УДК 004.056.53
DOI: 10.17586/0021-3454- 2017-60-10-980-985

ПРОЕКТИРОВАНИЕ МИКРОАРХИТЕКТУРЫ ВЫЧИСЛИТЕЛЕЙ НА БАЗЕ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ

А. А. Антонов

ООО „ЛМТ“, 199034, Санкт-Петербург, Россия
E-mail: 153287@niuitmo.ru

Университет ИТМО, 197101, Санкт-Петербург, Россия

Представлен метод проектирования микроархитектуры аппаратных вычислительных блоков (вычислителей) на платформах ПЛИС, ASIC или „система на кристалле“ и соответствующее инструментальное окружение (фреймворк). Метод основан на использовании иерархии „языковых IP-ядер“ (Language Intellectual Property, LIP) — узкоспециализированных языков описания аппаратуры со встроенными трансляторами, выполняющих генерацию проблемно-ориентированных вычислителей под управлением пользовательских спецификаций на этих языках. LIP-ядра представляет собой промежуточное решение между традиционными „аппаратными“ IP-ядрами, с зафиксированной аппаратной структурой, параметризируемой средствами языка описания аппаратуры, и полноценными самостоятельными трансляторами с собственными гибкими языками и автономной компиляторной инфраструктурой. По сравнению с указанными подходами использование LIP-ядер позволяет зафиксировать в процессе разработки вычислительных блоков удачные микроархитектурные решения с возможностью дальнейшей реализации на их основе произвольной пользовательской функциональности. Метод и фреймворк в виде прототипа САПР демонстрируются на примере LIP-ядра, реализующего механизм конвейеризации, и построенного на его базе учебного процессорного ядра с архитектурой DLX.

Ключевые слова: *встроенные системы, система на кристалле, микроархитектура, САПР, RTL, HLS, проблемно-ориентированный язык*

Введение. Применение разрабатываемых под приложение интегральных схем (Application-Specific Integrated Circuit, ASIC), глубоко реконфигурируемых вычислительных архитектур с многоуровневым параллелизмом, таких как FPGA, или гибридных (процессор+логика) „систем на кристалле“ делает разработку аппаратных вычислителей, наряду с традиционным программированием, неотъемлемой частью процесса проектирования встроенных вычислительных систем [1, 2].

К сожалению, проблема развития методов и инструментов для повышения уровня абстракции и эффективности проектирования вычислительной аппаратуры на сегодняшний день далека от разрешения. В научных и промышленных разработках широко представлены средства создания специализированных вычислителей в виде компиляторов процессорных ядер из описаний на „языках описания процессоров“ [3], а также инструменты для „высокоуровневого синтеза“ аппаратуры (High-Level Synthesis, HLS [4]). Указанные инструменты предоставляют проектировщику определенный, фиксированный набор абстракций проектирования (система команд процессора с описанием его обобщенной структуры, подмножество языка C с расширениями для управления микроархитектурой вычислителей, “guarded atomic actions”+”scheduling attributes” и т.п.), в рамках которых „вычислитель“ моделируется на выбранном уровне абстракции. Наравне с этим формируется набор запрограммированных преобразований — „вычислительных механизмов“, в соответствии с которыми исходная высокоуровневая модель транслируется в аппаратный вычислительный блок. Однако при несоответствии „модели исполнения“ (execution model) или возможностей транслятора исходного

языка ожидаемому результату использование этого способа высокоуровневого проектирования вычислителей оказывается неэффективным или невозможным. Так, HLS на базе C успешно применяются в проектировании вычислителей для потоковой обработки, однако их использование относительно несложного конвейерного CPU ведет к проблемам в автоматическом разрешении зависимостей по данным и управлению [5].

В целом, отставание методов использования вычислительных механизмов от развития самих этих механизмов (например, микроархитектур вычислителей, реализующих заданный абстрактный — для аппаратуры — алгоритм в виде цифровой электронной схемы) ведет к тому, что проектирование выполняется на нескольких уровнях параллельно и независимо, опираясь на имеющиеся на этих уровнях эффективные средства трансляции. Причем этот разрыв закономерно возрастает с повышением актуальных уровней проектирования. На практике это проявляется в появлении и развитии гибридных маршрутов проектирования (использование HLS и RTL в одном проекте, гибридные „системы на кристалле“ с четким разделением „программной“ и „аппаратной“ частей и пр.).

