Russian Technological Journal.
2025;13(3):44–53

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

УДК 004.2
https://doi.org/10.32362/2500-316X-2025-13-3-44-53
EDN QWXGNC

НАУЧНАЯ СТАТЬЯ

Управление топологическими ограничениями при реализации конвейерных вычислительных структур на базе программируемых логических интегральных схем

И.Е. Тарасов [@], **П.Н. Советов**, **Д.В. Люлява**, **Н.А. Дуксин**

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

• **Поступила:** 03.11.2024 • **Доработана:** 13.02.2025 • **Принята к опубликованию:** 20.03.2025

Резюме

Цели. Конвейеризация является эффективным приемом повышения тактовой частоты цифровых схем. При этом балансировка стадий конвейера при синтезе схемы на уровне регистровых передач еще не гарантирует сбалансированную по задержкам распространения сигнала топологическую реализацию такого конвейера в выбранном технологическом базисе. Это обусловлено спецификой алгоритмов размещения и трассировки компонентов цифровых устройств, которые не позволяют получать оптимальные решения в строгом математическом смысле за приемлемое время. В практике разработки цифровых устройств применяются подходы, основанные на комбинации ручного управления топологическими ограничениями, задающими общие правила размещения компонентов, и автоматической оптимизации для локализованных фрагментов схемы, которая в этом случае позволяет получать результаты, близкие к оптимальным. Конвейерные структуры имеют простую схему соединений отдельных стадий, что позволяет продемонстрировать на их примере эффект от применения топологических проектных ограничений. В то же время, на базе конвейерных структур возможна реализация ряда алгоритмов, эффективно дополняющих программируемые процессорные устройства и обеспечивающие аппаратное ускорение некоторых задач. Цель работы – разработка методических рекомендаций по управлению топологическими проектными ограничениями при реализации конвейерных вычислительных структур на базе программируемых логических интегральных схем (ПЛИС) с архитектурой field-programmable gate array (FPGA).

Методы. Использованы методы проектирования и моделирования цифровых систем.

Результаты. На основе проведенного анализа разработаны модификации конвейерного вычислителя 32-разрядного преобразования CORDIC для вычисления трансцендентных функций. Установлено, что добавление проектных ограничений по размещению групп регистров, соответствующих стадиям конвейера, позволяет существенно повысить тактовую частоту по сравнению с автоматическим размещением

и уменьшить время работы алгоритмов трассировки. Полученный эффект систематически воспроизводится в нескольких реализованных вариантах конвейера.
Выводы. Рассмотренные рекомендации позволяют управлять тактовой частотой и количеством стадий конвейерных вычислительных структур при одновременном уменьшении времени одной итерации размещения и трассировки модуля на базе ПЛИС.

Ключевые слова: ПЛИС, конвейер, проектные ограничения, CORDIC

Для цитирования: Тарасов И.Е., Советов П.Н., Люлява Д.В., Дуксин Н.А. Управление топологическими ограничениями при реализации конвейерных вычислительных структур на базе программируемых логических интегральных схем. *Russian Technological Journal*. 2025;13(3):44–53. https://doi.org/10.32362/2500-316X-2025-13-3-44-53, https://www.elibrary.ru/QWXGNC

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 on the basis of hardware and software cooptimization

Ilya E. Tarasov [@], Peter N. Sovietov, Daniil V. Lulyava, Nikita A. Duksin

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

• **Submitted:** 03.11.2024 • **Revised:** 13.02.2025 • **Accepted:** 20.03.2025

Abstract
Objectives. Pipelining is an effective method for increasing the clock frequency of digital circuits. At the same time, balancing the pipeline stages during circuit synthesis at the register transfer level does not yet guarantee a balanced topological implementation of such a pipeline in terms of signal propagation delays according to the selected technological basis. This is due to the specifics of the algorithms for placing and routing components of digital devices, which are not capable of optimizing solutions in a strict mathematical sense in an acceptable time. In practice, approaches for developing digital devices combine manual control of topological constraints that set general rules for placing components with automatic optimization for localized fragments of the circuit are used to obtain results close to optimal. Pipeline circuits are based on a simple connection diagram of individual stages to demonstrate the effect of using topological design constraints on their example. On the basis of pipeline structures, a number of algorithms can be implemented to effectively complement programmable processor devices and provide hardware acceleration of some tasks. The present work develops methodological recommendations for managing topological design constraints in the implementation of pipeline computing structures based on programmable logic devices (PLD) with field-programmable gate array (FPGA) architecture.
Methods. The work is based on accepted methods for designing and modeling digital systems.
Results. Based on the analysis, modifications to a 32-bit CORDIC transcendental function computation pipeline were developed. By adding design constraints on the placement of register groups corresponding to the pipeline stages a significant increase in the clock frequency can be achieved as compared to automatic placement to reduce the running time of the tracing algorithms. The resulting effect is systematically reproduced in several implemented versions of the pipeline.

