MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 29, NO. 1, 2022
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

Methods for Change Parallelism in Process of High-level VLSI Synthesis

I. N. Ryzhenko[1], O. V. Nepomnyaschy[1], A. I. Legalov[2], V. V. Shaidurov[3]

DOI: 10.18255/1818-1015-2022-1-60-72

[1]Siberian Federal University, 79 Svobodny pr., Krasnoyarsk 660041, Russia.
[2]Higher School of Economics, 20 Myasnitskaya str., Moscow 101000, Russia.
[3]Krasnoyarsk Science Centre of the Siberian Branch of Russian Academy of Science, 50 Akademgorodok, Krasnoyarsk 660036, Russia.

MSC2020: 94-04
Research article
Full text in Russian

Received September 2, 2021
After revision February 23, 2022
Accepted March 9, 2022

In this paper methods for increasing the efficiency of VLSI development based on the method of architecture-independent design are proposed. The route of high-level VLSI synthesis is considered. The principle of constructing a VLSI hardware model based on the functional-flow programming paradigm is stated.
The results of the development of methods and algorithms for transformation functional-parallel programs into programs in HDL languages that support the design process of digital chips are presented. The principles of assessment are considered and the classes of resources required for the analysis of design solutions are identified. Reduction coefficients and methods of their calculation for each resource class have been introduced. An algorithm for calculating the reduction coefficients and estimating the required resources is proposed. An algorithm for converting parallelism is proposed, taking into account the specified constraints of the target platform. A mechanism for the exchange of metrics with an architecture-dependent level has been developed. Examples of parallelism reduction for the FPGA platform and practical implementation of FFT algorithms in the Virtex® UltraScale FPGA basis are given. The developed methods and algorithms make it possible to use the method of architecture-independent synthesis for transferring VLSI projects to various architectures by changing the parallelism of the circuit and equivalent transformations of parallel programs. The proposed approach provides many options for hardware solutions for implementation on various target platforms.

Keywords: parallel computing; dataflow; functional programming; high-level synthesis; VLSI

INFORMATION ABOUT THE AUTHORS

Igor Nikolaevich Ryzhenko correspondence author	orcid.org/0000-0002-3069-9102. E-mail: rodgi.krs@gmail.com Senior lecturer, PhD student.
Oleg Vladimirovich Nepomnyaschy	orcid.org/0000-0002-2459-6414. E-mail: 2955005@gmail.com Head of the Department of Computer Science, PhD, Professor.
Aleksandr Ivanovich Legalov	orcid.org/0000-0002-5487-0699. E-mail: alegalov@hse.ru Dr. (Doctor) of Technical Science, Professor.
Vladimir Viktorovich Shaidurov	orcid.org/0000-0002-7883-5804. E-mail: shaidurov04@mail.ru Dr. (Doctor) of Physics and Mathematics Sciences, Corresponded Member of RAS.

For citation: I. N. Ryzhenko, O. V. Nepomnyaschy, A. I. Legalov, and V. V. Shaidurov, “Methods for Change Parallelism in Process of High-level VLSI Synthesis”, *Modeling and analysis of information systems*, vol. 29, no. 1, pp. 60-72, 2022.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 29, № 1, 2022
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

Методы преобразования параллелизма в процессе высокоуровневого синтеза СБИС

И. Н. Рыженко[1], О. В. Непомнящий[1], А. И. Легалов[2], В. В. Шайдуров[3]

DOI: 10.18255/1818-1015-2022-1-60-72

[1]Сибирский Федеральный Университет, пр. Свободный, д. 79, г. Красноярск, 660041 Россия.
[2]Национальный исследовательский университет "Высшая школа экономики", ул. Мясницкая, д. 20, г. Москва, 101000 Россия.
[3]ФГБНУ Федеральный исследовательский центр "Красноярский научный центр Сибирского отделения Российской академии наук", Академгородок, д. 50, г. Красноярск, 660036 Россия.

УДК 004.4'416+004.432.42
Научная статья
Полный текст на русском языке

Получена 2 сентября 2021 г.
После доработки 23 февраля 2022 г.
Принята к публикации 9 марта 2022 г.

Предложены методы повышения эффективности разработки СБИС на основе метода архитектурно-независимого проектирования. Рассмотрен маршрут высокоуровневого синтеза СБИС. Изложен принцип построения аппаратной модели СБИС на основе функционально-потоковой парадигмы программирования.
Представлены результаты разработки методов и алгоритмов трансформации, функционально-потоковых параллельных программ в программы на языках описания аппаратуры, обеспечивающих поддержку процесса проектирования цифровых однокристальных систем. Рассмотрены принципы оценки и выделены классы ресурсов, требуемых для анализа проектных решений. Введены коэффициенты редукции и методики их расчета по каждому классу ресурсов. Предложен алгоритм расчета коэффициентов редукции и оценки требуемых ресурсов. Предложен алгоритм преобразования параллелизма с учетом заданных ограничений целевой платформы. Разработан механизм обмена метриками с архитектурно-зависимым уровнем. Приведены примеры редукции параллелизма для платформы ПЛИС и практическая реализация тестовых алгоритмов БПФ в базисе ПЛИС Virtex® UltraScale. Разработанные методы и алгоритмы позволяют использовать метод архитектурно-независимого синтеза для переноса проектов СБИС на различные архитектуры с помощью изменения параллелизма схемы и эквивалентных преобразований параллельных программ. Предложенный подход обеспечивает множество вариантов аппаратных решений для реализации на различных целевых платформах.

