Information systems. Computer sciences. Issues of information security

Информационные системы. Информатика. Проблемы информационной безопасности

UDC 004.2
https://doi.org/10.32362/2500-316X-2022-10-3-24-33

RESEARCH ARTICLE

Prospects for using soft processors in systems-on-a-chip based on field-programmable gate arrays

Ilya E. Tarasov @, **Dmitry S. Potekhin, Olga V. Platonova**

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

Abstract
Objectives. Developing the element base of field-programmable gate arrays (FPGA) may significantly affect the design of electronic devices due to the enhanced logical capacity of such chips and the general tendency towards increased subsystem integration. The system-on-a-chip (SoC) concept is aimed at combining receiving, processing, and exchange subsystems onto a single chip, as well as at implementing control, diagnostic, and other auxiliary subsystems. The study aimed at developing a method for soft processor applications, i.e., processors based on configurable logical resources, for implementing control functions in an FPGA-based SoC.
Methods. A digital system design methodology was used.
Results. For soft processors, a unified design route based on selecting architectural parameters qualitatively corresponding to control tasks was considered. In particular, such parameters as instruction set addressness, number of pipeline cycles, and arithmetic logic unit configuration are adjustable at the design stage to allow the optimization of the soft processor in the discrete parameter space. An approach to rapid prototyping of the assembler based on stack-oriented programming language with regular grammar was also considered. The control of digital signal processing hardware as part of an SoC is the promising application area for soft processors. An implementation is considered on the example of an SoC based on Xilinx Virtex-7 FPGA containing several processor cores developed using the proposed methodology.
Conclusions. The considered approaches to soft processor design allow the rapid prototyping of the control processor core for operation as part of an FPGA-based SoC.

Keywords: processor, field-programmable gate array, system-on-a-chip, digital signal processing, compiler

• **Submitted:** 10.01.2022 • **Revised:** 21.03.2022 • **Accepted:** 22.04.2022

For citation: Tarasov I.E., Potekhin D.S., Platonova O.V. Prospects for using soft processors in systems-on-a-chip based on field-programmable gate arrays. *Russ. Technol. J.* 2022;10(3):24–33. https://doi.org/10.32362/2500-316X-2022-10-3-24-33

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

Перспективы применения софт-процессоров в системах на кристалле на базе программируемых логических интегральных схем

И.Е. Тарасов @, Д.С. Потехин, О.В. Платонова

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

Резюме
Цели. Развитие элементной базы программируемых логических интегральных схем (ПЛИС) качественно меняет требования к маршруту проектирования электронных средств вследствие роста логической емкости этих микросхем и тенденции к повышению степени интеграции подсистем. Преимущественным направлением применения данной платформы является концепция системы на кристалле (СнК), направленная на совмещение в одном кристалле подсистем приема, обработки и обмена данными, а также реализацию управляющих, диагностических и других вспомогательных подсистем. Цель работы – разработка методики применения софт-процессоров, т.е. процессоров, создаваемых на базе конфигурируемых логических ресурсов, для реализации функций управления в составе СнК на базе ПЛИС.
Методы. Использованы методы проектирования цифровых систем.
Результаты. Для софт-процессоров рассмотрен унифицированный маршрут проектирования, основанный на выборе архитектурных параметров, качественно соответствующих задачам управления. В частности, такие параметры, как адресность системы команд, количество тактов конвейера, конфигурация арифметико-логического устройства, являются регулируемыми на этапе проектирования, что позволяет проводить оптимизацию софт-процессора в дискретном пространстве параметров. Рассмотрен также подход к быстрому прототипированию ассемблера на основе стекового языка программирования с регулярной грамматикой. Перспективным направлением применения софт-процессоров является управление аппаратными компонентами цифровой обработки сигналов в составе СнК. В статье рассмотрен пример реализации СнК на базе ПЛИС Xilinx Virtex-7, в составе которого применены несколько процессорных ядер, разработанных по предложенной методике.
Выводы. Рассмотренные подходы к проектированию софт-процессоров позволяют проводить быстрое прототипирование управляющего процессорного ядра для работы в составе СнК на базе ПЛИС.

Ключевые слова: процессор, ПЛИС, система на кристалле, цифровая обработка сигналов, компилятор

• **Поступила:** 10.01.2022 • **Доработана:** 21.03.2022 • **Принята к опубликованию:** 22.04.2022

Для цитирования: Тарасов И.Е., Потехин Д.С., Платонова О.В. Перспективы применения софт-процессоров в системах на кристалле на базе программируемых логических интегральных схем. *Russ. Technol. J.* 2022;10(3):24–33. https://doi.org/10.32362/2500-316X-2022-10-3-24-33

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

Авторы заявляют об отсутствии конфликта интересов.

