УДК 004.451

На правах рукописи

Мешков Алексей Николаевич

Методы и средства программного моделирования для обеспечения процесса проектирования микропроцессорных систем

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Москва – 2013

Работа выполнена в ОАО «ИНЭУМ им. И.С. Брука» и ЗАО «МЦСТ».

Научный руководитель: доктор технических наук,
старший научный сотрудник
Фельдман Владимир Марткович

Официальные оппоненты: Сухомлин Владимир Александрович,
доктор технических наук,
профессор,
МГУ им. М.В. Ломоносова, факультет ВМК,
профессор кафедры автоматизации систем вычислительных комплексов

Пакулин Николай Витальевич,
кандидат физико-математических наук,
Институт системного программирования РАН, старший научный сотрудник отдела технологий программирования

Ведущая организация: Институт проблем информатики РАН

Защита состоится **«29» мая 2013 г. в 15 ч. 00 мин.** на заседании диссертационного совета Д 409.009.01 при ОАО «Институт электронных управляющих машин имени И.С. Брука» по адресу: 119334, г. Москва, ул. Вавилова, 24.

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

Автореферат разослан « » _____ **2013 г.**

Ученый секретарь
диссертационного совета
к.т.н., профессор Красовский В.Е.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы. Создание современных высокопроизводительных микропроцессорных систем (МС) — сложный процесс, требующий больших трудозатрат. Одним из способов повышения эффективности разработки на всем протяжении проектирования является применение комплекса программ, моделирующих работу микропроцессора и МС в целом и их базовых компонентов в отдельности. Это позволяет сопоставлять альтернативы и обоснованно вырабатывать решения как при реализации новых микропроцессорных архитектур, так и при создании вычислительных средств на основе стандартных архитектурных спецификаций. Сейчас так действуют многие ведущие компании в области компьютерных технологий, хотя существенные детали их методов остаются закрытыми.

Актуальность диссертационной работы заключается в создании методов и средств, позволяющих сократить время и повысить качество проектирования отечественных высокопроизводительных МС — вычислительных комплексов (ВК) серии Эльбрус - путем их моделирования в процессе проектирования. Диссертационная работа выполнена в ходе реализации предложенного подхода при создании двух линий ВК серии Эльбрус: построенных на базе оригинальной отечественной архитектуры «Эльбрус» и на базе стандартной архитектуры SPARC. Работы проводились в проектном центре ЗАО «МЦСТ» и «ОАО «ИНЭУМ им. И.С. Брука».

Успешное решение поставленной задачи в определяющей степени зависит от выполнения нескольких принципиальных условий, связанных со степенью поддержки, которую привносит моделирование в процесс разработки. Проектные установки могут изменяться в процессе их реализации. Коррекции носят глобальный характер, в особенности на ранних этапах проекта, или ограничиваются небольшими правками ближе к его завершению, что накладывает свои условия на конфигурацию средств моделирования. Основное из них — необходимость построения структуры с несколькими уровнями иерархии и инкапсуляции объектов, позволяющей быстро модифицировать и настраивать модель при изменении компонентов архитектуры.

Важнейшее значение имеет возможность, не дожидаясь готовности аппаратуры, использовать модель создаваемой микропроцессорной системы, чтобы начать разработку программного обеспечения для нее. Понимание взаимодействия компилятора, операционной системы и приложений с аппаратурой позволяет сконцентрировать внимание разработчиков на деталях и особенностях их программных проектов, существенно влияющих на показатели

вычислительного комплекса. Для этого модель должна обладать дополнительными средствами отладки, которые позволили бы провести удобный и достаточно точный анализ состояния исполняемой программы и функциональных блоков аппаратуры, причин возникновения недопустимых или неопределенных ситуаций при их функционировании.

Важным требованием является обеспечение производительности моделирующего комплекса, которая позволила бы существенно повлиять на сроки проектирования и отладки. Хотя системы автоматизации проектирования дают возможность синтезировать и моделировать работу логики, используя коды на таких языках описания аппаратуры, как Verilog и VHDL, программные модели, разработанные на основе имеющихся спецификаций и построенные с использованием высокоуровневых языков программирования, позволяют получить на несколько порядков большую скорость. Это особенно важно при модельных запусках системных и прикладных программ сложной структуры и большого объема.

Цель диссертационной работы. Целью диссертационной работы является создание методов построения моделирующих систем, предназначенных для применения в процессе проектирования ВК на базе новых и стандартных архитектур, а также формирование отладочных средств, существенно повышающих эффективность разработки аппаратуры и программного обеспечения этих ВК. Для достижения поставленной цели были определены следующие задачи:

- исследовать существующие подходы и принципы моделирования, рассмотреть разновидности программных моделей и аналогичные работы, сформировать требования к моделирующим их комплексам;
- разработать методы построения моделирующих комплексов, применимых для широкого класса архитектур, находящихся в стадии разработки;
- спроектировать и реализовать комплекс отладочных средств, позволяющих осуществлять разработку и отладку программного и аппаратного обеспечения;
- провести апробацию предложенных методов в моделирующих комплексах с архитектурами «Эльбрус» и SPARC.

Методы исследования

Для решения поставленных задач в диссертации использовались методы и технологии системного программирования, методы математического и имитационного моделирования.

Научная новизна исследования. К составляющим научную новизну диссертационной работы решениям следует отнести:

- архитектурно-независимые методы построения моделирующих комплексов,

обеспечивающих применимость к различным компьютерным архитектурам;

- методы построения моделирующих комплексов с неоднородной архитектурой памяти с произвольной топологией, состоящих из изменяемого числа процессоров;
- методы описания базовых функций периферийных устройств, шин и интерфейсов, позволяющих стандартизовать разработку моделей периферийных устройств;
- разработка комплекса отладочных средств, позволяющих ускорить и упростить отладку программного обеспечения.

Практическая ценность и результаты работы

Практическая ценность работы состоит в создании, на основе предложенных методов, моделирующих комплексов с архитектурами «Эльбрус» и SPARC, разрабатываемых в ЗАО «МЦСТ» и ОАО «ИНЭУМ им. И.С. Брука».

Моделирующие комплексы применялись для исследования архитектуры, написания и проверки архитектурных тестов в форме программ на ассемблере и языке Си, создания генераторов тестовых программ, а также разработки и отладки системного и прикладного программного обеспечения при разработке микропроцессоров Эльбрус-S, Эльбрус-2С+, Эльбрус-4С, МЦСТ-R500S, МЦСТ-R1000 и ВК серии Эльбрус на их основе. В процессе создания моделирующих комплексов были проведены дополнительные проверки документации и алгоритмов работы микропроцессорных систем на первоначальном этапе разработки до получения изготовленных микропроцессоров.