Ключевые слова: параллельные вычисления; потоки данных; функционально-потоковое параллельное программирование; цифровая интегральная схема; высокоуровневый синтез

ИНФОРМАЦИЯ ОБ АВТОРАХ

Игорь Николаевич Рыженко автор для корреспонденции	orcid.org/0000-0002-3069-9102. E-mail: rodgi.krs@gmail.com ст. преподаватель, аспирант.
Олег Владимирович Непомнящий	orcid.org/0000-0002-2459-6414. E-mail: 2955005@gmail.com зав. кафедрой вычислительной техники, к.т.н., профессор.
Александр Иванович Легалов	orcid.org/0000-0002-5487-0699. E-mail: alegalov@hse.ru доктор технических наук, профессор.
Владимир Викторович Шайдуров	orcid.org/0000-0002-7883-5804. E-mail: shaidurov04@mail.ru доктор физ.-мат. наук, член-корреспондент РАН.

Для цитирования: I. N. Ryzhenko, O. V. Nepomnyaschy, A. I. Legalov, and V. V. Shaidurov, "Methods for Change Parallelism in Process of High-level VLSI Synthesis", *Modeling and analysis of information systems*, vol. 29, no. 1, pp. 60-72, 2022.

Введение

При проектировании сверхбольших интегральных схем (СБИС) используют два основных подхода: классический, подразумевающий низкоуровневое представление исходных алгоритмов на языках описания аппаратуры (HDL – hardware description language), и высокоуровневый, при котором проект изначально абстрагируется от конечной реализации и описывается на системном уровне (ESL – Entire System Level).

В настоящее время классический подход не позволяет обеспечить требуемые сроки реализации и технические характеристики для сложно-функциональных проектов, реализуемых в базисе СБИС. Поэтому эффективные решения находят при использовании высокоуровневых подходов, что подтверждается мировыми тенденциями перехода на более высокие уровни абстракции в процессе разработки [1—3].

Так же известны попытки использования традиционного маршрута проектирования совместно с высокоуровневыми программными средствами, которые ранее были ориентированы на решение совершенно других задач, например, MatLab или LabView [1]. Но и данном случае, при высокой сложности проекта, конечная реализация имеет необоснованно высокие ресурсные требования и зачастую не соответствует требуемым техническим характеристикам.

При этом известные высокоуровневые подходы используют модели вычислений, зачастую плохо подходящие для представления СБИС. Например, компания Xilinx предлагает технологию HLS-синтеза [4], где применяется описание исходного алгоритма на императивном Си-подобном языке, что не обеспечивает переносимость программ на платформы сторонних производителей. Кроме того, переход от исходного Си-описания к низкоуровневому для последующей реализации подразумевает ручное или полуавтоматическое выделение параллелизма, что требует значительных временных затрат и высокой квалификации разработчика.

Поскольку СБИС является системой параллельной обработки данных, архитектурная независимость может быть достигнута при использовании подходов, обеспечивающих переносимость параллельных программ, в том числе использование моделей вычислений непосредственно задающих программу в виде графа потока данных [5]. Такой подход рассматривается в ряде работ по программированию универсальных параллельных вычислительных систем [6, 7]. Известны работы по использованию концепции ресурсно-независимого параллелизма и для реконфигурируемых вычислительных систем [8], где предлагается использовать язык программирования высокого уровня COLAMO. Близкий подход реализован в языке Set@l, программа на котором представляет собой описание алгоритма в виде архитектурно-независимого информационного графа и набора архитектурно-зависимых аспектов, что позволяет переносить ее между различными параллельными архитектурами без изменения алгоритма [9].

Таким образом, для обеспечения архитектурной независимости, требований по ресурсным ограничениям и основным техническим характеристикам, а также для сокращения сроков проектирования следует использовать максимальное абстрагирование исходных алгоритмов от целевого кристалла и создать механизм перехода на нижний, RTL-уровень с поддержкой сквозной верификации и параллелизма на уровне операций. Авторами предложен метод представления алгоритмов на верхнем уровне иерархии системного проектирования, который при последующем нисходящем проектировании обеспечивает сохранение параллелизма [10]. При этом предложенный подход дополняет известные маршруты проектирования, обеспечивая переносимость и архитектурную независимость исходных алгоритмов.

1. Метод и маршрут высокоуровневого синтеза

Цифровые СБИС функционируют в синхронном потоковом режиме, а это означает, что в данном случае реализуется граф потока данных, где узлами являются составляющие СБИС функциональ-

ные блоки, а ребрами – сигналы данных и управления. При этом блоки функционируют параллельно, а зависимость между ними определяется только сигналами управления, которые определяют готовность данных на его входах. Поэтому основная идея предложенного метода архитектурно-независимого синтеза СБИС заключается в переходе от высокоуровневого представления исходных алгоритмов на языке функционально-потокового параллельного программирования к низкоуровневому RTL путем использования модели вычислений с неограниченными ресурсами [11]. Такой подход использует одинаковую модель вычислений на всех уровнях разработки, а язык и метод разработки верхнего уровня ориентирован на описание параллельных программ, управляемых по готовности данных.

В работе [11] описаны используемые функционально-потоковая модель параллельных вычислений и язык программирования «Пифагор». В рамках дальнейших исследований разработаны инструментальные средства, поддерживающие отладку, верификацию и выполнение функционально-потоковых параллельных (ФПП) программ [12]. Архитектурная независимость системы обеспечивается следующим образом: считается, что виртуальная машина, предназначенная для выполнения ФПП программ, имеет неограниченные вычислительные ресурсы. Это позволяет выделять для каждой операции новый вычислительный ресурс. Процесс перехода от архитектурно-независимого описания параллелизма к конкретной вычислительной системе, в данном случае к цифровой схеме на кристалле, представляет собой редукцию параллелизма проекта СБИС под имеющиеся вычислительные ресурсы.

