The method of transforming algorithms' graphs for tasks mapping in the dynamically reconfigurable computer systems

I. Klymenko, O. Storozhuk, Y. Kulakov

https://doi.org/10.31174/NT2018-158VI18-14

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
Corresponding author. E-mail:ikliryna@gmail.com, storozhuk.om@gmail.com, ya.kulakov@gmail.com

Paper received 25.01.18; Accepted for publication 29.01.18.

Abstract. This article is dedicated to resolves the problem of reducing time overheads of task mapping process on dynamically reconfigurable computing structure in the reconfigurable computer systems based on FPGA. The proposed mathematical models for the basic stages of the data processing take into account the influence of the configuration data loading delay on the efficient of the reconfigurable computation. Based on these models we propose the method of transforming algorithms' graphs for tasks adaptive mapping in the dynamic reconfigurable computer systems. Simulation of the proposed means showed that the proposed approach allows reducing time overheads and improving the efficiency of reconfigurable computing for resolving tasks with frequent repetitions of the same type of functions.

Keywords: *Reconfigurable Computer Systems, Reconfiguration overhead, Partial dynamic reconfiguration, FPGA, Field Programmable Gate Array.*

Introduction. The high level of modern progress has led to the fact that extensive technologies for increasing the productivity of high-performance computing reach limitations, which is confirmed by violations of Moore's law in recent years [1]. The desire to further increase performance to the level of *exaflops* makes the challenge to find new intensive solutions. The technology of FPGA dynamic reprogramming is one of the perspective directions in this area. So the technology is rapidly developing today in the direction of creating reconfigurable high-performance computer systems or *Reconfigurable Computers* (RCs) [2 – 4]. The most perspective classes of tasks for solved by means of the dynamically RC are real-time control tasks, in particular computing in an indefinable basis or fuzzy computing, that having informational, multidimensional and dynamic nature [3, 4].

In contrast to the static reconfiguration that is limited to the implementation of the traditional principle of functional cores hardware acceleration [5, 6], *Partial Dynamic Reconfiguration* (PDR) creates preconditions for creating computing structures adapted to the requirements of *Run Time* mode tasks solving [2, 4]. Firstly, this allows to overcome the hard architecture limits of traditional high-performance computing systems of wide use and to bring their actual productivity to the declared peak in critical classes of tasks. Secondly, to significantly expand the functional capabilities of RCs based on FPGA and to implement computing structures high complexity on a limited space of the chips without a significant increase in prices. There is also an opportunity to increase energy saving efficiency based on e use of PDR technology [4].

Brief overview of related publications. The efficiency of data processing in RCs is most influenced by the time overhead in the reconfiguration process of the computing environment. There are many ways to reduce overhead, such as reusing reconfigurable computing resources [7], cached of configuration data [8], preemptive reconfiguration [9, 10]. Most known display mechanisms are realized at software level of the operating system superstructure, the overcoming of space restrictions of the FPGA is carried out by standard means, for example by defragmentation of the computing space of the FPGA [7, 8], by unloading of non-critical configurations [7, 10], by failure to perform tasks [7, 8, 10]. These methods are mainly aimed at reducing the overhead on the physical level of the static RCs functioning and their use in a dynamically RCs is ineffective.

The purpose of this article is solving the important problem of the present which impedes the intensive progress of reconfigurable computing that consists in the development of new method for adaptive mapping tasks in to change the computing structure of reconfigurable computing systems, taking into account their functional and hardware constraints.

Materials and methods. Problems with mixed type of parallelism are considered as initial problems. This allows them to transform their graphs of algorithms to configure the most efficient task-oriented computing structures with a high processing speed in the reconfigurable environment. *Macro Dataflow Graphs* (MDGs) present computational tasks with a mixed type of parallelism, in the nodes of which macro tasks (M-tasks) are placed [11].