Предложенный модульный подход к построению моделирующих комплексов, примененный в процессе разработки, показал свою востребованность в условиях параллельной разработки нескольких моделей микропроцессоров. При этом эффективность разработанных моделирующих комплексов сравнима с аналогичными виртуальными машинами, что было подтверждено измерениями производительности на нескольких наборах задач.

Апробация работы

Результаты работы докладывались на различных конференциях и семинарах:

- XXXV Международная молодежная научная конференция «Гагаринские чтения» (Москва, МАТИ, 2009 г.);
- международная научная конференция, посвященная 80-летию со дня рождения академика В.А.Мельникова (Москва, МИАН, 2009 г.);
- 52-ая и 55-ая научные конференции МФТИ (2009, 2012 гг.);
- седьмые научные чтения памяти М.К. Тихонравова (4 ЦНИИ МО РФ, 2009 г.);
- семинар отдела «Технологий программирования» института системного

программирования РАН (ИСП РАН, 2011 г.);

- конференция, посвященная 55-летию со дня образования НИИ автоматической аппаратуры им. В.С. Семенихина (2011 г.);
- конференция, посвященная 60-летию Московского физико-технического института (2011 г.).

Публикации

По теме диссертационной работы опубликованы 10 печатных работ, в том числе 4 в издании, рекомендованном ВАК РФ.

Структура и объём диссертации

Диссертация состоит из введения, четырёх глав с выводами, заключения. Основная часть работы изложена на 146 страницах, содержит 47 рисунков. Библиографический список составляет 66 наименований.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении дана общая характеристика работы, раскрыта ее актуальность, сформулированы основные задачи и цель исследования, определены научная новизна и практическая значимость полученных результатов, а также основные результаты, выносимые на защиту.

В первой главе анализируются и исследуются принципы моделирования, а также разновидности моделей. По используемым подходам к построению, модели можно разделить на несколько основных категорий, достоинства и недостатки которых представлены в табл. 1:

Таблица 1:

Подходы к построению моделей

	Достоинства	Недостатки
аналитическое моделирование	1) простота 2) возможность быстрого получения оценочных результатов	1) очень низкая точность
статистическое моделирование	1) относительная простота 2) возможность получения статистических результатов	1) низкая точность 2) обеспечивает качественные оценки только для стационарных режимов 3) необходимость задания статистических параметров
аппаратное моделирование	1) высокая скорость 2) очень высокая точность	1) трудоемкость разработки 2) высокая стоимость 3) невысокая гибкость 4) создание возможно после окончания разработки устройства

программное моделирование	1) точность не ограничена и определяется требуемой детальностью 2) возможность получения количественных результатов 3) гибкость	1) при повышении детальности скорость снижается 2) трудоемкость разработки

Как видно из таблицы, программное моделирование обеспечивает высокую детальность и, в отличие от первых двух методов, может дать не только качественные, но, что более важно, и количественные характеристики моделируемой аппаратуры. В отличие же от аппаратного, программное моделирование является более доступным и гибким средством, подходящим для решения широкого круга задач. Уровень детализации и точность приближения полученной таким образом модели не ограничен и определяется конкретными задачами, для решения которых создается модель.

Программные модели можно разделить на 2 категории. Первую категорию представляют *симуляторы вычислительного комплекса (full machine simulator)* или *системные виртуальные машины (virtual machine)*. Они целиком моделируют целевой компьютер, включая процессор, память и периферийные устройства, то есть предоставляют неотличимое от реального окружение, необходимое для исполнения системного программного обеспечения, такого как программа начальной загрузки, операционная система и т.д. Такие моделирующие комплексы позволяют решать гораздо более широкий спектр задач, чем многие другие средства, основанные, например, на анализе исходных кодов или их повторной компиляции.

Основными функциональными компонентами симулятора ВК являются модель одного или нескольких микропроцессоров, модели системной шины, оперативной памяти и периферийных устройств.

На рис. 1 изображена схема, иллюстрирующая взаимодействие между симулятором ВК и инструментальным комплексом. Целевые (гостевые) прикладные программы (application program) и операционная система (target OS) формируют гостевое программное обеспечение (target software). Инструкции целевых программ интерпретируются симулятором ВК (full machine simulator), представляющим собой целевую машину (target machine), включающую модель микропроцессора (target microprocessor) и модели устройств, формирующих структуру вычислительной системы (memory, peripheral devices, communication, images). Сам симулятор представляет из себя прикладную программу, исполняющуюся на инструментальной платформе (host machine) под инструментальной операционной системой (host OS).

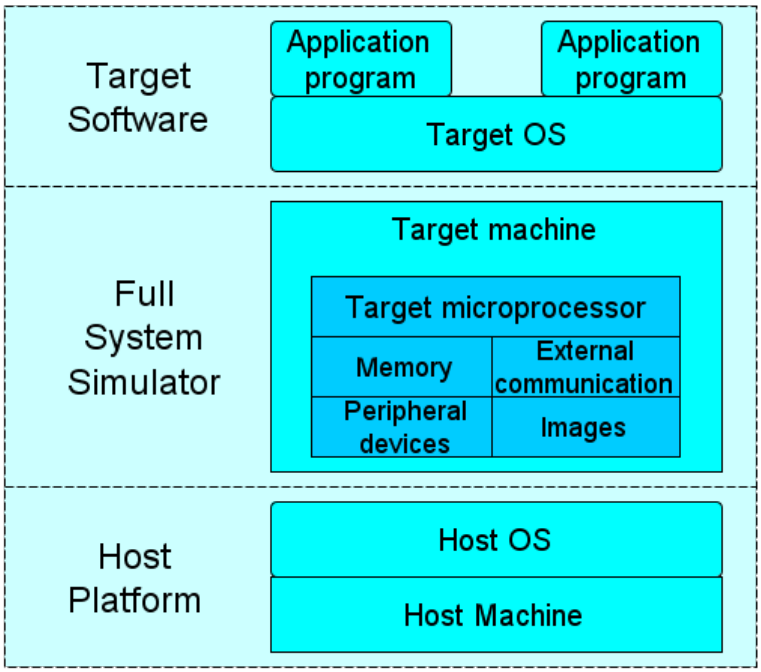

Рис. 1: Схема организации симулятора ВК

Задача разработки симулятора ВК системы является более сложной, чем разработка других средств анализа производительности. С другой стороны, данная модель может использоваться для таких целей как:

- анализ поведения задач, основанный на изменении поведения системы (например, сбор статистики по событиям, недоступный на реальной аппаратуре);
- возможность исполнения программного обеспечения в период, когда аппаратура еще не реализована;
- разработка и отладка операционных систем и системного программного обеспечения (симулятор обеспечивает воспроизводимость запусков и обладает широким спектром средств для отладки исполняемого кода и анализа его производительности);
- анализ особенностей работы различных архитектурных решений и их вклада в изменение производительности системы.

