УДК 004.318

На правах рукописи

Ба́йда Юрий Владимирович

Методы разработки и тестирования аппаратных потактовых моделей микропроцессоров на программируемых логических интегральных схемах

05.13.05 — Элементы и устройства вычислительной техники и систем управления

АВТОРЕФЕРАТ

диссертации на соискание учёной степени

кандидата технических наук

Москва

2013

Работа выполнена на кафедре микропроцессорных технологий Московского физико-технического института (государственного университета).

Научный руководитель: **Бутузов Александр Валерьевич**
кандидат технических наук,
руководитель подразделения по разработке микроархитектуры ЗАО «Интел А/О»

Официальные оппоненты: **Топорков Виктор Васильевич**
доктор технических наук, профессор,
заведующий кафедрой вычислительной техники НИУ «Московский энергетический институт»

Груздов Фёдор Анатольевич
кандидат технических наук,
начальник отдела разработки высокопроизводительных микропроцессоров ЗАО «МЦСТ»

Ведущая организация: ОАО «Институт точной механики и вычислительной техники им. С. А. Лебедева»

Защита состоится 27 ноября 2013 года в 15:00 на заседании диссертационного совета Д 409.009.01 при ОАО «Институт электронных управляющих машин им. И. С. Брука» по адресу: 119334, г. Москва, ул. Вавилова, д. 24.

С диссертацией можно ознакомиться в библиотеке ОАО «Институт электронных управляющих машин им. И. С. Брука».

Автореферат разослан 23 октября 2013 года.

Учёный секретарь
диссертационного совета
кандидат технических наук, профессор Красовский В. Е.

Общая характеристика работы

Актуальность темы исследования. Разработка микропроцессора как сложной системы включает в себя принятие большого количества проектных решений, существенно опирающихся на результаты имитационного моделирования, с помощью которого анализу подвергаются такие параметры, как производительность, потребляемая мощность и др.

Для измерения производительности проектируемого микропроцессора традиционно используются программные симуляторы, которые при достаточной точности обладают очень низкой скоростью, моделируя порядка одной тысячи команд в секунду реального времени.

При такой скорости моделирование одной секунды работы разрабатываемого микропроцессора потребует нескольких дней работы симулятора, что делает невозможным исследование производительности при запуске длинных тестов или при работе реальных операционных систем с приложениями.

Таким образом, существует техническое противоречие между низкой скоростью программных потактовых симуляторов, используемых на микроархитектурном этапе маршрута проектирования, и необходимой на данном этапе точностью.

Результаты исследований, опубликованных в современной литературе, свидетельствуют о том, что существенного (на 2-3 порядка) повышения скорости моделирования без потери точности можно достичь реализацией потактового симулятора на программируемых логических интегральных схемах (ПЛИС).

Поскольку конфигурация вентильной матрицы ПЛИС, в которой реализован симулятор, в отличие от прототипа, не должна повторять в точности конечную электрическую схему микропроцессора, а только *моделировать* её поведение и временны́е характеристики, то разработчик модели получает ряд преимуществ, недоступных разработчику прототипа. Например, симуляция одного такта моделируемого микропроцессора в этом случае может выполняться за несколько тактов ПЛИС.

Тем не менее, использование ПЛИС затруднено низким уровнем абстракции традиционных языков описания аппаратуры, гораздо более длительным циклом разработки по сравнению с разработкой программного обеспечения, следовательно, наличие эффективной методологии разработки является ключевым фактором успеха, что говорит об **актуальности** исследования.

Объект исследования — микропроцессоры.

Предмет исследования — методы разработки и тестирования потактовых моделей микропроцессоров.

Цель исследования — построение эффективных методов разработки и тестирования аппаратных потактовых моделей микропроцессоров на ПЛИС, позволяющих добиться без потери точности существенного повышения скорости моделирования при низкой сложности разработки.

Для достижения цели исследования решаются следующие **задачи**:

- исследование существующих способов повышения скорости потактовых моделей микропроцессоров и их классификация;
- исследование особенностей применения ПЛИС для потактового моделирования микропроцессоров, позволяющих снизить трудоёмкость по сравнению с разработкой прототипа;
- разработка эффективного метода разработки аппаратных потактовых симуляторов микропроцессоров на ПЛИС;
- разработка метода и системы автоматизированного тестирования аппаратных потактовых симуляторов микропроцессоров на ПЛИС;
- апробация разработанных методов и средств для разработки потактового симулятора современного многоядерного микропроцессора для подтверждения результатов исследования.

Научная новизна исследования определена решением поставленных задач и заключается в следующем:

- проведён анализ и предложена классификация способов повышения скорости потактовых моделей микропроцессоров, включая: сэмплинг, абстрагирование, распараллеливание и применение ПЛИС;
- выделены особенности применения ПЛИС для потактового моделирования микропроцессоров, позволяющие снизить трудоёмкость по сравнению с разработкой прототипа, за счёт того, что конфигурация вентильной матрицы не повторяет конечную электрическую схему микропроцессора;
- разработан восходящий метод модульной разработки аппаратных потактовых симуляторов микропроцессоров на ПЛИС, отличительной особенностью которого является использование существующего программного потактового симулятора в качестве эталона, что позволяет существенно снизить трудоёмкость разработки;
- разработан метод тестирования аппаратных потактовых симуляторов

микропроцессоров на ПЛИС, отличающийся тем, что тестирование и отладка модели возможны, начиная с самого нижнего уровня иерархии модулей;

— разработана система автоматизированного тестирования аппаратных потактовых симуляторов микропроцессоров на ПЛИС, включающая средства автоматической генерации служебного кода.

Методы исследования. Задача и способы повышения скорости потактовых симуляторов микропроцессора рассматривались с позиций системного анализа и синтеза. При создании методов разработки и тестирования потактовых симуляторов на ПЛИС были применены теория графов и теория множеств. Для разработки автоматизированной системы тестирования использовались методы объектно-ориентированного программирования.

Практическая значимость исследования заключается в существенном повышении скорости потактовых моделей микропроцессоров без потери точности при низкой сложности разработки.

Разработанные методы и средства были **внедрены** в экспериментальный комплекс предварительного проектирования микропроцессоров на сверхбольших интегральных схемах ЗАО «Интел А/О».

Теоретические исследования, связанные с разработкой моделей микропроцессоров, легли в основу разделов лекций курса «Основы программного моделирования ЭВМ» кафедры микропроцессорных технологий МФТИ (ГУ).

