Self-Configurable FPGA-Based Computer Systems

Anatoliy MELNYK[1, 2], Viktor MELNYK[2]
[1]*Yuriy Fedkovych Chernivtsi National University, Kotsjubynskyi Str. 2, Chernivtsi 58012, Ukraine*
[2]*Lviv Polytechnic National University, Bandera Str. 12, Lviv, 79013, Ukraine*
aomelnyk@polynet.lviv.ua, vmelnyk@intron-innovations.com

***Abstract*—Method of information processing in reconfigurable computer systems is formulated and its improvements that allow an information processing efficiency to increase are proposed. New type of high-performance computer systems, which are named self-configurable FPGA-based computer systems and perform information processing according to this improved method, is proposed. The structure of self-configurable FPGA-based computer systems, rules of application of computer software and hardware means, which are necessary for these systems implementation, are described and their execution time characteristics are estimated. The directions for further works are discussed.**

***Index Terms*—field programmable gate arrays, high performance computing, reconfigurable architectures, reconfigurable logic, self-configurable computer systems.**

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 the high characteristics of modern field-programmable gate arrays (FPGAs) – 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-operation of based on general-purpose processors computer system with synthesized in RCE application-specific processors, whose structure takes into account an executed algorithms features, allows to increase its productivity by 2-3 orders of magnitude. Reconfigurability and ability to synthesize an application-specific processor (ASP) with a new structure and functions in RCE allows to change the functional commitment of created thereby RCCS with preserving its high performance at the new class of problems.

An architecture and organization of RCCS at various levels are described in [1]-[4]. By analyzing of these studies one can conclude about difficulty of information processing in such systems, what is a consequence of the need to perform the design and synthesis of ASPs in RCE before their use. ASP design is performed by its architecture describing on hardware design languages (HDL), VHDL or Verilog, with use of the register transfer level design tools, or even by specifying of its characteristics and an algorithm to be executed on high-level programming language, as it provide advanced electronic system level design tools, for example, C-to-Verilog [5], System-C [6] and Handel-C [7] from Celoxica, Catapult-C [8] from Calypto Design Systems, DIMETalk [9] from Nallatech, CoDeveloper [10] from Impulse Accelerated Technologies, and ROCCC2.0 [11] from Jacquard Computing. These tools enable creation of the HDL-descriptions of computing devices at register transfer level from programs represented on high-level programming language, often on modified C language. Technology and tools "Chameleon" [12], [13] from Intron, as well as SPARK [14], [15] from University of California, are intended to design ASPs using algorithm description on ANSI C language. The basic platform for ASPs creation here is a configurable processor architecture, which provides creation of its desired configuration with use of following configuration parameters: the number of functional units, the instruction set of each functional unit, capacity of 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 must be submitted in addition to the description of the algorithm.

One approach to design of ASPs from high-level programming language provides their implementation as programmable complemented instruction set processors, whose instruction set is complemented by bringing in items for the hardware implementation of computationally complex parts of the program of implemented in reconfigurable logic library [16], [17]. Here synthesis of ASPs is made by automatic selection of program fragments that have the most computational load, and their transformation into configuration code of reconfigurable logic, integrated with general-purpose processor.

Similarly works the computational load automatic balancing system, proposed in [18]. However, this system is used not for ASPs design, but for computational load balancing between components of RCCS – namely, general-purpose computer and RCE. In contrast to the above mentioned approach, where each fragment of input program is substituted by the instruction of general-purpose processor, here input program is divided into two programs, first of which is carried by general-purpose computer, and the second – by ASP.

II. PROBLEM STATEMENT

In order to perform ASPs design and their synthesis in RCE, to change configuration of RCE, and before that to balance the computational load between general-purpose computer (further in the text – computer) and ASPs of RCCS, there is necessity to involve significant human resources and time, what dramatically reduces the effectiveness of RCCS. Therefore, an important task is to

Digital Object Identifier 10.4316/AECE.2013.02005

improve RCCS in the way to use all its potential, given by ability of RCE reconfiguration. In order to do this it is necessary to identify and study the steps of information processing in RCCS, the problems impeding the growth of their effectiveness, and to find solutions to these problems, and that is the goal of research, represented in this paper.