Авторами предложен маршрут проектирования, согласно которому в ходе трансляции с ФП языка строятся следующие промежуточные представления и структуры данных: информационный граф (ИГ), описывающий функциональные преобразования данных; управляющий граф (УГ), определяющий передачу управления между выполняемыми функциями. При этом передачи управляющих сигналов могут происходить параллельно, что является особенностью разрабатываемой программы. Кроме этого вводится дополнительный слой, так называемый HDL-граф, который используется на этапе эквивалентных преобразований параллелизма и переходе к низкоуровневому описанию.

Ключевой особенностью предложенного метода синтеза являются механизмы сокращения (редукции) параллелизма задачи из исходного максимально-параллельного описания алгоритма под конкретные ресурсные ограничения платформы. В работах [6, 7, 11] показано, что такой подход обеспечивает переносимость параллельных алгоритмов на различные платформы, допуская также преобразование в императивные последовательные программы [13]. Для решения проблемы эффективного сокращения параллелизма необходимо выполнить оценку ресурсов и производительности получаемого архитектурного решения. Для этого требуется провести расчет коэффициента редукции по каждому классу ресурсов (R_i) и выполнить свертку параллелизма схемы для достижения требуемого коэффициента редукции R.

2. Оценка ресурсов

Для оценки ресурсов используется промежуточное представление – HDL-граф, в котором уже заданы архитектурно-зависимые данные. HDL-граф представляет собой ациклический граф в ярусно-параллельной форме, в каждом узле которого заданы типы и разрядности данных. При оценке ресурсов необходимо определить классы ресурсов H_i, по которым оценивается схема.

В качестве примера рассмотрим платформу ПЛИС, в которой выделим основные, характерные именно для ПЛИС, ресурсы:

- количество логических ячеек H_{lut};
- количество регистров H_{ff};
- количество блочной памяти H_{ram};
- количество арифметических и иных специализированных вычислительных блоков H_{dsp}.

Означенные ресурсы можно сгруппировать в два класса/подмножества: подмножество ресурсов памяти $H_{ram/ff} = \{H_{ff}, H_{ram}\}$ и подмножество вычислительных ресурсов $H_c = \{H_{lut}, H_{dsp}\}$:

$$H^{FF} = \sum H_i^{FF}. \tag{1}$$

Ресурсы внутри подмножества взаимозаменяемы с некоторыми ограничениями. Например, для ресурсов памяти любое хранение данных может быть реализовано как на блочной памяти, так и на триггерах, но ограниченное по объему. Для вычислительных ресурсов для любой арифметической/логической и т. д. операции можно реализовать схему на логических ячейках, в то время как тип и набор операций для специализированных блоков ограничен.

При оценке ресурсов памяти суммарный объем ресурсов, доступных на конкретной архитектурной платформе, может быть приведен к общему объему, измеренному в бит, Кбит и т. д.

Для оценки требуемого для конкретной реализации схемы ресурса рассмотрим HDL-граф схемы. Каждый i-тый слой ЯПФ формы состоит из набора информационных входов V_i и набора операций O_i. Каждый информационный вход вершин HDL-графа после типизации имеет разрядность W_i. Исходя из количества входов и разрядности каждого входа для каждого слоя графа, можно определить количество ресурсов памяти, требуемых для хранения результата на соответствующей стадии графа:

$$H_i^{FF} = \sum V_i * W_i. \tag{2}$$

Для расчета ресурса необходимо выполнить обход графа и суммирование разрядностей входов и выходов всех вершин:

$$H^{FF} = \sum H_i^{FF}. \tag{3}$$

После первоначальной оценки требуемого ресурса памяти для исходной максимально-параллельной реализации схемы возможны два варианта:

- требуемый ресурс меньше доступного $H^{FF} < H_{ram/ff}$;
- требуемый ресурс больше доступного $H^{FF} >= H_{ram/ff}$.

Первый вариант не требует расчета коэффициента редукции по ресурсам памяти $R_{ram/ff}$, но в случае изменения схемы при редукции по другим ресурсам оценку и проверку ресурса памяти необходимо повторить.

Во втором случае рассчитывается коэффициент редукции $R_{ram/ff}$:

$$R_{ram/ff} = H^{FF}/H_{ram/ff}. \tag{4}$$

Этот коэффициент далее используется в алгоритме редукции параллелизма схемы.

Помимо ограничения на объем памяти необходимо учитывать также ограничение на производительность памяти (ширина интерфейса данных). Для памяти, построенной на регистрах/триггерах, данного ограничения не существует, так как ширина шины данных равна объему данных. Для блочной памяти объем считываемых за такт данных меньше, чем объем хранимых данных. В случае, если ресурс H^{FF} превышает доступный объем регистров H_{ff}, необходимо рассчитывать суммарный интерфейс данных к памяти и коэффициент редукции по интерфейсу памяти $R_{ram/data}$.

Для реализации алгоритма расчета $R_{ram/data}$ необходимо определить минимально-необходимый коэффициент редукции по интерфейсу памяти. Для этого требуется выбрать из всех стадий конвейера набор стадий с максимальным суммарным, реализуемым на регистрах, интерфейсом. Поэтому из множества стадий конвейера выбирается подмножество стадий, такое что:

$$max \sum H_i^{FF} \&\& \sum H_i^{FF} < (H_{ram/ff} - H_f^{FF} f). \tag{5}$$

Тогда коэффициент $R_{ram/data}$ определяется как отношение суммарного интерфейса памяти оставшихся стадий к суммарному интерфейсу блочной памяти I_{ram}.