Conclusions. The presented recommendations can be used to control the clock frequency and number of stages of pipeline computing structures while simultaneously reducing the time of one iteration and routing of a module based on PLD with FPGA architecture.

Keywords: PLD, pipeline, constraints, CORDIC

For citation: Tarasov I.E., Sovietov P.N., Lulyava D.V., Duksin N.A. Method for designing specialized computing systems on the basis of hardware and software cooptimization. *Russian Technological Journal.* 2025;13(3):44–53. https://doi.org/10.32362/2500-316X-2025-13-3-44-53, https://www.elibrary.ru/QWXGNC

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

The authors declare no conflicts of interest.

ВВЕДЕНИЕ

Высокопроизводительные вычислительные системы разрабатываются как комбинация универсальных и специализированных подсистем. При этом на стадии архитектурного проектирования необходимо выделить задачи, подлежащие решению с помощью специализированных подсистем, добавляемых к процессорам общего назначения. Такие задачи должны быть, с одной стороны, массово востребованными, а с другой – недостаточно эффективно решаться с помощью центрального процессорного устройства или существенно замедлять его работу. В цифровой электронике на базе специализированных вычислителей часто реализуются алгоритмы цифровой обработки сигналов [1], вычисление хеш-функций в подсистемах защиты информации [2], ускорение искусственных нейросетей [3] и т.д. В данной работе рассмотрены подходы к проектированию конвейерного вычислителя для ряда примеров.

При разработке вычислительного устройства, преобразующего входной вектор $\vec{x}$ в выходной вектор $\vec{y}$, производится преобразование функции, заданной на входном языке высокого уровня, в последовательность действий на каждой стадии преобразования. Для этого используется синтезатор, разработанный в лаборатории специализированных вычислительных систем РТУ МИРЭА [4]. Выходом синтезатора является текст на языке описания аппаратуры, формирующий регистры на стадиях конвейера и узлы комбинационной логики между ними, выполняющие преобразования f_1, f_2, f_3, … f_n. Аналогичный подход использован в ряде синтезаторов [5][1], однако для разработанного программного продукта имеется возможность управления синтезом на основе обратной связи, формируемой с помощью анализа результатов размещения и трассировки компонентов. При этом минимальный период сигнала тактовой частоты определится максимальной величиной задержки распространения сигнала между стадиями конвейера. Для того, чтобы синтезатор имел возможность равномерно распределить задержку распространения сигнала между стадиями конвейера, необходимо определить составляющие этой задержки для используемого топологического базиса.

Задержка распространения сигнала между регистрами программируемой логической интегральной схемы (ПЛИС) с архитектурой FPGA[2] определяется следующим образом[3]:

$$t = t_{\text{logic}} + t_{\text{route}}, \quad (1)$$

где t_{logic} – задержка, определяемая комбинационными элементами; t_{route} – задержка, определяемая трассировочными цепями ПЛИС.

Для достижения высокой тактовой частоты и равномерного распределения суммарной задержки сигнала между всеми стадиями конвейера синтезатору требуется оценивать составляющие, определяемые комбинационными элементами и определяемые трассировочными цепями. После распределения преобразований сигналов по узлам комбинационной логики получаемое RTL-представление[4] конвейера передается в систему автоматизированного проектирования (САПР) ПЛИС, выполняющую размещение компонентов схемы и трассировку соединений. При этом неоптимальное размещение компонентов приведет к появлению дополнительной задержки распространения сигналов, которая нарушит равномерность распределения задержки по стадиям конвейера.

Стадии разработки конвейерного вычислительного устройства проиллюстрированы на рис. 1.

[1] https://docs.amd.com/r/en-US/ug1399-vitis-hls/HLS-Programmers-Guide. Дата обращения 10.10.2024. / Accessed October 10, 2024.

[2] Field programmable gate array – программируемая пользователем вентильная матрица.

[3] https://docs.xilinx.com/r/en-US/ug906-vivado-design-analysis/Timing-Analysis. Дата обращения 10.10.2024. / Accessed October 10, 2024.

[4] Register transfer level – уровень регистровых передач.