III. METHOD OF INFORMATION PROCESSING IN RECONFIGURABLE COMPUTER SYSTEM

Information processing in RCCS can be represented as sequential execution of the four stages.

At the first stage user creates the program P_{in} written on high level programming language, divides this program into the subprogram P_{GPC} of computer and subprogram P_{RCE} of RCE, performs compilation of subprogram P_{GPC}, generates its executable file *obj* and stores it in the memory of computer. To perform these actions he uses the following tools:

1. In order to develop and debug the program, written on high level programming language, - integrated programming environments, such as Visual C + + [19] from Microsoft.
2. In order to balance the computational load between computer and RCE – the modeling tools or profilers, which provide the statistics of the program, including time and frequency of execution for its separate fragments. This enables detection at the program those fragments that require the largest amount of computations.
3. In order to compile the subprogram of computer and create its executable file – an arbitrary compiler from the language that subprogram is represented in into the object code that can be directly executed by computer.

At the second stage user develops (or uses ready-made solution) an HDL-model *ASPM* of ASP, which is intended to perform the subprogram P_{RCE} of RCE, performs logical synthesis of the ASP and loads configuration files $\mathbf{conf} = \left\{conf_q, q = \overline{1...K_{FPGA}}\right\}$ to RCE, where K_{FPGA} is the number of FPGAs, which form RCE, and thus creates an ASP in RCE. To perform these actions he uses the following tools:

1. In order to design and debug HDL-models of ASPs – integrated environments for design, modeling and verification of the projects described on hardware description languages, for example, ModelSIM from Mentor Graphics, Active-HDL from Aldec.
2. In order to perform logical synthesis of ASPs and RCE configuring – tools for hardware design in FPGA, for example, ISE, Alliance, Foundation from Xilinx, Quartus II, Max + II from Altera.

At the third stage, after program initialization, operating system loads the executable file *obj* of the computer subprogram to its main memory using the standard loader.

At the fourth stage RCCS executes program P_{in}. Computer executes its own subprogram P_{GPC}, RCE executes its own subprogram P_{REC}. Computer interacts with synthesized in RCE ASP under control of the operating system. In order to organize this interaction, the driver of RCE and hardware parts of the controllers of interfaces which this interaction is carried out through, should be used. These tools are provided by manufacturers together with RCE, for example, by DRC for their reconfigurable processor units RPU [20], by Nallatech for their reconfigurable accelerators of H100 series [21], by Celoxica for their reconfigurable accelerators RCHTX [22].

If the same program must be re-executed, user sequentially performs the third and fourth stages.

If he has an executable file and configuration files – he loads configuration files to RCE and further sequentially performs the third and fourth stages.

An execution time for information processing in RCCS according to a given program can be represented by expression:

$$T_{DP} = {}_U t^P_{distr} + t^{GPC}_{compile} + t^{obj}_{store} + {}_U t^{ASPM}_{develop} + {}_U t^{ASPM}_{synth} + {}_U t^{conf}_{load} + t^{obj}_{load} + t^P_{exe}, \quad (1)$$

where: ${}_U t^P_{distr}$ - duration of program distribution by user on computer subprogram P_{GPC} and RCE subprogram P_{RCE}; $t^{GPC}_{compile}$ - duration of computer subprogram compiling; t^{obj}_{store} - duration of computer subprogram executable file storing into its memory; ${}_U t^{ASPM}_{develop}$ - duration of ASP HDL-model designing; ${}_U t^{ASPM}_{synth}$ - duration of ASP logical synthesis; ${}_U t^{conf}_{load}$ - duration of configuration files loading into RCE; t^{obj}_{load} - duration of computer subprogram executable files loading into its main memory; t^P_{exe} - duration of RCCS program P_{in} executing. Index ${}_U *$ marks actions that are performed directly by the user.

