NESUS
cost IC1305
Network for Sustainable Ultrascale Computing

Proceedings of the Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015) Krakow, Poland

Jesus Carretero, Javier Garcia Blas
Roman Wyrzykowski, Emmanuel Jeannot.
(Editors)

September 10-11, 2015

Chameleon© C2HDL Design Tool In Self-Configurable Ultrascale Computer Systems Based On Partially Reconfigurable FPGAs

Anatoliy Melnyk, Viktor Melnyk, Lyubomyr Tsyhylyk

Intron Ltd, Ukraine

aomelnyk@intron-innovations.com vmelnyk@ intron-innovations.com l.tsyhylyk@gmail.com

Abstract

The FPGA-based accelerators and reconfigurable computer systems based on them require designing the application-specific processors soft-cores and are effective for certain classes of problems only, for which these soft-cores were previously developed. In Self-Configurable FPGA-based Computer Systems the challenge of designing the application-specific processors soft-cores is solved with use of the C2HDL tools, allowing them to be generated automatically. In this paper, we study the questions of the self-configurable computer systems efficiency increasing with use of the partially reconfigurable FPGAs and Chameleon© C2HDL design tool, corresponding to the goals of the project entitled "Improvement of heterogeneous systems efficiency using self-configurable FPGA-based computing" which is a part of the NESUS action. One of the features of the Chameleon© C2HDL design tool is its ability to generate a number of application-specific processors soft-cores executing the same algorithm that differ by the amount of FPGA resources required for their implementation. If the self-configurable computer systems are based on partially reconfigurable FPGAs, this feature allows them to acquire in every moment of its operation such a configuration that will provide an optimal use of its reconfigurable logic at a given level of hardware multitasking.

Keywords self-configurable computer systems, field-programmable gate arrays, high-performance computing, reconfigurable computing, C2HDL design tools.

I. Introduction

Today one of the most promising areas of activity in the field of high performance computing is creation of the reconfigurable computer systems (RCCS). RCCSs compete with other types of high-performance computer systems due to high speed characteristics of the modern field-programmable gate arrays (FPGAs) - the hardware base of a reconfigurable computing environment (RCE) of RCCS, and due to advances in the design technology of application-specific processors to be synthesized in RCE of RCCS.

Co-functioning of a computer system based on general-purpose processors with application-specific processors synthesized in RCE, whose structure considers executed algorithms features, allows increasing its overall performance by 2-3 orders of magnitude. The reconfigurability and ability to synthesize an application-specific processor (ASP) with a new structure and functionality in RCE allows changing the functional commitment of a created thereby RCCS with preserving its high performance in the new class of problems.

Along with high performance ensured by the RCCS, there are also challenges associated with their application. These particularly are significant timing expenses for task distribution between the processing units, often lack of IP cores required for implementation in the RCE which forces to develop them from scratch, and high additional requirements to the RCCS users qualification, as they, besides modeling and programming, must perform a system analysis, design the ASPs architecture, perform their synthesis and implementation in RCE.

Deprived of the above mentioned problems are self-configurable computer systems (SCCS), where these labor-intensive and time-consuming tasks are fully automated and replaced from the operator to the computer system. Taking into account the necessity of further improvement of the computer means performance and extension of the reconfigurable devices application in computer systems, further development in the field of self-configurable computer systems is a topical task of scientific research and engineering.

II. Related Work

The concept of the self-configurable FPGA-based computer systems design, the method of information processing in them, and their structure are proposed in paper [1]. The issue of labor-intensive and time-consuming tasks, which is a characteristic for RCCS, is removed from SCCS owing to the new method of information processing applied in it.

The software tools that should be used in a SCCS as its components are available today. In this regard, we may consider an approach to the development of computational load balancing systems between a general-purpose computer and the hardware accelerator proposed in [2]. The IP cores' generators [3], [4], IP cores' libraries [5], and system level design tools and solutions available on the market could be used for the ASP design on the basis of its algorithm description by a high-level programming language [6] - [8].

