Proceedings of the First International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2014)
Porto, Portugal

Jesus Carretero, Javier Garcia Blas
Jorge Barbosa, Ricardo Morla
(Editors)

August 27-28, 2014

Improvement of Heterogeneous Systems Efficiency Using Self-Configurable FPGA-based Computing

Anatoliy Melnyk, Viktor Melnyk

Lviv Polytechnic National University, Ukraine
aomelnyk@polynet.lviv.ua vamelnyk@lp.edu.ua

Abstract

Computer systems performance is is being improved today using two major approaches: general-purpose computers computing power increase (creation of multicore processors, multiprocessor computer systems, supercomputers), and adaptation of the computer hardware to the executed algorithm (class of algorithms). Last approach often provides application of the ASIC-based and FPGA-based hardware accelerators, also called reconfigurable, and is characterized by better performance / power consumption ratio and lower cost as compared to the general-purpose computers of equivalent performance. However, such systems have typical problems. The ASIC-based accelerators: 1) are effective for certain classes of algorithms only and 2) algorithms and software require adaptation for effective application. The FPGA-based accelerators and reconfigurable computer systems (that use FPGAs as a processing unit): 1) in the process of writing require a special program to perform computing tasks balancing between the general-purpose computer and FPGAs; 2) require designing the application-specific processor soft-cores; and 3) are effective for certain classes of problems only, for which application-specific processor soft-cores were previously developed. In this paper, we consider an emerging type of high-performance computer systems called self-configurable FPGA-based computer systems, which are deprived of specified challenges. We have analyzed the background of self-configurable computer systems creation, presented current results of our research, and introduced some ongoing works. Self-configurable computer systems are being developed within the project entitled "Improvement of heterogeneous systems efficiency using self-configurable FPGA-based computing" that is the part of the NESUS Action.

Keywords Field programmable gate arrays, high performance computing, reconfigurable computing, self-configurable computer systems.

I. Introduction

Computer systems performance increase continues to be a determinative factor for the progress in science and engineering branches, and also have become a catalyst of emerging a range of new spheres of human activity. It is hard to find a field today that does not depend on this characteristic of the computer systems. However in the beginning of the 21^{st} century the pace of performance increase of the general-purpose processors that make a base of personal computers and multiprocessor computer systems, fast until now, has begun getting slower. This trend is even more notable today, when the general-purpose processors clock rate practically remains the same within several last years. The reasons for this are the fundamental boundaries in the energy efficiency for the CMOS technology used today in the integrated circuits industry and limitations that a traditional method of information processing imposes on the general-purpose processor architecture, particularly, the sequential execution of instructions in the program, and the sequential access to instructions and data in the memory. Trying to provide the instruction set compatibility for each next processor generation with the previous one and to support the existing software by the processors of different generations, the hardware and software manufacturers do not introduce principal changes neither into the single-processor architecture nor into the method of information processing in it.

The task of computer systems performance increase is being solved today mostly by creating the multi-core processors, where the number of cores in the chip ranges from a few to tens of complex and to hundreds of simpler ones, and tends to grow. However, an approach to increase the number of cores in the chip in the system has principal disadvantages as well. Thus, having the parallel data processing, it is also necessary to investigate and develop much more energy-efficient computer systems on each technological level, including devices, hardware architecture and software.

Therefore, today one of the most perspective directions in high-performance computing is creation of the heterogeneous computer systems. These systems combine one or more general-purpose processors and hardware accelerators, including those built on the basis of reconfigurable environments - field-programmable gate arrays (FPGA).

The problems related to designing the heterogeneous computer systems with the use of the hardware accelerators, primarily the reconfigurable ones, are considered in the book [1]. In this paper, the design issues and the development trends of the heterogeneous computer systems - from those built on the ASIC-based hardware accelerators to the reconfigurable and self-configurable ones - are being discussed.

II. Improving the Computer Systems Performance Using the ASIC-based Hardware Accelerators

The ASIC-based hardware accelerator (AHA) is a device that contains a high-performance programmable processor with architecture dedicated to the specific class of algorithms and intended to increase the computer system performance on that class of algorithms. Typ-

ically, AHAs are used to execute the complex algorithms applied to big data arrays that cannot be processed by the general-purpose processor during the acceptable period of time.

