УДК 004

Современные тенденции оценки и контроля производительности микропроцессоров на стадии их разработки

Н.В. Николина, П.А. Чибисов, С.И. Аряшев

НИИ системных исследований РАН, nikolina@cs.niisi.ras.ru

***Аннотация* — Рассмотрены средства оценки производительности микропроцессора на этапе разработки, а именно: аналитические модели, потактовые эмуляторы, RTL-моделирование, ПЛИС-прототипирование. В статье анализируются достоинства и недостатки вышеуказанных средств. Также рассмотрены тестовые программы, позволяющие оценить производительность микропроцессора.**

***Ключевые слова* — оценка производительности, разработка микропроцессора.**

I. Введение

При разработке микропроцессора одним из наиболее актуальных вопросов является производительность будущей микросхемы. Под производительностью процессора принято понимать скорость выполнения им задачи (какого-либо приложения), то есть, чем меньше времени затрачивает процессор на реализацию той или иной задачи, тем выше его производительность. Отсюда следует уравнение производительности, ставшее известным под названием *Железного Закона* производительности процессора [1]:

$$\frac{1}{\textit{Производительность}} = \frac{\textit{Время}}{\textit{Программа}} = \frac{\textit{Инструкции}}{\textit{Программа}} \times \frac{\textit{Такты}}{\textit{Инструкции}} \times \frac{\textit{Время}}{\textit{Такты}}$$

Отношение *время/программа* может быть разложено на три составляющие: *инструкции/программа*, *такты/инструкция*, *время/такты*. Первый множитель показывает общее количество инструкций, необходимых для выполнения программы. Второй множитель указывает на то, сколько в среднем (по всему времени выполнения программы) требуется машинных тактов на выполнение каждой инструкции, обычно этот термин обозначается как CPI (cycles per instruction). Третий множитель учитывает количество времени, которое длится каждый машинный такт.

Производительность вычислительной системы определяют следующие факторы: время выполнения операций (регистровых, целочисленных, операций с плавающей запятой), время обращения к памяти на чтение/запись (в кэш-память разных уровней и во внешнюю память), организация подсистемы памяти, прерывания и исключительные ситуации, система ввода/вывода, время обмена (особенно для параллельных систем), компилятор, операционная система.

Для оценки производительности на этапе разработки микропроцессора применяют различные средства: аналитические модели, потактовые эмуляторы, RTL-моделирование, ПЛИС-прототипирование. В статье рассматриваются достоинства и недостатки вышеуказанных средств, их влияние на производительность готовой микросхемы. Также в статье рассматриваются тестовые программы, позволяющие оценить производительность микропроцессора.

II. Тесты производительности

Тесты производительности – это программы или фрагменты программ, призванные дать оценку производительности компонентов аппаратного обеспечения, взаимодействия аппаратного и программного обеспечения или всей компьютерной системы в целом.

Тесты производительности могут быть созданы как в фирме, где разрабатывается аппаратное обеспечение, так и взяты у сторонних организаций. Такими источниками могут считаться тесты, написанные пользователями и находящиеся в свободном доступе (например, Linpack [2]), а также наборы тестов, разработанные некоммерческими организациями (SPEC – Standard Performance Evaluation Corporation [3], EEMBC – Embedded Microprocessor Benchmark Consortium [4]). В качестве тестов производительности могут быть использованы и реальные приложения.

Ниже приведены примеры тестов производительности:

1) Linpack. Это набор высокопроизводительных библиотечных функций для линейной алгебры. Алгоритмы линейной алгебры весьма широко используются в самых разных задачах, и поэтому измерение производительности на LINPACK представляет интерес для многих пользователей. Также существует разновидность теста для многопроцессорных кластеров – HPL.

2) Whetstone. Синтетическая программа, разработанная для оценки производительности блока вещественной арифметики.

3) Dhrystone. Синтетический тест, который производит оценку производительности целочисленных операций. На сегодняшний день не является содержательно-информативным, так как целиком помещается в кэш-память первого уровня. Кроме того, ключи оптимизации компилятора позволяют сильно сократить тестирующий цикл.

4) SPEC. Некоммерческая организация, главной целью которой является разработка и публикация наборов тестов, предназначенных для измерения производительности компьютеров. Тестовые задачи SPEC, направленные на оценку производительности микропроцессора (SPEC CPU), разделены на две большие подгруппы — целочисленных задач и задач с плавающей точкой. Для сравнения результатов, полученных на тестах SPEC, в интернете можно найти таблицы, содержащие данные по процессорам различных архитектур и результаты запуска тестов SPEC с различными опциями компиляторов под разные операционные системы.

