Information systems. Computer sciences. Issues of information security

Информационные системы. Информатика. Проблемы информационной безопасности

UDC 004.2
https://doi.org/10.32362/2500-316X-2024-12-3-37-45
EDN PXKDKR

RESEARCH ARTICLE

Method for designing specialized computing systems based on hardware and software co-optimization

Ilya E. Tarasov @, Peter N. Sovietov, Daniil V. Lulyava, Dmitry I. Mirzoyan

MIREA – Russian Technological University, Moscow, 119454 Russia
@ Corresponding author, e-mail: tarasov_i@mirea.ru

Abstract

Objectives. Following the completion of development stages due to transistor scaling (Dennard's law) and an increased number of general-purpose processor cores (limited by Amdahl's law), further improvements in the performance of computing systems naturally proceeds to the stage of developing specialized computing subsystems for performing specific tasks within a limited computational subclass. The development of such systems requires both the selection of the relevant high-demand tasks and the application of design techniques for achieving desired indicators within the developed specializations at very large scales of integration. The purpose of the present work is to develop a methodology for designing specialized computing systems based on the joint optimization of hardware and software in relation to a selected subclass of problems.

Methods. The research is based on various methods for designing digital systems.

Results. Approaches to the analysis of computational problems involving the construction of a computational graph abstracted from the computing platform, but limited by a set of architectural solutions, are considered. The proposed design methodology based on a register transfer level (RTL) representation synthesizer of a computing device is limited to individual computing architectures for which the relevant circuit is synthesized and optimized based on a high-level input description of the algorithm. Among computing node architectures, a synchronous pipeline and a processor core with a tree-like arithmetic-logical unit are considered. The efficiency of a computing system can be increased by balancing the pipeline based on estimates of the technological basis, and for the processor—based on optimizing the set of operations, which is performed based on the analysis of the abstract syntax tree graph with its optimal coverage by subgraphs corresponding to the structure of the arithmetic logic unit.

Conclusions. The considered development approaches are suitable for accelerating the process of designing specialized computing systems with a massively parallel architecture based on pipeline or processor computing nodes.

Keywords: processor, RTL, synthesis, translator

• **Submitted:** 18.10.2023 • **Revised:** 04.12.2023 • **Accepted:** 22.03.2024

For citation: Tarasov I.E., Sovietov P.N., Lulyava D.V., Mirzoyan D.I. Method for designing specialized computing systems based on hardware and software co-optimization. *Russ. Technol. J*. 2024;12(3):37–45. https://doi.org/10.32362/2500-316X-2024-12-3-37-45

Financial disclosure: The authors have no a financial or property interest in any material or method mentioned.

The authors declare no conflicts of interest.

НАУЧНАЯ СТАТЬЯ

Методика проектирования специализированных вычислительных систем на основе совместной оптимизации аппаратного и программного обеспечения

И.Е. Тарасов [@], П.Н. Советов, Д.В. Люлява, Д.И. Мирзоян

МИРЭА – Российский технологический университет, Москва, 119454 Россия
[@] Автор для переписки, e-mail: tarasov_i@mirea.ru

Резюме

Цели. Следующим этапом повышения производительности вычислительных систем после завершения этапов роста за счет масштабирования транзисторов (закон Деннарда) и за счет увеличения количества процессорных ядер общего назначения (ограничиваемого законом Амдала) является переход к разработке специализированных вычислительных подсистем для работы в ограниченном подклассе задач. Создание таких систем требует как выбора соответствующих массово востребованных задач, так и применения методик проектирования, обеспечивающих достижение высоких технико-экономических показателей разрабатываемых специализированных сверхбольших интегральных схем. Цель работы – разработка методики проектирования специализированных вычислительных систем на основе совместной оптимизации аппаратного и программного обеспечения применительно к выбранному подклассу задач.

Методы. Использованы методы проектирования цифровых систем.