Вторую категорию симуляторов представляют *симуляторы пользовательского уровня (user level simulator)* — они позволяют моделировать поведение пользовательских процессов, эмулируя системные вызовы, исполняемые на целевой (моделируемой) системе, средствами инструментальной (той, на которой запущен симулятор). Модель пользовательского уровня предоставляет прикладным программам такое же окружение, как если бы они выполнялись под управлением операционной системы, для которой они скомпилированы. Основными функциональными компонентами симулятора пользовательского уровня являются модель

процессора, загрузчик исполняемых файлов (ELF loader), обработчик системных вызовов (system calls wrapper) и модель памяти пользовательского приложения (application memory);

На рис. 2 изображена схема организации симулятора пользовательского уровня (user level simulator) и его взаимодействия с исполняемой программой (application program), представляющей собой целевое программное обеспечение (target software). Инструкции, в частности, системные вызовы интерпретируются симулятором, предоставляющим приложению интерфейс операционной системы. Для этого в его состав включена модель микропроцессора (target microprocessor) и ряд дополнительных компонент, описанных выше. Также как и в случае симулятора ВК, симулятор пользовательского уровня, является прикладной программой, исполняющейся на инструментальной платформе (host machine) под инструментальной операционной системой (host OS).

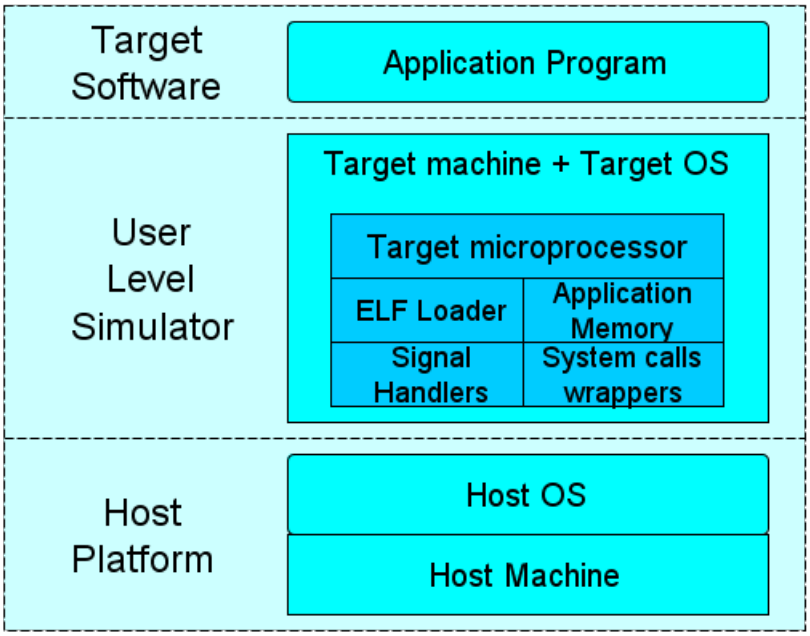

Рис. 2: Схема организации симулятора пользовательского уровня

Другим способом классификации программных моделей является разделение их на *функциональные* и *потактово-точные (потактовые)*. Последние моделируют аппаратуру с учетом временных задержек, в то время как функциональные их не учитывают. Потактовые модели с точки зрения аппаратуры являются более детальными, они позволяет измерять производительность и качество оптимизации приложений, но устроены более сложно — для точной потактовой работы необходимо моделировать конвейер микропроцессора и временные задержки в подсистеме памяти, а значит скорость моделирования в этом случае ниже, а трудозатраты на разработку - выше.

Для получения оценки эффективности современных моделирующих комплексов был

проведён анализ некоторых распространенных симуляторов ВК. Их сравнительные характеристики приведены в табл. 2.

Таблица 2:

Сравнение характеристик различных симуляторов.

	Кроссплатформенный	Динамич. трансляция	Многопроцессорные конфигурации	Моделируемые архитектуры	Область применения	Лицензия	Применимость при разработки архитектуры
SimICS	да	есть	есть	Alpha, ARM, x86, MIPS, PowerPC, POWER, SPARC и др.	Отладка программного обеспечения и системная интеграция	Проприетарная	частично
Qemu	да	есть	есть	x86, ARM, SPARC, PowerPC, MIPS	Разработка ПО, использование в качестве рабочей станции, сервера	GNU GPL	нет
Bochs	да	нет	есть	x86	Разработка ПО, безопасное низкоуровневое исследование и отладка, дизассемблирование	LGPL	нет
M5	да	нет	есть	SPARC, Alpha	Исследование архитектур компьютерных систем, процессорных микроархитектур	BSD License	нет
Shade	нет	есть	нет	SPARC, MIPS	Использование в качестве виртуальной машины и отладка приложений	OTN Developer License	нет
Strata	да	есть	нет	SPARC, MIPS, x86	Исследование систем двоичной трансляции	Не распространяется свободно	нет
FSS	да (jvm)	нет	есть	SPARC	Тестирование и верификация дизайна	Распространяется свободно	частично
Legion	нет	нет	есть	SPARC	Исследование и разработка программно-аппаратного и программного обеспечения	GNU GPL	частично

Существует ряд других достаточно известных симуляторов ВК (VMware, VirtualBox, DOSBox, SimOS), однако они либо являются проприетарными, либо предназначены только для моделирования x86 приложений на x86 платформе и не подходят для решения задач, поставленных в данной работе.

Другой важной особенностью является их ограниченная применимость для целей отладки программного обеспечения и аппаратуры. Сравнительные характеристики отладочных возможностей перечисленных выше и некоторых других симуляторов приведены в табл. 3.

Таблица 3:

Отладочные возможности различных симуляторов

	Трассировка	Слежение за содержимым регистров/памяти	Контрольные точки	Сбор статистики
SimICS	да	да	да	да
Qemu	да	нет	да	нет
M5	да	нет	да	нет
Shade	да	нет	нет	да
FSS	да	нет	нет	да
Legion	да	нет	да	да

Данные, приведенные в таблице, показывают, что большая часть рассмотренных симуляторов не обладают достаточным набором средств, необходимых для отладки программного и аппаратного обеспечения. Следует отметить, что в подавляющем большинстве работ производится моделирование уже существующих микропроцессоров и МС (например x86, ARM, SPARC). Основной целью разработки подобных моделей является эмуляция различных архитектур, защита информации, безопасное исследование, отладка и дизассемблирование программного обеспечения. То есть, класс задач, решаемых такими моделями, значительно уже того, которым должны обладать модели, изначально спроектированные для целей отладки аппаратуры и программного обеспечения. Это сильно затрудняет возможность приспособления стороннего программного обеспечения для моделирования МС собственной оригинальной разработки. Таким образом, задача создания программной модели для разрабатываемых ВК является актуальной.

