DOI: 10.14529/cmse170105

HYBRID COMPUTER SYSTEM PROGRAMMING TECHNOLOGY WITH ADAPTATION AND SCALING OF CALCULATIONS*

© 2017 г. A.A. Gulenok[1], A.I. Dordopulo[2], I.I. Levin[2], V.A. Gudkov[2]

[1] *Academician A.V. Kalyaev SRI multiprocessor computer system at Southern Federal University (GSP-284, Chekhov st. 2, Taganrog, 347928, Russia),*

[2] *Scientific Research Centre of Supercomputers and Neurocomputers (Italyansky lane 106, Taganrog, 347900, Russia),*

E-mail: andrei_gulenok@mail.ru, scorpio@mvs.sfedu.ru, levin@superevm.ru, Slava_Gudkov@mail.ru

Received: 20.10.2016

The paper considers the programming technology for hybrid computer systems, which contain reconfigurable and microprocessor computational nodes. The base of the programming technology for hybrid computer systems is the high-level programming language COLAMO with extensions, which allow descriptions of various types of parallel calculations such as structural, structural-procedural, multi-procedural and procedural forms of organization of calculations in a unified parallel-pipeline form. The suggested parallel-pipeline form allows modifications of forms of organization of calculations. Such modifications are performed automatically by the COLAMO language preprocessor, which takes into account current configuration of the hybrid computer system. Owing to the suggested technology, the program can be automatically adapted to the changed architecture or configuration of the hybrid computer system without any modifications of the source code made by the developer. Specially for this the source parallel program, developed in the programming language COLAMO, is transformed by the pre-processor into the canonical form. Then the pre-processor estimates the available computational resource, detects effective parameters of implementation of the program on the available resource and, if necessary, reduces the program performance to adapt it to the current configuration of the hybrid computer system. The technology provides two-way scaling: for increasing of the available computational resource (induction), and for reducing of the available computational resource (reduction), which provides resource independence of programming during implementation of the program, i.e. the developer is not "bound" to the available hardware resource of the computer system.

Keywords: performance reduction, high-level programming language, programming of hybrid computer systems, application adaptation, application scaling.

FOR CITATION

Gulenok A.A., Dordopulo A.I., Levin I.I., Gudkov V.A. Hybrid computer system programming technology with adaptation and scaling of calculations. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no. 1. pp. 73–86. DOI: 10.14529/cmse170105.

Introduction

The majority of real-world problems require combination of both sequential and parallel computational fragments within a single computational space for effective implementation of structural and procedural [1] fragments of calculations. Many developers consider design of computer systems with hybrid organization of calculations as a solution of this problem. Such computer systems can contain computational nodes with different architectures, united by data

* The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel computational technologies (PCT) 2017".

transfer channels, and allow implementation of structural and procedural calculations within a single computational space. Symbiosis of nodes with different architecture in one computer system theoretically allows the growth of the computer system real performance owing to the opportunity of effective implementation of both structural and procedural fragments of calculations in the nodes with different architecture.

Wide application of such computer systems for solving real-world problems is considerably limited by high complexity of their programming, as the effective use of architectural advantages of all computational nodes requires not only deep knowledge of various programming languages and development environments for designing computational nodes of various types, but also skills of independent synchronization of calculations within a single space.

The paper is organized as follows. In Section 1 we describe the main reasons that have lead us to development of the hybrid computer systems programming technology. Section 2 describes the single parallel-pipeline form of COLAMO-applications developed for HCS, that allows modifications of organization of calculations. In Section 3 we describe performance reduction methods as the base for two-way scaling of calculations in hybrid computer systems. Section 4 describes implementation of the suggested programming technology as mapping of parallel applications on the hardware resource of the hybrid computer system. In conclusion we summarize the main theoretical and experimental results of the developed hybrid computer systems programming technology.

1. Programming of hybrid computer systems

A hybrid computer system (HCS) contains computational nodes with different archi-tecture and organization of calculations. Such hybrid computer systems can contain reconfigurable computational nodes and nodes of general-purpose microprocessors, such as general-purpose processors, graphic processors or accelerators Intel Xeon Phi [2]. Nowadays, in order to program such computer systems (CS) we traditionally use programming technologies of heterogeneous computer systems, such as CUDA [3], OpenACC, OpenCL [4], etc., which are based on extensions of the programming languages C, C++, FORTRAN and which take into account the architecture of special-purpose microprocessor node. These programming technologies have considerable disadvantages such as poor portability of end solutions between CSs with different architectures and configurations, and poor scalability of applications.

The main reason of these disadvantages is the HCS programming approach, which involves task decomposition in separate fragments. Each fragment is implemented in a separate computational node (on a separate device) of the hybrid computer system. So, each occupied CS node is programmed independently, and as a result, each modification of the CS configuration or of the initial application code requires re-decomposition of the task and development of local applications for each node of the CS.

