УДК 004.318

Ю. В. Байда[1,2], *А. В. Бутузов*[2], *А. Г. Ефимов*[2], *М. С. Цветков*[2]

[1]Московский физико-технический институт (государственный университет)
[2]ЗАО «Интел А/О»

Методология перехода от программной потактовой модели микропроцессора к аппаратному симулятору на базе программируемой логики

Разработка новой микропроцессорной архитектуры требует принятия большого количества решений, основанных на результатах моделирования производительности. Использование ПЛИС позволяет создать симулятор микропроцессора, обладающий на три порядка большей скоростью, чем традиционные программные симуляторы. В статье представлена оригинальная методология перехода от существующего программного потактового симулятора к аппаратной модели на базе программируемой логики.

Ключевые слова: микропроцессор, микроархитектура, потактовая, симулятор, симуляция, производительность, модель, ПЛИС.

Введение

В настоящее время в области разработки микропроцессоров постепенно происходит переход от традиционных архитектур к архитектурам, существенно взаимодействующим с компиляторами [4]. Разработка новой архитектуры микропроцессора требует большого количества решений, при принятии которых архитекторы существенно опираются на результаты имитационного моделирования.

В качестве такой модели обычно используется специально разработанный программный потактовый симулятор микропроцессора, который при достаточной точности обладает катастрофически низкой производительностью, моделируя несколько тысяч модельных тактов за секунду своей работы [2]. При такой скорости моделирование двух минут работы проектируемого микропроцессора заняло бы порядка одного года работы симулятора. Это делает невозможным исследование производительности модели микропроцессора в длительных запусках, которые требуются, например, при выборе алгоритма предсказания переходов.

Рис. 1. Традиционное применение ПЛИС на позднем этапе маршрута проектирования микропроцессора для создания прототипа

Традиционно, программируемые логические интегральные схемы (ПЛИС) в маршруте проектирования микропроцессоров используются лишь на конечных этапах в качестве прототипов для схемотехнической отладки (см. рис. 1). Конечно, тактовая частота прототипа оказывается гораздо ниже тактовой частоты готовой микросхемы в кристалле: современные микропроцессоры фирмы Intel микроархитектур Atom и Nehalem были успешно запущены

в ПЛИС на тактовых частотах 50 МГц и 520 кГц соответственно [8, 10], но вместе с тем гораздо выше скорости моделирования той же схемы в схемотехнических симуляторах.

В последнее время внимание исследователей из академических и промышленных кругов все больше направлено на изучение возможности применения ПЛИС для *симуляции* работы микропроцессоров, т.е. на гораздо более раннем этапе проектирования (см. рис. 2). Конфигурация вентильной матрицы ПЛИС при этом не повторяет в точности конечную электрическую схему микропроцессора, а только *моделирует* ее поведение и временные характеристики. Например, симуляция одного такта моделируемого микропроцессора теперь может выполняться за несколько тактов ПЛИС. Такой поход позволяет эффективнее использовать ограниченные ресурсы ПЛИС, одновременно увеличивая ее тактовую частоту по сравнению с прототипом.

Рис. 2. Применение ПЛИС на раннем этапе маршрута проектирования микропроцессора для создания быстрого симулятора

Результаты исследовательских проектов ведущих американских университетов, таких как UT-FAST [2], ProtoFlex [3], RAMP Gold [9] и HAsim [6] (совместно с компанией Intel), показывают, что применение ПЛИС позволяет создать аппаратный потактовый симулятор микропроцессора, обладающий на порядки большей скоростью симуляции, чем традиционные программные симуляторы (см. рис. 3).

