14. *Casbeer D.W.* Forest fire monitoring with multiple small UAVs, *Proceedings of the 2005 American Control Conference*, 2005, pp. 3530-3535.
15. *Spry S.C., Girard A.R., Hedrick J.K.* Convoy Protection using Multiple Unmanned Aerial Vehicles: Organization and Coordination, *Proc. of the 24th American Control Conference, Portland, OR., June 2005*.
16. *Tonetti S., Hehn M., Lupashin S., D'Andrea R.* Distributed control of antenna array with formation of UAVs, *In World Congress*, 2011, August, Vol. 18, No. 1, pp. 7848-7853.
17. *Chung J.* Cooperative Control of UAVs Using a Single Master Subsystem for Multi-task Multi-target Operations, *Advances in Intelligent Systems and Computing*, 2015, Vol. 345, pp. 193-212.
18. *Korneev V.V., Kiselev A.V.* Sovremennye mikroprotsessory [Modern microprocessor]. Saint Petersburg: BHV-SPb. 2003, 448 p.
19. *King A.* Distributed Parallel Symbolic Execution. B.S., Kansas State University, 2005.
20. *Hwang K., Briggs F.A.* Computer Architecture and Parallel Processing, 1984, pp. 32-40.

Статью рекомендовал к опубликованию д.т.н., профессор Д.Б. Борзов.

Титенко Евгений Анатольевич – Юго-Западный государственный университет; e-mail: johntit@mail.ru; 305040 г. Курск, ул. 50 лет Октября, 94; тел.: +79051588904; кафедра информационных систем и технологий; к.т.н.; доцент.

Крипачев Александр Владимирович – e-mail: xray-007@mail.ru; зав. отделом компьютерных технологий научной библиотеки.

Марухленко Анатолий Леонидович – e-mail: proxy33@mail.ru; кафедра информационной безопасности; к.т.н.; доцент.

Titenko Evgeny Anatolievich – South-West State University; e-mail: johntit@mail.ru; 94, 50 years of October street, Kursk, 305040, Russia; phone: +79051588904; the department of IT; cand. of eng. sc.; associate professor.

Kripachev Alexander Vladimirovich – e-mail: xray-007@mail.ru; head of the department of computer technologies of the scientific library.

Marukhlenko Anatoly Leonidovich – e-mail: proxy33@mail.ru; the department of information security; cand. of eng. sc.; associate professor.

УДК 004.272.44 DOI 10.23683/2311-3103-2018-8-38-47

О.В. Непомнящий, И.Н. Рыженко, А.И. Легалов

МЕТОД АРХИТЕКТУРНО-НЕЗАВИСИМОГО ВЫСОКОУРОВНЕВОГО СИНТЕЗА СБИС

Одним из актуальных направлений развития технологий проектирования сверхбольших интегральных схем и вычислительных систем на их основе является высокоуровневый синтез. При описании проекта на верхних уровнях закладываются концепции общесистемного взгляда на организацию всего процесса проектирования. Поэтому на первый план выходит развитие маршрутов и технологий, базирующихся на принципах высокоуровневого, архитектурно независимого проектирования, позволяющих осуществлять формирование комплексного подхода к организации всех фаз создания проекта. Требуется создание методов эффективной выработки архитектурных решений для однокристальных систем параллельной обработки информационных потоков, не зависящих от конечной формы реализации. Необходима разработка инструментальных средств, обеспечивающих эффективный перенос архитектурно-независимого, высокоуровневого описания решаемых прикладных задач на целевую платформу. Авторами предложен новый метод синтеза проекта. Метод базируется на функционально-потоковой парадигме параллельных вычислений, это позволяет осуществлять архитектурно-независимую разработку алгоритмов функционирования СБИС. Предложена модель вычислений, использующая ряд промежуточных

структур, а именно управляющий, информационный и HDL-графы. Определены требования к языку функционально – потокового параллельного программирования и, с учетом специфики решаемых задач, выбран ФП - язык параллельного программирования «Пифагор». На основании разработанного перечня требований выполнена доработка языка, введены статические типы данных, исключен ряд функций и задержанные вычисления, при формировании списка повторением введены ограничения. Приведено описание ключевых моментов в семантике языка, принципов преобразования параллелизма и формирования промежуточных представлений при переходе к целевой платформе. Разработан маршрут и алгоритм высокоуровневого синтеза. Выделены основные требования к созданию архитектурно-независимых инструментальных средств, реализован программный инструментарий и выполнен ряд тестовых проектов.