Для определения коэффициента редукции по интерфейсу памяти используется следующий алгоритм:

1. Выбрать подмножество стадий конвейера такое, что сумма ресурсов H_i^{FF} выбранных стадий будет меньше H_{ff}^{FF} и сумма выбранных значений H_i^{FF} будут максимальны.
2. Рассчитать ресурс памяти, реализуемый на блочной памяти/памяти с последовательным доступом:

$$H_{ram}^{FF} = H^{FF} - \sum H_i^{FF}, \tag{6}$$

   где i принадлежит подмножеству стадий конвейера, выбранных на шаге 1.
3. Рассчитать коэффициент $R_{ram/data}$:

$$R_{ram/data} = int(H_{ram}^{FF}/I_{ram}) + 1. \tag{7}$$

Рассмотрим в качестве примера вычисление 4-точечного быстрого преобразования Фурье (БПФ). В качестве входных типов данных зададим 16-ти битовый, знаковый.

Информационный граф после преобразования в HDL-граф и приведения в ярусно-параллельную форму показан на рисунке 1.

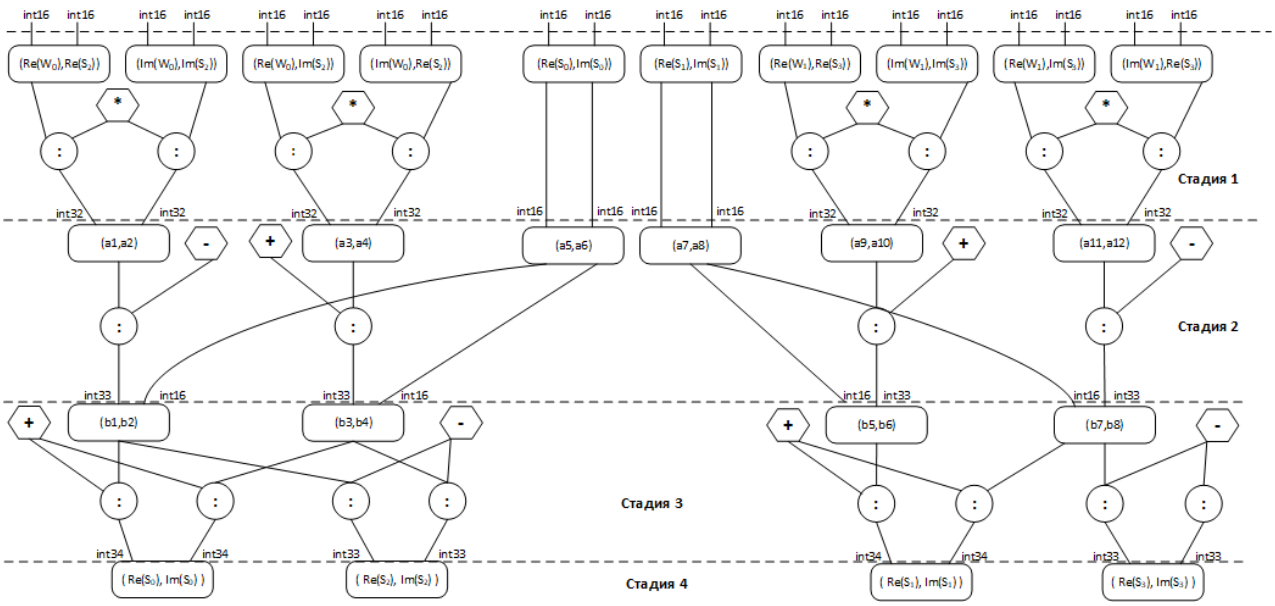

Fig. 1. HDL-graph max-parallel form of FFT size 4

Рис. 1. HDL-граф максимально-параллельной формы БПФ 4

Рассчитаем значение H_i^{FF} для каждой стадии конвейера:

H_1^{FF} = 6*2*16 = 192,

H_2^{FF} = 4*16 + 8*32 = 320,

H_3^{FF} = 4*16 + 4*33 = 196,

H_4^{FF} = 4*33 + 4*34 = 268.

Общее значение ресурса H^{FF} составит:

$$H^{FF} = \sum H_i^{FF} = 976 \text{ бит}. \tag{8}$$

Рассмотрим две архитектуры: A1 и A2. Значение $H_{ram/ff}$ для обеих архитектур равно 1536 бит. В архитектуре A1 весь ресурс памяти находится в регистрах, для такой архитектуры схема БПФ 4 в максимально-параллельной форме реализуется без изменений, как показано на рисунке 1.

В архитектуре A2 ресурс регистров составляет 512 бит и 1024 бит в виде блочной памяти с интерфейсом данных 36 бит. Значение суммарного интерфейса блочной памяти I_{ram} составит 36 бит. В соответствии с вышеприведенным алгоритмом выберем подмножество стадий, реализуемых на регистрах и имеющих максимальный интерфейс. В данном примере это может быть либо стадия 1, либо стадия 2.

Значение H_{ram}^{FF} в таком случае будет равно

$H_{ram}^{FF} = H^{FF} - H^{FF1} = 976 - 320 = 656$ бита.