5) EEMBC. Некоммерческий консорциум производителей встраиваемых систем и инструментов разработки. EEMBC предлагает базовые алгоритмы для тестирования микропроцессоров для различных областей применения: автомобильная электроника, потребительская электроника, цифровые устройства индустрии развлечений, Java/CLDC-приложения, информационные сети, автоматизация управленческих работ, телекоммуникации. Одним из наиболее популярных тестов этой компании является COREMARK.

6) BDTI (Berkeley Design Technology, Inc.). Частная консультационная компания, предлагающая ряд специализированных тестов для оценки параметров систем обработки сигналов. Компания BDTI наиболее известна базовыми контрольными тестами для DSP-приложений (BDTI DSP Kernel Benchmarks), которые охватывают такие широко распространенные алгоритмы как КИХ-фильтры и БПФ. Эти тесты оптимизированы (обычно ручным способом) и выполняются в изолированной среде, т.е. без запуска ОС или параллельного процесса. Такой подход хорош тем, что он отражает уровень оптимизации, применяемый в DSP-приложениях, и дает ясное представление о реальной производительности. С другой стороны, такие тесты могут оценить лишь производительность ядра центрального процессора и не измеряют производительность подсистемы памяти, интерфейса ввода/вывода и т.д.

К недостаткам можно отнести то, что исходный код теста распространяется на платной основе, также BDTI не открывает свою методику сертификации, что затрудняет правильную интерпретацию результатов тестирования.

7) UBCS (Universal Benchmark and Compare System). В системе сделана попытка исключить влияние компиляторов, библиотек и тактовой частоты на результат сравнения производительности систем [5]. Оценка производительности осуществляется путем сравнения времени выполнения тестовой программы на различных архитектурах или путем сравнения с эталонной системой. Для минимизации влияния тактовой частоты сравниваемые системы должны иметь либо одинаковую частоту, либо, если это невозможно, тактовая частота может быть нормализована на определенное значение (например, 1 МГц). Таким образом, можно сфокусировать внимание на эффективности ядра. Для того, чтобы повысить качество сравнения, UBCS реализует алгоритмы, не требующие использования библиотечных функций.

8) APEX-MAP [6]. Пример теста, направленного на оценку производительности подсистемы памяти. Это параметризированный синтетический тест, основанный на концепции временной и пространственной локализации данных. Результаты оценки выдаются в тактах, необходимых для выполнения операций загрузки с разными параметрами.

В табл.1 приведены примеры тестов, используемых для оценки производительности микропроцессоров.

Таблица 1

Тесты для микропроцессоров

Тест	Область тестирования	Доступность	Качество
Dhrystone	ALU	свободно	низкое
Whetstone	FPU	свободно	среднее
Linpack	FPU	свободно	высокое
UBCS	CPU	платно	среднее
APEX	Память	свободно	среднее
NASPAR [7] SPLASH [8]	CPU	свободно	высокое
SPEC CPU	CPU	платно	высокое
EEMBC	CPU для встроенных приложений	платно	высокое
BDTI	DSP-процессор	платно	высокое

Практически все рассмотренные выше тесты в основном предназначены для оценки производительности готовой микросхемы, тем не менее, в настоящее время существуют методики, позволяющие использовать их уже на этапе разработки. Эти методики будут рассмотрены в следующих главах.

III. Аппаратная поддержка оценки производительности

При оценке производительности микропроцессора удобно использовать такие дополнительные средства, как счетчики определенных событий, влияющих на производительность, например, счетчики промахов в кэш-память или буфер трансляции адресов, счетчики неправильно предсказанных переходов, позволяющие проанализировать эффективность механизма предсказания команд ветвления, и прочее [9]. Кроме того, счетчики могут измерять количество тактов процессора, что позволяет уже на этапе RTL-модели (модели на уровне регистровых передач) оценить время работы того или иного приложения на ядре процессора, обычно не имеющего таймеров.

Для эффективной работы счетчики должны различать события, произошедшие вследствие спекулятивной выборки инструкций, иначе результат такого подсчета может сильно отличаться от реального числа анализируемых событий. Для этого используют мониторы, состояние которых зависит от того, в каком режиме выполняются инструкции. Причем, информацию о поведении системы в спекулятивном режиме также можно использовать как для оптимизации программного обеспечения [10], так и для анализа корректности реализации спекулятивного выполнения. В данном случае монитор может записывать следующую информацию о поведении системы: количество циклов в спекулятивном режиме, были ли отменены спекулятивные инструкции и прочее.

