Обзор современных подходов и средств к «программистской» разработке аппаратного обеспечения алгоритмически сложных цифровых систем

Ставшее превалирующим в последнее десятилетие проектирование быстродействующих алгоритмически сложных цифровых систем на базе языков описания аппаратуры типа VHDL, Verilog и др. испытывает серьезные проблемы с производительностью труда инженеров. Поэтому все время ведутся разработки новых средств проектирования, которые обеспечат более удобные и эффективные средства ввода и отладки описаний алгоритмов функционирования цифровых систем, автоматически генерируя корректные «по построению» HDL-описания. Данный материал имеет своей целью представить предлагаемые сегодня исследовательские наработки и коммерческие продукты в этой области.

Михаил Долинский

dolinsky@gsu.unibel.by

Глобально подобные разработки можно классифицировать следующим образом:

1. По 'front-end' средствам, то есть по средствам ввода описаний алгоритмов:
   - системы визуального ввода;
   - системы текстового ввода (C/Java-программы).
2. По 'back-end' средствам, то есть по средствам аппаратной реализации отлаженных алгоритмов:
   - на выходе — HDL-описания, не ориентированные на конкретное аппаратное обеспечение;
   - на выходе — HDL-описания для специально разработанной архитектуры;
   - на выходе — модифицированное программное обеспечение, исполняющееся на спецпроцессоре, для которого сконфигурировано дополнительное аппаратное обеспечение, исходя из исследования исходного программного обеспечения.

Отдельно нужно рассматривать подход, получивший название «микропрограммирование».

Далее приводятся краткие описания представителей продуктов соответствующих направлений.

Интересно отметить, что исследования на уровне архитектуры — ярчайшая тенденция последней конференции DAC (Design Automation Conference), состоявшейся в июне 2003 года [18].

1. Системы визуального ввода высокоуровневых описаний алгоритмов

1.1. Академический компилятор MATLAB-HDL

В работе [1] представляется компилятор, который обрабатывает высокоуровневые алгоритмы обработки сигналов и образов, описанные в MATLAB, и генерирует оптимизированное аппаратное обеспечение для FPGA с внешней памятью. Предлагается алгоритм точного анализа, определяющий минимальное количество битов, требуемых для целых переменных, комбинированный алгоритм для анализа точности и ошибок, для подсчета минимального количества битов, требуемых для вещественных переменных. Приведенные результаты показывают, что в среднем эти алгоритмы генерируют аппаратное обеспечение, которому требуется в 5 раз меньше ресурсов FPGA в терминах CLB (Configurable Logic Blocks) по сравнению с аппаратным обеспечением, сгенерированным без этих оптимизаций. Анализ обеспечивает сокращение размеров таблиц поиска (lookup tables) для таких функций, как SIN, COS, SQRT, EXP и т. д. Предлагаемый анализ точности позволяет упаковывать различные элементы массива в одно место в памяти, чтобы сократить количество обращений к внешней памяти. Такой подход повышает производительность сгенерированного аппаратного обеспечения на 35%.

1.2. Altera DSP Builder

1 мая 2003 года Altera выпустила DSP Builder версии 2.1, который интегрирует MathWorks MATLAB и Simulink со средой разработки FPGA Quartus II, поддерживая эффективное создание определяемых пользователем сопроцессоров DSP на FPGA. DSP Builder 2.1 распространяется по подписке — $1995 на 12 месяцев. Возможна 30-дневная бесплатная оценка DSP Builder 2.1 (раздается с сайта **http://www.altera.com/codedsp**).

1.3. Xilinx System Generator for DSP

17 марта 2003 года Xilinx выпустила System Generator for DSP 3.1, который уменьшает время

симуляции от месяцев до минут для DSP-проектов новых поколений.

System Generator for DSP 3.1 резко сокращает время симуляции проектов за счет hardware-in-the-loop и HDL co-simulation. SGD автоматически транслирует DSP-системы из MATHLAB и Simulink-описаний в высокооптимизированные VHDL-описания и IP-компоненты для Xilinx FPGA.

«Hardware-in-the-loop» существенно ускоряет цикл проектирования, поскольку позволяет пользователям верифицировать проекты в FPGA непосредственно из среды Simulink. В других методологиях инженеры должны верифицировать проекты в различных средах, что усложняет и замедляет процесс.

«Hardware-in-the-loop» уже поддерживается такими разработчиками, как AlphaData, Annapolis, Lyr, Nallatech.