Результаты. Рассмотрены подходы к анализу вычислительных задач путем построения графа вычислений, абстрагированного от вычислительной платформы, однако ограниченного набором архитектурных решений. Предложена методика проектирования, использующая маршрут, основанный на применении синтезатора представления уровня регистровых передач (RTL-представления) вычислительного устройства, ограниченного отдельными вычислительными архитектурами, для которых производятся синтез и оптимизация схемы на основе высокоуровневого входного описания алгоритма. Среди архитектур вычислительных узлов рассмотрены синхронный конвейер и процессорное ядро с древовидным арифметико-логическим устройством. Повышение эффективности вычислительной системы осуществляется путем балансировки конвейера на основе оценок технологического базиса, а для процессора – путем оптимизации набора операций на основе анализа графа абстрактного синтаксического дерева с его оптимальным покрытием подграфами, соответствующим структуре арифметико-логического устройства.

Выводы. Рассмотренные подходы к разработке позволяют ускорить процесс проектирования специализированных вычислительных систем с массово-параллельной архитектурой, основанных на конвейерных вычислительных узлах.

Ключевые слова: процессор, RTL, синтез, транслятор

• **Поступила:** 18.10.2023 • **Доработана:** 04.12.2023 • **Принята к опубликованию:** 22.03.2024

Для цитирования: Тарасов И.Е., Советов П.Н., Люлява Д.В., Мирзоян Д.И. Методика проектирования специализированных вычислительных систем на основе совместной оптимизации аппаратного и программного обеспечения. *Russ. Technol. J*. 2024;12(3):37–45. https://doi.org/10.32362/2500-316X-2024-12-3-37-45

Прозрачность финансовой деятельности: Авторы не имеют финансовой заинтересованности в представленных материалах или методах.

Авторы заявляют об отсутствии конфликта интересов.

INTRODUCTION

The field of element base design for high-performance computing in Russia is characterized by trends corresponding to objective technical limitations and the need to intensify import substitution processes to ensure technological sovereignty. Analysis of the desired technical characteristics should take into account production capacity limitations, as well as the need to reduce technical and economic risks.

Analysis of architectural trends in the field of computing is presented by Patterson and Hennessy in [1]. The authors draw attention to a number of major milestones in the development of processor architectures since the 1970s. The first of these concerns the replacement of the CISC (complex instruction set computer) concept with the RISC (reduced instruction set computer) approach. Due to the ensuing reduction in combinational logic complexity, it became possible to increase the clock frequency of processor devices.

Further increases in clock frequency, however, turned out to be limited according to Dennard's Law, which describes an inherent limit to the possibility of increasing processor performance by scaling transistor sizes. The response to this limitation was the transition to multi-core processors, which took place in the mass segment of personal computers during the mid-2000s.

Improved performance by increasing the number of processor cores is in turn limited by Amdahl's law, which defines the potential performance improvement of a multiprocessor complex by defining the fraction of computations that can be performed in parallel. A related problem is the so-called *wall of interfaces* [2], which takes into account the fact that performance increases quadratically with decreasing process standards, while the bandwidth of memory and peripheral interfaces increases linearly. Therefore, the construction of multicore systems entails the problem of organizing inter-processor data exchange, which cannot be effectively implemented due to the outstripping growth of the volume of received data compared to the possibility of their transmission through existing communication channels designed according to comparable technological standards.

In [1], in connection with the above problems, a transition to domain-specific architectures is proposed by analogy with domain-specific languages (DSL). In this case, the specialization of a processor for performing certain classes of computations implies its reduced efficiency in other classes, necessitating the selection of specialized computational tasks corresponding to current technical needs. Such an approach can find wide application due to considerations of reducing design and production unit costs, as well as corresponding to technically feasible approaches to the design of digital devices.

ANALYSIS OF COMPUTATIONAL TASKS FOR IMPLEMENTATION AS A PART OF SPECIALIZED COMPUTING SYSTEMS

Important areas for the application of high-performance computing systems include video processing systems, virtual and augmented reality, robotics, industrial automation, digital radio communications, measurement technology [3, 4]. Among the implemented areas of signal processing are digital filtering [5], spectral analysis [6], machine learning algorithms [7], including those based on specialized neural processors [8], or reconfigurable gas pedals based on field-programmable gate array (FPGA).

Computational tasks that can be accelerated by specialized computing systems include the following subclasses:

1) solving differential equation systems by numerical methods;
2) operations with three-dimensional images;
3) digital signal processing based on mass application of "multiply and accumulate" operations;
4) computation of hash functions in information protection tasks;
5) implementation of neural networks in terms of calculating neural function values (neural net inference), not including neural net training tasks.

