5 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
COMPUTER SCIENCE

УДК 004.2

ПРИМЕНЕНИЕ HLD-МЕТОДОЛОГИИ ДЛЯ ПРОЕКТИРОВАНИЯ РЕКОНФИГУРИРУЕМЫХ ВСТРАИВАЕМЫХ СИСТЕМ

А.О. Ключев[a], П.В. Кустарев[a], Т.Т. Палташев[b, c], А.Е. Платунов[a]

[a] Университет ИТМО, Санкт-Петербург, Россия, platunov@lmt.ifmo.ru
[b] Северо-Западный политехнический университет, Фримонт, Калифорния, США
[c] АМД, Калифорния, США

Аннотация. Представлена HLD-методология проектирования встраиваемых систем, созданная и развиваемая специалистами Университета ИТМО и научно-производственной фирмы «ЛМТ». Актуальность темы обусловлена постоянным ростом архитектурной сложности реконфигурируемых встраиваемых вычислительных систем, повышением значимости вопросов системного проектирования. Показано применение HLD-методологии в ряде прикладных проектов. Ее использование повысило качество архитектурного и микроархитектурного проектирования. В основу методологии положены: система архитектурных абстракций; процесс проектирования архитектурной модели вычислительной системы, независимой от аппаратно-программной реализации; аспектная модель процесса проектирования вычислительной системы; модель актуализации вычислительного процесса на основе понятия унифицированного транслятора. Практическое применение предложенной HLD-методологии решает важные задачи проектирования. Обоснованно распределяются компоненты вычислительного процесса по различным фазам жизненного цикла системы (проектирования, исполнения), обеспечивается расширение пространства поиска проектных решений. Осуществляется синтез архитектуры на основе обобщающего взгляда на механизмы конфигурирования и программирования на базе модели актуализации вычислительного процесса. Обеспечивается возможность позднего закрепления конкретного способа реализации архитектурных решений. Применяются вертикальные архитектурные нотации. Гибко изменяются свойства встраиваемой системы посредством конфигурирования в рамках выбранного подмножества проектных аспектов. Это позволяет управлять затратами ресурсов на различных фазах ее жизненного цикла (разработки, производства, использования, поддержки). Предлагаемая HLD-методология проектирования рассматривает реконфигурируемую встраиваемую систему прежде всего через призму организации ее целевого вычислительного процесса на фазах проектирования, конфигурирования, исполнения в едином ключе. Разработчикам предоставлена возможность поиска эффективного распределения элементов вычислительного процесса по различным фазам. Методология включает в себя группы абстракций для работы с компонентами вычислительной системы и вычислительной системой в целом, с процессом проектирования встраиваемой системы и метриками архитектурных решений. В работе приведены основные положения предлагаемой авторами HLD-методологии. Демонстрируется ряд реконфигурируемых встраиваемых систем, разработанных с использованием элементов HLD-методологии.

Ключевые слова: встраиваемая система, реконфигурируемая система, вычислительная архитектура, процесс проектирования, системное проектирование, высокоуровневое проектирование.

Благодарности. Работа выполнена при государственной финансовой поддержке ведущих университетов Российской Федерации (субсидия 074-U01).

HLD-METHODOLOGY APPLICATION FOR RECONFIGURABLE EMBEDDED SYSTEMS DESIGN

A.O. Kluchev[a], P.V. Kustarev[a], T.T. Paltashev[b, c], A.E. Platunov[a]

[a] ITMO University, Saint Petersburg, Russia, platunov@lmt.ifmo.ru
[b] Northwestern Polytechnic University, Fremont, California, USA
[c] Advanced Micro Devices (AMD), California, USA

Abstract. The paper deals with HLD-methodology for embedded systems design (High Level Design Methodology for Embedded Systems), created and developed by specialists of ITMO University and "LMT" Research and Production Company. The currency of this topic is caused by constant growth of architectural complexity of reconfigurable embedded computing systems, by the importance increase of system design issues. Application of HLD-methodology in a number of applied projects is shown. Its usage has raised architectural and micro-architectural design quality. The methodology is based on: architectural abstractions system; architectural model design process of the computing system independent of hardware-software realization; aspect model of the computing system design process; actualization model of computational process on the basis of unified translator concept. Practical application of the proposed HLD-methodology solves important design problems. Computational process components are distributed reasonably on various phases of system life cycle (design, execution). Space expansion of design decisions search is provided. Architecture synthesis is implemented on the basis of a generalizing view at configuration and programming mechanisms based on computational process actualization model. Possibility of late fixing for concrete way of architectural decisions realization is provided. Vertical architectural notations are applied. Embedded system properties are flexibly changed by means of configuration within the framework of the chosen design aspects subset. It gives the possibility to control resources expenses for various phases of system life cycle (design, manufacture, usage, support). The proposed design HLD-methodology considers reconfigurable embedded system, first of all, through the prism of its target computational process organization at the design, configuration and execution phases in a unified key. De-

velopers have got possibility for effective distribution search of computational process elements on various phases. The methodology includes groups of abstractions for work with the computing system components and the computing system on the whole, with embedded system design process and architectural decisions metrics. Basic propositions of HLD-methodology suggested by the authors are given. A number of reconfigurable embedded systems developed with the usage of HLD-methodology elements is represented.
Keywords: embedded system, reconfigurable system, computational architecture, design process, system level design, high-level design.
Acknowledgements. This work was partially financially supported by the Government of the Russian Federation, Grant 074-U01.

