13. Esche M., Thiel F. Software Risk Assessment for Measuring Instruments in Legal Metrology // Proceedings of the 2015 Federated Conference on Computer Science and Information Systems. 2015. Vol. 5. P. 1113–1123. doi: 10.15439/2015f127
14. Software risk assessment and evaluation process (SRAEP) using model based approach / Sadiq M., Md. Khalid Imam Rahmani Mohd. Wazih Ahmad, Jung S. // 2010 International Conference on Networking and Information Technology. 2010. doi: 10.1109/icnit.2010.5508535
15. Jacobson J. Validation of software in measuring instruments // Computer Standards & Interfaces. 2006. Vol. 28, Issue 3. P. 277–285. doi: 10.1016/j.csi.2005.07.006
16. Thiel F., Grottker U., Richter D. The challenge for legal metrology of operating systems embedded in measuring instruments // OIML Bull. 2011. Vol. 52, Issue 1. P. 5–14.

UDC 004.272.26, 004.274
DOI: 10.15587/1729-4061.2018.127361

FORMALIZATION OF THE CONCEPT OF ADAPTIVE TASKS MAPPING IN THE RECONFIGURABLE COMPUTERS ON FPGA

I. Klymenko
Doctor of Technical Sciences, Associate Professor*
E-mail: ikliryna@gmail.com
V. Tkachenko
PhD, Associate Professor*
E-mail: tkavalivas@gmail.com
A. Serhienko
Postgraduate student**
E-mail: ananserr@yahoo.com
Y. Kulakov
Doctor of Technical Sciences, Professor*
E-mail: ya.kulakov@gmail.com
*Department of Computer Engineering***
Department of System Programming and Specialized Computer Systems*
***National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Peremohy ave., 37, Kyiv, Ukraine, 03056

Запропоновано формальні моделі основних етапів обробки даних в реконфігуровних комп'ютерних системах, що враховують вплив затримок передавання конфігураційних даних на ефективність обчислень та дозволяють оцінити і оптимізувати непродуктивні витрати часу на реконфігурацію обчислювального середовища на ПЛІС. Запропоновано формалізацію концепції адаптивного відображення алгоритмів на реконфігуровне обчислювальне середовище в режимі часу виконання, що базується на багаторівневому кешуванні конфігураційних даних

Ключові слова: реконфігуровні комп'ютерні системи, часткова динамічна реконфігурація, накладні витрати реконфігурації, відображення алгоритмів

Предложены формальные модели основных этапов обработки данных в реконфигурируемых компьютерных системах, учитывающие влияние задержек передачи конфигурационных данных на эффективность вычислений и позволяющие оценить и оптимизировать непроизводительные затраты времени на реконфигурацию вычислительной структуры на ПЛИС. Предложена формализация концепции адаптивного отображения алгоритмов на реконфигурируемую вычислительную среду в режиме времени выполнения, которая основана на многоуровневом кэшировании конфигурационных данных

Ключевые слова: реконфигурируемые компьютерные системы, частичная динамическая реконфигурация, накладные расходы реконфигурации, отображение алгоритмов

1. Introduction

The result of the high level of modern progress is that extensive technologies of increasing the performance of high-speed processing reach their extreme opportunities. This is confirmed by violations of Moore's law in recent years [1]. The desire to increase performance further to the *exaflops* level makes the challenge to find the new intensive solutions. The dynamically reconfigurable computer systems [2–4] are one of such solutions. Their creation became possible on the base of modern technology of the partial dynamic reconfiguration (PDR) of the FPGA [3]. This direction is rapidly expanding today. The most advanced classes of tasks, which are solved in the dynamically reconfigurable computers, are the tasks of real-time control, particularly in undefined conditions, which have informational, multidimensional and dynamical nature [5–8].

In contrast to the static reconfiguration [9, 10], the dynamical reconfiguration is a precondition for the creation of the computing structures, adapted to the requirements of *Run Time* mode tasks solution [3, 11–13]. Firstly, this allows to overcome the limitations of the firm architecture of the high-speed multipurpose computer systems and to approach the real processing performance to the declared peak performance in the critical classes of tasks. Secondly, this allows to extend significantly the functional capabilities of the reconfigurable computer systems based on FPGA and to implement the complex computational structures on the limited chip area without huge cost increase [14]. The opportunity of reducing energy consumption based on using PDR technology is also actual [13, 15, 16].

The known technologies of the parallel processing of data are oriented to the inflexible computational environment without any functional and volume limitations, which are typical for the dynamically reconfigurable computers. Therefore, the solved problem deals with the development of the theory of organization of parallel computing in the reconfigurable computers, which have functional and hardware capabilities limitations, caused by the use of the dynamically programmable element base.

2. Literature review and problem statement

In the known dynamically reconfigurable computers, the modifications of traditional dynamic scheduling algorithms are widely used to solve the problem of the algorithm mapping into the reconfigurable computational environment. These modifications are developed for the inflexible computational environments and they function at the level of operating system add-on software [17]. The widely used method is the embedding of different overheads reducing methods into the known algorithms of tasks scheduling and distribution. Thus, in [12] the method of computational resources reuse in the dynamically reconfigurable computers in the Run Time mode is described. In [2, 18], the methods of reconfigurable computational resources placement based on the beforehand reconfiguration are proposed. The overhead problem is localized on the level of the reconfigurable computer structure. So, the algorithm mapping into the reconfigurable environment at the operating system level complicates substantially the consideration of the FPGA gate parameters and it has the redundant time and performance costs [17]. Besides, a set of papers in this direction [12, 18] does not take into account the problem of the FPGA hardware limitations.

The problem of overcoming the FPGA hardware limitations is usually solved by the methods, which are embedded into the schemes of the algorithms mapping too. Such methods are the FPGA computational space defragmentation, described in [12, 19], the schemes of upsets of task processing [20] and releasing the non-critical configurations, which are considered in [19, 21]. These methods are effective ones in the statically reconfigurable computers as well. In the dynamically reconfigurable computers, the use of defragmentation and configurations releasing causes the additional uncontrolled time and performance costs, which could not be evaluated during the optimization.