Types of operations characteristic of the above tasks are given in the Table. The columns of the table are placed in such a way that the complexity of realization of the corresponding types of operations increases from left to right. Although significant difficulties are associated with high-complexity memory operations based on increasing the throughput capacity of a memory subsystem, some memory operations themselves do not have high complexity.

The Table uses the following scores to characterize use intensity of certain types of operations. The assessment *no*, which corresponds to the situation when the operation is not used in algorithms, does not require support. While the assessment *possibly* characterizes situations when such operations take place, these do not have a noticeable impact on the efficiency of the computing device due to their rare use. For such operations, it is possible to use non-optimized solutions or ready-made components having functional redundancy. The assessment *massively*, which corresponds to operations forming the basis of algorithms, determines the intended implementation efficiency of the computing device to the greatest extent.

According to the preliminary estimates, the implementation of hash functions and digital signal processing operations demonstrates the advantages of pipelined architectures to a greater extent, since it provides for streaming data processing without intensive exchange with external memory.

Table. Intensity of use of operations characteristic of a number of tasks requiring the use of high-performance computing systems

Task type	Shifts, addition, bitwise operations	Multiplication	Floating point operations	Transcendental functions	Memory operations
Hash functions	Massively	No	No	No	Possibly
Neural networks implementation	No	Massively	Possibly	Possibly	Possibly
Digital signals processing (filtration)	Massively	Massively	Possibly	Possibly	Possibly
Differential equation systems	No	Possibly	Massively	Possibly	Massively
3D graphics processing	No	Often	Massively	Often	Massively

COMPUTING NODE ARCHITECTURES

Architectural approaches to the realization of individual computational nodes in modern digital electronics are quite diverse. For their effective practical use, however, it is necessary to limit ourselves to a set of possible solutions for using computer-aided design methods at the level of mathematical and software models. Further transformations in the representation of the register transfer level circuit (RTL-representation) do not make significant changes in the characteristics of such a system. It can be noted, for example, that the use of means of description of high-level language (HLL) class implies automated construction of control diagrams (flow control) designed for a wide class of realized architectural approaches. This leads to excessive complication of synthesized control diagrams.

The following architectural approaches to the construction of computational nodes are considered for the practically realizable design methodology:

1) processor unit;
2) synchronous pipeline;
3) modification of synchronous pipeline with the possibility of reusing of individual stages.

Structural diagrams of the main computing nodes are shown in Fig. 1.

Given variants of nodes are considered as architectural templates for realization of selected subclasses of computations. In this case, the required set of operations for the processor is realized as part of the arithmetic-logic unit (ALU), while for the pipeline, it is realized in successive stages. To use a pipeline, it is necessary for the order of operations for the realization of the algorithm to remain unchanged, otherwise the adjustment will require complication of control diagrams. Provided that the frequency of receiving input data is significantly less than the clock frequency of the pipeline, repetition of operations can be used in algorithms such as computation of hash functions and realization of filters with finite impulse response. When this condition is fulfilled, the same stage of the pipeline can be used repeatedly until the arrival of a new input value.

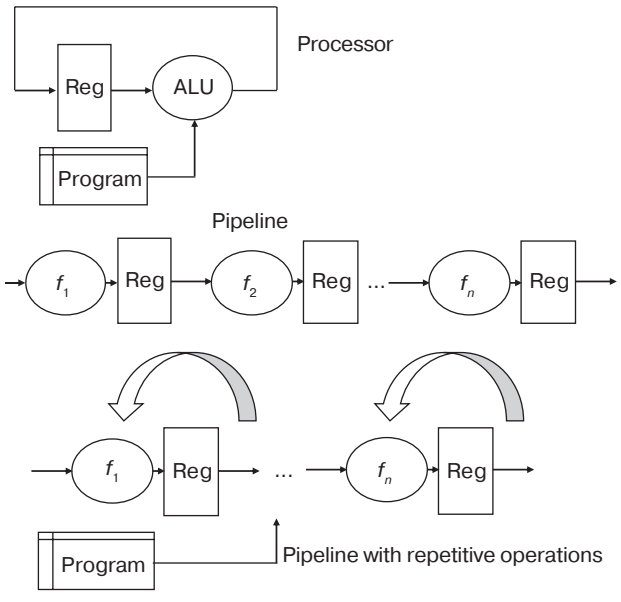

Fig. 1. Diagrams of the main computing nodes (architectural templates). f_1, f_2, ..., f_n are the functional devices

Since operations involving RTL-representation synthesis and optimization depends on the choice of a particular architectural pattern, the present work article

considers the application of the technique for pipelined computational structures. The addition of processor and pipeline architectural templates with operational repetition is provided for subsequent stages of the project.