Today the most widely used AHAs are [1]: the CELL processors developed jointly by IBM, Toshiba and Sony; the accelerator boards manufactured by ClearSpeed (Great Britain-USA); the GPGPU boards produced by ATI and Nvidia; and the GRAPE boards developed jointly by the scientists from the University of Tokyo and the National Astronomical Observatory of Japan.

Usually, when using AHA, the time of task execution is shortened by one/two orders of magnitude as compared to that spent for this task by the general-purpose processor. However, there are several problems that significantly reduce AHA effectiveness, namely:

1. AHAs are very complex devices. Their development requires years of work of a number of engineers (four years and nearly four hundred of engineers for the CELL processor creation).

2. AHAs have got the parallel architecture. In order to write the parallel programs for them, first the algorithms should be adapted to this architecture that practically means their re-designing and afterwards the parallel programs should be developed using special programming languages, and this requires a great amount of intellectual work. Moreover, not for all algorithms this architecture is effective.

3. An approach to combine a general-purpose computer with AHA gives an opportunity to get high performance for executing the tasks, on which the algorithms, that AHA is dedicated to, are based, but not an arbitrary task.

Having a goal to provide the computer systems with the opportunity to achieve high performance characteristics while executing arbitrary tasks, the accelerators are being built on the basis of programmable logic devices. Such hardware accelerators, as well as the computer systems with them, are called reconfigurable.

III. Reconfigurable Computer Systems

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

Co-functioning of the computer system based on the general-purpose processors with application-specific processors synthesized in RCE, whose structure considers an executed algorithms features, allows its productivity to be increased by 2-3 orders of magnitude. Reconfigurability and ability to synthesize an application-specific processor (ASP) with a new structure and functions in RCE allow one to change the functional commitment of RCCS created thereby with preserving its high performance at the new class of problems.

The RCCS architecture and organization on different levels are described in [2]-[4]. By analyzing these studies one may conclude difficulty of information processing in such systems, which is a consequence of a need to perform the ASPs design and synthesis in RCE before being used. The ASP design is performed by describing its architecture in the VHDL or Verilog hardware design languages (HDL) with the use of the register transfer level design tools, or even by specifying its characteristics and algorithm to be executed in the high-level programming language, as it is provided by the advanced electronic system level design tools, for example, System-C [5] from Celoxica, Catapult-C [6] from Calypto Design Systems, DIMETalk [7] from Nallatech, CoDeveloper [8]. These tools enable creation of the HDL-descriptions of computing devices on the register transfer level from the programs written in a high-level programming language, often a modified C language. The Chameleon Technology and tools [9] from Intron, as well as SPARK developed at the University of California, are intended to design ASPs using the algorithm description in the ANSI C language. The basic platform for ASPs creation here is a configurable processor architecture, which provides creation of its desired configuration with the use of the following configuration parameters: the number of functional units, the instruction set of each functional unit, the capacity of the program and data memories, the number of inputs and outputs of the communication network. These parameters together with the specification of the processor's interface should be submitted in addition to the algorithm description.

IV. Information Processing Method in RCCS. Challenges and Solutions

Information processing in RCCS can be represented as a sequential execution of the four stages [10]. At the first stage, the user creates the program P_{in} written in the high-level programming language, divides this program into the computer subprogram P_{GPC} and the RCE subprogram P_{RCE}, performs the P_{GPC} subprogram compilation, generates its executable file *obj* and stores it in the computer memory. At the second stage, the user develops (or uses a ready-made solution) an HDL-model *ASPM* of ASP intended to perform the P_{RCE} subprogram of RCE, performs the logical synthesis of the ASP and loads the configuration files $\mathbf{conf} = \{conf_q, q = \overline{1...K_{FPGA}}\}$, where K_{FPGA} is the number of FPGAs forming RCE, and, thus, creates an ASP in RCE. At the third stage, after the program initialization, the operating system loads the executable file *obj* of the computer subprogram to its main memory. At the fourth stage, RCCS executes the P_{in} program. Computer executes its own subprogram P_{GPC}, RCE executes its own subprogram P_{RCE}.

By analyzing the above-described method of information processing in RCCS, one can conclude the problems that significantly impede the improvement of its efficiency, namely:

- the need in the process of writing a program to perform computing tasks balancing between the general-purpose computer and the reconfigurable environment;

- the need of designing the application-specific processor soft-cores to be implemented in RCE that requires specialists and software packages for the IP Cores designing, testing, synthesis and implementation;

- in RCCS, there is an unsolvable problem caused by the inability to develop the application-specific processor HDL-models to be implemented in RCE that will be effective for the series of problems. This challenge imposes a fatal limitation on RCCS: these systems are effective only for certain classes of problems,

for which the application-specific processor HDL-models have been developed previously.