Апробация результатов работы. Результаты работы докладывались на всероссийских и международных научно-технических конференциях:

— 53-й Научной конференции МФТИ «Современные проблемы фундаментальных и прикладных наук», Москва, 2010;

— Международной молодёжной научной конференции «XXXVII Гагаринские чтения», Москва, 2011;

— XI Международной научно-практической конференции «Фундаментальные и прикладные исследования, разработка и применение высоких технологий в промышленности», Санкт-Петербург, 2011;

— Международной молодёжной научной конференции «XXXVIII Гагаринские чтения», Москва, 2012;

— XIII Международной научно-практической конференции «Современные проблемы гуманитарных и естественных наук», Москва, 2012;

— Всероссийском молодёжном конкурсе научных работ по современным проблемам фундаментальных и прикладных наук, Москва, 2013.

Публикация результатов работы. Основные результаты работы отражены в 10 публикациях, в том числе: три статьи опубликованы в изданиях, входящих в Перечень ВАК, и две статьи — в зарубежном издании.

Структура и объём работы. Диссертация состоит из введения, пяти глав, заключения и списка литературы, насчитывающего 145 наименований. Работа изложена на 131 страницах, содержит 31 рисунок и 12 таблиц.

Содержание работы

Во **введении** обоснована актуальность темы диссертационной работы, указаны цель и задачи исследования, научная новизна и практическая значимость исследования, описана структура диссертации.

В **первой главе** рассматриваются задача моделирования микропроцессоров, классификация симуляторов, рассматривается проблема повышения скорости программных потактовых симуляторов и анализируются существующие способы её решения.

Симулятор (англ. *simulator*) — это некоторая реализация конкретной имитационной модели микропроцессора. К основным характеристикам симуляторов микропроцессоров в настоящей работе относятся:

1) скорость — количество моделируемых тактов в секунду реального времени (Гц) или команд в секунду реального времени (к/с);
2) точность — степень соответствия симулятора объекту моделирования;
3) сложность — измеряется в количестве строк исходного кода или количестве человеко-часов, затраченных на разработку симулятора.

На разных этапах маршрута проектирования микропроцессоров применяются различные виды симуляторов, обладающие различной точностью и, как следствие, скоростью (см. рис. 1). Функциональный симулятор (англ. *functional simulator*) применяется на этапе разработки архитектуры системы команд (англ. *instruction set architecture*) микропроцессоров для проверки корректности его работы и имеет высокую скорость моделирования (точка A).

На этапе разработки микроархитектуры требуется точное определение параметров узлов микропроцессоров, влияющих на его производительность и энергопотребление, для чего применяется потактовый симулятор (англ. *cycle-accurate simulator*). В таком симуляторе моделируемое состояние расширяется микроархитектурными атрибутами, что приводит к существенному уменьшению скорости моделирования (точка B).

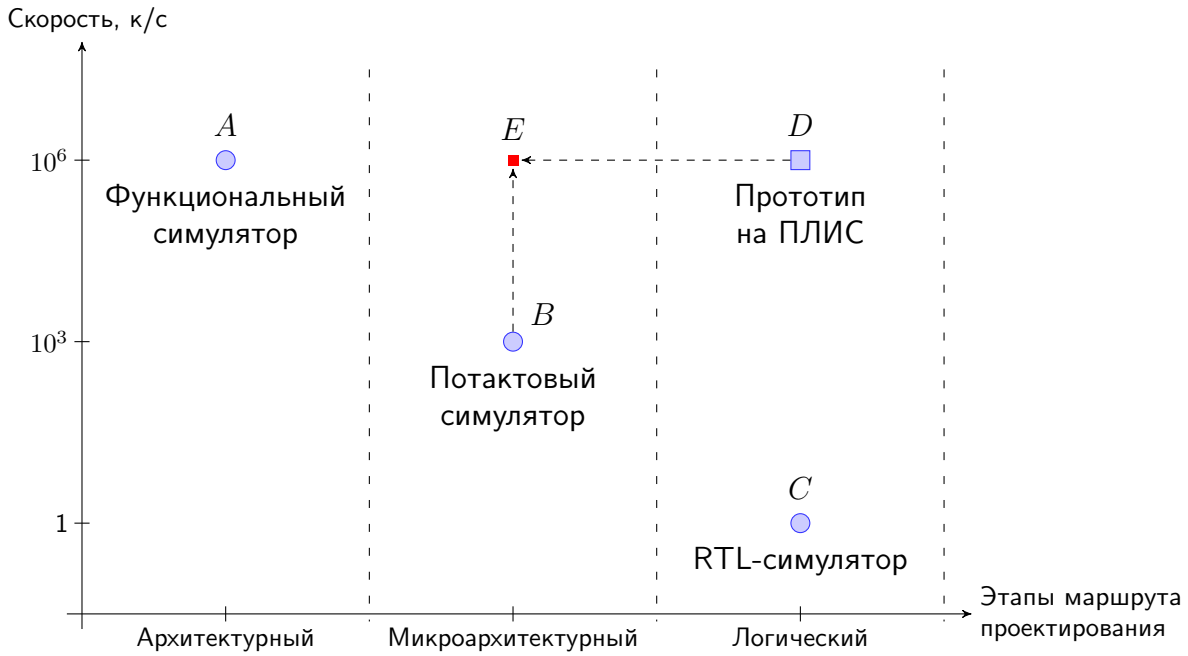

Рис. 1. Скорости различных программных (○) и аппаратных (□) моделей, применяющихся на различных этапах маршрута проектирования микропроцессора

Непосредственно перед этапом производства микропроцессоров требуется логически отлаженное описание аппаратуры на уровне регистровых передач (англ. *register transfer level, RTL*). Для отладки этого описания используются RTL-симуляторы (точка C) и прототипы на ПЛИС, которые позволяют повысить скорость за счёт применения аппаратуры (точка D).

Однако на микроархитектурном этапе RTL-описание, необходимое для применения прототипов, отсутствует. Таким образом, существует техническое противоречие между низкой скоростью программных потактовых симуляторов, используемых на данном этапе, и необходимой точностью (точка E).

Существующие способы повышения скорости потактовых симуляторов можно разделить на четыре основные группы. В табл. 1 приведены примеры реализаций способов из каждой группы с их основными характеристиками.

Сэмплинг (от англ. *sample* — выборка) объединяет способы ускорения, заключающиеся в модификации тестов, чтобы модель требовалось запускать только на репрезентативной части всей тестовой последовательности. Ошибки, связанные с длиной тестового участка, точностью его выбора и точностью инициализации состояния симулятора, приводят к ошибкам в результатах.

Абстрагированием называются подходы, направленные на модификацию модели и уменьшение степени её детализации. Примером такой модификации может служить переход от фиксации каждого такта к фиксации определённых событий (например, попадание или промах при обращении в кэш-память).