III. Problem Statement

Implementation of the SCCS basing on partially reconfigurable FPGAs enables organization of multiple-task execution in the reconfigurable environment. This opportunity is provided as the subprograms of different tasks are executed independently in FPGA's different reconfigurable regions, and each of them is loaded into the FPGA as a partial configuration after initialization of the respective program. Such SCCS operation has a number of advantages, among which - the actual multitasking, an effective use of the reconfigurable logic and rationalization of energy consumption. At the same time, this mode of the SCCS operation imposes additional requirements for the generating system to create the application-specific processors HDL-models. Depending on the workload of the computer system, the amount of available for one separate task reconfigurable logic resources at a time can range from a maximum value that corresponds to all FPGA dynamic part resource, to the minimum value that corresponds to one or a number of reconfigurable regions, and vice versa. The question arises to organizing dynamic reallocation of the reconfigurable logic resources and replacing some running application-specific processors with others performing the same tasks but differing by the equipment volume. This should be done to provide an effective use of resources and the required level of multitasking.

To address this challenge it is necessary, during the program compilation, for each subprogram executed in the reconfigurable environment, to generate a number of application-specific processors HDL-models $ASPM\left\{ASPM_{opt},...,ASPM_{min}\right\}$, where $ASPM_{opt}$ is an optimum HDL-model that uses all the space-time properties of an algorithm given by the subprogram and to be implemented requires the largest amount of the reconfigurable logic resources among the $ASPM$ models; $ASPM_{min}$ is an HDL-model that to be implemented requires the minimum amount of the resources. In this regard, we propose the Chameleon© C2HDL design tool, which for each algorithm, given by the ANSI C program, can generate a set of application-specific processors VHDL soft-cores that differ by the amount of equipment to be implemented.

The paper structure is the following:

Section IV shows the principles of information processing in SCCS and its structure organization.

Section V highlights the partially reconfigurable FPGAs operation features.

Section VI introduces the characteristics and features of the Chameleon© C2HDL design tool.

Section VII shows an example of application of the Chameleon© C2HDL design tool in the SCCS for creation of a set of FFT processors VHDL models.

In our experiment the reconfigurable environment of the SCCS is built on the Altera FPGA, therefore created processors models are targeted at being implemented in this FPGA and differ mainly by the number of the embedded DSP blocks they use. The duration of these FFT processors VHDL models generation and their technical characteristics are shown.

Section VIII concludes the paper.

IV. Self-Configurable Computer Systems And The Method Of Information Processing In It

The self-configurable computer system is the computer system with reconfigurable logic where the program compilation includes automatically performed actions of creation of configuration. and which acquires that configuration automatically in the time of program loading for execution [1]. The SCCS automatically executes: 1) computational load balancing between the general-purpose processor and reconfigurable environment (RCE); 2) creation of an application-specific processor (ASP) HDL-model. Loading of the configuration files obtained after logical synthesis into the RCE is carried out by the operating system in parallel with loading of the computer subprogram executable file into its main memory after program initialization [1].

The method of information processing in the SCCS consists of three stages: compiling the program, its loading, and execution. The user creates a program P_{in} written in a high-level programming language and submits it into the SCCS. During compiling the SCCS automatically performs the following actions: divides this program into the general-purpose processor's subprogram P_{GPP} and RCE's subprogram P_{RCE}, performs P_{GPP} compilation, generates P_{GPP} executable file obj, creates ASP's HDL-model $ASPM$ to perform P_{RCE} subprogram, performs ASP's logic synthesis, and stores the obtained executable file obj and configuration files of RCE $\mathbf{conf} = \{conf_q, q = \overline{1...K_{FPGA}}\}$, where K_{FPGA} is the number of FPGAs forming RCE, into the secondary memory.

