigenvalues. All matrixes are of the same size.

The Hartree-Fock equation solving is an iterative process. The coefficients (corresponding to an electric field) are used to build the Fock-operator F, with which the system of linear equations is solved again to get a new solution (a new electric field). The procedure is repeated until the solution no longer changes.

The Fock operator depends on orbitals, which in turn are its eigenfunctions. Therefore orbitals have to be calculated for each iteration of the whole Hartree-Fock algorithm. That is why the finite sum of exponential products contributes scientifically to the overall performance of the application.

Orbital function is expressed by equation 4 [5]:

$$\chi_{klm}(r) = r_x^k r_y^l r_z^m \sum_i C_i N_i e^{-\alpha_i r^2} \tag{4}$$

The algorithm (equ. 4) was split into several sections [8] to allow adoption of a modular design approach. Consequently, a fully pipelined structure was obtained.

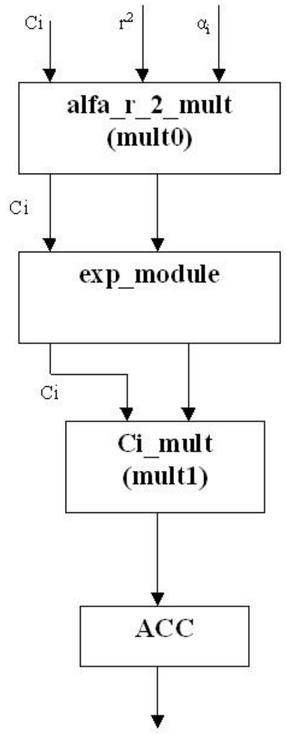

Fig. 1. EP module block diagram

Three different units have been employed within the EP module [Fig. 1]. All of them are specially designed, fully pipelined floating-point modules [6, 7] optimized to high speed performance. Input widths as well as their internal data path scales from single to double data precision. The input and output data format complies with the IEEE-754 floating-point standard. Nevertheless intermediate data representations employ different non-standard floating-point format in order to reduce hardware resources.

1. Multiplier
   For double precision data format inputs mantissa is 53-bit (54-bit including leading one) wide, therefore the product width is 108-bit wide. Such a bit-width is far beyond the required precision, therefore the LSBs of the product are usually disregarded [10]. Consequently the LSB's part of the multiplier performs operations which are not used in the next arithmetic operations. As a result, in the proposed architecture, some of the LSBs logic is not implemented at all. This approach allows for significant area reduction. A more detailed description of the multiplier will be available in a separate paper.
2. Exponential function
   This is mixed table-polynomial implementation of the exp function [6, 7], based on commonly known mathematical identities
   $$e^x = 2^x * log_2(e) = 2^{x_i} * e^{x - x_i / log_2 e} \tag{5}$$
   where x_i is an integer part of $x * log_2 e$.
3. Accumulator
   This is a fully pipelined mixed precision floating point unit. It can process one datum per clock cycle due to its advanced and optimized structure. A more detailed description of the accumulator will be available in a separate paper.

3 Platform

The Altix 4700 series [11] is a family of multiprocessor distributed shared memory (DSM) computer systems that currently ranges from 8 to 512 CPU sockets (up to 1,024 processor cores) and can accommodate up to 6TB of globally shared memory in a single system while delivering a teraflop of performance in a small-footprint rack. The RASC module communicates with the Altix system in the same manner as any CPU does, i.e. employing NUMALink interconnection. Data transfer between the FPGA and NUMALink bus is realized by an application-specific integrated circuit (ASIC) denoted as TIO.

SGI RASC RC100 Blade consists of two Virtex-4 LX 200 [9] FPGAs, with 40 MB of external SRAM logically organized as two 16MB blocks (as shown in Fig. 2) and an 8MB block. Each QDR SRAM block is capable of transferring 128-bit data every clock cycle (at 200MHz) both for reads and writes. The RC100 Blade is connected using the low latency NUMALink interconnect to the