Таблица 1

Способы увеличения скорости программных потактовых симуляторов

Реализация	Архитектуры	Конвейер	Ускорение	Вносимая ошибка		Сложность
				Средн.	Макс.	
1. Сэмплинг						
SMARTS	Alpha, PISA	Полный	33,8	0,6%	2,7%	Низкая
SimPoint	Alpha	Полный	19,6	3,7%	14,3%	Низкая
2. Абстрагирование						
Interval	x86, PowerPC	Схематично	8,5	4,6%	11%	Низкая
ArcSim-JIT	ARCompact	Событийный	3,8	1,4%	5%	Низкая
CMP$im	x86	Только кэш	435	4%	17%	Низкая
3. Распараллеливание						
SlackSim	PISA	Полный	7,8	1,8%	3,9%	Низкая
Graphite	x86	Полный	5,1	3,4%	7,6%	Низкая
DARSIM	MIPS	Полный	4,6	8%	?	Низкая
4. Применение ПЛИС						
FAST	x86	Полный	4	—	—	Высокая
RAMP Gold	SPARC V8	Только кэш	15	—	—	Средняя
HAsim	Alpha, MIPS	Полный	200	—	—	Высокая

Примечание. *Ускорение и вносимая ошибка приведены относительно используемых авторами эталонов.*

Распараллеливание заключается в параллельном исполнении кода симулятора, например, на вычислительном кластере. Потактовые симуляторы обладают высокой степенью параллелизма, однако распараллеливание не даёт желаемого ускорения из-за существенных накладных расходов на барьерную синхронизацию, которую нужно производить каждый моделируемый такт. Если производить синхронизацию не каждый модельный такт, а с некоторым интервалом (англ. *slack*), то возникает потеря точности.

Таким образом, первые три группы способов ускорения потактовых имитационных моделей, несмотря на низкую сложность реализации, решают проблему низкой скорости за счёт уменьшения точности моделирования.

Наконец, четвёртая группа решений предполагает применение ПЛИС для ускорения моделирования. Использование ПЛИС позволяет эффективно задействовать массовый параллелизм модели и решить проблему синхронизации

задач путём непосредственной передачи данных между аппаратными блоками, исполняющими независимые задачи, получая ускорение на 2-3 порядка *без потери точности*.

Несмотря на то, что использование ПЛИС может существенно увеличить скорость моделирования без потери точности, их применение затруднено низким уровнем абстракции традиционных языков описания аппаратуры, гораздо более длительным циклом разработки и ограниченной логической ёмкостью применяемых ПЛИС.

Следует учесть, что увеличение времени разработки симулятора может свести на нет преимущества модели на ПЛИС, связанные с сокращением времени проведения экспериментов, следовательно, использование эффективных методов разработки и тестирования является ключевым фактором успеха.

Во **второй главе** рассматриваются особенности применения ПЛИС как симулятора, позволяющие снизить сложность по сравнению с разработкой прототипа, вводится модельное представление, приводятся сведения о применяемом в работе высокоуровневом языке описания аппаратуры Bluespec.

При применении ПЛИС для моделирования микропроцессоров конфигурация вентильной матрицы не повторяет в точности конечную электрическую схему микропроцессора, а только моделирует её поведение и временные характеристики, за счёт чего существенно снижается сложность модели по сравнению с прототипом.

В частности, один такт моделируемого микропроцессора выполняется за несколько тактов ПЛИС, и характеристика FMR (англ. *FPGA-cycles-to-model-cycles ratio*) показывает среднее количеству тактов ПЛИС, затрачиваемых на моделирование одного такта микропроцессора:

$$\mathrm{FMR} = \frac{\sum_{i=0}^{\mathsf{Такты}_{\mathsf{Модель}}} \mathsf{Такты}^{i}_{\mathsf{ПЛИС}}}{\mathsf{Такты}_{\mathsf{Модель}}}, \tag{1}$$

откуда получаем

$$f_{\mathsf{Модель}} = \frac{f_{\mathsf{ПЛИС}}}{\mathrm{FMR}} \tag{2}$$

и

$$\mathrm{IPS}_{\mathsf{Модель}} = \frac{f_{\mathsf{Модель}}}{\mathrm{CPI}_{\mathsf{Модель}}} = \frac{f_{\mathsf{ПЛИС}}}{\mathrm{CPI}_{\mathsf{Модель}} \times \mathrm{FMR}}. \tag{3}$$

Дальнейшее снижение сложности модели в данной работе происходит за счёт её разделения на отдельные модули, обычно соответствующие узлам

микропроцессоров. Множество состояний модели при этом декомпозируется на состояния n отдельных модулей:

$$S = S_{m_1} \times S_{m_2} \times \ldots \times S_{m_n},\ n \in \mathbb{N}. \tag{4}$$

Модуль не опирается на такт синхросигнала моделируемого микропроцессора, и вычисления внутри него могут быть рассмотрены как бесконечно быстрые. Отсчёт тактов модели осуществляется введением задержки на передачу сообщений между модулями в так называемых портах.

Порт представляет собой очередь сообщений определённого формата, которые передаются между модулями. Параметрами порта являются:

— задержка (англ. *latency*) — количество модельных тактов, требуемых на передачу сообщения через порт;

— пропускная способность (англ. *bandwidth*) — максимальное количество сообщений, передаваемых через порт за один такт модели.

Пользуясь терминами теории графов, модель в данном представлении, именуемом кратко МП-представлением (от слов «модуль» и «порт»), образует ориентированный взвешенный мультиграф

$$G = \langle M, P \rangle,\ m_i \in M,\ p_j = (m_{s_j}, m_{t_j}, l_j, b_j) \in P, \tag{5}$$

где вершинами m_i являются модули модели, рёбрами p_j — порты, весами рёбер $l_j \in \mathbb{N}_0$ — задержки соответствующих портов, а весами $b_j \in \mathbb{N}$ — пропускные способности (см. рис. 2).

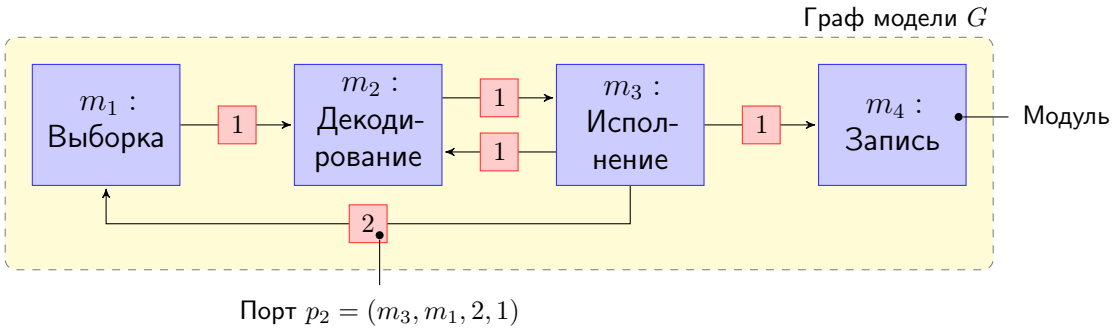

Рис. 2. МП-представление простейшего конвейера микропроцессора

Важнейшим свойством МП-представления модели является то, что любые взаимодействия между модулями происходят только через порты, что позволяет рассматривать модули независимо друг от друга.