Duration of information processing in the case of re-execution of the program can be represented by expression:

$$T_{DP'} = t^{obj}_{load} + t^P_{exe}. \quad (2)$$

If the computer subprogram executable file and RCE configuration files are available, the duration of information processing can be represented by expression:

$$T_{DP''} = {}_U t^{conf}_{load} + t^{obj}_{load} + t^P_{exe}. \quad (3)$$

IV. PROBLEMS THAT HINDER EFFECTIVE WORK OF RCCS

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

- in order to execute each new program in RCCS, all four stages, in two first of which all actions are performed by user, have to be sequentially performed, what requires significant amount of time;
- in order to execute the program when executable file and configuration files are available, user has to load configuration files into RCE before the third and fourth

stages, what also requires significant amount of time;
- list of algorithm, that RCCS is effective on, is narrow, and depends on the functional characteristics of implemented in RCE ASPs, and if the problems to be solved must be changed, then, at least, steps of the second stage of the method of information processing described above have to be performed;
- complexity of an information processing is high, because user, besides modeling and programming, should also perform systems analysis, design the ASPs architecture, perform their logical synthesis and implementation in FPGAs.

V. WAYS TO IMPROVE RCCS

A key approach to solve the above problems is the automation of all stages of information processing in RCCS. Let us analyze information processing in RCCS.

The development of the ASP's HDL-model at the second stage requires from user significant amount of time and knowledge of the system-level design technology. However, as mentioned above, there are software tools today that allow automatically create HDL-models of ASPs from high-level description of the algorithm to be implemented in. This software tools transform the algorithm, described on high-level programming language, into HDL-model of ASP. By linking the operations of ASP's HDL-model generation, ASP's logical synthesis and configuring of RCE in automatically executable sequence, one can load the configuration codes into RCE automatically without user intrusion.

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

1. to tie the operations of load balancing and compilation of computer subprogram in one startup sequence, and as result, to get executable file of subprogram without user intrusion;
2. to tie the operations of load balancing, generation of ASP HDL-model, ASP's logical synthesis and RCE configuration in one startup sequence, and as result, to load configuration codes into RCE automatically without user intrusion.

Consequently, automation of steps that are performed on 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:

- to reduce the execution time for information processing;
- to reduce the complexity of information processing since user no longer has to perform the systems analysis, design ASPs architecture, ASPs logical synthesis.

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

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

The proposed improvements of the method of information processing in RCCS are shown in the diagram, which is presented in Fig. 1.

As a result of those improvements of the method of information processing in RCCS, and automation of its first two stages, all the problems pointed above will be solved, because:

1. All 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 intrusion.
2. In the case of executable file and configuration files availability, loading of these files into computer's main memory and into RCE is respectively performed by the operating system after program initialization. Thereby the task of expanding the list of algorithms that RCCS are effective on is solved.
3. The requirements to the system user experience are simplified up to the knowing of high level programming language.

VI. CONCEPT OF THE SELF-CONFIGURABLE COMPUTER SYSTEM

We suggest it to call self-configurable the computer system that performs the information processing according to above illustrated method including the introduced improvements.

Self-configurable computer system (SCCS) consists of computer, RCE, which is based on FPGA (or other types of programmable logic), and a software. This software during program compilation solely allocates fragments there, whose execution in RCE speeds up the computer system, and divides the program into computer subprogram and RCE subprogram, which is formed from allocated fragments, compiles computer subprogram into executable file and RCE subprogram into file of its configuration, during program loading after program initialization solely creates an ASP in RCE to perform corresponding subprogram and, at runtime, self-organizes its functioning and interaction with computer.