«HDL co-simulation» позволяет пользователям импортировать HDL-код и симулировать всю систему в целом. Mentor Graphics ModelSim автоматически вызывается из Simulink и производится совместная симуляция Simulink- и HDL-моделей.

Кроме того, поддерживается одновременная симуляция булевых выражений MATLAB M-Code и процессора Xilinx PicoBlaze.

Xilinx Virtex-II и Virtex-II Pro FPGA обеспечивают до 556 встроенных умножителей 18×18 и до 10 мегабит блочной и распределеной памяти.

System Generator for DSP 3.1 продается для Xilinx Virtex и Spartan по цене $1995 [17–19].

26 ноября 2002 года Xilinx выпустила System Generator for DSP 2.3. System Generator for DSP 2.3 автоматически транслирует DSP-системы, описанные с использованием MATLAB и Simulink от MathWorks в хорошо оптимизированные VHDL и IP-компоненты для Xilinx FPGA. Версия 2.3 позволяет задействовать конвейеризованные встроенные умножители, используя пре-размещенные входные и выходные регистры, чтобы достичь более высокой и предсказуемой производительности — до 285 МГц для Xilinx Virtex II.

Версия 2.3 также имеет:

- в три раза ускоренную генерацию кодов;
- более дюжины примеров проектов, чтобы сократить время обучения;
- примеры создания пользовательских DSP периферийных устройств на базе архитектуры шины IBM CoreConnect.

Вот что говорит Jim Waite (Voyan Technology): «Когда мы начали наш проект OptiFusion, наши DSP-разработчики быстро осознали, что использование FPGA — единственный путь достичь требуемой производительности. Однако наши инженеры не знали, как программировать на VHDL — естественном языке для FPGA-проектов. Используя System Generator for DSP, мы смогли исследовать проектное пространство и автоматически создать VHDL, тест-бенчи и симуляционные файлы для ModelSim. В результате мы сократили трудозатраты на четыре человеко-месяца».

System Generator for DSP + Xilinx ISE 5.1 — это весьма продуктивный способ создания DSP-систем.

Версия 2.3 System Generator for DSP содержит большое число примеров проектов для цифровой коммуникации и обработки образов, включая 16-QAM-приемник, адаптивный эквалайзер, CORDIC-процессор, 2-D DWT, фильтры и др.

Оценочная копия System Generator for DSP может быть загружена с сайта. Xilinx EDK (Embedded Development Kit) позволяет отлаживать системы с процессором MicroBlaze или PowerPC на FPGA Virtex-II и Spartan-II. Имеются примеры, демонстрирующие, как встроенный процессор выполняет on-line перезагрузку DSP-данных под управлением программы на хост-PC.

Цена System Generator for DSP 2.3 — $1995, (2495 — с трехдневным обучением «DSP Design Flow for FPGAs»).

Инициатива Xilinx XtremeDSP была введена в 2000 году. Цель — создание и распространение IP-компонентов для DSP (**http://www.xilinx.com/systemgenerator_dsp**, **http://www.xilinx.com/dsp**, **http://www.xilinx.com/pw2003**, **http://www.xilinx.com/xapp/xapp264.pdf**).

1.4. Aldec Active HDL 5.2 + AccelChip

12 ноября 2002 года AccelChip и Aldec интегрировали свои разработки.

В результате AccelFPGA от AccelChip поддерживает средства верификации от Aldec. Большинство разработчиков DSP-систем начинают свои работы в MathWorks. Они используют язык MATLAB, на котором можно эффективно выразить свои математические алгоритмы на поведенческом уровне. После отладки этих алгоритмов в MATLAB AccelFGPA от AccelChip автоматически синтезирует по MATLAB-моделям оптимизированные RTL-описания. Интеграция с Aldec позволяет DSP-разработчикам симулировать полученные RTL-коды. Такой комбинированный подход экономит человеко-месяцы при разработке DSP-систем. С AccelFPGA интегрированы Active-HDL 5.2 и Riviera 2002.09.

AccelChip основана в 2001 году. Занимается разработкой и распространением средств высокоуровневого синтеза, ускоряющих процесс проектирования чипов.