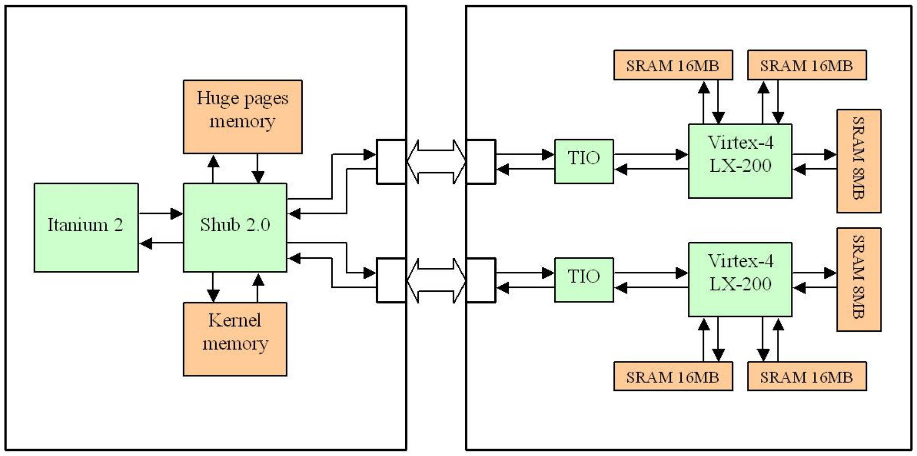

Fig. 2. RASC architecture and the host-fpga data link

SGI Altix 4700 Host System, for a rated peak bandwidth of 3.2GB per second each direction. The RASC algorithm execution procedure, from the processor's perspective, is composed of several functions which reserve resources, queue commands and perform other preparation steps. The resource reservation procedure, once conducted, allows many runnings of the algorithm - which amounts to a huge time savings, since the procedure takes approximately 7.5 ms. Therefore optimal structure of the hardware-accelerated application should contain as few initializations of the FPGA chips as possible so the FPGAs configuration time will be only a fraction of the algorithm execution time.

4 Implementation and acceleration

The EP module implemention results on the RASC [2] platform are presented below. Core services, provided by SGI together with the RASC module, is an extra logic incorporated in an FPGA which provides communication with the on-board hub. Units which the EP module is built of are strongly parametrized, therefore resources occupation varies depending on the data width (different calculation precision) of the module.

Implementation results	# 4-input LUT	# flip-flops	# 18-Kb BRAMs
EP module alone	2229 (1%)	1975(1%)	2(0.006%)
EP module with the core services	11560 (7%)	15922(9%)	25(7%)

Table 1. Implementation results of the EP module - single precison

Implementation results	# 4-input LUT	# flip-flops	# 18-Kb BRAMs
EP module alone	8684 (4.8%)	7891(4%)	6(0.02%)
EP module with the core services	18015 (10%)	21838(12%)	29(8%)

Table 2. Implementation results of the EP module - double precison

Implementation results	# 4-input LUT	# flip-flops	# 18-Kb BRAMs
Exp()	820 (0.5%)	920(0.5%)	2(0.006%)
Multiplier	549 (0.3%)	444(0.25%)	0
Accumulator	860 (0.5%)	605(0.4%)	0

Table 3. Implementation results of the elementary modules - single precision

Implementation results	# 4-input LUT	# flip-flops	# 18-Kb BRAMs
Exp()	5025 (3%)	5223 (3%)	6 (1.8%)
Multiplier	2316 (1%)	1850(1%)	0
Accumulator	1343 (0.5%)	818(0.4%)	0

Table 4. Implementation results of the elementary modules - double precison

Exp() function and the accumulator module consume the highest amount of resources, due to their complexity.

Module	Multiplier	Exp()	Accumulator	EP module
Pipeline delay [clk]	4	21	8	33

Table 5. Delays introduced by the EP module's components - single precision

Module	Multiplier	Exp()	Accumulator	EP module
Pipeline delay [clk]	5	30	10	45

Table 6. Delays introduced by the EP module's components - double precision

The overall pipeline latency of the exp module is 33 clock cycles for the single precison data format. The latency strongly depends on the width of input data.

It is worth underlining that all the modules have adjustable widths of data paths within the range from single to double precision. The Hartree-Fock algorithm is executed differently, depending on the set of molecules it is employed for. This, in turn impacts the precision requirements at different stages of computations. It may happen that for a certain chemical substance some sections of the algorithm process much more data than others. Besides, in the Hartree-Fock algorithm, the result precision increases (error decreases) with every algorithm iteration, therefore the calculation precision may be somehow scaled with the iteration number. Utilizing FPGAs in Hartree-Fock computations allows for the adjustment of data buses widths within the system so the proper precision is maintained across all the stages of the algorithm execution. The FPGA configuration time (7.5 ms) is significantly shorter than the overhead introduced by the oversized data format. It may appear that some sections of calculations can be successfully computed at single precision (or any other precision, e.g. 48-bits), saving a huge amount of resources that would otherwise be wasted if double precision data format was adopted. A hardware approach allows gradual switching from single to double data format together with the algorithm's rising demand for precision.