INTRODUCTION

Disadvantages inherent to programmable logic cell arrays in digital system designs based on field-programmable gate arrays (FPGA) include inferior clock rate, chip area, and power consumption characteristics. However, such disadvantages may be compensated by combining hardware architectures to allow the reconfiguration of circuits, as well as the development of devices having optimized architectures for particular tasks [1]. Moreover, since FPGA functionality involves a number of digital nodes and subsystems, hardware architecture has evolved to progressively add specialized non-reconfigurable hardware components embedded in logic cell design. For example, dual-port static memory blocks and hardware multipliers of independent operands later converted into digital signal processing (DSP) modules were added to the FPGA architecture at the turn of 2000s [2]. Later on, the PowerPC (Xilinx Virtex-II Pro, Virtex-4, Virtex-5)[1] and Advanced RISC Machine (ARM) (Xilinx Zynq-7000, Xilinx Zynq UltraScale+, Xilinx Versal[2], Intel Cyclone V[3]) hardware processor cores were added to FPGAs. By separating the ARM processor subsystem into an independently operating part of the chip, the Xilinx Zynq-7000 family became a fully programmable system-on-a-chip (SoC)[4] rather than FPGA.

The practical use of fully programmable SoCs has demonstrated the increasingly auxiliary role of processors due to the significant lag in their performance as compared with the overall performance of the array of configurable logic elements supplemented with DSP components. However, attempts to build a system around a processor core with configurable peripheral devices resulted in economically inefficient solutions as compared to similar capabilities provided by a wide element base range of microcontrollers based on ARM, MIPS (microprocessor without interlocked pipeline stages), RISC-V (reduced instruction set computer) cores, etc. Thus, the key factor determining the competitive technical and economic performance of FPGAs is the demand for reconfigurable resources as the main element of a computing system.

Among the main areas of application for high-performance computing are video processing systems, virtual and augmented reality, robotics, industrial automation, digital radio communication, and measurement equipment systems [3, 4]. Current areas of signal processing include digital filtering [5], spectral analysis [6], and machine learning algorithms [7], including those based on specialized neuroprocessors [8] or FPGA-based reconfigurable accelerators.[5]

ARCHITECTURE OF DIGITAL SoC BASED ON PROCESSOR

The requirements for a processor used as part of an SoC are determined by its tasks and role in the system. Since the main computational load is provided by hardware accelerators implemented on the basis of logic cells, static memory blocks, and DSP hardware modules, it becomes hard to ensure the continuous participation of a processor (or even several processor cores) in dataflow computing. Therefore, hardware accelerators implemented in FPGA should provide dataflow computing that does not require constant processor participation, but instead allows the processing of parameters, reconfiguration, monitoring, and similar tasks, to be controlled by software. The implementation of complex software protocols for data exchange, such as support for wired and wireless networks, user interfaces, etc., also becomes a significant challenge. The flowchart illustrating interaction between the processor and hardware acceleration subsystem is shown in Fig. 1.

Figure 1 shows that the main computational performance is determined by a hardware accelerator implemented based on reconfigurable FPGA resources. At the same time, the control of low-speed peripherals may be assigned to the soft processor, while its low capacity allows the use of a second (and, if necessary, the third, fourth, etc.) processor core for simplifying software development. Here, a significant factor is the need to reduce response time to events generated by peripheral controllers.

Two significant tasks may arise in the development of control processors:

- providing an efficient topological implementation;
- tool support by software development tools.

The design route of an FPGA-based digital device can be divided into principal stages of synthesis and topological implementation. At the synthesis stage, the main part of circuit design is performed to convert the design source code into a hardware-independent netlist. The predicted clock rate may be significantly reduced by the subsequent placing of components and tracing of configurable connections. Achieving a high clock rate usually involves additional efforts by developers to clarify the mutual arrangement of the processor subsystems and even its individual digital nodes. Therefore, newly developed processors, even with implementing additional features, are initially disadvantaged compared to widespread processor cores optimized at the FPGA topology level.