На основании проведенного исследования в диссертационной работе ставится задача создания программной модели разрабатываемых ВК, обладающей широким набором отладочных средств.

Во второй главе формулируются требования к построению моделирующих комплексов, применимых как к новым, так и существующим архитектурам.

С принципиальной точки зрения, моделирование в процессе проектирования решает две основные задачи:

выбор архитектурных решений, дающих оптимальный или, по крайней мере, приемлемый результат при заданных условиях и ограничениях проектирования;

возможность на ранних стадиях проектирования приступить к созданию и отладке программного обеспечения разрабатываемых вычислительных систем.

Ключевым требованием при решении первой задачи является *гибкость* модели, особенно существенная в случае создания высокопроизводительных МС с многокомпонентным и сложным оборудованием, что накладывает свои условия на систему моделирования. Основное из них — необходимость построения моделирующей структуры с несколькими уровнями иерархии и инкапсуляции объектов, соответствующих реальным аппаратным модулям. Это позволяет автономно настраивать элементы модели в процессе архитектурных исследований и модифицировать ее общую конфигурацию соответственно изменению конфигурации комплекса. Интерфейс между модулями следует проектировать в абстрактной форме, что позволяет объединять их в различных комбинациях для достижения требуемых свойств.

Необходимым инструментом при решении второй задачи является наличие *средств диагностики и отладки*, таких как дизассемблирование исполняемой программы, трассировка, мониторинг состояний блоков и других. Очень важной функцией модели является контроль возникновения недопустимых или неопределенных ситуаций, вызванных действиями создаваемого программного обеспечения. Он позволяет выявлять неочевидные программные ошибки, возникновение которых по различным причинам не контролируется аппаратурой.

Другим важным условием разработки программного обеспечения с использованием модели создаваемого ВК является *воспроизводимость* ее поведения. Это означает, что при заданных параметрах запуска и исполнения программы поведение системы должно быть строго детерминировано, то есть, обеспечивается повторяемость ее работы.

Ввиду взаимосвязи обоих задач, как правило, достаточно тесной, следует привести еще несколько общих требований, в равной степени актуальных для всей системы моделирования. В первую очередь, это *точность* и *производительность* модели. В этом аспекте под *точностью* имеется в виду степень соответствия реализации модели целевой архитектуре. Она должна быть достаточна как для опытного исполнения большого числа

различных программ, включая создаваемое системное программное обеспечение, так и для того, чтобы приступить к разработке программ, предназначенных для верификации аппаратуры – от процессоров до периферийных устройств и интерфейсов.

Производительность моделирующего комплекса обычно подразумевает количество исполняемых инструкций целевой архитектуры в секунду. От производительности в первую очередь зависит возможность отладки на модели первоочередных задач большого объема и сложности, таких как загрузка ОС, которая требует исполнения нескольких десятков или даже сотен миллионов инструкций. На производительность моделирующего комплекса оказывают влияние подходы к реализации гибкости и точности модели, варианты применения технологий интерпретации и компиляции, механизмов сохранения и загрузки состояния модели и другие факторы. Их выбор должен обеспечить приемлемую продолжительность исполнения во всех вариантах запуска моделирующих программ, сопутствующих проектированию.

Далее в главе рассматриваются две архитектурные линии - «Эльбрус» (архитектура VLIW/EPIC) и SPARC (архитектура RISC), используемые для построения высокопроизводительных ВК серии Эльбрус. Наличие двух линеек микропроцессоров и ВК на их основе с различающимися архитектурами, предполагает разбиение моделирующего комплекса на архитектурно-зависимую и архитектурно-независимую части. Кроме того, даже в пределах одной линейки микропроцессоров архитектура МС системы может существенно различаться. Так, например, в каждой из линеек присутствуют микропроцессоры как с архитектурой симметричной мультипроцессорности (Symmetric Multiprocessing - SMP), так и с архитектурой неоднородного доступа к памяти (Non-Uniform Memory Access - NUMA). Принадлежность процессоров к различным архитектурам представлена в табл. 4.

Таблица 4:

Архитектура микропроцессоров Эльбрус и SPARC

	Архитектура «Эльбрус»	Архитектура SPARC
Архитектура SMP	Эльбрус-3М	МЦСТ-R500S
Архитектура NUMA	Эльбрус-S Эльбрус-2С+ Эльбрус-4С	МЦСТ-R1000

На основе сформулированных требований и описания объекта моделирования разработана функциональная схема моделирующего комплекса, используемая для проектирования и разработки новых МС (рис. 3).

