Russian Technological Journal.
2022;10(3):24–33

ISSN 2782-3210 (Print)
ISSN 2500-316X (Online)

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

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 на 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.

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.

ВВЕДЕНИЕ

В проектах цифровых систем на базе программируемых логических интегральных схем (ПЛИС) важную роль играет комбинирование аппаратных архитектур для компенсации недостатков матрицы программируемых ячеек. Разработчикам устройств на базе ПЛИС хорошо известно, что по сравнению со специализированной элементной базой матрица конфигурируемых логических ячеек имеет худшие характеристики тактовой частоты, площади кристалла и энергопотребления. Это компенсируется возможностью неограниченного реконфигурирования схемы и создания устройства с архитектурой, оптимизированной для конкретной задачи [1]. Однако для ряда цифровых узлов и подсистем функциональность является ожидаемой и хорошо известной, поэтому в ходе эволюции ПЛИС их аппаратная архитектура претерпевала изменения, которые сводились к постепенному добавлению специализированных аппаратных компонентов, не подлежащих реконфигурированию, однако встраиваемых в проект, состоящий из логических ячеек. Например, в исходной архитектуре FPGA (field-programmable gate array) на рубеже 2000-х гг. к матрице логических ячеек были добавлены блоки статической двупортовой памяти и аппаратные умножители независимых операндов, позже преобразованные в модули

цифровой обработки сигналов [2]. В дальнейшем, с учетом широкого распространения процессорных систем, в ПЛИС были добавлены аппаратные ядра процессоров PowerPC (Xilinx Virtex-II Pro, Virtex-4, Virtex-5)[1] и Advanced RISC Machine (ARM) (Xilinx Zynq-7000, Xilinx Zynq UltraScale+, Xilinx Versal[2], Intel Cyclone V[3]). Выделение подсистемы процессоров ARM в независимо работающую часть кристалла позволило компании Xilinx заявить о классификации семейства Zynq-7000 не как о FPGA, а как о полностью программируемой системе на кристалле (СнК, англ. system-on-a-chip, SoC)[4].

Практическое применение полностью программируемой СнК показало, что процессоры играют в них преимущественно вспомогательную роль. Это объясняется существенным отставанием их производительности от суммарной производительности матрицы конфигурируемых логических элементов, дополненных компонентами цифровой обработки сигналов. Попытка построения системы вокруг процессорного ядра с настраиваемыми периферийными компонентами приводит к экономически неэффективному решению, поскольку аналогичные возможности обеспечиваются широким спектром элементной базы микроконтроллеров на базе ядер ARM, MIPS (microprocessor without interlocked pipeline stages), RISC-V (reduced instruction set computer) и др. Ключевым фактором, обуславливающим конкурентоспособные технико-экономические показатели ПЛИС, является востребованность реконфигурируемых ресурсов в качестве основного вычислительного элемента системы.

Важными направлениями для применения высокопроизводительных вычислительных комплексов являются системы обработки видео, виртуальная и дополненная реальность, робототехника, промышленная автоматика, цифровая радиосвязь, измерительная техника и ряд других [3, 4]. Среди реализуемых направлений обработки сигналов можно выделить цифровую фильтрацию [5], спектральный анализ [6], алгоритмы машинного обучения [7], в т.ч. на базе специализированных нейропроцессоров [8] или реконфигурируемых ускорителей на базе ПЛИС[5].

АРХИТЕКТУРА ЦИФРОВОЙ СнК НА БАЗЕ ПРОЦЕССОРА

Требования к процессору в составе СнК определяются его задачами и ролью в системе. Основная вычислительная нагрузка обеспечивается аппаратными ускорителями, реализуемыми на базе логических ячеек, блоков статической памяти и аппаратных модулей цифровой обработки сигналов. Большое количество таких устройств делает практически невозможным постоянное участие процессора (или даже нескольких процессорных ядер) в потоковой обработке данных. Поэтому реализуемые в ПЛИС аппаратные ускорители должны обеспечивать потоковую обработку данных, не требующую постоянного участия процессора, однако допускающую программное управление с целью изменения параметров обработки, реконфигурирования, мониторинга и подобных задач. Также важным является реализация сложных программных протоколов обмена данными, например, поддержка проводных и беспроводных сетей, интерфейса пользователя и др. Структурная схема, иллюстрирующая взаимодействие процессора и подсистемы аппаратного ускорения, приведена на рис. 1.