Введение

Бурное развитие встраиваемых вычислительных систем (ВсС) определяется растущей степенью автоматизации и компьютеризации всех областей жизни человека. Это предполагает постоянный рост числа и сложности таких систем наряду с требованием кардинального увеличения их надежности, безопасности, производительности, мобильности при одновременном сокращении сроков проектирования и стоимости. По прогнозу ведущих специалистов, доминирующие сегодня вычислительные системы (ВС) с традиционными программируемыми архитектурами в обозримом будущем не смогут обеспечивать улучшение перечисленных показателей.

В качестве альтернативы рассматриваются ВС с реконфигурируемой архитектурой, известные с конца пятидесятых годов прошлого века, но по ряду серьезных причин находящиеся по-прежнему «в тени». Реконфигурируемые ВС в силу своей организации проблемно-ориентированы, что хорошо согласуется с задачами в сегменте ВсС. При этом они чрезвычайно разнообразны по архитектуре и требуют сложных технологий создания конфигурационного обеспечения (configware), что делает задачу их проектирования нетривиальной. Это подтверждается опытом создания большого числа как экспериментальных, так и серийных ВсС с реконфигурируемой архитектурой (РВсС) [1–3]. С помощью принципов реконфигурируемости специалисты решают различные задачи. На первом месте обычно стоит повышение производительности, функциональная гибкость и адаптивность, улучшение комплекса надежностных характеристик, сокращение энергопотребления и размеров. Другие достоинства реконфигурируемости для понимания требуют анализа жизненного цикла изделия и возможных подходов к его проектированию: удовлетворение нужд потребителей, быстрая разработка и выведение на рынок, повышение качества продукта, дифференциация продуктовой линии, адаптация к новым стандартам, снижение стоимости разработки.

Широкое внедрение РВсС требует решения большого числа сложных проблем, среди которых на первом месте стоят вопросы системного (или высокоуровневого) проектирования [4, 5]. Перечислим основные проблемы в проектировании ВсС и РВсС [3–7]:

- доминирует шаблонное (особенно на уровне реализаций вычислительных платформ) проектирование в области аппаратно-программных систем, которое должно отойти на второй план, уступив в идеале место гибкому автоматизированному заказному проектированию;
- в большинстве проектов проявляется искусственное ограничение проектных требований, что приводит к сужению пространства поиска проектных решений архитектурного и микроархитектурного уровня в результате использования традиционных технологических и инструментальных цепочек, сложности их комплексного анализа и изменения (нет адекватных средств и методик);
- отсутствуют эффективные методики и средства проектирования подсистем с различными, в том числе изменяемыми, моделями вычислений и гетерогенной архитектурой.

Специалистами Университета ИТМО и научно-производственной фирмы (НПФ) «ЛМТ» [8, 9] создана и развивается HLD-методология проектирования встраиваемых систем (High Level Design Methodology for Embedded Systems) [6, 10]. Методология направлена на решение перечисленных проблем, одним из ее практических применений является проектирование гетерогенных РВсС с многоуровневой архитектурой. В настоящей работе рассматриваются основные положения HLD-методологии и опыт создания ряда РВсС с ее использованием.

HLD-методология проектирования встраиваемых систем

HLD-методология обеспечивает проектирование ВсС от уровня технического задания до спецификаций (в терминах, не выходящих из области известных технических решений), необходимых для реализации. Методология направляет проектировщика ВсС на создание спецификации, в минимальной степени фиксирующей конкретные элементы реализации, обеспечивая при этом максимальную детализацию применяемых известных технических решений.

Основу HLD-методологии проектирования ВсС составляют [6, 10, 11]:

- система архитектурных абстракций, связывающая воедино компоненты методологии;
- принцип и методы позднего закрепления конкретного способа реализации архитектурных решений;

– модель актуализации вычислительного процесса (ВП) на основе понятия унифицированного транслятора;
– аспектная модель процесса проектирования вычислительной системы;
– класс объектно-событийных моделей вычислений (наряду с иными моделями вычислений).

В рамках HLD-методологии проектировщику предлагается подход, при котором процесс проектирования архитектурной модели вычислительной системы выполняется на уровне, инвариантном к аппаратно-программному (HW/SW) разделению. Данный принцип в определенном объеме используется в современных методологиях проектирования на базе Hardware/Software Codesign [5]. Базовый процесс проектирования ВсС в рамках HLD-методологии представлен на рис. 1 [12].

Рассмотрим подробнее основные положения HLD-методологии. Предлагаемая нами система архитектурных абстракций включает группы для представления отдельных компонентов вычислительной системы и вычислительной системы в целом, для представления процессов проектирования вычислительной системы, для оценки и анализа архитектурных решений.