The computational task is presented by a MDG $G_M = \{N_G, D_G\}$, where N_G – the set of nodes corresponding to the M-tasks; D_G – the set of edges that determine the relationship between M-tasks, $N_M = \{N_1, N_2, \ldots, N_i, \ldots, N_g\} \mid i = \overline{1,g}$ – the set of M-tasks in the graph nodes; g – the number of graph nodes, $W_k \mid k = \overline{1,w}$ – the identifier of the MDG tier; *w* – number of MDG tier. Each M-task is put into hardware task compliance (HW-task), which is determined by vector [10] $Task_j = \{S_j, T_{SUM\,j} | (T_j + R_j), I_j, N_i\}$, where S_j – the area of rectangle that containing the task $Task_j$ on the FPGA reconfigurable computing space; R_j – the time sending the configuration data and reconfiguration of the computing structure on the FPGA to perform the HW-task; $T_j = T_{HW\,j}$ – the HW-task implementation time on the FPGA equipment, taking into account of time for input-output data to computation; $T_{SUM\,j}$ – total task N_i execution time, $I_j \mid j = \overline{1,m}$ – the computation function implemented by HW-task; *m* – the number of HW-tasks synthesized and stored in the of *Configuration Data Library* (CDL).

In order to determine the dynamically RCs performance criteria, improved acceleration indicator has been proposed, which, in contrast to the well-known [12], takes into account the time complexity of parallel control scheduling processes and allocation of reconfigurable computational resources:

$$\rho = \frac{T_{SW}}{[T_{CONTROL} + R] + T_{HW}}, \; (1)$$

where T_{SW} – is time of M-task computation on the processor core; T_{HW} – time of HW-task computation on FPGA equipment; R – computing environment reconfiguration time for the M-task, $T_{CONTROL}$ – process complexity time of mapping M-task on to reconfigurable computing environment. The amount of $(T_{CONTROL} + R)$ determines the overhead of mapping process computing M-tasks to reconfigured computing environment.

Based on modified efficiency criterion (1), the target function of reducing the overhead of mapping tasks process to reconfigurable computing environment is obtained, which depends entirely on the delays that accompany this process:

$$\min(T_{CONTROL} + \sum_j R_j) = \min(T_{CONTROL}) + \sum_j \min(T_{COMM\ j}) + \sum_j T_{CONFIG\ j}, \; (2)$$

where T_{COMM} – transmission time of configuration data from external storage in FPGA interfaces; T_{CONFIG} – the configuring time of computing environment, while doing so $R = T_{COMM} + T_{CONFIG}$.

The basic efficiency criteria of realization computing tasks algorithms in dynamically RC are determined, according to which the length of the critical path and width of the MDG presented tier-parallel algorithm form are estimated by the following relations:

$$(R + T_{HW}) < T_{SW}; \; \max[H_k] \mid k = \overline{1, w} \le n, \; (3)$$

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

The functional features of RCs 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 that spent on to the process of transferring configuration data from remote libraries to the FPGA chip interfaces, which preceding the programming process of the computing area on the FPGA chip surface. This time defines an unproductive part of the reconfiguration time and it is a critical criterion for the effectiveness of dynamically reconfigurable computations.

A new approach of transforming the MDG into a dynamic RC. For RCs in the article a new approach for modification of the MDG presented tier-parallel algorithm is proposed, which is based on criteria (2) and (3). The proposed approach is to transfer the time component of the M-tasks mapping from the graph critical path to other tiers. To reduce the critical time, proposed complex integration of technologies of preemptive computation structure reconfiguration and the reusing of HW-tasks resources, which prevents the reload of the configuration data. This part of the reconfiguration time component will be removed from the critical path, and the part is moved to the previous tier of the MDG graph. As result of such graph modification, the critical path will only be determined by the sequence M-tasks computation time.

In order to estimate reconfiguration time, the main mapping tasks stages were determined and studied, on which basis it was determined that, according to the location of configuration data, there are possible three reconfiguration sequences. To implement the proposed method, it is suggested to use multilevel caching of configuration data at different levels of RC (Fig. 1), that allows realizing three basic loading sequences of configuration data:

Sequence I. *Standard reconfiguration process of the FPGA computation space.* Download hardware data for configuration task $Task_j$ from global CDL. $T^{I}_{SUM\,j} = R^{I}_j + T_j = (T_{COMM\_NET\,j}) + T_j$, where $T_{COMM\_NET\,j}$ – it's the search time and configuration data transfer from the global CDL to the level of RCU by network communications and the programming time of FPGA.