It is possible to formulate principal programming problems of modern HCS which contain reconfigurable and microprocessor computational nodes:

1) FPGAs and microprocessors are programmed in different programming languages and independently of each other.
2) The application is developed specially for the current HCS configuration, and each modification of the system structure requires modification of the application code.
3) The developer is responsible for synchronization of the data flows within the task structure.

4) Porting of the application to another system with similar configuration leads to complete re-development of the application.
5) The programming and debugging time required for development of the HCS application is about 6–12 months.

That is why HCS programming requires tools for description of various kinds of organization of calculations (a single language for various architectures) and tools for translation of parallel applications, united into the technology of resource independent HCS programming. From our point of view, the technology of resource independent HCS programming is a combination of knowledge, methods, technological approaches and tools, which provides flexible modification and scaling of the application according to a new computational architecture or configuration of the computer system.

In order to provide functioning of general-purpose processor and reconfigurable computational nodes in a single space, we need a new technology of resource independent HCS programming [5], based on the following principles:

- adaptation of the application to the current HCS configuration is performed automatically by a specialized software tool – a pre-processor based on performance reduction methods [5];
- effective parameters of scaling and performance reduction must be determined without any participation of the developer, only by computer-aided programming tools;
- for computer-aided transformation to the current HCS configuration, the application must be represented in a canonical form (a single parallel-pipeline form).

Transformation of the application into the single parallel-pipeline form makes it possible to increase the task parallelism (induction) if hardware resource is growing, and to reduce (reduction) if hardware resource is decreasing, is the base for application of computer-aided tools. To implement the technology of resource independent HCS programming we must choose a programming language, which allows description of various forms of organization of calculations and programming of general-purpose processor and reconfigurable computational nodes in a single computational space.

Specialized high-level languages [6, 7] for reconfigurable computer system (RCS) [8] programming have C-like syntax, which is usual for the majority of PC developers, and differ from each other by semantic features of call and use of operators [1]. To describe parallel processes in RCS, these languages use a C-language paradigm which is initially sequential. Semantic of the C-language is oriented to interaction of sequential processes, and it does not allow the use of all abilities of RCS during development of parallel applications in these languages. This leads to a semantic gap between the initial information graph of the task, its description in the high-level language and its circuit solution generated by the translator. The result of this gap is a considerable decrease in effectiveness of the parallel application – as a rule, the performance is in 3–5 times lower in comparison with applications developed with the use of HDL-languages.

A promising direction for RCS programming is a high-level language COLAMO [1, 5], developed in Scientific Research Institute of Multiprocessor Computer Systems (SRI MCS SFU, Taganrog, Russia). The language COLAMO is used for description of parallel algorithms and generation of special-purpose computing structures within RCS architecture according to the principles of structural-procedural organization of calculations. Each special-purpose computing structure sequentially performs structurally (hardwarily) implemented fragments of the task information graph. Each graph is a computational pipeline of an instruction flow. So, the

RCS application (the RCS task) consists of a structural component, presented as a set of hardwarily implemented fragments of calculations, and of a procedural component – one and the same control program for all structural fragments, which provides sequential change of computing structures and organizes data flows.

In order to implement calculations in general-purpose processors, the language COLAMO contains instructions for description of procedural organization of calculations and provides fast transition from procedural implementation of calculations in general-purpose processors to structural organization of calculations on reconfigurable computational nodes. A structure *Implicit* is used for implicit declaration of organization of calculations (structural or procedural) for the application fragment. Re-declaration of implementation of the structure *Implicit* allows the developer to use procedural organization of calculations instead of structural one, and vice-versa without any considerable modification of the parallel program. Owing to this, the developer can create a single application using one and the same programming language for all HCS nodes. This allows the high-level programming language COLAMO to be considered as a base for the technology of resource independent programming for both reconfigurable computational nodes and general-purpose nodes of the HCS.

However, for effective HCS programming it is necessary to have language tools which allow description of fragments of calculations, which use different frequencies, data delay ratio and digital capacity of processed data. Owing to this, it is possible to scale both fragments and single circuit cores in both cases – when hardware resource is increasing or decreasing, and, in addition, it is also possible to use data with variable capacity for effective use of HCS hardware resource.

2. The single parallel-pipeline form of COLAMO-applications developed for HCS

Here and forth, the parallel-pipeline form for representation of variables and arrays is a description of data, made by means of the high-level programming language COLAMO, which provides both parallel and sequential access at the same time. In the language COLAMO such access types are declared by the keywords Vector (BitVector) and Stream (BitStream). Fig. 1 shows an example of simultaneous use of parallel and sequential data and bit access types to the arrays A, B, and C.

Such a form of applications is a canonical form; it allows automatic modification of the principal parameters of any parallel application such as the number of simultaneously implemented computational subgraphs, the capacity of processed data, the number of operations, etc. Such modification can be performed by the pre-processor tool for adaptation of applications to the HCS without any participation of the developer.