2. Системы, генерирующие на выходе HDL-описания, не ориентированные на конкретное аппаратное обеспечение

2.1. Aldec Active-HDL 6.1 + Celoxica

22 мая 2003 года Aldec добавила синтез из C-описаний от Celoxica в Active-HDL 6.1. Active-HDL 6.1 параллельно поддерживает HDL и C/C++ описания вплоть до реализации. Все результаты C-синтеза от Celoxica аннотируются.

2.2. Behavioral Design Suite, CynLib/C++, GigaScale и Cynthesizer C++-to-HDL от Forte Design Systems

15 сентября 2003 года Behavioral Design Suite от Forte Design Systems автоматизирует процесс получения из высокоуровневых C++ моделей множества RTL-реализаций с различными характеристиками (площадь кристалла, потребляемая мощность, производительность). BDS автоматически создает RTL для операционного автомата (datapath), управляющего автомата с жесткой логикой (FSM and control logic). Синтезатор BDS создает верифицированный RTL-код (готовый для логического синтеза) за часы вместо недель, которые потребовались бы при разработке таких RTL-описаний вручную [18].

19 ноября 2001 года Synplicity и Forte Design Systems впервые обеспечили полный путь проектирования и верификации от C++ до PLD. Теперь они взаимодействуют с Altera, чтобы оптимизировать этот C++ поток для SOPC-проектов.

Разработчики могут использовать Cynthesizer C++-to-HDL (от Forte) и Synplify Pro (от Synplicity), чтобы синтезировать из C++ кода нет-лист для широкого круга ПЛИС, включая Excalibur от Altera.

Synplicity и Forte работают над тем, чтобы использовать Cynlib C++ как альтернативный поток проектирования SOPC.

4 июня 2000 года Bravara выбрала средства проектирования и верификации от Forte Design Systems.

Forte Design Systems образована слиянием CynApps и Chronology, поставляет интегрированное ПО для проектирования и верификации больших и сложных электронных систем. Это ПО включает в себя C++/Cynlib, Cynthesizer, Cyn++, Cynchronizer, Co-Cym и GigaScale Hub.

Традиционно системные архитекторы и разработчики аппаратного обеспечения работали на разных языках и на различных уровнях абстракции, что делало невозможным использование ценных наработок в проекте и верификации. По мере усложнения ПЛИС это становится серьезной проблемой.

GigaScale от Forte обеспечивает быструю симуляцию C++ описания проекта и автоматический переход от C++ к HDL.

Цена Synplify Pro — от $19 тыс. Цена GigaScale — от $40,8 тыс. Cynlib распространяется бесплатно по лицензии «open source».

CynLib 1.2 — библиотека классов C++ для описания HW. Есть интерфейс с Verilog-косимуляцией.

Cyn++ — макропрепроцессор, упрощающий описание HW по сравнению с HDL-описаниями. Инженеры кодируют в синтаксисе, близком к HDL, Cyn++ генерирует текст CynLib/C++, который можно симулировать бесплатно. В течение месяца с момента выкладывания на сайт произведено несколько тысяч загрузок этого ПО.

CynApps поддерживает автоматическую генерацию синтезируемых HDL-описаний.

CynApps продвигает методологию последовательного улучшения от абстрактных C-моделей до C-моделей RTL-уровня. Для C-моделей RTL-уровня CynApps продает инструменты автоматической генерации синтезируемых HDL-описаний.

Чтобы поддержать Verilog-компоненты, SynApps предлагает Synchronizer, конвертирующий синтезируемые Verilog-описания в CynLib-описания для последующей совместной симуляции [3].

2.3. Handel-C и DK1 от Celoxica

Handel-C — язык, разработанный в Оксфордском университете, добавляет в C конструкции управления временем, параллелизма и взаимодействия между параллельными процессами. Handel-C соответствует стандарту ISO/ANSI C. При компиляции Handel-C вначале код преобразуется в древовидную структуру, которая затем конвертируется в EDIF-описание.

Celoxica [23] разработала среду DK1, которая позволяет переводить описания Handel-C прямо в оптимизированные EDIF-описания для FPGA. DK1 может быть бесплатно загружена с сайта для симуляции.

Два программиста из Creative Labs за 7 недель с помощью Handel-C и DK1 (Celoxica) перевели 10 тыс. строк C-кода, реализующего аудиообработку на Pentium-166, в описание Handle-C и затем FPGA, которая на частоте 6 МГц обеспечивает ту же обработку в реальном времени [4].