На рис. 1 видно, что основная производительность вычислений определяется аппаратным ускорителем, реализуемым на базе реконфигурируемых ресурсов ПЛИС. При этом управление низкоскоростными периферийными устройствами также может быть возложено на софт-процессор, причем его небольшой объем позволяет использовать второе (а при необходимости третье, четвертое и т.д.) ядро процессора для упрощения разработки программного обеспечения. Немаловажным фактором является уменьшение времени реакции на события, генерируемые периферийными контроллерами.

---

[1] Virtex-4 FPGA. User Guide. UG070 (v2.6). December 1, 2008. URL: https://www.xilinx.com/support/documentation/user_guides/ug070.pdf, дата обращения 27.12.2021. [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, дата обращения 27.12.2021. [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, дата обращения 27.12.2021. [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 (in Russ.).]

[4] Xilinx Adaptive SoCs. URL: https://www.xilinx.com/products/silicon-devices/soc.html, дата обращения 27.12.2021. [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, дата обращения 27.12.2021. [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.]

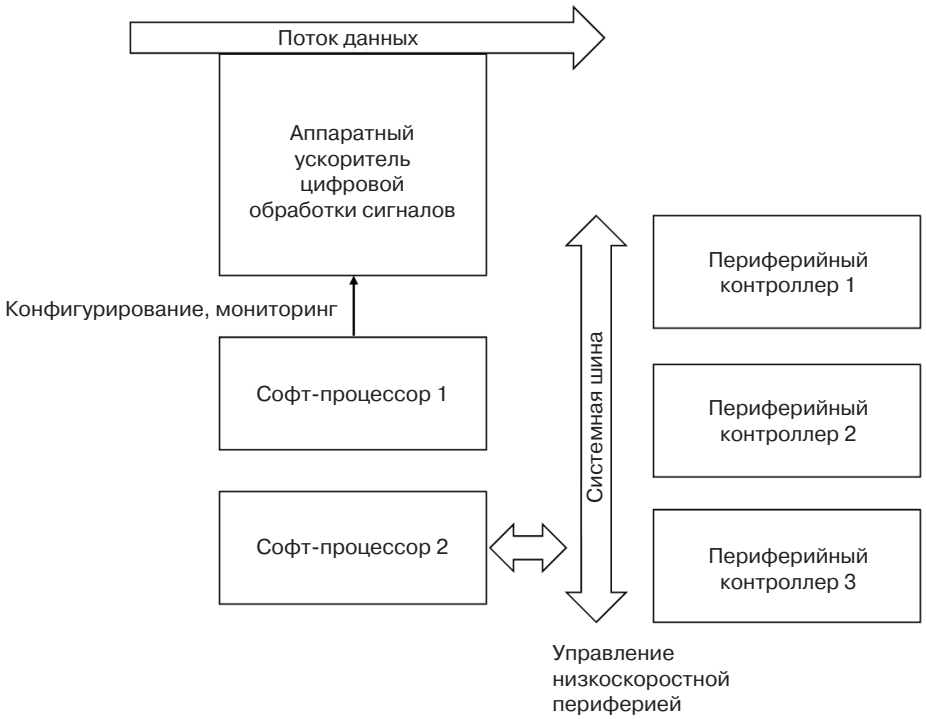

Рис. 1. Взаимодействие процессора и подсистемы аппаратного ускорения вычислений в устройстве класса СнК

При разработке управляющих процессоров возникают две важные задачи:

- обеспечение эффективной топологической реализации;
- инструментальная поддержка со стороны средств разработки программного обеспечения.

Маршрут проектирования цифрового устройства на базе ПЛИС разбит на крупные этапы синтеза (synthesis) и топологической реализации (implementation). Основная часть схемотехнического проектирования происходит на этапе синтеза, в результате которого исходные тексты проекта преобразуются в независимый от аппаратной платформы список связей (netlist). Последующий этап размещения компонентов и трассировки конфигурируемых соединений может заметно снизить прогнозируемую тактовую частоту. Достижение высокой тактовой частоты обычно сопряжено с дополнительными усилиями разработчиков по уточнению взаимного расположения подсистем процессора и даже его отдельных цифровых узлов. Поэтому вновь разрабатываемые процессоры, даже при условии реализации дополнительных возможностей, изначально находятся в невыгодных условиях по сравнению с широко распространенными процессорными ядрами, для которых была реализована оптимизация на уровне топологии ПЛИС.