Традиционно разработка устройств на ПЛИС осуществляется с применением языков описания аппаратуры низкого уровня, таких как VHDL,

Verilog и SystemVerilog. Эти языки дают разработчикам полный контроль над разрабатываемой аппаратурой, но их низкоуровневый код сложен и негибок.

Поскольку при разработке моделей микропроцессоров полный контроль получаемого описания не требуется, то в настоящей работе предложено использовать полностью синтезируемый объектно-ориентированный высокоуровневый язык описания аппаратуры Bluespec.

Высокий уровень абстракции сокращает время разработки, строгая проверка типов ведёт к снижению количества ошибок, а поддержка полиморфизма как мощного языкового средства позволяет создавать шаблонные модули, увеличивая повторное использование кода.

В **третьей главе** предлагается и обосновывается восходящий метод разработки аппаратных потактовых симуляторов микропроцессоров на ПЛИС. В тексте диссертации основные этапы разработки по предложенному методу подробно описаны на примере модуля кэша третьего уровня.

Поскольку в МП-представлении взаимодействие модулей осуществляется только посредством передачи сообщений через порты, то разработка и тестирование каждого модуля симулятора может выполняться независимо от других модулей, благодаря чему существенно сокращается сложность разработки.

Таким образом, разработка аппаратного симулятора является в данном методе восходящей: модули проектируются, разрабатываются, тестируются и отлаживаются, начиная с самого нижнего уровня иерархии модулей модели, в отличие от нисходящих способов, когда тестирование и отладка возможны только после создания полной модели.

Разработанный метод был реализован в работе на примере ускорения существующего программного симулятора, разработанного в инфраструктуре Asim, с использованием инфраструктуры HAsim для создания симулятора на ПЛИС, однако может быть применён и при использования других существующих или вновь разрабатываемых инфраструктур, использующих МП-представление.

Чтобы использовать преимущества ПЛИС для быстрого моделирования аппаратуры, а инструментальной ЭВМ для обработки редких, но сложных по реализации в ПЛИС событий, разрабатываемая модель создаётся гибридной: временна́я часть симулятора располагается в ПЛИС, а функциональная, уже обладающая достаточной скоростью, — в инструментальной ЭВМ.

Снижение трудоёмкости разработки гибридного симулятора осуществляется за счёт использования так называемой виртуальной платформы (англ. *vir-*

tual platform) — стандартизированного набора высокоуровневых интерфейсов, аппаратно-независимой в отличие от физической платформы, специфичной для конкретной ПЛИС.

Виртуальная платформа обеспечивает разработчика средствами для сбора статистики, отслеживания событий и вывода отладочной информации, а также предоставляет набор служб, абстрактных устройств, иерархию памяти и протокол коммуникации для различных физических платформ. Основной интерфейс между симулятором и платформой представляет асинхронный протокол RRR (англ. *remote request response*).

Первый этап разработки аппаратного симулятора по предлагаемому методу заключается в изучении исходного кода одного из модулей эталонной программной модели. На этом этапе составляется список входных и выходных портов модуля, определяются их имена, задержки, пропускные способности и форматы сообщений, изучается алгоритм работы модуля.

На *втором этапе* создаётся прототип модуля на языке Bluespec, содержащий все входные и выходные порты, а также вспомогательные компоненты, предназначенные для тестирования разрабатываемого модуля. Вспомогательные компоненты могут создаваться автоматически с помощью средств генерации кода и включают специальный служебный модуль.

В функции служебного модуля входит создание мнимого окружения для тестируемого модуля, чтение эталонных последовательностей сообщений входных портов из инструментальной ЭВМ и передача их в ПЛИС, а также сохранение последовательностей выходных сообщений модуля и передача их на инструментальную ЭВМ.

На *третьем этапе* создаётся описание модуля на языке Bluespec. Для этого может использоваться как исходный код программного симулятора, так и документация с описанием работы модуля, наличие которой является желательным, но не обязательным. Написание алгоритма работы модуля является самой трудоёмкой частью разработки: разработчик должен не только понять реализацию алгоритма на языке C++ в эталонной модели, но и написать эквивалентную реализацию на языке Bluespec.

Порты инфраструктуры HAsim в процессе реализации предложенного метода были дополнены возможностью передачи нескольких сообщений в один модельный такт, а также функциональностью фиксации и сохранения проходящих через порт сообщений, требующейся для автоматического тестирования модели.

Кроме того, стандартная библиотека языка Bluespec была дополнена собственной библиотекой компонентов, с помощью которых шаг написания модуля существенно упрощался. Разработанная библиотека включила в себя шаблоны односвязного и двусвязного списков, кэша, алгоритмы вытеснения из кэша и др.

Четвёртым этапом разработки является генерация тестовых последовательностей для модуля. Для получения тестовых последовательностей используется существующий программный потактовый симулятор (эталон), в котором исходные порты, используемые для передачи сообщений между модулями, заменяются на порты с расширенной функциональностью, позволяющие наряду с передачей сообщений между модулями сохранять поток сообщений с временными метками в специальные двоичные файлы.

Наконец, *пятым этапом* разработки является проверка модуля с использованием полученных тестовых последовательностей. Для этого к тестируемому модулю подключается служебный модуль, который считывает тестовые последовательности, полученные из эталонной модели, и записывает сообщения во входные порты тестируемого модуля. Затем выходные сообщения тестируемого модуля сравниваются с эталонными.

Четвёртая глава посвящена разработке метода и автоматизированной системы модульного тестирования, а также инструментов автоматической генерации служебного кода, позволяющих существенно сократить трудоёмкость разработки.

С формальной точки зрения, учитывая уравнение (4), восходящий способ разработки и модульное тестирование означает, что разработчику необходимо проверить только $|S_{m_i}|$ состояний для каждого отдельного модуля и $\sum_{i=0}^{n} |S_{m_i}|$ состояний для всех модулей вместо $|S_{m_1} \times S_{m_2} \times \ldots \times S_{m_n}| = \prod_{i=0}^{n} |S_{m_i}|$ состояний в случае тестирования полной системы.

Для проведения тестирования к разработанному модулю подключается специальный служебный модуль (англ. *unit stub*), который управляет сбором статистики, подаёт на входные порты последовательность сообщений из файла (журнала), сохранённого ранее при запуске эталонной модели, а также сохраняет выходные отклики модуля в отдельные файлы по портам (см. рис. 3).