These actions are performed in the SCCS with the following means:

1. The computational load balancing system for load balancing between a general-purpose processor and RCE. This system automatically selects fragments from the program P_{in} whose execution in the RCE reduces its execution time, and divides the program P_{in} into the P_{GPP} subprogram replacing the selected fragments in there by instructions for interaction with the RCE, and the P_{RCE} subprogram, formed from the selected fragments. An example of such a system is described in [2]. This system creates the RCE subprogram in x86 assembly language, thus it must be supported by the means for the assembly language code translation into a high-level language to be used in the SCCS. The tools of this type are available on the market, for example Relogix Assembler-to-C Translator [9] from MicroAPL.
2. A compiler for compiling P_{GPP} subprograms from the language that they are represented in into the object codes obj that can be directly executed by the general-purpose processor.
3. A generating system for the ASPs HDL-models creation. The system automatically generates models $ASPM$ from the RCE subprograms P_{RCE}, like Chameleon© [7], [10], that is discussed in this article, or Agility Compiler [11] and DK4 Design Suite [12] from Celoxica, or CoDeveloper from Impulse [13].
4. Logic synthesis tools and FPGA configuring tools for the ASPs HDL-models logic synthesis during the program compilation stage and FPGA configuring during the program loading stage. These tools are available from the FPGA vendors, for example, Vivado Design Suite, ISE, Alliance, Foundation from Xilinx; Quartus II, Max + II from Altera.

From the **conf** and obj files a combined executable file is formed and stored into the secondary memory. At the stage of the program loading after its initialization, the SCCS loads the executable file obj of the general-purpose processor's subprogram into the main memory using a conventional loader and, at the same time, loads the configuration files $\mathbf{conf} = \{conf_q, q = \overline{1...K_{FPGA}}\}$ into the RCE and thus creates an ASP in there using the FPGA configuring tools. Then, the stage of the program execution is performed.

The major part of the SCCS basic software means represents a compiler which combines the computational load balancing system, a compiler for GPPs subprograms compilation, a generating system for ASPs HDL-models creation, and ASPs logical synthesis tools.

The reconfigurable environment of the SCCS can be realized on the base of partially reconfigurable FPGAs, which gives an opportunity to organize the hardware multitasking there and brings a number of other benefits. The partially reconfigurable FPGAs operation features are briefly highlighted below.

V. Partially Reconfigurable FPGAs Operation Features

The ability to reconfigure a part of an FPGA circuitry after its initial configuration while the other parts remain unaffected is referred to as partial reconfiguration. The direct benefits of using this ability is a significant reduction of the duration of reconfiguring and reduction of the memory size required for the configuration storage (the size of the bit-stream is directly proportional to the number of resources being configured). Also, this ability opens new possibilities for the reconfigurable logic application in computers, particularly, it allows organizing hardware multitasking in FPGA and embodying the concept of Virtual Hardware, that is combined extremely well with the concept of SCCS design.

Partial reconfiguration is carried out in FPGA by downloading partial configurations files after its initial configuration, and thus - during the operation. These files specify only the configuration of the FPGA parts called Reconfigurable Partitions or Reconfigurable Regions, each of them contains separate device's modules. Reconfigurable partitions contain a certain amount of equipment and have a clearly defined location and boundaries in the FPGA circuitry. In this regard, the device needs a modular structure. The modules loaded into the reconfigurable partitions are called Reconfigurable Modules.

Partial reconfiguration can be static, when the device is not active during the reconfiguration process (while the partial configuration data is sent into the FPGA, the rest of it is stopped and brought up after the configuration is completed), and dynamic, also known as active partial reconfiguration, which enables changing the part of the FPGA while the rest of it is still operating.

Besides one or more reconfigurable regions, a partially reconfigurable FPGA also contains a static region which remains unchanged during partial reconfiguration. For example, partial reconfiguration controller, memory and interface logic can operate in this region. The Partial Reconfiguration Controller automates the mentioned process. The user can develop a controller by himself or can use ready available on the market solution. The controller can also be external to the FPGA device.