Аудиообработка включала в себя 4 стадии:

- извлечение цифрового аудиосигнала;
- преобразование данных из временного представления в частотное и сортировка по 32 поддиапазонам;
- кодирование данных;
- снабжение закодированных данных заголовками для поддержки фреймов и формирование окончательного цифрового потока аудиоданных.

2.4. CycleC, Csim и System Compiler Designer от C Level Design

Инженеры проектируют и верифицируют с высокой производительностью, описывая системы с помощью методологии CycleC. Затем генерируют синтезируемые описания VHDL/Verilog с помощью System Compiler.

C Level Design разработала System Compiler Designer — завершенную среду разработки аппаратного обеспечения на базе C/C++. System Compiler Designer обеспечивает поддержку графического анализа результатов симуляции посредством стандартных вьюверов VCD (value-change-dump), автоматизацию регрессионного тестирования для верификации сгенерированных HDL-описаний, мощные средства компиляции, позволяющие эффективно разрабатывать аппаратное обеспечение.

Это первая система синтеза HDL-описаний из ANSI C/C++ описаний. Сейчас мы обеспечили возможность использовать C/C++ параллельно с Verilog/VHDL, выполнять анализ проекта во время компиляции, отлаживать проект с помощью традиционных средств отладки и автоматизировать верификацию результатов синтеза C/C++ -> HDL.

Анализатор StyleChecker, встроенный в System Compiler Designer, выполняет анализ проекта на C/C++, указывая на конструкции, которые будут формировать неэффективный или некорректный HDL-код при синтезе.

С StyleCheker можно использовать код C++ аналогично тому, как используется HDL-код в проекте, но производить симуляцию в 300 раз быстрее.

System Compiler Designer имеет возможность автоматически генерировать векторные регрессионные тесты для сгенерированных HDL-описаний. Входные воздействия и эталонные реакции сохраняются. Они могут быть использованы при симуляции сгенерированных HDL-описаний, с автоматическим сравнением новых и старых результатов симуляции и сообщением об ошибках.

System Compiler Designer выпущен во втором квартале 2001 года по цене $95 тыс.

CSim обеспечивает интеграцию симуляции C/C++ со средствами от Cadence для HDL-синтеза и симуляции, Verilog-XL, NC Verilog, NC VHDL.

C Level Design поддерживают полный синтаксис ANSI C/C++ для использования при моделировании от системного до RTL-уровня.

2.5. SystemC и CoCentric SystemC Compiler от Synopsis

В ноябре 2000 года Synopsis и Xilinx объявили о годичной программе для разработки полной системы проектирования с SystemC в Xilinx Virtex FPGA.

SystemC введен в сентябре 1999 года, свободно распространяется на сайте **www.systemc.org**, включает все конструкции, необходимые для эффективного описания проектов аппаратного обеспечения. Цена CoCentric SystemC Compiler — от $40 тыс.

2.6. SystemC и A|RT Designer от Frontier Design

Frontier Design (www.frontierd.com) разработала продукт A|RT Designer, который переводит алгоритмы на ANSI C и SystemC в синтезируемые описания VHDL/Verilog для чипов Altera APEX и Xilinx Virtex [5, 6].

По словам создателей, не важно, что будет генерироваться — HDL или EDIF, одно можно сказать со всей определенностью — разработка SoC скоро станет чисто программной проблемой.

2.7. SystemCenter от Future Design Automation

Системные архитекторы создают функциональности на ANSI C, по которым SystemCenter генерирует синтезируемые HDL-описания (Verilog/VHDL) [18].

2.8. Конвертор C->Verilog

Разработчики 256-битного графического акселератора Neon попутно создали конвертор C->Verilog, модифицировав портируемый C-компилятор LCC [7].

2.9. Проект SPARK Калифорнийского университета в Ирвине

Проект SPARK, разрабатываемый в Калифорнийском университете в Ирвине, обеспечивает ввод алгоритмов на ANSI C (с запретом использовать указатели и рекурсивные функции). Один из генерируемых выходов — синтезируемое VHDL-описание.

По мнению авторов, уже 15 лет разрабатываются подобные системы, но коммерческие аналоги не получили пока широкого распространения по причине низкого качества генерируемых описаний, особенно для условий и циклов [8].

2.10. Конвертор Java->HDL

В работе [2] предложены методы и средства для компиляции программного обеспечения (Java) в реконфигурируемое аппаратное обеспечение.

