ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № ?, 2014, с. ??–??

УДК 681.3

Б. Я. Штейнберг, А. П. Баглий, Д. В. Дубров,
Ю. В. Михайлуц, А. С. Рошаль, Р. Б. Штейнберг

# Развитие проекта компилятора с высокоуровневого языка на компьютер с программируемой архитектурой

Аннотация. В настоящей работе рассматривается автоматическое отображение высокоуровневых программ на языке C на вычислительные системы с программируемой конвейерной архитектурой. Примером подобной системы может служить традиционный процессор с ПЛИС-ускорителем или соответствующая система на кристалле. Отображение реализуется на основе Оптимизирующей Распараллеливающей Системы (http://ops.rsu.ru) и конвертера с языка C в язык VHDL (C2HDL). Часть функциональности OPC и C2HDL доступна через web-интерфейс (http://ops.opsgroup.ru). В статье приводится обзор близких по тематике работ. Развитие настоящей работы ведёт к созданию компилятора языка C для программируемых конвейерных архитектур.

*Ключевые слова и фразы:* HDL, конвейер, автоматическое распараллеливание.

## Введение

Использование программируемых конвейерных вычислительных устройств увеличивается с каждым годом, а развитие инструментов разработки высокоуровневых программ для них отстаёт. В данной статье рассматривается проект компилятора с языка C на компьютер с программируемой архитектурой и работы по его реализации. Под целевым компьютером понимается либо система на кристалле, содержащая как процессорное ядро, так и матрицу конфигурируемых логических блоков [1], либо стандартный процессор с ПЛИС-ускорителем. Создание компилятора языка C на программируемые архитектуры может ускорить их применение и развитие.

Вычислители с программируемой и перепрограммируемой архитектурами разрабатываются для широкого круга приложений и показывают высокую эффективность [2, 3]. Интересными представляются, в частности, мультиконвейерные (многоконвейерные, параллельно-конвейерные) системы, которые можно рассматривать как обобщение и конвейерных систем, и многоядерных. Архитектуры мультиконвейерных систем известны по работам [4–7]. Алгоритмы автоматического отображения высокоуровневого языка на многоконвейерные (мультиконвейерные) системы рассмотрены в [8–10]. В данной работе обсуждается автоматическая генерация многоконвейерных схем и отображение на них высокоуровневых программ.

Разрабатываемый компилятор основан на Оптимизирующей распараллеливающей системе (ОРС) [11]. Компилятор включает в себя конвертор [12, 13] из внутреннего представления ОРС на язык описания электронных схем VHDL. Данный конвертор на входе имеет кроме исходного текста программы еще и диапазоны входных данных. Поэтому, такие же диапазоны должны задаваться и на входе компилятора. Следует подчеркнуть, что было бы сложно генерировать VHDL-код конвейерной системы из низкоуровневого регистрового внутреннего представления, какими являются LLVM компилятора Clang или RTL семейства компиляторов GCC. Высокоуровневым внутренним представлением (в котором производятся оптимизирующие преобразования) кроме ОРС обладает распараллеливающая система SUIF [14]. Раннее состояние рассматриваемого компилятора описывалось в [13].

## 1. Обзор состояния проблемы

В настоящее время инструменты высокоуровневого синтеза завоёвывают всё больший интерес. Основные разработчики инструментов EDA (Electronic Design Automation — автоматизация разработки электронных схем) встраивают в свои продукты конвертеры из языка высокого уровня в дизайн электронной схемы. Эти инструменты нацелены либо на отдельно работающие ПЛИС/ASIC (Application Specific Integration Circuits — микросхемы заказной логики), либо на гибридные системы с ПЛИС [15].

Можно выделить два главных подхода к построению инструментов высокоуровневого синтеза. Первый подразумевает использование традиционного языка программирования, такого как C и C++,

для выражения реализации алгоритма в аппаратуре. Системы Catapult C, Vivado Design Suite, Impulse CoDeveloper, Altium Designer и HDL Coder for MATLAB представляют собой примеры коммерческих продуктов, использующих данный подход. Академические исследовательские проекты включают C-to-Verilog [16], TCE [17] и Parallel Intellectual Compiler [18]. Иногда средств высокоуровневых языков, используемых в данных системах, недостаточно для представления реконфигурируемых электронных схем. Например, Trident Compiler [19] требует от пользователя разбиения вручную кода на программную и аппаратную части.