The application-specific methods and tools for the scheduling of reconfigurable computational resources and algorithms mapping, which are developed directly for the dynamically reconfigurable computers, are the most efficient ones. Such methods and tools simplify the use of the FPGA gate-level parameters during the task allocation. They take into account the reconfiguration overheads and FPGA hardware limitations.

In [11], the authors propose their own algorithm for reducing the time and energy consumption overheads, based on the known EDF (Earliest Deadline) algorithm for dynamical scheduling. The proposed algorithm solves the task of the hardware task allocation optimization based on the fragmentation scheduling and the beforehand reconfiguration of the FPGA computational environment. In [15], the heuristic task mapping algorithm, which generates the minimized task assignment taking into account the overloading, the task of overheads minimization, the priorities and relations between the tasks is proposed. In [13], the scheme of the configuration data caching for overheads reducing is described. The proposed approach is based on the use of FPGA inner memory for the caching of configuration data. It allows reducing the time costs of the configuration data downloading and energy consumption. The authors do not take into account the critical limitation of the FPGA inner memory volume. Therefore, the proposed method is not efficient for the huge number of performed tasks. The algorithms, proposed by their authors, are mostly oriented to solving the task of optimal hardware task placement on the FPGA surface from the point of view of minimization of the time overheads and energy consumption.

In [4], the author proposes the scheduling and mapping algorithm for the specific hardware operating system of the self-reconfigurable computational system. Various methods of the algorithms mapping optimization are applied, including the scheme of beforehand reconfiguration. The hardware implementation of the application-specific operating system allows solving in the most effective way the task of algorithms mapping but limits the computational system functionality. The research of this direction is described in detail in [17].

The overheads during the reconfiguration of the computational environment and hardware limitations of FPGA are the important problem, which nowadays blocks the intensive development of the reconfigurable computing. Therefore, the development of new methods and tools for algorithms mapping into the flexible computational environment, taking into account the functional and hardware limitations of reconfigurable computers, is needed.

3. The aim and objectives of the study

The aim of the study is the improvement of efficiency of the process of tasks mapping to the reconfigurable computing structure of the dynamically RC by the dynamic adaptation to parameters of the changing computing environment considering the overhead time and FPGA hardware limitations.

To achieve the aim, the following tasks were accomplished:

– development of the concept of adaptive tasks mapping in the dynamically RC to minimize an overhead time considering FPGA hardware limitations and parameters of computing environment changing during the process of mapping;

– formalization of the process of tasks mapping in the dynamically RC to estimate an overhead time at all levels of the system;

– development and estimation of data processing in the dynamically RC during the adaptive tasks mapping based on criteria of minimization of overhead time considering FPGA hardware limitations.

4. Materials and methods of the research of adaptive tasks mapping in the dynamically RC

4. 1. The research objects and equipment used during the research

Simulation and research of the concept of reconfigurable computations based on functional blocks of hardware tasks, synthesized in Verilog hardware language and implemented on the Altera Cyclone II EP2C35F672C6 FPGA. Altera Development Kit DE2 board was used for verification and research of functional blocks' time characteristics.

The research was conducted for a series of computational algorithms presented by the MDG macrographs. Graphs of algorithms with different numbers of similar tasks and different values of correlation were used in the research. Algorithms for the research are randomly synthesized on the basis of the developed library of hardware implementations of functional cores that correspond to certain macrofunctions.

4. 2. Mathematical models of adaptive tasks mapping into the dynamically RCs

4. 2. 1. Modification means of determination of RCs effectiveness criteria

Computational algorithms with mixed type of parallelism are considered as initial algorithms. This allows to transform their graphs of algorithms to configure the most efficient task-oriented computing structures with a high processing speed in the reconfigurable environment. Computational algorithms with a mixed type of parallelism are presented by Macro Dataflow Graphs (MDGs) in the nodes of which macrofunctions (M-functions) are placed [22].

The computational algorithm is presented by an MDG

$$G_M=\{N_G, D_G\},$$

where N_G – the set of nodes corresponding to the M-functions; D_G – the set of edges that determine the relationship between M-functions, $N_M=\{N_1,N_2,...,N_i,...,N_g\}\big|i=\overline{1,g}$ – the set of M-functions in the graph nodes; g – the number of M-graph nodes, $W_k\big|k=\overline{1,w}$ – the identifier of the M-graph tier; w – the number of M-graph tiers. Each M-function at the nodes of the M-graph corresponds to a task that is a definite monatomic process of the computation. Each task is put into *hardware task* compliance (HW task), which is determined by the vector [12]

$$Task_j=\{S_j,T_{SUM\,j}\,|(T_j+R_j),\ I_j,\ N_i\},\tag{1}$$

where S_j – the area of the rectangle that contains the HW task $Task_j$ on the reconfigurable FPGA space surface; R_j – the time of the configuration data loading and HW task configuration on the FPGA to execute corresponding tasks; $T_j=T_{HWj}$ – the HW task execution time on the FPGA device, taking into account the time of data input-output for computation; T_{SUMj} – total execution time of the task N_i, $I_j\big|j=\overline{1,m}$ – the computation M-function implemented by the HW task; m – the number of HW tasks synthesized and stored in the Configuration Data Library (CDL).