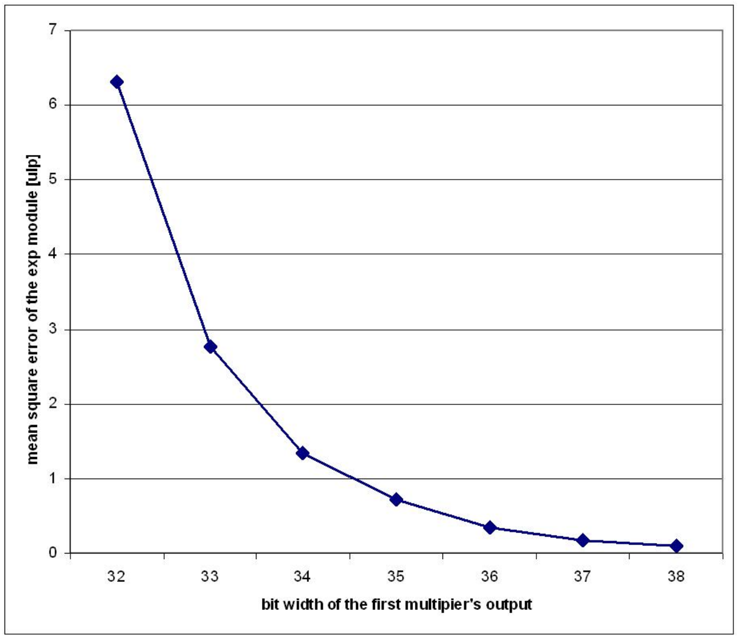

Fig. 3. Example of the precision adjustment within the EP module

All the data connections within the EP module are adjustable so it is possible to change them in order to meet precision requirements. Different approach can be taken to parameterize the EP module but the most common is a top-down method, which involves gradual modification of the inter module interface's width. Eventually number of guard bits of the basic components is established.

For instance (Fig. 3) optimal data width with regards to the precision of the interface between first multiplier and that exp() unit is 37 bits.

The EP module is intended to be a part of a larger system performing exchange-correlation potential calculation, so some preliminary tests were carried out in order to determine top achievable performance of the module on the RASC platform.

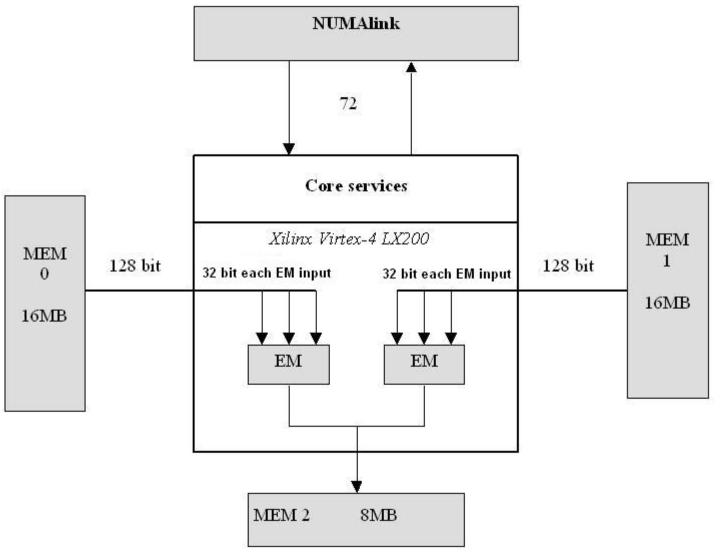

Fig. 4. Configuration of EP modules on the RASC platform

Each of the EP modules performs calculations in single precision so it is possible to aggregate two of them to increase overall throughput. Additionally due to the algorithm's structure some of the data (atom base coefficients) is used multiple times. So only a single 32-bit data chunk is streamed in and out the FPGA. This in turn allows to aggregate up to four EP module (single precision) on the single FPGA. This holds as the parallization degree is limited by the external memory data transfer rate rater than the FPGA resources. Single exponential operation calculation together with two-directional data transfer takes roughly 8 ns on the RASC platform whereas the same operation executed on the Itanium takes roughly 20 ns for a highly optimized code. Both the values were obtain as the result of a conducted tests.

