Hardware implementation of the orbital function for quantum chemistry calculations

Maciej Wielgosz[1], Ernest Jamro[1], Pawel Russek[1], Grzegorz Mazur[2], Marcin Makowski[2] and Kazimierz Wiatr[1]

[1]AGH University of Science and Technology,
al. Mickiewicza 30, 30-059 Krakow
[1]ACK Cyfronet AGH
ul. Nawojki 11, 30-950 Krakow
{wielgosz/jamro/russek/wiatr}@agh.edu.pl
[2]Jagiellonian University, Faculty of Chemistry,
ul. Ingardena 3, 30-060 Krakow, Poland
{mazur/makowskm}@chemia.uj.edu.pl

Abstract. This paper presents FPGA acceleration and implementation results of the module for generating orbital function. The authors have implemented some of the computationally demanding part of the GPP quantum chemistry source code in FPGA. The orbital function core is composed of the authors' customized floating-point hardware modules. These modules are scalable from single to double precision, capable of working at frequency ranging from 100 to 200 MHz. Besides hardware implementation, the design process also involved reformulation of the algorithm in order to adapt them to the platform profile. The computational procedure presented in this paper is part of an algorithm for generating exchange-correlation potential, and is also recognized as one of the most computationally intensive routines. This feature justifies the effort devoted to develop its hardware implementation.

Key words: High Performance Reconfigurable Computing, FPGA, quantum chemistry, Custom Computing, HPC

1 Introduction

Field Programmable Gate Arrays (FPGAs) as a promising solution capable of delivering GFLOP/s of sustainable performance have been incorporated into many hardware accelerators. This approach has sparked a rising interest among researchers who find these platforms to be an attractive alternative solution to GPPs (General Purpose Processor). Coarse-grain parallelism has been exploited for many years, which has resulted in many implementations of different quantum chemistry algorithms on the SMP (Symmetric Multi-Processor) and cluster machines. At the same time, using FPGAs allows for a broad range of potential improvements with regards to computational time at the fine-grain level of algorithm complexity.

It is worth emphasizing that the precision of floating-point operations also becomes a primary concern when dealing with low-level quantum chemistry procedures, thus the authors have taken various measures to optimize them, both in terms of resource consumption and processing speed. These goals seem to be contradictory, but FPGA technology allows manipulation at the bit level which can be regarded as a huge advantage over GPP and GPU (Graphics Processing Unit) solutions which operate on data of fixed bit-length. The flexibility of the precison adjustment also justifies the choice of VHDL as the design language instead of one of the HLLs (High Level Language).

The proposed implementation employs the RASC platform, a detailed description of which, along with Altix system transfer modes, is covered in [1, 7].

2 Algorithm consideretion

One of the most common and the simplest approximation theories for solving the Scheodeinger equation is the Hartree-Fock algorithm. The general procedure for solving the Hartree-Fock equation is to make the orbitals self-consistent with the potential field they generate. This 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 for solving the Hartree-Fock equation leads to the following equation in matrix formulation:

$$FC - SCE = 0 \tag{1}$$

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

Solving the Hartree-Fock equation 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 a solution reaches a previously established level of accuracy.

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 orbital calculation presented in this paper contributes significantly to the overall performance of the application.

Orbital function is expressed by equation 4 [2]:

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

where r_x, r_y, r_z denote atom centered spatial coordinates of each point in the grid. C_x, C_y, C_z are normalization coefficients. The k, l, m indices depend on the type of atom shell (s, p, d or f). An atom base is represented by C_i and a_i coefficients.

3 Architecture of the orbital module

The implementation of the orbital generation function requires both designing hardware modules and software routines. To enable adoption of a modular design approach eq. 2 was decomposed into two sections - the exponential part (denoted as EP):

$$\chi_e(r) = \sum_i C_i N_i e^{-\alpha_i r^2} \tag{3}$$

and the polynomial part (PP):

$$\chi_p(r) = r_x^k r_y^l r_z^m C_n \tag{4}$$

Both of them are designed as separate units which make up the orbital function module. The polynomial part (PP) module generates coefficients for the forthcoming atom and at the same time evaluates orbital values for a currently processed atom. Such a pipeline approach requires an EP module to provide the exponential part computed in advance so the PP unit can sustain data processing. Both modules work independently to some degree - controlled by the Fine State Machine (FSM). It should be noted that the same set of polynomial coefficients can used several times for different EP results. This holds for all the orbitals within the same atom. On the other hand, calculating an EP result takes several clock cycles (one clock cycles per a sum iteration). Therefore for large atoms (composed of many shells), the calculation bottleneck is the EP module. Conversely, for a small one, the calculation bottleneck is in the PP module. As the size of an atom changes, in terms the number of shells, the load balance of the EP and PP modules shifts. A set of FIFO memories have been employed to avoid data transfer stalls between the units and to evenly distribute the load balance for different values.

The RASC accelerator is controlled by the host processor which handles hardware algorithm execution on the FPGA. Several routines are to be performed by the host processor in order to send data, launch the accelerator and fetch results afterwards. As the FPGA internal memory resources are limited, the maximum single computational data block is parameterized and its size can be adjusted within the range between 128 and 512 of the grid points. The maximum number of atoms composing the molecule is 32. Furthermore it is assumed that the number of atom base coefficients (C_i and a_i) does not exceed 64.