Design route according to the proposed methodology of designing computing modules of a specialized computing system is shown in Fig. 2.

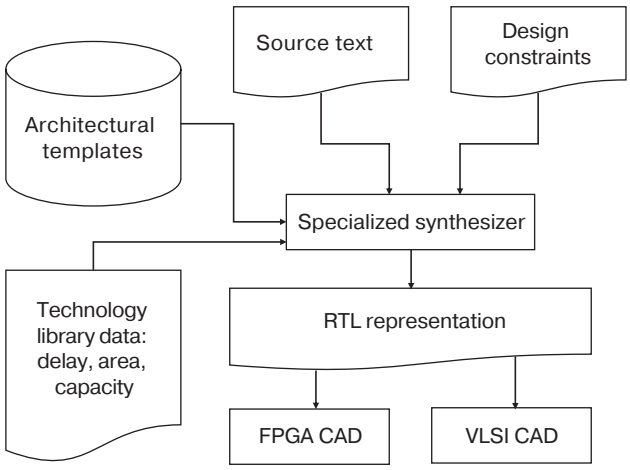

Fig. 2. Route for designing modules of a specialized computer system

According to the presented route, it can be seen that the input data are the source code of the algorithm to be implemented, while design constraints are presented in the form of the limiting characteristics of the required solution. The specialized synthesizer developed by P.N. Sovetov [9] based on architectural templates generates the RTL representation of a module using the data on the technological library for preliminary evaluation of its characteristics. The resulting RTL representation is further used in FPGA or very large-scale integration (VLSI) design paths, where the corresponding computer-aided design systems (CAD) are used to estimate the module characteristics following synthesis or after performing placement and tracing (which gives a more accurate estimation of characteristics compared to the estimation after synthesis).

For example, the values of signal propagation delays become the basis for re-synthesis of RTL representation with additional pipelining of the identified critical circuits. In addition, the synthesizer provides additional information about the interconnections of the synthesized nodes to generate design constraints specifying the coordinates of individual nodes of the synthesized circuit (stages of the pipeline). While such capabilities are also partially provided by such design tools as *Vitis HLS*[1], the characteristics of the technological platform are set out in the form of libraries. Using the route developed as part of the present work, these characteristics can to be refined during the process of project optimization.

OPTIMIZATION OF COMPUTATIONAL NODE BASED ON SOFTWARE MODEL OF COMPUTATIONS

The development of compilers for new processor architectures is an important element in the provision of tools for designing systems based on them [10]. Translators of DSL into RTL representation represent a promising approach to fast design of hardware gas pedals for some narrow class of architectures [11]. Let us briefly outline the design steps of a tool system consisting of an embedded DSL based on a subset of the Python language and a translator for synthesizing hardware gas pedals based on pipelining of a linear program section.

The user program, representing a behavioral description of the hardware gas pedal being synthesized, is input to the translator. This program is automatically converted into the form of an abstract syntax tree (AST) using the ast module from the Python standard library. The AST checks and propagates information about the types used in the input program based on Python's type annotation mechanism.

In addition, the user provides a table of delays and operation combination rules for the selected chip type, as well as one of the selected pipeline control templates in the Verilog language. The result of the translator is the code of the synthesized pipelined gas pedal in the Verilog language.

After all functions in the input program are embedded in the main function, the loops are fully unrolled. Then the program is transformed into an acyclic data dependency graph (DDG), where the convolution and promotion of constants, removal of matching expressions, and removal of dead code are performed based on the numbering of values. In order to achieve a potentially greater parallelism of calculations, balancing of expression tree heights is carried out.

Prior to the direct synthesis of the pipeline, auxiliary steps are performed: a step of partial coverage of the DDG using multiple input single output (MISO) subgraphs, as well as a step of calculating the maximum delays between pairs of nodes in the DDG.