Transformation of the initial application into the canonical form is performed in two steps. In the first step variables and structures of the application are transformed for parallel-pipeline processing on data level, and in the second step – on bit level.

In general, the method of application transformation into the canonical form can be represented as follows:

1) All arrays in the initial COLAMO-application are transformed into the parallel-pipeline form. If it is necessary, Vector or Stream access types are added.
2) All variables (except for loop counters) of the parallel COLAMO-application are transformed into the format of the Union-structure, which provides both direct access to a

variable according to its type, and parallel (bitvector) and sequential (bitstream) access types.

3) All subcadrs of the parallel COLAMO-application are transformed into Implicit-structures.
4) All structures and operators of the application are transformed according to the modified parameters of the variables.
5) For the generated single parallel-pipeline form of the COLAMO-application, a global preprocessor directive of performance reduction of functional blocks, which is equal to 1, is declared.

```
Const N = 10; Const M = 100;
Const Bv = 32; Const Bs = 1;
Type Type32: Integer [BV: BitVector, BS:BitStream] of Int;
Var a,b,c : Array Type32 [N : Vector, M:Stream] Mem;
Var i, j, k, t : Number;
Cadr ExpParallelStream;
   For i := 0 to N-1 do
        For j := 0 to M-1 do
           For k := 0 to Bv-1 do
               For t := 0 to Bs-1 do
                  Begin
                      c[i,j][k,t] := a[i,j][k,t] - b[i,j][k,t];
                  End;
EndCadr;
```

Fig. 1. Concurrent use of parallel and pipeline types of access to data arrays and digits

Variables are transformed to parallel and sequential access types (a mixed access type) according to the following rules:

- if we use only sequential access to array items, then the parameter Vector, which means parallel access, with the dimension of unity is added to the declaration of the array;
- if we use only parallel access to array items, then the parameter Stream, which means sequential access, with the dimension of unity is added to the declaration of the array;
- if we use mixed access to array items, then no transformations are performed for such array.

So, when the declaration of any array is modified, its dimension is increased. For example, when we declare the array A as:

```
Var A : Array Integer [N : Stream] Mem
```

It is necessary to transform its declaration as follows:

```
Var A : Array Integer [M : Vector, K : Stream] Mem,
```

where M=1, $K = N/M$. To provide equivalency of the information graphs of the initial application and the modified one, the initial value of M is 1.

Fig. 2 shows transformation of the initial application a) to the canonical form b). As we can see from the text of the parallel-pipeline application (see Fig. 2b), all linear arrays are transformed into two-dimensional arrays with mixed access type (M : Vector, K : Stream), a new loop operator with a loop counter VC_1 is added, and all references to the variables are modified.

```
Const N = 10;
Var a, b, c, d : Array Integer [N: Vector]Mem;
Var i : Number;
Cadr ExpParallel;
   For i := 0 to N-1 do
     Begin
        If(A[i]>5)
           C[i] :=A[i]-B[i];
        Else
           C[i] :=A[i]+D[i];
     end;
EndCadr;
```

a) The initial program

```
Const N = 10; Const M = 10; Const K = N/M;
Var a, b, c, d : Array Integer [M: Vector, K: Stream]Mem;
Var i,vc_1 : Number;
Cadr ExpParallelConvData;
   For vc_1 := 0 to K-1 do
     Begin
        For i := 0 to M-1 do
           Begin
              If(A[i,vc_1]>5)
                C[i,vc_1] :=A[i,vc_1] – B[i,vc_1];
              Else
                C[i,vc_1] :=A[i,vc_1] + D[i,vc_1];
           end;
     end;
EndCadr;
```

b) The transformed program

Fig. 2. Transformation of the program to the canonical form on data level

Effective adaptation of the application for HCS requires similar transformations for bits.

3. Scaling of calculations in HCS on the base of performance reduction

The base for simple scaling and adaptation of the application in both cases – increasing and decreasing of available hardware resource – is reduction of the application performance [5], which means proportional performance reduction in all the fragments of the task information graph and possible reduction of hardware resources necessary for implementation of the computing structure. Performance reduction of parallel applications allows variation of their key parameters (the number of used circuit cores, the number of memory channels, the operand capacity, the frequency, etc.), because structural implementation of the task can lead to a lack of available hardware resource. This is especially urgent if the task is ported between HCS with different architectures and configurations. In contrast to traditional technologies and methods of multiprocessor computer system programming (MPI, CUDA, OpenACC, etc.), in which the basic information subgraph of the task is parallelized depending on the configuration of available hardware resource, the HCS programming technology, in order to apply performance reduction, requires description of the information graph of the task represented in the initial parallel form with the maximum possible degree of parallelism. According to the number of available HCS computational nodes, the initial information graph is reduced by special reduction transformations, which in a balanced manner reduce performance of all fragments of the information graph and, in several cases, reduce HCS hardware resource used by the task. It is possible to single out four kinds of reduction:

- performance reduction according to circuit operations;
- performance reduction according to memory channels;
- performance reduction according to capacity;
- performance reduction according to frequency.