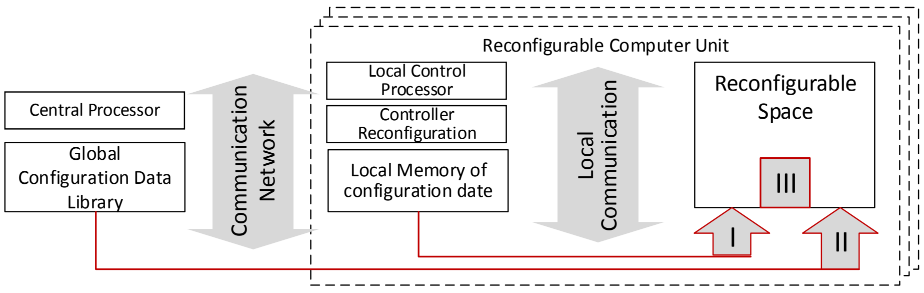
Reconfigurable Computer Unit
Central Processor
Global Configuration Data Library
Communication Network
Local Control Processor
Controller Reconfiguration
Local Memory of configuration date
Local Communication
Reconfigurable Space
III
I
II

Fig. 2. The basic loading sequence of configuration data

Sequence II. *Caching configuration data in the Local Memory* (LM) of the *Reconfigurable Computer Unit* (RCU). Loading hardware configuration data task from LM of the RCU: $T^{II}_{SUM\ j} = R^{II}_j + T_j = T_{COMM\_LBUS\ j} + T_j$ where $T_{COMM\_LBUS\ j}$ – the search and transfer time of configuration data from the LM and the time programming of FPGA.

Sequence III. *Caching SW-tasks in to the FPGA* reconfigurable computation space. The HW-task in reconfigurable area of the FPGA surface is already configured: $T^{III}_{SUM\ j} = T_j, R_j = 0$.

Mathematical models of data processing, which allow to estimate reduction of the critical time execution of MDG, and overhead time amount, are summarized in Table. 1, where T_B is the execution time of sequence of interconnected tasks $I_j \mid j = \overline{1, K}$ from which the critical path of the MDG is formed.

Table 1. Basic stages mathematical models organization of reconfigurable computation

	Time to complete tasks on a critical path, (T^B)	Overhead time, (ΔR_{remove})
Standard reconfiguration process (Sequence I)	$T^B = \sum_{j=1}^{K} P_j R_j + \sum_{j=1}^{K} P_j T_j$	–
Loading from the central library of caching configurations on the FPGA	$T^B_{rapid\_\mathrm{I}} = \sum_{j=1}^{K} R_j{}^{\mathrm{I}} + \sum_{j=1}^{K} P_j T_j$	$\Delta R_{remove\_\mathrm{I}} = \sum_{j=1}^{K} R^{\mathrm{I}}{}_j (P_j - 1)$
Load from the local memory of the cache module on the FPGA (Sequence II)	$T^B_{rapid\_\mathrm{II}} = \sum_{j=1}^{K} R_j{}^{\mathrm{II}} + \sum_{j=1}^{K} P_j T_j$	$\Delta R_{remove\_\mathrm{II}} = \sum_{j=1}^{K} P_j R_j{}^{\mathrm{I}} + \sum_{j=1}^{K} R^{\mathrm{II}}{}_j (P_j - 1)$
Storage in a cache memory on a FPGA (Sequence III)	$T^B_{rapid\_\mathrm{III}} = \sum_{j=1}^{K} P_j T_j$	$\Delta R_{remove\_\mathrm{III}} = \sum_{j=1}^{K} P_j R_j{}^{\mathrm{I}} + \sum_{j=1}^{K} P_j R_j{}^{\mathrm{II}}$

The following notation is used for the formal time estimation: the execution time of sequence of M-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 – calculation time for each HW-task. The component P_jT_j corresponds to the total productive execution time of all instances of the HW-task I_j on the reconfigurable area of the FPGA surface. Based on developed basic stage formalization of reconfigurable computing organization, a mathematical model of process of adaptive tasks mapping, presented by MDG graphs in to reconfigurable computation structure is proposed:

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

$$, R_j = 0, \quad (4)$$

Expression components (4) include the reconfiguration time of the first tier nodes, configuring time of reconfigurable computation structure for each task; critical time of computational algorithm performing, execution time of sequential processes that can't be separated in time, for example, processes synchronization and data exchange through the common communication environment.

Results and discussion. Based on the formal estimation of the time presented above, we offer a simulation RC model that allows allocating reconfigurable resources in a time approximating to real, which is a convenient tool for modeling and studying the time characteristics of functional data processing process in a dynamic RC.