The main efficiency criteria of RCs are determined on the basis of the known acceleration index [20]. Using it for the dynamically RCs is ineffective because it does not consider additional time expenditures due to the high complexity of the planning processes implementation for reconfigurable resources and tasks mapping, which is due to the need to take into account the physical parameters and constraints of the FPGA chips and associated changing display conditions. In order to determine the dynamically RCs performance criteria, an improved acceleration indicator has been proposed [23], which, in contrast to the well-known [20], takes into account the time complexity of parallel control scheduling processes and allocation of reconfigurable computational resources:

$$\rho=\frac{T_{SW}}{[T_{CONTROL}+R]+T_{HW}},\tag{2}$$

where T_{SW} – the time of task execution on the processor core; T_{HW} – the time of the HW task execution on the FPGA device; R – reconfiguration time of the FPGA area for the HW task, $T_{CONTROL}$ – timing complexity of the process of tasks mapping on the reconfigurable FPGA space. Time of computational FPGA space reconfiguration determines the overhead time of data processing. The sum ($T_{CONTROL}$+R) determines the overhead time of tasks mapping on the reconfigurable FPGA space.

Based on the modified efficiency criterion (2), the target function of reducing the overhead of the tasks mapping process, which depends on the delays that accompany this process was obtained:

$$\min\left(T_{CONTROL}+\sum\nolimits_j R_j\right)=$$
$$=\min\left(T_{CONTROL}\right)+\sum\nolimits_j \min\left(T_{COMM\,j}\right)+\sum\nolimits_j T_{CONFIG\,j},\tag{3}$$

where T_{COMM} – the transmission time of configuration data from external storage to FPGA interfaces; T_{CONFIG} – the configuration time of the computing FPGA area, while $R=$ $=T_{COMM}+T_{CONFIG}$. Configuration time depends on the technical characteristics of the chip and it is determined by $T_{CONFIG}=T_{INI}+N_{BIT}+T_W$, where T_{INI} – the time of reading and processing the first configuration word from the bitstream; N_{BIT} – the number of configuration words in the bitstream; T_W – the transmission time of one data word through the FPGA configuration interface [19].

The basic efficiency criteria of execution of computational algorithms in the dynamically RCs are determined, according to which the length of the critical path and the width of the MDG are estimated by the following relations:

$$(R+T_{HW})<T_{SW};\quad \max[H_k]\,|\,k=\overline{1,w}\le n,\tag{4}$$

where H_k – the number of nodes in the MDG tier; W_k; $k=\overline{1,w}$ – tier index; w – the number of tiers, $h=\overline{1,H_k}$ – the number of the node on the tier k; n – the number of HW tasks on the FPGA.

4. 2. 2. Main characteristics of the structural level organization of the dynamically RCs

Moving the process of tasks mapping from the operating system level to the structural level of Reconfigurable Computer Unit (RCU) was proposed to effectively solve the problem of adaptive tasks mapping. The research of the effectiveness of the realization of control processes on the various abstraction levels of RCs was carried in the authors' previous work [17]. On the base of it, the conclusion was made that the most effective is localization of the control means of FPGA space reconfiguration on the RCU local level. According to the results of the research, the most effective were the specialized hardware means for reconfiguration control.

In the authors' previous work [24], the decentralized hardware control means of tasks mapping that are localized on the structural level of RCU of the dynamically RCs were proposed. These means are based on parallelization of the control process and organization of multilevel multifunction cash memory at the RCU local level. The proposed organization of the structural level of the dynamically RCs gives an opportunity for an objective estimation of the time complexity of this process and its optimization. It is difficult to influence on the time of FPGA chip configuring, because this time depends on the technical characteristics of the chip. However, the functional features of reconfigurable computing systems implementation provide extensive opportunities for reducing communication delays during the reconfiguration. The nature of the emergence of communication delays is influenced by the following factors: the structure of the computer system; address space organization; communication environment structure; configuration data location; configuration data amount. Communication delays determine the time spent on the configuration data transmission from remote libraries to the FPGA chip interfaces, which precedes the programming process of the FPGA area. This time defines an unproductive part of the reconfiguration time and it is a critical criterion of the effectiveness of the dynamically reconfigurable computations.

time component will be removed from the critical path, and the other part is moved to the previous tiers of the MDG.

When comparing the diagrams on the left and on the right (Fig. 1), it is seen that the large part of the overhead time is deleted from the critical time of MDG by means of moving the reconfiguration process to the previous tiers. As a result of such transformation, the critical time of MDG will be determined only by the productive execution time of the tasks of every tier.

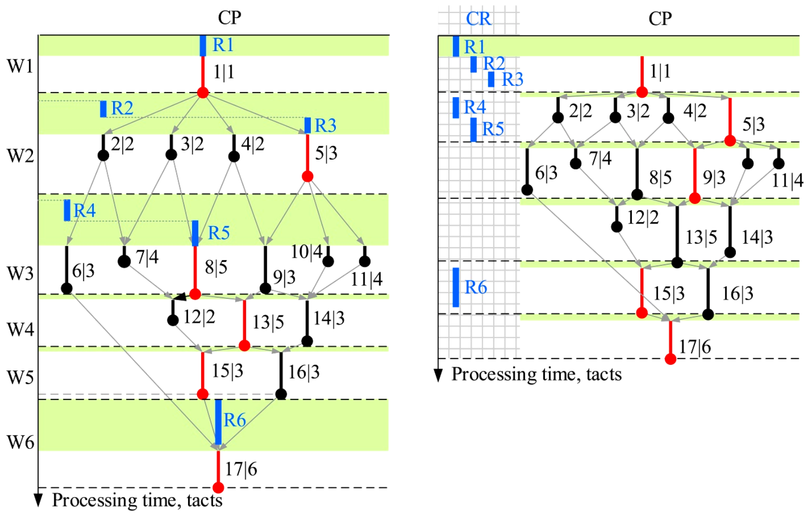

Fig. 1. An example of applying a complex approach to reducing the critical time of data processing: on the left – reuse of reconfigurable resources; on the right – preemptive reconfiguration; where i|j – the number of graph nodes Ni|type of operation Ij; ▮ – time of reconfiguration Rj; ● – the time of the task Tj; ● – the time of the task on the critical path Tj; 🟩 – overhead time; CP – control processor; CR – controller for reconfiguration

4. 2. 3. The new approach of transforming the computational algorithm MDG into the dynamically RCs

Hardware limitations and functional features of RCs do not allow the effective use of traditional methods of algorithms graph transformation that are oriented on the fixed computing environment without any spatial and functional limitations. In this case, the limitations that characterize the dynamically RCs make traditional approaches to modifying algorithms graphs with the aim of reducing the critical path ineffective. For example, the reduction of the critical path due to the expansion of the graph is impossible due to hardware limitations and the reduction of the graph width due to the increase of the critical path is impossible due to time constraints.