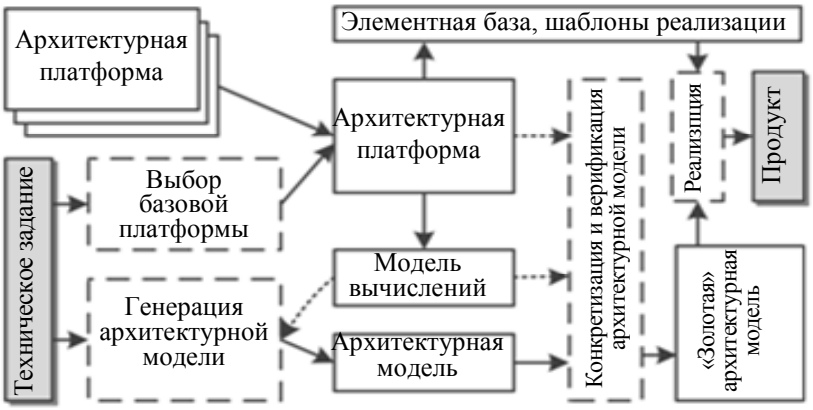

Рис. 1. Процесс проектирования ВсС в рамках HLD-методологии

HLD-методология проектирования в терминах данных абстракций рассматривает ВсС прежде всего через призму организации ее целевого ВП на фазах проектирования (design-time), конфигурирования (config-time), исполнения (run-time) в едином ключе, приглашая разработчика к поиску эффективного распределения элементов ВП по этим фазам. Термин config-time сегодня применяют как в контексте процессов подготовки и развертывания программных систем, так и применительно к РВсС, обозначая процедуры перенастройки.

Проектирование архитектурной модели выполняется по возможности без аппаратно-программного разделения до момента формирования спецификаций для последующей реализации. Технические решения закрепляются в требуемом для конкретной реализации виде с отражением в выделенных проектировщиком сегментах (аспектах). Это согласуется с базовыми принципами Hardware/Software Codesign второго (Platform-Based Design Methodology – PBD [4]) и третьего (Design Space Exploration – DSE) поколения по условной классификации из [5].

Предлагаемый методологией комплексный подход позволяет в полной мере использовать определение архитектуры системы по стандарту [13], которое, наряду с составом компонентов системы, принципами их взаимодействия между собой и с окружением, включает принципы ее проектирования и развития. Такая трактовка понятия вычислительной архитектуры приобретает все большую популярность среди специалистов, выводя инфраструктуру проекта ВсС, включая инструментарий, на один уровень значимости с создаваемой целевой системой.

В HLD-методологии процесс проектирования ВсС рассматривается как организация ВП в пространстве и времени от исходных спецификаций (технического задания) до их реализации на всем жизненном цикле системы.

Модель актуализации ВП (МАВП) демонстрирует совокупность преобразователей (трансляторов), участвующих в его организации. Для базовых элементов модели – трансляторов – различаются конфигурирование и программирование, что естественным образом выстраивает отношение этих фундаментальных механизмов вычислительной техники между собой. Конфигурирование задает правила преобразования, а входная программа для транслятора является обрабатываемыми данными. МАВП эффективно позволяет работать с трансляторами, которые функционируют на различных фазах жизненного цикла ВсС, и в первую очередь на фазах проектирования, конфигурирования, исполнения.

Используемый в HLD-методологии набор взаимодополняющих архитектурных спецификаций позволяет представлять:
– архитектурную платформу (горизонтальная архитектурная спецификация);

– модель актуализации ВП (граф актуализации);
– композицию уровней организации ВП в совокупности с вычислительными функциональными блоками (ФБ), включая фазы проектирования, конфигурирования, исполнения (вертикальная архитектурная спецификация).

В основе нашего понимания вычислительной иерархии во многом лежит модель уровней абстракции вычислительной системы, предложенная Эдвардом Ли [14]. Суть аспектного процесса проектирования (в отличие от [15]) состоит в выделении на начальном шаге проекта важных, по мнению разработчика, сегментов (аспектов) проектного пространства, каждый из которых отражает частную проблему проекта по ходу его выполнения. Наряду с функциональным (поведенческим) аспектом выделяются: аспект аппаратно-программной реализации, аспект этапов жизненного цикла проекта, аспект производительности, аспекты надежности, энергопотребления, временной (реального времени и (или) синхронизации) и другие. Это позволяет разработчику на всех шагах проекта концентрироваться в значительной мере формально на соблюдении функциональных и нефункциональных ограничений, в том числе, управляя текущими приоритетами ограничений (через аспекты). По усмотрению разработчика аспектная модель может быть распространена на различные фазы жизненного цикла системы, что особенно удобно для РВсС с многоуровневой статической и (или) динамической [ре]конфигурацией.

Формализмы, предлагаемые HLD-методологией, выступают инструментами поддержки в первую очередь для потенциально плохо автоматизируемых задач проектирования. Это задачи, лежащие в «проектных доменах» организации ВП, пространства поиска архитектурных и микроархитектурных решений (Design Space Exploration), выбора технологий разработки и реализации.