A uniform input data stream is well suited for FPGA implementation, but unfortunately this is not a case of the algorithm described in this paper. Quantum chemistry complexity is reflected by the diverse structure of input data which imposes some difficulties related to their efficient relocation. Thus a dedicated method of data formatting was introduced and implemented on the host processor to consolidate the data, which are subsequently sent over to the RASC accelerator.

Shell	# Orbital type (k,l,m)	# Normalization coefficient C_n	# Polynomial coefficient (PP)
s	1	1	1
p	r_x	1	r_x
p	r_y	1	r_y
p	r_z	1	r_z
d	r_x^2	0.33333	$0.33333r_x^2$
d	r_xr_y	1	r_xr_y
d	r_xr_z	1	r_xr_z
d	r_y^2	0.33333	$0.33333r_y^2$
d	r_yr_z	1	r_yr_z
d	r_z^2	0.33333	$0.33333r_z^2$
f	r_x^3	0.06667	$0.06667r_x^3$
f	$r_x^2r_y$	0.33333	$0.33333r_x^2r_y$
f	$r_x^2r_z$	0.33333	$0.33333r_x^2r_z$
f	$r_xr_y^2$	0.33333	$0.33333r_xr_y^2$
f	$r_xr_yr_z$	1	$r_xr_yr_z$
f	$r_xr_z^2$	0.33333	$0.33333r_xr_z^2$
f	r_y^3	0.06667	$0.06667r_y^3$
f	$r_y^2r_z$	0.33333	$0.33333r_y^2r_z$
f	$r_yr_z^2$	0.33333	$0.33333r_yr_z^2$
f	r_z^3	0.06667	$0.06667r_z^3$

Table 1. Orbital coefficients generated by the PP module

4 Implementation and acceleration results

Once the hardware module was implemented on the FPGA, several computational tests were conducted to compare the RASC performance with the Itanium 2 processor. Some of the tests were done for the water molecule calculated in the block composed of 512 point of the three dimensional grid. It took Itanium 2 processor roughly 2885 us to perform such an operation, which is close to the 3174 us consumed by FPGA. It is worth noting that the predominate shell of the water molecule is s. Due to the architecture of the hardware module, shell types have an impact on the overall performance. Consequently, an increase of the atom shell size allows full advantage to be taken of LUT and pipeline mechanisms implemented on the FPGA and the accelerated system starts to outperform GPP processor implementation, as depicted in the figures 2,3.

Figure 2 presents various speed-ups achieved for different atom shells dominating in the computations. As presented in table 1, the atom shell impacts the volume of computations which must be performed to obtain a single orbital result. Number of C_i and a_i coefficients also affects calculation effectiveness(Fig. 3). Discrepancy in the number of C_i and a_i coefficients and the type of atom shell is the main source of dynamic variation of load balance between the EP and PP modules. The ideal case would be an equal number of polynomial C_i and

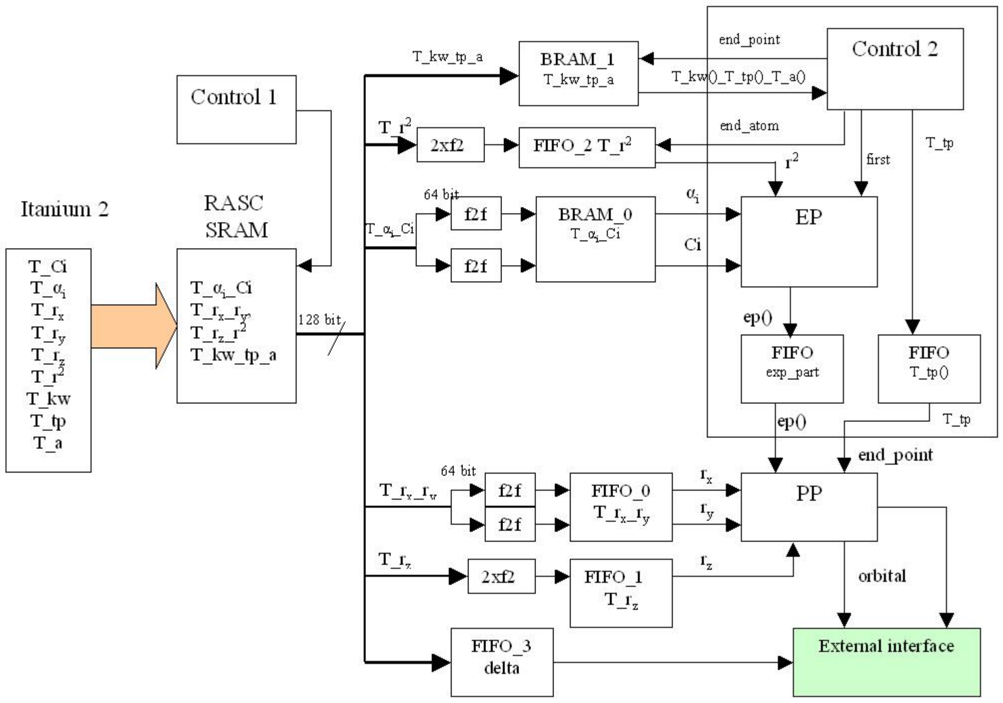

Fig. 1. Block diagram of the orbital generation module

a_i, but unfortunately, this rarely occurs in the computations. Therefore various buffering methods have been employed (Fig.1).