Разработанный front-end Java-компилятор (Caladrier) поддерживает следующее подмножество Java:

- Типы: boolean, byte, shortint, long, char, array.
- Операции: / * + — сдвиги, логические, отношений.
- Управляющие: циклы (while, for, dowhile), условия (if, switch).

Caladrier анализирует исходный java-текст, строит по нему DFG (Data Flow Graph) и HPDG (Hierarchical Program Dependency Graph), извлекая параллелизм на уровне операций.

Back-end компилятор NeNya строит для полученных DFG и HPDG VHDL-представление для последующего синтеза и загрузки в FPGA.

В качестве основных достоинств предложенного ими подхода авторы подчеркивают следующие:

- отладка и тестирование программного обеспечения проводится независимо от целевой архитектуры;
- в качестве исходных данных для анализа и синтеза служат байт-коды Java, с одной стороны, платформонезависимые, а с другой — байт-коды Java — это исполняемая спецификация алгоритмов;
- представление программного обеспечения в виде графов для упрощения поиска параллелизма всех уровней;
- максимальное использование множественного потока управления и внутреннего параллелизма на уровне инструкций.

3. Системы, обеспечивающие на выходе HDL-описания для специально разработанной архитектуры

3.1. Reconfigurable Communication Processor фирмы Chameleon

Фирма Chameleon разработала архитектуру RCP (Reconfigurable Communication Processor) [9], представляющую массив вычислительных элементов (RPF — Reconfigurable Processing Fabric) с программируемыми связями между ними. Имеются также 4 банка, содержащих по 40 программируемых контактов ввода-вывода и подсистема встроенного процессора ARC, включающая 8K инструкций, 8K данных и интерфейсы для взаимодействия с внешней памятью.

Вычислительные элементы (DPU — Data Processing Units, 84 штуки) программируются с помощью 8 определяемых пользователем инструкций, хранящихся в локальной памяти инструкций DPU. Каждый блок из 7 DPU (tile) имеет доступ к четырем компонентам локальной памяти (LSM — Local Store Memory) — 128 слов RAM по 32 бита. Все эти элементы памяти также прямо адресуются из адресного пространства ARC-процессора и на чтение, и на запись.

Функционирующая в рабочем режиме, система «на лету» может быть подготовлена к реконфигурации менее чем за 3 микросекунды.

Переключение на работу в новой конфигурации осуществляется за 1 такт. Система CS2112 функционирует на частоте 125 МГц.

RPC позволяет добиться оптимального использования параллелизма и на уровне задач, и на уровне инструкций. При этом цикл проектирования включает три основных этапа:

- определить поток данных через систему;
- отобразить его на архитектуру RCP;
- ввести проект с помощью средств программирования RCP.

3.2. Reconfigurable Algorithm Processing фирмы Elixent

27 января 2003 года Toshiba и Elixent анонсировали совместные разработки в области реконфигурируемых SoC-платформ. Эти платформы будут интегрировать конфигурируемый массив выполнения алгоритов D-Fabric от Elixent (www.elixent.com) и конфигурируемый процессор MeP от Toshiba (www.mepcore.com).

Комбинация динамической реконфигурации и изменения архитектуры выполнения алгоритма «на лету» под управлением процессора Toshiba MeP существенно сокращает стоимость и энергопотребление. Получается полностью программируемое решение с производительностью, превосходящей самые производительные DSP.

Платформа Elixent D-Fabrix RAP (reconfigurable algorithm processing) разрабатывает алгоритмы в «виртуальном аппаратном обеспечении», позволяя создание аппаратной акселерации любого алгоритма системы. Благодаря реконфигурируемости она может реализовывать множество аппаратных акселераторов на одной и той же области чипа, что дает высокую эффективность использования площади кристалла. Более того, реконфигурируемость позволяет добавлять и изменять функциональность после изготовления чипа, исправляя ошибки или добавляя новые функции.

Это достигается благодаря отображению алгоритма в обрабатывающий массив из АЛУ, регистров и памятей, предоставляя уникальную возможность адаптировать любой алгоритм, тем самым обеспечивая гибкость программных решений с производительностью ASIC.

При сравнении на бенчмарках со стандартными DSP D-Fabrix обеспечивает 10-кратное преимущество в производительности при меньших размерах кристалла и существенно сокращенном потреблении энергии. D-Fabrix — это новый класс устройств, поддерживающих мультифункциональность и адаптируемость к изменению спецификаций.