Automation of all stages of information processing in RCCS is a key approach to solve the above problems. The development of the ASP's HDL-model at the second stage requires from the user a significant amount of time and knowledge of the system-level design technology. However, as mentioned above, today the software tools are available allowing automatic creation of the ASP's HDL-models from high-level description of the algorithm to be implemented in. This software tools transform the algorithm described in the high-level programming language into the HDL-model of ASP. By linking the operations of ASP's HDL-model generation, ASP's logical synthesis and RCE configuring in automatically executable sequence, the computer system can load the configuration codes into RCE automatically with no user intrusion.

It should be noted that the use of generation tools imposes condition of availability of a high-level algorithm description to be implemented in ASP. The user creates this description at the first stage during dividing the input program into two subprograms, and this also requires from the user a significant amount of time. Automation of load balancing, besides reduction of amount of time, will allow one to:

1. link operations of load balancing and compilation of computer subprogram in one startup sequence, and, as a result, to obtain a subprogram executable file without user intrusion;

2. link operations of load balancing, generation of ASP HDL-model, ASP's logical synthesis and RCE configuration in one startup sequence, and, as a result, to load configuration codes into RCE automatically without user intrusion.

Consequently, automation of steps performed at the first two stages of information processing method in RCCS, i.e. automatic obtaining of computer subprogram executable file and automatic creation and loading of ASP configuration files into RCE for RCE subprogram execution, will allow one to reduce:

- the execution time for information processing;

- the complexity of information processing since the user has no longer to perform the systems analysis, ASPs architecture design and ASPs logical synthesis.

However, automation of the two first stages execution does not solve another problem mentioned above - namely, the list of problems that RCCS is effective on remains short and depends on the functional characteristics of ASPs implemented in RCE. Their change requires, at least, repeating the second stage's steps of the above-mentioned method of information processing, which should be done by the user.

This challenge can be solved by improving the method of information processing in RCCS in the way that loading configuration files obtained after the logical synthesis into RCE is carried out not by the user but by the operating system, and not at the second stage but at the third one, in parallel with loading the computer subprogram executable file into its main memory after the program initialization. This implies that configuration files should be stored in the computer memory after the logical synthesis. Thus, because the configuration files are formed automatically in parallel with the computer subprogram executable file and stored in its memory, the entire sequence of actions from the beginning of the load balancing up to obtaining the executable file and the configuration files should be treated as a single stage of program compiling.

V. Self-Configurable Computer Systems

We suggest to call self-configurable the computer system with reconfigurable logic, where program compilation includes automatically performed actions of configuration creation, and which acquires that configuration automatically during the program loading for execution [10].

In the Self-Configurable Computer System (SCCS), 1) execution of the computational load balancing between the general-purpose computer and RCE and 2) creation of the ASP's programming model are automated, and the method of information processing is improved in the way that loading the configuration files obtained after the logical synthesis into RCE is carried out not by user but by the operating system in parallel with loading the computer subprogram executable file into its main memory after the program initialization [10].

The diagram of the method of information processing in the SCCS is shown in figure 1. This method can be represented as a sequential execution of three stages: program compiling, loading and execution.

The user creates a program P_{in} written in a high level programming language and submits it into SCCS. SCCS during compiling automatically performs the following actions: divides this program into the computer subprogram P_{GPC} and the RCE subprogram P_{RCE}, performs computer subprogram P_{GPC} compilation, generates its executable file obj, creates ASP's HDL-model $ASPM$ to perform the RCE subprogram P_{RCE}, performs the ASP's logical synthesis, and stores the obtained executable file obj and the RCE configuration files $\mathbf{conf} = \{conf_q, q = \overline{1...K_{FPGA}}\}$ in computer memory, where K_{FPGA} is the number of FPGAs that form RCE.