The second significant factor hindering the active spread of using new soft processors optimized for the combined action with hardware accelerators is the requirement for tool support. Since additional instructions entered into the processor are not automatically supported by universal compilers, the basic soft processor model can additionally be defined by focusing on common retargetable compilers, such as GNU compiler collection (GCC) and low-level virtual machine (LLVM). This may drastically reduce the value

[1] Virtex-4 FPGA. User Guide. UG070 (v2.6). December 1, 2008. URL: https://www.xilinx.com/support/documentation/user_guides/ug070.pdf. Accessed December 27, 2021.

[2] Versal ACAP Technical Reference Manual. AM011 (v1.3). October 27, 2021. URL: https://www.xilinx.com/support/documentation/architecture-manuals/am011-versal-acap-trm.pdf. Accessed December 27, 2021.

[3] Cyclone® V FPGA и SoC FPGA. URL: https://www.intel.ru/content/www/ru/ru/products/details/fpga/cyclone/v.html. Accessed December 27, 2021.

[4] Xilinx Adaptive SoCs. URL: https://www.xilinx.com/products/silicon-devices/soc.html. Accessed December 27, 2021.

[5] Versal Architecture and Product Data Sheet: Overview DS950 (v1.0). October 2, 2018. Advance Product Specification. URL: https://www.xilinx.com/support/documentation/data_sheets/ds950-versal-overview.pdf. Accessed December 27, 2021.

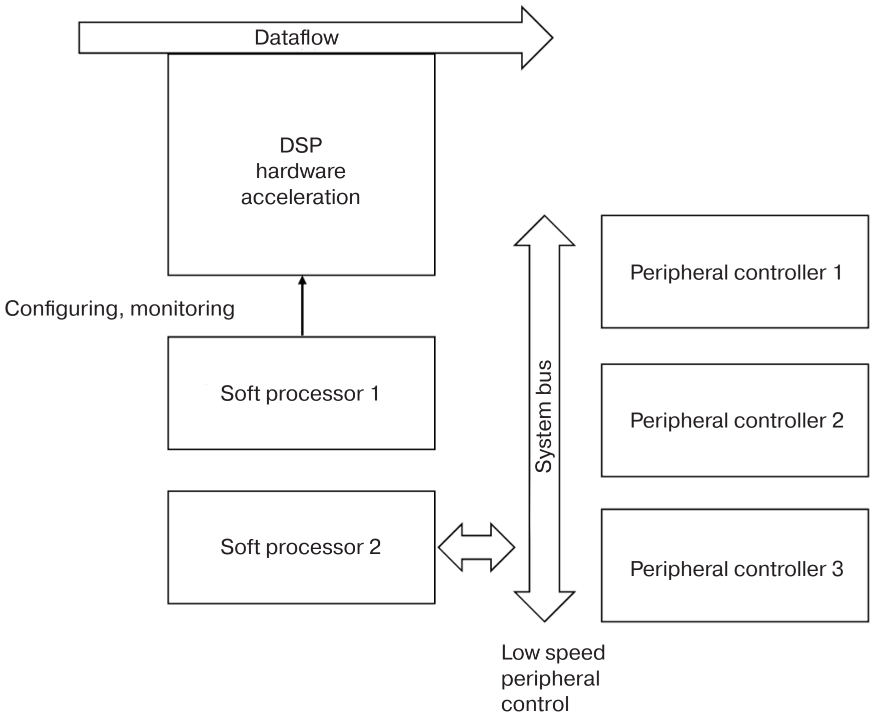

Fig. 1. Interaction between processor and hardware acceleration subsystem in SoC class device

of hardware modifications by allowing the efficient use of low-level programming.

SOFT PROCESSOR ARCHITECTURES

The reconfigurable nature of the programmable cell array defines a sufficient range of possibilities for implementing soft processors. On the other hand, using FPGA for processor layout intended for subsequent implementation in very large-scale integration (VLSI) should be distinguished from developing SoC components intended primarily for control, monitoring, and interface support functions. In this case, the soft processor added to the system should not overload it in terms of occupying logical resources or unnecessarily reduce the clock rate of the design due to excessive trace complexity.

The selection of soft processor architecture is strongly dependent on the hardware architecture. For example, ARM[6], MIPS[7], and RISC-V[8] solutions are currently widely used. Nevertheless, implementing processors based on FPGA configurable logic cells is only trivially less efficient as compared with a hardware solution. At the same time, the widespread MicroBlaze soft processor has been strongly influenced by the PowerPC and then ARM hardware core architecture. Here, a significant advantage of a soft processor has been software compatibility with corresponding hardware solutions present in the market. Therefore, for early FPGA families with PowerPC hardware cores, MicroBlaze cores were considered as a lower performance alternative suitable for low-cost FPGAs allowing the transfer of partially methodological developments and program code for PowerPC.