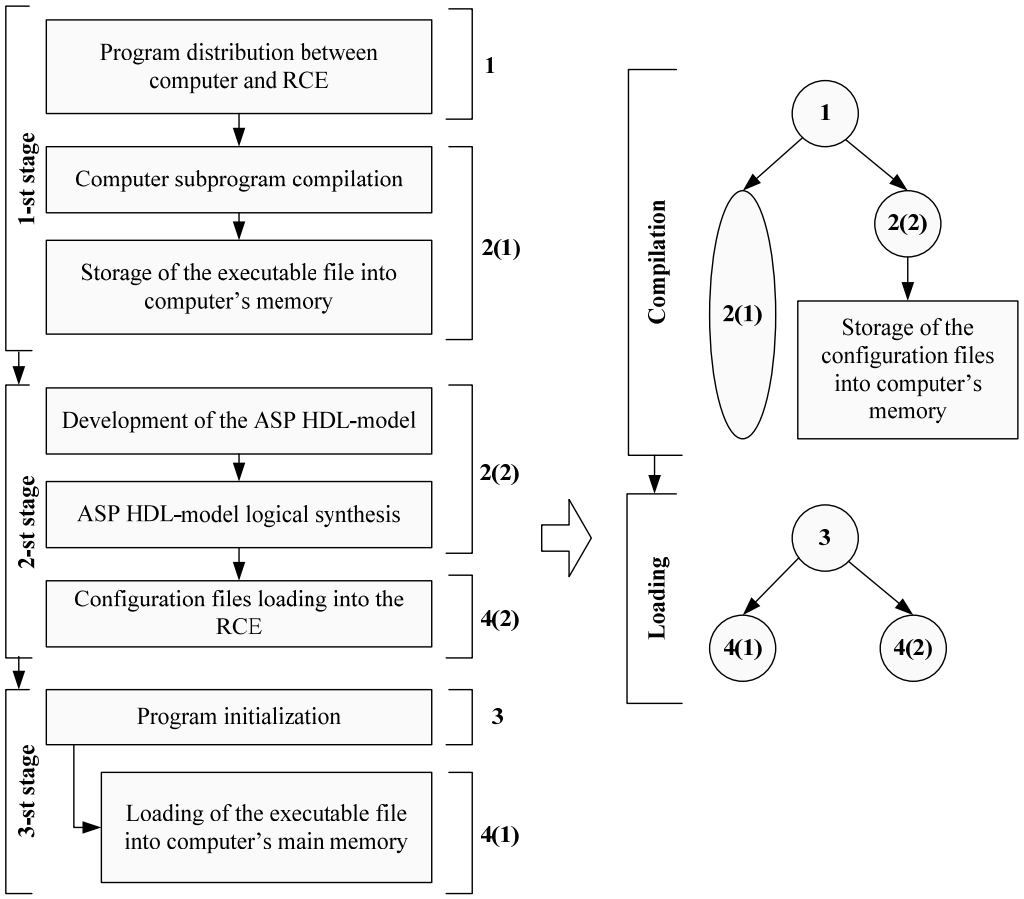

Figure 1. Diagram that shows how to improve the information processing in RCCS

The term "self-configurable" against the FPGA-based computer system is used for the first time, and implies that all the steps of information processing in there after program initialization, ranging from the load balancing between computer and RCE up to obtaining the executable file of RCE the computer system performs by itself, as well as RCE configuring.

VII. METHOD OF INFORMATION PROCESSING IN SELF-CONFIGURABLE COMPUTER SYSTEM

Method of information processing in SCCS can be represented as a sequential execution of three stages: compiling the program, its loading and execution.

User creates a program P_{in} written in high level programming language and submits it into SCCS. The SCCS during compiling automatically performs the following actions: divides this program into the computer subprogram P_{GPC} and RCE subprogram P_{RCE}, performs computer subprogram P_{GPC} compilation, generates its executable file *obj*, creates ASP's HDL-model *ASPM* to perform RCE subprogram P_{RCE}, performs ASP's logical synthesis, and stores in computer memory 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 that form RCE.