For the dynamically RCs, a new approach for modification of the MDG is proposed, which is based on using efficiency criteria (2)–(4). The reduction of critical time for performing a computational task is shown on the timelines (Fig. 1). The execution time of each task, which is the node of the MDG, consists of the time of task mapping and the time of HW task execution on the FPGA surface. The proposed approach consists in transferring the time component of tasks mapping from the graph critical path on other levels. To reduce the critical time, the complex integration of technologies of the preemptive reconfiguration of the FPGA computing space and the reuse of HW tasks resources, which prevents the retransmission of the configuration data are proposed. As a result, this part of the reconfiguration

The proposed approach is based on the parallelization the control process on the RCU structural level. For the realization of such parallelization, the use of the controller of reconfiguration (CR) for the configuration data loading and the local control processor (CP) for the computational process (Fig. 1) on the RCU level were proposed in the authors' previous work [24].

Estimation of the execution time of each tier of the MDG, an example of which is shown in Fig. 1, is given in Table 1.

Table 1

Comparative estimation of reconfiguration time

Tier number	Standard reconfiguration process	Removing conѳ figuration data re-loading	Preemptive reconfiguration
W_1	$T_{W_1}=R_1+T_1$	$T_{W_1}=R_1+T_1$	$T_{W_1}=R_1+T_1$
W_2	$T_{W_2}=3R_2+R_3+T_5$	$T_{W_2}=R_2+R_3+T_5$	$T_{W_2}=R_4+R_5$
W_3	$T_{W_3}=2R_3+3R_4+T_8$	$T_{W_3}=R_3+R_4+T_8$	$T_{W_3}=T_8$
W_4	$T_{W_4}=R_2+R_5+R_3+T_{13}$	$T_{W_4}=T_{12}$	$T_{W_4}=T_{12}$
W_5	$T_{W_5}=2R_3+T_{15}$	$T_{W_5}=T_{14}$	$T_{W_5}=R_6$
W_5	$T_{W_6}=R_6+T_{17}$	$T_{W_6}=R_6+T_{17}$	$T_{W_6}=T_{17}$

The expressions in Table 1 determine that the execution time of every tier of MDG is reduced in comparison with the standard reconfiguration sequence by preventing the retransmission of the configuration data. The preemptive reconfiguration on the base of parallelization of the control process intensively reduces the execution time of every tier.

4. 2. 4. Time estimation of the functional process of basic stages in RC

In order to estimate reconfiguration time, the main tasks mapping stages were determined and studied, on the basis of which it was determined that, according to the location of initial configuration data, three different reconfiguration sequences are possible. We define three basic sequences for the realization of which the multilevel configuration data caching on various levels of the dynamically RCs (Fig. 2) is possible. The means of realization of the multilevel configuration data caching of the RCU are described in the authors' previous work [24]. The mathematical expressions for determining the time of base sequences algorithms execution are given below.

Sequence I. *Standard reconfiguration process of the FPGA computing space.* Download of the configuration data of the HW task $Task_j$ from Global LCD:

$$T^{\mathrm{I}}_{SUM\,j} = R^{\mathrm{I}}_j + T_j = (T_{COMM\_NET\,j}) + T_j, \qquad (5)$$

where $T_{COMM\_NET}j$ – the time of configuration data search and loading from the Global LCD to the RCU level by network communications and the time of FPGA configuration.

Sequence II. *Caching configuration data in local memory.* Loading configuration data of the HW task from Local Configuration Date Memory (LMCD):

$$T^{\mathrm{II}}_{SUM\,j} = R^{\mathrm{I}}_j + T_j = T_{COMM\_LBUS\,j} + T_j, \qquad (6)$$

where T_{COMM_LBUSj} – the time of configuration data search and loading from the LCD and the time of FPGA configuration. At the same time, $T_{COMM\_NETj} << T_{COMM\_NETj}$ leads to $R^I_j >> R^{II}_j$.

Sequence III. *HW tasks caching on the FPGA surface,* when the HW task is already configured on the FPGA surface and the reloading of the configuration data doesn't happen:

$$T^{\mathrm{III}}_{SUM\,j} = T_j, \quad R_j = 0. \qquad (7)$$

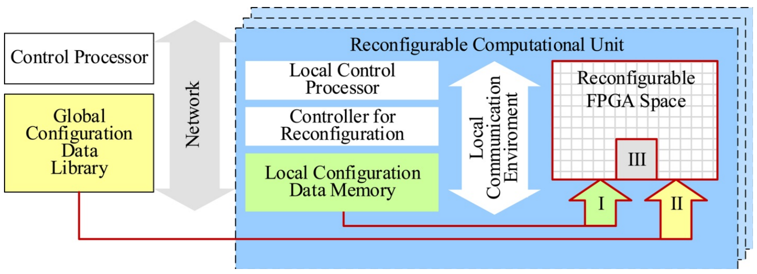
Control Processor
Global Configuration Data Library
Network
Reconfigurable Computational Unit
Local Control Processor
Controller for Reconfiguration
Local Configuration Data Memory
Local Communication Environment
Reconfigurable FPGA Space
III
I
II

Fig. 2. The basic loading sequences of configuration data into FPGA

Expressions (5)–(7) determine the time of data processing and allow to estimate the reduction of the critical execution time of algorithm's MDG and volume of overhead time, which is generalized in Table 2.