Требование инструментальной поддержки является вторым важным фактором, тормозящим активное распространение подхода использования новых софт-процессоров, оптимизируемых для совместной работы с аппаратными ускорителями. Ориентация на распространенные перенацеливаемые (retargetable) компиляторы, такие как GCC (GNU compiler collection) и LLVM (low level virtual machine), по сути, обуславливает и базовую программную модель процессора, поскольку дополнительные команды, вводимые в процессор, не получают автоматическую поддержку универсальных компиляторов. Это резко снижает ценность аппаратных модификаций, поскольку их эффективное использование становится возможным при условии использования низкоуровневого программирования.

АРХИТЕКТУРЫ СОФТ-ПРОЦЕССОРОВ

Реконфигурируемый характер матрицы программируемых ячеек определяет достаточно широкие возможности реализации софт-процессоров. С другой стороны, следует различать применение ПЛИС для макетирования процессора, предназначенного для последующей реализации в СБИС, и разработку компонента СнК, предназначенного для выполнения преимущественно функций управления, мониторинга и поддержки интерфейсов. В этом случае добавляемый в систему софт-процессор не должен перегружать ее по объему занимаемых логических ресурсов или неоправданно снижать тактовую частоту проекта из-за чрезмерной сложности трассировки.

При выборе архитектуры софт-процессора наблюдается сильное влияние архитектур аппаратно реализованных устройств. Например, такие решения, как ARM[6], MIPS[7], RISC-V[8] широко представлены в проектах, однако необходимо обратить внимание, что реализация таких процессоров на базе конфигурируемых логических ячеек FPGA заведомо менее эффективна по сравнению с аппаратным решением. В то же время на широко распространенный софт-процессор MicroBlaze оказала сильное влияние архитектура аппаратного ядра PowerPC, а позже – ARM. Одной из важных причин этого является то, что в период присутствия на рынке соответствующих аппаратных решений важным преимуществом софт-процессора была программная совместимость с ними. Поэтому для ранних семейств ПЛИС с аппаратными ядрами PowerPC ядра MicroBlaze рассматривались как менее производительная альтернатива, пригодная к применению в недорогих ПЛИС, однако позволяющая частично перенести методические наработки и программный код для PowerPC.

Таким образом, реализация на базе логических ячеек широко распространенных процессорных архитектур имеет целью сохранение инструментального и методического обеспечения, однако не в полной мере отвечает требованиям адаптации софт-процессора к системной архитектуре проекта. Возможным решением могла бы стать реализация архитектуры, в большей степени приспособленной к особенностям ПЛИС FPGA, а также к специфике решаемых задач.

Можно отметить, что для софт-процессоров важным параметром является плотность машинного кода. Это обусловлено относительно невысоким удельным весом статической памяти на кристалле ПЛИС. Кроме того, блоки статической памяти могут быть использованы и аппаратными ускорителями, поэтому экономия памяти для хранения программ является важной при выборе архитектуры процессора.

Общий подход к кодированию команд предполагает два фундаментальных направления – сильное и слабое кодирование [9]. При сильном кодировании связь между двоичным представлением команды и ее действием неочевидна, и машинный код требует преобразования. При слабом кодировании отдельные поля двоичного представления напрямую отвечают за те или иные устройства процессора, кодируя операцию, операнды и дополнительные признаки.

Обобщенный формат команды можно представить следующим образом:

<Dest> = <Operand1> op <Operand2>,

где dest – устройство (регистр) назначения для помещения результата операции;

Operand1 – первый операнд; op – тип операции; Operand2 – второй операнд.

Для системы команд используется понятие адресности, отражающее количество регистров, описываемых в коде команды. На рис. 2 показано представление команды с различной адресностью.

