ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 1, No. 1, 2016

TASKS SCALING WITH CHAMELEON© C2HDL DESIGN TOOL IN SELF-CONFIGURABLE COMPUTER SYSTEMS 3 BASED ON PARTIALLY RECONFIGURABLE FPGAS

Anatoliy Melnyk, Viktor Melnyk, Liubomyr Tsyhylyk

Lviv Polytechnic National University, 12, Bandera Str., Lviv, 79013, Ukraine
Author e-mail: *aomelnyk@lp.edu.ua*

Submitted on 24.03.2016

Abstract: **The FPGA-based accelerators and reconfigurable computer systems based on them require designing the application-specific processor soft-cores and are effective for certain classes of problems only, for which application-specific processor soft-cores were previously developed. In Self-Configurable FPGA-based Computer Systems the problem of designing the application-specific processor 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. 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 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.**

Key words: **self-configurable computer systems, field-programmable gate arrays, high-performance computing, reconfigurable computing, hardware multitasking, C2HDL design tools.**

I. INTRODUCTION

The self-configurable computer system is the computer system with reconfigurable logic, where program compilation includes automatically performed actions of creating configuration, and which acquires that configuration automatically in the time of program loading for execution [1]. The property of the self-configurability is intended to be employed not only in the general-purpose and high-performance computer systems with reconfigurable logic, but also in embedded and application specific ones. Particularly, it concerns the reconfigurable hardware platforms of the cyber-physical systems.

Implementation of the SCCS basing on partially reconfigurable FPGAs [2] enables organization of multiple-task execution in the reconfigurable environment [3], [4]. 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 there are 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 organize dynamic reallocation of the reconfigurable logic resources and replace some running application-specific processors with others performing the same tasks but differ 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* $\{ASPM_{opt}, \ldots, ASPM_{min}\}$; 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 [5], [6], 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 II shows the partially reconfigurable FPGAs operation basics. Section III shows programs execution features in SCCS based on partially reconfigurable FPGAs. Section IV introduces the characteristics and features of the Chameleon© C2HDL design tool. Section V 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 VI concludes the paper.

II. PARTIALLY RECONFIGURABLE FPGAS OPERATION BASICS

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 [2]). Also, this ability opens new possibilities for the reconfigurable logic application in computers, particularly it allows organizing hardware multitasking in FPGA [3], [4] and embodying the concept of Virtual Hardware [7], [8], 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 [9].

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 [10]. 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 [11] – CLB: 50×1; DSP48: 10×1; RAM: 10×1; in the UltraScale devices [12] – CLB: 60×1; DSP48: 24×1; RAM: 12×1.

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 [13].

III. HARDWARE MULTITASKING AND TASKS SCALING IN SCCS BASED ON PARTIALLY RECONFIGURABLE FPGAS

Deploying partially reconfigurable FPGAs in SCCS gives an opportunity to execute in these FPGAs a number of P_{RCE} subprograms simultaneously. Taking into account that duration of the partial reconfiguration versus the full one is significantly shorter, we can confidently say that it makes it possible to organize a real hardware multitasking in the SCCS reconfigurable environment.

The basic principles of hardware multitasking realization in SCCS based on the partially reconfigurable FPGAs are:

1. At the stage of the program compiling the HDL-model ASPM to perform PRCE subprogram, represented as a reconfigurable module, is being generated. At the same stage this reconfigurable module is being compiled into the FPGA partial configuration file.
2. At the stage of the program loading, after its initialization, partial configuration file is being loaded into the reconfigurable region specified by the SCCS operating system.
3. Partially reconfigurable FPGA comprises a number of reconfigurable regions for the reconfigurable module synthesis.
4. Single reconfigurable module may occupy one to NRP reconfigurable regions, where NRP is a total number of reconfigurable regions in the partially reconfigurable FPGA.
5. In the partially reconfigurable FPGA may simultaneously operate up to NRP reconfigurable modules.
6. If necessary, reconfigurable modules can use single reconfigurable region RRk in a time multiplexing mode. To replace modules in this reconfigurable region the context switching is performed.

Besides the reconfigurable regions, in the partially reconfigurable FPGA the auxiliary means for the hardware multitasking support also have to be placed, which will enable quick loading of the partial configuration and context switching. Together with the reconfigurable regions these means form the platform for hardware multitasking realization in the partially reconfigurable FPGA. The structure of such platform, which is based on the abovementioned principles of hardware multitasking realization, is shown in Fig. 1. The means for the hardware multitasking support are located here within the static region of FPGA, while the reconfigurable regions have predefined placement, dimensions and interfaces.

