Russian Technological Journal.
2024;12(3):37–45

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

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

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.

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.

ВВЕДЕНИЕ

Ситуация в области проектирования элементной базы для высокопроизводительных вычислений определяется рядом тенденций, соответствующих как объективным техническим ограничениям, так и необходимости интенсификации процессов импортозамещения и обеспечения технологического суверенитета. В связи с этим, анализируя желаемые технические характеристики, необходимо учитывать также возможности их достижения с учетом ограничений возможности производства и необходимости уменьшения технических и экономических рисков.

Анализ архитектурных тенденций в области вычислительных средств представлен Паттерсоном и Хеннесси в работе [1]. Авторы обращают внимание на ряд крупных этапов в области развития процессорных архитектур, начиная с 1970-х гг. Первым из отмеченных архитектурных переходов стала смена концепции CISC (complex instruction set computer) на RISC (reduced instruction set computer). Уменьшение сложности комбинационной логики, ставшее следствием такого перехода, позволило повысить тактовую частоту процессорных устройств.

Дальнейшее повышение тактовой частоты оказалось ограниченным после прекращения действия закона Деннарда, объясняющего увеличение производительности процессоров масштабированием размеров транзистора при переходе к следующему поколению технологических процессов. Реакцией на этот эффект стал переход к многоядерным процессорам, состоявшийся в массовом сегменте персональных компьютеров в середине 2000-х гг.

В свою очередь, повышение производительности за счет увеличения количества процессорных ядер ограничено законом Амдала, который определяет потенциальное повышение производительности многопроцессорного комплекса через долю вычислений, которые могут быть выполнены параллельно. Связанной проблемой является так называемая «стена интерфейсов» [2], учитывающая тот факт, что с уменьшением нормы технологического процесса производительность растет квадратично, а пропускная способность интерфейсов памяти и периферийных устройств – линейно. Поэтому построение многоядерных систем влечет за собой проблему организации межпроцессорного обмена данными, которая не может быть эффективно реализована из-за опережающего роста объема получаемых данных по сравнению с возможностью их передачи по существующим каналам связи, проектируемым по сопоставимым технологическим нормам.

В связи с изложенными проблемами в [1] предлагается переход к проблемно-ориентированным архитектурам (DSA, domain-specific architecture) по аналогии с проблемно-ориентированными языками программирования (DSL, domain-specific language). При этом специализация процессора для выполнения определенных классов вычислений, по сути, означает снижение его эффективности в других классах, что требует выбора для специализации таких целевых вычислительных задач, которые соответствовали бы актуальным техническим потребностям, допускали бы широкое применение из соображений снижения удельной стоимости проектирования и подготовки производства, а кроме того, соответствовали бы технически реализуемым подходам к проектированию цифровых устройств.

АНАЛИЗ ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ ДЛЯ РЕАЛИЗАЦИИ В СОСТАВЕ СПЕЦИАЛИЗИРОВАННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

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 программируемых логических интегральных схем (ПЛИС).

К вычислительным задачам, которые могут быть ускорены специализированными вычислительными системами, можно отнести следующие подклассы:

Таблица. Интенсивность использования операций, характерных для ряда задач, требующих применения высокопроизводительных вычислительных систем

Вид задачи	Сдвиги, сложение, поразрядные операции	Умножение	Операции с плавающей точкой	Трансцендентные функции	Операции с памятью
Хеш-функции	Массово	Нет	Нет	Нет	Возможно
Реализация нейросетей	Нет	Массово	Возможно	Возможно	Возможно
Цифровая обработка сигналов (фильтрация)	Массово	Массово	Возможно	Возможно	Возможно
Системы дифференциальных уравнений	Нет	Возможно	Массово	Возможно	Массово
Обработка трехмерной графики	Нет	Часто	Массово	Часто	Массово

1. Решение систем дифференциальных уравнений численными методами.
2. Операции с трехмерными изображениями.
3. Цифровая обработка сигналов на основе массового применения операций «умножение с накоплением (multiply and accumulate)».
4. Вычисление хеш-функций в задачах защиты информации.
5. Реализация нейросетей в части вычисления значений функций нейронов (neural net inference), не включая задачи обучения нейросети.

