FPGA Implementation of the Selected Parts of the Fast Image Segmentation

Maciej Wielgosz, Ernest Jamro, Dominik Żurek, and 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,
dominik.zurek1102@gmail.com

Abstract. This paper presents preliminary implementation results of the SVM (Support Vector Machine) algorithm. SVM is a dedicated mathematical formula which allows us to extract selective objects from a picture and assign them to an appropriate class. Consequently, a black and white images reflecting an occurrence of the desired feature is derived from an original picture fed into the classifier. This work is primarily focused on the FPGA implementation aspects of the algorithm as well as on comparison of the hardware and software performance. A human skin classifier was used as an example and implemented both on AMD AthlonII P320 Dual-Core2.10 GHz and Xilinx Spartan 6 FPGA. It is worth emphasizing that the critical hardware components were designed using HDL (Hardware Description Language), whereas the less demanding or standard ones such as communication interfaces, FIFO, FSMs were implemented in HLL (High Level Language). Such an approach allowed us both to cut a design time and preserve a high performance of the hardware classification module.

Keywords: Picture segmentation, FPGA, reconfigurable logic, SVM.

1 Introduction

This work is part of the Synat project embracing several initiatives aiming to create a repository of images which are assigned a descriptive name according to their contents. Such a database of tagged images will significantly reduce search time since only picture tags will be processed instead of images so the process will involve simple string operations rather than image recognition. It is worth noting that such a database may also provide a mean to combat Internet threats and crimes through an access to the large well classified repository of visual information.

The project is a huge challenge due to an immense volume of data collected over the past years denoted today as the Internet resources. Therefore the core part of the undertaking is to design and implement a classification system which

R. Bembenik et al. (Eds.): Intelligent Tools for Building a Scient. Info. Plat., SCI 390, pp. 203–216.
springerlink.com

should be both reliable and fast. In order to achieve the high performance of a search engine the most computationally intensive operations are to be ported to hardware. Thus FPGAs due to their strongly parallel structure, huge logic resources and growing processing speed [1] seem to be the best choice.

Image segmentation is a process which aims to separate a picture into several regions based on objects or features of interest. A single SVM system may embrace several modules trained to recognize different features so the unit as a whole is capable of tracing multidimensional objects in terms of a number of features.

It is worth emphasizing that a segmentation may also be regarded as a form of data compression, the classfier accepts images and yields information regarding objects which usually occupies much less memory resources than corresponding original data.

There are plantiful image segmentation algorithms [2,3,4] and their number is still growing to meet constantly rising demands of data analysis systems. However, reliability and data processing speed are the factors which are at a premium when it comes to a real life application of a given algorithm. SVM meets both those criterions and therefore was chosen as a classification algorithm for the project.

2 SVM Classifiers

Support vector machines were originally devised and described by Vapnik [5,6]. They are used for binary classification which means that there are exactly two classes of objects (e.g. black and white rectangles) and a classification formula is found in a training process of the classifier.

The SVM algorithm can be envisioned as a process of creating a hyperplane which separates data in an n-dimensional space. It is conducted in an iterative manner in which a selected plane is gradually adjusted to provide the optimal so-called generalization margin. The following cases may occur:

- Input data is linearly separable and the SVM method guaranties that at least one plane of the best separation margin exists and will be adopted
- A dimension incrementation is used to bring data to a space of more dimensions space so a separation plane can be found

A feature space for 2D can be modeled as a sphere with its center and radius (see Fig. 1).

The sphere is build upon a set of supportive vectors which constitute its structure. A classification process of a priorly trained SVM maybe perceived as probing whether a given point (input data) belongs to the sphere or it's located outside of it. In the first case a point is positively classified whereas in the second one it's considered to be an outliner.