Поясним подробнее ряд новых абстракций, разработанных нами для HLD-методологии, которые необходимы для дальнейшего обсуждения механизма [ре]конфигурирования.

Вычислительный механизм (ВМх) – архитектурный шаблон, демонстрирующий принципы организации части ВП. В отличие от распространенного понятия «паттерн проектирования», для которого жестко не закреплены требования абстрактности описания и демонстрации внутреннего устройства, ВМх должен прозрачно предоставлять полезные «вычислительные» технические принципы без обязательного закрепления условий их конкретной реализации. Таким образом, ВМх следует рассматривать в качестве определенной категории шаблонов (паттернов) проектирования для вычислительной техники.

ВсС рассматривается в виде композиции ФБ. В состав каждого ФБ входит ряд ВМх, демонстрирующих принципы организации соответствующего ВП. ФБ представлен четверкой спецификаций: функциональной (поведенческой), компонентной (перечень используемых ВМх), интерфейсной (API, характеризующей ФБ как платформу), количественной (набор метрик). Сложность ФБ может меняться в широких пределах – от «покрытия» атомарных вычислительных операций до сколь угодно сложных виртуальных машин (ВМ). Таким образом, под ВМ понимается программируемый ФБ или программный интерпретатор (процессор) независимо от способа его реализации и временной фазы существования.

Унифицированное представление механизмов конфигурирования и программирования в архитектурном проектировании встраиваемых систем

К РВсС специалисты относят архитектуры от однородных вычислительных сред с различной гранулярностью элементов (например, FPGA) до вычислительных сетей различного масштаба (как пространственно-распределенные конфигурации, так и сети на кристалле) [1, 2, 7]. При этом в литературе по-прежнему не дается четкое разделение механизмов программирования и [ре]конфигурирования.

Традиционное понимание реконфигурируемости как способности к изменению организации вычислительной системы и (или) способа ее функционирования, обязательно изменяющее ее аппаратное устройство, назовем реконфигурируемостью «в узком смысле» (это актуально в рамках традиционных процессов проектирования ВсС, когда элементы реализации закрепляются уже на ранних стадиях).

В рамках HLD-методологии мы стремимся к максимуму проектирования на абстрактных уровнях и возможно более позднему HW/SW разделению, в пределе – только на этапе реализации. Следовательно, реконфигурируемое проектное решение архитектурного и (или) микроархитектурного уровня при указанном выше подходе к процессу проектирования от программируемого решения отделить можно либо принудительно (закрепив априори HW/SW-реализацию), либо «волевым порядком» (например, по критерию значимости конфигурационных изменений). Примером второго варианта будет изменение платформы (в терминологии PBD) или системы команд (ISA) процессора (в терминах традиционной технологии последовательного проектирования).

Можно ввести следующие определения:
– программирование ВсС – задание кода приложения для одного или более процессоров или ВМ;
– конфигурирование ВсС – смена спецификации (правил функционирования, состава ФБ, системы команд и так далее) процессора, ВМ или вычислительной платформы в составе ВсС.

Потенциально можно распространить принцип реконфигурируемости ВсС на область программирования, рассматривая программирование в качестве варианта [ре]конфигурирования. При этом под про-

граммной реализацией понимается код приложения, последовательно интерпретируемый процессором. Пример такой обобщающей трактовки приведен в работах Р. Хартенштейна [3, 16].

Для процессов проектирования в парадигме Hardware/Software Codesign необходимо использовать «широкую трактовку» реконфигурируемости как способности к изменению спецификации ВМ, вычислительной платформы или программного кода приложения. В рамках HLD-методологии на основе «широкой трактовки» мы получаем модель механизмов реконфигурируемости для HW/SW-инвариантного процесса проектирования многоуровневой архитектурной модели РВсС. Это также дает возможность классифицировать РВсС по глубине конфигурирования и участвующим в конфигурировании архитектурным уровням в рамках вертикального представления архитектуры.

Таким образом, [ре]конфигурируемость в обобщающей трактовке есть свойство вычислительной системы, выраженное в способности к изменению ее функциональности.

Одна из основных целей в рамках HLD-методологии – это расширение проектного пространства поиска решений, прежде всего, в части HW/SW-реализации и фаз проектирования, конфигурирования, исполнения жизненного цикла системы. Так как при этом формальными средствами на этапе проектирования процедуры конфигурирования и программирования становятся неразличимы, а логическая организация создаваемой ВсС может включать в себя произвольное число иерархий ВМ, определение конкретного механизма в качестве конфигурирующего или программирующего отдается проектировщику.

Вопрос принадлежности конфигурационных изменений к временной фазе существования ВсС требует дополнительного обсуждения и вынесен за рамки работы.

Реконфигурируемые встраиваемые системы, созданные с применением HLD-методологии

Принципы и положения HLD-методологии создавались и проверялись в течение ряда лет совместно специалистами Университета ИТМО и НПФ «ЛМТ» в ходе выполнения НИОКР в области ВсС и систем на кристалле (СнК) с активным использованием механизмов реконфигурирования на разных уровнях архитектурной организации. Кратко рассмотрим наиболее показательные из таких проектов.