In order to perform these actions the SCCS has to contain the following means:

1. A computational load balancing system for load balancing between the computer and RCE. This system should automatically select from the program P_{in} fragments, whose execution in RCE reduces its execution time, and divide the program P_{in} into the computer subprogram P_{GPC}, replacing the selected fragments in it by instructions for interaction with RCE, and the RCE subprogram P_{RCE}, formed from the selected fragments. An example of such system is described in [11]. This system creates the RCE subprogram in the x86 assembly language, thus it must be supported by the means for the assembly language code translation into the high-level language to be used in SCCS. The tool of this type is available on the market, for example Relogix Assembler-to-C Translator [12] from MicroAPL.

2. A generating system for the ASP HDL-model creation, which should automatically generate a model $ASPM$ from the RCE subprogram P_{RCE}, like Chameleon system from Intron, Agility Compiler and DK4 Design Suite from Celoxica, CoDeveloper from Impulse.

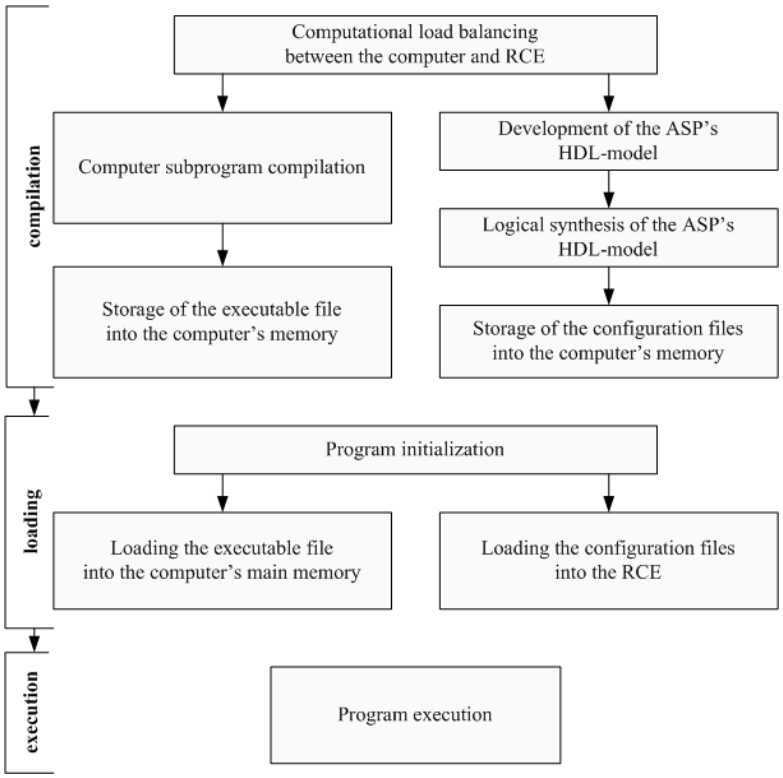

Figure 1: The diagram of the method of information processing in the SCCS.

3. The tools that are used in RCCS for performing the computer subprogram compilation and creation of its executable file, and for the logical synthesis of the ASPs.

At the stage of the program loading after its initialization, SCCS loads the executable file *obj* of the computer subprogram into its main memory using the standard loader and, at the same time, loads the configuration files $\mathbf{conf} = \{conf_q, q = \overline{1...K_{FPGA}}\}$ into RCE and, thus, creates an ASP in it. Then, the stage of the program execution is performed in the same way as in the RCCS. In order to perform these actions the same tools can be used in SCCS as in RCCS.

The structure of the self-configurable computer system that implements the proposed method of information processing is shown in figure 2. The term "self-configurable" against the FPGA-based computer system implies that here the computer system performs by itself all the steps of information processing after the program initialization, ranging from the load balancing between the computer and RCE up to obtaining the executable file of RCE, as well as the RCE configuring.

VI. SCCS Benefits

- **Ensured effective use of reconfigurable logic to perform arbitrary tasks** - loading of executable files and configuration files into the computer main memory and into RCE is, respectively, performed by the operating system after program initialization.
- **Shortened information processing time** - all the actions, starting from the load balancing and till obtaining the executable file and the configuration files, are executed automatically at the stage of program compiling without user's intrusion.
- **Reduced information processing complexity** - requirements to the user experience are simply reduced to knowing the high level programming language.
- **Simplified programming** - the user utilizes the ANSI C language that requires no additional constructions (parallel operators, directives etc.) and works with SCCS in the same manner as with the conventional personal computer.