Другие системы, а именно Mitrion C [20], Handel-C [21] и HaSCoL [22] используют специально разработанные языковые конструкции для представления аппаратных абстракций, отсутствующих в традиционных языках программирования: манипулирования отдельными битами, параллельно выполняющихся процессов с обменом данными, синхронизаций и т. д. Это заставляет пользователя переписывать старый код и также делает эти языки более низкоуровневыми. Тем не менее, эти языки имеют преимущества над языками HDL: многие рутинные задачи, требующие много времени, могут быть выполнены автоматически (например, генерирование конвейерных схем). С другой стороны, инструменты, использующие традиционные языки, стараются применять более сложный анализ программного кода для получения из него недостающей информации. Например, обобщение графа потока данных (data flow graph) под названием «граф потока битов» (bit-flow graph) может помочь в генерировании эффективных аппаратных реализаций алгоритмов, представленных при помощи битовых операций языка C (сдвиг, обращение, выделение битов и т. д.) [23].

## 2. Структура компилятора с языка C на программируемую вычислительную архитектуру

Задача компиляции исходного кода на языке C в программу, выполняющуюся на компьютере с программируемой архитектурой, состоит из четырёх подзадач, каждой из которых будет заниматься отдельный модуль компилятора (рис. 1).

- Преобразователь C в VHDL (C2HDL). Конвейеризация циклов и преобразование части кода исходной программы в описание вычислительного ядра на языке VHDL.

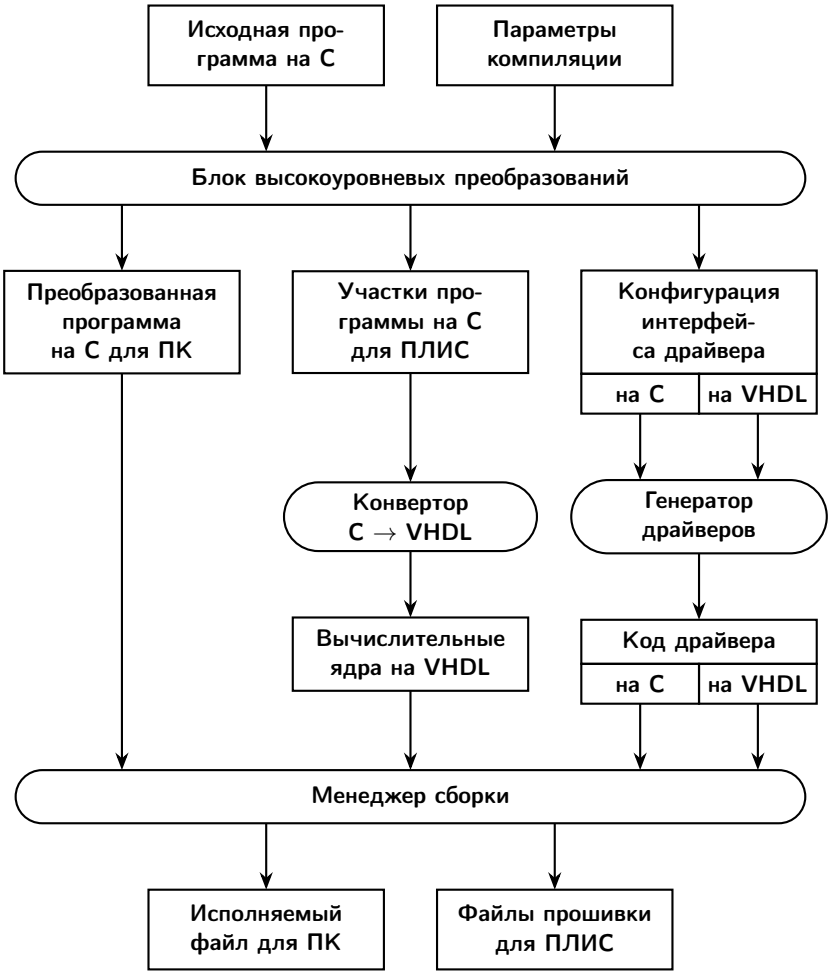


Рис. 1. Структура компилятора с языка C на программируемую вычислительную архитектуру

- Генератор драйверов. Автоматическая генерация на C и VHDL вспомогательных модулей, обеспечивающих:
  - Передачу данных и управляющих команд между центральным процессором и вычислительными ядрами на ПЛИС по выбранному протоколу.