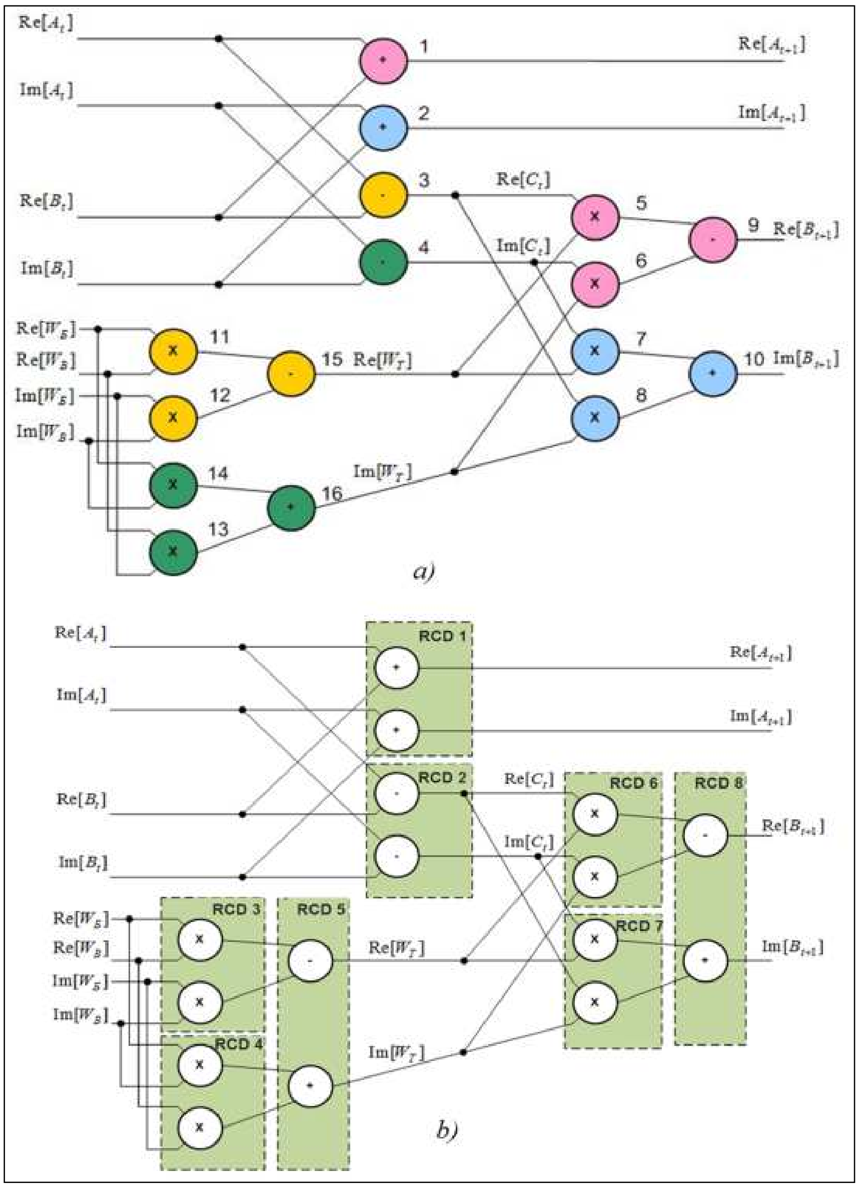

Fig. 3. Principles of performance reduction according to circuit operations illustrated by the operation of fast Fourier transformation (a – the initial information graph of the FFT operation, b – structural implementation of the FFT operation on RCS with conditional filling of FPGAs)

Performance reduction according to circuit operations is based on reducing the number of circuit blocks operating simultaneously and performing computing operations. The example of the fast Fourier transformation (FFT), shown in Fig. 3 and Fig. 4, illustrates principles of performance reduction according to circuit operations.