Thus, the implementation of widespread processor architectures based on logic cells aims at preserving methodological and tool support, but does not fully meet the requirements for the adaptation of the soft processor to the system architecture of the design. A possible solution to this problem may be the implementation of architecture features better adapted to FPGA, as well as to those designed for specific tasks.

It may be noted that the machine code density is the significant parameter for soft processors. This is due to the relatively low specific capacity of the FPGA on-chip static memory. In addition, since static memory blocks may be also used by hardware accelerators, memory saved for storing programs may be significant when selecting the processor architecture.

The general approach to instruction coding distinguishes between the two fundamental trends of strong and weak coding [9]. In strong coding, the binary representation of the instruction and its action is not accessible requiring transformation of machine code. In weak coding, conversely, some binary representation fields are directly responsible for particular processor devices encoding the operation, operands, and additional features.

[6] URL: https://www.arm.com/products/silicon-ip-cpu. Accessed December 27, 2021.
[7] URL: https://www.mips.com/products/architectures. Accessed December 27, 2021.
[8] URL: https://riscv.org/risc-v-foundation/16. Accessed December 27, 2021.

The generalized instruction format may be represented as follows:

<Dest> = <Operand1> op <Operand2>

where Dest is the destination device (register) for placing the operation result; Operand1 is the first operand; op is the type of operation; and Operand2 is the second operand.

For the instruction set, the notion of addressness reflecting the number of registers described in the instruction code is used. The representation of instructions having different addressness is shown in Fig. 2.

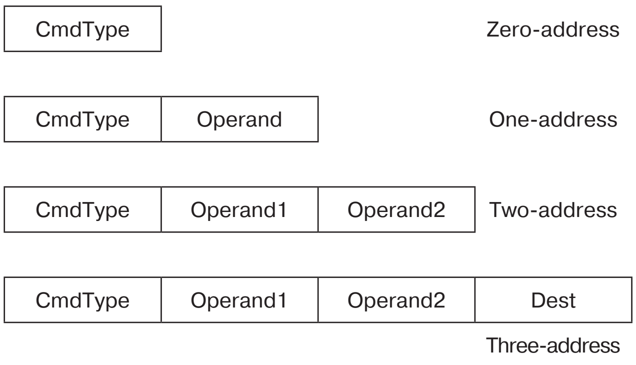

Fig. 2. Representation of instructions with different addressness

Figure 2 shows that, although an overall increase in the addressness results in increased tool software capabilities, the control word length is also increased. Therefore, the memory size required for storing the program also increases. The absence of a reference to a particular type of resource follows implicitly from the type of operation being performed or coincides with specified resources. For example, in the x86 processor instruction set, the destination register coincides with the first operand, while, in accumulator architectures, the accumulator is always the destination register and the first operand. In stack architectures, the operands are always positioned at the top of the data stack along with the result.

The instruction set architecture may be supplemented with hardware microarchitecture defining the interaction of the main circuitry components forming the processor core. While the instruction set architecture and microarchitecture are generally designed independently, the list of permitted operations is largely determined by the presence of hardware components and communication between them. Basic versions of the processor microarchitecture limited to the 5-cycle architecture are depicted in Fig. 3.

As well as supporting a number of instruction formats, raising the number of pipeline stages allows the clock rate to be increased. For example, only one data memory operation (read or write) may be performed in 3-cycle architecture. Here, the synchronous data memory interface conditions the outcome of the read instruction to be available at the "execution" stage only, while one additional cycle would be required for writing the read data to the destination register. However, 5-cycle architecture is free from this limitation.

While soft processors do not permit a further increase in the number of the pipeline clock cycles, the introduction of pipelining in an arithmetic logic unit (ALU) may reduce