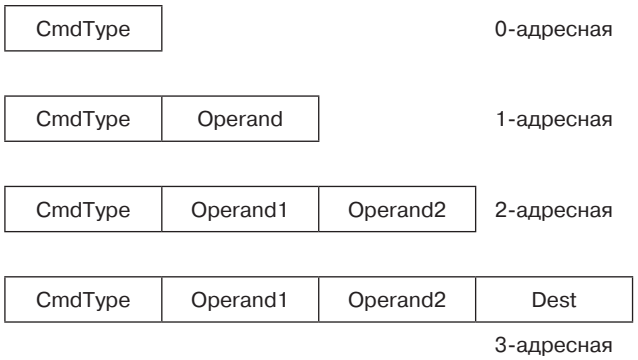

Рис. 2. Представление команд с различной адресностью

На рис. 2 можно видеть, что увеличение адресности в целом увеличивает возможности инструментального программного обеспечения, однако также увеличивает и разрядность командного слова. Следовательно, увеличивается и объем памяти, требуемой для хранения программы. При этом отсутствие указания на тот или иной тип ресурса означает, что он неявным образом следует из типа выполняемой операции или совпадает с указанными ресурсами. Например, в системе команд процессоров x86 регистр назначения совпадает с первым операндом, в аккумуляторных архитектурах регистром назначения и первым операндом всегда является аккумулятор. В стековой архитектуре операнды всегда расположены на вершине стека данных, и туда же помещается результат.

Архитектура системы команд дополняется аппаратной микроархитектурой, определяющей взаимодействие основных схемотехнических узлов, образующих ядро процессора. В целом, архитектура системы команд и микроархитектура проектируются независимо, однако наличие аппаратных компонентов и связи между ними во многом определяют перечень допустимых операций. На рис. 3 показаны основные варианты микроархитектуры процессора, ограниченные 5-тактной архитектурой.

[6] URL: https://www.arm.com/products/silicon-ip-cpu, дата обращения 27.12.2021. [URL: https://www.arm.com/products/silicon-ip-cpu. Accessed December 27, 2021.]

[7] URL: https://www.mips.com/products/architectures, дата обращения 27.12.2021. [URL: https://www.mips.com/products/architectures. Accessed December 27, 2021.]

[8] URL: https://riscv.org/risc-v-foundation/16, дата обращения 27.12.2021. [URL: https://riscv.org/risc-v-foundation/16. Accessed December 27, 2021.]

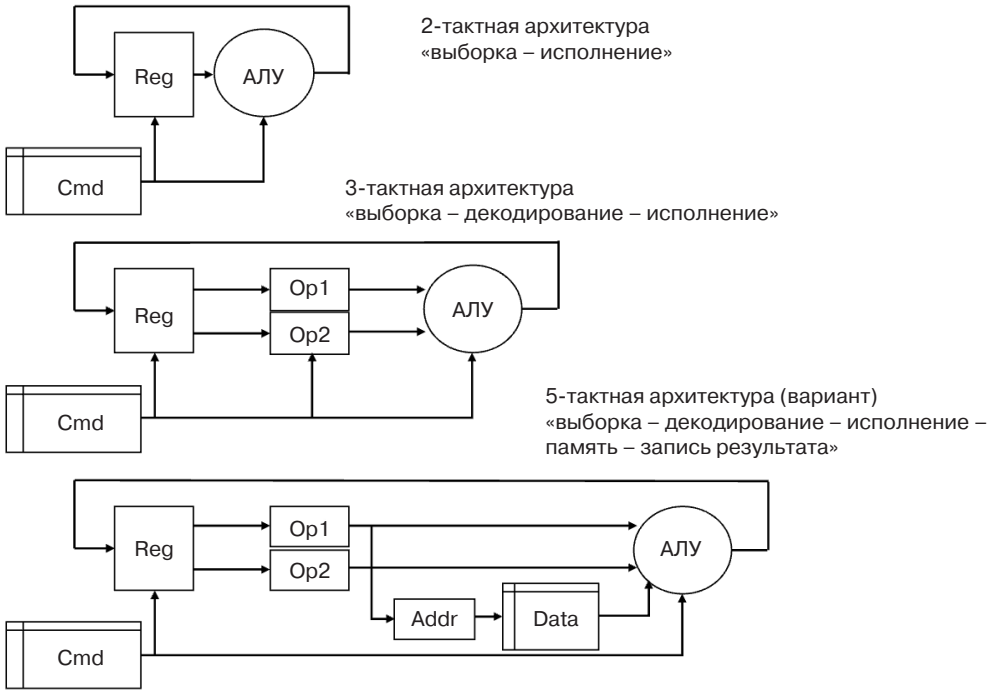

Рис. 3. Основные варианты микроархитектуры процессора. АЛУ – арифметико-логическое устройство