Виды операций, характерных для указанных задач, приведены в таблице. При этом столбцы таблицы размещены так, что сложность реализации соответствующих видов операций возрастает слева направо. Операции с памятью обозначены как имеющие наибольшую сложность вследствие того, что увеличение пропускной способности подсистемы памяти сопряжено с существенными затруднениями, хотя отдельные операции с памятью сами по себе высокой сложностью не обладают.

В таблице использованы следующие оценки, характеризующие интенсивность использования тех или иных видов операций. Оценка «нет» соответствует ситуации, когда операция не используется в алгоритмах и не требует поддержки. Оценка «возможно» характеризует ситуацию, когда такие действия имеют место, но в силу редкого использования не оказывают заметного влияния на эффективность вычислительного устройства. Для таких операций возможно применение неоптимизированных решений или готовых компонентов с функциональной избыточностью. Оценка «массово» соответствует операциям, которые являются основой алгоритмов и в наибольшей степени определяют эффективность вычислительного устройства, предназначенного для их реализации.

По предварительным оценкам видно, что реализация хеш-функций и операций цифровой обработки сигналов позволяет в большей степени продемонстрировать преимущества конвейерных архитектур, поскольку предусматривает потоковую обработку данных без интенсивного обмена с внешней памятью.

АРХИТЕКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ УЗЛОВ

Архитектурные подходы к реализации отдельных вычислительных узлов в современной цифровой электронике достаточно разнообразны. Для их эффективного практического использования следует ограничиться набором возможных решений, допускающих применение методов автоматизированного проектирования на уровне математических и программных моделей с тем, чтобы дальнейшие преобразования в представление схемы уровня регистровых передач (RTL-представление[1]) не вносили существенных изменений в характеристики такой системы. Можно отметить, например, что применение средств высокоуровневого описания класса HLL (high-level language) предполагает автоматизированное построение управляющих схем (flow control), рассчитанных на широкий класс реализуемых архитектурных подходов. Это ведет к излишнему усложнению синтезируемых управляющих схем.

Для практически реализуемой методики проектирования рассмотрены следующие архитектурные подходы к построению вычислительных узлов:

1. Процессорный узел.
2. Синхронный конвейер.
3. Модификация синхронного конвейера с возможностью повторного использования отдельных стадий.

Структурные схемы основных вычислительных узлов показаны на рис. 1.

[1] RTL – register transfer level.

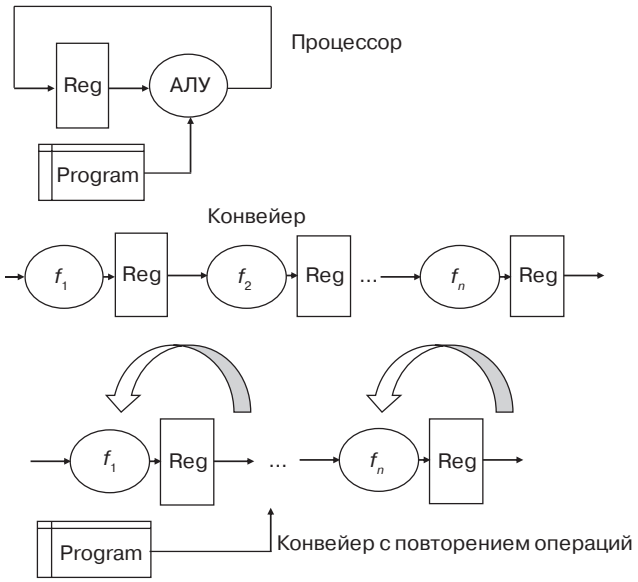

Рис. 1. Схемы основных вычислительных узлов (архитектурные шаблоны). Reg – регистр, Program – программа, f_1, f_2, ..., f_n – функциональные устройства

Приведенные варианты узлов рассматриваются в качестве архитектурных шаблонов для реализации выбранных подклассов вычислений. При этом для процессора требуемый набор операций реализуется в составе арифметико-логического устройства (АЛУ), а для конвейера – в последовательных стадиях. Для использования конвейера необходимо, чтобы порядок действий для реализации алгоритма оставался неизменным, иначе корректировка потребует усложнения управляющих схем. Повторение операций может быть использовано в таких алгоритмах, как вычисление хеш-функций и реализация фильтров с конечной импульсной характеристикой, при условии, что частота получения входных данных существенно меньше, чем тактовая частота конвейера. Выполнение этого условия позволит использовать одну и ту же стадию конвейера многократно до прихода нового входного значения.