Two modes of the partial reconfiguration are used:

- Module-based - implies creation of a reconfigurable module and, with the help of relevant software, generation of its partial configuration code. This code completely replaces the previously synthesized reconfigurable module in the selected reconfigurable region. Note that this approach requires interfaces interoperability of all reconfigurable modules that operate in one reconfigurable region.
- Difference-based - implies introducing small changes to the scheme of the previously synthesized reconfigurable module. Partial configuration code contains information about the differences between the structures of the existing and new modules operating in the reconfigurable region, and is formed by "fusion" of the binary codes of the previously loaded to the FPGA configuration with the new one, for example, using XOR operation. This approach makes it possible to significantly reduce the size of configuration code. It is used, for example, to replace the contents of table operating device, memory contents, etc. This approach is especially interesting for implementation of evolutionary algorithms.

Partial reconfiguration design flow and mechanisms are being continuously improved. For example, in Virtex, Virtex-II, Virtex-II Pro and Virtex-E FPGAs from Xilinx, the configuration can be changed only by full columns of the reconfigurable matrix, and their numbers have to be multiples of 4 (4, 8, 12, ...). In Virtex-4 FPGAs this restriction is eliminated, while it is possible to change the configuration of an arbitrary rectangular area of the matrix, with some restrictions on its height. In modern Xilinx FPGAs (today it is 7th generation: Artix-7, Kintex-7, Virtex-7 and Zynq-7000 SoC) the minimum regions whose configuration can be changed independently are called Reconfigurable Frames. The width of the reconfigurable frames is one element (there are different types of elements, including CLB, BRAM, DSP), while the height - the one clock region or input/output block. Some examples are as follows: in the Xilinx FPGAs 7th generation devices [14] - CLB: 50 x 1; DSP48: 10 x 1; RAM: 10 x 1; in the UltraScale devices [15] - CLB: 60 x 1; DSP48: 24 x 1; RAM: 12 x 1.

A partial configuration file consists of a certain number of configuration frames (not to be confused with the reconfigurable frames). The configuration frame is the minimum unit of information of this file and sets a configuration for one reconfigurable frame.

In the Altera FPGAs, the partial reconfiguration is implemented similarly.

VI. Chameleon© C2HDL Design Tool

The Chameleon© C2HDL design tool is initially targeted for use in the heterogeneous ultrascale computer systems. It is intended for the ASP's HDL-model automatic generation from the algorithm described in the ANSI C language [7], [10]. The developer, specifying an algorithm of the data processing on ANSI C, in return gets a fully debugged and synthesizable VHDL RTL model of the device that implements the described algorithm. The architecture of the device is fully optimized for the executed algorithm and maximally uses its ability for paralleling. The obtained VHDL design may be further implemented in the FPGA by any FPGA design solution, e.g. the Xilinx Vivado Design Suit or Altera Quartus II.

Besides the algorithm of the data processing, the input information for the Chameleon© C2HDL design tool are also the ASP's interface specification and technical characteristics, for example, desired performance or algorithm execution time boundaries. The platform for the ASP synthesis is configurable processor architecture configured according to the following input parameters: the number of Functional Units, an instruction set for each Functional Unit, the size and content of the instruction and data memory, the communication network structure.

The basic scheme of the Chameleon© C2HDL design tool operation is shown in Figure 1.

The Chameleon© C2HDL design tool features:

1. Short generation time. For example, generation of the FFT processor VHDL model with 50 Functional Units takes several minutes on a conventional PC.
2. Desired pre-set level of the algorithm parallelization.
3. Quick search of the appropriate level of parallelization to achieve the desired ASP's performance or power consumption.
4. The architecture of the ASP is tested and verified, which eliminates the probability of synthesis and operation errors.

Thus, this tool can be effectively used in the SCCS, and the example of its usage is shown in the next section.

VII. Experimental Results

We have used the Chameleon© C2HDL design tool as one of the basic software means of the SCCS compiler. The SCCS hardware platform is realized on the base of the conventional personal computer running on the Windows OS and the reconfigurable environment built on the Cyclone V FPGA from Altera. The RCE subprogram chosen for the experiment represents the algorithm of the 64-point Fast Fourier Transformation in the ANSI C language, its code is given in Figure 2. This program has been submitted to the input of the Chameleon© C2HDL design tool, and a set of the RTL VHDL-models of the 64-points FFT processors, whose structures contain a different number of Functional Units, has been automatically generated. As a most productive the one containing 15 Functional Units was determined by the Chameleon© C2HDL design tool; in all the models a number of these modules is determined automatically. The Functional Units are implemented in the Cyclone V FPGA as an embedded DSP blocks in relation 1x1.