VII. Background of SCCS

SCCS creation is the result of years of engineering and research work of the authors in the field of application-specific processors development, their ASIC-and FPGA-implementation, and creation of the methods and means for their high-level design. The background of SCCS is built on the:

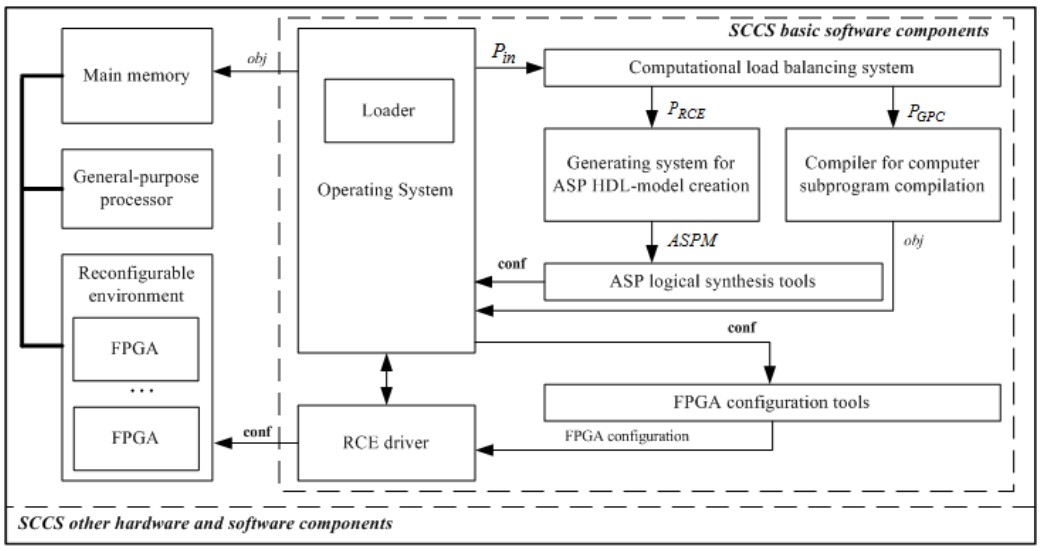

Figure 2: The structure of the self-configurable computer system.

1. ESL Design Tools and Solutions (developed in cooperation with Intron ltd (http://intron-innovations.com)):

   - Chameleon - System-Level Design Solution intended for the ASIC design automatic generation from the algorithm described in the ANSI C language. The developer, specifying an algorithm of the data processing on ANSI C, gets on return 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. Obtained VHDL design may be further implemented in FPGA by any FPGA design solution, e.g. the Xilinx ISE Web-PACK.
   - OSCAR - System-Level Design Solution intended for the VHDL design automatic generation from the algorithmic representation. The customer may select one of the following options for synthesizing the algorithmic computing device (ACD):
     (a) a single-stage ACD;
     (b) a multiple-stage ACD;
     (c) a pipeline ACD;
     (d) an ACD with the scalable layers/operators.
   - IP Cores Generator -high-level software tools, which allow IP Core with required characteristics to be obtained automatically on the basis of the scalable IP modules' sources. Some of them are the tools for generating: 1) the Fast Fourier Transformation IP Cores; 2) the Fast Cosine Transformation IP Cores; 3) the Data Encryption Standard IP Cores; 4) the Multiple-Block RAM Access Controller IP Cores [13], [14].

     Generator's input consists of the scalable parameterizable IP modules' sources and user-defined IP Core parameters from the given list. The generator output consists of the IP Core source HDL files; the IP Core test bench with the test patterns; text documentation etc.

2. Our new computer architecture.
3. Our new multiport computer memory with the parallel conflict-free data access [15].

VIII. What We Have Done Within the Project

- The theoretical principles of the SCCS design and operation have been developed, the characteristics of information processing duration in SCCS have been investigated and the SCCS's fundamental advantages over the reconfigurable ones have been proven.
- The method of information processing in SCCS for their single- and multi-processor implementations has been developed.
- The principles of the SCCS structural organization and the conceptual bases of their components design have been developed.
- The theoretical foundations of approaches to computational load balancing system design in SCCS and the method of computational load balancing between the general-purpose computer and the reconfigurable environment have been investigated. An instance of the computational load balancing system with the use of the LLVM compiler has been created.
- The high-level design tools have been developed allowing the application-specific processor VHDL IP-Cores to be generated on the basis of the ANSI C descriptions of their algorithms of

operation that are the advanced products for the system-level design.