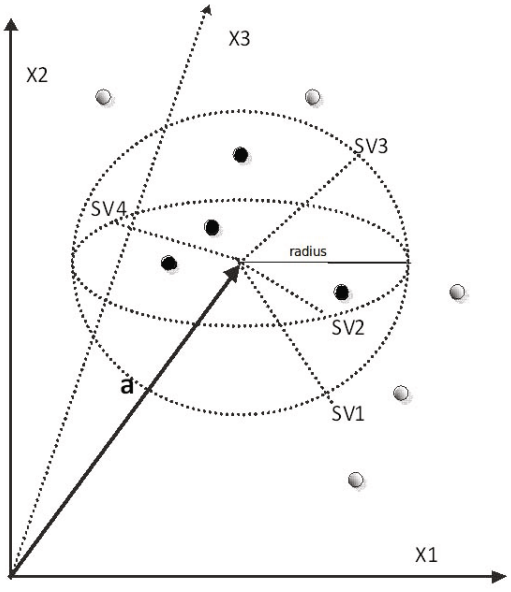

Fig. 1. N-sphere

3 A Choice of a Hardware Platform

It is very important to choose a proper architecture and appropriate data transfer protocol since it affects the overall performance of the computational system. It also may balk an effort invested in the development of the hardware algorithm. Therefore the authors decided to review available FPGA platforms.

FPGAs have been developed since late 1980s, and have a lot of advantages over processors. The most important ones are: massive parallel architecture, reconfigurability, low energy consumption, ability to shape freely its internal architecture.

Design and effective use of computing system based on FPGA is a difficult task, as evidenced by the long history of such trials. The father of the concept of using reconfigurable logic in computational systems is considered Erstina Gerald, who first introduced a vision of a machine consisting of a number of reprogrammable matrices in the 1960s. But the idea had not been fulfilled until the late 80 century, when it first appeared as a reconfigurable computer Algotronix CHS2x4, built with 8 CAL1024 units. Since then the continuing development of modules, which in various ways integrate FPGAs in a computing system.

Existing HPRC (High Performance Reconfigurable Computing) solutions can be classified based on their integration with other computing nodes in the system.

1. Solutions based on FPGAs as the processor directly attached to a computational system. Usually such modules communicate over PCI or PCI Express or Ethernet bus. Such an arrangement is most common in small systems consisting of several nodes. A system bus bandwidth is often the limiting

performance parameter of such solutions. Alpha-Data [7], Nallatech [8], Pico Computing [9] cards can be given as example.
2. System with a dedicated point-to-point connection between FPGAs. This solution has a great potential for parallelization and scaling algorithm. A communication with the nearest neighbor is particularly effective. An example of such a system may be a computer Maxwell [10]
3. Solution based on two-domain approach which divide FPGA part of the system from CPU section. The advantage of this configuration is the ability to communicate across of the all computational units within a system. It is worth taking into account that overloading of a communication bus results in a sudden drop in system performance. Cray XD1 computer can be stated as an example of such a solution.
4. A system based on non uniform memory access. This approach allows for virtually any data exchange between computing nodes in the system and creates a challenge in the form of memory coherence. The best-known commercially available solution of this type is SGI RASC [11], which utilizes the NUMAlink (Non Uniform Memory Access link) bus to access the global memory. The downside of this solution is unpredictability of data access time, what could potentially create difficulties for algorithms for which this parameter is critical.
5. The idea of populating CPU sockets with FPGAs. This approach can be considered as a modification of the architecture described in section 2. An example of such a system is the topology of the DRC coprocessor system Accelium (Xilinx Virtex-5) [12]. This architecture allows for equal access of processor and FPGA to the system resources. However, this has a significant drawback, the occurrence of any error in the processing FPGA automatically affects the stability of the whole system, in particular shared memory read and write operations.

The project described in this paper will be launch on the 1, 4 and 5 architecture since those platforms are available at ACC Cyfronet AGH.

The main research objectives from the hardware perspective are:

1. Implement a large part of the SVM algorithm in FPGA. Design a set of hardware modules which constitute the SVM classifier.
2. Design of effective mechanisms for the inter-module data transfer.
3. Research on the possibility of an effective integration of the FPGA modules within a single computational system using MPI, OpenMP or OpenCL. Such an approach would significantly facilitate integrating of the building modules within a system.

4 System Overview

A human skin classifier OC-SVM (One Class Supportive Vector Machine) was implemented as a preliminary project which allows us to estimate performance

and resource consumption for other classifiers. As a result of an experiment a black-and-white image is generated which reflect human skin location in the original picture which was fed into the classifier. A complete computational procedure is composed of several steps:

- SVM vectors and τ generation (training of the classifier)
- Input image fetch (the step is different for hardware and software implementation)
- Image resize and normalization
- Classification
- Noise and skin-like objects filtration

4.1 The Classification Algorithm

The classification algorithm is given by the following formula:

$$\sum_i \alpha_i K(X_x, X_i) \geq \sum_i \alpha_i K(X_s, X_i) = \tau \tag{1}$$

where τ is the sphere radius, X_s and α_i are supportive vectors derived in a training process, X_x is an input pixel.