The execution time of the calculations is estimated by the method commonly used at the design stage of the computing systems, by the following expression:

$$T_B = \sum_{j=1}^{K} (P_j T_j) \Big| P = \overline{1,K},$$

where T_B – the execution time of the sequence of tasks $[I_j \mid j = \overline{1,K}]$ on the critical path B, K – the number of task types, P_j – the number of tasks instances, R_j – the loading and programming time of each j-th HW task on the FPGA space, T_j – the calculation time for each HW task.

Computation acceleration is measured by the absolute increase in ΔT_B speed and the reconfiguration acceleration ratio K_R. Absolute increment index allows you to estimate the amount of overhead time: $\Delta T_B = R_{remove}$.

Table 2

The mathematical models of basic stages of execution of algorithms in the RCs

Basic stages of configuration data loading	Time to complete tasks on the critical path, (T^B)	Unproductive time, (ΔR_{remove})
Standard reconfiguration process (5)	$T^B = \sum_{j=1}^{K} P_j R_j + \sum_{j=1}^{K} P_j T_j$	–
Loading from the Global CDL with caching HW tasks on the FPGA	$T^B_{rapid\_\mathrm{I}} = \sum_{j=1}^{K} R_j{}^{\mathrm{I}} + \sum_{j=1}^{K} P_j T_j$	$\Delta R_{remove\_\mathrm{I}} = \sum_{j=1}^{K} R^{\mathrm{I}}{}_j (P_j - 1)$
Loading from the LCDM with caching on the FPGA (6)	$T^B_{rapid\_\mathrm{II}} = \sum_{j=1}^{K} R_j{}^{\mathrm{II}} + \sum_{j=1}^{K} P_j T_j$	$\Delta R_{remove\_\mathrm{II}} = \sum_{j=1}^{K} P_j R_j{}^{\mathrm{I}} + \sum_{j=1}^{K} R^{\mathrm{II}}{}_j (P_j - 1)$
Storage in the FPGA cache memory (7)	$T^B_{rapid\_\mathrm{III}} = \sum_{j=1}^{K} P_j T_j$	$\Delta R_{remove\_\mathrm{III}} = \sum_{j=1}^{K} P_j R_j{}^{\mathrm{I}} + \sum_{j=1}^{K} P_j R_j{}^{\mathrm{II}}$

Table 2 provides mathematical models for the formal estimation of the execution time of the tasks sequence located on the critical path of the algorithm's MDG for different loading sequences (5)–(7). The sequences are considered to prevent the reloading of the configuration data. The $P_j T_j$ component corresponds to the total execution time of all instances I_j of the HW task on the reconfigurable FPGA area.

However, the reduction of the critical time of algorithm execution by transferring overhead time to other tiers of MDG can lead to the fact that other paths of the graph can become longer than the critical one. There arises the need for a sequential analysis of all paths to identify the most effective one. It is solved most effectively in a static mode. However, such solution is ineffective in a dynamic mode when the conditions of tasks mapping are unpredictable. Hence, another solution to the problem of adaptive task mapping is proposed. The proposed approach is based on the modification of the well-known branch and bound algorithm [23] applied for each graph tier of the MDG. The modified algorithm is based on the analysis of each node descendants within a certain tier of MDG and launch of preemptive reconfiguration procedure for each of them. The developed mathematical model of adaptive tasks mapping is based on the formalization of basic sequences of the algorithms execution (Table 2):

$$T_{G\_DAG}=\sum_{h=1}^{H_1}R_h+$$
$$+\sum_{k=1}^{w}\max\left(\sum_{h=1}^{H_{(k+1)}}R_h,\{T_h\,|\,h=\overline{1,H_k}\}\right)+\sum_{j=1}^{K}T_{IO\_j}. \qquad (8)$$

Algorithms are presented as MDGs and mapped into reconfigurable space of the dynamically RC. Expression (8) determines the total reconfiguration time of the HW tasks of the first tier nodes of the algorithm's MDG; critical execution time of the algorithm's MDG; execution time of sequential processes that can't be separated in time, for example, processes of synchronization and data exchange through the common communication environment.

The execution time of each tier of the algorithm's MDG is determined by the maximum value between the time of one-time configuration of this tier HW tasks set and execution time of the longest HW task.

5. The experimental time estimation and research of adaptive tasks mapping in the dynamically RCs

On the basis of the proposed formalization of the process of adaptive tasks mapping, the simulation model of the dynamically RC was developed. Convenient tools for modeling and researching the time characteristics of data processing in the dynamically RSC were developed. The developed means allow to allocate reconfigurable computing resources in near real time. In detail, the architecture of the RC simulation model was described by the authors in previous works [23, 25]. A simulation was performed and the process of adaptive tasks mapping in the dynamically RC according to the mathematical model (8) was researched.

The parameters of the functional blocks of the functional cores library [23, 25], in particular, the execution time and the firmware size of the HW tasks, were used as the source data for simulation modeling. A set of typical computational tasks with various complexity is realized, such as computing polynomials, cyclic functions, operations with matrices, solving SLAE (System of Linear Algebraic Equations). Data from other literary sources [19, 20] were also used. The configuration time of each HW task was determined according to the method described in [19]. The transmission time of the configuration data was estimated in proportion to the size of the HW task firmware with considering the delay of access to the external data memory. These delays were measured during the simulation of each functional block at the debugging stand. To determine the loading time of the configuration data from the centralized LCD, an accidental transmission delay factor for network communications channels was considered. Different conditions of tasks mapping were researched, depending on available time resources, free space on the FPGA surface, the set of pre-reconfigured HW tasks, the need to optimize the FPGA surface, the frequency of repeating similar functions in the dynamically entered tasks flow. In different mapping conditions, one or another strategy of tasks mapping is selected according to mathematical models from Table 1. The emulation tools of the dynamically RC determine the optimal place for caching of configuration data based on information about available resources of the FPGA computational field and the permissible amount of unproductive time.