Но даже наличие разноуровневых средств и гибридных маршрутов проектирования оставляет высокой гранулярность формально специфицированных технических решений, особенно при использовании закрытых интегрированных вычислительных платформ (готовой аппаратуры, компиляторов и пр.) [6]. При этом для разработчика затруднено фиксирование особенных элементов микроархитектуры или полноценных микроархитектурных шаблонов „в отрыве“ от особенностей функциональности конкретного проекта. Сложность состоит в том, что для каждого такого микроархитектурного шаблона необходимо принять сложные пользовательские структуры данных и подпрограммы со специализированными управляющими конструкциями в качестве параметров и выполнить трансляцию в соответствии с реализуемым набором вычислительных механизмов. Таким образом, актуальна разработка методологических и технических средств для решения задачи поддержки пользовательских механизмов спецификации и трансляции.

Состояние области исследования. Аппаратные статически конфигурируемые структуры характеризуются высокой степенью повторного использования. Промышленные языки проектирования аппаратуры (SystemVerilog, VHDL) имеют ряд встроенных средств параметризации с различной глубиной: от настройки разрядности сигналов до процедурного генерирования логики (конструкция “generate”). К ограничениям этих средств можно отнести невозможность встроить “generate” в определенные конструкции языка и нетривиальность параметризации сложными структурами данных и подпрограммами. Соответственно приобретают актуальность альтернативные схемы организации таких объектов проектирования.

Логика внутренней организации устройства и логика формирования этой организации могут быть интегрированы путем введения пользовательских расширений для конфигурирования (обрабатываемых препроцессором) либо, наоборот, встраивания аппаратно-ориентированных абстракций в виде библиотечных компонентов в язык программирования общего назначения (Embedded Domain-Specific Languages, EDSL). Примерами последнего подхода служат SystemC, Chisel и MyHDL (встроенные с C++, Scala и Python соответственно). К недостатку этого подхода можно отнести то, что спецификации разных уровней „смешиваются“, усложняя синтаксическую и семантическую совместимость моделей исполнения новых „уровней“, порожденных новыми, специализированными языками [7].

Проблему смешения уровней можно преодолеть разделением разноуровневых спецификаций по разным модулям программы и разработкой стека самостоятельных специализированных компиляторов для соответствующих проблемно-ориентированных языков. Большинство систем HLS используют именно такую организацию. Автоматизация проектирования таких средств достигается посредством использования некоторых готовых компонентов и систем САПР (Lex/YACC для синтаксического и лексического анализа: LLVM — для преобразования

программного кода, „языковые САПР“, такие как JetBrains MPS и пр.). Препятствиями для внедрения подобного подхода в качестве неотъемлемой части маршрута проектирования вычислителей являются проблема совместимости моделей исполнения (например, для LLVM это виртуальный последовательный процессор с RISC-подобными инструкциями), а также требования к компетенциям проектировщика в области всего используемого стека технологий для синтаксического и лексического анализа, трансформаций и генерации кода.

Характеристики подходов к проектированию представлены в табл. 1. Как можно видеть, ни один из рассмотренных подходов не удовлетворяет всем приведенным требованиям, что затрудняет их внедрение в практику проектирования, поэтому актуальной является проработка альтернативных схем.