Depending on the workload, the SCCS operating system can choose the FPGA partial configuration that contains an appropriate by the equipment amount or a performance FFT processor, and replace the operating in the RCE instance of the processor to another on

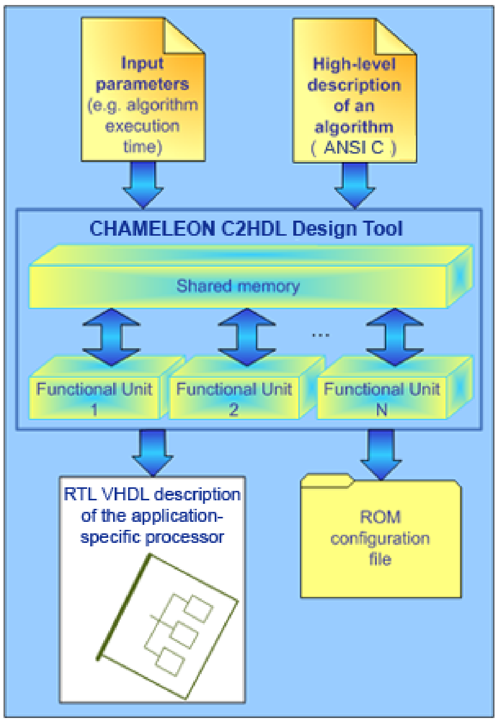

Input parameters (e.g. algorithm execution time)
High-level description of an algorithm (ANSI C)
CHAMELEON C2HDL Design Tool
Shared memory
...
Functional Unit 1
Functional Unit 2
Functional Unit N
RTL VHDL description of the application-specific processor
ROM configuration file

Figure 1: Basic Scheme of Chameleon© C2HDL Design Tool Operation.

the run. Table 1 shows, the technical characteristics of the FFT processors VHDL-models generated with Chameleon© C2HDL design tool, and synthesised in the Cyclone V 5CSEMA5F31C6 device by the Quartus II 13.1.0 Web Edition.

Number of the Functional Units	LUT utilization	Maximum Frequency (MHz)	Command count	FFT time (us)
1	1.809	204.08	1900	9.31
2	2.380	200,32	1015	5,07
4	3.054	216,8	575	2,65
7	4.806	174,52	388	2,22
8	4.858	190,73	352	1,85
10	6.715	171,85	311	1,81
13	9.200	149,7	190	1,27
15	10.198	138,48	180	1,30

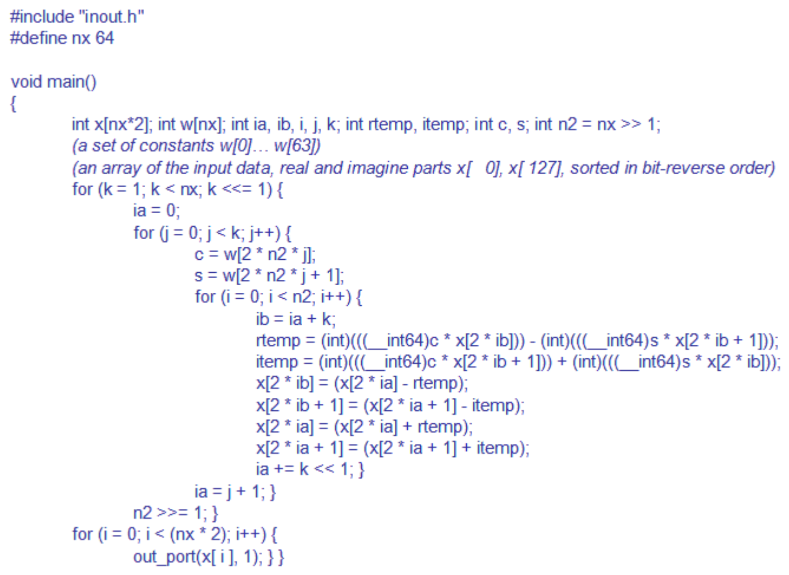
Table 1: Technical Characteristics of FFT Processors.