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

1. Computational load balancing system for load balancing between 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 selected fragments in there by instructions for interaction with RCE, and on RCE subprogram P_{RCE}, formed from the selected fragments. An example of such system is described in [18]. This system creates the RCE subprogram on x86 assembly language, thus it must be supported by the means for the assembly language code translation into 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 [23] from MicroAPL.
2. Generating system for ASP HDL-model creation, which should automatically generate a model *APPM* from the RCE subprogram P_{RCE}, like Chameleon system from Intron [12, 13], Agility Compiler [24] and DK4 Design Suite [25] from Celoxica, CoDeveloper from Impulse [10].
3. The tools that are used in RCCS for performing computer subprogram compilation and creation of its executable file, and for logical synthesis of the ASPs.

At the stage of program loading after its initialization SCCS loads the executable file *obj* of the computer subprogram into its main memory using 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

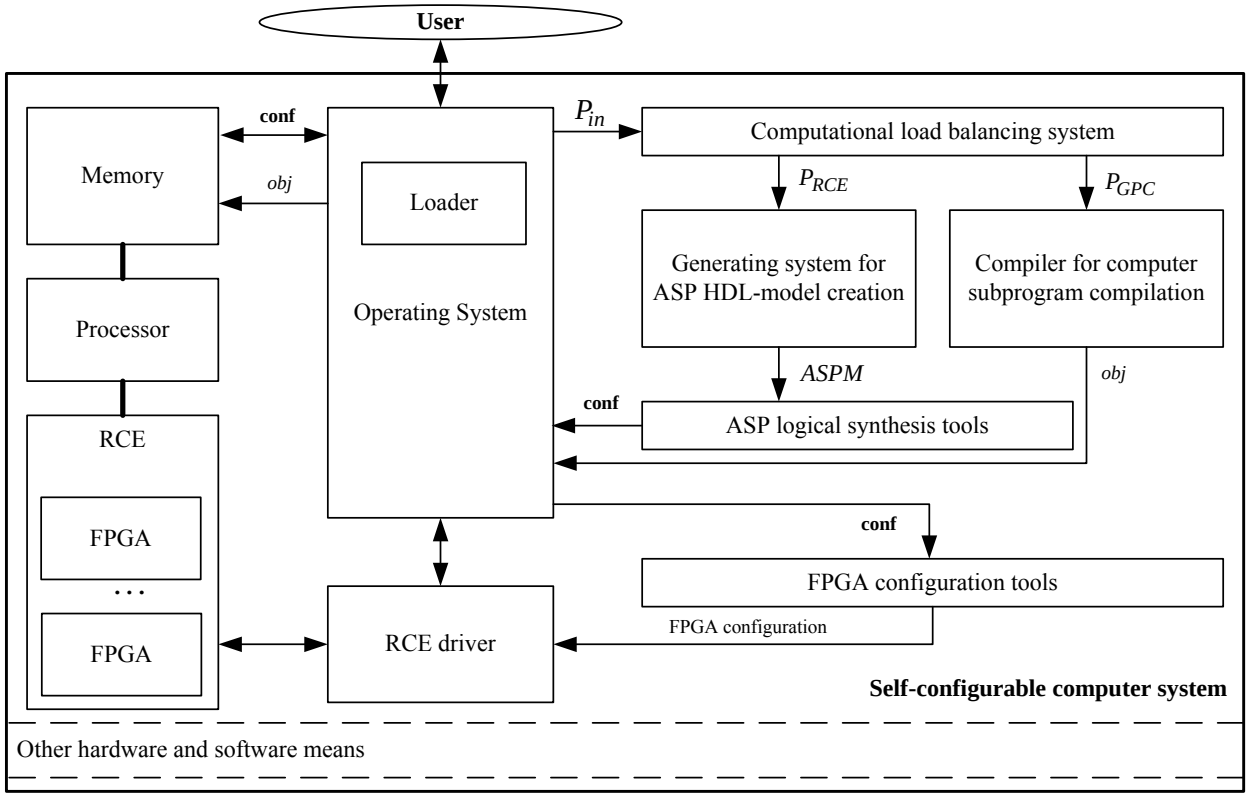

Figure 2. The structure of the self-configurable computer system

ASP in there. 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.