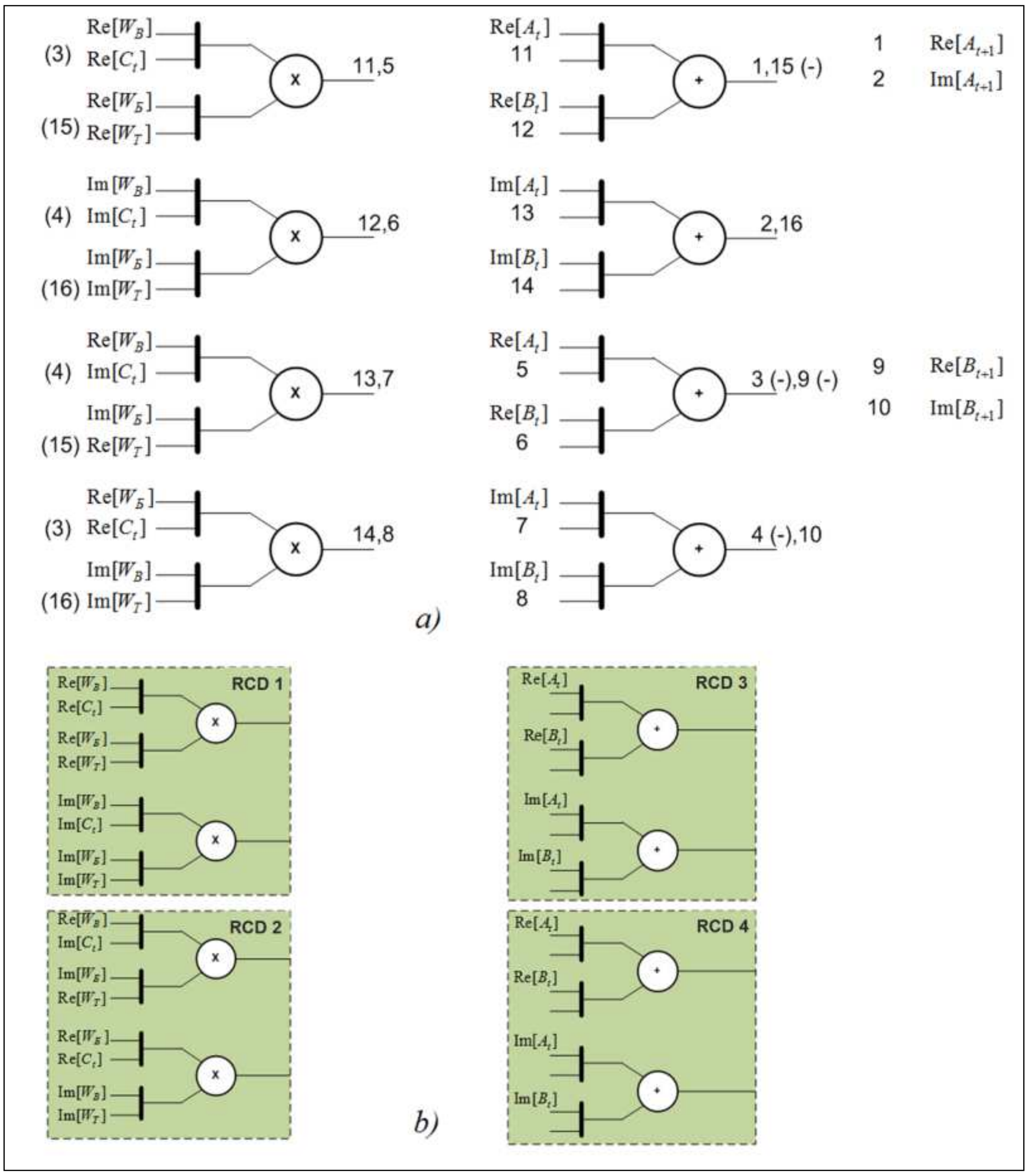

Fig. 4. Result of performance reduction according to circuit operations with the degree equal to 2 for the basic operation of the FFT (a – the information graph of the FFT reduced in 2 times, b – structural implementation of the FFT on RCS with conditional filling of FPGAs)

One of the most important types of reduction, which provide HCS resource independent programming, is performance reduction according to memory channels – an operation of concerted reduction of the number of concurrently used memory channels for some fragment of the task information graph. Fig. 5 and Fig. 6 illustrate principles of performance reduction according to memory channels.

Fig. 5 shows a fragment of the initial information graph, which contains 4 input channels. Fig. 6 shows the result of performance reduction according to memory channels with the degree equal to 2. After performance reduction according to memory channels the number of concurrently performed circuit operations is not changed but the time of processing data flows increases in 2 times. This decreases the performance of the fragment in 2 times.

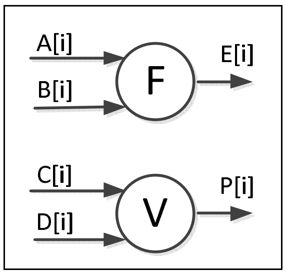

Fig. 5. An initial information graph of some reducible fragment

The result of performance reduction according to capacity is not the reduced number of operations in a computing structure, but the reduced capacity of processing data owing to the use of operations with smaller capacity. This leads to increasing of the processing time and to decreasing of hardware burden for implementation of the computing structure. After performance reduction according to capacity, data flows are controlled by a multiplexer, and data flows enter the computing structure with the data delay ratio equal to the degree of performed reduction.