Параллельные вычисления; поток данных; функциональное программирование; СБИС; высокоуровневый синтез; архитектурно-независимое параллельное программирование.

O.V. Nepomnyaschy, I.N. Ryzhenko, A.I. Legalov

THE METHOD OF ARCHITECTURALLY INDEPENDENT HIGH-LEVEL SYNTHESIS OF VLSI

The problem of high-level design of complex functional circuits and systems intended for implementation in the form of VLSI is considered. The basic shortcomings of existing approaches are revealed and a conceptually new method of project synthesis is proposed. The method is based on the functional-streaming paradigm of parallel computing, it allows for implementation of an architecture-independent development of algorithms for VLSI operation. A computation model is proposed that uses a number of intermediate structures, namely, control, information, and HDL graphs. The requirements for the language of functional-streaming programming and, taking into account the specifics of the tasks to be solved, the language is selected and modified. The description of the key points in the semantics of the language, the principles of parallelism transformation and the formation of intermediate representations in the transition to the target platform are given. The route and algorithm of high-level synthesis has been developed. The basic requirements for the creation of architecture-independent tools are identified, software tools are implemented, and a number of test projects are executed.

Parallel computing; data flow; functional programming; VLSI; high-level synthesis, Architecture-Independent Parallel Programming

Введение. В настоящее время проектирование сложных однокристальных систем становится все более трудоемким в связи с постоянным увеличением размера разрабатываемых схем. Увеличивается разрыв между числом логических элементов, которые могут быть размещены на одном кристалле, и числом элементов, которые могут быть реально спроектированы и верифицированы в необходимые сроки. Для сложных проектов на первый план выходят процессы, связанные с тестированием разработанных решений. Сегодня в маршруте проектирования тестирование занимает 60-80% от общего времени разработки [1]. Решение означенных проблем лежит в области высокоуровневых подходов к синтезу СБИС. Этот факт подтверждается мировыми тенденциями в развитии современных технологий проектирования, в которых наблюдается переход на более высокие уровни абстракции в процессе разработки.

Однако существующие высокоуровневые подходы к синтезу СБИС [2–5] не в полной мере обеспечивают решение всех проблем, поскольку, в первую очередь они подразумевают использование моделей вычислений, зачастую плохо подходящих для описания схемы СБИС, представляющую собой параллельную схему обработки потоков данных. Например, в технологии HLS [5] – синтеза, предлагаемой компанией Xilinx, применяется описание исходного алгоритма функционирования схемы на императивном Си-подобном языке программирования, что приводит к проблемам при переходе от исходного описания к реализации на СБИС. Та-

кой переход выполняется либо в автоматическом режиме средствами САПР [6], либо полуавтоматически с использованием директив, заданных программистом. Автоматический переход с выделением параллелизма неэффективен и сложен, полуавтоматическое распараллеливание требует больших временных затрат и высокой квалификации разработчика. Кроме того, при существующих методах высокоуровневого синтеза СБИС используется явное управление вычислениями, то есть, схема управления вычислениями изначально синтезируется под конкретную платформу. Это затрудняет и делает малоэффективным перенос проекта на различные платформы СБИС, поскольку такой подход является архитектурно-зависимым.

Метод высокоуровневого синтеза. Сформулируем основные требования к модели вычислений, необходимой для эффективных преобразований исходных алгоритмов в СБИС. Вычисления на платформе цифровых СБИС представляют собой параллельную схему обработки потоков данных (граф потока данных). Поэтому необходимо применять такую модель вычислений, которая позволяла бы описывать исходный алгоритм с учетом параллелизма и при этом максимально соответствовала модели вычислений на аппаратном уровне. Кроме того, модель вычислений должна поддерживать неявную поддержку параллелизма, что позволит отказаться от использования ручного или автоматического распараллеливания исходного алгоритма и обеспечит переносимость между платформами.