В дальнейшем выходные последовательности из эталонного и разработанного модулей проверяются на эквивалентность, причём сравнение происходит не в ПЛИС, а в инструментальной ЭВМ — там же, где хранятся тестовые последовательности сообщений. Это обусловлено, в первую очередь, ограниченными

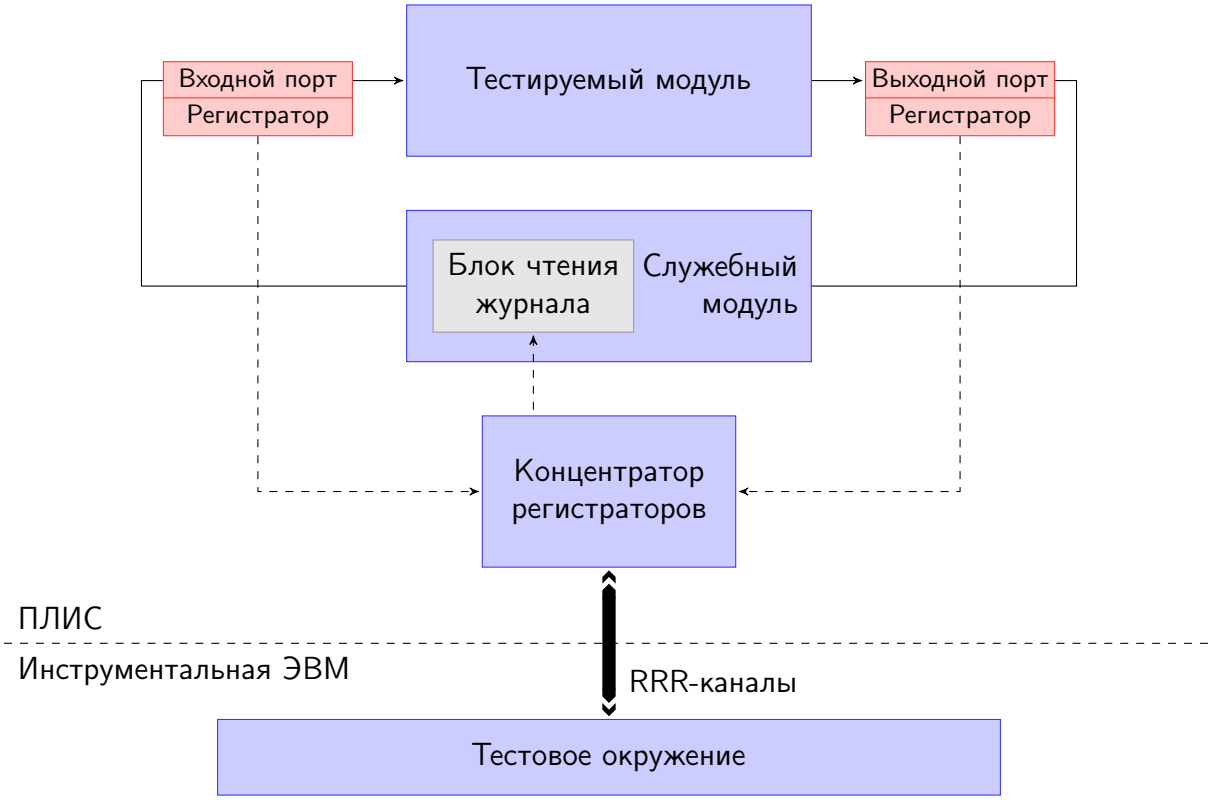

Рис. 3. Структура модели для тестирования отдельного модуля

ресурсами памяти ПЛИС: двоичный файл работы модифицированного порта может занимать несколько гигабайт.

Работа тестируемого модуля считается корректной, если на заданном множестве тестовых последовательностей двоичные трассы выходных сообщений тестируемого модуля совпадают с трассами выходных сообщений модуля эталонной модели.

Система тестирования в автоматическом режиме сравнивает последовательность выходных сообщений тестируемого модуля с эталонной. При обнаружении расхождения указывается порт, в котором было оно определено, номер модельного такта, а также отображаются эталонные сообщения и сообщения от модуля аппаратного симулятора.

Автоматизированная система тестирования существенно упрощает процесс проверки корректности работы модуля, например, автоматизируя создание служебного модуля, подключаемого к тестируемому модулю или произвольной комбинации модулей разрабатываемого симулятора.

Система тестирования автоматизирует запуск как отдельного теста, так и большого набора тестов. Отдельные тесты позволяют разработчику проверить алгоритмы работы написанного модуля, большие наборы тестов — проверить функциональность модуля в различных сценариях, собрать статистику работы модуля и измерить скорость работы симулятора.

Таблица 2

Сравнение характеристик тестового микропроцессора с аналогами

Характеристика	Тестовый МП	Intel Core	HAsim	sim-outorder	RAMP Gold
Число потоков	16	2	1	1	1
Число АЛУ	4	4	4	4	1
Кэш команд	64 кБ	32 кБ	16 кБ	64 кБ	32 кБ
Кэш данных	16 кБ	32 кБ	16 кБ	64 кБ	32 кБ
Кэш 2-го уровня	128 кБ	256 кБ	256 кБ	8 МБ	8 МБ
Кэш 3-го уровня	2 МБ	2 МБ	—	—	—

Разработанные средства генерации кода позволяют автоматически получать исходный код для следующей функциональности:

— сериализация выходных сообщений модулей программной и аппаратной моделей;

— десериализация входных сигналов модулей программной модели;

— чтение тестовых последовательностей на стороне инструментальной ЭВМ, их передача в ПЛИС;

— приём выходных последовательностей из ПЛИС и их запись на стороне инструментальной ЭВМ;

— управляющая логика служебного модуля.

Инфраструктура разработанной системы тестирования локализована, позволяя оставить код непосредственно модулей незатронутым, что делает разработанный метод тестирования масштабируемым и низкозатратным по сравнению с традиционными методами тестирования, в которых правкам подвергается код симулятора.

Пятая глава посвящена результатам применения предложенных методов и средств для разработки аппаратной модели симулятора четырёхядерного суперскалярного микропроцессора с внеочередным исполнением команд (далее — тестовый микропроцессор), характеристики которого приведены в табл. 2, а иерархия модулей — на рис. 4.

В качестве аппаратной платформы для реализации симулятора в настоящей работе был выбран инструментальный модуль Xilinx ML605 на базе ПЛИС Virtex-6 LX240T ёмкостью 241 тыс. логических ячеек. В качестве инструментальной ЭВМ использовалась рабочая станция HP Z800 на базе двух процессоров Intel Xeon X5670, работающих на тактовой частоте 2,93 ГГц.