Реконфигурируемый вычислитель М3М [17] – ведущий процессорный модуль линейного пункта централизации систем железнодорожной автоматики – представляет собой вычислительное ядро, обладающее высокой безопасностью и надежностью функционирования. В архитектурной проработке проекта была использована новая система архитектурных абстракций и принцип максимально позднего разделения на аппаратную и программную составляющие.

М3М имеет многопроцессорную несимметричную структуру, включающую двоированный процессор x86, оперативную память с контролем и исправлением ошибок, программируемую логику контроля и диагностики, сервисный процессор, коммуникационные контроллеры штатного и инструментального режимов, традиционные средства поддержки реального масштаба времени. Реконфигурируемой частью является уровень вычислительной (или функциональной) платформы. Процесс реконфигурации затрагивает вычислительные механизмы, отвечающие за надежность в пределах платформы, но изменения не выходят за ее границы и затрагивают только состав и алгоритмы работы «непрограммируемых» функциональных блоков.

Средства программируемой логики и сервисный процессор в составе М3М позволили реализовывать операционные, тестирующие, диагностирующие блоки различной сложности, обеспечивая требуемую глубину диагностики (в сочетании с программными средствами), перераспределение ресурсов, изменение правил функционирования интерфейсов и ряд других свойств по требованию заказчика. Результаты разработки показали перспективность выбранных подходов и стимулировали дальнейшее развитие HLD-методологии.

Реконфигурируемые вычислительные платформы LIC5091 [18] и MiniLab [19] ориентированы на создание ВсС сложных аналитических приборов и инструментальных комплексов прототипирования СнК. В данных разработках была успешно опробована аспектная методика проектирования, применена на практике объектно-событийная модель вычислений и использована модель актуализации.

Основу первой платформы составляет одноплатный реконфигурируемый вычислитель MEC5091 (форм-фактор PC-104) с оригинальным soft-процессором в FPGA, коммуникационным микроконтроллером и системой программирования. Вторая платформа, MiniLab, предоставляет пользователю возможности реконфигурирования и программирования на трех уровнях – от уровня аппаратуры (FPGA) до библиотек управления экспериментом в консольном персональном компьютере. Основу составляет параллельный процессор с оригинальной архитектурой NL3, конфигурируемый на уровне топологий функциональных блоков (DPU – Data Processing Unit) и программируемый на языке Java. Процессор NL3 сочетает в себе элементы суперскалярной и VLIW-архитектуры, основной обрабатывающей единицей выступает DPU – реализация функционального блока объектно-событийной модели вычислений [19].

Организация вычислений производится посредством статического и динамического конфигурирования композиции DPU, для чего была разработана инструментальная система, включающая графиче-

скую систему подготовки net-листов, компилятор на основе системы темпоральных неравенств, библиотеки функциональных блоков, design-time и run-time библиотеки драйверов DPU.

Простой вариант реконфигурации заключается в доставке программного обеспечения в конфигурационную память. Более глубокий вариант конфигурации состоит в замене самих DPU и изменении их количества. В рамках архитектурной модели системы реконфигурация затрагивает уровни виртуальных машин (ВМ), вычислительной платформы и прикладной задачи.

Примерами успешного использования элементов HLD-методологии являются перспективная «Платформа автоматизированного проектирования реконфигурируемых аппаратных ускорителей» и проект «Инфраструктурные IP-ядра реконфигурируемых СнК». Оба проекта направлены на поддержку проектирования реконфигурируемых СнК (РСнК) различного назначения.

В первом проекте [20] создан прототип платформы автоматизированного проектирования реконфигурируемых аппаратных ускорителей (РАУ) на основе архитектурного шаблона крупногранулярного РАУ, специального маршрута подготовки конфигураций и программ для системы с ускорителем. Основным рабочим элементом шаблона РАУ является двумерный массив операционных элементов (ОЭ), соединенных посредством иерархической конфигурируемой коммутационной среды между собой и с сервисным процессором. ОЭ, в свою очередь, имеют трехуровневую организацию и набор алгоритмов для быстрых приближенных вычислений. Маршрут проектирования и инструментальные средства обеспечивают выбор параметров аппаратуры РАУ, оптимальное отображение целевой задачи на массив ОЭ и сервисный процессор.

РАУ представляет собой пример системы, аппаратное реконфигурирование которой проходит на верхнем, прикладном уровне. Суть реконфигурирования состоит в соединении крупнозернистых вычислительных элементов друг с другом и в настройке операционных блоков внутри каждого функционального блока посредством списка цепей, который получен в результате анализа работы приложения.

Второй проект направлен на создание набора soft IP-ядер для организации в FPGA и РСнК коммуникационной среды уровня ФБ, отладочной и диагностической исполнительской (run-time) инфраструктуры. Проект находится в активной фазе [21].