Одним из подходов, направленным на реализацию этой идеи, является использование моделей вычислений непосредственно задающих программу в виде графа потока данных [7, 8]. В частности данных подход рассматривается в ряде работ, например в работах [9, 10] применительно к программированию для универсальных параллельных вычислительных систем. Также существует концепция ресурсно-независимого параллелизма применительно к реконфигурируемым вычислительным системам [10, 11]. Для эффективного программирования гибридных реконфигурируемых вычислительных систем авторы предлагают использовать язык программирования высокого уровня COLAMO, позволяющий описывать различные формы организации параллельных вычислений.

На основании результатов анализа известных подходов выделены основные требования к методу высокоуровневого синтеза СБИС. Метод должен использовать:

- модель вычисления на основе графа потока данных;
- отсутствие явного управления вычислениями;
- описание параллелизма на уровне элементарных операций;
- принцип неограниченности ресурсов для вычислений;
- сжатие параллелизма при переносе на реальные ресурсно-ограниченные платформы, вместо распараллеливания последовательных фрагментов;
- статическую типизацию, при которой типы данных определяются на этапе привязки алгоритмов к конкретной реализации.

Анализ полученных требований позволил выбрать функционально-потоковую модель параллельных вычислений, как наиболее полно соответствующую означенным положениям. Данная модель легла в основу языка функционально-потокового параллельного (ФПП) программирования Пифагор, который обеспечивает представление программы в виде графа потока данных [12]. Вершины графа представляют собой операторы обработки данных, а дуги – пути передачи информации. Вычисления происходят по готовности данных на входе, при этом вычислительные ресурсы считаются неограниченными. Параллелизм ограничен только информационными зависимостями и алгоритмом решаемой задачи.

Преобразование параллелизма. Для обеспечения архитектурной независимости исходное описание алгоритма на языке функционально-потокового параллельного программирования Пифагор описывается с максимальным параллелизмом и не содержит явного управления вычислениями. При реализации алгоритма необходимо преобразование параллелизма под ресурсные ограничения архитектуры целевой вычислительной системы и используемую в этой архитектуре стратегию управления вычислениями. Так как исходное описание алгоритма содержит только информационные связи и описано с максимальным параллелизмом на уровне операций, задача преобразования параллелизма при переходе к целевой системе сводится к сжатию параллелизма. Как показано в [13], характер обработки данных (параллелизм), можно условно разделить на 4 типа:

1) последовательный;
2) параллельно-независимый;
3) конвейерный;
4) параллельно-зависимый.

Для описания параллелизма в языке ФПП Пифагор используются 2 конструкции: параллельный и асинхронный список [12]. Так как алгоритм описывается с максимальным параллелизмом, присущим конкретной задаче, то характер параллелизма и обработки данных реализуется путем преобразований из указанных конструкций. Вариант параллельно-зависимой обработки невозможен, так как в функционально-потоковой модели отсутствуют конфликты по данным. Варианты преобразований из последовательной обработки в параллельно-независимую и конвейерную невозможны для данного подхода, так как описание алгоритма предполагает максимальный параллелизм и отсутствие необходимости в формировании параллелизма на основе преобразования последовательных участков. Рассмотрим остальные варианты преобразований параллелизма подробнее.

Преобразование параллельно-независимой обработки в последовательную и конвейерную. Параллельно-независимая обработка описывается в языке с использованием параллельного списка, который обозначается квадратными скобками [12]. Каждый элемент такого списка может обрабатываться одновременно с другими элементами. Например, функция F над параллельным списком может быть преобразована в одновременное вычисление всех элементов списка путем применения этой функции:

$$[x1,x2,x3]:F \Rightarrow [x1:F, x2:F, x3:F].$$

При этом параллельным списком может быть как набор данных, так и набор функций. То есть возможны ситуации, когда над одним аргументом могут одновременно выполняться несколько функций:

$$X:[f1, f2, f3] \Rightarrow [X:f1, X:f2, X:f3].$$

Рассмотрим варианты преобразования параллельного списка длины N. Исходный информационный граф (ИГ) функции F над параллельным списком данных представлен на рис. 1. Здесь вершина №1 – параллельный список данных, №2 – вершина интерпретации (обработки данных).

Помимо информационного графа строится управляющий граф, который формирует сигналы готовности для вершин информационного графа. В схеме с неограниченными ресурсами и управлением по готовности данных обработка всего списка произойдет параллельно за один квант времени – на вершину обработки данных из управляющего графа будет сформировано N сигналов готовности данных. Для перехода к последовательной обработке достаточно уменьшить кратность дуги готовности данных из управляющего графа в вершину интерпретации до 1 – каждый элемент параллельного списка будет обрабатываться по одному за

такт. Результирующий сигнал готовности в вершину, следующую за вершиной интерпретации, сформируется по окончании обработки всего набора данных из параллельного списка.