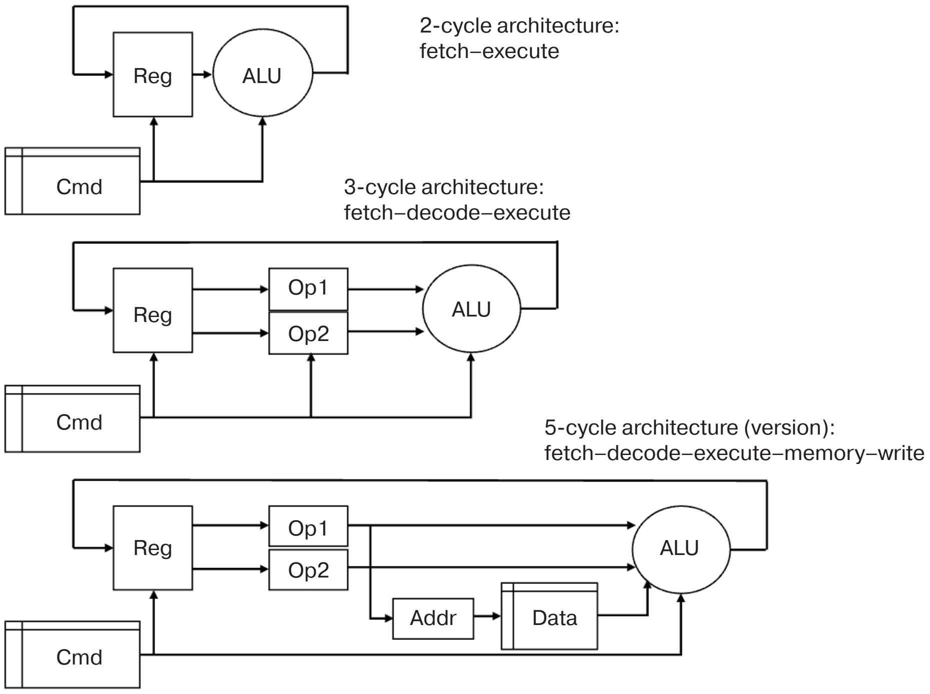

Fig. 3. Basic versions of the processor microarchitecture. ALU is the arithmetic logic unit

the critical path length in this module. Nevertheless, this approach may be less efficient for FPGAs where computational implementation is based on logic cells due to the larger granularity of cells compared with some of VLSI gates.

Considering the reconfigurable nature of FPGAs and the possibility of creating an instruction set optimized for a subclass of tasks, attention should be drawn to the possibilities of describing ALU. For example, in [10], a unified description of a computational node through four parameters (*I*, *O*, *D*, and *S*) is considered, where *I* is the number of instructions executed per clock cycle, *O* is the number of operations defined by instruction, *D* is the number of operands (pairs of operands) related to operations, and S is the degree of pipelining.

For SIMD (single instruction, multiple data) architectures, $D > 1$; for MIMD (multiple instruction, multiple data) architectures, $I > 1$, $D > 1$. Datapaths at the level of a separate computational node may be additionally considered for mass parallelism, thus allowing the architecture to be described according to the node in the form of four (*R*, <*O*>, <*D*>, and <*S*>), where: *R* is the number of independently calculated results, <*O*> is the vector of the number of operations performed by the instruction for each of the data paths, <*D*> is the vector of the number of operands (pairs of operands) for each of the datapaths, and <*S*> is the vector of pipelining for each of the datapaths.

This approach implies the replacement of parameters *O*, *D*, and *S* by vectors describing the characteristics of the corresponding datapaths forming R outputs.

The examples of ALU implementation are shown in Fig. 4. This description does not consider possible variations in the composition of computational nodes shown as ALU1, ALU2, and ALU3, although their functionality may differ both within a single path and for separate paths.

The datapath design route may be as follows:

1) selecting an architecture model in space (*R*, <*O*>, <*D*>, and <*S*>);
2) compiling pseudocode by the set of model tasks, identifying the most frequent sequences of operations along with creating computational nodes for their single-cycle execution, checking the result using emulator, and creating the processor element nodes at the register transfer level description;
3) checking the synthesizability of the obtained scheme and performing functional verification at the system level;
4) evaluating the topological implementation in the selected technological basis.

USE OF STACKED PROCESSORS IN FPGA-BASED DESIGNS

When providing tools for designing systems for new processor architectures, an important component consists in the design of compilers [11, 12]. Although so-called regular grammar imposes significant limitations on the input language, its low implementation complexity is of interest. A regular grammar may be conveniently supported by a stack-oriented computational model.