Увеличение количества стадий конвейера позволяет повышать тактовую частоту, а также делает возможной поддержку ряда форматов команд. Например, в 3-тактной архитектуре возможна только одна операция с памятью данных (чтение или запись). При этом синхронный интерфейс памяти данных обуславливает то, что результат чтения будет доступен только на стадии «исполнение», и потребуется еще один такт для записи прочитанных данных в регистр назначения. От этого ограничения свободна 5-тактная архитектура.

Дальнейшее увеличение числа тактов конвейера представляется нецелесообразным для софт-процессоров. Введение конвейеризации в арифметико-логическое устройство (АЛУ) позволяет уменьшить длину критического пути в этом модуле, однако для ПЛИС, где реализация вычислений производится на базе логических ячеек, такой подход менее эффективен из-за более крупной гранулярности ячеек по сравнению с отдельными вентилями СБИС.

С учетом реконфигурируемого характера ПЛИС и возможности построения системы команд, оптимизированной для подкласса задач, следует обратить внимание на возможности описания АЛУ. Например, в [10] рассмотрено унифицированное описание вычислительного узла четверкой параметров (I, O, D, S), где

I – количество инструкций, выполняемых за один такт;

O – количество операций, определяемое инструкцией;

D – количество операндов (пар операндов), относящихся к операциям;

S – степень конвейеризации.

Для SIMD-архитектур (англ. single instruction, multiple data — одиночный поток команд, множественный поток данных) $D > 1$, а для MIMD (англ. multiple instruction, multiple data – множественный поток команд, множественный поток данных) также $I > 1$, $D > 1$. Для массового параллелизма можно дополнительно рассматривать параллелизм трактов обработки данных (datapath) на уровне отдельного вычислительного узла, что позволяет применить архитектурное описание узла в виде четверки (R, $<O>$, $<D>$, $<S>$), где:

R – количество независимо вычисляемых результатов;

$<O>$ – вектор количества операций, выполняемых инструкцией для каждого из трактов данных;

$<D>$ – вектор количества операндов (пар операндов) для каждого из трактов данных;

$<S>$ – вектор конвейеризации для каждого из трактов данных.

Данный подход подразумевает замену параметров O, D, S на векторы, описывающие характеристики соответствующих путей обработки данных, которые образуют R выходов.

На рис. 4 показаны примеры реализации АЛУ. Такое описание не рассматривает возможные вариации состава вычислительных узлов, показанных как АЛУ1, АЛУ2, АЛУ3, хотя их функциональные возможности могут отличаться как внутри одного тракта, так и для отдельных трактов.

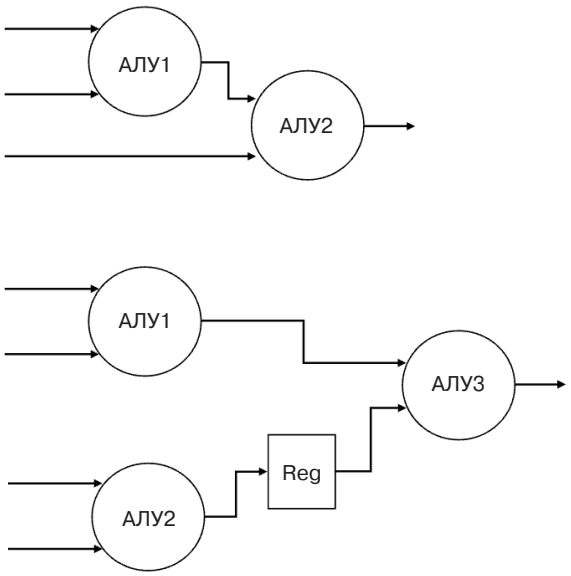

Рис. 4. Примеры реализации архитектуры АЛУ на основе унифицированного представления