The target chip type may use resources that enable combination, i.e., combining the execution of several operations in time, for example, using a look-up table (LUT). The translator uses partial DDG coverage of combined operations using a variant of the MAXMISO algorithm for instruction synthesis.

[1] https://www.xilinx.com/products/design-tools/vitis/vitis-hls.html. Accessed October 10, 2023.

This algorithm allows us to enumerate in DDG non-intersecting subgraphs having the number of inputs not more than a given number and one output.

The result of the pipeline synthesis is DDG with added nodes representing pipelining registers. By realizing the pipeline synthesis mainly using third-party constraint programming and linear programming solvers, the implementation of the code generator is simplified. The pipeline synthesis is implemented in one of the following ways: with minimization of the pipeline depth, with minimization of the total size of the pipeline registers, or with minimization of the pipeline depth followed by minimization of the total size of the pipeline registers. The pipeline is synthesized using information about the end-user nodes that takes the DDG representation of the program into account. This simplifies the algorithm synthesis process by reducing the number of generated constraints. A similar approach was used to create a compiler for a specialized controller based on a field-programmable gate array (FPGA) [12]. Here, the presence of similar approaches to the translation of high-level program representation [13, 14] can be noted; however, these do not involve feedback from CAD, which determines the actually achieved time delays by tracing the project. At the same time, attention is paid to pipelined architectures for FPGA [15].

METHODOLOGY FOR EVALUATING THE TOPOLOGICAL IMPLEMENTATION OF A COMPUTATIONAL NODE

The synthesized RTL representation discussed above uses the estimated delays introduced by individual operations as input information. For the selected technological basis, it is therefore necessary to:

1) determine the delays of individual elements that implement the computations supported by the synthesizer;
2) identify the possibility of using an additive delay model or determine a way to determine the total delay of a combinational node taking into account the interaction of individual elements.

Verification of the topological realization of the example of the pipeline calculator was carried out in *AMD/Xilinx Vivado*[2] FPGA CAD. The pipeline calculator implements the COordinate Rotational DIgital Computer (CORDIC) vector rotation algorithm, forming the basis for the computation of transcendental functions. This choice is related to the inclusion of the CORDIC IP core in the library components of *AMD/Xilinx Vivado* CAD; thus, its characteristics can be compared with the obtained results. The calculation of the steps of the CORDIC algorithm is combined with the sequential calculation of the result of multiplication with accumulation. In this way, an excess of functionality with respect to the CORDIC IP core is ensured. A quantitative criterion for assessing the quality of preliminary delay modeling is the histogram of time delay stocks (slack histogram)[3]. In static timing analysis, the slack value is the difference between the value of the clock signal period and the maximum signal propagation delay between synchronous nodes of the circuit. Depending on the complexity of expressions and mutual arrangement of nodes, the delay will be individual for each circuit. On this basis, a histogram is constructed showing the number of circuits that have appropriate time reserves before the arrival of the next edge of the clock signal. Such a histogram, which is referred to here as a stock histogram, is generated in *Vivado* CAD at the operator's request based on the static time analysis performed in CAD.

Based on the considerations of balancing the stages of the pipeline in the ideal case, we can assume that the stocks will be grouped around the minimum values to indicate the absence of circuits having an excessively short delay and consequent inefficient use of hardware resources. An example of a histogram of time delay stocks is shown in Fig. 3.

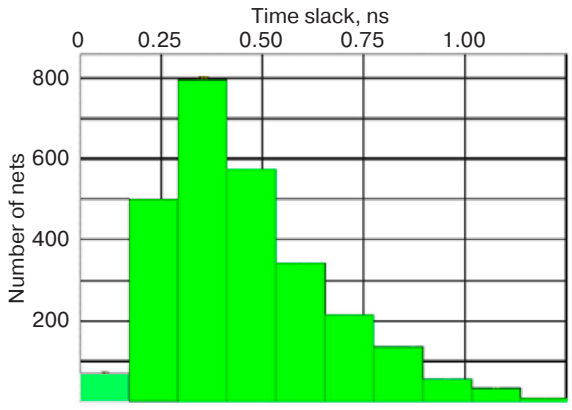

Fig. 3. Histogram of time delay stocks for the pipeline example

The grouping of circuits on the histogram shows that balancing of the pipeline stages is performed correctly: the main proportion of circuits falls in the left part of the histogram, which corresponds to small values of time reserve.