Employing both Xilinx Virtex 4 chips available on the RASC platform leads 20x speed-up comparing to Itanium 2 1.6 GHz processor.

There is a huge difference between optimized and a simple code in execution time. Tables 8 and 9 present different degree of oprtimization applied to

Number of EP modules	[RASC/Itanium] speed-up
1	2.5
2	5
4	10

Table 7. Acceleration results

Itanium algorithm execution	Itanium a single operation execution time [us]	[RASC/Itanium] speed-up
Not optimized C code	0.2	25
Not optimized C code with compiler highest effort (-O3 compiler flag set)	0.16	14.6
Fully optimized C code	0.02	2.5

Table 8. Acceleration results as the results of the processor source code optimization

Not optimized C code	Optimized C code
`for(t=0;t<size;t++)` `result += a[t]*exp_table(b[t]*c[t]);`	`for(t=0;t<size;t++)` `b_c_table[t] = b[t]*c[t];` `for(t=0;t<size;t++)` `exp_table[t] = (b_c_table[t]);` `for(t=0;t<size;t++)` `result = result + a[t]*exp_table[t];`

Table 9. C implementation of the orbital calculation routine

the processor source code. What shows how such an acceleration evaluation can somethimes be misleading, especially when hardware performance is not compared with the optimized software one.

5 Summary

It is worth noting that a finite sum of the exponential products (eq.4) is, as a computational routine, ubiquitous in scientific calculations because of its universal function. Many different processes in the real world can be described by exponential function. Combination of the exp() leads to an even more uniform formula. Performance tests revealed that the FPGA is much faster than a processor - even with data stored in the processor's memory and the full optimization provided. It is worth taking into consideration that the Hartree-Fock algorithm, due to its iterative execution, allows the adoption of gradually adjustable data precision, which will give a speed boost to FPGAs in the final application. So there is still a huge potential in hardware implementation of quantum chemistry computational routines.

References

1. Underwood, K. D., Hemmert, K. S., and Ulmer, C.: Architectures and APIs: assessing requirements for delivering FPGA performance to applications. In Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (Tampa, Florida, November 11 - 17, 2006). SC '06. ACM, New York, NY, 111. DOI= `http://doi.acm.org/10.1145/1188455.1188571`
2. Silicon Graphics, Inc. Reconfigurable Application-Specific Computing User's Guide, Ver. 005, January 2007, SGI
3. Gothandaraman A., Peterson G., Warren G., Hinde R., Harrison R.:FPGA acceleration of a quantum Monte Carlo application. Parallel Computing 34(4-5): 278-291, 2008.
4. Gothandaraman, A., Warren, G. L., Peterson, G. D., and Harrison, R. J.: Reconfigurable accelerator for quantum Monte Carlo simulations in N-body systems. In Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (Tampa, Florida, November 11 - 17, 2006). SC '06. ACM, New York, NY, 177. DOI= `http://doi.acm.org/10.1145/1188455.1188638`
5. G.Mazur, M.Makowski: Development and Optimization of Computational Chemistry Algorithms, KDM'2008, March-2008, Zakopane, Poland
6. M.Wielgosz, E. Jamro, K. Wiatr, Highly Efficient Structure of 64-Bit Exponential Function Implemented in FPGAs, ARC 2008, Lecture notes in Springer-Verlag, London LNCS 4943, pp. 274 - 279
7. E. Jamro, M. Wielgosz, K. Wiatr, FPGA implementation of 64-bit exponential function for HPC,FPL Netherlands, 27-29 August 2007, FPL Proceedings
8. Omondi A.R., Computer Arithmetic Systems, Prentice Hall. Cambridge, 1994
9. Xilinx Virtex-4 Family Overview `http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf`
10. Parhi K. K., Chung J.G., Lee K.C., Cho K.J. Low-error fixed-width modified booth multiplier, US Patent: 957244.
11. SGI Altix 4700 `http://www.sgi.com/products/servers/altix/4000/`