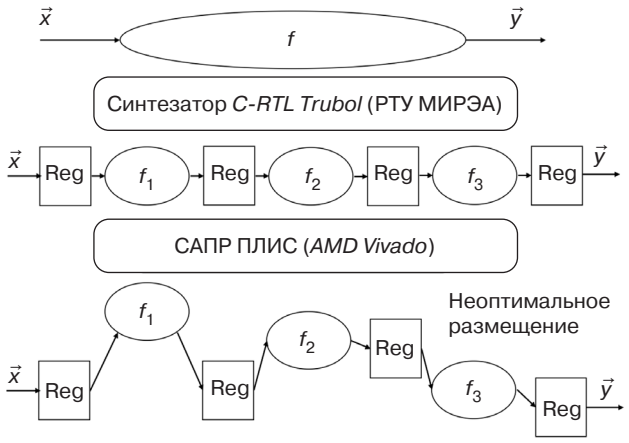

Рис. 1. Стадии разработки конвейерного вычислительного устройства. Синтезатор *C-RTL Trubol* – программное обеспечение, разработанное коллективом авторов. Синтезатор является инструментом формирования описания в формате электронных САПР

Таким образом, для достижения высокой тактовой частоты работы конвейера необходимо обеспечить оценку составляющих задержек распространения сигнала и устранить негативные эффекты неоптимального взаимного размещения соединяемых компонентов на кристалле ПЛИС. Следует отметить, что задача оптимизации размещения в строгом математическом смысле, хотя и может быть сформулирована, не имеет практического решения за приемлемое время ввиду взрывного роста сложности алгоритмов оптимизации для общей формулировки. В практике проектирования на базе ПЛИС применяются проектные ограничения, устанавливающие области размещения групп компонентов – т.н. топологические проектные ограничения (area constraints). Для установленных таким образом групп компонентов (обозначаемых P-blocks) удается выполнить оптимизацию алгоритмами САПР за приемлемое время с достижением субоптимальных результатов. Технические способы управления проектными ограничениями изложены в методическом руководстве AMD UltraFast™ Design Methodology Guide for FPGAs and Systems-on-a-Chip[5]. Исследования, связанные с использованием топологических проектных ограничений, отражены в ряде публикаций и датируются, начиная с 2011 г. [6], что связано с появлением таких инструментов проектирования, как *Xilinx PlanAhead* [7, 8]. В настоящее время применение проектных ограничений продолжает находить применение для обработки сетевых пакетов [9] и задач цифровой обработки сигналов [10].

[5] https://docs.amd.com/r/en-US/ug949-vivado-design-methodology. Дата обращения 10.10.2024. / Accessed October 10, 2024.

УПРАВЛЕНИЕ ОПЕРАЦИЯМИ СИНТЕЗА ФУНКЦИОНАЛЬНЫХ УЗЛОВ КОНВЕЙЕРА

В работе рассмотрена следующая последовательность проектирования конвейерной вычислительной структуры. Для оценки задержки, определяемой комбинационной логикой, создается набор тестовых конвейерных цепочек, содержащих узлы с соответствующей логической функцией. Для этих узлов выполняется синтез и размещение конвейера, а экспериментально полученная оценка записывается в структуру данных, передаваемую синтезатору *C-RTL*. Поскольку синтезатор является оригинальной разработкой, в него были введены соответствующие модификации, позволяющие учитывать задержки, передаваемые из внешних источников.

Следует отметить, что при использовании ПЛИС не требуется оценивать широкий спектр возможных арифметических и логических операций. Поскольку поразрядные операции выполняются на базе таблиц истинности, а операции сложения и вычитания – с помощью специальных узлов «цепь ускоренного переноса» (fast carry chain), достаточно оценить задержку, определяемую этими двумя классами операций.

При наличии архитектурного шаблона и определенных параметров задержек может быть принята последовательность проектирования конвейерной вычислительной структуры, представленная на рис. 2.

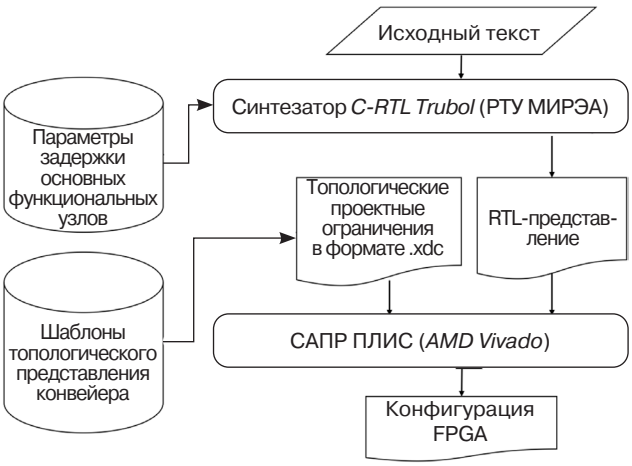

Рис. 2. Последовательность автоматизированного проектирования конвейерной вычислительной структуры

Для разработанной последовательности подразумевается, что на входе имеются исходные тексты программы на проблемно-ориентированном языке высокого уровня, а также проектные ограничения, сформированные на основе исследования характеристик аппаратной платформы. При синтезе используются параметры задержки основных

функциональных узлов, которые предварительно оцениваются в процессе синтеза конвейеров с заранее заданной схемой, явно выделяющей те или иные схемотехнические фрагменты, подлежащие оценке. Специализированный синтезатор формирует RTL-представление модуля на языке описания аппаратуры. Это представление дополняется файлом проектных ограничений в форматах .xdc или .sdc (в зависимости от используемой САПР), который создается путем параметризации одного из разработанных шаблонов топологического представления конвейера.