Taking into account that the target class of tasks is algorithms with frequently repeated functions, the dependences of the computation time on the number of types of performed tasks (Fig. 3) are obtained during the experiments. For this class of tasks, it is found that mechanisms of reuse of reconfigurable resources allow accelerating the process of algorithms mapping. According to the results of experiments, it is clear that the complex integration of reuse of computing resources and preemptive reconfiguration allows to eliminate almost all unproductive time of the mapping process irrespective of the number of similar tasks. According to the obtained indicators, the proposed complex approach reduces the tasks mapping time 2.5 times compared with standard approaches.

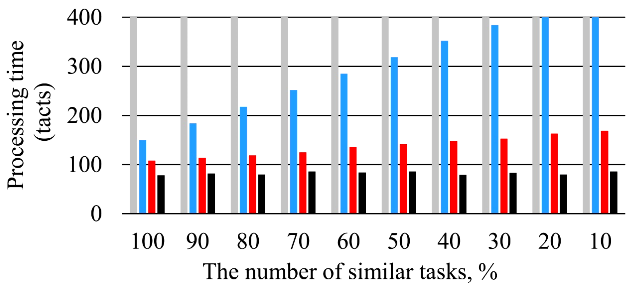

Fig. 3. The dependence of the mapping time on the number of similar tasks in the implementation of: ⬜ – standard reconfiguration process, 🟦 – removing configuration data re-loading; 🟥 – complex approach based on preemptive reconfiguration; ⬛ – execution time of HW tasks on FPGA

The rate of acceleration of the algorithms mapping process with a high density of arrival of new types of tasks and conversely was researched (Fig. 4). The rate of acceleration is calculated as the ratio of the execution time of the standard sequence of tasks mapping to the computation time on the basis of the proposed means (Table 2).

In general, the positive dynamics of an increase in the acceleration factor of computation is observed, but if the number of similar tasks exceeds 50 %, there is an increase of overhead time due to the need of multiple copying of active HW tasks. The way of optimization based on the storage of copies of the configuration data of all HW tasks in the local memory of the RCU was proposed and researched. It is determined that if the parameters of the executed tasks and the structure of the reconfigurable computing environment are commensurate, there is no sense in copying the HW tasks through the embedded memory of the FPGA chip. This helps to save the FPGA embedded memory. On the graph (Fig. 4), it is seen that copying through embedded memory allows on average of 1.16 times to accelerate the process of tasks mapping, if the width of the reconfigurable FPGA space is commensurate with the M-graph width.

The effectiveness of the proposed concept of adaptive tasks mapping on the reconfigurable structure of the dynamically RC was researched (Fig. 5, 6). The comparison of the simulation results with the results of the authors' previous researches is shown on the graphs in Fig. 5, 6. The developed simulation model has been used to realize the method of overhead time optimization in the dynamically RC. It was described in detail in [23]. The method for determining the optimal granularity of the calculations was described in [25].

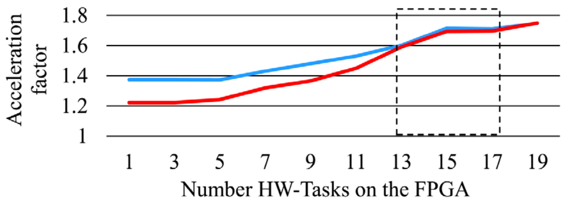

Fig. 4. The analysis of the acceleration factor on the size of the FPGA reconfigurable area by using adaptive tasks mapping: ▬ – with duplicating HW-tasks using FPGA ROM; ▬ – with loading HW-tasks duplicates from LCDM; ╍ – when the MDG graph width and the number of similar tasks are commensurate with the size of the FPGA reconfigurable area

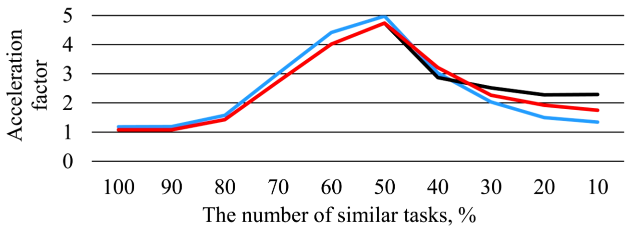

Fig. 5. The analysis of acceleration factor in the implementation of: ▬ – adaptive tasks mapping; ▬ – adaptive tasks mapping with overhead time optimization; ▬ – adaptive tasks mapping with calculations granularity optimization

The nature of the curves in Fig. 5 determines the common to all developed methods trend of dependence of the computation time on the ratio of parameters of computational algorithms and reconfigurable FPGA environment. A complex approach to the mapping time reduction allows to accelerate the algorithms execution by an average of 63 %, if the MDG graph width and the number of similar tasks are commensurate with the size of the reconfigurable FPGA area. Critical areas of the graphs correspond to the process of overcoming the space constraints of FPGAs on the basis of standard means (for example, defragmentation, unloading noncritical configurations). In critical areas (Fig. 5), it is seen that a sharp decrease of the acceleration intensity is on average up to 85 %.

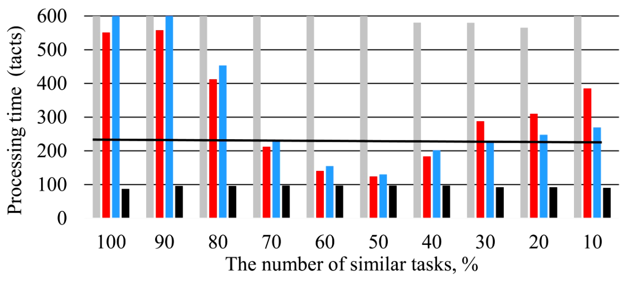

Fig. 6. The dependence of the processing time on the number of similar tasks in the research of: ⬜ – standard reconfiguration process (Sequence I); 🟥 – adaptive tasks mapping; 🟦 – adaptive tasks mapping with overhead time optimization; ⬛ – execution time of HW-tasks on FPGA; where ▬ – external limit of the processing time

When comparing the diagrams in Fig. 3, 6, it is seen that the influence of spatial constraints of the FPGA reconfigurable space reduces the positive effect from using any means of tasks mapping acceleration. The tasks mapping is accompanied by additional overhead time taking place in overcoming spatial constraints of the reconfigurable FPGA space. On the graph in Fig. 6, it is seen that the minimum computation time is achieved if the MDG graph width is commensurate with the size of the reconfigurable FPGA area. Optimization of the overhead time of tasks mapping [23] reduces the influence of spatial constraints of the reconfigurable FPGA area on the computation speed.