In order to organize a context switching, which should be done in a minimal runtime, the fast embedded memory blocks are used for short-term data storage – the partial configurations cache memory and the context memory. The context switching, as well as the partial configurations loading from the respected cache memory, performs the partial reconfiguration controller. Instructions memory includes programs for the partial configurations loading, context switching, which include context saving and restoring, and tasks relocation in the FPGA. Besides these programs, the controller provides overall control of the platform components.

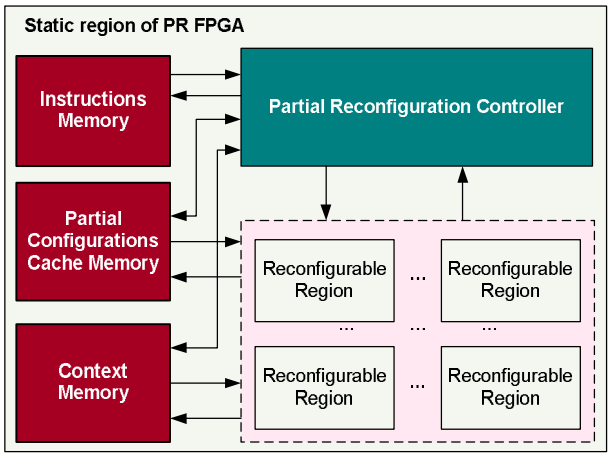


Fig. 1. The Structure of a platform for hardware multitasking realization in the partially reconfigurable FPGA

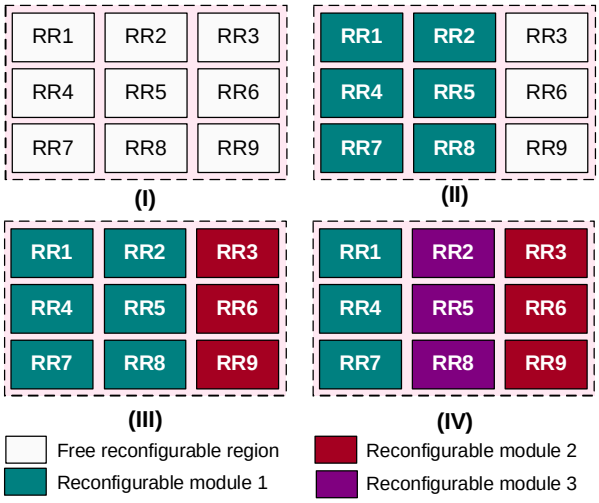


Fig. 2. Reconfigurable regions reallocations for the reconfigurable module 1 running in the FPGA in order to load the reconfigurable module 3

In the course of the SCCS operation, different reconfigurable modules performing different P_{RCE} subprograms are being loaded into the FPGA reconfigurable regions. Depending on the workload of the SCCS, the amount of available resources for one separate reconfigurable module 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 regions 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. An example in Fig. 2 shows the situation when the reconfigurable regions must be reallocated for the reconfigurable module 1 running in the FPGA in order to load the reconfigurable module 3.

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* {$ASPM_{opt}$, …, $ASPM_{min}$}; 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 have used 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. This provides tasks scaling in the reconfigurable environment. The Chameleon© C2HDL design tool characteristics and features are discussed below.

IV. CHAMELEON© C2HDL DESIGN TOOL

The Chameleon© C2HDL design tool is initially targeted for use in the heterogeneous FPGA-based computer systems. It is intended for the ASP's HDL-model automatic generation from the algorithm described in the ANSI C language [5], [6], [14]. The developer, specifying an algorithm of the data processing on ANSI C, gets in return 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 imple mented 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 boundary. The platform for the ASP synthesis is configurable processor architecture configured according to the following input parameters:

- desired performance (the number of parallel Functional Units),
- the width of data structure,
- the minimal percentage of commands that should load each parallel Functional Unit,
- the communication network structure.

The Chameleon© C2HDL design tool features are following:

1. Short generation time. For example, generation of the FFT 64-points processor VHDL model with 15 parallel Functional Units takes several seconds 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.

The architecture of the ASP is tested and verified automatically, 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. The basic scheme of the Chameleon© C2HDL design tool operation is shown in Fig. 3.

V. 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-points Fast Fourier Transformation in the ANSI C language, its code is given in Fig. 4. 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 13 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 1×1.