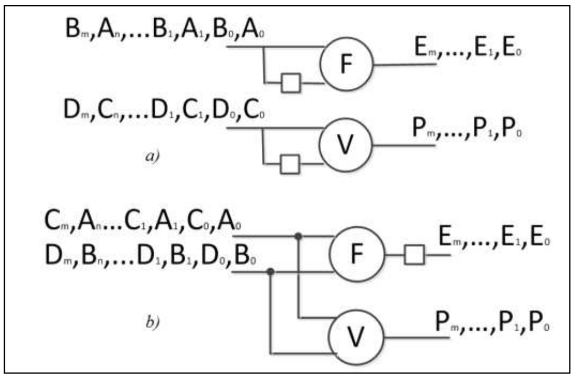

Fig. 6. Result of performance reduction according to memory channels with the degree equal to 2 (a – performance reduction according to memory channels with data placement in one channel for each circuit operation, b – performance reduction according to memory channels with data placement in different channels for each circuit operation).

Performance reduction according to frequency is applied for increasing of the data flow processing time proportionally to the degree of reduction at the cost of multiple reduction of the fragment frequency. Here, the delay ratio of data, entering the computing structure, remains constant. Performance reduction is an auxiliary reduction transformation, which cannot be used separately, and is only applied for matching processing rates between reduced and non-reduced fragments of the information graph in the task structure.

For practical use of the considered reduction transformations the developer must mark the fragments of the parallel application, written in the high-level programming language COLAMO, which can be reduced, by pre-processor directives. The directive of reduction can be declared as follows:

```
#Reduction of <type of reduction> <degree of reduction>;
        Block of operators
EndReduction;
```

During translation of the parallel application the pre-processor automatically transforms the information graph of the application to the available HCS configuration according to the reduction directives placed by the developer. This provides adaptation of the application to the current HCS configuration without considerable modification of the initial program code.

In conclusion of analysis of the transformations, we can describe functioning of the technology of resource independent programming of hybrid computer systems, which contain reconfigurable and microprocessor computational nodes, as follows. The pre-processor unit of the language COLAMO, which performs analysis of the source parallel application, transforms it into the canonical form, determines hardware resource required for its implementation and compares it with the current HCS configuration. Then it determines the maximum reduction degree and types of all required transformations. If the current HCS hardware resource is sufficient for implementation of the application, then bitstream files and loadable files are generated for all HCS nodes involved into implementation of the task. Otherwise special reduction transformations are performed. They reduce the application performance and the involved HCS hardware resource in a balanced manner. Performance reduction is performed as follows: reduction according to simultaneously performed subgraphs of the application, reduction according to capacity, reduction according to instructions (cores), reduction according to data delay ratio (clock rate). The analysis unit of the pre-processor reduces the involved hardware resource for each type of reduction, taking into account its theoretically permissible degree. It determines the most reasonable use of reduction which can provide the maximum possible performance of the application for the current HCS configuration. The text of the reduced parallel application, generated by the pre-processor in an automatic mode, is passed to the translator of the programming language COLAMO, which generates a detailed information graph of the application. The information graph of the application, which contains fragments structurally implemented in reconfigurable computational nodes and procedurally implemented in microprocessor computational nodes, is passed to a synthesizer tool, which automatically distributes the fragments among reconfigurable and microprocessor computational nodes available in the current HCS configuration.

4. Mapping of parallel applications on HCS hardware resource

The problem of distribution of parallel application fragments on hardware resource of a multiprocessor HCS is in automatic decomposition of computational structure of the parallel application, described by the information graph, into disjoint fragments, and in distribution of these fragments into separate computational nodes of the HSC, such as reconfigurable computational nodes and microprocessor computational nodes). Besides, the fragments placed in reconfigurable computational nodes are decomposed into smaller fragments, each of which is implemented in a separate FPGA chip.

All blocks of the computational structure of the parallel application can be divided into two groups. The first group contains hardwarily implemented blocks, which must be mapped into reconfigurable computational nodes. The second group contains blocks, which correspond to the *Implicit* structures. Each block has its own C++ or C# procedure. The blocks from the second group are placed into HCS microprocessor computational nodes.

Automatic mapping of the computational structure of the parallel application on HCS hardware resource consists of three steps. The first step is decomposition of the computational structure of the parallel application into disjoint fragments, which must be placed in the nodes of the multiprocessor HCS. Only one block is placed in each microprocessor node of the computational structure, because in one node it is possible to run only one sequential subroutine with intensive data exchange with other microprocessor and reconfigurable nodes. Fragments of parallel applications, which contain hardwarily implemented blocks, are placed into reconfigurable computational nodes. In this case, the total hardware resource occupied by the blocks of one fragment must not exceed the total hardware resource of FPGA chips of one reconfigurable computational node.

The second step of the algorithm of application synthesis is distribution of the fragments, which are placed in reconfigurable computational nodes, into separate FPGA chips of these nodes.

The third and the last step of the algorithm of application synthesis is synchronization of external and internal data flows of the computational structure of the parallel application, and placement of interface units for matching data exchange between heterogeneous fragments of the computational structure. These interfaces match data exchange between procedural and pipeline computational nodes, between the nodes which are operating at different clock frequencies, between the nodes which are operating at similar clock frequencies but connected to different clock generators (because of different phases and inaccuracy of clock generators), etc.