It should also be noted that according to the results of experiments, optimization of overhead time by the criteria of time and hardware limitations of RCs reduces the influence of spatial constraints on the computation speed by approximately 10 %, as compared with the standard process of reconfiguration (Fig. 5) [23]. Optimization of the calculation granularity by the criteria of the ratio of parameters of computational algorithms and reconfigurable FPGA space provides computation acceleration on the critical parts of data processing by on average 12 % compared with the implementation of large-grained computation [25].

It should also be noted that the nature of the diagrams in Fig. 3, 6, obtained as a result of the research of the RC simulation model, coincides with the results of simulation of the acceleration means of algorithms mapping on the hardware simulation models, described in detail in [24].

6. Discussion of the research and experiments results

The research has shown that complex approaches to choosing the task mapping strategy, which integrate static and dynamic approach are the most effective for solving the problem of adaptive task mapping in the dynamically RCs. The implementation of multi-level multifunctional cache-memory has allowed choosing the optimal task mapping strategy based on determined criteria of constraints of the dynamically RCs. An important unresolved aspect in the presented research is the problem of task mapping optimization according to the criteria of minimizing energy consumption. Solving this problem concerns future papers on expanding the proposed adaptive task mapping concept.

The research is carried out as a part of a series of studies focused on developing the elements of the theory of increasing the data processing efficiency in the dynamically RCs. The proposed adaptive task mapping concept allows accounting for changing mapping conditions and the nature of computational algorithms, notably the number of tasks at a particular graph's tier and repetition frequency of tasks of the same type. (Fig 5, 6). The research in the authors' previous papers [23, 25] is focused on optimization of overhead time at different levels of the dynamically RCs. Based on the series of studies, it has been determined that the target functions of reducing overheads and computing time require solving the multicriteria problem of optimizing the data processing in RCs by the integral criterion of the ratio of data processing overheads, computational algorithm parameters and computational environment structure, taking into account its hardware constraints. This research area also concerns future papers of the authors.

The proposed tools can be applied in problem-oriented systems characterized by the heterogeneous computational environment, which contains both discrete processors and computational structures based on the dynamically programmed FPGA. Problem orientation of the mentioned systems is attained by the development of problem-oriented functional cores libraries based on FPGA. General features of target tasks' algorithms are regular parallelism, a great number of information exchanges, frequent repeats of similar functions, a large volume of computations, the capability of algorithms' representation in the form of subtasks with the same complexity and input data level.

Systems with numerical program control, operating in real-time mode, serve as an example of practical application of RC and proposed methods and tools for increasing their efficiency. In such systems, there is a need to solve tasks of interpolation of different functions depending on a large number of coordinates in every step of control. Time limits constrained by the real-time mode impose strict constraints on the control cycle duration which cannot be exceeded. The consequence of limitations is ineffective usage of discrete processors and microcontrollers. Distributed systems of flight data processing and analysis, which solve the task of remote processing of a large amount of information, are another example of application. Acceleration of data processing, optimization of equipment usage, reduction of energy consumption, size and cost of computer systems are considered to be the main effect of the practical usage of the proposed tools.

7. Conclusions

1. The concept of adaptive tasks mapping into the dynamically RCs, which is based on a new approach to the transformation of the MDG and multilevel caching of the configuration data is developed. It allows the implementation of different strategies of tasks mapping due to the criterion of time overheads minimization. It takes into account the FPGA hardware limitations and the flexible computational environment parameters during the dynamical tasks mapping as well.

2. The mathematical models of the base stages of tasks mapping are developed. These models take into account the multilevel structure of the dynamically RCs and overhead time at all their functioning levels. The mathematical model for the adaptive tasks mapping into the reconfigurable computational structure of the dynamically RCs is developed.

3. The simulation model of the dynamically reconfigurable computer based on the proposed mathematical models is developed. It allows for investigating and evaluating the data processing based on the adaptive algorithm mapping due to the criteria of unproductive time minimization and the FPGA hardware limitations. It was proved that the mapping adaptation to the parameters of the reconfigurable environment requires the optimization of the existing methods at different levels of the dynamically reconfigurable computers. The following was found in an experimental way. It is shown that for the repetitive algorithms the unproductive time is reduced by 63 % on average, compared to the standard methods, using the proposed approach. The optimization of the unproductive time reduces the space limitations influence on the processing speed by approximately 10 %, taking into account the limitations of the reconfigurable FPGA space. The optimization of the computational granularity taking into account the FPGA hardware limitations provides the computation speed increase by 12 % on average, depending on the technical characteristics of the FPGA chips, compared to the coarse grain computations.