ФОРМИРОВАНИЕ ТОПОЛОГИЧЕСКИХ ПРОЕКТНЫХ ОГРАНИЧЕНИЙ ДЛЯ КОНВЕЙЕРНОЙ ВЫЧИСЛИТЕЛЬНОЙ СТРУКТУРЫ

При размещении конвейера в ПЛИС необходимо указать правила размещения его отдельных компонентов. Распространенной практикой является наличие в САПР ПЛИС режима иерархического (модульного) размещения, когда алгоритмы размещения используют модули проекта на уровне RTL в качестве локализованных единиц проекта, размещая их компоненты по возможности компактно. В то же время, разработчик имеет возможность сопроводить проект специальными файлами проектных ограничений (constraints), управляющих процессами группы Implementation – временным анализом и размещением. Соответствующие группы команд языка описания проектных ограничений затрагивают временные ограничения (timing constraints) и топологические ограничения (area constraints).

Команда для описания проектных ограничений в рамках одной стадии имеет вид:

```
create_pb <название pblock>
<координата pblock по оси X>
<координата pblock по оси Y>
<ширина pblock> <высота pblock>
<список элементов, связанных с pblock>
```

В этом примере для триггеров, имя которых соответствует шаблону, устанавливаются границы на кристалле ПЛИС с заданными координатами (рис. 3). Размеры границ на кристалле в точности, как и координаты, сопоставляются с общей топологией кристалла, количеством регистров и логических элементов, задействованных в каждой конкретной стадии. Шаблон имени подбирается, исходя из формата описания на RTL-уровне.

В процессе выполнения работы было подтверждено предположение о целесообразности описания топологических проектных ограничений только для регистров конвейера. Это обусловлено тем, что комбинационная логика между группами регистров,

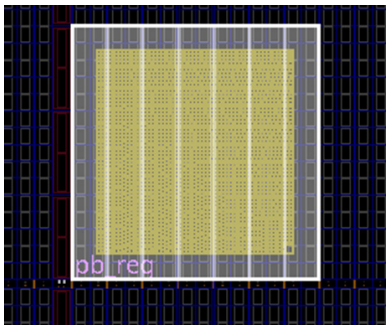

Рис. 3. Иллюстрация к операции выделения P-блока в ПЛИС с архитектурой FPGA

относящихся к стадиям конвейера, имеет локализованные связи с этими группами. В этом случае установка регионов для расположения стадий конвейеров приведет к компактному размещению связанных с регистрами узлов комбинационной логики, оставляя при этом возможность для САПР выполнять локальные оптимизации. В этом случае ручное управление отдельными компонентами конвейера, включая отдельные триггеры и узлы комбинационной логики, чрезмерно трудоемко.

Для разработанного в РТУ МИРЭА синтезатора схемы в формате RTL рекомендована модификация в виде генерации имен регистров с внедрением в них фрагментов, однозначно идентифицирующих стадию конвейера, которой принадлежит этот регистр. Данные сведения имеются во внутреннем представлении синтезатора, однако ранее не использовались. Анализ мировых аналогов показал, что идентификация стадии конвейера в них также невозможна, в экспортируемом RTL-представлении обычно используется сквозная нумерация отдельных триггеров схемы. При этом внедрение информации о стадии конвейера позволяет выделять соответствующую этой стадии группу регистров, используя регулярное выражение вида:

```
*/pipeline_unit/*/*reg_Ki\_*
```

ПРИМЕРЫ ПРАКТИЧЕСКОЙ АПРОБАЦИИ МЕТОДИКИ

Практическая апробация методики была выполнена путем реализации конвейеров нескольких видов. Например, алгоритм CORDIC[6] [11] заключается в последовательном применении операции поворота вектора. Схожие действия в виде комбинации сложения и сдвига используются при последовательном умножении с накоплением (Multiply and Accumulate).

[6] Метод CORDIC от англ. COordinate Rotation DIgital Computer – цифровой вычислитель поворота системы координат; метод «цифра за цифрой». [CORDIC is an acronym for COordinate Rotation DIgital Computer; a “digit by digit” method.]