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 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.

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 Fig. 5). The main part of the generation time is spent on the algorithm parallelization and schematic optimization.

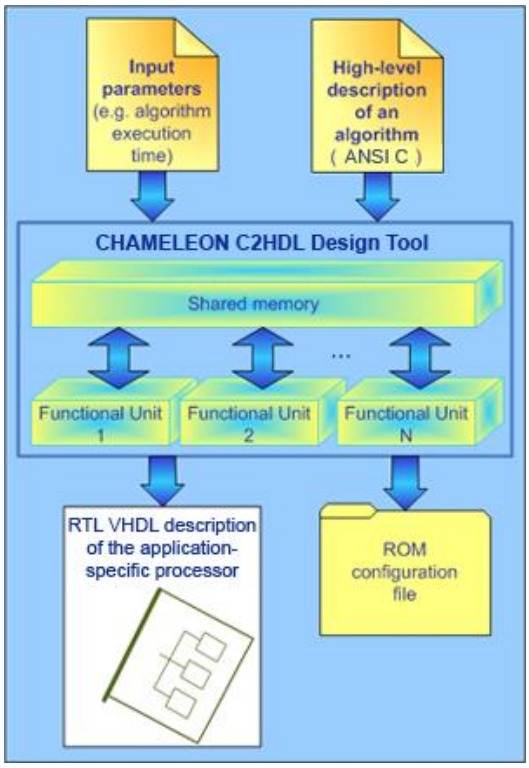


Fig. 3. Basic Scheme of Chameleon© C2HDL Design Tool Operation

```
#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); } }
```

Fig. 4. Program of 64-Point FFT Algorithm in ANSI C

In Fig. 6, the dependencies of the FFT execution time and the amount of the reconfigurable logic resources to the number of the Functional Units (regarded as the Parallel ALUs) are shown. 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.

Table 1

Technical Characteristics of FFT Processors

Number of the Functional Units	LUT utilization	Maximum Frequency (MHz)	Commands count	FFT time (us)
1	1, 809 / 32, 070 (6 %)	204, 08	1900	9, 31
2	2, 380 / 32, 070 (7 %)	200, 32	1015	5, 07
4	3, 054 / 32, 070 (10 %)	216, 8	575	2, 65
7	4, 806 / 32, 070 (15 %)	174, 52	388	2, 22
8	4, 858 / 32, 070 (15 %)	190, 73	352	1, 85
10	6, 715 / 32, 070 (21 %)	171, 85	311	1, 81
13	9, 200 / 32, 070 (29 %)	149, 7	190	1, 27
15	10, 198 / 32, 070 (32 %)	138, 48	180	1, 30