Regardless of a choice of vectors in right side of the equation 1 the result is constant and equals τ. Each pixel fed into the classifier is compared against all the support vectors in order to determine if it is located inside the sphere (see Fig.1).

In this implementation a Gaussian computational kernel was used:

$$K = e^{-\gamma \| X_i - X_j \|^2} \tag{2}$$

where γ is a spread of the kernel.

If the (1) is met a given point is classified as belonging to the desired class. For SVM classifies the best results are achieved when input data is normalized (i.e. fall in the range [-1;1]).

4.2 Architecture of the Hardware Module

The computationally intensive routines were ported to hardware to offload the GPP (General Purpose Processor) and to accelerate the computations. It was possible due to several features of the algorithm which makes it well suited for the FPGA implementation such as: fixed-point arithmetic, parallel structure (easy to pipeline), narrow-range input argument. Consequently a series of hardware units were designed which constitute the internal structure of computational module as presented in Fig.2. All the modules are parameterized and pipelined blocks which process a single input vector X every clock cycle. For a sake of the software compatibility the base data format employed in the application is 32 bit fixed-point (16 bits of both fractional and integer part) but it can be adjusted to meet different precision requirements in the future. Each module is equipped

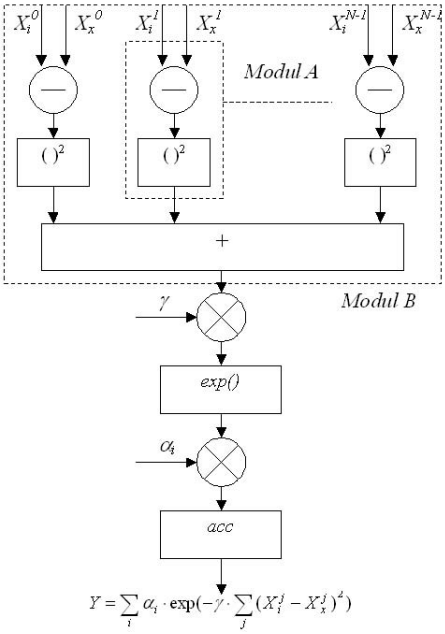

Fig. 2. Block diagram of the classification module

with the overflow signal which propagates across all the units composing the classification module. Such an approach allows us to avoid corruptions of the result just by simply examining the overflow output.

It is possible to connect several classification modules to form a parallel structure as it is depicted in Fig. 3. Furthermore, it is worth noting that supportive vectors (denoted as SV in Fig. 3) are fetched from an external memory only once for the whole computations and therefore can be stored in the internal memory for all the computation. Moreover, the number of the supportive vectors as well as a is not large and usually not exceed tens, thus internal BRAM memory suffice to accommodate those coefficients (e.g. for the human skin classification only 17 supportive vectors are used). Increase of a number of supportive vectors improves the classifier accuracy at the expense of the accumulator throughput decrease (see Fig.2) which in turn affects an overall system performance.

The classifier (presented in Fig. 2) yields one bit results which reflects an occurrence of a feature of interest within an image. Therefore in order to take a full advantage of an external bus throughput, classification results are compacted into 32 bit bundles and sent to a host processor as such. Thereafter the GPP transforms those binary values into pixels to form a black-and-white image depicting the features of interest.