Выбор конкретного архитектурного шаблона определяет операции по синтезу RTL-представления и его оптимизации. В данной статье рассмотрено применение методики для конвейерных вычислительных структур. Добавление архитектурных шаблонов процессора и конвейера с повторением операций предусмотрено для последующих этапов проекта.

Маршрут проектирования согласно предлагаемой методике проектирования вычислительных модулей специализированной вычислительной системы представлен на рис. 2.

В представленном маршруте видно, что входными данными являются исходные тексты реализуемого

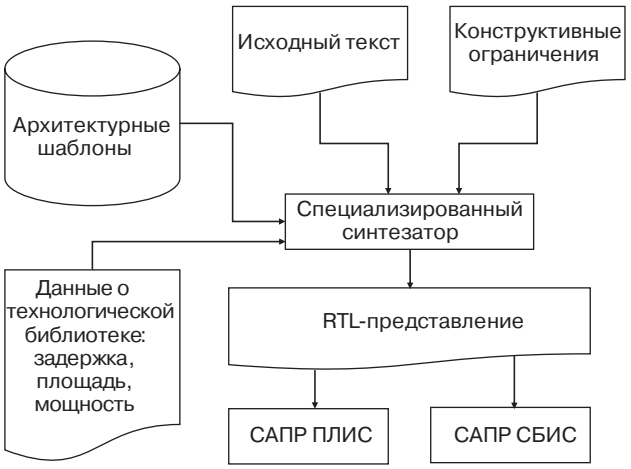

Рис. 2. Маршрут проектирования модулей специализированной вычислительной системы

алгоритма и конструктивные ограничения, представленные в виде предельных характеристик требуемого решения. Разработанный П.Н. Советовым специализированный синтезатор [9] на основе архитектурных шаблонов генерирует RTL-представление модуля, используя для предварительной оценки его характеристик данные о технологической библиотеке. Получаемое RTL-представление в дальнейшем используется в маршрутах проектирования ПЛИС или сверхбольших интегральных схем (СБИС), где соответствующие системы автоматизированного проектирования (САПР) позволяют оценить характеристики модуля после синтеза или после выполнения размещения и трассировки (что дает более точную оценку характеристик по сравнению с оценкой после синтеза).

Например, величины задержек распространения сигналов становятся основанием для повторного синтеза RTL-представления с дополнительной конвейеризацией выявленных критических цепей. Кроме этого, синтезатор обеспечивает дополнительные сведения о взаимосвязях синтезируемых узлов, которые позволяют генерировать проектные ограничения для задания координат отдельных узлов синтезированной схемы (стадий конвейера). Подобные возможности частично обеспечивают такие инструменты проектирования, как *Vitis HLS*[2], однако, в отличие от разработанного маршрута, в *HLS* характеристики технологической платформы задаются в виде библиотек и не подлежат уточнению в процессе оптимизации проекта.

[2] https://www.xilinx.com/products/design-tools/vitis/vitis-hls.html. Дата обращения 10.10.2023. / Accessed October 10, 2023.

ОПТИМИЗАЦИЯ ВЫЧИСЛИТЕЛЬНОГО УЗЛА НА ОСНОВЕ ПРОГРАММНОЙ МОДЕЛИ ВЫЧИСЛЕНИЙ

Разработка компиляторов для новых процессорных архитектур является важной составляющей в обеспечении инструментальных средств проектирования систем на их базе [10]. Трансляторы предметно-ориентированных языков (DSL) в RTL-представление представляют собой перспективный подход к быстрому проектированию аппаратных ускорителей некоторого узкого класса архитектур [11]. Кратко изложим этапы проектирования инструментальной системы, состоящей из встроенного DSL на основе подмножества языка Python и транслятора для синтеза аппаратных ускорителей на основе конвейеризации линейного участка программы.

На вход транслятора подается программа пользователя, представляющая собой поведенческое описание синтезируемого аппаратного ускорителя. Эта программа с помощью модуля ast из стандартной библиотеки Python автоматически преобразуется в форму дерева абстрактного синтаксиса (abstract syntax tree, AST). Проверка и распространение по AST информации о типах, используемых во входной программе, осуществляются на основе механизма аннотации типов Python.

Кроме того, пользователь предоставляет таблицу задержек и правил комбинирования операций для выбранного типа микросхемы, а также один из выбранных шаблонов управления конвейером на языке Verilog. Результатом работы транслятора является код синтезированного конвейеризованного ускорителя на языке Verilog.