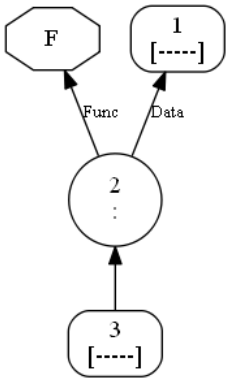

Рис. 1. Информационный граф параллельно-независимой схемы

Переход к конвейерной обработке происходит также путем преобразования управляющего графа и получением производного информационного графа в ярусно-параллельной форме. Для параллельного списка данных длины N можно сформировать конвейерную схему с длиной конвейера от 1 (параллельно-независимая схема) до N. Число сигналов готовности (одновременно вычисляемых функций F на стадии конвейера) в вершину интерпретации при этом определяется как N/l, где l – длина конвейера, 0< l <=N. При переходе к ЯПФ также необходимо добавлять стадии задержки сигналов данных от ранних стадий конвейера. На рис. 2 приведен пример преобразования ИГ параллельной обработки списка размерности N=6 к конвейерной схеме с длиной конвейера l=3 и вычислением на каждой стадии конвейера 2 функций F.

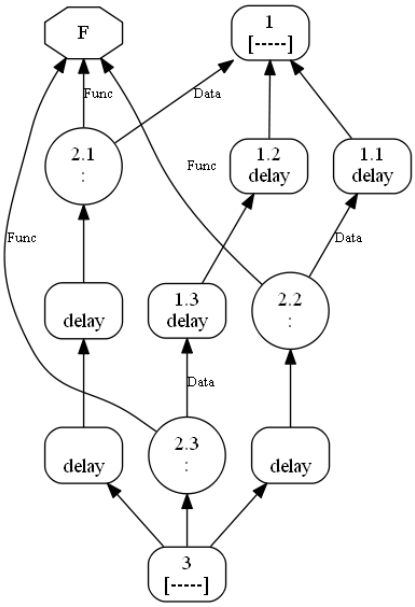

Рис. 2. Пример информационного графа конвейерной схемы

Преобразование конвейерной обработки в параллельно-независимую и последовательную. Аналогичным образом возможно и преобразование конвейерной формы в остальные формы. Конвейерную форму вычислений можно либо получить из параллельно-независимой, как описано выше, так и описать с помощью асинхронного списка. Преобразования, полученные из параллельно-независимой формы, могут быть проведены в обратную сторону. Возможности преобразования асинхронного списка описаны в [15, 16].

Отметим, что в силу специфики платформы СБИС некоторые конструкции языка ФПП программирования при синтезе не поддерживаются. Среди них деление и деление по модулю, а так же задержанные вычисления (в силу их бесполезности на СБИС). Оператор формирования списка повторением поддерживается с ограничениями – так как он может формировать данные динамически изменяемой размерности, что неприменимо для СБИС.

Маршрут проектирования. Разработанный на основании предложенного подхода маршрут проектирования СБИС представлен на рис. 3.