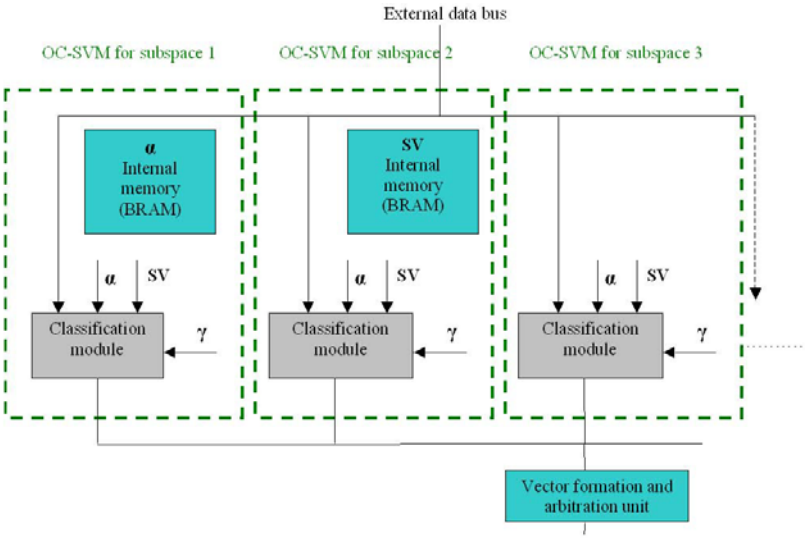

Fig. 3. Block diagram of the multimodule structure

4.3 Exp Module

The exp() function described in this paper was designed in VHDL from scratch. According to authors knowledge there are only few available libraries of fixed point modules including exp() e.g. [13] but they do not meet design requirements such as a parameterization range. Therefore the authors implemented their own exp() module.

The exp hardware algorithm is composed of two steps: reduction of a calculation range and approximation of the function within a priory defined range. It is based on a similar algorithm for floating point numbers [14] and the block diagram of the module was presented in Fig.4.

After applying the following formula, the input argument is separated to integer X_i and fractional X_f part. It should be noted that $2 \times X_i$ can be easily calculated employing a barrel shifter. Therefore the main problem is calculation of $exp(X_f)$.

$$e^x = e^{X/ln(2)} \tag{3}$$

A fractional part X_f = X - ($\lfloor X/ln(2) \rfloor \times ln(2)$) is obtained according to Fig.4. It should be noted that integer part is calculated employing base 2 (thus barrel shifter can be used) but the fractional part is calculated employing base e (thus a simply Taylor expansion can be used). It is worth emphasizing that the exp() is a strongly declining function (the argument is always a negative number) consequently a relatively small input value (absolute value) yields roughly zero as an exp() result. For instance an argument larger than $ln(2^{16})$ =11.09 (which integer part can be mapped on 4 bits)gives zero as the fractional part of the exp() result is represented only on 16-bits. Therefore calculation of $\lfloor X/ln(2) \rfloor \times$

$ln(2)$ can be conducted with the limited bit-width, which significantly reduces hardware requirements of the 1/ln(2) and ln(2) multipliers. The fractional part X_f is then bit separated into -the most X_{msb} and less X_{lsb} significant part. The X_{msb} is fed to the LUT (Look-Up Table) memory. And the X_{lsb} is calculated employing Taylor expansion.

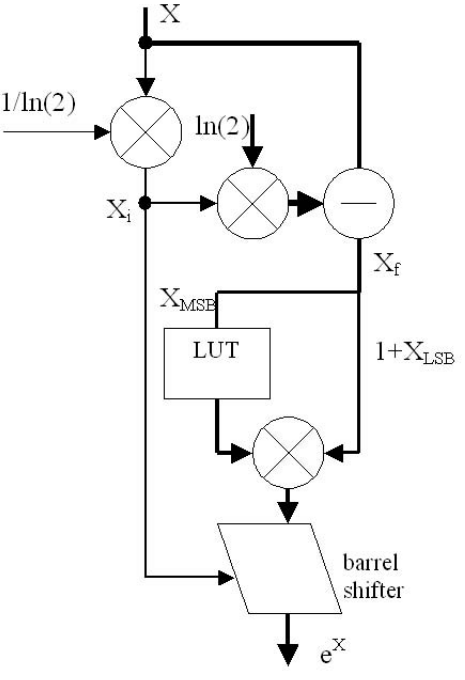

Fig. 4. Block diagram of the exp() module

The complete exp() algorithm is given as follows:

$$e^x = e^{x/ln(2)} = 2^{x_i} \cdot e^{(x-x_i)/ln(2)} \tag{4}$$

$$e^{X_f} = e^{X_{MSB}+X_{LSB}} = e^{X_{MSB}} \cdot e^{X_{LSB}} = LUT(X_{MSB}) \cdot (1 + X_{LSB}) \tag{5}$$

The $e^{X_{LSB}}$evaluation is performed as a LUT operation which coefficients are stored in a internal FPGA block memory denoted as a BRAM. The second part of the equation - $1+X_{LSB}$ was implemented as a short Taylor expansion. It is achievable because X_{LSB} is so small that the next term of the Taylor series, $x^2/2$, does not influence the result. The presented exp() implementation is fast and consumes insignificant resources. Consequently a single FPGA can accommodate several of such modules along with the other units.