After setting of all required interfaces and synchronization elements, the synthesizer tool generates bitstream files *.bit for reconfigurable nodes and loadable files *.exe for microprocessor nodes of the HCS, and generates a control program which controls computational process and is single for all HCS computational modules.

Conclusion

For effective programming of hybrid computer systems we suggest the high-level programming language COLAMO as a part of the developed technology of resource-independent programming. The language COLAMO allows description of various forms of organization of calculations in one and the same computational space. The suggested single parallel-pipeline form of applications along with the developed performance reduction methods provide automatic adaptation of applications to the modified HCS architecture or configuration. Owing to the suggested technology we can reasonably use resources of nodes with different architectures during HCS programming, and we have a set of necessary tools for quick development of effective resource-independent scalable parallel applications in a single language space. It simplifies HCS programming and speeds up development of parallel applications.

We tested the developed HCS programming technology, using three test tasks from different problem areas: digital signal processing, symbolic processing and monitoring of computer networks.

The task of digital signal processing consists in processing of input data according to the direct fast Fourier transformation algorithm with frequency decimation. The task of symbolic processing consists in transformation of input data according to the algorithm of symmetric block encryption GOST 28147-89. In the third task we implement a procedure of patterns search in a data flow, which detects frequencies of all found patterns within one string and within the input data flow.

For all three tasks we developed applications in the language COLAMO, and each application was translated for different HCS architectures with different quantity of computational nodes. Our research has proved effectiveness of the developed technology for adaptation of the parallel application to the used hardware resource in both cases – when resource is increasing and when the application is ported to the HCS with smaller hardware resource.

The project has been funded in part by the scholarship of the President of the Russian Federation for young scientists and graduate students (SP-173.2016.5).

References

1. Kalyaev I.A., Levin I.I., Semernikov E.A., Shmoilov V.I. Reconfigurable multipipeline computing structures. New York, Nova Science Publishers, 2012. 330 p.
2. Dong X., Chai J., Yang J., Wen M., Wu N., Cai X., Zhang C., Chen Z. Utilizing multiple xeon Phi coprocessors on one compute node. *Lecture Notes in Computer Science*, 2014, Vol. 8631, Issue PART 2, pp. 68–81. DOI:10.1007/978-3-319-11194-0_6
3. Liang T.-Y., Li H.-F., Lin Y.-J., Chen B.-S. A Distributed PTX Virtual Machine on Hybrid CPU/GPU Clusters. *Journal of Systems Architecture*, 2016. Vol. 62. pp. 63–77. DOI: 10.1016/j.sysarc.2015.10.003
4. Li H.-F., Liang T.-Y., Lin Y.-J. An OpenMP programming toolkit for hybrid CPU/GPU clusters based on software unified memory. *Journal of Information Science and Engineering*, 2016, Vol. 32, Issue 3. pp. 517–539.
5. Dordopulo A., Levin I., Kalyaev I., Gudkov V., Gulenok A.. Programming of hybrid computer systems based on the performance reduction method. *CEUR Proceedings*, 2016, Vol. 1576, pp. 131–140.
6. El-Araby E., Taher M., Abouellail M., El-Ghazawi T., Newby G.B. Comparative analysis of high level programming for reconfigurable computers: Methodology and empirical study. *2007 3rd Southern Conference on Programmable Logic*, Mar del Plata; 2007; pp. 99–106. DOI: 10.1109/SPL.2007.371731
7. Xu J, Subramanian N, Alessio A, Hauck S. Impulse C vs. VHDL for accelerating tomographic reconstruction. *18th IEEE International Symposium on Field-Programmable Custom Computing Machines*, 2010, pp. 171–174. DOI: 10.1109/fccm.2010.33
8. Dordopulo A., Kalyaev I., Levin I., Slasten L. High-performance reconfigurable computer systems. *Lecture Notes in Computer Science*, 2011, Vol. 6873. Chapter Parallel Computing Technologies. pp. 272–283. DOI:10.1007/978-3-642-23178-0_24

Gulenok Andrey Aleksandrovich, PhD, Senior staff scientist, Academician A.V. Kalyaev SRI multiprocessor computer system at Southern Federal University (Taganrog, Russian Federation)

Dordopulo Alexey Igorevich, PhD, Head of Department, Scientific Research Center of Supercomputers and Neurocomputers (Taganrog, Russian Federation)

Levin Ilya Izrailevich, Dr. Sc., professor, Director of Scientific Research Center of Supercomputers and Neurocomputers (Taganrog, Russian Federation)

Gudkov Vyacheslav Aleksandrovich, PhD, Senior staff scientist, Scientific Research Center of Supercomputers and Neurocomputers (Taganrog, Russian Federation)