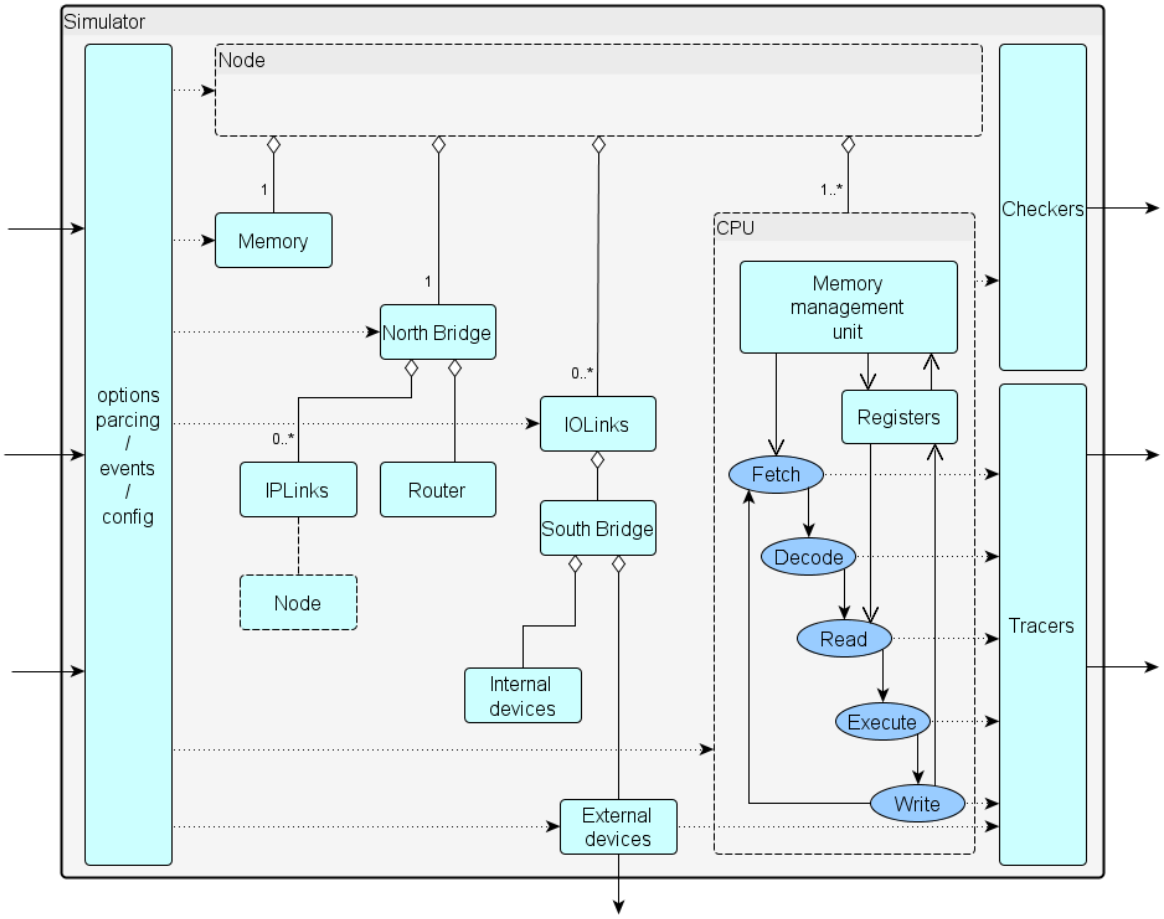

Рис. 3: Функциональная схема моделирующего комплекса

Модулем верхнего уровня является класс Simulator, содержащий все остальные модули и отвечающий за разбор ресурсов и предоставление возможности трассировки всем дочерним классам. В зависимости от настроек конфигурации, может создаваться несколько процессорных модулей, представляющих микросхему «системы на кристалле» (Node). Каждый такой модуль, в свою очередь, содержит некоторое количество объектов процессоров (CPU), «северный мост» (North Bridge), контроллеры и модули оперативной памяти принадлежащей данному узлу (Memory) и каналы ввода-вывода (IOLinks).

Основными исполнительными блоками моделирующего комплекса являются процессорные ядра. Их число определяется архитектурой «системы на кристалле» и они, в свою очередь, имеет модульную архитектуру. В качестве модулей выступают основные устройства процессора такие как устройство управления памятью (Memory Management Unit), регистры (Registers) и другие. Основную часть времени ядро исполняет инструкции моделируемой архитектуры, выполняя в цикле последовательность (Fetch-Decode-Read-Execute-Write).

Все модули должны поддерживать возможность конфигурирования и периодического исполнения заложенной в модуль функциональности. Это реализуется с помощью блоков

инициализации и событий (options parcing/events/config). Кроме того, большая часть устройств должна поддерживать возможность трассирования последовательности изменения своего состояния. Такую функциональность для каждого из устройств обеспечивает модуль Tracer.

В третьей главе представлены разработанные методы, применяемые при построении моделирующих комплексов.

В начале главы предложена общая технология разработки – *маршрут моделирования*, на базе которого должно происходить построение моделей любого уровня. Схематично последовательность действий изображена на рис. 4.

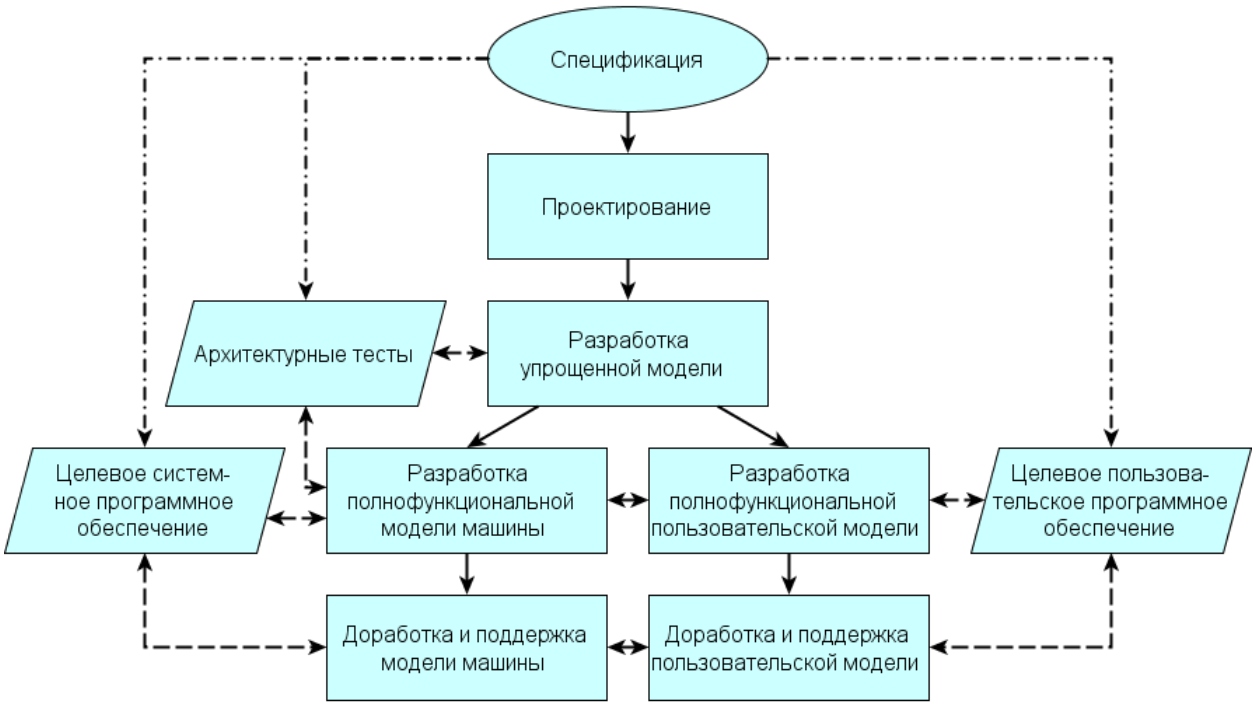

Рис. 4: Маршрут моделирования

Основой разработки модели служит документация на разрабатываемую архитектуру - набор команд, микро- и макроархитектурные особенности реализации системы.

На первом этапе на основе этой документации проектируется каркас будущей модели - базовые типы и структуры данных, регистры, модель конвейера процессора, иерархия модулей процессора и ключевые вспомогательные модули симулятора.

На втором этапе реализуется «легковесная» модель. Она должна включать базовую функциональность, необходимую для корректного исполнения архитектурных тестов – разработанные базовые модули и модель конвейера, реализацию базисных инструкций (целочисленная арифметика, инструкции передачи управления, основные инструкции

доступа к памяти, доступ к регистрам), упрощенную модель памяти, возможность добавления и управления трассировкой, реализацию загрузки тестов в память. Данная модель отлаживается на архитектурных тестах и в то же время сама служит эталонной моделью при разработке тестов. На данном этапе возможны существенные корректировки всей иерархии.