References

1. Kumar S. Fundamental limits to Moore's law // arXiv. 2015. URL: https://www.researchgate.net/profile/Suhas_Kumar5/publication/284219009_Fundamental_Limits_to_Moore's_Law/links/5663fd9408ae192bbf901e85.pdf
2. Facilitating Preemptive Hardware System Design Using Partial Reconfiguration Techniques / Dondo Gazzano J., Rincon F., Vaderrama C., Villanueva F., Caba J., Lopez J. C. // The Scientific World Journal. 2014. Vol. 2014. P. 1–15. doi: 10.1155/2014/164059
3. Koch D. Partial reconfiguration on FPGAs. Architectures, tools and applications. Springer-Verlag, 2013. 296 p. doi: 10.1007/978-1-4614-1225-0
4. Melnyk V. Self-configurable FPGA-based computer systems: basics and proof of concept // Advances in cyber-physical systems. 2016. Vol. 1, Issue 1. P. 39–50.
5. Prototyping an Automated Video Surveillance System Using FPGAs / Singh S., Saurav S., Shekhar C., Vohra A. // International Journal of Image, Graphics and Signal Processing. 2016. Vol. 8, Issue 8. P. 37–46. doi: 10.5815/ijigsp.2016.08.06
6. DynamIA: Dynamic Hardware Reconfiguration in Industrial Applications / Mentens N., Vandorpe J., Vliegen J., Braeken A., da Silva B., Touhafi A. et. al. // Lecture Notes in Computer Science. 2015. P. 513–518. doi: 10.1007/978-3-319-16214-0_47
7. Minaev Yu., Filimonova O. Fuzzy Mathematics on the Basic of Uncertainty Tensor Models. Chapter I. Tensor-variable in the Fuzzy Set System // Elektronnoe modelirovanie. 2008. Vol. 30, Issue 1. P. 43–59.
8. On the Evaluation of Different High-Performance Computing Platforms for Hyperspectral Imaging: An OpenCL-Based Approach / Guerra R., Martel E., Khan J., Lopez S., Athanas P., Sarmiento R. // IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2017. Vol. 10, Issue 11. P. 4879–7897. doi: 10.1109/jstars.2017.2737958
9. Rajasekhar Y., Sass R. Architecture and Applications for an All-FPGA Parallel Computer // 2012 41st International Conference on Parallel Processing Workshops. 2012. doi: 10.1109/icppw.2012.22
10. George A., Lam H., Stitt G. Novo-G: At the Forefront of Scalable Reconfigurable Supercomputing // Computing in Science & Engineering. 2011. Vol. 13, Issue 1. P. 82–86. doi: 10.1109/mcse.2011.11
11. Runtime Scheduling, Allocation, and Execution of Real-Time Hardware Tasks onto Xilinx FPGAs Subject to Fault Occurrence / Iturbe X., Benkrid K., Hong C., Ebrahim A., Arslan T., Martinez I. // International Journal of Reconfigurable Computing. 2013. Vol. 2013. P. 1–32. doi: 10.1155/2013/905057

12. Al-Wattar A., Areibi S., Saffih F. Efficient On-line Hardware/Software Task Scheduling for Dynamic Run-time Reconfigurable Systems // 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. 2012. doi: 10.1109/ipdpsw.2012.50
13. Achieving energy efficiency through runtime partial reconfiguration on reconfigurable systems / Liu S., Pittman R. N., Forin A., Gaudiot J.-L. // ACM Transactions on Embedded Computing Systems. 2013. Vol. 12, Issue 3. P. 1–21. doi: 10.1145/2442116.2442122
14. Klymenko I., Rudnytsky M. Classification of reconfigurable computing systems // Visnyk of Vinnytsia Politechnical Institute. 2014. Issue 5 (116). P. 120–128.
15. Jing C., Zhu Y., Li M. Energy-efficient scheduling on multi-FPGA reconfigurable systems // Microprocessors and Microsystems. 2013. Vol. 37, Issue 6-7. P. 590–600. doi: 10.1016/j.micpro.2013.05.001
16. Sergiyenko A. Klymenko I., Sergiyenko P. Reconfigurable many-core computer based on FPGA // Visnyk NTUU "KPI". Informatyka, upravlinnia ta obtchislyuvalna technika. 2016. Issue 64. P. 47–50.
17. Klymenko I. A. The effectiveness analysis of resources management in reconfigurable computer systems // Visnyk NTUU "KPI". Informatyka, upravlinnia ta obtchislyuvalna technika. 2015. Issue 62. P. 11–21.
18. Smith M. C., Peterson G. D. Optimization of Shared High-Performance Reconfigurable Computing Resources // ACM Transactions on Embedded Computing Systems. 2012. Vol. 11, Issue 2. P. 1–22. doi: 10.1145/2220336.2220348
19. Dunets R., Tykhanskyy D. Problems of partially reconfigurable FPGA-based system design // Radioelectronic and computer systems. 2010. Issue 7 (48). P. 200–204.
20. Adaptive resource management for simultaneous multitasking in mixed-grained reconfigurable multicore processors / Ahmed W., Shafique M., Bauer L., Henkel J. // CODES+ISSS '11 Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. Taipei, Taiwan, 2011. P. 365–374.
21. Happe M., Traber A., Keller A. Preemptive Hardware Multitasking in ReconOS // Lecture Notes in Computer Science. 2015. P. 79–80. doi: 10.1007/978-3-319-16214-0_7
22. Dümmler J., Rauber T., Rünger G. Scalable computing with parallel tasks // Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers – MTAGS '09. 2009. doi: 10.1145/1646468.1646477
23. Kulakov Y. O., Klymenko I. A., Rudnytskyi M. V. The method for providing quality of service time requirements in reconfigurable computing systems // Eastern-European Journal of Enterprise Technologies. 2015. Vol. 4, Issue 4 (76). P. 25–30. doi: 10.15587/1729-4061.2015.47227
24. Kulakov Y. O., Klymenko I. A. The multilevel memory in the reconfigurable computing system // Visnyk NTUU «KPI». Informatyka, upravlinnia ta obchislyuvalna technika. 2014. Issue 61. P. 18–26.
25. The method for providing quality of service time requirements in reconfigurable computing systems / Klymenko I., Kulakov Y., Tkachenko V., Storozhuk O. // Eastern-European Journal of Enterprise Technologies. 2016. Vol. 5, Issue 9 (83). P. 4–12. doi: 10.15587/1729-4061.2016.81003