Elixent основана в октябре 2000 года группой сотрудников из Hewlett-Packard Research Labaratories. Первоначальное финансирование обеспечено альянсом HP, Actel и 3i.

RAA (Reconfigurable ALU Array) фирмы Elixent состоит из 4-битных АЛУ и блоков регистров-буферов, которые могут конфигурироваться для обработки слов разной длины (от 8 до 24 бит).

Реконфигурируемость — горячая тема для полупроводников, поскольку:

1. Появились коммерческие чипы по технологии 90 нм, обеспечивающие 4-кратное увеличение плотности по сравнению с технологией 0,18 мкм и двукратное увеличение плотности по сравнению с 0,13 мкм. Стоимость маски для технологии 90 нм примерно в 6 раз выше, чем стоимость маски для технологии 0,18 мкм.
2. В то же время лучшие фабрики переходят на подложки 300 мм, которые примерно вдвое увеличивают количество устройств на подложке по сравнению с последним поколением подложек размером 200 мм.

Эта комбинация убийственна для отрасли. 6-кратное увеличение маски поднимает планку массовости выпуска, при которой выгодно производить ASIC.

Решением этой проблемы являются реконфигурируемые SoC-платформы, которые содержат набор программируемых компонентов, таких, как RISC и RAP (Reconfigurable Algorithm Processor), возможно, со специальными интерфейсами. Такая SoC может реконфигурироваться, чтобы служить в различных рыночных нишах.

Два основных достоинства:

- цена маски разделяется между множеством проектов;
- SoC полностью программируема, следовательно, если будет найдена ошибка или потребуются изменения функциональности, не будет необходимости изготавливать новый чип (и платить за маску повторно).

Реконфигурируемость «на лету» и «во времени» позволит использовать одну область кристалла для выполнения множества функций, что приводит к сокращению размеров чипа.

4. Системы модификации ПО, исполняющегося на спецпроцессоре, для которого сконфигурировано дополнительное аппаратное обеспечение, исходя из исследования исходного ПО

4.1. Program In, Chip Out (Hewlett-Packard Labaratories)

Разработчики PICO (Program In, Chip Out) [10, 11] назвали такой подход архитектурным синтезом, чтобы отделить его от поведенческого и логического синтеза.

Для приложения, написанного на C, автоматически строится множество проектов и выбирается оптимальный в смысле Парето, а также строится структурное описание VHDL-компонентов и скомпилированный код для программного обеспечения.

Поток проектирования в PICO-подходе:

1. Входное приложение на C содержит циклы интенсивных вычислений.
2. PICO обнаруживает и извлекает каждый из них.
3. PICO-N spacewalker специфицирует NPA.
4. PICO-N преобразовывает NPA в выделенное ядро с требуемым параллелизмом, вводит локальную память, минимизируя нагрузку на глобальную память, и генерирует RTL-описание для NPA и нужный код.
5. Для каждого ядра, реализованного как NPA, синтезированный код заменяет исходный код в приложении.
6. PICO-N spacewalker специфицирует VLIW-процессор.
7. PICO-VLIW создает архитектуру и микроархитектуру VLIW-процессора, генерирует их RTL-описание и описание в виде базы данных для Elcor (VLIW-компилятора).
8. Elcor автоматически перестраивается на новую VLIW-архитектуру и компилирует модифицированное приложение.
9. Cache Spacewalker оценивает и указывает конфигурацию иерархии кэш-памяти.
10. System Level PICO Spacewalker объединяет совместимые VLIW-процессор, оптимальный в смысле Парето, кэш и NPA-проекты и определяет, что будет разработано как NPA, а что — как программное обеспечение на VLIW.

По программе строится MDES (Machine DEScription database), которая используется для автоматической перенастройки системного ПО на эту архитектуру.

4.2. Jazz/SOLO на базе Java-описаний алгоритмов

Jazz — это мультипроцессорная платформа. Специализированные Java-описания алгоритмов обрабатываются системой компиляции SOLO, которая обеспечивает автоматическое распределение задач по процессорам платформы Jazz и конфигурацию Jazz под эту систему задач [12].

4.3. Java HDL и Configurable Computing Machines

JHDL (Java HDL) и CCMs (Configurable Computing Machines) используются для определения и отладки системы на кристалле [13].