Значение коэффициента редукции по интерфейсу памяти составит

$R_{ram/data} = H_{ram}^{FF}/I_{ram} = 656/36 = 19$.

В этом случае период подачи данных становится равным $R_{ram/data}$ и стадии конвейера 2,3,4 или 1,3,4 реализуются последовательно, так как результат записывается в один блок памяти.

HDL-граф после редукции на $R_{ram/data}$ представлен на рисунке 2. Запись значений a1-a12, вычисление значений b1-b8, S0-S3 производится в данном случае последовательно. Так как стадии 2,3 исходной схемы после редукции требуют 22 такта на выполнение, стадия 1 также может быть увеличена до 22 тактов без влияния на общую производительность системы, что приведет к пропорциональному уменьшению вычислительного ресурса стадии 1.

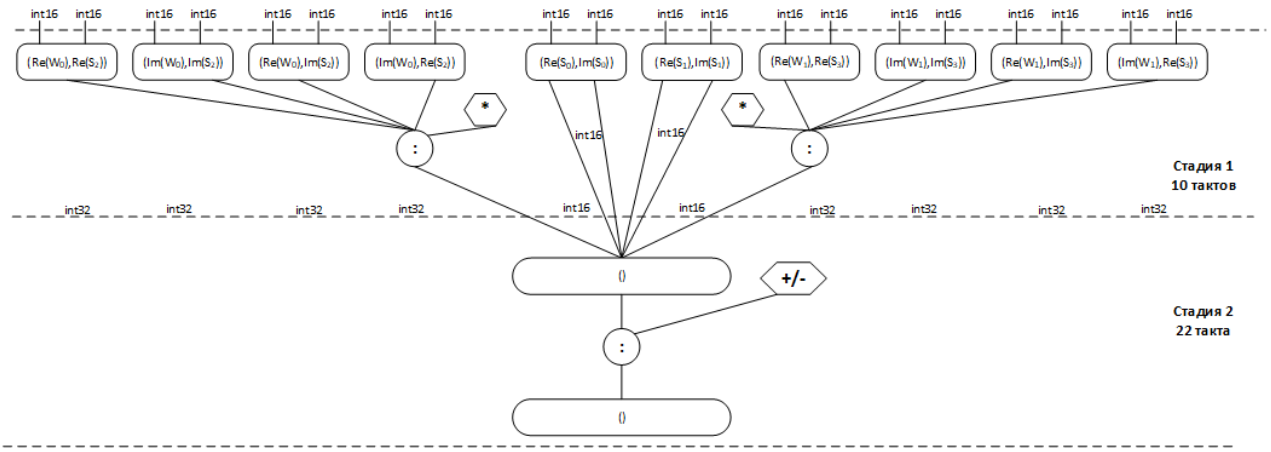

Fig. 2. HDL-graph after reduction

Рис. 2. HDL-граф после редукции

Для оценки вычислительных ресурсов рассмотрим слои (стадии) HDL-графа. Обозначим общее количество слоев (стадий конвейера) – M. На каждом j-том слое графа реализуется определенное множество операций $O^j = \{O_1^j, O_2^j \ldots O_i^j\}$. Для всего HDL-графа количество каждой операции типа i – N_i равно

$$N_i = \sum_{j=0}^{M} O_i^j. \quad (9)$$

Обозначим общее количество различных типов операций L, i=0. . . L. Под типом операции понимается тип арифметической/логической и т. д. операции вместе с указанием разрядностей данных, например, сложение 16 битных данных, сравнение 20 битных данных и т. д.

После расчета общего количества операций каждого типа необходимо исходя из доступного ресурса конкретной архитектуры оценить с какой степенью параллелизма возможно реализовать схему.

Методы и способы оценки ресурса для каждого конкретного типа операции рассмотрим далее, на данном этапе считаем, что количество ресурса для каждого конкретного типа операции известно.

Обозначим за $\dotplus$ – тип ресурса (логические ячейки, специализированные арифметические блоки DSP и т. д.), U_i^T – количество ресурса типа $\dotplus$, необходимое для реализации операции типа i. Тогда суммарный ресурс типа T для всех операций в HDL графе равен

$$U^T = \sum_{i=0}^{L} U_i^T * N_i. \tag{10}$$

По каждому классу вычислительных ресурсов можно рассчитать коэффициент редукции R_T как отношение суммарного требуемого ресурса к ресурсу целевой архитектуры, округленное до ближайшего большего целого:

$$R^T = int(U^T/H_T). \tag{11}$$

Итоговый коэффициент редукции по вычислительным ресурсам определяется как максимальный среди всех R_T:

$$R_{\text{выч}} = max\{R_T\}. \tag{12}$$

Рассмотрим данный подход на примере схемы графа изображенной на рисунке 1. Общее количество стадий конвейера данного графа M=4. Количество типов операций L составит 5: 16-битное умножение (i=0), сложение и вычитание 33 и 16 бит (i=1,2), сложение 34-битных данных и вычитание 33-битных данных(i=3,4). Количество операций по слоям графа составит:

$O^1 = 8, 0, 0, 0, 0,$

$O^2 = 0, 2, 2, 0, 0,$

$O^3 = 0, 0, 0, 4, 4,$

$O^4 = 0, 0, 0, 0, 0.$

Количество каждой операции i-го типа:

$N_0 = 8, N_1 = 2, N_2 = 2, N_3 = 4, N_4 = 4.$

В качестве примера возьмем архитектуру ПЛИС с двумя типами ресурсов для реализации вычислений: логические ячейки ($T = 0$) и DSP($T = 1$). Значения ресурса для каждого типа операции возьмем следующие:

$U_0^0 = 0, U_0^1 = 1$

$U_1^0 = 5, U_1^1 = 0,$

$U_2^0 = 5, U_2^1 = 0,$

$U_3^0 = 10, U_3^1 = 0,$

$U_4^0 = 10, U_4^1 = 0.$

Тогда, суммарный ресурс по каждому типу по всем операциям в графе составит:

$U^0 = \sum_{i=0}^{L} U_i^0 * N_i = 0 * 8 + 5 * 2 + 5 * 2 + 4 * 10 + 4 * 10 = 100,$

$U^1 = \sum_{i=0}^{L} U_i^1 * N_i = 1 * 8 + 0 * 2 + 0 * 2 + 0 * 10 + 0 * 10 = 8.$

Возьмем архитектуру, где количество DSP $H_1 = 2$, количество логических ячеек $H_0 = 400$. Тогда коэффициент редукции по каждому типу ресурсов составит:

$R_0 = int(U^0/H_0) = 100/400 = 0.25,$

$R_1 = int(U^1/H_1) = 8/2 = 4.$

Значение коэффициента редукции для вычислительных ресурсов $R_{\vee\cong\Xi}$ равно

$R_{\vee\cong\Xi} = max\{R_T\} = max\{0.25, 4\} = 4.$

После редукции каждой стадии с коэффициентом 4 задержка каждой стадии вырастет до 4 тактов, ресурс при этом снизится также в 4 раза. Результирующая схема приведена на рисунке 3. Необходимо также учитывать, что при изменении вычислительного ресурса может измениться и ресурс памяти.

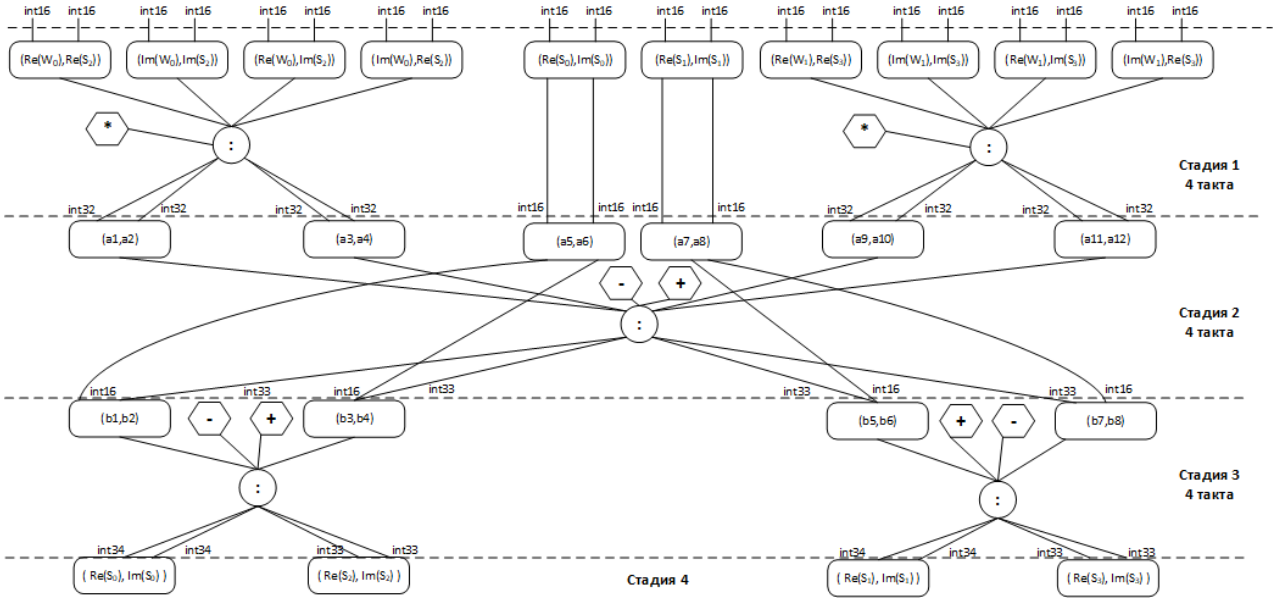

Fig. 3. HDL-graph after reduction with coefficient 4

Рис. 3. HDL-граф после редукции с коэффициентом редукции 4

3. Алгоритм преобразования параллелизма

Рассматривая задачу преобразования параллелизма в общем виде можно выделить 3 варианта преобразования исходной максимально-параллельной схемы в зависимости от доступного ресурса целевой платформы. Под доступным ресурсом целевой архитектуры при этом понимается наименьший из ресурсов – памяти или вычислительного ресурса.

Обозначим отношение доступного на платформе ресурса к ресурсу, требуемому для реализации в исходном максимально-параллельном виде, как P.

Возможные варианты отношения следующие:

- $P < 0.5$, возможна индукция (увеличение) количества схем;
- $0.5 < P < 1$, ресурса достаточно для размещения 1 максимально-параллельного варианта схемы;
- $P > 1$, необходима редукция параллелизма.

Первый вариант не требует преобразований максимально-параллельной схемы, при этом возможно увеличение производительности путем размещения нескольких схем параллельно:

$S = int(1/P)$.

Второй вариант схемы также не требует преобразований. Для третьего варианта схемы рассмотрим алгоритм с использованием коэффициентов редукции, описанных в выше.