Предлагаемый маршрут проектирования тракта данных выглядит следующим образом:

1) выбор модели архитектуры в пространстве (R, $<O>$, $<D>$, $<S>$);
2) компиляция псевдокода по набору модельных задач, выявление наиболее часто встречаемых последовательностей операций и создание вычислительных узлов для их однотактного исполнения, проверка результата с помощью эмулятора и создание узлов процессорного элемента на уровне регистровых передач (register transfer level, т.е. RTL-описания);
3) проверка синтезируемости полученной схемы и выполнение функциональной верификации на системном уровне;
4) выполнение оценки топологической реализации в выбранном технологическом базисе.

ПРИМЕНЕНИЕ СТЕКОВЫХ ПРОЦЕССОРОВ В ПРОЕКТАХ НА БАЗЕ ПЛИС

Разработка компиляторов для новых процессорных архитектур является важной составляющей в обеспечении инструментальных средств проектирования систем на их базе. В [11, 12] активно освещается тематика построения компиляторов, которая представляет собой обширную область. В этой связи представляет интерес т.н. регулярная грамматика, которая накладывает существенные ограничения на используемый входной язык, однако имеет небольшую сложность реализации. Регулярную грамматику удобно поддержать с помощью стековой вычислительной модели.

Стековый процессор представляет собой пример 0-адресной архитектуры, которая способствует сокращению объема программы. Это свойство полезно для

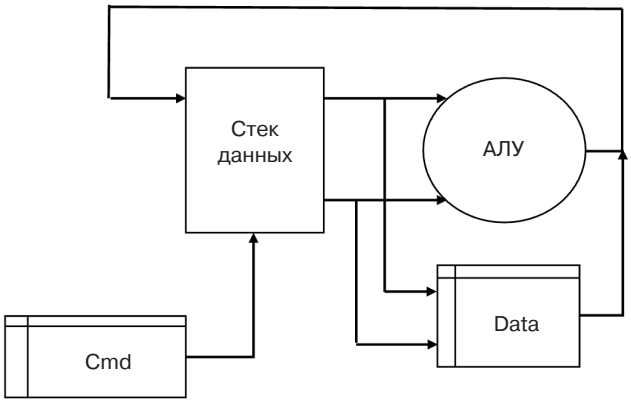

Рис. 5. Микроархитектура стекового процессора

систем на базе ПЛИС, в которых объем накристальной статической памяти ограничен. Микроархитектура стекового процессора показана на рис. 5.

Для стековой вычислительной модели существуют формализованные подходы к генерации программного кода, в частности, использованные еще в 1970-х гг. в стековом языке программирования Forth [13] и реализованные, например, в виртуальных машинах Java Virtual Machine (JVM), промежуточном представлении кода технологии .net и ряде других. В настоящее время сам по себе язык Forth используется крайне ограниченно, но заложенные в него принципы стековых вычислений позволяют использовать подобный подход для организации низкоуровневого программирования стековых процессорных ядер, которые ввиду компактного кода и относительно небольших аппаратных затрат могут эффективно использоваться в качестве вспомогательных процессоров в составе СнК. Стековая вычислительная модель в сочетании с регулярной грамматикой позволяют проводить быстрое прототипирование инструментального программного обеспечения. Это актуально для обеспечения возможности быстрого перестроения АЛУ и модификации системы команд.

На рис. 6 показан внешний вид разработанного 16-канального анализатора спектра на базе ПЛИС с многопроцессорной управляющей подсистемой на базе стековых процессорных ядер.

В процессе разработки и опытной эксплуатация анализатора спектра были подтверждены характеристики стековых процессорных ядер и их положительное влияние на системные характеристики. В частности, выделение процессорных ядер для решения вспомогательных и коммуникационных задач позволило существенно упростить программирование комплекса, поскольку дополнительные ядра позволили сосредоточиться на основных вычислительных процессах, не выделяя ресурсы основного вычислительного узла для обработки редко возникающих событий. В сочетании с большим объемом используемой ПЛИС (более 500 тыс. логических ячеек) это