4.4. Nimble Compiler фирмы Synopsis

На входе системы — C-тексты программ и описание целевой архитектуры на ADL (Architecture Description Language). Nimble Compiler обеспечивает компиляцию, анализ производительности, разбиение задач между программной и аппаратной реализацией, синтезирует требуемую аппаратную поддержку, модифицирует программное обеспечение. На выходе — bit-stream для FPGA и C-код для исполнения на CPU [14].

4.5. Средства C-проектирования от Celoxica

22 мая 2003 года Celoxica анонсировала развитие поддержки C-проектирования для новых 90 нм Xilinx FPGA Spartan-3.

Используя средства от Celoxica, инженеры могут разбить проект на программную и аппаратную части, верифицировать единую систему и непосредственно синтезировать аппаратное обеспечение с C-описаний в FPGA Spartan-3. В проекте можно использовать процессоры MicroBlaze, языки C, C++, System-C и Handel-C.

4.6. Xtensa Xplorer от Tensilica

Xtensa Xplorer — это IDE для разработки SoC на базе конфигурируемых процессоров Xtensa. Поддерживается ввод и отладка прикладного программного обеспечения (ПО), конфигурация и оптимизация процессора (генерацией специальных инструкций) под прикладное ПО [18].

4.7. Cascade Tool Suite от Criticalblue

Продукт может быть использован для ускорения любого микропроцессорного приложения для произвольной архитектуры.

Система включает множество уникальных средств синтеза сопроцессора, спроектированных специально для ускорения выбранных задач прикладного ПО. При этом оптимизированная архитектура сопроцессора и соответствующий RTL-код получаются непосредственно по прикладному ПО. Таким образом, исполнение части кода перекладывается с процессора на сопроцессор [18].

5. Инструменты поддержки микропрограммирования

5.1. Средства поддержки MPGL

MPGL — это высокоуровневый язык микропрограммирования. Он позволяет писать микропрограммы в последовательном и процедурном стиле для описанной целевой машины. Имеется специальный симулятор, который принимает описание целевой машины и микропрограмму. Его характеристики таковы:

1. Симулируется машинный код микропрограмм и поддерживается отладка по исходным (высокоуровневым) текстам микропрограмм.
2. Обеспечена применимость к целому спектру целевых машин, поскольку описание целевой машины используется в качестве входной информации.
3. Упрощается логическая проверка как целевой машины, так и микропрограмм для нее.

Специальный компилятор микропрограмм преобразует последовательно написанную исходную микропрограмму в эффективный код машинных инструкций. В процессе компиляции выделяются 4 фазы:

- синтаксический анализ;
- предобработка для оптимизации;
- переупорядочивание и объединение микроинструкций;
- адресация и битовое кодирование.

Для обеспечения большей применимости поддерживается ввод описаний целевой машины и автоматическое обнаружение параллельно исполняемых микроопераций.

5.2. Средства поддержки обучения разработке микроархитектур и микропрограммированию

Одним из традиционных подходов к поддержке обучения является разработка средств симуляции, соответствующих распространенным учебным пособиям. По теме разработки микроархитектур и микропрограммирования одним из таких учебных пособий является классическая монография Э. Таненбаума «Архитектура компьютера» [17]. В ней, в частности, последовательно и исключительно детально описаны четыре модификации микроархитектуры MIC: от MIC-1 (простая машина с программным управлением, последовательным выполнением команд и полным отсутствием параллелизма) до MIC-4 (высокопараллельная микроархитектура с семистадийным конвейером). Неудивительно, что имеется огромное количество средств симуляции и отладки микропрограмм для микроархитектуры MIC (например, http://www.cs.qub.ac.uk/~J.Campbell/myweb/proj/dpsim/doc, http://cis.stvincent.edu/carlsond/cs330/mic1/mic1.html, http://www.ontko.com/mic1/user_guide.html).

Заключение

Высокоуровневое проектирование (проектирование на системном уровне), обеспеченное инструментами симуляции и анализа — эффективное средство исследования проектного пространства. Однако, подстегиваемые сжатыми сроками на разработку, инженеры зачастую вынуждены пропускать этот этап ввиду разрыва между средствами разработки на системном и последующих уровнях.

Масштабность предпроектных исследований обязывает проводить их не с использованием HDL-языков описания аппаратного обеспечения, а на более высоком уровне абстракции, например, полагаясь на языки программирования типа C/C++. В то же время необходимость сокращения сроков проектирования требует еще более эффективного использования результатов проектирования на архитектурном и системном уровнях. Один из идеальных вариантов — генерировать HDL-описания из отлаженных высокоуровневых описаний (C/C++/SystemC).