IX. Current and Future SCCS Development

In our belief, SCCS represents definitely an emerging direction in developing the heterogeneous computer systems particularly and in the high-performance computing in general, and there is a need in further works in developing the SCCS theoretical basis, software and hardware design and implementation, testing, optimization etc. Some ongoing works on the SCCS creation are as follows:

- **Development of the SCCS operating system.** The goal of this work is to develop the theoretical background and the appropriate system software means for the SCCS operation during program compilation, loading for execution, and execution.
- **Load balancing in the SCCS.** The goal of this work is to develop the theoretical background and the appropriate software tools for balancing the computational algorithms specified by the input program between the general-purpose programmable processor and reconfigurable environment that fully explore the spatial and the temporal properties of the algorithm.
- **Automatic mapping of the soft-cores synthesized by the HLL2HDL tools into the target FPGA architecture** (the HLL - high-level programming language). The goal of this work is to develop the theoretical background and the appropriate software means for adaptation of the structure of the soft-cores synthesized by the HLL2HDL tools to that of the target FPGAs, as well as to integrate these tools with other software SCCS components.
- **SCCS based on the partially reconfigurable FPGAs.** The goal of this work is to improve the SCCS efficiency by developing the methods of the parallel execution of multiple computing tasks in their reconfigurable environment, to implement and investigate the self-configurable FPGA-based computer systems based on the partial dynamic reconfigurable FPGAs.

X. Conclusions

In this paper, we have described an emerging type of the heterogeneous computer systems, namely, the self-configurable FPGA-based computer systems.

The method of information processing in SCCS has been presented and the rules of application of the computer software and hardware tools necessary for its implementation have been described.

SCCS benefits are as follows: 1) an ensured effective use of the reconfigurable logic to perform arbitrary tasks; 2) a shortened information processing time; 3) a reduced information processing complexity; and 4) a simplified programming.

We have highlighted the background of SCCS, which is built on: 1) our ESL Design Tools and Solutions; 2) our new computer architecture; and 3) our new multiport computer memory with the parallel conflict-free data access.

Finally, we have shown some ongoing works on the SCCS creation, targeted on the development of the SCCS theoretical basis and their software and hardware components design and implementation.

References

[1] A. Melnyk, V. Melnyk, *Personal Supercomputers: Architecture, Design, Application*, Lviv Polytechnic National University Publishing, 2013. - 512 p.

[2] S. Hauck, A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*, Morgan Kaufmann, 2008. - 944 p.

[3] C. Bobda, *Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications*, Springer, 2010. - 352 p.

[4] T.J. Todman, G.A. Constantinides, S.J.E. Wilton, O. Mencer, W. Luk and P.Y.K. Cheung, "Reconfigurable Computing: Architectures and Design Methods," *IEEE Proceedings: Computer and Digital Techniques*, vol. 152, no. 2, March 2005, pp. 193-208.

[5] *IEEE Standard for Standard SystemC Language Reference Manual*, IEEE Std 1666-2011, 9 January 2012. - 638 p.

[6] "Catapult LP for a Power Optimized ESL Hardware Realization Flow," [Online]. *Available: http://calypto.com/en/blog/2012/11/10/catapult-lp-for-a-power-optimized-esl-hardware-realization-flow/* - 11.10.2012.

[7] "DIMETalk. FPGA Development Tools," [Online]. *Available: http://www.nallatech.com/FPGA-Development-Tools/dimetalk.html*

[8] "C-to-FPGA Tools form Impulse Accelerated Technologies. Impulse CoDeveloper C-to-FPGA Tools," [Online]. *Available: http://www.impulseaccelerated.com/products_universal.htm*

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

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

[11] 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.

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

[13] A. Melnyk, "Newest Computer Devices Design Technology on a Base of Configurable Models," in *Proceedings of First International Conference "Advanced Computer Systems and Networks"*, Lviv, Ukraine, Setember 2003, pp. 10-12.

[14] A. Melnyk, V. Melnyk, "IP Cores Generators in SoC Design," in *Proceedings of the 5th international Conference for Students and Young Scientists "Telecommunication in XXI Century"*, Wolka Milanowska, Poland, 24-26 November 2005, pp. 23-28.

[15] A. Melnyk, *Ordered Access Memory*, Lviv Polytechnic National University Publishing, 2014. - 296 p.