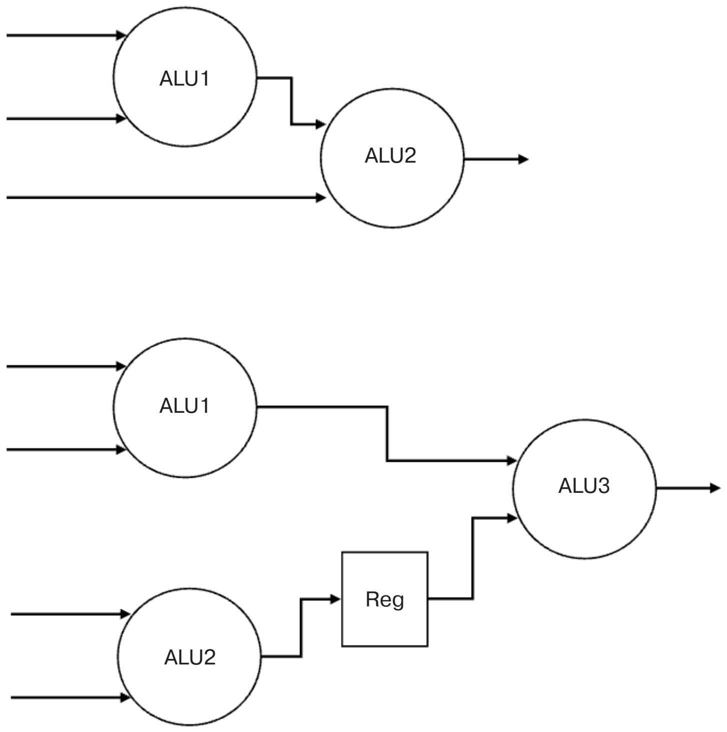

Fig. 4. Examples of the ALU architecture implementation based on unified representation

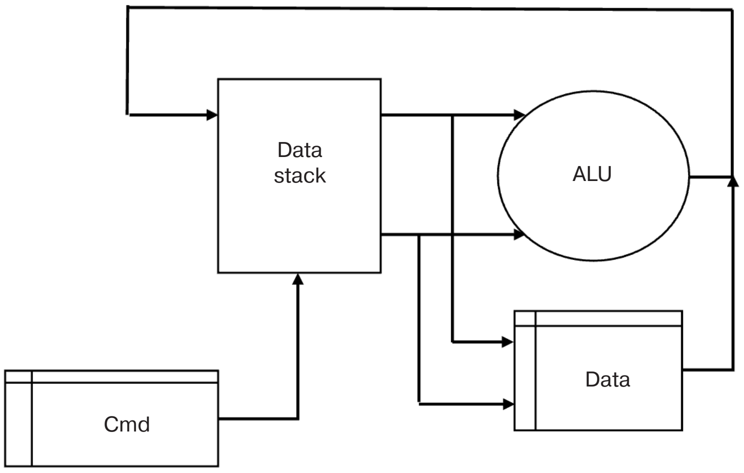

Fig. 5. Stacked processor microarchitecture

The stacked processor is an example of a zero-address architecture allowing the reduction of instruction size. This feature may be useful for FPGA-based systems in which on-chip static memory size is limited. Such a stacked processor microarchitecture is shown in Fig. 5.

Formalized approaches to program code generation for the stack-oriented computing model include those originally developed during the 1970s in the Forth programming language [13] and implemented, e.g., in the Java Virtual Machine, the intermediate code representation of .NET technology, and others. Although the Forth programming language itself is no longer widely used, the stack-oriented computing principles it embodies allow this approach to be used for low-level programming of stacked processor cores for efficient use as auxiliary processors in SoCs due to their compact code requirement and relatively low hardware costs. When combined with regular grammar, such a stack-oriented computing model supports rapid prototyping of tool software. This may be relevant for enabling the ALU rapid reconfiguration and the instruction set modification.

The external appearance of the developed 16-channel spectrum analyzer based on FPGA with multiprocessor control subsystem based on stacked processor cores is shown in Fig. 6.