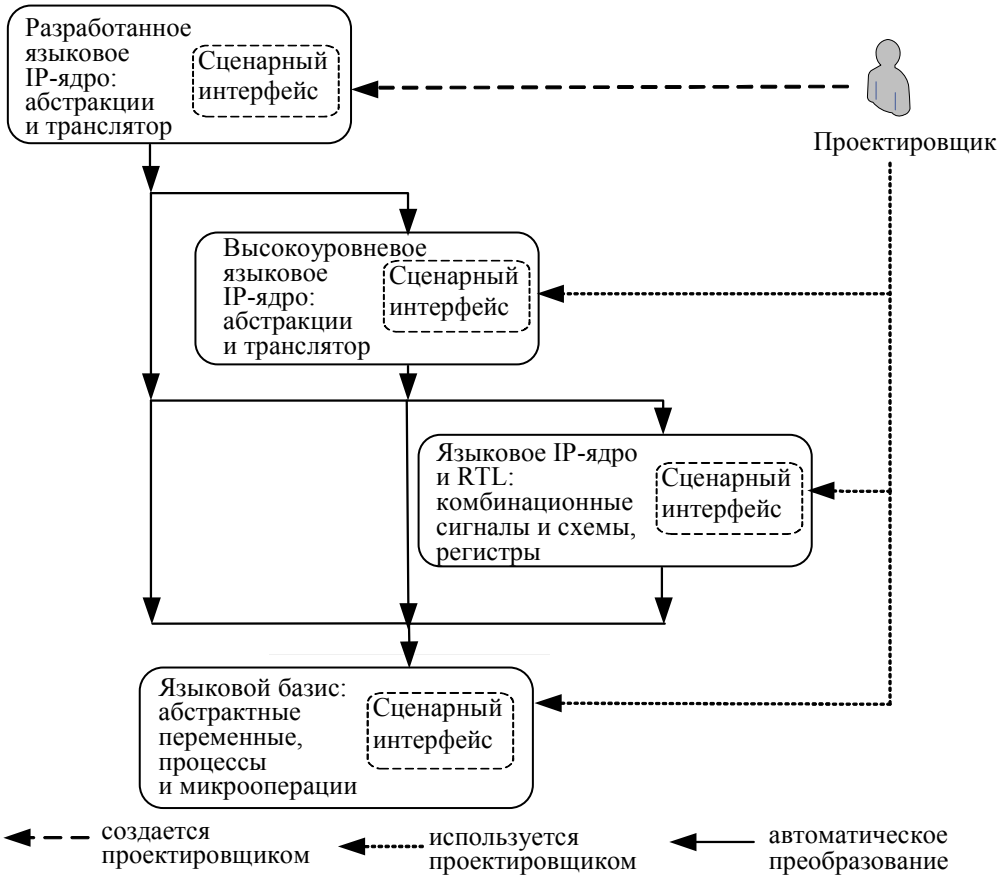
Таблица 1

Возможность	Подход			
	„Чистые“ промышленные HDL	Расширение HDL с предобработкой	„Самостоятельный“ язык и компилятор	Встроенный проблемно-ориентированный язык
Абстрагирование от описания на уровне регистровых передач	—	+	+	+
Введение произвольной пользовательской функциональности	+	+	+	+
Реализация специализированной модели выполнения	—	+	—	+
Повторное использование компиляторной инфраструктуры	Не определено	Частично	Частично	+
Процедурное порождение логики (метапрограммирования)	Частично	—	Зависит от возможностей базового языка	+
Лексико-синтаксическая „изоляция“ для элементов функциональности, реализованных в различных моделях выполнения	Не определено	—	+	—

Концепция „языковых IP-ядер“ и архитектура разработанной САПР. Для интенсификации внедрения технологий проектирования на базе аппаратных ядер с глубоко конфигурируемой пользователем функциональностью разработана архитектура проблемно-ориентированной языковой САПР (“language workbench”) на базе EDSL-подхода, ориентированной на быструю, формализованную и с прогнозируемым качеством реализацию специализированных микроархитектур вычислителей (см. рисунок).