VIII. Structure and Characteristics of the Self-Configurable Computer System

The structure of the self-configurable computer system that implements the proposed method of information processing is shown in Fig. 2.From a user perspective, SCCS operates similarly to the traditional computer, as he, accordingly to the proposed method of information processing a) develops a program written in high level language and submits it into SCCS for compilation b) initiates a program after compilation; c) loads data to be processed and gets results. Thus, SCCS reconfigures itself accordingly to the features of described by computer program computational algorithm, unlike it does RCCS, where these actions are performed by user. The configuration is also here initiated by the operating system, not by user.

An execution time for information processing according to a given program in SCCS can be represented by expression:

$$T'_{DP} = \max\left(t^{GPC}_{compile} + t^{obj}_{store}, t^{ASPM}_{generate} + t^{ASPM}_{synth} + t^{conf}_{store}\right) + \\ + \max\left(t^{conf}_{load}, t^{obj}_{load}\right) + t^{P}_{distr} + t^{P}_{exe}, \quad (4)$$

where, in relation to the expression (1), value t^{conf}_{store} - duration of configuration files storing in computer memory – is added.

As can be seen from expression (4), significant reduction of the execution time for information processing in SCCS versus RCCS is ensured by:

- much less duration of automatic execution of four presented in expressions (1) and (2) actions than the duration of their execution by user, i.e. $t^{P}_{distr} <<_U t^{P}_{distr}$, $t^{ASPM}_{generate} <<_U t^{ASPM}_{develop}$, $t^{ASPM}_{synth} <<_U t^{ASPM}_{synth}$, i $t^{conf}_{load} <<_U t^{conf}_{load}$;
- parallel execution of several actions.

In two cases, namely, the re-execution of the program, and the presence of pre-formed executable file and configuration files, the duration of information processing in SCCS will be the same and can be represented by expression:

$$T'_{DP'} = \max\left(t^{conf}_{load}, t^{obj}_{load}\right) + t^{P}_{exe}. \quad (5)$$

In comparison with the expression (2) it can be noted that in case of repeated execution of the same program, if $t^{conf}_{load} > t^{obj}_{load}$ the duration of information processing in SCCS versus RCCS will increase. Therefore it is worth to explore in more details these two processes and, if necessary, develop the solution that will ensure the least possible value of t^{conf}_{load}.

IX. Directions of Further Work on SCCS Creation

A concept and principles of SCCS design were shown above in the paper. Therefore, we can highlight a list of further works on the SCCS creation. Into this list primarily could be included the following tasks:

- selection of the generation system for creation of the

ASP's HDL-model and its adaptation for use in SCCS;

- selection of the computational load balancing system and its adaptation for use in SCCS;
- development of the interaction models of SCCS software means according to the proposed method of information processing;
- trial operation of SCCS and its testing.

Implementation of these tasks will allow to use all the potential given by the property of self-configuration and will provide for SCCS one of the leading places among the most promising means of high-performance computing.

X. Conclusion

In this paper the method of information processing in RCCS is formulated and the rules of application of computer software and hardware tools, which are necessary for its implementation, are described.

It is shown that the main problems that hinder the effectiveness of RCCS are that in order to implement each new program four stages of information processing in there must be sequentially executed, two first of which are performed by user, firstly the computational load balancing and design of ASPs that require significant amount of time.

Performed analysis has shown that identified problems can be solved by automatic load balancing between computer and RCE, by automatic synthesis of ASPs HDL-models, and by improving the method of information processing in such way that the loading of configuration files into RCE is carried out by the operating system instead of user, and it is done at the same time as the loading of computer executable file into its main memory.

Basing on above investigations the new type of high-performance computer systems, which are named self-configurable computer systems, is proposed. Their operation is based on the method of information processing in RCCS with specified improvements. Thus the new method of information processing for SCCS is formed. The rules of application of computer software and hardware tools, which are necessary to implement this new method, are described.