На третьем этапе разрабатывается полнофункциональная модель. Создается окончательная модель процессора с полным набором инструкций, включающая модели всех устройств. Происходит разветвление процесса разработки на два направления — модель микропроцессорной системы и модель пользовательского уровня. Для первого разрабатывается окружение, представляющее набор устройств, формирующих вычислительную систему — подсистема прерываний, модель внешней памяти, модели периферийных устройств и их окружения. Также возможна реализация многопроцессорных конфигураций, межпроцессорного взаимодействия в соответствии с архитектурой памяти (SMP, NUMA), поддержка протокола когерентности. Для модели пользовательского уровня разрабатывается модель памяти пользовательского приложения, загрузчик исполняемых файлов, обработчик системных вызовов. Кроме того, на данном этапе разрабатывается основная часть отладочных средств — реализация слежения за ресурсами моделируемой системы, реализация контрольных точек, механизмы управления работой и трассировкой по событиям и т.д. Отладка каждой из моделей также производится с использованием различных средств. Модели микропроцессорной системы отлаживаются на доступном системном программном обеспечении — программе начальной загрузки, операционных системах и полном наборе архитектурных тестов, в то время как модели пользовательского уровня отлаживаются на программах, скомпилированных для данной архитектуры и поддерживаемой операционной системы.

Дальнейший жизненный цикл модели состоит в ее доработке и сопровождении. Он включает внесение обновлений документации в модель, оптимизацию работы, разработку дополнительных отладочных возможностей, таких как сбор статистики работы процессора и устройств, средств выявления неопределенных ситуаций, не контролируемых аппаратурой, регистрацию обработчиков сигналов и т.д. Для отладки используется то же программное обеспечение, что и на третьем этапе.

В главе описываются различные особенности и методы, применяемые при построении моделирующих комплексов. Ключевым устройством, определяющим архитектуру системы, является ядро микропроцессора, его построению должно быть уделено особое внимание. Рассмотрены как архитектурно-независимые, а так архитектурно-зависимые методы. Первые применимы к обеим линейкам МС, рассмотренным в работе, а также, потенциально, и для

любой другой архитектуры. Сюда можно включить методы, описывающие организацию хранения, декодирования и исполнения инструкций, реализацию регистрового файла. Несмотря на то, что эти методы часто упоминаются в литературе, возможность их реализации в общем виде для различных архитектур не упоминается. Архитектурно-зависимые методы рассматриваются в основном применительно к уникальным свойствам архитектуры «Эльбрус» и охватывают реализацию в системе моделирования таких особенностей, как механизм защиты адресной информации, вторичное адресное пространство и организация памяти в моделях пользовательского уровня.

Помимо микропроцессорной архитектуры важным элементом является архитектура подсистемы памяти. В основу разработок ВК серии Эльбрус положен принцип использования многопроцессорности. Принцип симметричного мультипроцессирования, используемый в части разработок, имеет ограничение по числу процессоров, обусловленное возрастающей нагрузкой на шину, объединяющую процессоры друг с другом и памятью. Это особенно актуально для архитектуры «Эльбрус», где даже один процессор оказывает большую нагрузку на подсистему памяти. Данное ограничение можно отчасти снять построением NUMA-систем. Эта схема лежит в основе новейших ВК серии Эльбрус. NUMA-системы строятся на основе нескольких узлов - «систем на кристалле», каждая из которых содержит одно или несколько процессорных ядер и собственную локальную память. Таким образом, каждый процессор имеет доступ к локальной памяти, расположенной в одном с ним узле, и к удаленной памяти, расположенной в узле не принадлежащем данному процессору.

В рамках создания моделей таких ВК, был разработан способ построения моделей с неоднородным доступом в память. Разработка такого механизма является следствием того, что рассматриваемые в литературе моделирующие комплексы строятся в большинстве своем по схеме SMP, являющейся более простой с точки зрения программной реализации. Удачных примеров реализации программных NUMA систем встречается значительно меньше, а детали их реализации практически не описаны. С функциональной точки зрения, моделирующий комплекс с архитектурой NUMA представляет собой многопроцессорную систему, включающую несколько моделей «систем на кристалле», именуемых «узлами». Каждый узел представляется отдельным объектом, обладающим помимо внутренних структур, несколькими модулями, позволяющими получить доступ к таким же объектам. Полное число узлов и правила соединения их друг с другом определяется параметрами запуска модели. Модули связи являются программным представлением каналов доступа, соединяющих узлы. Таким образом, топология системы задается путем настройки связей между модулями в каждом из узлов. Это позволяет создавать практически любые

комбинации соединений.

Для поддержки NUMA–архитектуры в моделях были реализованы доступные системному программному обеспечению представления системных коммутаторов микропроцессоров «Эльбрус-S» и «МЦСТ-R1000». Помимо основной функциональной нагрузки, в модель системного коммутатора также был встроен механизм сбора статистики о событиях, характерных для NUMA систем. Его суть состоит в раздельном подсчете количества запросов в память, приходящих из различных направлений — из локального процессора, чипсета, из удаленных узлов и т.д. Для оптимальной работы число запросов, обращающихся вне своего узла, должно быть минимизировано, поскольку они выполняются дольше локальных. Данный механизм позволяет на основе анализа содержимого отладочных регистров, оценить успешность произведенного распределения ресурсов.

Для построения моделирующего комплекса всей микропроцессорной системы необходимо помимо модели «системы на кристалле» разработать также модели контроллеров периферийных устройств. Принципы и методы, используемые при их создании, сильно отличаются от тех, что используются при разработке моделей процессоров. Набор периферии никак не привязан к конкретной архитектуре, так что процессоры с различными версиями одной архитектуры могут входить в состав комплексов с разными контроллерами периферийных устройств, а процессоры с различной архитектурой — с одинаковым набором устройств. Традиционно, большинство существующих моделирующих комплексов используют близкий к стандартному набор устройств, минимально необходимый для корректной работы ОС. Такой подход не оправдывает себя, когда необходимо разрабатывать модели периферийных устройств собственного производства. В этом случае, необходимо обеспечить возможность разработки большого многообразия зачастую незначительно различающихся устройств. Предложены методы разработки общих базовых классов и моделей ряда периферийных контроллеров на основе принципов модульности и инкапсуляции. Рассмотрим в качестве примера программную реализацию устройств с интерфейсом шины PCI. Все устройства, имеющие этот интерфейс характеризуются схожей частью функциональности, определенной в спецификации шины. Примером может служить конфигурационное пространство — набор регистров фиксированного размера, за некоторыми из которых закреплен определенный смысл и функциональность. Другой пример — единый для всех способ указания на наличие внутренних ресурсов - регистров или памяти, которые должны быть расположены в адресном пространстве памяти машины и также единый механизм задания программным обеспечением адресов для описанных ресурсов. Наличие базовых классов реализующих такую функциональность, позволяет

облегчить работу при создании нового устройства — требуется лишь описать значения регистров и права доступа к ним и добавить в нужном месте вызов функций базового класса. После этого требуется реализовать уникальную функциональность создаваемого устройства, но наличие четко прописанного интерфейса упрощает и эту задачу.