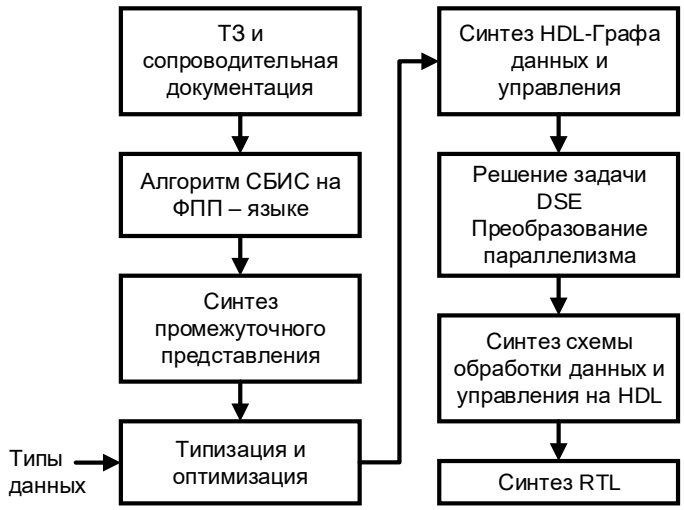

Рис. 3. Маршрут проектирования

Первый этап – формирование перечня спецификаций к проекту на основе технического задания, не отличается от существующих подходов. Далее происходит разработка и отладка алгоритма функционирования схемы СБИС на языке функционально-потокового параллельного программирования Пифагор. Разработанное описание схемы транслируется в промежуточное представление – информационный граф (ИГ). На базе информационного графа строится вторая часть промежуточного представления – управляющий граф (УГ). Данное представление является форматом обмена между транслятором с ФПП языка и синтезатором схемы для СБИС. Этот формат представления отчасти напоминает используемые в существующих высокоуровневых подходах DFG и CFG графы [13]. На этапе типизации и оптимизации происходит привязка к конкретной целевой платформе с использованием подключаемого файла ограничений и типов. В языке Пифагор используется динамическая типизация, поэтому при переходе к платформе СБИС осуществляется явное задание статических типов данных через файл ограничений, что позволяет получить из одного и того же описания алгоритма различные варианты реализации под разные размерности данных.

В процессе синтеза формируется внутреннее представление в виде HDL-графа. На следующем этапе путем преобразования параллелизма происходит решение задачи оптимального выделения ресурсов и планирования схемы [17, 18]. Результатом этого этапа является внутреннее представление, из которого синтезируется описание схемы на языке описания аппаратуры HDL. Далее, используя синтезаторы с HDL языков из состава САПР требуемого производителя, получают описание схемы в RTL виде. Начиная с данного этапа, применяются действующие маршруты проектирования СБИС.

Алгоритм высокоуровневого синтеза. При синтезе HDL описания применяются несколько базовых принципов и ограничений:

- одна вершина операции интерпретации соответствует одной стадии конвейера;
- вершины аргумента и результата формируют начальную и конечную стадии конвейера функции;
- одна функция ФПП языка формирует один HDL модуль;
- вызов функции формирует экземпляр модуля HDL.

Алгоритм синтеза описания схемы СБИС из описания алгоритма на ФПП языке приведен на рис. 4.

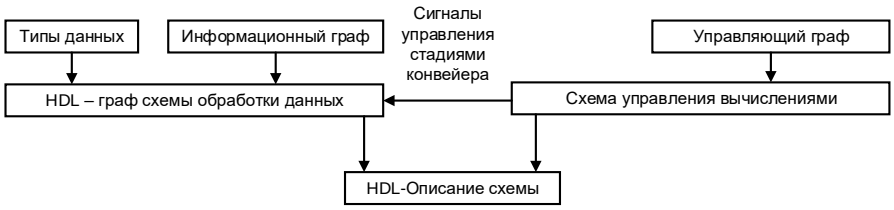

Рис. 4. Процесс синтеза

Входом является ФПП программа. Первый этап заключается в трансляции ФПП программы в промежуточные представление, задаваемое информационным и управляющим графами [19]. После этого на вход синтезатору HDL подается промежуточное представление и типизированный формат аргумента.

Синтезатор преобразует программу во внутреннее промежуточное представление в виде HDL графа. Задачей данного этапа является оптимизация информационного графа за счет возможного вычисления и преобразования некоторых вершин при известных типах данных и определение входных и выходных типов данных для каждой вершины информационного графа.

Далее на основе сформированного HDL-графа синтезируется описание схемы на HDL языке. Синтез заключается в преобразовании вершин HDL графа в соответствующие конструкции HDL языка, описывающие элементы СБИС. Результат каждой операции в вершине интерпретации записывается в соответствующий выходной регистр вершины интерпретации, сгенерированный на этапе синтеза таблицы регистров. Изначально синтезируется схема обработки данных, в которой для разрешения срабатывания каждой стадии конвейера формируется сигнал разрешения, который в дальнейшем используется для подключения управляющих сигналов от схемы управления, формируемой по УГ. Затем формируется схема генерации управляющих сигналов по УГ и осуществляется подключение этих сигналов к управляющим входам схемы обработки данных. При формировании схемы управления вычислениями происходит преобразование (редукция) параллелизма под конкретные ресурсные ограничения платформы.