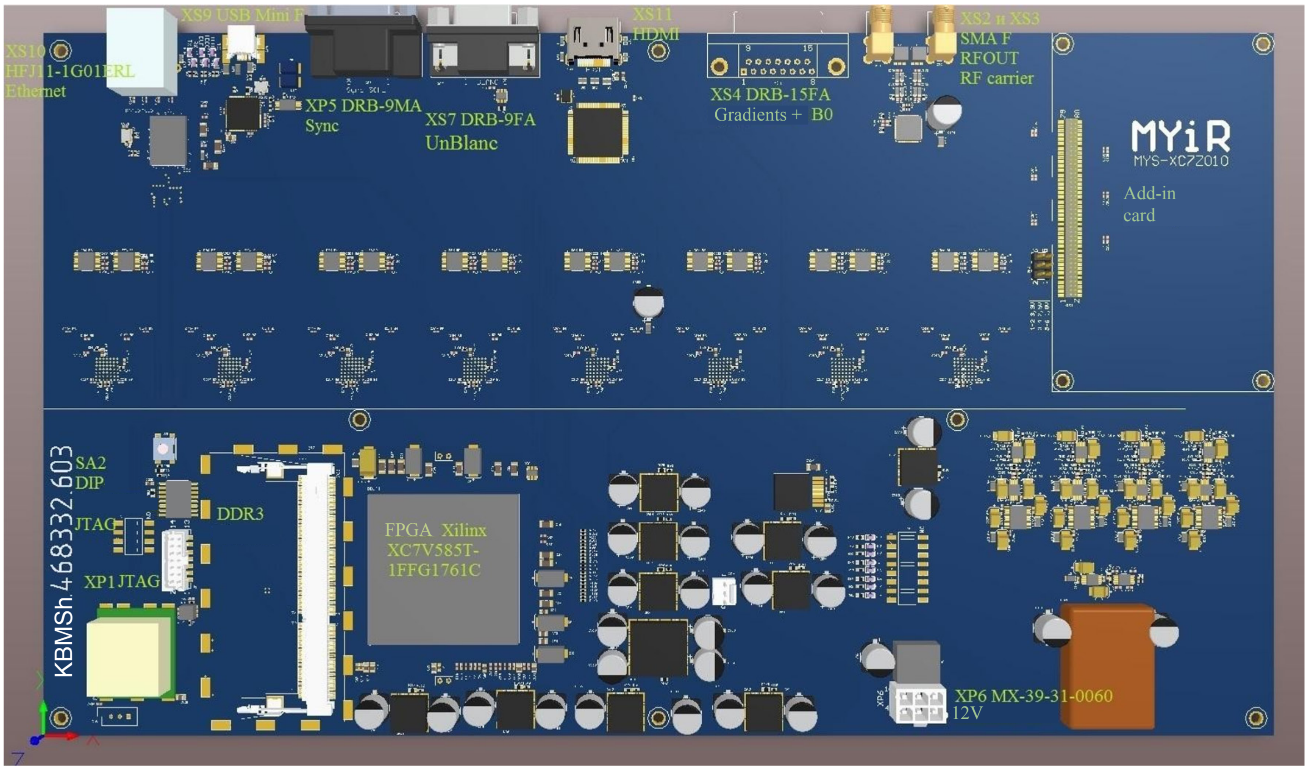

Fig. 6. External appearance of the developed 16-channel spectrum analyzer based on FPGA with multiprocessor control subsystem

Along with their positive impact on the system performance, the characteristics of stacked processor cores have been noted in the process of spectrum analyzer development and pilot operation. In particular, the selection of processor cores for auxiliary and communication tasks has allowed the complex programming requirement to be significantly simplified by focusing auxiliary cores on main computational processes without allocating resources of the main computational node for processing rarely occurring events. Combined with the large size used by FPGA (more than 500 000 logic cells), this confirms the feasibility of extensive use of soft processors for computational load distribution in complex computer systems.

CONCLUSIONS

In the paper, described approaches to the use of soft processors are aimed at accelerating the design of FPGA-based systems having medium and high logical capacity. The design methods allow the selection of microarchitecture versions and approaches to designing the processor instruction set intended mainly for supporting main hardware auxiliaries and performing auxiliary tasks as a part of an SoC. The use of stacked soft processors to reduce program memory size is relevant for solving auxiliary tasks associated with a deficit of on-chip FPGA memory.

Authors' contributions. 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*). Los Angeles, CA, USA, June 1–6, 2018. https://doi.org/10.1109/ISCA.2018.00011
2. Tarasov I.E. *PLIS Xilinx. Yazyki opisaniya apparatury VHDL i Verilog, SAPR, priemy proektirovaniya* (*FPGA Xilinx. Hardware description languages VHDL and Verilog, CAD, design techniques*). Moscow: Goryachaya liniya – Telekom; 2020. 538 p. (in Russ.). ISBN: 978-5-9912-0802-4
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*). Moscow: Tekhnosfera; 2017. 528 p. (in Russ.). ISBN 978-5-94836-424-7
7. Abadi M., 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*). 2016. P. 265–283. Available from URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
8. Nurvitadhi E., et al. Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC. In: *2016 International Conference on Field-Programmable Technology*. 2016. P. 77–84. https://doi.org/10.1109/fpt.2016.7929192