  - Синхронизацию обмена данными и оптимальное распределение пропускной способности канала передачи данных.
  - Синхронизацию вычислительных потоков на ПЛИС с основным потоком на центральном процессоре.
- Менеджер сборки. Из всех сгенерированных файлов собирает два проекта на C и на VHDL соответственно, с описанием параметров компиляции каждого проекта. Также управляет процессом компиляции исходного кода в исполняемые файлы целевых платформ.

На вход компилятор принимает исходный код программы и параметры компиляции. Результатом являются преобразованная программа на языке C и описание вычислительного ядра на VHDL. Далее программа на C может быть откомпилирована в исполняемый файл любой целевой платформы, для которой имеется компилятор языка C. Для компиляции VHDL-кода в файл прошивки конкретной ПЛИС также нужен компилятор VHDL для соответствующей модели ПЛИС.

На выходе могут быть два варианта:

(1) Без установленных компиляторов целевых платформ
   (a) Код преобразованной программы на C (вместе с кодом драйвера);
   (b) Проект на VHDL с кодом вычислительного ядра и кодом драйвера.
(2) При наличии установленных компиляторов для целевых платформ
   (a) Исполняемый файл для ПК;
   (b) Файл прошивки для ПЛИС.

## 3. Отображение программ на программируемую архитектуру

Ускорение следует применять к долго считаемым фрагментам программ. В данной работе будем рассматривать ускорение фрагментов кода, содержащих гнезда программных циклов. Рассмотрим гнездо из $n$ вложенных циклов:

```
for (I1 = L1; I1 <= R1; ++ I1)
  for (I2 = L2; I2 <= R2; ++ I2)
    ...
      for (In = Ln; In <= Rn; ++ In)
      {
        LOOPBODY(I1, I2, ..., In);
```

```
}
```

Конвейеризуется самое глубоко вложенное гнездо циклов. Счётчики более высоко вложенных циклов могут рассматриваться как параметры, определяющие узел кластера, на ускорителе которого должен выполняться самый глубоко вложенный цикл (этот же цикл для разных значений счетчиков внешних циклов выполняется на разных узлах). Здесь используются:

- методы отображения гнёзд циклов на многоконвейерную (параллельно-конвейерную) архитектуру, которые описаны в работах [8–10] и частично реализованы в системе ОРС;
- преобразование (в ОРС) «фрагмент в программу», которое выделяет совокупность зависимых по данным и по управлению операторов и переменных в самостоятельную программу, которая эквивалентна исходному фрагменту на эквивалентных входных данных;
- конвертор C2HDL, который по полученному коду настраивает программируемую часть процессора.

## 4. Конвертор C2HDL и генерация многоконвейерной системы

На данный момент конвертор C2HDL преобразует некоторое узкое множество входных программ на языке C содержащими только один цикл и жёсткими ограничениями в VHDL описание конвейерной схемы.

При развитии проекта предполагается возможность генерации нескольких конвейеров (для гнезда вложенных циклов), работа которых при необходимости может быть синхронизирована специальными задержками между стартами конвейеров. При расчёте таких задержек используется анализ информационных зависимостей между точками пространства итераций. Такие информационные зависимости описываются решётчатыми графами, которые хранятся в памяти в виде функций [24–34].

Пример 1. Рассмотрим фрагмент программы, состоящий из двух вложенных циклов:

```
for (I1 = L1; I1 <= R1; ++ I1)
  for (I2 = L2; I2 <= R2; ++ I2)
  {
    X[I1][I2] = X[I1 - 1][I2] + X[I1][I2 - 1];
  }
```

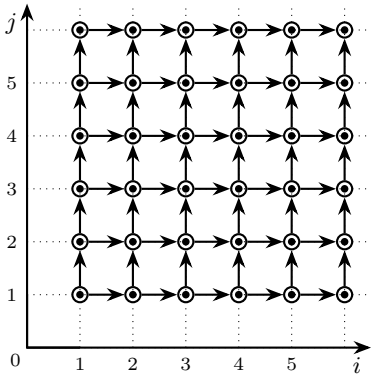


Рис. 2. Решётчатый граф программы, показывающий зависимости между точками пространства итераций. Подобные информационные зависимости возникают при некоторых сеточных методах решения задач матфизики и при выравнивании нуклеотидных последовательностей

Граф информационных зависимостей между точками пространства итераций (решётчатый граф или граф алгоритма) имеет вид, представленный на рис. 2.