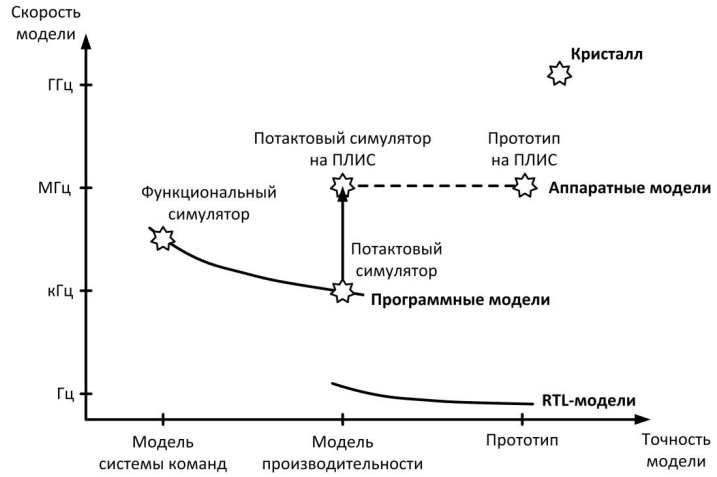

Рис. 3. Сравнение скоростей различных видов симуляторов микропроцессора

Однако, несмотря на то, что использование ПЛИС может существенно увеличить скорость симуляции, их применение затруднено низким уровнем абстракции традиционных языков описания аппаратуры, гораздо более длительным циклом разработки по сравнению с разработкой программного обеспечения и ограниченной логической емкостью применяемых ПЛИС.

В статье описывается эффективная методика помодульного перехода от существующего программного потактового симулятора микропроцессора к аппаратному симулятору на ПЛИС. Предлагаемый подход позволяет эффективно использовать усилия, уже затрачен-

ные на разработку программной модели, а сохранение иерархической структуры и графа потока данных исходной модели позволяет обеспечить быструю и надежную валидацию получаемого аппаратного симулятора.

Обзор применяемых технологий

Разработанная методология описывается на примере перехода от программного симулятора микроархитектуры, разработанного в окружении Asim, к аппаратному симулятору с использованием окружения HAsim, однако может быть применена и в случае использования других собственных или сторонних окружений.

Целью проекта Asim являлось создание окружения для разработки программных потактовых симуляторов микропроцессоров на языке Си++. При этом симулятор делится на две логические части: функциональную и временну́ю. Функциональная часть отвечает за корректное исполнение команд на уровне архитектуры системы команд (декодирование команд, обновление памяти симулятора и т.д.). Временная же часть отвечает за отслеживание событий с точностью до такта (напр., попадание или промах в кеш) и микроархитектурное поведение (например, какую следующую команду отправлять на декодирование). Такое разделение, в том числе, позволяет сократить усилия на разработку симулятора, поскольку одна и та же функциональная часть может использоваться с различными временными.

Время в такой модели вводится с помощью *портов*, которые устанавливают границы между модулями, обычно соответствующими тем или иным узлам будущего микропроцессора, и определяют задержку прохождения сигнала между ними. Конкретная реализация модуля может изменяться внутри границ: значения сигналов вычисляются за нулевое модельное время, а время передачи сигнала определяется только значением атрибута соответствующего порта.

Проект HAsim (хэйсим), ведущийся компанией Intel совместно с Массачусетским технологическим институтом, является своего рода развитием проекта Asim. Он представляет собой открытую, гибкую и расширяемую инфраструктуру для разработки симуляторов на базе ПЛИС. Проект наследует лучшие идеи, разработанные в рамках проекта Asim, а также содержит ряд нововведений в связи с переходом на ПЛИС.

Концепция портов, применяемых в Asim, была адаптирована для применения в программируемой логике. Это позволяет переносить существующую модель из Asim в HAsim с сохранением структуры модулей и портов и, кроме того, упрощают распределённую синхронизацию блоков модели и уменьшают использование ресурсов ПЛИС [7].

В HAsim введено разделение модельного такта от такта ПЛИС. Такое разделение дает возможность экономить ресурсы ПЛИС за счет увеличения времени моделирования (частично это компенсируется возможностью повышения тактовой частоты) [6], т.к. алгоритмы, наиболее эффективно задействующие ресурсы ПЛИС, могут работать несколько тактов, чтобы смоделировать один такт модели. Для сравнения различных реализаций тех или иных блоков вводится характеристика FMR (англ. *FPGA-cycles-to-model-cycles ratio*), равная количеству тактов ПЛИС, затрачиваемых на моделирование одного такта микропроцессора.