FPGA deliveres sustained performance while the processor efficiency drops with the growth of the moledule size. Presented results of a $3,5\times$ speed-up do not seem to be impressive but some enhacements to be implemented are expected to improve the performance of the system. The most important one is a top atom shell prediction mechanism which will eliminate the necessity of generating a complete set of polynomial coefficients for every atom. Only the orbitals which are used for the current calculation will be generated.

Implementation results	# 4-input LUT	# flip-flops	# 18-Kb BRAMs
Orbital module	8034 (4.5%)	7025 (3.4%)	14 (4.1%)
Orbital module + core services	17365 (9%)	20972 (11%)	37(11%)

Table 2. Implementation results of the Orbital module - single precison

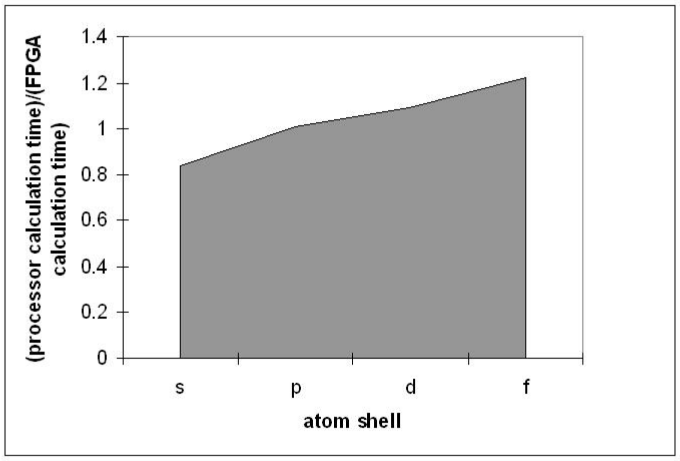

Fig. 2. Impact of the predominating shell on the speed-up (for the constant number of C_i and a_i coefficients = 1)

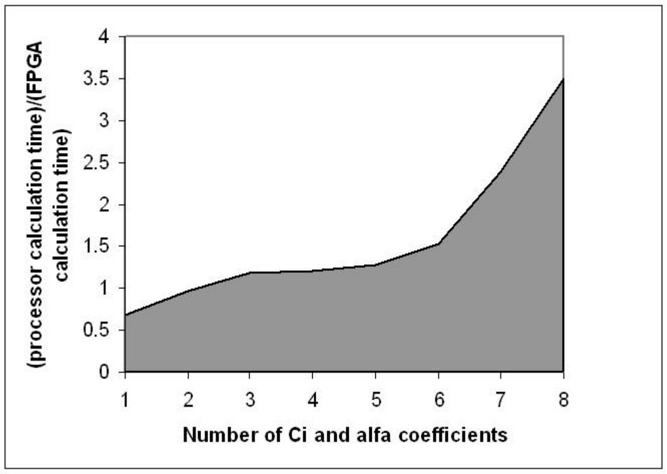

Fig. 3. Impact of the number of atom base coefficients on the speed-up(for the s shell)

FPGA resources consumed by double precision implementation of the orbital module are roughly three times higher than presented in (Tab. 2).

Parameter	Value
Frequency [Mhz]	100
Max. Error [ulp]	1
RMSE	0,67
Pipeline dalay [clk]	64

Table 3. Primary parameters of the module

5 Summary

In this paper, a novel approach to generating orbital function in quantum chemistry has been presented. The authors aim to implement the exchange-correlation potential generation and the orbital function is considered to be a milestone on the way to achieving this. On the other hand, the presented implementation was also considered to be a benchmark meant to deliver reliable test results which allow estimation of quantum chemistry acceleration effectiveness on FPGAs. The obtained speed-up is promising and is expected to be higher along with improvements introduced. Furthermore, resource consumption is relatively low due to the 32-40 bit data precision adopted across building units of the orbital module. An additional role of the presented module is also a data serialization for subsequent modules of the system for exchange-correlation potential generating which are expected to contribute significantly to the overall speed-up.

References

1. Silicon Graphics, Inc. *Reconfigurable Application-Specific Computing User's Guide*, Ver. 005, January 2007, SGI
2. G.Mazur, M.Makowski: Development and Optimization of Computational Chemistry Algorithms, KDM'2008, March-2008, Zakopane, Poland
3. Omondi A.R., Computer Arithmetic Systems, Prentice Hall. Cambridge, 1994
4. 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`
5. 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.
6. 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.
7. M.Wielgosz, E. Jamro, K. Wiatr *Accelerating calculations on the RASC platform. A case study of the exponential function, ARC 2009*, Lecture notes in Springer-Verlag, London LNCS 5453, pp. 306-311