Пространство итераций данного гнезда циклов можно разбить на полосы шириной в две точки. Итерации каждой такой полосы можно вычислять на двух конвейерах, причём один из конвейеров должен будет отставать от другого (рис. 3). Алгоритмы расчёта задержек в таком отставании представлены в работах [**8**–**10**].

## Заключение

Результаты работ данного проекта направлены на существенное упрощение доступа к параллельно-конвейерным системам, в результате чего:

(1) должен расшириться класс прикладных задач для этой области вычислений;
(2) должно расшириться множество пользователей;
(3) должно сократиться время разработки параллельно-конвейерных программ;
(4) должно стимулироваться развитие процессоров систем на кристалле с программируемой архитектурой.

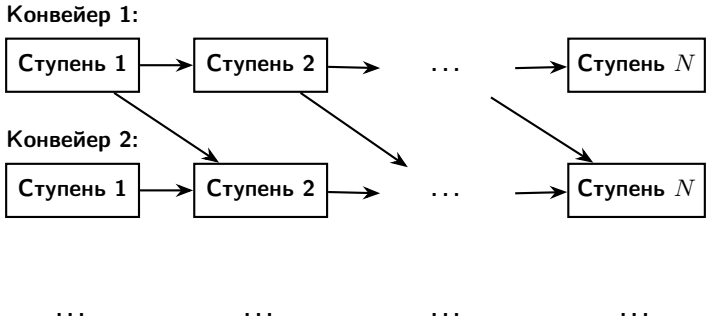


Рис. 3. Организация вычислений с передачей информации между конвейерами

## Список литературы

[1] *Zynq-7000 All Programmable SoC Overview. Preliminary Product Specification / XILINX*, http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf (Дата обращения: 3.11.2014). ↑1

[2] K. Bondalapati. *Modeling and Mapping for Dynamically Reconfigurable Hybrid Architecture*, Ph.D. Thesis, University of Southern California, (2001). x + 184 p. (english) ↑2

[3] А. В. Каляев, И. И. Левин. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. М.: «Янус-К», 2003. — 380 с. (russian) ↑2

[4] В. В. Корнев. Архитектура вычислительных систем с программируемой структурой. Новосибирск: Наука, 1985. — 166 с. (russian) ↑2

[5] А. В. Каляев, И. И. Левин, Е. А. Семерников, В. И. Шмойлов. Реконфигурируемые мультиконвейерные вычислительные структуры / ред. И. А. Каляев. Изд. 2-е, перераб. и доп. Ростов-н/Д: Издательство ЮНЦ РАН, 2009. — 344 с. (russian) ↑2

[6] М. С. Яджак. *Высокопараллельные алгоритмы и методы для решения задач массовых арифметических и логических вычислений*, Диссертация на соискание учёной степени д. ф.-м. н., Институт прикладных проблем механики и математики, Львов, (2009), на украинском языке. 298 с. (russian) ↑2

[7] К. Г. Самофалов, Г. М. Луцкий. Основы теории многоуровневых конвейерных вычислительных систем. М.: Радио и связь, 1989. — 272 с. (russian) ↑2

[8] Р. Б. Штейнберг. *Отображение гнёзд циклов на многоконвейерную архитектуру* // Программирование, 2010, № 3, с. 69–80 (russian). ↑2, 6, 7

[9] Р. Б. Штейнберг. *Вычисление задержки в стартах конвейеров для суперкомпьютеров со структурно процедурной организацией вычислений* // Искусственный интеллект. Научно-теоретический журнал / Институт проблем

искусственного интеллекта НАНУ. — Украина, Донецк, ДонДИШИ: «Наука и Освита», 2003, № 4, с. 105–112. (russian) ↑2, 6, 7

[10] Р. Б. Штейнберг. *Использование решётчатых графов для исследования многоконвейерной модели вычислений* // Известия вузов. Северо-кавказский регион. Естественные науки, 2009, № 2, с. 16–18 (russian). ↑2, 6, 7

[11] *Оптимизирующая распараллеливающая система*, http://www.ops.rsu.ru/about.shtml (Дата обращения: 2.11.2014). ↑2