Трудоемкость разработки аппаратного симулятора сокращается за счет введения стандартизированного набора интерфейсов виртуальной платформы LEAP (англ. *logic-based environment for application programming*), которая представляет единый набор служб, абстрактных устройств, иерархию памяти и протокол коммуникации для различных физически платформ [6]. Основной интерфейс между собственно симулятором и платформой представляет собой набор виртуальных устройств и протокол удаленного вызова процедур RRR (англ. *remote request response*). Виртуальная платформа обеспечивает разработчика средствами для сбора статистики, отслеживания событий и вывода отладочной информации.

Для описания аппаратуры применяется язык высокого уровня Bluespec SystemVerilog [5], который за счет повышения уровня абстракции сокращает время разработки, позволяет

активнее переиспользовать код и избежать низкоуровневых ошибок.

Методология конвертации моделей Asim в модели HAsim

Оригинальная методология, применяемая в проекте HAsim, предполагает нисходящую разработку и не предоставляет инструментов для помодульной разработки симулятора. Для снижения ресурсоёмкости разработки модели, нами был предложен восходящий путь разработки с использованием кода существующего программного симулятора на базе инфраструктуры проекта Asim, при котором происходит миграция из Asim в HAsim модуль за модулем. При этом исходный блок Asim-модели служит в качестве эталона при тестировании и валидации модуля аппаратного симулятора.

Процесс миграции начинается с изучения исходного кода одного из модулей эталонной программной модели и затем разработки аналогичного модуля на языке описания аппаратуры Bluespec SystemVerilog с эквивалентным набором входных и выходных портов.

На следующем шаге порты рассматриваемого модуля исходной модели заменяются портами с регистраторами, которые используются для сохранения в файл (журнал) последовательностей сообщений, прошедших через них.

Далее, к разработанному модулю подключается специальный служебный модуль, который подает на входные порты последовательность сообщений из ранее сохраненного файла, полученного при запуске эталонной модели, а также сохраняет последовательность сообщений из всех выходных портов разработанного модуля. В дальнейшем выходные последовательности из эталонного и разработанного модулей проверяются на эквивалентность (см. рис. 4).

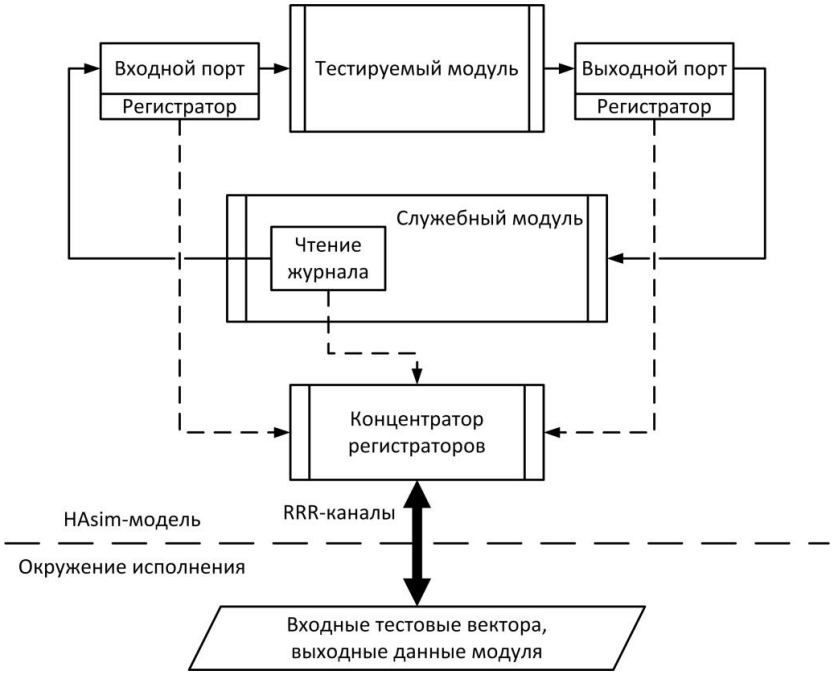

Рис. 4. Структура модели для тестирования отдельного модуля

Тестирование модели при этом возможно на всех этапах маршрута проектирования: на этапе написания исходного кода на языке Bluespec SystemVerilog с помощью симулятора Bluesim, после компиляции исходного кода на языке Bluespec SystemVerilog в RTL-описание на языке Verilog — с помощью известных в индустрии RTL-симуляторов, например Synopsis VCS. Наконец, тестирование возможно после этапов синтеза, размещения и загрузки конфигурационного файла в ПЛИС. При этом набор тестовых последовательностей используется один и тот же.