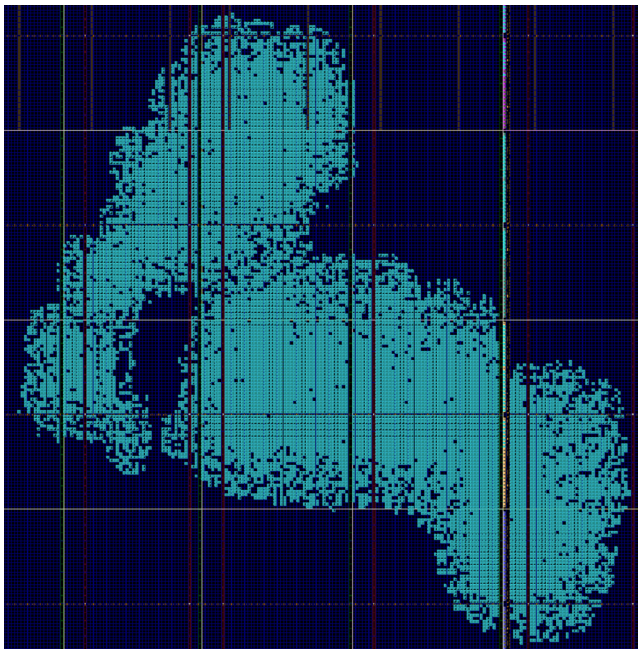

Рис. 4. Размещение стадий конвейера в автоматическом режиме в САПР *Vivado*

Данные операции можно совместить в одном конфигурируемом конвейере. Фрагмент схемы, сгенерированный в САПР *Vivado*[7], приведен на рис. 4. Можно видеть, что автоматическое размещение компонентов конвейера с последующей оптимизацией привело к получению локально оптимального решения.

Поиск оптимального решения с точки зрения САПР не учитывает разбиение на стадии, свойственные конвейерной архитектуре. Исходя из этого предположения, характеристики при размещении можно улучшить с применением описанного ранее подхода. В результате экспериментов был сделан вывод о повышении основных схемотехнических показателей схемы в случае, если границы блоков каждой стадии будут частично перекрывать границы соседних блоков. Результат, полученный с применением соответствующей стратегии, представлен на рис. 5.

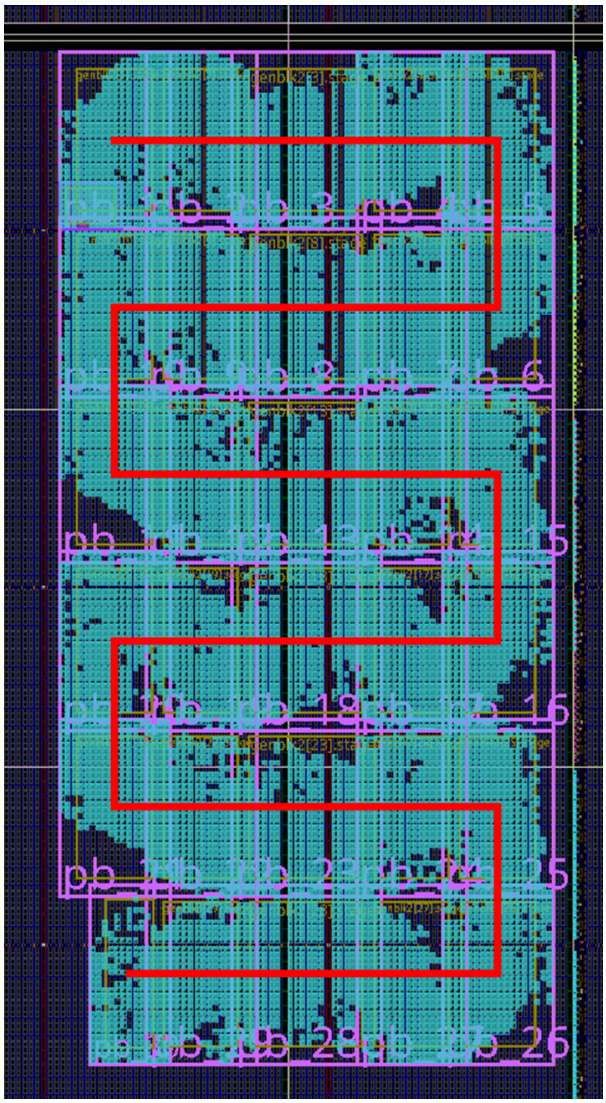

Рис. 5. Размещение стадий конвейера при использовании топологических проектных ограничений

Размещение проводилось в ПЛИС AMD FPGA Virtex™ UltraScale™+xcvu440_CIV-flga2892-3-e[8] и, по сравнению со стандартным размещением (рис. 6), позволило добиться тактовой частоты в 1 ГГц (рис. 7).

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	−0.179 ns	Worst Hold Slack (WHS):	0.016 ns	Worst Pulse Width Slack (WPWS):	−0.176 ns
Total Negative Slack (TNS):	−45.105 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	−0.176 ns
Number of Failing Endpoints:	970	Number of Failing Endpoints:	0	Number of Failing Endpoints:	1
Total Number of Endpoints:	96075	Total Number of Endpoints:	96075	Total Number of Endpoints:	94415

Рис. 6. Фрагмент отчета САПР с временными характеристиками проекта для схемотехнического решения, полученного в автоматическом режиме

[7] https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vivado.html. Дата обращения 10.10.2024. / Accessed October 10, 2024.