Basing on this data, the SCCS operating system can choose which FFT processor configuration to acquire at a certain moment of its operation, depending on the actual workload. For example, the configuration consuming 1809 LUTs executes FFT in 9.31 us, and configuration consuming 9200 LUTs - in 1.27 us.

The time required by the SCCS for the FFT processors VHDL-models generation generally increases linearly with increasing the number of parallel Functional Units (see Figure 3). The main part of the generation time is spent on the algorithm parallelization and schematic optimization.

VIII. Conclusions

Implementation of the SCCS basing on partially reconfigurable FPGAs enables organization of the simultaneous multiple-task execution in the reconfigurable environment of the SCCS as the subprograms of different tasks are executed independently in different reconfigurable regions of the FPGA. Such SCCS operation has a number of advantages, among which, besides the actual multitasking - effective use of the reconfigurable logic and rationalization of energy consumption. At



```
#include "inout.h"
#define nx 64

void main()
{
        int x[nx*2]; int w[nx]; int ia, ib, i, j, k; int rtemp, itemp; int c, s; int n2 = nx >> 1;
        (a set of constants w[0]… w[63])
        (an array of the input data, real and imagine parts x[  0], x[ 127], sorted in bit-reverse order)
        for (k = 1; k < nx; k <<= 1) {
                ia = 0;
                for (j = 0; j < k; j++) {
                        c = w[2 * n2 * j];
                        s = w[2 * n2 * j + 1];
                        for (i = 0; i < n2; i++) {
                                ib = ia + k;
                                rtemp = (int)(((__int64)c * x[2 * ib])) - (int)(((__int64)s * x[2 * ib + 1]));
                                itemp = (int)(((__int64)c * x[2 * ib + 1])) + (int)(((__int64)s * x[2 * ib]));
                                x[2 * ib] = (x[2 * ia] - rtemp);
                                x[2 * ib + 1] = (x[2 * ia + 1] - itemp);
                                x[2 * ia] = (x[2 * ia] + rtemp);
                                x[2 * ia + 1] = (x[2 * ia + 1] + itemp);
                                ia += k << 1; }
                        ia = j + 1; }
                n2 >>= 1; }
        for (i = 0; i < (nx * 2); i++) {
                out_port(x[ i ], 1); } }
```

Figure 2: Program of 64-Point FFT Algorithm in ANSI C.

the same time, this mode of the SCCS operation imposes additional requirements for the generating system to create the application-specific processors HDL-models. The question arises to organizing dynamic re-allocation of the reconfigurable logic resources and replacing some running application-specific processors with others performing the same task but differing by the equipment volume. This should be done to provide an effective use of resources and the required level of multitasking. To address this challenge, it is necessary, during the program compilation, for each subprogram executed in the reconfigurable environment, to generate a number of application-specific processors HDL-models. We propose in this regard to use the Chameleon© C2HDL design tool.

In the article we consider the SCCS structure and the method of information processing in it, the partially reconfigurable FPGAs operation features, and the Chameleon© C2HDL design tool operation and features among which short generation time, desired pre-set level of the algorithm parallelization, automatic generation of tested and verified ASP HDL models. One of the features of the Chameleon© C2HDL design tool is its ability to generate a number of application-specific processor soft-cores executing the same algorithm differing by the amount of FPGA resources required for their implementation. For the self-configurable computer systems based on partially reconfigurable FPGAs this feature allows acquiring in every moment of its operation configuration that will provide an optimal use of its reconfigurable logic at a given level of hardware multitasking.