Для поддержки процесса высокоуровневого синтеза СБИС разработан комплект инструментальных средств разработки ФПП программ [16], позволяющих:

♦ преобразовать исходный текст с языка ФПП программирования в промежуточные представления в виде информационного и управляющего графа;

♦ осуществлять оптимизационные преобразования информационного и управляющего графов, что позволяет повысить эффективность ФПП программ;

♦ проводить отладку и анализ функционально-потоковых параллельных программ до их преобразования в машинно-зависимые формы, обеспечивая тем самым поиск ошибок и трассировку.

Заключение. Предложенный подход использует описание схем на базе языка с неявным заданием параллелизма, что позволяет при их проектировании повысить эффективность за счет исключения этапа выделения параллелизма. Разработаны алгоритмы и маршрут синтеза для преобразования высокоуровневого архитектурно-независимого программного описания цифровых схем на языке параллельного программирования в низкоуровневое архитектурно-зависимое представление на языках описания аппаратуры, позволяющее осуществлять реализацию системы на целевой платформе [20].

Разработаны алгоритмы и инструментальные средства поддержки проектирования СБИС с реконфигурируемой архитектурой, что позволяет ускорить процесс разработки и повысить возможности переноса высокоуровневых исходных алгоритмов на топологию разрабатываемых конфигурируемых систем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Непомнящий О.В., Алекминский С.Ю.* Проблемы верификации проекта при сквозном проектировании вычислительных систем на кристалле // Нано- и микросистемная техника. – 2010. – № 9 (122). – С. 4-7.
2. Киберлинка. – URL: https://cyberleninka.ru/article/n/formalnaya-verifikatsiya-pri-proektirovanii-sverhbolshih-integralnyh-shem*.
3. *Dennis McCain, Tejas M. Bhatt.* Matlab as a Development Environment for FPGA Design // Design Automation Conference, Proceedings 42nd – 2005.
4. *Bailey B., Martin G., Piziali A.* ESL Design and Verification // ESL Des. Verif. – 2007. – P. 113-138.
5. *Bhasker J.* A SystemC Primer. – 2nd ed. – Star Galaxy Publishing, 2004.
6. Vivado Design Suite User Guide. High-Level Synthesis. UG902 – Xilinx – 2015. – Режим доступа: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_4/ug902-vivado-high-level-synthesis.pdf , свободный (дата обращения 08.05.2018).
7. *Ejnioui A., Namballa R., Ranganathan N.* Control and data flow graph extraction for high-level synthesis // VLSI Proceedings. IEEE Computer society Annual Symposium on. – 2004. – P. 187-192.
8. *Alves C., Ferreira P., Ferreira C.* Erlang inspired Hardware // International Conference on Field Programmable Logic and Applications, Milano – 2010.

a. *Mycroft A., Sharp R.* Hardware/software co-design using functional languages // Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '01). – 2001. – P. 236-251.

9. *Dongarra J., Bosilca G., Bouteiller A., Danalis A., Faverge M., Herault T.* PaRSEC: A programming paradigm exploiting heterogeneity for enhancing scalability // IEEE Computing in Science and Engineering. – 2013. – Issue #6. – Vol. 15. – P. 36-45.
10. *Danalis A., Bosilca G., Bouteiller A., Herault T., Dongarra J.* PTG: An Abstraction for Unhindered Parallelism // Proceedings of the Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing. – 2014. – P. 21-30.
11. *Дордопуло А.И., Левин И.И.* Ресурсонезависимое программирование гибридных реконфигурируемых вычислительных систем // Суперкомпьютерные дни в России: Труды международной конференции (25–26 сентября 2017 г., г. Москва). – М.: Изд-во МГУ. – 2017. – С. 714-723.