[8] https://www.xilinx.com/products/boards-and-kits/1-66ql3z.html. Дата обращения 10.10.2024. / Accessed October 10, 2024.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.005 ns	Worst Hold Slack (WHS):	0.018 ns	Worst Pulse Width Slack (WPWS):	–0.176 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	–0.176 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	1
Total Number of Endpoints:	142287	Total Number of Endpoints:	142287	Total Number of Endpoints:	140609

Рис. 7. Фрагмент отчета САПР с временными характеристиками проекта для схемотехнического решения при использовании топологических проектных ограничений

Еще более ярко заметен результат применения подхода при увеличении числа стадий алгоритма CORDIC. На рис. 8 и 9 представлены результаты размещения вычислителя для 64 стадий при тактовой частоте 600 МГц.

Такой же подход был применен для размещения рассматриваемой схемы на чипах AMD FPGA Artix-7 xc7a100tcsg324-1[9] (16 стадий, тактовая частота 400 МГц) и AMD FPGA Kintex™ UltraScale™ xcku115_CIV-flvf1924-3-e[10] (32 стадии, тактовая частота 850 МГц). Работоспособность подхода доказывается, исходя из сравнения результатов временного анализа для рассматриваемых схем (рис. 10–13).

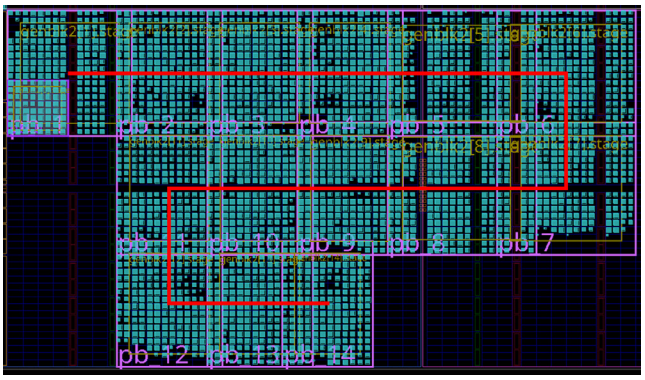

Рис. 10. Размещение стадий конвейера при использовании топологических проектных ограничений (ПЛИС xc7a100tcsg324-1)

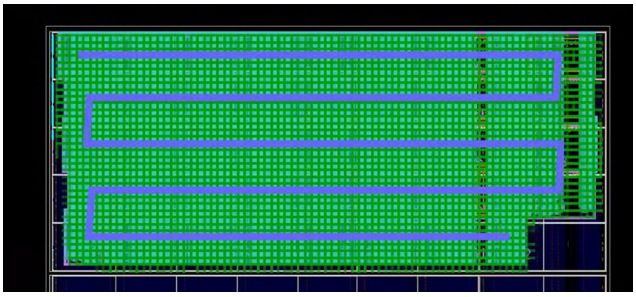

Рис. 8. Размещение стадий конвейера (64 стадии, тактовая частота 600 МГц)

Design Timing Summary

Setup		Hold	
Worst Negative Slack (WNS):	0.024 ns	Worst Hold Slack (WHS):	0.010 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	16911	Total Number of Endpoints:	16911

All user specified timing constraints are met.

Рис. 11. Фрагмент отчета САПР с временными характеристиками проекта для полученного варианта размещения (ПЛИС xc7a100tcsg324-1)

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.005 ns	Worst Hold Slack (WHS):	0.016 ns	Worst Pulse Width Slack (WPWS):	0.100 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	780243	Total Number of Endpoints:	780243	Total Number of Endpoints:	775890

All user specified timing constraints are met.

Рис. 9. Фрагмент отчета САПР с временными характеристиками проекта конвейерного вычислителя CORDIC (64 стадии, тактовая частота 600 МГц)

[9] https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga/artix-7.html. Дата обращения 10.10.2024. / Accessed October 10, 2024.

[10] https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga/kintex-ultrascale-plus.html. Дата обращения 10.10.2024. / Accessed October 10, 2024.

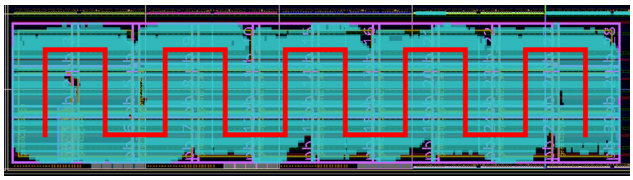

Рис. 12. Размещение стадий конвейера при использовании топологических проектных ограничений (ПЛИС xcku115_CIV-flvf1924-3-e)

Design Timing Summary

Setup		Hold	
Worst Negative Slack (WNS):	0.040 ns	Worst Hold Slack (WHS):	0.030 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	142195	Total Number of Endpoints:	142195

Timing constraints are not met.