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*). Los Angeles, CA, USA, June 1–6, 2018. https://doi.org/10.1109/ISCA.2018.00011
2. Тарасов И.Е. *ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования.* М.: Горячая линия – Телеком; 2022. 538 с. ISBN 978-5-9912-0802-4
3. Сесин И.Ю., Болбаков Р.Г. Сравнительный анализ методов оптимизации программного обеспечения для борьбы с предикацией ветвлений на графических процессорах. *Russ. Technol. J.* 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. *Russ. Technol. J.* 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. *Russ. Technol. J.* 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*. М.: Техносфера; 2017. 528 с. ISBN 978-5-94836-424-7
7. Abadi M., 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*). 2016. P. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
8. Nurvitadhi E., et al. Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC. In: *2016 International Conference on Field-Programmable Technology*. 2016. P. 77–84. https://doi.org/10.1109/fpt.2016.7929192

9. Korneev V., Kiselev A. *Sovremennye mikroprotsessory* (*Modern microprocessors*). St. Petersburg: BHV-Petersburg; 2003. 448 p. (in Russ.).
10. Sima D., Fountain T., Kacsuk P. *Advanced Computer Architectures: A Design Space Approach*. Addison-Wesley; 1997. 790 p.
11. Akho A.V., Lam M.S., Seti R., Ul'man D.D. *Kompilyatory: printsipy, tekhnologii i instrumentarii* (*Compilers: Principles, Techniques, and Tools*). Transl. from Engl. Moscow: Vil'yams; 2017. 1184 p. (in Russ.). ISBN 978-5-8459-1932-8.
12. Pratt T., Zelkovits M. *Yazyki programmirovaniya: razrabotka i realizatsiya* (*Programming Languages: Design and Implementation*). Matrosov A. (Ed.). St. Petersburg: Piter; 2002. 688 p. (in Russ.). ISBN 5-318-00189-0
13. Baranov S.N., Nozdrunov N.R. *Yazyk Fort i ego realizatsii* (*The Forth Language and its Implementations*). Leningrad: Mashinostroenie; 1988. 157 p. (in Russ.). ISBN 5-217-00324-3
14. 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.).
15. 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.).

9. Корнеев В., Киселев А. *Современные микропроцессоры*. СПб.: БХВ-Петербург; 2003. 448 с.
10. Sima D., Fountain T., Kacsuk P. *Advanced Computer Architectures: A Design Space Approach*. Addison-Wesley; 1997. 790 p.
11. Ахо А.В., Лам М.С., Сети Р., Ульман Д.Д. *Компиляторы: принципы, технологии и инструментарий*: пер. с англ. М.: «И.Д. Вильямс»; 2017. 1184 с. ISBN 978-5-8459-1932-8.
12. Пратт Т., Зелковиц М. *Языки программирования: разработка и реализация*: под ред. А. Матросова. СПб.: Питер; 2002. 688 с. ISBN 5-318-00189-0
13. Баранов С.Н., Ноздрунов Н.Р. *Язык Форт и его реализации*. Л.: Машиностроение; 1988. 157 с. ISBN 5-217-00324-3
14. Советов П.Н. Синтез линейных программ для стековой машины. *Высокопроизводительные вычислительные системы и технологии*. 2019;3(1):17–22.
15. 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.

About the authors

Ilya E. Tarasov, Dr. Sci. (Eng.), Associated Professor, Professor, Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: tarasov_i@mirea.ru. Scopus Author ID 57213354150, http://orcid.org/0000-0001-6456-4794

Dmitry S. Potekhin, Dr. Sci. (Eng.), Associated Professor, Professor, Computer Technology Department, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: potehin@mirea.ru. Scopus Author ID 57213839310.

Olga V. Platonova, Cand. Sci. (Eng.), Associated Professor, Head of the Computer Technology Department, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: platonova@mirea.ru. Scopus Author ID 57222119478.

Об авторах

Тарасов Илья Евгеньевич, д.т.н., доцент, профессор кафедры корпоративных информационных систем Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: tarasov_i@mirea.ru. Scopus Author ID 57213354150, http://orcid.org/0000-0001-6456-4794

Потехин Дмитрий Станиславович, д.т.н., доцент, профессор кафедры вычислительной техники Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: potehin@mirea.ru. Scopus Author ID 57213839310.

Платонова Ольга Владимировна, к.т.н., доцент, заведующий кафедрой вычислительной техники Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: platonova@mirea.ru. Scopus Author ID 57222119478.

Translated by K. Nazarov
Edited for English language and spelling by Thomas Beavitt