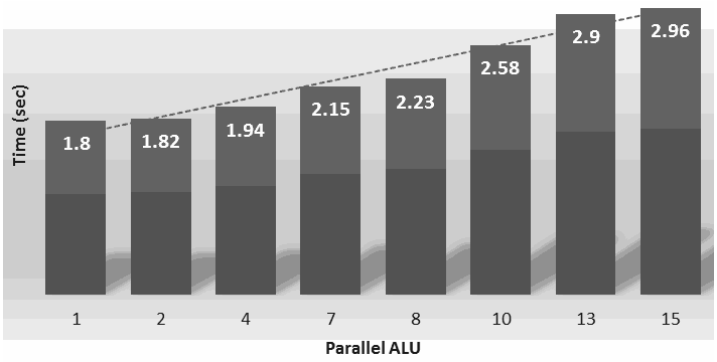
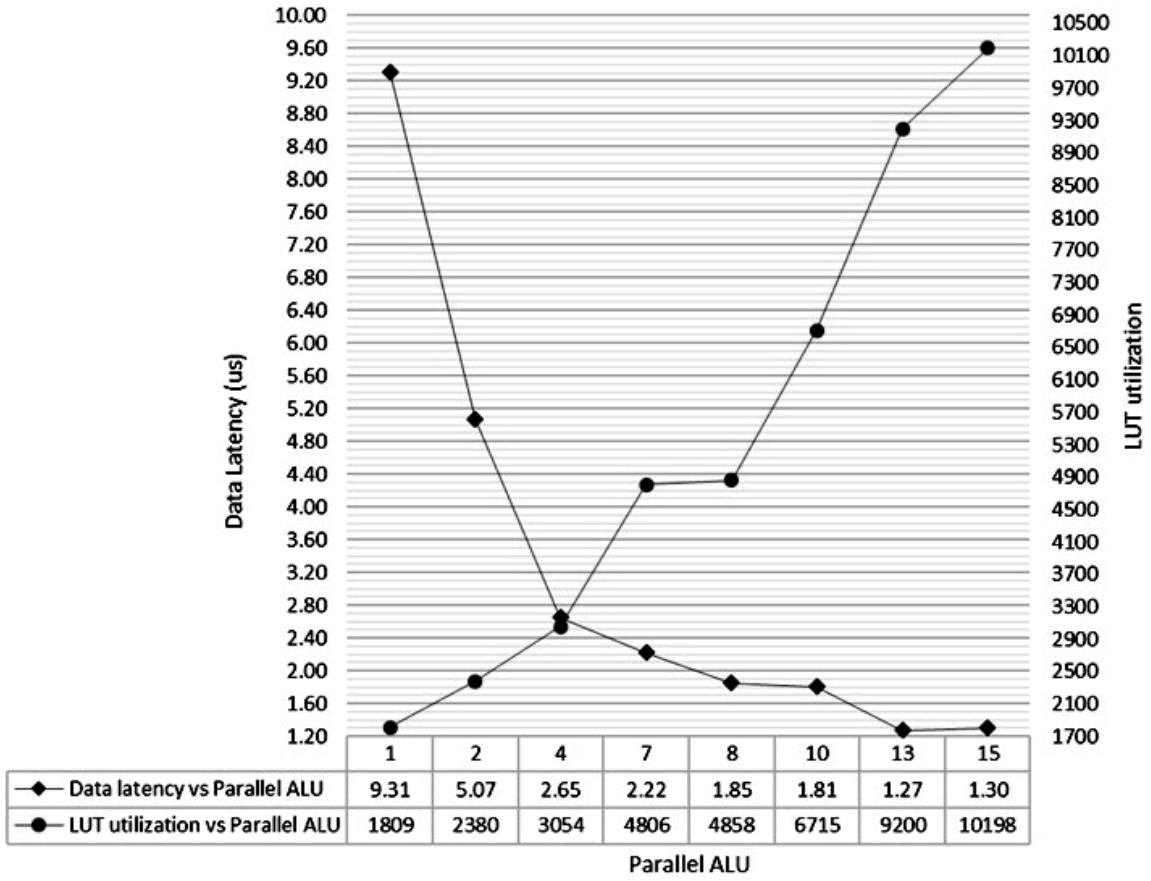


Fig. 5. Time Required for 64-Points FFT Processors VHDL-Models Generation



	1	2	4	7	8	10	13	15
Data latency vs Parallel ALU	9.31	5.07	2.65	2.22	1.85	1.81	1.27	1.30
LUT utilization vs Parallel ALU	1809	2380	3054	4806	4858	6715	9200	10198

Fig. 6. FFT Execution Time and LUT Usage Dependency vs Number of Functional Units for 64-Point FFT Processors

VI. CONCLUSIONS

Implementation of the SCCS based 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 is 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. The question arises to organize dynamic re-allocation of the reconfigurable logic resources and replace 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. We highlight the partially reconfigurable FPGAs operation basics. We identify the basic principles of hardware multitasking realization in SCCS based on the partially reconfigurable FPGAs propose the structure of a platform for hardware multitasking realization in the partially reconfigurable FPGA. We also consider 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-points 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-points 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.

REFERENCES

[1] Melnyk, A., Melnyk, V., "Self-Configurable FPGA-Based Computer Systems" Advances in Electrical and Computer Engineering, vol. 13, no. 2, pp. 33–38, 2013, doi:10.4316/AECE.2013.02005. [Online]. Available: http://www.aece.ro/abstractplus.php?year=2013&number=2&article=5

[2] E.J.McDonald, Runtime FPGA Partial Reconfiguration. Aerospace Conference, 2008 IEEE, Los Angeles, 2008, pp. 1–7.

[3] X. Iturbe, K. Benkrid, T. Arslan, R. Torrego, and I. Martinez, "Methods and mechanisms for hardware multitasking: executing and synchronizing fully relocatable hardware tasks in Xilinx FPGAs" inProceedings of the 21st International Conference on Field Programmable Logic and Applications (FPL '11), pp. 295–300, September 2011.

[4] H. Kalte and M. Porrmann, Context Saving and Restoring for Multitasking in Reconfigurable Systems, Proc. of the International Conference on FieldProgrammable Logic and Applications, pp. 223–228, 2005.