Конечно, аппаратная поддержка оценки производительности в первую очередь дает большие возможности для разработчиков программного обеспечения, тем не менее, они могут продуктивно использоваться на стадии разработки микропроцессора. В этом случае счетчики событий позволяют точно измерить время выполнения того или иного набора инструкций. Кроме того, после выхода микросхемы счетчики производительности используются для анализа узких мест и последующего их устранения в микропроцессорах следующего поколения.

IV. Разработка микропроцессора

Проектирование и оптимизация высокопроизводительных микропроцессоров становится все более и более сложной задачей из-за растущей сложности современных архитектур и большого количества времени, необходимого для детального моделирования.

Часто исследования архитектуры осуществляются с помощью потактовых эмуляторов, являющихся наиболее точными для оценки, но слишком медленными, что существенно ограничивает исследование. Поэтому процесс разработки микропроцессора значительно выиграет при использовании методологии более быстрой оценки производительности. Для этого разрабатывают аналитические модели, которые не обязательно должны иметь такую же точность, как и потактовые эмуляторы. Так как аналитические модели значительно быстрее и достаточно точны, они могут использоваться для отбраковки архитектурных решений. В дальнейшем для усовершенствования полученной архитектуры применяют потактовые эмуляторы. Такой подход значительно ускорит процесс разработки микропроцессора и даст возможность более широкого исследования архитектуры в более короткие сроки (рис.1).

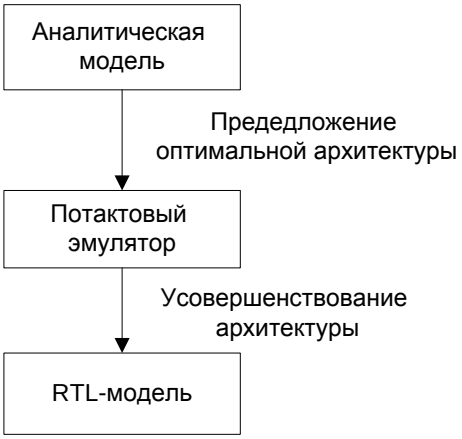

Рис. 1. Процесс разработки микропроцессора

Следует отметить также тот факт, что при выборе архитектуры разрабатываемого микропроцессора под определенную операционную систему этап создания аналитической модели может быть опущен.

A. Аналитические модели

В последнее время аналитические модели предсказания производительности пользуются повышенным интересом. Существует два основных подхода к созданию аналитической модели: эмпирическое [11, 12] и механистическое [13, 14] моделирование. Так как эмпирические модели предполагают в качестве входных параметров статистические данные, полученные в результате потактового моделирования, они не подходят для ранней стадии разработки, когда потактовый эмулятор не доступен, или содержит неполную функциональность.

При механистическом моделировании аналитическая модель строится на фундаментальном понимании разрабатываемой системы. К

механистическим моделям относится модель, предложенная Карханисом и Смитом [13], основанная на интервальном анализе, где выполнение программы разделено на интервалы, состоящие из непрерывно выполняемых инструкций и задержек, вызванных различными событиями. Это могут быть промахи в кэш-память, в TLB (таблицу трансляции адресов), неправильно предсказанные инструкции перехода. Таха и Виллс [14] предложили механистическую модель, которая разбивает выполнение программы на так называемые макро блоки, разделенные неправильно предсказанными командами перехода. Модель оценивает производительность процессора, основываясь на ключевом наборе метрик архитектуры и приложений. Эти метрики определены в спецификации. Метрики архитектуры описывают структуру ключевых компонентов процессора, например, количество инструкций, которое процессор может выполнять параллельно, число функциональных блоков, размер кэш-памяти и длительность задержки при обращении за данными. Спецификация приложения представляет набор параметров, характеризующих общее поведение выполняемой трассы инструкций. Эти параметры определяют параллелизм задач и данных, доступ к памяти и получаются после однократного запуска программы на анализирующем программном средстве.

Механистическая модель позволяет построить CPI-стеки для различных программ. CPI-стек – это разделение CPI на компоненты, а именно: базовый CPI и задержки от различных событий (промахов в кэш-память, неправильно предсказанных инструкций перехода). На рис.2 показан пример CPI-стека для теста twolf (SPEC CPU2000), запущенном на суперскалярном out-of-order процессоре, выдающем 4 инструкции на исполнение [15].