Одним из перспективных сегментов использования принципов реконфигурируемости вычислительной архитектуры являются графические процессоры [22]. Под графическим процессором понимается полностью функционально завершенное и программируемое устройство для генерирования изображений, которое может работать по собственной метапрограмме, формируемой специальной программой-драйвером, исполняемой на одном или нескольких сопряженных центральных процессорах или центральных процессорных ядрах. Функциональность графического процессора и его API, определенных каким-либо стандартом (к примеру, OpenGL1/2/3/4.x, OpenCL, CUDA или D3D 9/10/11.x), как правило, связаны между собой, и поддержка тех или иных стандартов на аппаратном уровне средствами графического процессора является одной из ведущих характеристик изделия.

Специалистами коллектива разворачиваются работы по использованию HLD-методологии для анализа и проектирования элементов СнК и соответствующего программного обеспечения для реконфигурируемых сигнальных и графических процессоров.

В представленных проектах ВсС механизмы реконфигурирования вычислительной архитектуры решают различные задачи. Для вычислителя М3М – это обеспечение требуемого уровня надежности и безопасности функционирования в соответствии с отраслевыми стандартами. Для приборных контроллеров LIC-5091 и MiniLab – обеспечение гибкости прикладного программирования эксперимента с возможностью выбора соотношения «прикладная производительность/требуемая квалификация программиста». Для ускорителя РАУ приоритетным является достижение максимальной производительности за счет эффективного отображения прикладной задачи на реконфигурируемый массив вычислительных элементов с помощью разработанных алгоритмов отображения. На рис. 2 показано, что разработанный эвристический метод отображения, основанный на алгоритме «разделения и выталкивания» вершин графа (OPB), для РАУ превосходит известные методы отображения Dynamically Reconfigurable Embedded System Compiler (DRESC), Split Push Kernel Mapping (SPKM) и Quantum-inspired Evolutionary Algorithm (QEA). Эксперименты производились на тестовом наборе приложений для цифровой обработки сигналов DSPStone на массиве РАУ 5×5 32-разрядных вычислительных элементов, объединенных каналами передачи данных и конфигурационной сетью.

Поясним для рассмотренных РВсС глубину и уровни реализованной реконфигурируемости на примере. Упрощенно уровневую организацию ВсС представим совокупностью двух подсистем: вычислительного устройства и приложения. Вычислительное устройство состоит из набора специализированных аппаратных блоков (ФБ) и ФБ программируемого процессора (ВМ). На рис. 3 представлена модель уровней реконфигурируемости в такой системе. Нижний уровень представляет собой платформу, верхний – исполняемое средствами платформы приложение. Изменения могут касаться приложения, состава ФБ, спецификации ВМ.

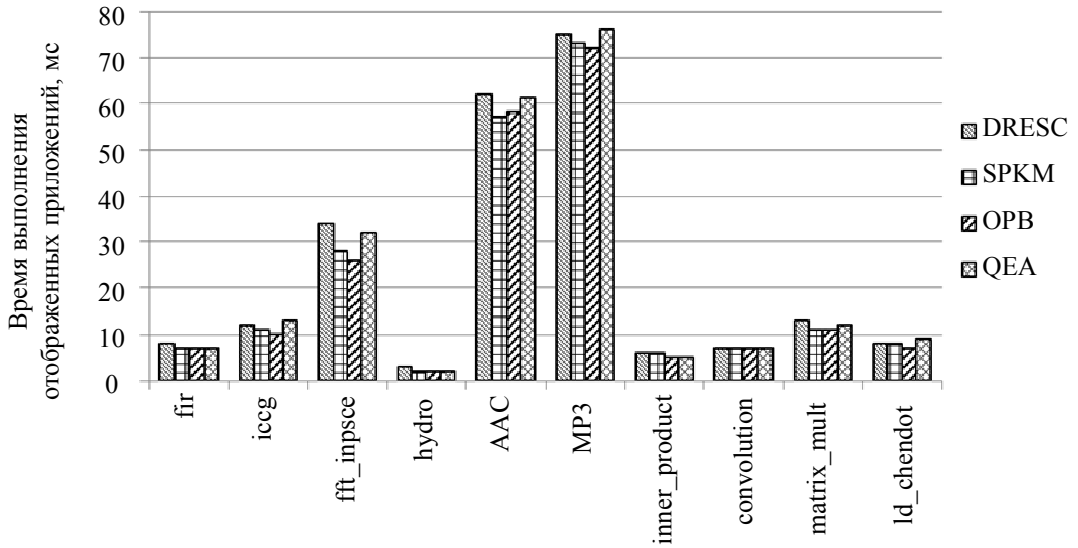

Рис. 2. Время выполнения приложений из тестового набора DSPStone на РАУ. На оси абсцисс показаны названия приложений, входящих в тестовый набор DSPStone

Большинство реконфигурируемых систем работают в рамках верхнего, прикладного уровня. Проект реконфигурируемого аппаратного ускорителя, представленный выше, принадлежит к данной категории. Следующий вариант – реконфигурирование платформы (пример – вычислитель систем железнодорожной автоматики М3М). Самый глубокий уровень реконфигурирования меняет средства организации прикладного уровня, т.е. ВМ (проекты LIC5091 и MiniLab). В системе MiniLab реконфигурирование реализовано на всех уровнях. Чем глубже уровень погружения, тем большее влияние он может оказать на все последующие уровни архитектуры и тем более фундаментальные изменения можно внести в результате реконфигурации в ВсС.