The research was conducted for a series of algorithms presented by the MDG graphs. Graphs of algorithms with different numbers of similar tasks and different power of connectivity were investigated. The investigated algorithms are randomly synthesized on the basis of the developed library of hardware implementations of functional cores that correspond to certain macro problems. Functional blocks of hardware tasks are synthesized in Verilog hardware language and implemented on the Altera Cyclone II EP2C35F672C6 FPGA. Altera Development Kit DE2 board is used for verification and research of functional blocks time characteristics. On the experiments basis, the reconfiguration time which is depends on the number of performed tasks types (Fig. 2) were obtained.

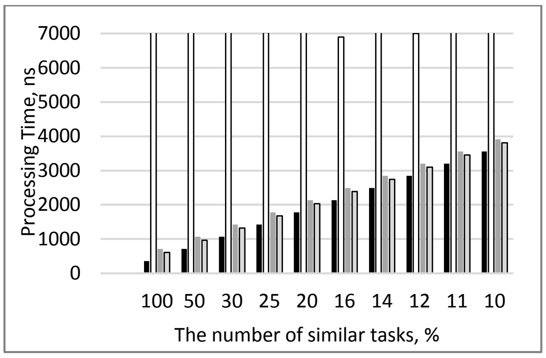

Fig. 2. The dependence of the critical processing time on the number of similar tasks: ■ – reused on the FPGA reconfigurable space (Sequence III); □ – loading from the global library without reuse (Standard); ▨ – loading from the global library with reuse (Sequence I); ▩ – loading from local memory with reuse (Sequence II).

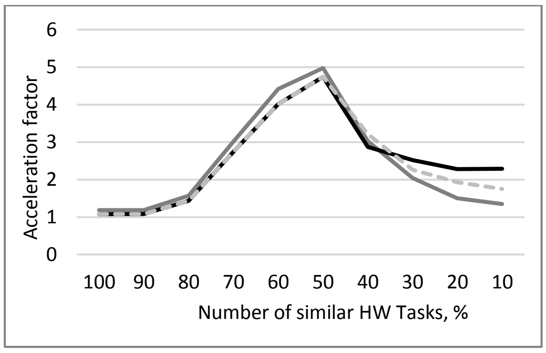

Fig. 3. Optimization of data processing: ━ – adaptive HW Tasks mapping; ╌ – adaptive HW Task mapping with optimization overhead time reconfiguration; ━ – adaptive HW Task mapping with optimization ratio of MDG width to FPGA reconfigurable space.

The efficiency of the reusing resources algorithms depends on the number of similar tasks in the computational algorithm (Fig. 2). According to the results of experiments, it can be seen that complex integration of reusing reconfigurable computing resources and preemptive reconfiguration (4) allows to eliminate practically all unproductive reconfiguration time for any computational algorithms, regardless of similar tasks amount. During the investigation of acceleration values for reconfigurable computations, it was found that complex approach reduces the reconfiguration time by a factor of 2.5 compared to the standard approach. Curves character in Fig. 3 are determines the dependence tendency of the computation time on the ratio of the solvable tasks parameters and the reconfigurable environment. For the number of tasks of the same type, more than 50% of increase in overhead time is due to the introduction of additional means for active tasks copying. Optimization is done by using a copy of the configurations of all tasks in the computing module local memory. However, by the proportional relationship between the parameters of solvable problems and the structure of the reconfigurable computing environment, there is no point in copying HW tasks through the internal memory of the FPGA chip. It helps to preserve the internal memory of the FPGA and allows an average of 1.16 times to accelerate the computational process.

Conclusions. Considering the investigation results of the proposed method of HW-tasks adaptive mapping in to computation structure of RCs, it was found that provides an intensive acceleration to an average of 63%, provided that the MDG graph width and the large number of similar tasks are commensurate with the size of the reconfigurable section FPGA. In this case, in the process of overcoming the FPGA space constraints, which corresponds to the critical sections on the charts dependency, there is a sharp decrease in the acceleration intensity of reconfiguration by an average of 85%.

It is determined that the ratio of the dimension of the reconfigurable computing environment and the degree of parallelization algorithm of the solvable problem, as well as the frequency of execution of similar functions, is significantly influenced by the reconfiguration speed. Violation of these ratios is accompanied by additional unproductive costs in the process of tasks adaptive mapping in to the reconfigured computation structure (4), that found the necessity to optimize the data processing in the RCs. Such optimization by the integral optimization criterion is described in detail in the previous papers by the authors [10, 13]. According the results of experiments is determined that the efficiency of the proposed method of tasks adaptive mapping increases due to optimization of overhead time considering the restrictions of the FPGA reconfigurable space. Such an optimization reduces the intensity of the influence the limitations of the FPGA reconfigurable space on the total computation speed by about 10%. However, this quantity may fluctuate within the region of optimal reconfiguration parameters [13]. Optimization of the computation granularity [13] provides computing acceleration at the critical parts of the data processing process, on average up to 12%, depending on the technical parameters of the FPGA chip, compared with the implementation of large-grained computation.