12. *Легалов А.И.* Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. – 2005. – № 1 (10). – С. 71-89.
13. *Bian Jinian, Wu Quang, Wang Yunfeng, Wu Weimin.* A hierarchical CDFG as Intermediate Representation for Hardware/Software Codesign // Communications, Circuits and systems and West Sino Expositions, IEEE 2002 International Conference. – 2002. – Vol. 2. – P. 1429-1432.
14. *Левин И.И., Дордопуло А.И., Писаренко И.В., Мельников А.К.* Подход к архитектурно-независимому программированию вычислительных систем на основе аспектно-ориентированного языка SET@L // Известия ЮФУ. Технические науки. – 2018. – № 3 (197). – С. 46-58.
15. *Легалов А.И., Редькин А.В., Матковский И.В.* Функционально-потоковое параллельное программирование при асинхронно поступающих данных // Труды международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2009)». Нижний Новгород, 30 марта – 3 апреля 2009 г.). – Челябинск: Изд-во ЮУрГУ, 2009. – С. 573-578.
16. *Легалов А.И.* Использование асинхронно поступающих данных в потоковой модели вычислений // Третья сибирская школа-семинар по параллельным вычислениям: Сб. докладов. – Томск. Изд-во Томского ун-та, 2006. – С. 113-120.
17. *Narasimhan M., Ramanujam J.* A fast approach to computing exact solutions to the resource-constrained scheduling problem // ACM TODAYES. – 2001. – Vol. 6 (4). – P. 490-500.
18. *Amellal S., Kaminska B.* Scheduling of a control and data flow graph // IEEE Int. Symp. on Circuits and Systems. – 1993. – Vol. 3. – P. 1666-1669.
19. *Легалов А.И., Васильев В.С., Матковский И.В., Ушакова М.С.* Инструментальная поддержка создания и трансформации функционально-потоковых параллельных программ. // Труды ИСП РАН. – 2017. – Т. 29. – Вып. 5. – С. 165-184.
20. *Непомнящий О.В., Легалов А.И., Рыженко И.Н.* Технология архитектурно-независимого, высокоуровневого синтеза сверхбольших интегральных схем // Доклады Академии наук высшей школы Российской Федерации. – 2014. – № 1 (22). – С. 93-103.

REFERENCES

1. *Nepomnyashchiy O.V., Alekminskiy S.Yu.* Problemy verifikatsii proekta pri skvoznom proektirovanii vychislitel'nykh sistem na kristalle [Problems of verification projects computational system on chip], *Nano- i mikrosistemnaya tekhnika* [Nano- and Microsystem technology], 2010, No. 9 (122), pp. 4-7.
2. Kiberlinka [CyberLink]. Available at: https://cyberleninka.ru/article/n/formalnaya-verifikatsiya-pri-proektirovanii-sverhbolshih-integralnyh-shem*.
3. *Dennis McCain, Tejas M. Bhatt.* Matlab as a Development Environment for FPGA Design, *Design Automation Conference, Proceedings 42nd – 2005*.
4. *Bailey B., Martin G., Piziali A.* ESL Design and Verification, *ESL Des. Verif.*, 2007, pp. 113-138.
5. *Bhasker J.* A SystemC Primer. 2nd ed. Star Galaxy Publishing, 2004.
6. Vivado Design Suite User Guide. High-Level Synthesis. UG902 – Xilinx – 2015. Available at: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_4/ug902-vivado-high-level-synthesis.pdf , свободный (accessed 08 May 2018).
7. *Ejnioui A., Namballa R., Ranganathan N.* Control and data flow graph extraction for high-level synthesis, *VLSI Proceedings. IEEE Computer society Annual Symposium on*, 2004, pp. 187-192.
8. *Alves C., Ferreira P., Ferreira C.* Erlang inspired Hardware, *International Conference on Field Programmable Logic and Applications, Milano – 2010.*
9. *Mycroft A., Sharp R.* Hardware/software co-design using functional languages, *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '01)*, 2001, pp. 236-251.
10. *Dongarra J., Bosilca G., Bouteiller A., Danalis A., Faverge M., Herault T.* PaRSEC: A programming paradigm exploiting heterogeneity for enhancing scalability, *IEEE Computing in Science and Engineering*, 2013, Issue 6, Vol. 15, pp. 36-45.
11. *Danalis A., Bosilca G., Bouteiller A., Herault T., Dongarra J.* PTG: An Abstraction for Unhindered Parallelism, *Proceedings of the Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing*, 2014, pp. 21-30.