Отметим, что вся инфраструктура для автоматической валидации локализована на

уровне библиотек Asim и HAsim, и не затрагивает функциональность самого модуля. Благодаря этому, методология валидации является гибкой и хорошо масштабируется.

Расширение функциональности портов

Рассматриваемая методология была применена для разработки аппаратной модели участка конвейера, занимающегося подготовкой команд к исполнению (англ. *front-end*) и состоящего из 3 блоков: выборки команд, декодирования команд и очереди декодированных команд. Разработка велась на базе существующего программного симулятора перспективного микропроцессора с внеочередным исполнением команд.

В процессе разработки HAsim-модели выяснилось, что оригинальные порты поддерживают передачу только одного сообщения в модельный такт. Однако в моделях современных микропроцессоров, как правило, требуется передача нескольких сообщений за модельный такт. В разрабатываемой нами модели часто встречались порты с пропускной способностью до 32 сообщений за такт.

На начальном этапе были рассмотрены два способа расширения портов, дающие возможность передавать более одного сообщения за такт: объединение сообщений в вектор и передача вектора сообщений как единого целого, а также статическое расширение пропускной способности порта до требуемого (пикового) количества сообщений. Существенным недостатком первого способа является значительное увеличение ресурсов ПЛИС, затрачиваемых на преобразование последовательности сообщений в вектор и обратно; недостатком второго способа является низкая эффективность передачи сообщений через порт, поскольку отправка каждого сообщения занимает один такт ПЛИС, то отправка всех сообщений сильно увеличивает параметр FMR, что негативно сказывается на скорости симуляции.

Для борьбы с этим негативным эффектом был применен подход динамически изменяемой пропускной способности. В основе этого подхода лежит тот факт, что пиковая пропускная способность порта всегда выше среднего количества сообщений, передаваемых через порт. Результаты наших экспериментов показали существенное отличие среднего количества отправляемых сообщений от максимально допустимого спецификацией.

Для того чтобы при этом сохранить протокол синхронизации, применяемый в HAsim, каждое сообщению было дополнено флагом, который маркирует последнее сообщение в данном порту в текущем модельном такте.

Для демонстрации эффекта от предложенной модификации были простроены две модели: со статической пропускной способностью и модифицированная. Эксперименты показали значительное увеличение скорости симуляции: параметр FMR уменьшился в 3–10 раз. В то же время использование модифицированных портов увеличивает использование ресурсов ПЛИС незначительно. Результаты синтеза с помощью САПР Xilinx ISE для ПЛИС Xilinx Virtex 6 LXT240 приведены в табл. 1.

Таблица 1

Изменение параметров модели при переходе от портов с фиксированной пропускной способностью к динамически изменяемой

Блок	**Параметр модели**	
	FMR	**Использование ресурсов ПЛИС**
Выборка команд	-86%	+4%
Декодирование команд	-77%	+8%
Очередь декодированных команд	-78%	+1%
Участок подготовки команд	-86%	+9%

Мультиплексирование с разделением по времени

При необходимости моделирования нескольких ядер в HAsim используется мультиплексирование с разделением по времени, когда одно и то же *физическое* ядро используется для последовательной симуляции нескольких виртуальных экземпляров ядра, как показано на рис. 5. Внутреннее состояние ядра, например регистровый файл, дублируется, но комбинационная логика используется одна и та же.