[2] https://docs.xilinx.com/r/en-US/ug910-vivado-getting-started. Accessed October 10, 2023.

[3] https://docs.xilinx.com/r/en-US/ug906-vivado-design-analysis/Timing-Analysis. Accessed October 10, 2023.

EXAMPLES OF PRACTICAL TESTING OF THE METHODOLOGY

Practical approbation of the method was carried out on the basis of a number of computational nodes of pipeline type. A configurable pipeline with the combination of calculations of the result of multiplication with accumulation and vector rotation based on the CORDIC algorithm was realized. The synthesized pipeline can be used to calculate a pair of values of sine, cosine or multiply independent 32-bit operands in the mode of switching functional nodes.

When estimating the achievable clock speed of FPGA-based digital circuits, the concept of logic levels is used to refer to the number of nodes connected in series in a circuit of maximum length. This circuit is a limiting factor where the achievable clock frequency is preliminarily estimated as the system clock frequency divided by the logic levels figure. With a system clock frequency on the order of 700–750 MHz for a modern FPGA architecture, achieving logic levels equal to 1 is quite a challenging technical task. Nevertheless, the balancing of the pipeline allowed us to obtain a clock signal period of 1.6–1.7 ns, which corresponds to a clock frequency of 600–625 MHz for the *AMD/Xilinx Kria* platform constructed according to 16 nm FinFET[4] technological standards.

CONCLUSIONS

Approaches considered in the article enable the optimization of pipeline calculators designed to work as part of VLSI. The obtained positive results can be used to extend the methodology for designing processor nodes and pipelines with repetition of operations, whose architectural templates were considered in the introductory part of the article. The calculator is optimized in accordance with the selected quality criteria by means of joint analysis of the design at several levels (software model, circuit, and topological representations), including controlled increase of clock frequency for high-performance computing systems due to balancing delays of functional nodes of the pipeline.

ACKNOWLEDGMENTS

The work was performed within the framework of the State assignment of the Ministry of Science and Higher Education of the Russian Federation (theme No. FSFZ-2022-0004 “Architectures of specialized computing complexes, methods, algorithms, and tools for designing digital computing devices”).

Authors’ contribution. All authors equally contributed to the research work

REFERENCES

1. Hennessy J.L., Patterson D.A. A new golden age for computer architecture: Domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development. In: *Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture* (*ISCA*). IEEE; 2018. https://doi.org/10.1109/ISCA.2018.00011
2. Hennessy J.L, Patterson D.A. *Computer Architecture*: *A Quantitative Approach*. 6th ed. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann; 2017. 936 p.
3. Sesin I.Yu., Bolbakov R.G. Comparative analysis of software optimization methods in context of branch predication on GPUs. *Russ. Technol. J.* 2021;9(6):7–15 (in Russ.). https://doi.org/10.32362/2500-316X-2021-9-6-7-15
4. Sleptsov V.V., Afonin V.L., Ablaeva A.E., Dinh B. Development of an information measuring and control system for a quadrocopter. *Russ. Technol. J*. 2021;9(6):26–36 (in Russ.). https://doi.org/10.32362/2500-316X-2021-9-6-26-36
5. Smirnov A.V. Optimization of digital filters performances simultaneously in frequency and time domains. *Russ. Technol. J*. 2020;8(6):63–77 (in Russ.). https://doi.org/10.32362/2500-316X-2020-8-6-63-77
6. Umnyashkin S.V. *Osnovy teorii tsifrovoi obrabotki signalov* (*Fundamentals of the Theory of Digital Signal Processing*). 3rd ed. Moscow: Litres; 2022. 551 p. (in Russ.). ISBN 978-5-4576-1810-7
7. Abadi M., Barham P., Chen J., et al. TensorFlow: A system for Large-Scale Machine Learning. In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI ’16*). USENIX Association; 2016. P. 265–283.
8. Nurvitadhi E., Sheffield D., Sim J., et al. Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC. In: *2016 International Conference on Field-Programmable Technology* (*FPT*). IEEE; 2016. P. 77–84. https://doi.org/10.1109/FPT.2016.7929192
9. Sovetov P.N. Synthesis of linear programs for a stack machine. *Vysokoproizvoditel’nye vychislitel’nye sistemy i tekhnologii = High-Performance Computing Systems and Technologies*. 2019;3(1):17–22 (in Russ.).
10. Aho A.V., Lam M.S., Sethi R., Ullman J.D. *Kompilyatory: printsipy, tekhnologii i instrumentarii* (*Compilers: Principles, Techniques, & Tools*): transl. from Engl. Moscow: Vil’yams; 2018. 1184 p. ISBN 978-5-8459-1932-8 (in Russ.).
[Aho A.V., Lam M.S., Sethi R., Ullman J.D. *Compilers: Principles, Techniques, & Tools*. Pearson Addison Wesley; 2007. 1035 p.]