Все функции во входной программе встраиваются в главную функцию, а циклы полностью разворачиваются. Далее программа преобразуется в ациклический граф зависимостей по данным (DDG, data dependency graph), при этом на основе нумерации значений осуществляются свертка и продвижение констант, удаление совпадающих выражений и удаление мертвого кода. Для достижения потенциально большего параллелизма вычислений используется сбалансирование высоты деревьев выражений.

Перед непосредственным синтезом конвейера выполняются вспомогательные этапы: этап частичного покрытия DDG с помощью MISO-подграфов (multiple input single output, множество входов и один выход) и этап вычисления максимальных задержек между парами узлов в DDG.

В целевом типе микросхемы могут использоваться ресурсы, позволяющие комбинировать, т.е. совмещать во времени выполнение нескольких операций, например, с помощью таблицы истинности (LUT, look-up table). В трансляторе используется частичное покрытие DDG комбинированными операциями с использованием варианта алгоритма MAXMISO для синтеза команд. Этот алгоритм позволяет перечислить в DDG непересекающиеся подграфы, имеющие количество входов, не более заданного, и один выход.

Результатом синтеза конвейера является DDG с добавленными узлами – конвейеризующими регистрами. Синтез конвейера реализуется преимущественно с использованием сторонних решателей программирования в ограничениях и линейного программирования, что упрощает реализацию генератора кода. Синтез конвейера осуществляется одним из следующих способов: с минимизацией глубины конвейера, с минимизацией общего размера конвейеризующих регистров или же с минимизацией глубины конвейера, за которой следует минимизация общего размера конвейеризующих регистров. При синтезе конвейера учитывается представление программы в виде DDG и используется информация об узлах – конечных потребителях, что упрощает алгоритмы синтеза и позволяет сократить число формируемых ограничений. Аналогичный подход был применен для создания компилятора для специализированного контроллера на базе программируемой пользователем вентильной матрицы (FPGA, field-programmable gate array) [12]. Можно отметить наличие подобных подходов к трансляции высокоуровневого представления программ [13, 14], которые, тем не менее, не предполагают использование обратной связи от САПР, выполняющей трассировку проекта и определение реально достигнутых временных задержек. При этом уделяется внимание конвейерным архитектурам для FPGA [15].

МЕТОДИКА ОЦЕНКИ ТОПОЛОГИЧЕСКОЙ РЕАЛИЗАЦИИ ВЫЧИСЛИТЕЛЬНОГО УЗЛА

Синтезированное RTL-представление, рассмотренное выше, в качестве входной информации использует оценочные сведения о задержках, вносимых отдельными операциями. В этой связи для выбранного технологического базиса требуется:

1. Определить задержки отдельных элементов, реализующих вычисления, поддерживаемые синтезатором.
2. Выявить возможность использования аддитивной модели задержек или определить способ определения суммарной задержки комбинационного узла с учетом взаимодействия отдельных элементов.

Проверка топологической реализации примера конвейерного вычислителя была проведена в САПР ПЛИС *AMD/Xilinx Vivado*[3]. Конвейерный

[3] https://docs.xilinx.com/r/en-US/ug910-vivado-getting-started. Дата обращения 10.10.2023. / Accessed October 10, 2023.

вычислитель реализует алгоритм вращения векторов CORDIC[4], являющийся основой для вычисления трансцендентных функций. Этот выбор связан с тем, что IP-ядро CORDIC входит в состав библиотечных компонентов САПР *AMD/Xilinx Vivado*, и его характеристики могут быть сопоставлены с полученными результатами. Вычисление шагов алгоритма CORDIC совмещено с последовательным вычислением результата умножения с накоплением. Таким образом, обеспечено превышение функциональных возможностей относительно IP-ядра CORDIC. Количественным критерием оценки качества предварительного моделирования задержек является гистограмма запасов временных задержек (slack histogram)[5]. При статическом временном анализе величиной запаса является разность между значением периода тактового сигнала и максимальной задержкой распространения сигнала между синхронными узлами схемы. В зависимости от сложности выражений и взаимного расположения узлов задержка будет индивидуальной для каждой цепи, что позволяет построить гистограмму, показывающую количество цепей, имеющих соответствующие запасы по времени до прихода следующего фронта тактового сигнала. Такая гистограмма называется гистограммой запасов и формируется в САПР *Vivado* по запросу оператора на основе проведенного в САПР статического временного анализа.