Рис. 13. Фрагмент отчета САПР с временными характеристиками проекта для полученного варианта размещения (ПЛИС xcku115_CIV-flvf1924-3-e)

Полученные результаты демонстрируют систематическое улучшение характеристик проектов вычислителей с конвейерной структурой при установке проектных ограничений для регистров отдельного модуля, описывающего конвейер. Продолжающийся интерес к конвейерным вычислительным узлам [12] позволяет рассматривать выбранное направление исследований как перспективное. Кроме этого, конвейерные устройства могут выступать в качестве подсистем вычислительных комплексов, увеличивая их эффективность как в широко распространенных задачах [13], так и в качестве ускорителей узкого подкласса вычислений [14, 15].

ЗАКЛЮЧЕНИЕ

Рассмотренные материалы отражают результаты, достигнутые лабораторией специализированных вычислительных систем РТУ МИРЭА при разработке методики проектирования специализированных ускорителей вычислений с конвейерной архитектурой. Ориентация на простую аппаратную архитектуру с локализованными связями между узлами позволила разработать набор алгоритмов и проектных мероприятий, которые обеспечили систематический эффект улучшения характеристик топологического представления вычислительного устройства из его исходного описания на языке высокого уровня. Полученные результаты могут быть адаптированы к архитектурным шаблонам других типов с целью расширения номенклатуры специализированной электронной компонентной базы.

БЛАГОДАРНОСТИ

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

ACKNOWLEDGMENTS

The research is carried out within the framework of the State Assignment of the Ministry of Science and Higher Education of the Russian Federation (No. FSFZ-2022-0004 “Architectures of special-purpose computing units and procedures, algorithms, and tools for the design of digital computing units”).

Вклад авторов

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

Authors’ contribution

All authors equally contributed to the research work.

СПИСОК ЛИТЕРАТУРЫ

1. Saidov B.B., Telezhkin V.F., Gudaev N.N., et al. Development of Equipment for Experimental Study of Digital Algorithms in Nonstationary Signal Processing Problems. *Ural Radio Engineering Journal.* 2022;6(2):186–204. https://doi.org/10.15826/urej.2022.6.2.004
2. Jasek R. SHA-1 and MD5 Cryptographic Hash Functions: Security Overview. *Communications* (*Komunikacie*). 2015;17(1):73–80.
3. Carrión D.S., Prohaska V., Diez O. Exploration of TPUs for AI Applications. In: Daimi K., Al Sadoon A. (Eds.). *Proceedings of the Second International Conference on Advances in Computing Research* (*ACR’24*). ACR 2024. Lecture Notes in Networks and Systems. Springer; 2024. V. 956. P. 559. https://doi.org/10.1007/978-3-031-56950-0_47
4. 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. *Russian Technological Journal.* 2024;12(3):37–45 https://doi.org/10.32362/2500-316X-2024-12-3-37-45
5. Алехин В.А. Проектирование электронных систем с использованием SystemC и SystemC–AMS. *Russian Technological Journal.* 2020;8(4):79–95. https://doi.org/10.32362/2500-316X-2020-8-4-79-95

6. Pham-Quoc C., Dinh-Duc A.-V. Automatic generation of area constraints for FPGA implementation. In: *2011 IEEE 3rd International Conference on Communication Software and Networks* (*ICCSN*). 2011. P. 469–472. https://doi.org/10.1109/ICCSN.2011.6014937
7. Li K., Lei L., Guang Q., Shi J.-Y., Hao Y. Improving the performance of an SOC design for network processing based on FPGA with PlanAhead. In: 2011 *International Conference on Electronics, Communications and Control* (*ICECC*). 2011. P. 297–300. https://doi.org/10.1109/ICECC.2011.6066640
8. Sarker A.L. Md, Lee M.H. Synthesis of VHDL code for FPGA design flow using Xilinx PlanAhead tool. In: *2012 International Conference on Education and e-Learning Innovations* (*ICEELI*). 2012. https://doi.org/10.1109/ICEELI.2012.6360614
9. Song X., Lu R., Guo Z. High-Performance Reconfigurable Pipeline Implementation for FPGA-Based SmartNIC. *Micromachines.* 2024;15(4):449. https://doi.org/10.3390/mi15040449
10. Anderson T., Wheeler T.J. An FPGA-based hardware accelerator supporting sensitive sequence homology filtering with profile hidden Markov models. *BMC Bioinformatics.* 2024;25:247. https://doi.org/10.1186/s12859-024-05879-3
11. Тарасов И.Е., Советов П.Н. *Устройство для вычисления трансцендентных функций и умножения двоичных чисел*: пат. 222880 U1 РФ. Заявка № 2023131099; заявл. 28.11.2023; опубл. 22.01.2024. Бюл. № 3.
12. Oishi R., Kadomoto J., Irie H., Sakai S. FPGA-based Garbling Accelerator with Parallel Pipeline Processing. *IEICE Trans. Inform. Syst.* 2023;E106.D(12):1988–1996. https://doi.org/10.1587/transinf.2023PAP0002
13. 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*). 2016. P. 77–84. https://doi.org/10.1109/FPT.2016.7929192
14. 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*). 2018. P. 27–29. https://doi.org/10.1109/ISCA.2018.00011
15. Hennessy J.L., Patterson D.A. *Computer Architecture*: *A Quantitative Approach:* 6th ed. The Morgan Kaufmann Series in Computer Architecture and Design. 2017. 936 p.