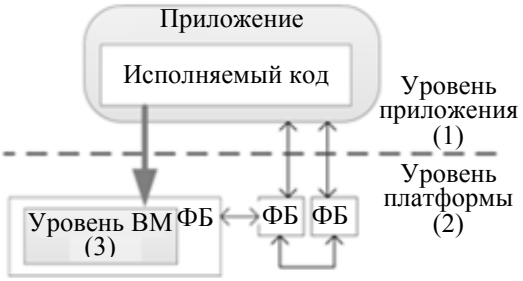

Рис. 3. Уровни вычислительной системы по степени «погружения»: приложение, платформа и виртуальная машина

На рис. 4 приведено сравнение РВсС из примеров по степени влияния реконфигурирования на ВП и на место вносимых изменений в рамках вертикальной архитектуры. Как мы видим, наибольшую гибкость и пространство решений обеспечивает система MiniLab. На втором месте оказывается М3М, на третьем – РАУ.

Рис. 4. Сравнение рассматриваемых примеров по уровню влияния изменений на вычислительный процесс и месту изменений в вертикальной архитектуре

В примере с упрощенной ВсС мы затронули случаи реконфигурирования только в пределах одного уровня ВМ, на практике ВсС представляют собой композиции ВМ и соответствующих платформ с элементами иерархии, что существенно расширяет границы и сложность процессов конфигурирования.

Заключение

Развитие HLD-методологии и практическое ее применение в проектировании реконфигурируемых встраиваемых систем различного назначения позволило решить целый ряд важных задач. Среди них в первую очередь следует отметить возросший уровень архитектурного и микроархитектурного проектирования, который позволил реализовать достаточно сложные проекты небольшим коллективам разработчиков в приемлемые сроки и с высоким качеством.

Разработанная HLD-методология позволяет на практике отказаться от шаблонного проектирования с изначальным закреплением аппаратной или аппаратно-программной платформы. Отказ от стандартных и шаблонных решений в проектировании ряда сложных встраиваемых систем позволил в большинстве случаев получить более эффективные, а значит, более конкурентоспособные продукты. Применение аспектного подхода позволило расширить пространство поиска проектных решений и улучшить учет требований заказчика, а также внутренних требований разработчиков на всех этапах проектирования.

Использование в проектах реконфигурируемых встраиваемых систем гетерогенных архитектур с необходимым количеством различных моделей вычислений позволило разработчикам успешно преодолеть в ряде случаев «взрыв сложности», преследующий разработчиков сложных гомогенных систем.

Коллектив развивает работы в контексте HLD-методологии в двух основных направлениях: создание архитектурных решений с широким использованием механизмов многоуровневого конфигурирования для применения в системах на кристалле и совершенствование системы архитектурного специфицирования с набором инструментальных утилит. На повестке стоит сложная и очень важная задача формализации различных этапов HLD-проектирования встраиваемых систем и создание инструментальных цепочек и технологий, позволяющих поддержать разработанную методологию проектирования.