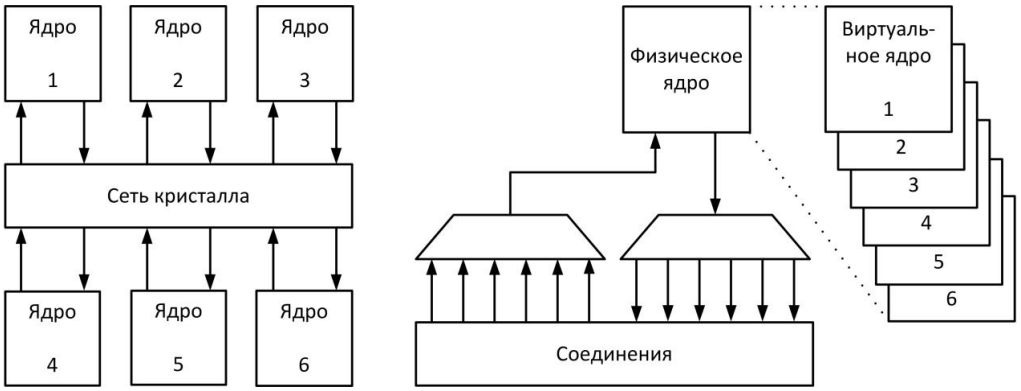

Рис. 5. Моделирования нескольких ядер и сети кристалла напрямую (слева) и с помощью мультиплексирования с разделением по времени (справа)

Из табл. 2 видно, что при переходе к моделированию четырех ядер вместо одного, параметры модели, такие как количество тактов ПЛИС, затрачиваемых на симуляцию одного такта модели и использование ресурсов ПЛИС, изменяются непропорционально увеличению количества ядер.

Таблица 2

Изменение параметров модели при увеличении количества ядер от 1 до 4 и использовании мультиплексирования с разделением по времени

Блок	**Параметр модели**	
	FMR	**Использование ресурсов ПЛИС**
Выборка команд	$\times 1,23$	$+35\%$
Декодирование команд	$\times 2,75$	$+24\%$
Очередь декодированных команд	$\times 2,67$	$+16\%$
Участок подготовки команд	$\times 1,24$	$+38\%$

Результаты

Разработанная методология была применена для разработки аппаратной модели участка конвейера, занимающегося подготовкой команд к исполнению, на базе существующего программного симулятора перспективного микропроцессора с внеочередным исполнением команд. Было достигнуто повышение скорости симуляции на два порядка по сравнению с программной моделью (см. табл. 3).

В ходе разработки библиотека HAsim была расширена для поддержки применяемой аппаратной платформы на базе ПЛИС Xilinx Virtex 6, чему способствовала открытость исходных кодов библиотеки.

Для оценки эффективности предлагаемой методологии было произведено сравнение моделей рассматриваемого микропроцессора, разработанных на языках Си++, Bluespec и Verilog, в терминах количества строк кода.

Как видно из диаграммы на рис. 6, по количеству строк кода аппаратная модель, разработанная на языке описания аппаратуры Bluespec SystemVerilog, примерно соответствует программной модели, разработанной на языке Си++, и существенно опережает RTL-

Таблица 3

Сравнение скорости полученного аппаратного симулятора на ПЛИС со скоростью эталонного программного симулятора

Блок	FMR	Скорость симуляции, МГц	Ускорение, раз*
Выборка команд	67	0,98	208
Декодирование команд	16	4,00	851
Очередь декодированных команд	64	1,03	219
Участок подготовки команд	67	0,98	208

* При средней скорости программной модели, равной 4,7 кГц и тактовой частоте ПЛИС в 66 МГц

описание на языке Verilog. Таким образом, сложность разработки аппаратного симулятора на ПЛИС с применением разработанной методологии примерно соответствует сложности разработки программной модели на языке Си++.