Алгоритм редукции состоит из следующих шагов:

1. Рассчитать коэффициенты редукции R_{ram} и $R_{\text{выч}}$;
2. Выбрать максимальный коэффициент из R_{ram} и $R_{\text{выч}}$ - R_{max};
3. Редуцировать параллелизм схемы на R_{max};
4. Для измененной схемы пересчитать коэффициенты R_{ram} и $R_{\text{выч}}$, если они меньше 1, то следует завершение алгоритма;
5. Если какие-либо операции возможно реализовать с использованием другого типа ресурсов, то изменить данные операции на другой тип ресурсов без изменения коэффициента R и пересчитать коэффициенты R_{ram} и $R_{\text{выч}}$;
6. Если какой-либо из коэффициентов больше 1, то увеличить $R_{max} = R_{max} + 1$ и возврат на шаг 3.

На шаге 6 последовательное увеличение коэффициента редукции позволяет подобрать минимально-возможный коэффициент для удовлетворения требований по ресурсам и достижения при этом максимальной производительности (минимально-возможной редукции по производительности относительно исходного максимально-параллельного варианта).

4. Обмен метриками с архитектурно-зависимым уровнем

При оценке вычислительного ресурса необходимо иметь оценку ресурсоемкости операций, выделяемых в информационном HDL-графе. Данная оценка является архитектурно-зависимой, поскольку одна и та же операция на разных архитектурах занимает разный ресурс. Кроме этого набор базовых типов ресурсов, по которым выполняется оценка, так же может различаться на разных архитектурах.

Для описания архитектуры, требуемой при решении задачи оценки ресурсов, формируется модель, состоящая из 3 основных секций: секция описания типов ресурсов, секция описания общей емкости ресурсов платформы и секция описания ресурсоемкости базовых операций.

Для описания архитектуры, требуемой при решении задачи оценки ресурсов, формируется модель, состоящая из 3 основных секций: секция описания типов ресурсов, секция описания общей емкости ресурсов платформы и секция описания ресурсоемкости базовых операций. Секция описания содержит записи по количеству типов ресурсов. Каждая запись содержит общее количество данного типа ресурсов на конкретной целевой платформе. Описание ресурсоемкости операции содержит тип операции, количество операндов, разрядности операндов и количество по каждому типу ресурсов, необходимое для реализации данной операции с заданной разрядностью.

Рассмотрим в качестве примера описания ресурса платформу ПЛИС. Ресурсами памяти на платформе ПЛИС являются блоки памяти и регистры. Для вычислительных ресурсов это логические ячейки и арифметические блоки DSP. Пример описания типов ресурсов для платформы Xilinx Ultrascale приведен в таблице 1.

Table 1. Xilinx Ultrascale Platform Resource Types

Таблица 1. Типы ресурсов платформы Xilinx Ultrascale

Тип	Класс	Ед. измерения	Объем, бит	Размер типа данных, бит
LUT	Выч.	шт.	-	1
FF	Память	бит	1	1
BRAM	Память	бит	36864	36
DSP	Выч.	шт.	-	25

Описание общего количества ресурса для конкретной целевой платформы приведено в таблице 2.

Table 2. Resources for Virtex® UltraScale ™ FPGAs XCVU065

Таблица 2. Объем ресурсов для Virtex® UltraScale™ FPGAs XCVU065

Тип	Количество, шт
LUT	358080
FF	716160
BRAM	1260
DSP	600

Для операций, используемых в схеме на рисунке 1, секция описания ресурсоемкости приведена в таблице 3.

Как видно из таблицы 3, некоторые операции могут быть реализованы на разных типах вычислительных ресурсов, что необходимо учитывать в алгоритме редукции параллелизма, поскольку это расширяет количество вариантов реализации схемы без изменения коэффициента редукции R.

Table 3. Resource requirement for Virtex® UltraScale ™ XCVU065 FPGA

Таблица 3. Требуемый ресурс для ПЛИС Virtex® UltraScale™ XCVU065

Тип операции	Количество операндов	Разрядность операндов	LUT	FF	BRAM	DSP
+	2	33/33	33	0	0	0
-	2	33/33	33	0	0	0
+	2	34/34	34	0	0	0
-	2	34/34	34	0	0	0
+	2	32/16	32	0	0	0
-	2	32/16	32	0	0	0
*	2	16/16	0	0	0	1
+	2	33/33	0	0	0	1
-	2	33/33	0	0	0	1
+	2	34/34	0	0	0	1
-	2	34/34	0	0	0	1
+	2	32/16	0	0	0	1
-	2	32/16	0	0	0	1

5. Практические результаты

Для оценки полученных результатов, приведенные на рисунках 1–3 схемы были синтезированы с помощью входящего в разработанный пакет прикладных программ синтезатора HDL-описания с языка ФПП [14]. Для оценки ресурсов использована платформа Xilinx Ultrascale+. Синтез полученного HDL-описания производился с использованием САПР Vivado 2019.1. [15]. Оценка ресурсов производилась по 4 классам: логические ячейки (LUT), регистры (FF), память (BRAM) и арифметические блоки (DSP).

В качестве базы для сравнения взят максимально-параллельный вариант, приведенный на рисунке 1. Результаты синтеза и оценки ресурсов для базы приведены в таблице 4. Так как для синтеза арифметических операций на данной платформе существуют два варианта – использование LUT и использование DSP, в данной реализации использован сбалансированный вариант – синтез операций умножения с использованием DSP блоков и синтез операций сложения с использованием LUT.