В области проектирования аппаратного обеспечения с помощью диалектов C расширяется переход от академических исследований к коммерческим разработкам и, соответственно, практическому использованию. Интересно, что главными движущими силами внедрения такого подхода в практику являются поставщики ПЛИС и разработчики средств автоматизации проектирования ПЛИС: Altera, Xilinx, Aldec и др.

Внедрение синтеза FPGA непосредственно из C-описаний может расширить круг разработчиков аппаратного обеспечения, повысить их производительность труда, сократив тем самым сроки и стоимость разработки.

Основной источник превышения смет и временных лимитов проектов — ошибки разработчиков. Поэтому естественно стремление индустрии развивать средства, генерирующие безошибочные «по построению» проекты из более высокоуровневых отлаженных описаний.

Многие из представленных разработок позволяют использовать MATLAB в качестве средства высокоуровневого описания, моделирования и исследования проектного пространства встроенной системы. А затем автоматически генерировать синтезируемое HDL-описание, которое может быть загружено в соответствующую FPGA.

Отметим, что среди рассматриваемых решений не приводились системы типа SoC, когда на чипе просто размещается процессорное ядро, и алгоритмы реализуются в виде программного обеспечения для данного процессора. Дело в том, что существует множество прикладных областей, в которых такой подход не приемлем из-за ограничений по производительности, потребляемой мощности, стоимости и т. д.

Литература

1. Nayak A., Haldar M., Choudhary A., Banerjee P. Precision and Error Analysis of MATLAB Applications during Automated Hardware Synthesis for FPGAs // Proc. of DATE 2001. Germany, Munich.
2. Cardoso J., Neto M. Compilation for FPGA-based Reconfigurable Hardware // Design & Test of Computers. 2003. Vol. 20. No 2.
3. Johnson E. Is C Up to Design Environment Goals // EE Times. 2000. 25 September.
4. Songy M. Creative Software Engineers tackle Rapid Development of Audio Processor Hardware // Integrated System Design Magazine. 2001. July (www.isdmag.com/oeg20010711s0048.html).
5. Larner D. Tactics to win the SOC socket // Embedded Systems. 2001. Vol. 5. No 35.
6. Swart R., Janssen R., Pool A., Heikamp C. Speech Recognition SOC through Architectural Synthesis // Integrated System Design Magazine. 2000. June (http://www.isdmag.com/applicat.htm).
7. McCormack J., McNamara R., Gianos C., Jouppi N. Implementing NEON: A 256-bit Graphic Accelerator // IEEE Micro. 1999. Mar-Apr.
8. Gupta S., Savoiu N., Kim S., Dutt N., Gupta R., Nicolau A. Speculation Techniques For High Level Synthesis of Control Intensive Design // Design Automation Conference. 2001.
9. Salefski B., Caglar L. Reconfigurable Computing in Wireless // Design Automation Conference. 2001.
10. Ramakrishna B., Schlansker M. Embedded Computer Architecture and Automation // IEEE Computer. 2001. Vol. 34. No 4.
11. Aditya S., Mahlke S., Ramakrishna B. Code Size Minimization and Retargetable Assembly for Custom EPIC // ACM TODAES. 2000. Vol. 5. No 4.
12. Levia O. Programming System Architectures with Java // IEEE Computer. 1999. August.
13. Hutchings B., Nelson B., Wirthlin M. Designing and Debugging Custom Computing Applications // IEEE Design & Test of Computers. 2000. Jan/Mar.
14. Li Y., Callahan T., Darnell E., Harf R., Kurkuse U., Stockwood HW/SW Co-Design of Embedded Reconfigurable Architectures // Design Automation Conference. 2000.
15. B. Takanobu et al. Microprogram Simulator of MPG // PSJ Magazine Abstract. Vol. 19. No. 05 (http://www.ipsj.or.jp/members/Magazine/Eng/1905/article005.html).
16. B. Takanobu et al. Microprogram compiler of MPG // IPSJ Magazine Abstract/ Vol. 19. No. 01. (http://www.ipsj.or.jp/members/Magazine/Eng/1901/article004.html).
17. Э. Таненбаум. Архитектура компьютера. СПб: «Питер Пресс». 2002.
18. J. Happich. DAC Report: highlighting trends in EDA // Electronic Product News. 2003. No 7.