Исходя из соображений балансировки стадий конвейера в идеальном случае, можно предположить, что запасы будут сгруппированы около минимальных значений, что будет говорить об отсутствии цепей, имеющих слишком малую задержку, а, следовательно, неэффективно использующих аппаратные ресурсы. Пример гистограммы запасов временных задержек приведен на рис. 3.

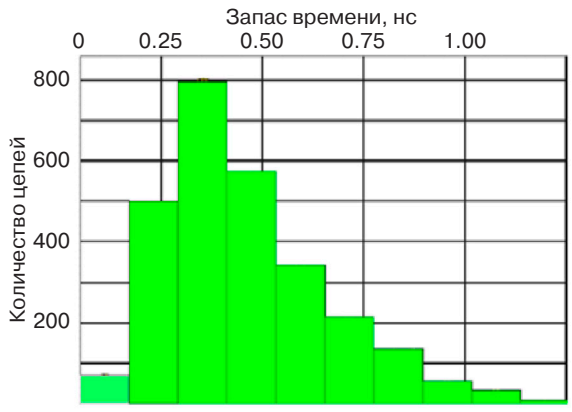

Рис. 3. Гистограмма запасов временных задержек для примера конвейера

[4] Coordinate rotational digital computer.

[5] https://docs.xilinx.com/r/en-US/ug906-vivado-design-analysis/Timing-Analysis. Дата обращения 10.10.2023. / Accessed October 10, 2023.

Группировка цепей на гистограмме показывает, что балансировка стадий конвейера в целом выполнена корректно, поскольку основная часть цепей находится в левой части гистограммы, которая соответствует небольшим величинам запаса по времени.

ПРИМЕРЫ ПРАКТИЧЕСКОЙ АПРОБАЦИИ МЕТОДИКИ

Практическая апробация методики проведена на базе ряда вычислительных узлов конвейерного типа. В качестве примера реализован конфигурируемый конвейер с совмещением вычислений результата умножения с накоплением и поворота вектора на базе алгоритма CORDIC. Синтезированный конвейер позволяет в режиме коммутации функциональных узлов вычислить пару значений синуса, косинуса или умножать независимые 32-разрядные операнды.

При оценке достижимой тактовой частоты цифровых схем на базе FPGA используется понятие logic levels (логические уровни), которое означает количество последовательно соединенных узлов в цепи максимальной длины. Эта цепь является ограничивающим фактором, где достижимая тактовая частота предварительно оценивается как системная тактовая частота, деленная на показатель logic levels. При системной тактовой частоте порядка 700–750 МГц для современных ПЛИС с архитектурой FPGA достижение logic levels, равного 1, представляет собой достаточно сложную техническую задачу. Тем не менее, проведенная балансировка конвейера позволила получить период тактового сигнала 1.6–1.7 нс, что соответствует тактовой частоте 600–625 МГц, для платформы *AMD/Xilinx Kria*, выполненной по технологическим нормам 16 нм FinFET[6].

ЗАКЛЮЧЕНИЕ

Рассмотренные в статье подходы позволяют проводить оптимизацию конвейерных вычислителей, предназначенных для работы в составе СБИС. Полученные положительные результаты позволяют расширить методику для проектирования процессорных узлов и конвейеров с повторением операций, архитектурные шаблоны которых были рассмотрены во вводной части статьи. Совместный анализ проекта на нескольких уровнях (программной модели, схемотехнического и топологического представлений) позволяет проводить оптимизацию вычислителя в соответствии с выбираемыми критериями качества, в т.ч. управляемым образом повышать тактовую частоту для высокопроизводительных вычислительных систем за счет балансировки задержек функциональных узлов конвейера.

[6] https://www.xilinx.com/products/som/kria/k26c-commercial.html. Дата обращения 10.10.2023. / Accessed October 10, 2023.

БЛАГОДАРНОСТИ

Работа выполнена в рамках государственного задания Министерства науки и высшего образования Российской Федерации (тема № FSFZ-2022-0004 «Архитектуры специализированных вычислительных комплексов, методики, алгоритмы и инструменты проектирования цифровых вычислительных устройств»).

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

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.

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

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.]
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

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

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