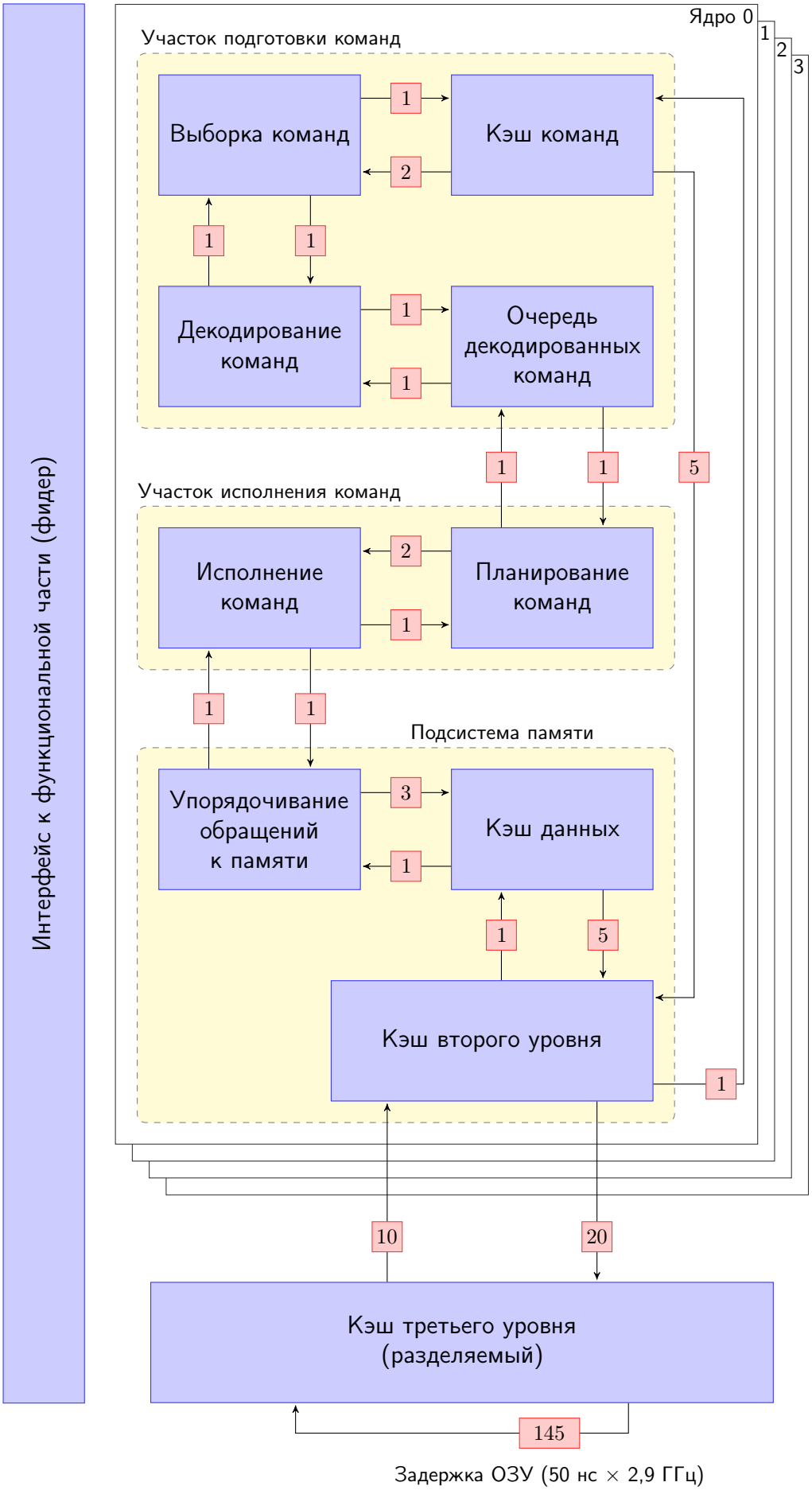

Рис. 4. Структура временно́й части разрабатываемой модели микропроцессора

Этапы логического синтеза модели выполнялись в системе автоматизированного проектирования Synopsys Synplify Pro F-2011.09, а этапы физического синтеза, размещения и трассировки — в системе Xilinx ISE 13.1.

Основной мерой сложности модели, предлагаемой в данной работе, является мера количества строк исходного кода, написанного вручную (без помощи автоматических генераторов кода), без учёта комментариев и пустых строк. Как видно из рис. 5, по количеству строк кода, написанных вручную, аппаратная модель, разработанная в рамках предложенного метода, всего в 1,4 раза сложнее программной модели. Оценка количества строк описания тестового МП на языке Verilog составляет 138 тыс., что почти в 10 раз больше.

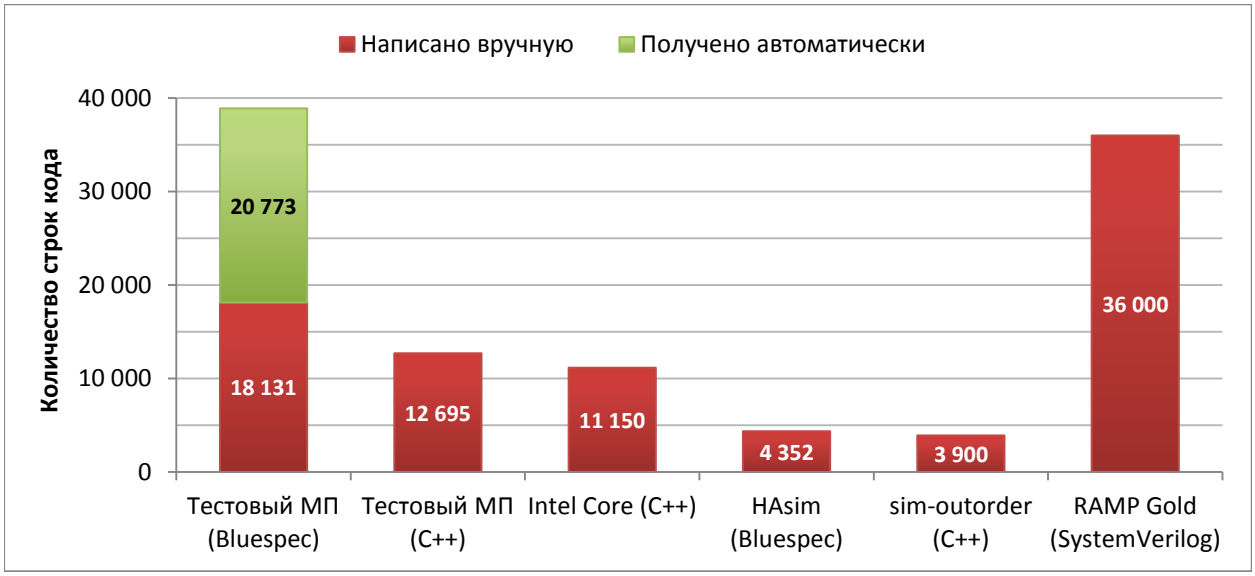

Рис. 5. Сравнение количества строк исходного кода различных симуляторов

Для оценки эффективности процесса разработки в рамках данного исследования предложено использовать модель издержек разработки COCOMO II (англ. *constructive cost model*), в которой проекты делятся на три класса: органические (разработка с мягкими ограничения), полуразделённые (смешанные ограничения), встроенные (множество жёстких ограничений).