REFERENCES

1. Saidov B.B., Telezhkin V.F., Gudaev N.N., et al. Development of Equipment for Experimental Study of Digital Algorithms in Nonstationary Signal Processing Problems. *Ural Radio Engineering Journal.* 2022;6(2):186–204. https://doi.org/10.15826/urej.2022.6.2.004
2. Jasek R. SHA-1 and MD5 Cryptographic Hash Functions: Security Overview. *Communications* (*Komunikacie*). 2015;17(1):73–80.
3. Carrión D.S., Prohaska V., Diez O. Exploration of TPUs for AI Applications. In: Daimi K., Al Sadoon A. (Eds.). *Proceedings of the Second International Conference on Advances in Computing Research* (*ACR'24*). ACR 2024. Lecture Notes in Networks and Systems. Springer; 2024. V. 956. P. 559. https://doi.org/10.1007/978-3-031-56950-0_47
4. 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. *Russian Technological Journal.* 2024;12(3):37–45. https://doi.org/10.32362/2500-316X-2024-12-3-37-45
5. Alekhin V.A. Designing Electronic Systems Using SystemC and SystemC–AMS. *Russian Technological Journal.* 2020;8(4):79–95 (in Russ.). https://doi.org/10.32362/2500-316X-2020-8-4-79-95
6. Pham-Quoc C., Dinh-Duc A.-V. Automatic generation of area constraints for FPGA implementation. In: *2011 IEEE 3rd International Conference on Communication Software and Networks* (*ICCSN*). 2011. P. 469–472. https://doi.org/10.1109/ICCSN.2011.6014937
7. Li K., Lei L., Guang Q., Shi J.-Y., Hao Y. Improving the performance of an SOC design for network processing based on FPGA with PlanAhead. In: 2011 *International Conference on Electronics, Communications and Control* (*ICECC*). 2011. P. 297–300. https://doi.org/10.1109/ICECC.2011.6066640
8. Sarker A.L.Md., Lee M.H. Synthesis of VHDL code for FPGA design flow using Xilinx PlanAhead tool. In: *2012 International Conference on Education and e-Learning Innovations* (*ICEELI*). 2012. https://doi.org/10.1109/ICEELI.2012.6360614
9. Song X., Lu R., Guo Z. High-Performance Reconfigurable Pipeline Implementation for FPGA-Based SmartNIC. *Micromachines.* 2024;15(4):449. https://doi.org/10.3390/mi15040449
10. Anderson T., Wheeler T.J. An FPGA-based hardware accelerator supporting sensitive sequence homology filtering with profile hidden Markov models. *BMC Bioinformatics.* 2024;25:247. https://doi.org/10.1186/s12859-024-05879-3
11. Tarasov I.E., Sovetov P.N. *Device for Calculating Transcendental Functions and Multiplying Binary Numbers*: Pat. 222880 U1 RF. Publ. 22.01.2024 (in Russ.).
12. Oishi R., Kadomoto J., Irie H., Sakai S. FPGA-based Garbling Accelerator with Parallel Pipeline Processing. *IEICE Trans. Inform. Syst.* 2023;E106.D(12):1988–1996. https://doi.org/10.1587/transinf.2023PAP0002
13. 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*). 2016. P. 77–84. https://doi.org/10.1109/FPT.2016.7929192

14. 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*). 2018. P. 27–29. https://doi.org/10.1109/ISCA.2018.00011
15. Hennessy J.L., Patterson D.A. *Computer Architecture*: *A Quantitative Approach:* 6th ed. The Morgan Kaufmann Series in Computer Architecture and Design. 2017. 936 p.

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: peter.sovietov@gmail.com. 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

Дуксин Никита Александрович, инженер, лаборатория специализированных вычислительных систем, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: duksin@mirea.ru. SPIN-код РИНЦ 1082-8956, Scopus Author ID 58811361100, https://orcid.org/0009-0009-0014-7065

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: peter.sovietov@gmail.com. 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

Nikita A. Duksin, Engineer, Laboratory of Specialized Computing Systems, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: duksin@mirea.ru. RSCI SPIN-code 1082-8956, Scopus Author ID 58811361100, https://orcid.org/0009-0009-0014-7065