В системе по умолчанию определяется небольшое языковое ядро, включающее в себя базовые средства для описания аппаратно-ориентированной функциональности (аналогичное ядро имеется в промышленных HDL): переменные с произвольной разрядностью и основные арифметико-логические операции (в том числе те, которые манипулируют отдельными разрядами переменных), группируемые в пользовательские процессы. Эта информация сохраняется в системе в виде набора древовидных структур данных. Однако семантика этих пользовательских процессов не зафиксирована инструментальной платформой, а определяется LIP-ядрами — подключаемыми к САПР программными компонентами с интерфейсом на базе сценарного языка. LIP-ядра выполняют поддержку новых, специфичных для микроархитектуры, опера-

ций и осуществляют трансляцию спецификаций пользовательских процессов, спроектированных с использованием этих операций, в спецификации для LIP-ядер более низкого уровня. Таким образом, LIP-ядра формируют иерархию, в основании которой находится ядро, соответствующее представлению вычислителя на уровне регистровых передач. Это представление может быть автоматически экспортировано в HDL-код для последующей аппаратной реализации с использованием промышленных инструментальных средств.



В рамках разработанного прототипа САПР логика LIP-ядер реализована в виде статической библиотеки на языке C++ с интерфейсными функциями, „обернутыми“ в текстоориентированный язык Tcl.

По сравнению с иными решениями предлагаемый метод не накладывает ограничений на способ интерпретации (семантику) пользовательских процессов и допускает реализацию различных моделей их исполнения. Пользовательский интерфейс на базе сценарного языка программирования обеспечивает инфраструктуру для синтаксического анализа исходных спецификаций, возможность метапрограммирования, в то же время обработка языковых конструкций, соответствующих различным расширениям, „разнесена“ по различным LIP-ядрам.

Недостатком предложенной схемы являются синтаксические ограничения на порождаемые языки, поскольку эти новые языки базируются на представлении вычислений в виде множества процессов со специализированными командами. Однако по совокупности свойств предложенная в работе схема на базе LIP-ядер может, наряду с традиционными фреймворками, быть неотъемлемой частью микроархитектурного проектирования специализированных аппаратных структур.

Разработка „языкового IP-ядра“ для конвейеризации и процессорного ядра на его основе. В качестве примера было разработано LIP-ядро `pipe`, реализующее механизм конвейеризации на уровне тактов синхросигнала цифровой схемы. Конвейеризация в настоящее время широко используется в ключевых компонентах вычислительных систем, включая

процессорные ядра общего назначения, ускорители, шины и сети на кристалле, компоненты иерархии памяти. Расширения языкового базиса, реализованные в этом ядре, представлены в табл. 2.

Таблица 2

Команда	Описание

`pproc clk_name rst_name`
…
`endpproc`
	Начало и завершение конвейеризованного процесса, сигналы синхронизации и перезапуска

`pstage pstage_name`	Граница стадии конвейера
`pwe data signal_name`	Запись данных во внешнюю переменную (сигнал)
`pre signal_name`	Чтение данных из внешней переменной (сигнала)
`prr pstep_name signal_name`	Чтение значения переменной из другой стадии конвейера
`pstall`	Прерывание текущей транзакции и повторный запуск всех транзакций на предыдущих стадиях конвейера
`pbreak`	Прерывание текущей транзакции
`prepeat`	Повторный запуск всех транзакций на предыдущих стадиях конвейера (без прерывания текущей транзакции)

На базе `pipe` было разработано конвейеризованное 5-ступенчатое процессорное ядро DLX, реализующее „учебное“ подмножество команд (без операций с плавающей точкой), и синтезировано для FPGA серии Cyclone V GX. Использование предложенной технологии позволяет снизить сроки проектирования процессорного ядра за счет автоматизированной генерации логики синхронизации элементов конвейера и использования языковых средств для явного разрешения конфликтов по данным и управлению: предварительная оценка показала снижение сроков на 50 %. При этом объем требуемых ресурсов FPGA составил 1240 ALM, в то время как для аналогичного ядра, спроектированного непосредственно на Verilog — 1299.

Выводы. В статье описываются разработанный метод и архитектура САПР для проектирования аппаратных вычислителей с возможностью глубокой статической конфигурации для платформ ASIC, FPGA и „система на кристалле“. По сравнению с альтернативами предложенный подход сочетает следующие свойства:

1) унифицированное представление вычислений в виде набора HDL-описаний с пользовательскими расширениями в виде специализированных команд „языковых IP-ядер“ (LIP-ядер);

2) определяемые разработчиком модели исполнения и алгоритмы трансляции высокоуровневых описаний аппаратных вычислителей;

3) применение готовой компиляторной инфраструктуры на базе широко используемых языков программирования, поддержка метапрограммирования.