References

1. Jozwiak L., *Nedjah* N. Modern architectures for embedded reconfigurable systems – a survey. *Journal of Circuits, Systems, and Computers,* 2009, vol. 18, no. 2, pp. 209–254. doi: *10.1142/S0218126609005034*
2. Chattopadhyay A. Ingredients of adaptability: a survey of reconfigurable processors. *User Modeling and User-Adapted Interaction*, 2013, vol. 2013, art. no. 683615. doi: 10.1155/2013/683615
3. Hartenstein R. The relevance of reconfigurable computing. In: *Reconfigurable Computing* (Eds. J.M.P. Cardoso, M. Hübner). Springer, 2011, pp. 7–34. doi: 10.1007/978-1-4614-0061-5_2
4. *Sangiovanni-Vincentelli A. Quo vadis, SLD? Reasoning about trends and challenges of system-level design. Proceedings of the IEEE, 2007, vol. 95, no. 3, pp. 467–506. doi: 10.1109/JPROC.2006.890107*
5. Teich J. Hardware/Software codesign: the past, the present, and predicting the future. *Proceedings of the IEEE*, 2012, vol. 100, pp. 1411–1430. doi: 10.1109/JPROC.2011.2182009
6. Platunov A., Kustarev P. Problems of abstract representation of embedded systems at high-level stages design. *Networked Embedded and Control System Technologies: European and Russian R and D Cooperation - Proceedings of the 1st International Workshop - NESTER 2009 In Conjunction with ICINCO.* Milan, Italy, 2009, pp. 100–107.
7. Jozwiak L., Nedjah N., Figueroa M. Modern development methods and tools for embedded reconfigurable systems: a survey. *Integration, the VLSI Journal,* 2010, vol. 43, no. 1, pp. 1–33. doi: 10.1016/j.vlsi.2009.06.002
8. *Nauchno-obrazovatel'noe napravlenie "Vstroennye vychislitel'nye sistemy"* [Scientific and educational direction "Embedded Computing Systems"]. Available at: http://embedded.ifmo.ru (accessed 03.04.2014)
9. *Nauchno-proizvodstvennaya firma "LMT"* [Scientific and Production Company "LMT"]. Available at: http://lmt.ifmo.ru (accessed 04.04.2014).
10. Platunov A., Nickolaenkov A., Penskoy A. Architectural representation of embedded systems. *Proc. of Mediterranean Conference on Embedded Computing, MECO 2012.* Montenegro, 2012, art. no. 6268930, pp. 80–83.
11. Platunov A., Nickolaenkov A. Aspects in the design of software-intensive systems. *Proc. of Mediterranean Conference on Embedded Computing, MECO 2012.* Montenegro, 2012, art. no. 6268931, pp. 84–87.
12. Platunov A.E., Postnikov N.P. Perspektivy formalizatsii metodov proektirovaniya vstroennykh sistem [Prospects of formalization techniques for embedded systems]. *Elektronnye Komponenty,* 2005, no. 1, pp. 24–29.
13. *ISO/IEC/IEEE 42010:2011, Systems and software engineering – Architecture description.* 24.11.2011. Geneva, International Organization for Standardization. *37 p.*
14. Lee E.A., Neuendorffer S., Wirthlin M.J. Actor-oriented design of embedded hardware and software systems. *Journal of Circuits, Systems, and Computers,* 2003, vol. 12, no. 3, pp. 231–260. *doi: 10.1142/S0218126603000751*
15. Broman D., Lee E.A., Tripakis S., Toerngren M. Viewpoints, formalisms, languages, and tools for cyber-physical systems. *Proc. of 6th International Workshop on Multi-Paradigm Modeling, MPM 2012*. Innsbruck, Austria, 2012, pp. 49–54. doi: 10.1145/2508443.2508452
16. Becker J., Hartenstein R. Configware and morphware going mainstream. *Journal of Systems Architecture*, 2003, vol. 49, pp. 127–142. doi: 10.1016/S1383-7621(03)00073-0
17. Gavrikov V.O., Platunov A.E., Nikiforov N.L. Kompleks tekhnicheskikh sredstv dlya sistem zheleznodorozhnoi avtomatiki [Complex of hardware systems for railway automation]. *Avtomatika, Telemekhanika i Svyaz' na Zheleznykh Dorogakh,* 1998, no. 11, pp. 5–10.
18. Golubok A.O., Platunov A.E., Sapozhnikov I.D. Sistema upravleniya skaniruyushchim zondovym mikroskopom [Control system for a scanning-probe microscope]. *Nauchnoe Priborostroenie,* 2003, vol. 13, no. 3, pp. 25–31.
19. Bolgarov I.S., Makovetskaya N.A., Platunov A.E., Postnikov N.P. Proektirovanie pribornykh kontrollerov [Design of instrument controllers]. *Izv. vuzov. Priborostroenie,* 2012, vol. 55, no. 10, pp. 73–78.

20. Rumyantsev A.S. Organizatsiya i instrumental'nye sredstva rekonfiguriruemykh vychislitel'nykh sistem [Organization and design tools of reconfigurable computing systems]. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics,* 2012, no. 4 (80), pp. 79–84.
21. *Nauchno-obrazovatel'noe napravlenie "Sistemy na kristalle"* [Scientific and educational direction "Systems on a chip"]. Available at: http://soc.ifmo.ru (accessed 03.03.2014).
22. Paltashev T. New graphics API and GPU hardware architecture co-development. Proc. of *Graphicon 2005 - International Conference on Computer Graphics and Vision. 2005.* Available at: http://www.graphicon.ru/2005/proceedings/papers/Paltashev.pdf (accessed 03.03.2014).

Ключев Аркадий Олегович – кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, Россия, kluchev@lmt.ifmo.ru

Кустарев Павел Валерьевич – кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, Россия, kustarev@lmt.ifmo.ru

Палташев Тимур Турсунович – доктор технических наук, профессор, профессор, Фримонт, Калифорния, США, Северо-Западный политехнический университет; старший менеджер, АМД, Калифорния, США, Timour.Paltashev@amd.com

Платунов Алексей Евгеньевич – доктор технических наук, профессор, профессор, Университет ИТМО, Санкт-Петербург, Россия, platunov@lmt.ifmo.ru

Arkady O. Kluchev PhD, Associate Professor, ITMO University, Saint Petersburg, Russia, kluchev@lmt.ifmo.ru

Pavel V. Kustarev PhD, Associate Professor, ITMO University, Saint Petersburg, Russia, kustarev@lmt.ifmo.ru

Timur T. Paltashev D.Sc., Professor, Professor, Northwestern Polytechnic University, Fremont, California, USA; Senior Manager, Advanced Micro Devices (AMD), California, USA, Timour.Paltashev@amd.com

Alexei E. Platunov D.Sc., Professor, Professor, ITMO University, Saint Petersburg, Russia, platunov@lmt.ifmo.ru

Принято к печати 14.03.14
Accepted 14.03.14