REFERENCES

1. Kumar, S. Fundamental limits to Moore's law. https://www.researchgate.net/profile/Suhas_Kumar5 /publication/284219009_Fundamental_Limits_to_Moore's_Law/ links/5663fd9408ae192bbf901e85.pdf, last accessed 2017/11/28.
2. Koch D. Partial reconfiguration on FPGAs. Architectures, tools and applications// Springer-Verlag, 2013)
3. Melnyk V. Self-configurable fpga-based computer systems: basics and proof of concept. Advances in cyber-physical systems, 2016. 1(1). P. 39-50.
4. Klymenko, I., Rudnytsky, M. Classification of reconfigurable computing systems// Visnyk of Vinnytsia Politechnical Institute, 2014. (116). P. 120-128.
5. Rajasekhar, Y., Sass, R.: Architecture and applications for an all-FPGA parallel computer// In: 41st International Conference on Parallel Processing Workshops (ICPPW) on Proceeding, US, PA, Pittsburgh, 2012. P. 157-164.
6. George, A., Lam, H., Stitt, G. Novo-G: At the forefront of scalable reconfigurable supercomputing// Computing in Science & Engineering, 2011. 13 (1). 82-86.
7. Al-Wattar, A., Areibi, S., Saffih, F. Efficient On-line Hardware/Software Task Scheduling for Dynamic Run-time Reconfigurable Systems// In: 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW) on Proceedings, China, Shanghai, 2012. P. 401-406.
8. Liu, S., Pittman, R.N., Forin, A., Gaudiot, J.-L. Achieving Energy Efficiency through Runtime Partial Reconfiguration on Reconfigurable Systems// Transactions on Embedded Computing Systems (TECS), 2013. P. 72:1-72:21.
9. Zhao, G., Hou, Y, Wang S. Research on reconfiguration technology based on SOPC for PXI instrument. In: AUTOTESTCON on Proceedings, USA, Virginia, 2015. P. 280-283.
10. Kulakov, Y.O., Klymenko, I.A., Rudnytskyi, M.V. Development of the reconfiguration acceleration method in the dynamically reconfigurable computing systems// Eastern-European Journal of Enterprise Technologies, 2015. 4/4 (76). P. 25-29.
11. Dümmler, J., Rauber, T., Rünger G. Scalable computing with parallel tasks// In: the 2nd Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS '09) on Proceedings, US, Oregon, Portland, 2009. P. 1-10.
12. Ahmed, W., Shafique, M., Bauer, L., Henkel, J. Adaptive Resource Management for Simultaneous Multitasking in Mixed-Grained Reconfigurable Multi-core Processors// In: 9th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) on Proceedings, Taipei, 2011. P.365-374.
13. Kulakov, Y., Klymenko, I., Tkachenko, V., Storozhuk, O. The method for providing quality of service time requirements in reconfigurable computing systems// Eastern-European Journal of Enterprise Technologies,2016. 5/9 (83). P. 1-12.

Метод трансформации графов алгоритмов для отображения задач в динамически реконфигурируемых компьютерных системах

И. А. Клименко, А. Н. Сторожук, Ю. А. Кулаков

Аннотация. Статья посвящена решению проблемы сокращения накладных расходов времени процесса отображения задач на динамически реконфигурируемую вычислительную структуру в реконфигурируемых компьютерных системах на базе ПЛІС. Предлагаемые математические модели основных этапов обработки данных учитывают влияние задержек передачи конфигурационных данных на эффективность реконфигурируемых вычислений. На основе этих моделей мы предлагаем метод трансформации графов алгоритмов для адаптивного отображения задач в динамических реконфигурируемых компьютерных системах. Моделирование разработанных средств показало, что использование предлагаемого подхода позволяет сократить накладные расходы времени и повысить эффективность реконфигурируемых вычислений для решения задач с частыми повторениями одного и того же типа функций.

Ключевые слова: *Реконфигурируемые компьютерные системы, накладные расходы реконфигурации, частичная динамическая реконфигурация, ПЛИС, программируемые логические интегральные схемы.*