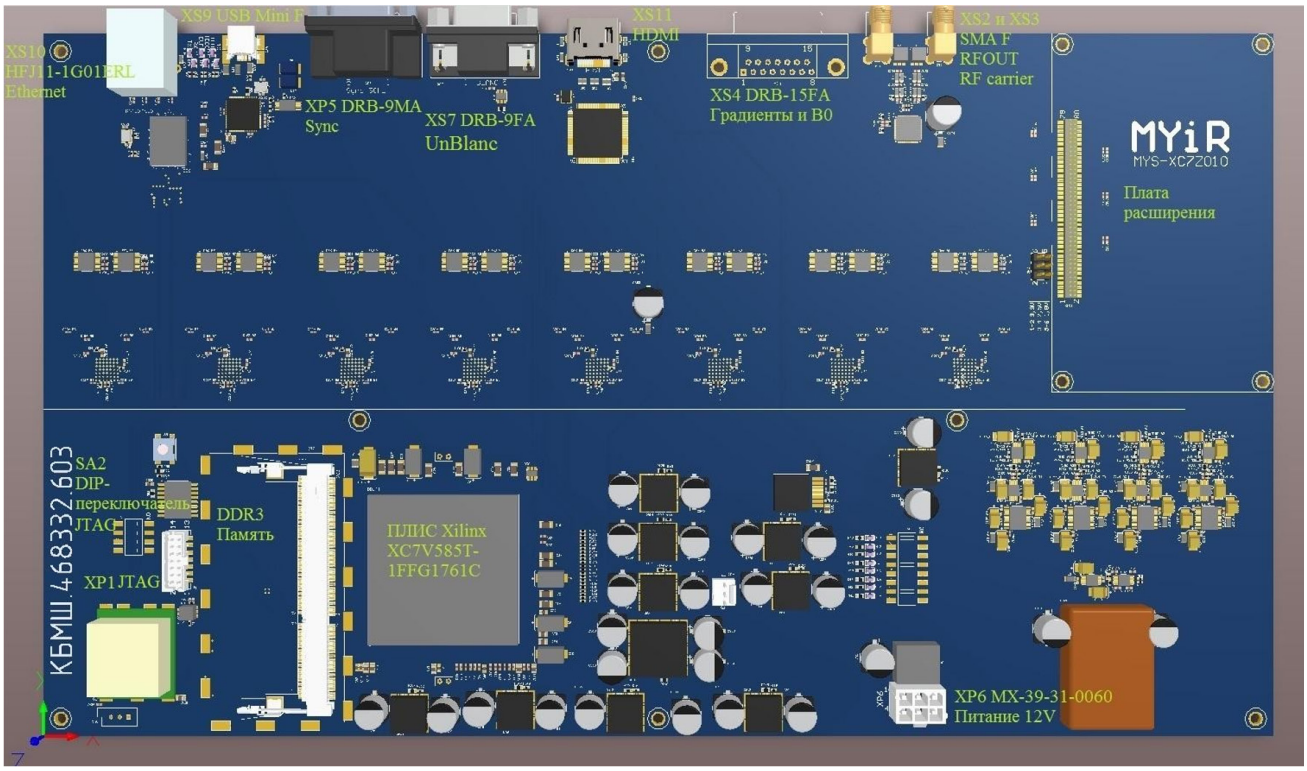

Рис. 6. Внешний вид 16-канального анализатора спектра
на базе ПЛИС с многопроцессорной управляющей подсистемой

подтвердило целесообразность активного применения софт-процессоров для распределения вычислительной нагрузки в сложных вычислительных комплексах.

ЗАКЛЮЧЕНИЕ

Рассмотренные в статье подходы к применению софт-процессоров направлены на ускорение проектирования систем на базе ПЛИС среднего и большого логического объема. Предложенные способы проектирования позволяют выбирать как варианты микроархитектуры, так и подходы к формированию системы команд процессора, основным назначением которого становится поддержка основных аппаратных ускорителей и выполнение вспомогательных задач в составе «системы на кристалле». Отдельный интерес представляют стековые софт-процессоры, позволяющие уменьшить размер памяти программ, что актуально для решения вспомогательных задач в условиях дефицита накристальной памяти ПЛИС.

Вклад авторов. Все авторы в равной степени внесли свой вклад в исследовательскую работу.

Authors' contributions. All authors equally contributed to the research work.

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

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. 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. 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.

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
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.).

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.

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.