5 Implementation Results

The classification algorithm was initially implemented on GPP in C++ and the OpenCV library was used. A set of 17 support vectors was generated which described a human skin, each of which are 32bit RGB colors.

The figures below represent experimental results for the randomly chosen images. It can be noticed that the system wrongly classified some parts of the image. Unfortunately the system often confuses bright objects with a human skin. One way to improve the accuracy is increasing the contrast between an object and a background. Similar result of accuracy improvement may be achieved when a larger number of supportive vectors is employed but it is done at a expanse of a loss of classifier's generalization feature.

Fig. 5. Original image (before segmentation)

Time required to execute the following algorithm on GPP: AMD Athlon II P320 Dual-Core 2.10 GHz (on a single core) for an image of 480x480 pixels takes roughly 0.015s. Hardware implementation according to the formula (1) (assuming that no input data fetch delay is introduced) can be calculated as follows: $17(SVM) \times 480(pixels) \times 480(pixels) \cong 4 \times 10^6$ clock cycles. Consequently theoretical processing time for 200 MHz equals 0.02s. Due to a low resources consumption a single FPGA can accommodate several modules which boost a performance several times.

Fig. 6. Results of human skin segmentation with the proposed method

Fig. 7. Original image (before segmentation)

Fig. 8. Results of human skin segmentation with the proposed method

The implementation results of the module on Xilinx Spartan 6 (XC6SLX75) FPGA were presented in Tab. 1 and Tab. 2.

Table 1. Implementation results of the module building blocks (see Fig.2)

Module	# 4-input LUT	# flip-flops
Square module - A	660 (1%)	220 (1%)
Module - B	1953 (4%)	760 (1%)
exp() module	312 (1%)	277 (1%)

It is worth noting that a number of coefficients has a large impact on the resources occupation. In this particular implementation the number of the coefficient is three (R, G, B).

The classification module is a fully pipelined structure and it is capable of working at the frequency of 200 MHz. Each module generates a single result every n clock cycles where n denotes number of the support vectors employed. This processing speed constrain results from the internal structure of the accumulator (see Fig. 2) which requires n clock cycles to generated a single result.

Table 2. Implementation results of the classifier module for different number of [j] vector coefficients

Module	# 4-input LUT	# flip-flops
2	2689 (5%)	1048 (1%)
3	3377 (7%)	1311 (1%)
4	6319 (13%)	1834 (2%)

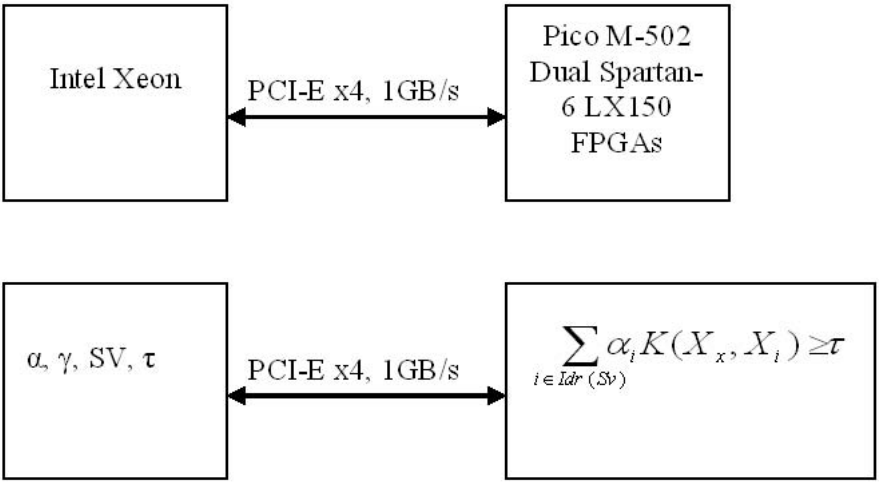

Fig. 9. Data transfer scheme between FPGA and the GPP

Initially the platform of choice was Pico M-502 equipped with two Spartan-6 LX150 FPGA and 512 MB DDR3 of local memory but as the calculations below show the data bandwidth is too low. The board is connected with GPP via PCI-express x4 interconnect which allows for 1 GB/s data transfer in each directions.