УДК 004.382.2 **DOI: 10.14529/cmse170105**

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ГИБРИДНОГО ТИПА С АДАПТАЦИЕЙ И МАСШТАБИРОВАНИЕМ ВЫЧИСЛЕНИЙ

А.А. Гуленок[1], А.И. Дордопуло[2], И.И. Левин[2], В.А. Гудков[2]

[1] *Научно-исследовательский институт многопроцессорных вычислительных систем Южного федерального университета (347928 Таганрог, ул. Чехова, д. 2, ГСП-284)*
[2] *Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров (347900 Таганрог, Итальянский пр., д. 106)*

Поступила в редакцию: 20.10.2016

В статье рассматривается технология программирования вычислительных систем гибридного типа, содержащих реконфигурируемые и микропроцессорные вычислительные узлы. *В качестве основы технологии программирования вычислительных систем гибридного типа предлагается язык программирования высокого уровня COLAMO с расширениями, с помощью которых можно описывать различные виды параллельных вычислений – структурную, структурно-процедурную, мультипроцедурную и процедурную формы организации вычислений в единой параллельно-конвейерной (канонической) форме. Предложенная параллельно-конвейерная форма позволяет изменять формы организации вычислений автоматизировано препроцессором языка COLAMO с учетом текущей конфигурации вычислительной системы гибридного типа. На основе канонической формы и возможностей описания различных форм организации вычислений на языке программирования высокого уровня COLAMO предложена технология ресурсонезависимого программирования, которая позволяет адаптировать программу под изменившиеся архитектуру или конфигурацию вычислительной системы гибридного типа в автоматическом режиме без корректировки кода программистом. Для этого исходная параллельная программа на языке программирования COLAMO препроцессором преобразуется в каноническую форму, после чего препроцессор проводит оценку доступного вычислительного ресурса, определяет эффективные параметры реализации программы на доступном ресурсе и, при необходимости, выполняет редукцию производительности программы для адаптации под текущую конфигурацию вычислительной системы гибридного типа. Технология позволяет осуществлять масштабирование в обе стороны как в случае увеличения доступного вычислительного ресурса (индукция), так и в случае сокращения доступного вычислительного ресурса (редукция), что обеспечивает ресурсонезависимость программирования при разработке программы — программист не привязывается к доступному аппаратному ресурсу вычислительной системы.*

Ключевые слова: редукция производительности, язык программирования высокого уровня, программирование вычислительных систем гибридного типа, адаптация программы, масштабирование программы.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Gulenok A.A., Dordopulo A.I., Levin I.I., Gudkov V.A. Hybrid computer system programming technology with adaptation and scaling of calculations // Вестник ЮУрГУ Серия: Вычислительная математика и информатика. 2017. Т. 6. № 1. С. 73–86. DOI: 10.14529/cmse170105.

Литература

1. Kalyaev I.A., Levin I.I., Semernikov E.A., Shmoilov V.I. Reconfigurable multipipeline computing structures. New York, Nova Science Publishers, 2012. 330 p.
2. Dong X., Chai J., Yang J., Wen M., Wu N., Cai X., Zhang C., Chen Z. Utilizing multiple Xeon Phi coprocessors on one compute node // Lecture Notes in Computer Science, 2014, Vol. 8631, Issue PART 2, P. 68–81. DOI:10.1007/978-3-319-11194-0_6
3. Liang T.-Y., Li H.-F., Lin Y.-J., Chen B.-S. A Distributed PTX Virtual Machine on Hybrid CPU/GPU Clusters // Journal of Systems Architecture, 2016. Vol. 62. P. 63–77. DOI: 10.1016/j.sysarc.2015.10.003
4. Li H.-F., Liang T.-Y., Lin Y.-J. An OpenMP programming toolkit for hybrid CPU/GPU clusters based on software unified memory // Journal of Information Science and Engineering, 2016, Vol. 32, Issue 3. P. 517–539.
5. Dordopulo A., Levin I., Kalyaev I., Gudkov V., Gulenok A.. Programming of hybrid computer systems based on the performance reduction method // CEUR Proceedings, 2016, Vol. 1576. P. 131–140.
6. El-Araby E., Taher M., Abouellail M., El-Ghazawi T., Newby G.B. Comparative analysis of high level programming for reconfigurable computers: Methodology and empirical study // 2007 3rd Southern Conference on Programmable Logic, Mar del Plata; 2007; P. 99–106. DOI: 10.1109/SPL.2007.371731
7. Xu J, Subramanian N, Alessio A, Hauck S. Impulse C vs. VHDL for accelerating tomographic reconstruction // 18th IEEE International Symposium on Field-Programmable Custom Computing Machines, 2010, P. 171–174. DOI: 10.1109/fccm.2010.33
8. Dordopulo A., Kalyaev I., Levin I., Slasten L. High-performance reconfigurable computer systems // Lecture Notes in Computer Science, 2011, Vol. 6873. Chapter Parallel Computing Technologies. P. 272–283. DOI:10.1007/978-3-642-23178-0_24.