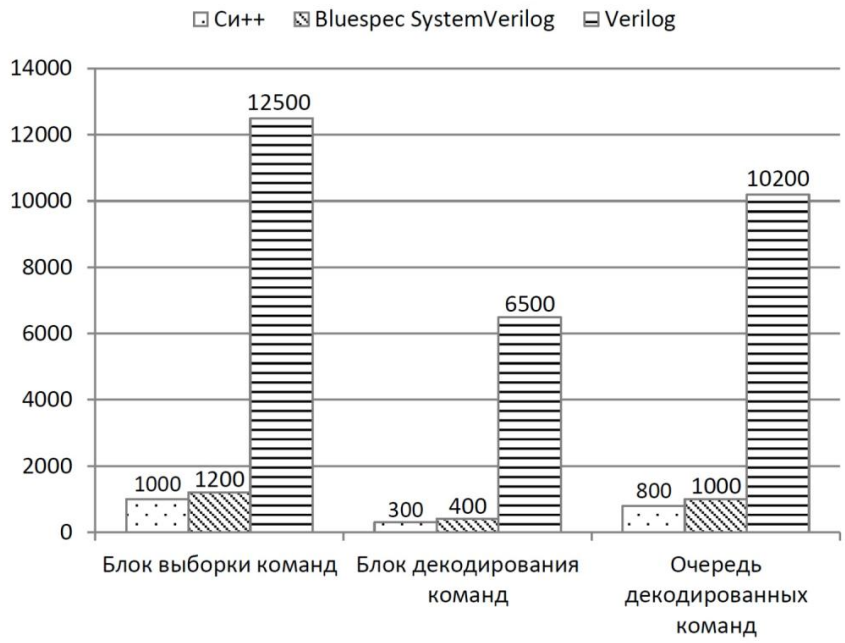

Рис. 6. Сравнение количества строк кода, необходимых для реализации модулей симулятора, на различных языках

Заключение

Рассмотрена оригинальная методика помодульного перехода от существующего программного потактового симулятора микропроцессора к симулятору на ПЛИС с использованием окружения HAsim. На модели входного тракта конвейера современного микропроцессора с внеочередным исполнением команд достигнута скорость симуляции в 1 МГц.

На опыте разработки показана эффективность предложенной методики, а также использования окружения HAsim и языка описания аппаратуры высокого уровня Bluspec SystemVerilog за счет существенного уменьшения затрат ресурсов на разработку по сравнению с традиционными подходами.

Дальнейшим развитием модели является реализация полного конвейера микропроцессора и создание аппаратной модели, способной к запуску операционной системы с системой динамической двоичной трансляции.

Литература

1. *Chiou D., Sanjeliwala D., Kim J.,* [*et al*]. The FAST methodology for high-speed SoC/computers simulation // Proceedings of IEEE/ACM international conference on Computer-aided design. — 2007. — P. 295–302.
2. *Chiou D., Sunwoo D., Kim J.,* [*et al*]. FPGA-accelerated simulation technologies (FAST): fast, full-system, cycle-accurate simulators // Proceedings of the 40th IEEE/ACM annual international symposium on Microarchitecture. — 2007. — P. 249–261.
3. *Chung E.S., Nurvitadhi E., Hoe J.C.,* [*et al*]. Protoflex: FPGA-accelerated hybrid functional simulator // Proceedings of IEEE international symposium on Parallel and distributed processing. — 2007. — P. 1–6.
4. *Kim H.S., Smith J.E.* An instruction set and microarchitecture for instruction level distributed processing // Proceedings of 29th international symposium on Computer architecture. — 2002. — P. 71–81.
5. *Nikhil R.* Bluespec System Verilog: efficient, correct RTL from high level specifications // Proceedings of the 2nd ACM/IEEE international conference on Formal methods and models for co-design. — 2004. — P. 69–70.
6. *Pellauer M., Adler M., Kinsy M.,* [*et al*]. HAsim: FPGA-based high-detail multicore simulation using time-division multiplexing // Proceedings of the 17th IEEE international symposium on High performance computer architecture. — 2011. — P. 406–417.
7. *Pellauer M., Vijayaraghavan M., Adler M.,* [*et al*]. A-Port networks: preserving the timed behavior of synchronous systems for modeling on FPGAs // ACM Transactions on reconfigurable technology and systems. — 2009. — V. 2, N. 3. — P. 1–26.
8. *Schelle G., Collins J., Schuchman E.,* [*et al*]. Intel Nehalem processor core made FPGA synthesizable // Proceedings of the 18th ACM/SIGDA international symposium on Field programmable gate arrays. — 2010. — P. 3–12.
9. *Tan Z., Waterman A., Avizienis R.,* [*et al*]. RAMP gold: an FPGA-based architecture simulator for multiprocessors // Proceedings of the 47th ACM/IEEE design automation conference. — 2010. — P. 463–468.
10. *Wang P.H., Collins J.D., Weaver C.T.,* [*et al*]. Intel Atom processor core made FPGA-synthesizable // Proceeding of the 17th ACM/SIGDA annual international symposium on Field programmable gate arrays. — 2009. — P. 209–218.

Поступила в редакцию 15.12.2011.