[12] D. Dubrov, A. Roshal. *Generating Pipeline Integrated Circuits Using C2HDL Converter* // East-West Design & Test Symposium, 2013, September 2013, p. 1–4. (english) ↑2

[13] Д. В. Дубров, А. С. Рошаль, Б. Я. Штейнберг, Р. Б. Штейнберг. *Автоматическое отображение программ на процессор с ПЛИС-ускорителем* // Вестник Южно-Уральского государственного университета. Серия «Вычислительная математика и информатика», 2014. Т. **3**, № 2, с. 117–120 (russian). ↑2

[14] *Affine transformations for optimizing parallelism and locality*, http://suif.stanford.edu/research/affine.html (Дата обращения: 3.11.2014). ↑2

[15] J. M. P. Cardoso, P. C. Diniz. Compilation Techniques for Reconfigurable Architectures: Springer US, 2009. — xii + 223 p. (english) ↑2

[16] Y. Ben-Asher, N. Rotem, E. Shochat. *Finding the Best Compromise in Compiling Compound Loops to Verilog* // J. Syst. Archit., September 2010. Vol. **56**, no. 9, p. 474–486 (english). ↑3

[17] O. Esko, P. Jääskeläinen, P. Huerta, C. S. de La Lama, J. Takala, J. I. Martinez. *Customized Exposed Datapath Soft-Core Design Flow with Compiler Support* // Proceedings of the 2010 International Conference on Field Programmable Logic and Applications. FPL '10. — Washington, DC, USA: IEEE Computer Society, 2010, p. 217–222. (english) ↑3

[18] G. A. Polyakov, V. V. Lysykh. *A formal method of functional SNS- synthesis of problem-oriented parallel-pipelined devices* // Proceedings of the National Supercomputer Forum (NSCF-2013). — Pereslavl-Zalessky, Russia, November 2013. (english) ↑3

[19] J. L. Tripp, M. B. Gokhale, K. D. Peterson. *Trident: From High-Level Language to Hardware Circuitry* // Computer, 2007. Vol. **40**, no. 3, p. 28–37 (english). ↑3

[20] V. V. Kindratenko, R. J. Brunner, A. D. Myers. *Mitrion-C Application Development on SGI Altix 350/RC100* // Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. FCCM '07. — Washington, DC, USA: IEEE Computer Society, 2007, p. 239–250. (english) ↑3

[21] R. P. Self, M. Fleury, A. C. Downton. *Design methodology for construction of asynchronous pipelines with Handel-C* // IEEE Proceedings — Software, February 2003. Vol. **150**, no. 1, p. 39–47 (english). ↑3

[22] D. Boulytchev, O. Medvedev. *Hardware description language based on message passing and implicit pipelining* // East-West Design & Test Symposium (EWDTS), 2010, September 2010, p. 438–441. (english) ↑3

[23] J. Zhang, Zh. Zhang, Sh. Zhou, M. Tan, X. Liu, X. Cheng, J. Cong. *Bit-level Optimization for High-level Synthesis and FPGA-based Acceleration* // Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field

Programmable Gate Arrays. FPGA '10. — New York, NY, USA: ACM, 2010, p. 59–68. (english) ↑3

[24] С. А. Гуда. *Операции над представлениями кусочно-квазиаффинных функций в виде деревьев* // Информатика и её применение, 2013. Т. **7**, № 1, с. 58–69 (russian). ↑6

[25] А. М. Шульженко. *Исследование информационных зависимостей программ для распараллеливающих преобразований*, Диссертация на соискание учёной степени к. т. н., Ростовский государственный университет, Ростов-н/Д, (2006). 202 с. (russian) ↑6

[26] А. М. Шульженко. *Автоматическое определение циклов ParDo в программе* // Известия вузов. Северо-кавказский регион. Естественные науки. Приложение, 2005, № 11, с. 77–87 (russian). ↑6

[27] J.-F. Collard, P. Feautrier, T. Risset. *Construction of DO loops from systems of affine constraints*: Technical report 93–15 LIP, Ecole Normale de Lyon, 1993. — 26 p. (english) ↑6

[28] P. Feautrier. *Parametric Integer Programming* // RAIRO Recherche Opérationnelle, September 1988. Vol. **22**, p. 243–268 (english). ↑6

[29] P. Feautrier. *Dataflow analysis of array and scalar references* // International Journal of Parallel Programming, 1991. Vol. **20**, no. 1, p. 23–53 (english). ↑6