В табл. 5 приведены характеристики разработанных моделей. Из приведенной таблицы видно, что архитектурно-независимая часть кода составляет значительную составляющую объема кода в сравнении с архитектурно-зависимой. Таким образом, выделение архитектурно-независимых инструментов, функций и классов представляется полезным, так как избавляет от необходимости разработки большого объема функциональности для каждой новой модели, что приводит к сокращению времени разработки вновь создаваемых моделей.

Таблица 5:

Характеристики разработанных моделей

Архитектура	Название вычислительного комплекса	Архитектурно-независимая часть	Архитектурно-зависимая часть
Эльбрус-3М	Эльбрус-3М1, УВК/С	34171 строк	138730 строк
Эльбрус-S	Эльбрус-3С		125650 строк
R500/R500S (SPARC v8)	Эльбрус-90микро		24958 строк
R1000 (SPARC v9)	Эльбрус-90С		40287 строк

В конце главы приводятся результаты анализа производительности для пользовательских моделей и моделей системы для различных архитектур. Для измерения производительности моделей микропроцессорной системы в качестве системного программного обеспечения использовалась ОС linux, портированная на архитектуры «Эльбрус» и SPARC. Кроме того, поскольку система двоичной трансляции, разработанная для ВК с архитектурой «Эльбрус», позволяет выполнять произвольные двоичные коды для архитектуры IA-32, также измерялась производительность при моделировании работы ОС Windows. Для измерения производительности моделей пользовательского уровня, использовались тесты из пакета SPEC, а также пакета векторных тестов для архитектуры SPARC.

Как упоминалось во введении, в подавляющем большинстве доступные на рынке инструменты моделирования не предназначены для разработки и отладки программного обеспечения, поэтому потребовалось разработать моделирующий комплекс, способный

облегчать и ускорить решение этих задач. **В четвертой главе** описываются инструменты, предназначенные, прежде всего, для отладки программ, запускаемых на моделирующих комплексах.

Задача отладки приложения чаще всего сводится к исследованию потока выполнения. Для ускорения этого процесса в моделирующем комплексе заложены *механизмы трассировки.* В большинстве случае оказывается полезной трасса потока команд, выполняющихся на процессорном ядре. Кроме исполнительных устройств, определяющих набор инструкций, любой процессор содержит ряд дополнительных устройств, также оказывающих влияние на ход выполнения программы. Для этого реализована трассировка изменения внутреннего состояния устройств микропроцессора. И наконец, процессор всегда работает в составе вычислительной системы, включающей в свой состав набор периферийных шин и устройств. Поэтому для полноты понимания поведения всей системы требуется иметь возможность слежения за изменением их состояния. Все описанные виды трассировки реализованы в составе моделирующих комплексов. В случае, когда не требуется подробное изучение, а надо выяснить изменение состояния устройства в результате входного воздействия, оказывается полезным *механизм слежения за ресурсами* — регистрами или ячейками памяти. Вывод информации происходит только в момент изменения наблюдаемого ресурса. Использование данной возможности оказывается критичным для длительных интервалов выполнения, когда объем обычной трассировки оказывается слишком большим.

Для целей оценки производительности и оптимизации приложений часто требуется анализ статистики о возникновении разного рода событий. Полученная информация бывает очень полезна также и для дальнейшего развития архитектуры, а также обеспечивает возможность набрать статистические данные и, на их основе, выполнить оптимизацию и ускорение работы. Так как симулятор реализуется еще на этапе разработки архитектуры, реализованные в нем *механизмы сбора статистики* позволяют накопить информацию, на основе которой можно принять решение о целесообразности отдельных аппаратных реализаций, выбрать наилучший размер кэш-памятей и прочее.

Полезными с точки зрения отладки, являются инструменты *контроля неопределенных ситуаций и правил планирования*. В любом достаточно сложном устройстве, таком как микропроцессор, в результате некоторых воздействий возможно возникновение ситуаций, которые являются неопределенными. По умолчанию, программное обеспечение не должно вызывать возникновение таких ситуаций и правильным поведением можно считать прерывание выполнения и сигнализация о типе ошибке Однако некоторые неопределенные ситуации не являются критическими, например запись по адресу, не соответствующему

никакой памяти или устройству. Для того, чтобы поддержать обе политики, в моделирующих комплексах реализован подход при котором все места возникновения неопределенных ситуаций снабжаются специальным макросом, сигнализирующем о ее возникновении, и, кроме того, определяется некоторое действие, представляющее реакцию данной реализации аппаратуры на воздействие. В зависимости от опций запуска, макрос может вызывать либо прерывание выполнения с выводом диагностического сообщения, либо продолжение работы с поведением, определенным данной реализацией.

В архитектуре «Эльбрус» в ряде случаев некоторые взаимные расположения инструкций в разных широких командах могут вызывать блокировки конвейера на несколько тактов. Причиной подобных блокировок является конфликт по использованию регистров, которые не успели освободиться из-за не завершившегося выполнения одной из предыдущих инструкций. Избежать возникновения блокировки можно, если вставить некоторое число пустых инструкций или инструкций, не использующих регистры, не успевшие освободится к текущему такту. При этом потеря тактов даже для случаев вставки пустых инструкций может оказаться меньше, чем длительность блокировки. Поиск и устранение таких блокировок является одним из способов повышения производительности приложений. Использование симулятора делает возможным сбор и анализ статистики возникающих блокировок, а также использование ее для помощи компилятору, который должен по возможности избегать генерации кода, вызывающего подобные ситуации. Этот механизм полезен, прежде всего, при анализе качества оптимизаций компилятором приложений, хотя и применим в других ситуациях

При исполнении на моделирующих комплексах некоторых объемных задач их выполнение может занимать продолжительное время. При этом, если во время выполнения в самой программе или в моделирующем комплексе обнаружена какая-то ошибка, время ее повторного воспроизведения может быть сопоставимо с первоначальным, что существенно затрудняет поиск, разбор и исправление такого рода ошибок. *Механизм контрольных точек* позволяет «запомнить» состояние всего комплекса в определенный момент работы и позднее продолжить выполнение с этого момента. Это позволяет в разы сократить время, необходимое для исправления ошибок при работе с продолжительными задачами.