To estimate the benefit, we have experimented with the Chameleon© C2HDL design tool as one of the basic software means of the SCCS compiler. The SCCS hardware platform is realized on the base of the conventional personal computer running on the Windows OS and the reconfigurable environment built on the Cyclone V FPGA from Altera. Chosen for the experiment RCE subprogram represents the algorithm of the 64-point Fast Fourier Transformation in the ANSI C language. This program has been given to the input of the Chameleon© C2HDL design tool, and a set of the RTL VHDL-models of the 64-point FFT processors has been automatically generated. The experimental results have shown that the Chameleon© C2HDL design tool generates a set of FFT processors with high technical characteristics in very short time, and satisfies the basic requirements for a generating system of the SCCS to provide its effective operation.

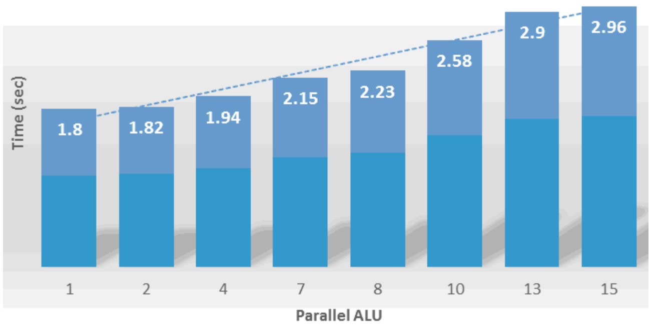


Figure 3: Time Required for 64-Point FFT Processors VHDL-Models Generation.

References

[1] A. Melnyk, V. Melnyk, "Self-Configurable FPGA-Based Computer Systems," *Advances in Electrical and Computer Engineering*, vol. 13, no. 2, pp. 33-38, 2013.

[2] V. Melnyk, V. Stepanov, Z. Sarajrech, "System of load balancing between host computer and reconfigurable accelerator," *Computer systems and components, Scientific Journal of Yuriy Fedkovych Chernivtsi National University*, vol. 3, no. 1, pp. 6-16, 2012.

[3] "A Proven EDA Solutions Provider makes all the difference". [Online]. *Available: http://www.aldec.com/en.*

[4] Xilinx Core Generator. Xilinx Inc. [Online]. *Available: http://www.xilinx.com/ise/products/coregen_overview.pdf - 2005.*

[5] Melnyk, A, Melnyk, V. "Organization of libraries of standardized and custom IP Cores for high-performance hardware accelerators", Proceedings of IV-th all-Ukrainian conference "Computer Technologies: Science and Education", Ukraine, Lutsk, 9-11 October 2009. -P. 113-117.

[6] Genest, G. "Programming an FPGA-based Super Computer Using a C-to-VHDL Compiler: DIME-C", Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference, 5-8 Aug. 2007. - P. 280 - 286.

[7] "Chameleon - the System-Level Design Solution," [Online]. *Available: http://intron-innovations.com/?p=sld_chame*

[8] ANSI-C to VHDL Compiler. [Online]. *Available: http://www.nallatech.com/FPGA-Development-Tools/dimetalk.html.*

[9] "Relogix Assembler-to-C translator," [Online]. *Available: http://www.microapl.co.uk/asm2c/*

[10] Melnyk, A., Salo, A., Klymenko, V., Tsyhylyk, L. "Chameleon - system for specialized processors high-level synthesis", Scientific-technical magazine of National Aerospace University "KhAI", Kharkiv, 2009. N5, pp. 189-195.

[11] Agility Compiler for SystemC. Electronic System Level Behavioral Design & Synthesis Datasheet. 2005. [Online]. *Available: http://www.europractice.rl.ac.uk/vendors/agility_compiler.pdf*

[12] Handel-C Language Reference Manual For DK Version 4. Celoxica Limited, 2005. - 348p.

[13] Impulse CoDeveloper C-to-FPGA Tools. [Online]. *Available: http://www.impulseaccelerated.com/products_universal.htm*

[14] Vivado Design Suite User Guide. Partial Reconfiguration. UG909 (v2015.2) June 24, 2015.

[15] UltraScale Architecture. Online. *Available: http://www.xilinx.com/products/technology/ultra-scale.html*