Трудоёмкость разработки E (англ. *effort*) в человеко-месяцах как функция размера проекта S (англ. *size*) в тысячах строк кода вычисляется в рамках применяемого нами базового уровня модели по формуле $E = a \times S^b$, где коэффициенты a и b определяются классом проекта.

Рассматривая проект разработки аппаратного потактового симулятора микропроцессора как встроенный, получим следующую оценку трудоёмкости:

$$E = a \times S^b = 3{,}6 \times 18{,}1^{1{,}20} = 116 \text{ человеко-месяцев.} \qquad (6)$$

Таблица 3

Скоростные характеристики разработанного аппаратного симулятора

Модуль	FMR	$f_{\text{Модель}}$, кГц	Скорость, кк/с	Ускорение
Кэш команд	90	733	2 933	367
Выборка команд	123	537	2 146	268
Декодирование команд	19	3 474	13 895	1 737
Очередь декодированных команд	76	868	3 474	434
Участок подготовки команд	429	154	615	77
Исполнение команд	14	4 714	18 857	2 357
Планирование команд	92	717	2 870	359
Участок исполнения команд	98	673	2 694	337
Кэш данных	20	3 300	13 200	1 650
Кэш второго уровня	20	3 300	13 200	1 650
Кэш третьего уровня	14	4 714	18 857	2 357
Упорядочивание обращений к памяти	73	904	3 616	452
Подсистема памяти	172	384	1 535	192
Интерфейс функциональной части	339	195	779	97
Вся модель	482	137	548	68

Примечание 1. *Частота ПЛИС во всех моделях равна 66 МГц.*
Примечание 2. *Ускорение вычислено на основании частоты программной модели, равной 2 кГц.*

Фактическое же значение трудоёмкости разработки составило 44 человеко-месяцев, что говорит о существенной эффективности разработанных методов и средств, которые позволяют разработчику автоматизировать большинство рутинных операций и использовать существующую программную модель как эталон.

Код служебного модуля, полученный автоматически, гораздо проще кода логики самой модели, поэтому для оценки по модели COCOMO II были взяты коэффициенты для органического класса:

$$E = a \times S^b = 2{,}5 \times 20{,}8^{1{,}05} = 60 \text{ человеко-месяцев} \qquad (7)$$

Таким образом, средства автоматической генерации кода позволили сэкономить 60 человеко-месяцев, в то время как разработка самих средств генерации заняла примерно 3 человеко-месяца.

Полученные скоростные характеристики разработанного аппаратного симулятора микропроцессора, приведённые в табл. 3, показывают, что при средней скорости работы эталонного программного симулятора, равной 2 кГц, достигнуто увеличение скорости моделирования в 68 раз (см. рис. 6).

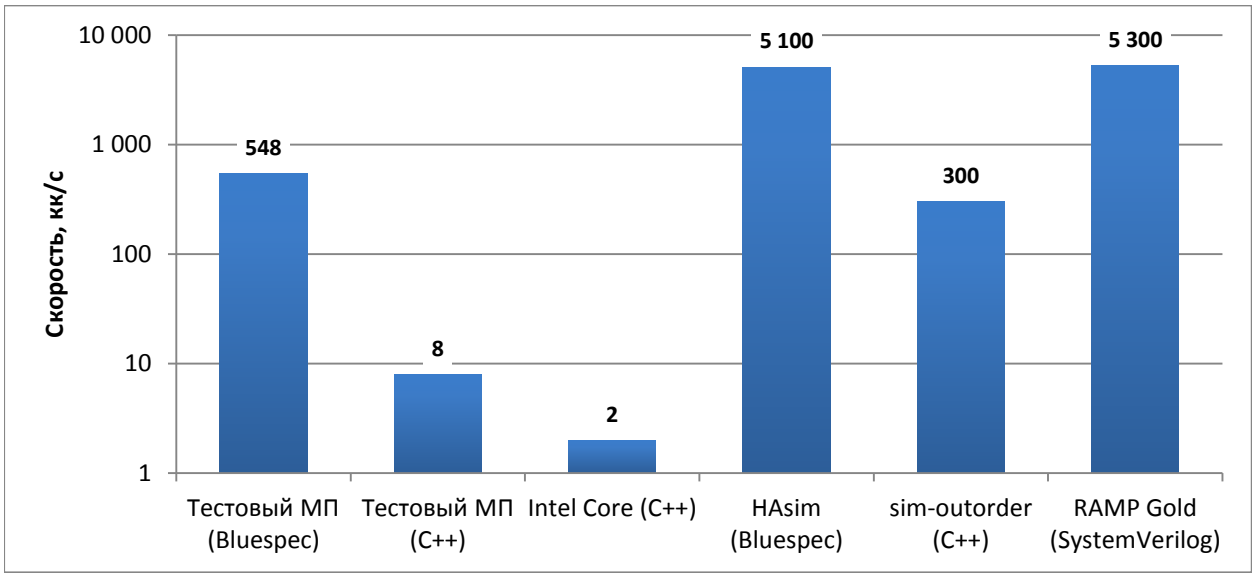

Рис. 6. Сравнение скоростей различных аппаратных и программных симуляторов

Оценка логической сложности прототипа тестового микропроцессора, основанная на масштабировании данных по функциональным блокам аналогичных проектов, даёт значение в 1,9 млн логических ячеек, что примерно в 11 раз больше, чем логическая сложность разработанной модели (см. рис. 7).