[30] P. Feautrier. *Some efficient solutions to the affine scheduling problem. I. One-dimensional time* // International Journal of Parallel Programming, 1992. Vol. **21**, no. 5, p. 313–347 (english). ↑6

[31] P. Feautrier. *Some efficient solutions to the affine scheduling problem. Part II. Multidimensional time* // International Journal of Parallel Programming, 1992. Vol. **21**, no. 6, p. 389–420 (english). ↑6

[32] В. В. Воеводин. Математические модели и методы в параллельных процессах. М.: Наука, 1986. — 296 с. (russian) ↑6

[33] В. В. Воеводин. Математические основы параллельных вычислений. М.: Изд-во МГУ, 1991. — 345 с. (russian) ↑6

[34] В. В. Воеводин, Вл. В. Воеводин. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. — 608 с. (russian) ↑6

[35] *Автоматический распараллеливатель. OPC Web-интерфейс*, http://ops.opsgroup.ru/ (Дата обращения: 2.11.2014). ↑

*Об авторах:*

**Борис Яковлевич Штейнберг**

Решил проблему: (Meister E., Speck F.-O. Some multi-dimensional Wiener–Hopf equations with applications of Pure Mathematics to mechanics, Kozubnik, Poland, 12–17 September, 1977). Инициатор и руководитель проекта OPC www.ops.rsu.ru, д. т. н., зав. каф., зав. лаб. Ангстрем–ЮФУ.

*e-mail:* borsteinb@mail.ru

**Антон Павлович Баглий**

Младший научный сотрудник Академии биологии и биотехнологии Южного Федерального университета. Научные интересы связаны с автоматическими преобразованиями программ, компиляторами и технологиями программирования.

*e-mail:* taccessviolation@ya.ru

**Денис Владимирович Дубров**

Кандидат физико-математических наук, доцент кафедры информатики и вычислительного эксперимента мехмата ЮФУ.

*e-mail:* dubrov@sfedu.ru

**Юрий Вячеславович Михайлуц**

Магистрант мехмата ЮФУ. Выпускник Физического факультета ЮФУ. Защитил дипломную работу по теме: «Диэлектрическая релаксация в гетерогенных системах: поликристаллы и нанокомпозиты».

*e-mail:* aracks@mail.ru

**Александр Сергеевич Рошаль**

Разработчик конвертера с языка C в HDL на основе «Диалогового высокоуровневого оптимизирующего распараллеливателя». Создатель комплекса ПОЭМА для анализа напряжённости электромагнитного поля. Ведущий программист высоконагруженных проектов компании `http://www.tabor.ru`. Аспирант мехмата ЮФУ.

*e-mail:* teacplusplus@gmail.com

**Роман Борисович Штейнберг**

Кандидат физико-математических наук, старший преподаватель кафедры алгебры и дискретной математики мехмата ЮФУ. Начальник группы разработчиков программного обеспечения для многопроцессорных систем ОАО «Ангстрем».

*e-mail:* romanofficial@yandex.ru

*Образец ссылки на эту публикацию:*

Б. Я. Штейнберг, А. П. Баглий, Д. В. Дубров, Ю. В. Михайлуц, А. С. Рошаль, Р. Б. Штейнберг. *Развитие проекта компилятора с высокоуровневого языка на компьютер с программируемой архитектурой* // Программные системы: теория и приложения: электрон. научн. журн. 2014. Т. ??, № ?, с. ??–??.

*URL:* http://psta.psiras.ru/read/

Boris Steinberg, Anton Bugliy, Denis Dubrov, Yuriy Mikhayluts, Alexander Roshal, Roman Steinberg. *Developing a project of the compiler from a high-level language for a computer with programmable architecture.*

Abstract. This work presents automatic mapping of high-level programs in C language onto computing systems with programmable pipeline architecture. Examples of such systems include a common CPU with an FPGA accelerator and the corresponding system on a chip. The mapping implementation is based on Optimizing Parallelizing System (http://ops.rsu.ru) and a converter from C language to VHDL language (C2HDL). A part of OPS and C2HDL functionality is available via the web interface (http://ops.opsgroup.ru). The similar works review is provided. The development of the current work leads to building the C language compiler for programmable pipeline architectures (*in Russian*).

*Key Words and Phrases:* HDL, pipeline, automatic parallelizing.