[4] https://www.xilinx.com/products/som/kria/k26c-commercial.html. Accessed October 10, 2023.

11. Pratt T.W., Zelkowitz M.V. *Yazyki programmirovaniya: razrabotka i realizatsiya* (*Programming Languages. Design and Implementation*): transl. from Engl. St. Petersburg: Piter; 2002. 688 p. (in Russ.).
[Pratt T.W., Zelkowitz M.V. *Programming Languages. Design and Implementation*. Prentice Hall; 2001. 649 p.]
12. Tarasov I.E., Potekhin D.S., Khrenov M.A., Sovetov P.N. Computer-aided design of multicore system for embedded applications. *Ekonomika i Menedzhment Sistem Upravleniya*. 2017;25(3–1):179–185 (in Russ.).
13. Huang S., Wu K., Jeong H., Wang C., Chen D., Hwu W.M. PyLog: An Algorithm-Centric Python-Based FPGA Programming and Synthesis Flow. *IEEE Trans. Comput*. 2021;70(12):2015–2028. https://doi.org/10.1109/TC.2021.3123465
14. Jiang S., Pan P., Ou Y., Batten C. PyMTL3: A Python Framework for Open-Source Hardware Modeling, Generation, Simulation, and Verification. *IEEE Micro*. 2020;40(4):58–66. https://doi.org/10.1109/MM.2020.2997638
15. Oishi R., Kadomoto J., Irie H., Sakai S. FPGA-based Garbling Accelerator with Parallel Pipeline Processing. *IEICE Transactions on Information and Systems*. 2023;E106-D(12):1988–1996. https://doi.org/10.1587/transinf.2023PAP0002