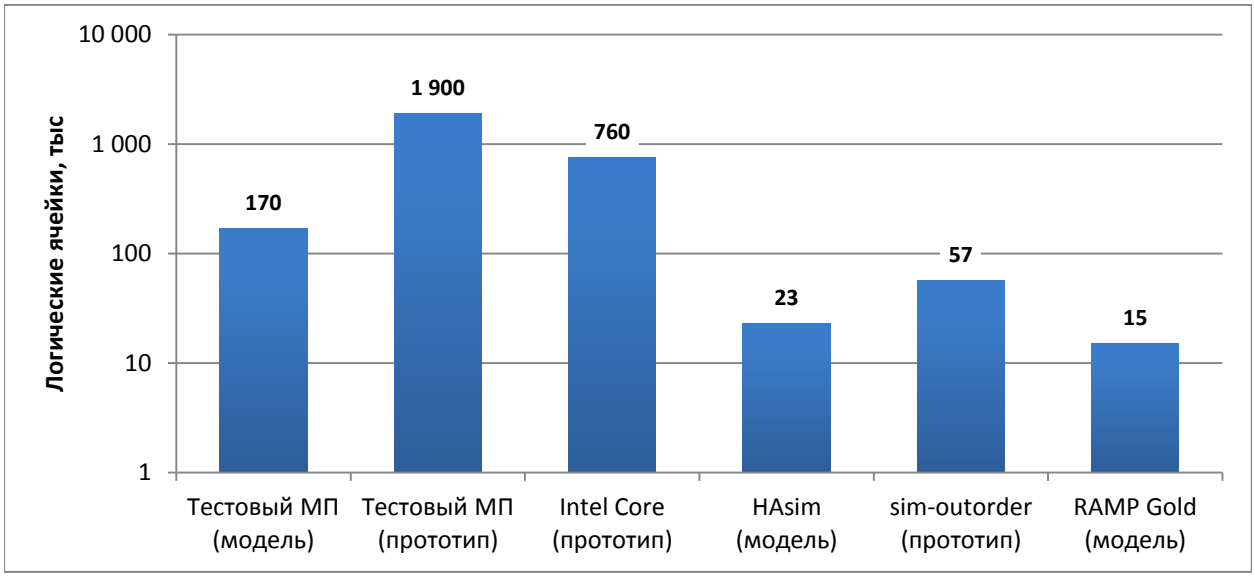

Рис. 7. Сравнение логической сложности различных моделей и прототипов

Относительно высокая скорость и низкая логическая сложность моделей HAsim, sim-outorder и RAMP Gold объясняется их более низкой точностью, поскольку они разрабатывались в исследовательских целях.

Основные результаты и выводы работы

1. Проведён анализ существующих способов повышения скорости потактовых симуляторов микропроцессоров, в частности:
   — предложена классификация способов повышения скорости потактовых моделей микропроцессоров, включая: сэмплинг, абстрагирование, распараллеливание и применение ПЛИС;
   — показано, что первые три вида способов решают проблему низкой скорости за счёт уменьшения точности моделирования, а применение ПЛИС позволяет существенно увеличить скорость моделирования без потери точности, но отличается высокой сложностью.
2. Выделены особенности применения ПЛИС для потактового моделирования микропроцессоров, позволяющие снизить трудоёмкость по сравнению с разработкой прототипа, а именно:
   — показаны преимущества разделения тактов синхросигнала ПЛИС и моделируемого микропроцессора;
   — предложено применять МП-представление, позволяющее единым образом описывать как программные, так и аппаратные модели;
   — предложено использовать для описания аппаратуры язык высокого уровня Bluespec SystemVerilog.
3. Разработан восходящий метод модульной разработки аппаратных потактовых симуляторов микропроцессоров на ПЛИС с использованием существующего программного потактового симулятора в качестве эталона, который:
   — позволяет проектировать и разрабатывать модель, начиная с самого нижнего уровня иерархии модулей;
   — позволяет существенно сократить трудоёмкость разработки;
   — допускает независимую работу коллектива разработчиков.
4. Разработаны метод и система автоматизированного тестирования аппаратных потактовых симуляторов микропроцессоров на ПЛИС, включающая в себя средства автоматической генерации кода, что позволяет:
   — тестировать и отлаживать модель на всех этапах разработки, в отличие от методов, когда тестирование и отладка возможны только после создания полной модели;
   — существенно упростить процесс тестирования симулятора за счёт автоматизации большинства операций;
   — исключить ручное написание служебного кода.

5. Проведена апробация разработанных методов и средств для разработки потактового симулятора четырёхядерного суперскалярного микропроцессора с внеочередным исполнением команд, которая показала следующие результаты:
   — трудоёмкость разработки в 3 раза ниже по сравнению с данными модели издержек COCOMO II;
   — скорость моделирования увеличилась в 68 раз по сравнению с эталонным программным потактовым симулятором;
   — использование логических ресурсов ПЛИС в 11 раз ниже по сравнению с оценкой логической сложности прототипа.

Список публикаций по теме исследования

1. *Байда Ю. В.* Программно-аппаратный симулятор процессора с векторным счетчиком инструкций на базе программируемых логических интегральных схем // Труды 53-й научной конференции МФТИ «Современные проблемы фундаментальных и прикладных наук». Часть I. Радиотехника и кибернетика. — Т. 1. — М.: МФТИ, 2010. — Ноябрь. — С. 80–81.

2. *Байда Ю. В.* Переход от программной имитационной модели микропроцессора к потактовому симулятору на базе программируемой логики // Сборник статей XI Международной научно-практической конференции «Фундаментальные и прикладные исследования, разработка и применение высоких технологий в промышленности» / Под ред. А. П. Кудинова. — Т. 3: Высокие технологии, образование, промышленность. — СПб.: Издательство Политехнического университета, 2011. — Апрель. — С. 62–64.

3. *Байда Ю. В.* Применение программного потактового симулятора микропроцессора при разработке программно-аппаратной имитационной модели на базе программируемых логических интегральных схем // Научные труды Международной молодёжной научной конференции «XXXVII Гагаринские чтения». — Т. 4. — М.: МАТИ, 2011. — Апрель. — С. 35–36.

4. *Байда Ю. В.* Задача и способы повышения скорости потактового симулятора микропроцессора // Сборник статей XIII Международной научно-практической конференции «Современные проблемы гуманитарных и естественных наук». — Т. 1. — М.: Спецкнига, 2012. — Декабрь. — С. 97–99.

5. *Байда Ю. В.* Опыт разработки аппаратного потактового симулятора микропроцессора с внеочередным исполнением команд на базе программируемой логики // Научные труды Международной молодёжной научной конференции «XXXVIII Гагаринские чтения». — Т. 4. — М.: МАТИ, 2012. — Апрель. — С. 58–60.

6. *Байда Ю. В.* Особенности применения программируемых логических интегральных схем для имитационного моделирования микропроцессоров // Инновации и инвестиции. — 2013. — № 6. — С. 146–149.

7. *Байда Ю. В.* Повышение качества проектных решений при разработке микропроцессоров путём радикального увеличения скорости имитационного моделирования // Качество. Инновации. Образование. — 2013. — № 11.

8. Методология перехода от программной потактовой модели микропроцессора к аппаратному симулятору на базе программируемой логики / Ю. В. Байда, А. В. Бутузов, А. Г. Ефимов, М. С. Цветков // Труды Московского физико-технического института (государственного университета). — 2012. — Т. 4, № 3 (15). — С. 114–122.

9. *Baida Yu., Butuzov A., Efimov A.* Method of converting a microprocessor software performance model to FPGA-based hardware simulator // Computer science and engineering. — 2013. — Vol. 3, no. 2. — P. 35–41.

10. Method and system for automated unit-level testing of FPGA-based cycle-accurate microprocessor simulator using software simulator as a golden model / Yu. Baida, A. Lechenko, A. Efimov, A. Butuzov // Computer science and engineering. — 2013. — Vol. 3, no. 3. — P. 51–55.