12. *Legalov A.I.* Funktsional'nyy yazyk dlya sozdaniya arkhitekturno-nezavisimykh parallel'nykh programm [Functional language for creation of architecture-independent parallel programs], *Vychislitel'nye tekhnologii* [Computation Technologies], 2005, No. 1 (10), pp. 71-89.
13. *Bian Jinian, Wu Quang, Wang Yunfeng, Wu Weimin.* A hierarchical CDFG as Intermediate Representation for Hardware/Software Codesign, *Communications, Circuits and systems and West Sino Expositions, IEEE 2002 International Conference*, 2002, Vol. 2, pp. 1429-1432.
14. *Levin I.I., Dordopulo A.I., Pisarenko I.V., Mel'nikov A.K.* Podkhod k arkhitekturno-nezavisimomu programmirovaniyu vychislitel'nykh sistem na osnove aspektno-orientirovannogo yazyka SET@L [Approach to architecture-independent programming of computer systems in aspect-oriented SET@L language], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2018, No. 3 (197), pp. 46-58.
15. *Legalov A.I., Red'kin A.V., Matkovskiy I.V.* Funktsional'no-potokovoe parallel'noe programmirovanie pri asinkhronno postupayushchikh dannykh [Functional-stream parallel programming with asynchronously incoming data], *Trudy mezhdunarodnoy nauchnoy konferentsii «Parallel'nye vychislitel'nye tekhnologii (PaVT'2009)»* [Proceedings of the international scientific conference "Parallel computing technologies (Pavt'2009)"]. Nizhniy Novgorod, 30 marta – 3 aprelya 2009 g.). Chelyabinsk: Izd-vo YuUrGU, 2009, pp. 573-578.
16. *Legalov A.I.* Ispol'zovanie asinkhronno postupayushchikh dannykh v potokovoy modeli vychisleniy [Using asynchronous data in functional data-flow model calculations], *Tret'ya sibirskaya shkola-seminar po parallel'nym vychisleniyam: Cb. dokladov* [The third Siberian school-seminar on parallel computing: Collection of reports]. Tomsk. Izd-vo Tomskogo un-ta, 2006, pp. 113-120.
17. *Narasimhan M., Ramanujam J.* A fast approach to computing exact solutions to the resource-constrained scheduling problem, *ACM TODAYES*, 2001, Vol. 6 (4), pp. 490-500.
18. *Amellal S., Kaminska B.* Scheduling of a control and data flow graph, *IEEE Int. Symp. on Circuits and Systems*, 1993, Vol. 3, pp. 1666-1669.
19. *Legalov A.I., Vasil'ev V.S., Matkovskiy I.V., Ushakova M.S.* Instrumental'naya podderzhka sozdaniya i transformatsii funktsional'no-potokovykh parallel'nykh program [Instrumental support of creating and transformation functional-dataflow parallel programs], *Trudy ISP RAN* [ISP RAN], 2017, Vol. 29, Issue 5, pp. 165-184.
20. *Nepomnyashchiy O.V., Legalov A.I., Ryzhenko I.N.* Tekhnologiya arkhitekturno-nezavisimogo, vysokourovnevogo sinteza sverkhbol'shikh integral'nykh skhem [Technology of architecture-independent, high-level synthesis of ultra-large integrated circuits], *Doklady Akademii nauk vysshey shkoly Rossiyskoy Federatsii* [Reports of the Academy of Sciences of the higher school of the Russian Federation], 2014, No. 1 (22), pp. 93-103.

Статью рекомендовал к опубликованию д.т.н., профессор А.И. Водяхо.

Непомнящий Олег Владимирович – Сибирский федеральный университет; e-mail: 2955005@gmail.com; 660032, г. Красноярск, ул. Белинского, 1-155; тел.: +79048955005; зав. кафедрой вычислительной техники; к.т.н.; доцент.

Легалов Александр Иванович – e-mail: legalov@mail.ru; тел.: +79138305976; д.т.н.; профессор.

Рыженко Игорь Николаевич – АО КБ “Искра”, e-mail: rodgi.krs@gmail.com; г. Красноярск, ул. Борисова, 36-105; тел.: +79029208195; ведущий инженер.

Nepomnyaschy Oleg Vladimirovich – Siberian Federal University, e-mail: 2955005@gmail.com; 1-155, Belinskogo street, Krasnoyarsk, 660032, Russia; phone: +79048955005; head of the department of computer systems; cand. of eng. sc.; assistant professor.

Legalov Aleksandr Ivanovich – e-mail: legalov@mail.ru; phone: 79138305976; dr. of eng. sc.; professor.

Ryzhenko Igor Nikolaevich – JSC “R&D Iskra”; e-mail: rodgi.krs@gmail.com; 36, Borisova street, ap. 105, Krasnoyarsk, Russia; phone: +79029208195; leading engineer.