СПИСОК ЛИТЕРАТУРЫ

1. Hennessy J.L., Patterson D.A. A new golden age for computer architecture: Domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development. In: *Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture* (*ISCA*). IEEE; 2018. https://doi.org/10.1109/ISCA.2018.00011
2. Hennessy J.L, Patterson D.A. *Computer Architecture*: *A Quantitative Approach*. 6th ed. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann; 2017. 936 p.
3. Сесин И.Ю., Болбаков Р.Г. Сравнительный анализ методов оптимизации программного обеспечения для борьбы с предикацией ветвлений на графических процессорах. *Russian Technological Journal*. 2021;9(6):7–15. https://doi.org/10.32362/2500-316X-2021-9-6-7-15
4. Слепцов В.В., Афонин В.Л., Аблаева А.Е., Динь Б. Разработка информационно-измерительной и управляющей системы квадрокоптера. *Russian Technological Journal*. 2021;9(6):26–36. https://doi.org/10.32362/2500-316X-2021-9-6-26-36
5. Смирнов А.В. Оптимизация характеристик цифровых фильтров одновременно в частотной и временной областях. *Russian Technological Journal*. 2020;8(6):63–77. https://doi.org/10.32362/2500-316X-2020-8-6-63-77
6. Умняшкин С.В. *Основы теории цифровой обработки сигналов*. 6-е изд. М.: Litres; 2022. 551 с. ISBN 978-5-4576-1810-7
7. Abadi M., Barham P., Chen J., et al. TensorFlow: A system for Large-Scale Machine Learning. In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI '16*). USENIX Association; 2016. P. 265–283.
8. Nurvitadhi E., Sheffield D., Sim J., et al. Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC. In: *2016 International Conference on Field-Programmable Technology* (*FPT*). IEEE; 2016. P. 77–84. https://doi.org/10.1109/FPT.2016.7929192
9. Советов П.Н. Синтез линейных программ для стековой машины. *Высокопроизводительные вычислительные системы и технологии*. 2019;3(1):17–22.
10. Ахо А.В., Лам М.С., Сети Р., Ульман Д.Д. *Компиляторы: принципы, технологии и инструментарий*: пер. с англ. М.: Вильямс; 2018. ISBN 978-5-8459-1332-8
11. Пратт Т., Зелковиц М. *Языки программирования: разработка и реализация*: пер. с англ. СПб.: Питер; 2002. 688 с.
12. Тарасов И.Е., Потехин Д.С., Хренов М.А., Советов П.Н. Автоматизация проектирования многопроцессорной системы на базе ПЛИС для управления во встраиваемых приложениях. *Экономика и менеджмент систем управления*. 2017;25(3–1):179–185.
13. Huang S., Wu K., Jeong H., Wang C., Chen D., Hwu W.M. PyLog: An Algorithm-Centric Python-Based FPGA Programming and Synthesis Flow. *IEEE Trans. Comput*. 2021;70(12):2015–2028. https://doi.org/10.1109/TC.2021.3123465
14. Jiang S., Pan P., Ou Y., Batten C. PyMTL3: A Python Framework for Open-Source Hardware Modeling, Generation, Simulation, and Verification. *IEEE Micro*. 2020;40(4):58–66. https://doi.org/10.1109/MM.2020.2997638
15. Oishi R., Kadomoto J., Irie H., Sakai S. FPGA-based Garbling Accelerator with Parallel Pipeline Processing. *IEICE Transactions on Information and Systems*. 2023;E106-D(12):1988–1996. https://doi.org/10.1587/transinf.2023PAP0002

About the authors

Ilya E. Tarasov, Dr. Sci. (Eng.), Associated Professor, Head of the Laboratory of Specialized Computing Systems, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: tarasov_i@mirea.ru. Scopus Author ID 57213354150, RSCI SPIN-code 4628-7514, http://orcid.org/0000-0001-6456-4794

Peter N. Sovietov, Cand. Sci. (Eng.), Senior Researcher, Laboratory of Specialized Computing Systems, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: sovetov@mirea.ru. Scopus Author ID 57221375427, RSCI SPIN-code 9999-1460. http://orcid.org/0000-0002-1039-2429

Daniil V. Lulyava, Junior Researcher, Laboratory of Specialized Computing Systems, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: lyulyava@mirea.ru. Scopus Author ID 58811698000, RSCI SPIN-code 1882-0989, http://orcid.org/0009-0009-9623-7777

Dmitry I. Mirzoyan, Senior Researcher, Laboratory of Specialized Computing Systems, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: mirzoyan@mirea.ru. Scopus Author ID 57432027000, ResearcherID JJE-7844-2023, RSCI SPIN-code 8135-9802, http://orcid.org/0009-0002-4703-8340

Об авторах

Тарасов Илья Евгеньевич, д.т.н., доцент, заведующий лабораторией специализированных вычислительных систем, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: tarasov_i@mirea.ru. Scopus Author ID 57213354150, SPIN-код РИНЦ 4628-7514, http://orcid.org/0000-0001-6456-4794

Советов Петр Николаевич, к.т.н., старший научный сотрудник, лаборатория специализированных вычислительных систем, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: sovetov@mirea.ru. Scopus Author ID 57221375427, SPIN-код РИНЦ 9999-1460. http://orcid.org/0000-0002-1039-2429

Люлява Даниил Вячеславович, младший научный сотрудник, лаборатория специализированных вычислительных систем, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: lyulyava@mirea.ru. Scopus Author ID 58811698000, SPIN-код РИНЦ 1882-0989, http://orcid.org/0009-0009-9623-7777

Мирзоян Дмитрий Ильич, старший научный сотрудник, лаборатория специализированных вычислительных систем, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: mirzoyan@mirea.ru. Scopus Author ID 57432027000, ResearcherID JJE-7844-2023, SPIN-код РИНЦ 8135-9802, http://orcid.org/0009-0002-4703-8340

Translated from Russian into English by Lyudmila O. Bychkova
Edited for English language and spelling by Thomas A. Beavitt