For detail investigation of proposed solutions the structure of SCCS and the expressions for estimation of the duration of information processing in RCCS and SCCS are obtained.

The directions for further works on the SCCS creation, whose implementation will allow to use all the potential given by the property of self-configuration and provide for the SCCS one of the leading places among the most promising means of high-performance computing, are discussed.

References

[1] Scott Hauck, André DeHon. "Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation". Morgan Kaufmann, 2008. – 944 p.

[2] Melnyk A., Melnyk V. "Personal Supercomputers: Architecture, Design, Application". – Lviv National Polytechnic University Publishing, 2013. – 516 p.

[3] Christophe Bobda. "Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications". – Springer, 2010. – 362 p.

[4] T. Todman, G. Constantinides, S. Wilton, O. Mencer, W. Luk and P. Cheung. Reconfigurable Computing: Architectures, Design Methods, and Applications. IEE Proceedings on Computers and Digital Techniques 152 (2) pp.193-207 (2005).

[5] C-to-Verilog. Cirquit Design Automation. [Online]. Available: http://c-to-verilog.com/index.html.

[6] IEEE Standard for Standard SystemC Language Reference Manual. IEEE Std 1666-2011. 9 January 2012. – 638 p.

[7] Handel-C Synthesis Methodology – Mentor Graphics. [Online]. Available: http://www.mentor.com/products/fpga/handel-c/.

[8] 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.

[9] DIMETalk. FPGA Development Tools. [Online]. Available: http://www.nallatech.com/FPGA-Development-Tools/dimetalk.html

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

[11] ROCCC2.0. Jacquard Computing. [Online]. Available: http://www.jacquardcomputing.com/roccc/

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

[13] Chameleon – the System-Level Design Solution. [Online]. Available: http://intron-innovations.com/?p=sld_chame.

[14] S. Gupta, N.D. Dutt, R.K. Gupta and A. Nicolau. SPARK: a high-level synthesis framework for applying parallelizing compiler transformations. Proc. International Conference on VLSI Design, January 2003.

[15] Gupta, S., Gupta, R.K., Dutt, N.D., Nicolau, A.: SPARK: A Parallelizing Approach to the High-Level Synthesis of Digital Circuits. Kluwer Academic, Dordrecht (2004).

[16] Laurence H Cooke, Christopher E Phillips, Dale Wong: Method for compiling high level programming languages into an integrated processor with reconfigurable logic. Oct, 12 1999: US 5966534.

[17] Laurence H Cooke, Christopher E Phillips, Dale Wong: Method for compiling high level programming languages into embedded microprocessor with multiple reconfigurable logic. Intel Mar, 16 2004: US 6708325.

[18] V. Melnyk, V. Stepanov, Z. Sarajrech. System of load balancing between host computer and reconfigurable accelerator. Proceedings "Computer systems and components" of Tchernivtsi National University. – Tchernivtsi. 2012. T. 3. Ed. 1. pp. 6-16.

[19] Ivor Horton. Beginning Visual C++ 2005. – John Wiley & Sons., 2005. – 1224 p.

[20] DRC Computer Corporation. RPU100-L60 DRC Reconfigurable Processor Unit. A breakthrough in coprocessor technology. [Online]. Available: http://www.drccomputer.com/pdfs/DRC_RPU100_datasheet.pdf.

[21] H100 Series FPGA Application Accelerators. Version 1.9. September 2008. [Online]. Available: http://www.skyblue.de/nallatech/5595.pdf.

[22] Celoxica Ltd. RCHTX-XV4 High Performance Computing (HPC) Application Acceleration Board Datasheet. Version 1.0. 2006. [Online]. Available: http://www.hypertransport.org/docs/tech/rchtx_datasheet_screen.pdf.

[23] Relogix Assembler-to-C translator. [Online]. Available: http://www.microapl.co.uk/asm2c/.

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

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