Support vectors along with α,γ,τ are generated on the host side (by GPP) and are sent to the FPGA only once for the whole computations (see Fig.9). The module implemented in the FPGA performs the classification for all the X vectors and sends the results back to the host processor.

The data bus is 32 bits width and each vector consist of 32×3 bits. However, taking into account the range in which X vectors fall i.e.[-1;1], it can be noticed that only 16 LSB bits are occupied. Therefore the amount of data to be transfer over the data bus is reduced twice and it results in 16x3 bits. Nevertheless still more than a single bus cycle is required to transfer a single x vector on Pico platform.

Time essential to process a single x vector is given by the following formula (FPGA clocked at 200 MHz):

$$n \cdot 5ns/NoM \tag{6}$$

where NoM stands for the Number of Modules (classification modules) working in parallel and n - denotes a number of support vectors.

Back-of-the-envelope calculations for the human skin classifier (17 support vectors) show that in order to provide compatibility with HD (High Definition) 1080p ($1920 \times 1080 \times 25$) standard FPGA should process each pixel every 25 ns. It means that at least 4 parallel classification modules should be used ($17 \times 5ns/25ns = 3,4$) which in turn requires an external bus throughout of 4,8 GB/s ($16b \times 3 \times 4 \times 200MHz = 6B \times 4 \times 200MHz$).

It turns out that in the case of Pico platform data throughput is a bottleneck. Therefore the DRC AC2020 [12] will be used to implement the classifier. The DRC platform is capable of achieving 9.6 GB/s of aggregated HT bus bandwidth.

6 Summary

In this paper preliminary implementation results of the selected parts of the fast image segmentation were presented along with some performance analysis. Both software and hardware approached were discussed with their critical aspects such as bandwidth limitations and data precision. Furthermore several HPRC platforms were described with a special focus of their architecture and inter-module data exchange capabilities.

As a future work presented segmentation algorithm will be extended with some additional functionality and implemented on the DRC platform [12].

Acknowledgments. The work presented in this paper was financed through the research program - *Synat*.

References

1. Mueller, R., Teubner, J., Alonso, G.: Data Processing on FPGAs, Systems Group, Department of Computer Science, ETH Zurich, Switzerland, VLDB 2009, August 24-28, Lyon, France (2009)
2. Jun, T.: A color image segmentation algorithm based on region growing. In: 2010 2nd International Conference on Computer Engineering and Technology (ICCET), April 16-18, vol. 6, pp. V6-634–V6-637 (2010)
3. Farmer, M.E., Jain, A.K.: A wrapper-based approach to image segmentation and classification. IEEE Transactions on Image Processing 14(12), 2060–2072 (2005)
4. Lan, Y., Li, C., Zhang, Y., Zhao, X.: A novel image segmentation method based on random walk. In: Asia-Pacific Conference on Computational Intelligence and Industrial Applications, PACIIA 2009, November 28-29, vol. 1, pp. 207–210 (2009)
5. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Heidelberg (2000)

6. Ben-Hur, A., Horn, D., Siegelmann, H.T., Vapnik, V.: A support vector clustering method. In: Proceedings. 15th International Conference on Pattern Recognition 2000, vol. 2, pp. 724–727 (2000)
7. http://www.alpha-data.com/
8. http://www.nallatech.com/
9. http://www.picocomputing.com/
10. Baxter, R., Booth, S., Bull, M., Cawood, G., Perry, J., Parsons, M., Simpson, A., Trew, A., McCormick, A., Smart, G., Smart, R., Cantle, A., Chamberlain, R., Genest, G.: Maxwell - a 64 FPGA Supercomputer. In: Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007), pp. 287–294 (2007)
11. http://www.silicongraphics.ru/pdf/rasc_data.pdf
12. http://www.drccomputer.com/pdfs/DRC_Accelium_Overview.pdf
13. http://www.vhdl.org/fphdl/
14. Wielgosz, M., Jamro, E., Wiatr, K.: Hardware implementation of the exponent based computational core for an exchange-correlation potential matrix generation. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2009. LNCS, vol. 6067, pp. 115–124. Springer, Heidelberg (2010)