Все вышеописанные средства нацелены, прежде всего, на исследование работы аппаратуры, а также отладку и оптимизацию работы программного обеспечения. Однако, отладочные возможности могут применяться и для поиска ошибок в проектируемой аппаратуре. Для этого, тестовая задача запускается как на моделирующем комплексе, так и на симуляторе RTL (register transfer level), являющимся эталонной функциональной моделью

разрабатываемой аппаратуры. В случае расхождения моделирование прекращается и выводится информация о содержимом различающегося ресурса. Результаты сравниваются с эталоном, определяемым документацией и выясняется какая из моделей (а возможно и обе) отработала неверно. После исправления ошибки, запуск повторяется заново до полной идентичности результатов. Единственным ограничением такого подхода является крайне низкая скорость работы симулятора RTL, проигрывающая 3-4 порядка относительно скорости низкоуровневой модели. Это является ограничением на набор тестовых задач, используемых для целей верификации.

Описанный принцип используется на этапе разработки аппаратуры. Однако не всегда удается исправить все ошибки в процессе проектирования. Если некорректная работа обнаружена после начала производства микросхем — исправление ее уже невозможно. В этом случае программное обеспечение должно уметь обходить аппаратные ошибки, то есть работать так, чтобы избегать их возникновения. Контроль такого обхода реализован по аналогии с контролем правил планирования, при одновременном возникновении ряда условий приводящих к возникновению аппаратной ошибки, моделирование может завершиться с выдачей диагностического сообщения. Однако, в случае если прекращение работы все же нежелательно, возможно его отключение. При этом, в моделирующий комплекс необходимо намеренно вносить аналогичную ошибку для того, чтобы ситуация на аппаратуре и на модели выполнялась одинаково.

Описанные в данной главе механизмы разработаны для решения основных задач, поставленных перед моделирующим комплексом - ускорение разработки и отладки программного обеспечения для разрабатываемых архитектур, оценка производительности и выбор параметров микропроцессоров на этапе проектирования.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

Основные результаты диссертационной работы связаны с разработкой методов построения моделирующих комплексов, применимых, прежде всего, к разрабатываемым комплексам различных компьютерных архитектур. Анализ существующих разновидностей программных моделей показал, что в большинстве случаев они не ориентированы на использование в качестве отладочных средств. Поэтому разрабатываемые моделирующие комплексы должны также обладать набором отладочных возможностей, удовлетворяющих потребностям разработчиков программного и аппаратного обеспечения.

В процессе выполнения диссертационной работы были получены следующие основные результаты, выносимые на защиту:

1. Проведено исследование существующих подходов и принципов моделирования, анализ существующих разновидностей программных моделей, рассмотрение аналогичных работ. На основании проведенного исследования сделан вывод о необходимости создания программной модели разрабатываемых комплексов, позволяющей упростить их отладку на стадии проектирования.
2. Разработан маршрут моделирования — общая технология разработки моделирующих комплексов, позволяющая интегрировать их применение в маршруте проектирования МС.
3. Разработаны архитектурно-независимые методы построения моделирующих комплексов, обеспечивающих высокую скорость работы и гибкость использования для различных микропроцессорных систем, в том числе для архитектур VLIW и RISC.
4. Разработаны методы построения многопроцессорных моделирующих комплексов с неоднородной архитектурой памяти. Предложенные методы использовались для разработки моделей микропроцессорных систем с NUMA архитектурой.
5. Разработаны общие методы описания периферийных устройств, шин и интерфейсов, отличительным свойством которых является возможность стандартизировать разработку моделей устройств данного типа. В сочетании с другими, приведенными выше методами, предложенный спектр средств позволяет разрабатывать модель всей микропроцессорной системы.
6. Разработан комплекс отладочных средств, включающих трассировку работы процессора и устройств, слежение за ресурсами, контроль неопределенных ситуаций, отличительными свойствами которых является обеспечение гибкости отладки и исследования поведения приложений.

Список работ, опубликованных по теме диссертации

1. Мешков А.Н. Реализация программного комплекса, моделирующего вычислительные комплексы с архитектурой SPARC V9 // Международная научная конференция, посвященная 80-летию со дня рождения академика В.А.Мельникова. Сборник докладов, 2009, С. 138-140.
2. Мешков А.Н. Разработка модели вычислительного комплекса "Эльбрус-S" // Труды

52-й научной конференции МФТИ, Часть 1, Том 1, М.:МФТИ, 2009, С. 58-60.

3. Мешков А.Н. Реализация программного комплекса, моделирующего многопроцессорные ВК с архитектурой SPARC V9 // Вопросы радиоэлектроники, серия ЭВТ, Выпуск 3, 2009, С 79-89.
4. Баратов Р.А., Гурин К.Л., Мешков А.Н., Разработка модели архитектуры, поддерживающей когерентные обращения в подсистеме памяти // Седьмые научные чтения памяти М.К. Тихонравова. Секция №6 «Новые информационно-телекоммуникационные технологии и их применение при решении специальных задач», 2009.
5. Мешков А.Н. Реализация программного комплекса, моделирующего вычислительные комплексы с архитектурой SPARC V9 // XXXV Гагаринские чтения. Научные труды Международной молодежной научной конференции в 8 т. Т. 4. М.: “МАТИ”–РГТУ, 2009, С. 138-139.
6. Гурин К.Л, Мешков А.Н., Сергин А.В., Якушева М.А. Развитие модели подсистемы памяти вычислительных комплексов серии «Эльбрус» // Вопросы радиоэлектроники, серия ЭВТ, Выпуск 3, 2010, С. 62-70.
7. Куцевол В.Н., Мешков А.Н., Черных С.В. «Разработка методологии и инструментов создания потактовых программных моделей цифровых устройств» // Труды 54-й научной конференции МФТИ, 2011, С. 25-26 Труды 54-й научной конференции МФТИ, 2011
8. Мешков А.Н. Модель подсистемы памяти для вычислительных комплексов на базе микропроцессора «Эльбрус-2S» // Труды 55-й научной конференции МФТИ, Радиотехника и кибернетика, Том 1, М.:МФТИ, 2012, С. 70-71.
9. Куцевол В.Н., Мешков А.Н., Петроченков М.В. Методология верификации протокола когерентности микропроцессора «Эльбрус-2S» // Вопросы радиоэлектроники, серия ЭВТ, Выпуск 3, 2013
10. Баратов Р.А., Камкин А.С., Майорова В.М., Мешков А.Н., Сортов А.А., Якушева М.А. Трудности модульной верификации аппаратуры на примере буфера команд микропроцессора «Эльбрус-2S» // Вопросы радиоэлектроники, серия ЭВТ, Выпуск 3, 2013