Применение разработанного фреймворка показало, что предложенная технология имеет потенциал для тесной интеграции в процесс микроархитектурного проектирования аппаратных вычислителей.

СПИСОК ЛИТЕРАТУРЫ

1. *Hartenstein R., Kaiserslautern T. U., Karlsruhe K. I. T.* SE Curricula are Unqualified to Cope with the Data Avalanche. 2017. P. 1—20 [Электронный ресурс]: <http://hartenstein.de/publications/CS.pdf>.

2. *Платунов А. Е.* Реконфигурируемые встраиваемые системы и системы на кристалле // Изв. вузов. Приборостроение. 2014. Т. 57, № 4. С. 49—52.

3. *Mishra P., Dutt N.* Processor Description Languages. Morgan Kaufmann Publishers Inc., 2008.

4. *Nane R.* et al. A Survey and Evaluation of FPGA High-Level Synthesis Tools // IEEE Trans. Comput. Des. Integr. Circuits Syst. 2015. Vol. 35, N 10. P. 1591—1604.

5. *Skalicky S.* Designing Customized ISA Processors using High Level Synthesis // Intern. Conf. ReConFigurable Comput. FPGAs. 2015.

6. *Ключев А. О., Антонов А. А.* Измерение производительности компонентов подсистемы памяти для гетерогенных систем на кристалле // Программные продукты и системы. 2016. № 4. Р. 78—84.

7. *Greaves D. J.* Layering RTL, SAFL, Handel-C and Bluespec Constructs on Chisel HCL // ACM/IEEE Intern. Conf. Form. Methods Model. Codesign. 2015. P. 108—117.

Сведения об авторе

Александр Александрович Антонов — ООО „ЛМТ“; инженер; Университет ИТМО; кафедра вычислительной техники; аспирант; E-mail: 153287@niuitmo.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 03.07.17 г.

Ссылка для цитирования: *Антонов А. А.* Проектирование микроархитектуры вычислителей на базе проблемно-ориентированных языков // Изв. вузов. Приборостроение. 2017. Т. 60, № 10. С. 980—985.

DESIGN OF COMPUTER MICROARCHITECTURE BASING ON PROBLEM-ORIENTED LANGUAGES

A. A. Antonov

LMT Ltd., 199034, St. Petersburg, Russia
E-mail: 153287@niuitmo.ru

ITMO University, 197101, St. Petersburg, Russia

An original method is proposed for designing architecture of hardware computational units on FPGA, ASIC, and hybrid SoC platforms, and corresponding CAD prototype framework is developed. The method is based on hierarchy of “Language IP” (LIP) cores — specialized hardware description languages with embedded translators that implement target hardware unit generation based on input user specification. In terms of configurability, LIP cores lay between the traditional cores, which are configured by the standard means of hardware description language itself, and full standalone translators with their own specific languages and autonomous compiler infrastructure. In comparison with designing based on clear industrial HDLs or using standalone translators from high-level languages, the proposed method facilitates selective fixation of useful microarchitectural decisions with support of implementation of custom user functionality and, at the same time, does not require specific engineering qualification in the field of formal syntaxes of programming languages. The method and CAD prototype are demonstrated by the example of LIP implementing the pipeline mechanism, and a training CPU core with DLX architecture built on the base of the LIP. Advantages and shortcomings of the proposed method are evaluated, and directions of future research are formulated.

Keywords: embedded systems, system-on-chip, SoC, microarchitecture, CAD, RTL, HLS, domain-specific language

Data on author

Alexander A. Antonov — LMT Design-Center Ltd.; Engineer; ITMO University, Department of Computation Technologies; Post-Graduate Student; E-mail: 153287@niuitmo.ru

For citation: Antonov A. A. Design of computer microarchitecture basing on problem-oriented languages. *Journal of Instrument Engineering*. 2017. Vol. 60, N 10. P. 980—985 (in Russian).

DOI: 10.17586/0021-3454- 2017-60-10-980-985