Table 4. Estimating resources of case with maximum parallelism

Таблица 4. Оценка ресурсов базового варианта с максимальным параллелизмом

Схема	LUT	FF	BRAM	DSP
Максимально-параллельная схема	392	980	0	8
Редукция по памяти/регистрам	457	332	1	1
Редукция по вычислительным ресурсам	398	889	0	2

Результаты синтеза схемы, редуцированной по интерфейсу памяти/регистров, приведенной на рисунке 2, отражены в таблице 4. Полученный ресурс соответствует требованиям архитектуры A2.

Из таблицы 4 видно, что ресурс логических ячеек увеличился по сравнению с вариантом «до редукции». Это связано с накладными расходами, требуемыми для реализации схемы сериализации/мультиплексирования расчетов на меньшем количестве блоков. Для конечной реализации данный ресурс требует оценки на высокоуровневом этапе. Проблема разницы в оценке ресурсов на высокоуровневом и низкоуровневом этапе требует дальнейшего изучения.

Результаты синтеза схемы, редуцированной по вычислительным ресурсам (рисунок 3), приведены в таблице 4. Результат редукции по вычислительным ресурсам DSP ячеек соответствует требованиям, но при этом количество логических ячеек не изменилось при редукции. Этот эффект

также связан с накладными расходами при переходе к более последовательной схеме и требует учета при оценке ресурсов и дальнейшем выборе коэффициента редукции в п.4 вышеприведенного алгоритма.

Заключение

Представленные методы редукции позволяют реализовать изменение параллелизма исходного описания алгоритма и обеспечить реализацию механизма переноса на различные архитектуры с разными ресурсными ограничениями. В отличии от методов индукции параллелизма предложенный метод снижает сложность процесса переноса за счет уменьшения перебора количества вариантов реализации, получаемых в процессе синтеза под новые ресурсные ограничения.

Вместе с тем методы оценки ресурса на высокоуровневом этапе требуют учета накладных расходов при изменении параллелизма к более последовательным схемам. Текущий вариант оценки требует увеличения точности оценки. Для этого могут использоваться нейронные сети и методы машинного обучения, которые на основе параметров оценки схемы на высокоуровневом этапе могут с необходимой точностью предсказать значения параметров схемы после реализации на низком уровне.

Методы прямого подсчета ресурса результата реализации схемы осложнены ввиду множества преобразований, которые происходят при реализации схемы на этапах синтеза и имплементации на низкоуровневом этапе предлагаемого метода. Реализация точных методов оценки ресурса позволит дополнительно уменьшить количество вариантов схемы, рассматриваемых в процессе синтеза под ресурсные ограничения.

References

[1] T. M. Bhatt and D. McCain, “Matlab as a Development Environment for FPGA Design”, in *Design Automation Conference, Proceedings 42nd*, 2005, pp. 607–610.

[2] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer, *Electronic Design Automation for IC System Design, Verification, and Testing*. CRC Press, 2017.

[3] Z. Navabi, *System-Level Design and Modeling: ESL Using C/C++, SystemC and TLM-2.0*. Springer, 2015.

[4] *Vivado Design Suite User Guide. High-Level Synthesis. UG902.* [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug902-vivado-high-level-synthesis.pdf.

[5] Z. Yuan, Y. Ma, J. Bian, and K. Zhao, “Automatic enhanced CDFG generation based on runtime instrumentation”, in *IEEE 17th International Conference on*, 2013, pp. 92–97.

[6] G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Hérault, and J. J. Dongarra, “PaRSEC: Exploiting Heterogeneity to Enhance Scalability”, *IEEE Computing in Science and Engineering*, vol. 15, no. 6, pp. 36–45, 2013.

[7] A. Danalis, G. Bosilca, A. Bouteiller, T. Herault, and J. Dongarra, “PTG: An Abstraction for Unhindered Parallelism”, in *Proceedings of the Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing*, 2014, pp. 21–30.

[8] I. I. Levin and A. I. Dordopulo, “Resource-independent programming of hybrid reconfigurable computer systems”, in *Proceedings of Russian Supercomputing Days*, 2017, pp. 714–723.

[9] I. I. Levin, A. I. Dordopulo, I. V. Pisarenko, and A. K. Melnikov, “An approach to architecture-independent programming of computing systems based on the aspect-oriented Set@l language”, *Proceedings of the Southern Federal University. Technical sciences*, vol. 197, no. 3, pp. 46–57, 2018.

[10] O. V. Nepomnyaschy, I. N. Ryzhenko, and A. I. Legalov, "The method of architecturally independent high-level synthesis of VLSI", *Proceedings of the Southern Federal University. Technical sciences*, vol. 202, no. 8, pp. 38–47, 2018.

[11] A. I. Legalov, "Functional language for creation of architectural independent parallel programmes", *Computational Technologies*, vol. 10, no. 1, pp. 71–89, 2005.

[12] A. I. Legalov, V. S. Vasilyev, I. V. Matkovskii, and M. S. Ushakova, "A toolkit for the development of data-driven functional parallel programmes", in *International Conference on Parallel Computational Technologies, Springer*, 2018, pp. 16–30.

[13] V. Vasilev, A. I. Legalov, and S. V. Zykov, "The System for Transforming the Code of Dataflow Programs into Imperative", *Modeling and Analysis of Information Systems*, vol. 28, no. 2, pp. 198–214, 2021.

[14] O. V. Nepomnyaschy and I. N. Ryzhenko, "The method of high-level synthesis and software toolkit for description algorithm of VLSI", *Software Engineering*, vol. 11, no. 1, pp. 34–39, 2020.

[15] *Vivado Design Suite User Guide. Using the Vivado IDE UG893.* [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug893-vivado-ide.pdf.