[5] A. Melnyk, V. Melnyk. "Personal Supercomputers: Architecture, Design, Application". Lviv Politechnic National University Publishing. – 2013. – 516 pp.

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

[7] G. J. Brebner, A Virtual Hardware Operating System for the Xilinx XC6200. Proc. of the International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers, 1996.

[8] G. Brebner. The swappable logic unit: a paradigm for virtual hardware. In K. L. Pocek and J. M. Arnold, editors, The 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines (FCCM'97), pages 77–86, Los Alamitos, CA, Apr. 1997. IEEE Computer Society Press.

[9] Virtex-4 Configuration Guide, Xilinx, Inc. http://www.xilinx.com/support/documentation/user_guides/ug071.pdf

[10] P. Sedcole, B. Blodget, T. Becker, J. Anderson, and P. Lysaght. Modular dynamic reconfiguration in Virtex FPGAs. IEE Proceedings Computers and Digital Techniques, 153(3):157–164, 2006.

[11] Vivado Design Suite User Guide. Partial Reconfiguration. UG909 (v2015.2) June 24, 2015. Online. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_2/ug909-vivado-partial-reconfiguration.pdf

[12] UltraScale Architecture. Online. Available: http://www.xilinx.com/products/technology/ultrascale.html

[13] Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs. July 2010, Altera Corporation. Online. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01137-stxv-dynamic-partial-reconFig. pdf.

[14] 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. No. 5, P. 189–195.

Anatoliy Melnyk since 1994 is a Head of Computer Engineering Department at Lviv Polytechnic National University. He graduated from Lviv Polytechnic Institute with the engineer degree in computer engineering in 1978. In 1985 he obtained his Ph.D. in Computer Systems from Moscow Power Engineering Institute. In 1992 he received his D.Sc. degree from the Institute of Modeling Problems in Power Engineering of the National Academy of Science of Ukraine. He was recognized for his outstanding contributions to high-performance computer systems design as a Fellow Scientific Researcher in 1988. He became a Professor of Computer Engineering in 1996. Since 1982 to 1994 he has been a Head of Department of Signal Processing Systems at Lviv Radio Engineering Research Institute. Since 1994 to 2008 he has been Scientific Director of the Institute of Measurement and Computer Technique at Lviv Polytechnic National University. Since 1999 to 2009 he has been Dean of the Department of Computer and Information Technologies at the Institute of Business and Perspective Technologies, Lviv, Ukraine. He has served since 2000 as President and CEO of Intron ltd. He has also been a visiting professor at Kielce University of Technology, University of Information Technology and Management, Rzeszow, University of Bielsko-Biala. Currently he is a visiting professor at the Department of Numerical Analysis and Programming of John Paul II Catholic University of Lublin.

He is an editor in chief of the proceedings "Computer Systems and Networks" and of the journal "Advances in Cyber-Physical Systems". He is a head of the international conference "Advanced Computer Systems and Networks: Design and Application" and of the scientific workshop "Cyber-Physical Systems: Achievements and Challenges". He has taken part as a project leader in a large number of research projects in the field of computer systems. He has published 9 monographs, 1 handbook and over 400 scientific papers and patents. He is a member of IEEE, ACM, IEE, IACSS, AESU.

Viktor Melnyk is a professor in the Department of Information Technologies Security of Lviv Polytechnic National University in Ukraine. He was awarded with his Ph.D in 2004 and Doctor of Technical Sciences in 2013 at Lviv Polytechnic National University. He has gained scientific, academic and hands-on experience in the field of computers and computer systems research and design, proven contribution into IP Cores design methodology and high-performance reconfigurable computer systems design methodology. He is experienced in computer data protection, including cryptographic algorithms, cryptographic processors design and implementation, wireless sensor network security. Mr. Melnyk is an author of more than 70 scientific papers, patents and monographs.

Liubomyr Tsyhylyk is an assistant in the Department of Computer Engineering of Lviv Polytechnic National University. He has gained hand-on and scientific experience in the field of IP core design using VHDL, developing Touch solutions using microcontrollers, research of methodology for automation generic reconfigurable computer systems. Mr. Tsyhylyk is an author of over 10 scientific papers.

ACKNOWLEDGEMENT

The scientific results, presented in this article, were obtained within the frame of research project number 0115U000446, 01.01.2015–31.12.2017, financially supported by the Ministry of Education and Science of Ukraine.