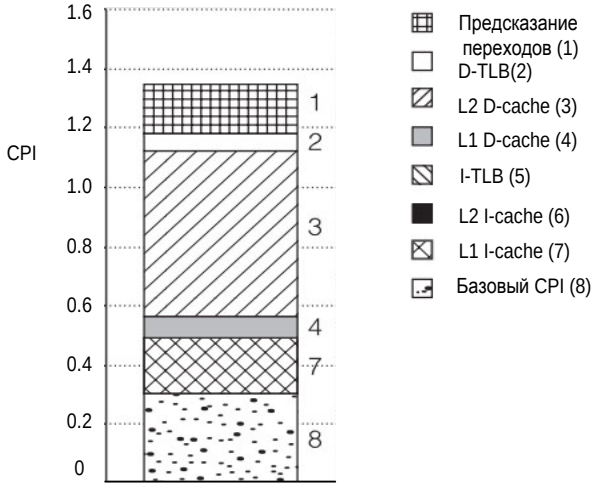

Рис. 2. Пример CPI-стека для теста twolf

При использовании механистической модели результаты оценки производительности очень близки к результатам оценки при потактовом моделировании; ошибка составляет примерно 5,5%, а моделирование ускоряется в $1{,}6\times10^6$ раз [14]. Хотя эта модель не отменяет необходимость применения потактовых эмуляторов, она может быть использована на ранней стадии для отбраковки большого количества архитектурных решений. Модель позволяет архитекторам оценить больше альтернатив в реализации архитектуры процессора на гораздо большей базе приложений в рамках данного времени. Это позволяет осуществить более эффективный и полный поиск архитектурных решений для разработки процессора под данные приложения.

B. Потактовые эмуляторы

После того, как определена архитектура разрабатываемого процессора, проводится исследование и усовершенствование отдельных ее компонентов. Для этой цели идеально подходят потактовые эмуляторы, позволяющие со 100-процентной точностью моделировать задержки конвейера, что дает возможность выявить узкие места и скорректировать модель гораздо меньшими усилиями, чем потребовалось бы на стадии разработки RTL-модели. Кроме того, потактовые эмуляторы используются при разработке компиляторов, специализированных библиотек и ядра операционной системы.

Потактовые эмуляторы могут быть написаны на языке высокого уровня, например, на языке C++ [16, 17], также возможно использование специализированных библиотек SystemC [18] для создания поведенческих моделей цифровых устройств.

Процесс написания потактовой модели микропроцессора является трудоемким, поэтому предприняты попытки его автоматизации. Например, Чангом был разработан язык спецификации вместе с транслятором [19], который позволяет автоматизировать процесс создания потактового эмулятора.

С началом разработки RTL-модели потактовые эмуляторы используются для верификации микропроцессора. Состояние RTL-модели и эмулятора сравнивается после каждого моделируемого такта. Это позволяет найти не только функциональные ошибки, но и ошибки производительности, связанные с задержками, внесенными разработчиками RTL-модели для исправления функциональных ошибок.

C. RTL-моделирование

Несмотря на то, что аналитическая модель производительности и потактовое моделирование дают достаточно точный результат оценки, это не отменяет полностью стадию оценки производительности RTL-модели разрабатываемого микропроцессора.

При разработке тестовых программ оценки производительности для RTL-модели начинают с простейших ситуаций:

1) оценка длительности единичных инструкций, проверка ее соответствия требованиям спецификации.

2) Длительность обработки промахов и попаданий в кэш-память разных уровней для инструкций загрузки/сохранения.

3) Задержки конвейера в случае зависимых инструкций.

Затем рассматривается влияние ограниченных ресурсов на производительность. Такие тесты определяют влияние размера очередей и буферов ядра микропроцессора на производительность. Например, предполагая бесконечной очередь для сохраняемых данных, можно определить эталонное значение для тестовой ситуации. Однако, если размер очереди конечен, тестовый случай может заполнить её, что приведет к остановкам конвейера и падению производительности. Для правильно организованного тестового случая можно соотнести разницу между идеальным и реальным значением производительности с размером очереди.

Также можно получить более сложное значение, определяющее производительность, называемое сигнатурой производительности. Сигнатура – это набор различных параметров, характеризующих поведение системы для данного тестового случая. Тестовые случаи могут быть получены с помощью генератора псевдослучайных тестов. Результаты прохождения теста на RTL-модели и на потактовом эмуляторе сравниваются и, в случае несовпадения, сигнализируется ошибка в производительности.

Все тестовые ситуации сохраняются для дальнейшего регрессионного тестирования и прогоняются на проекте после любых изменений, внесенных в RTL-модель. Это необходимо для того, чтобы убедиться, не отразились ли эти изменения на производительности.

Следующим этапом является адаптация существующих тестов для оценки производительности для запуска их на RTL-модели [20]. Обычно такие тесты предполагают запуск под управлением операционной системы, что не всегда возможно обеспечить на RTL-модели микропроцессора из-за неполного функционального состава системы, а также большой длительности загрузки самой операционной системы. Для адаптации тестов предлагается использовать вместо системных функций получения текущего времени счетчики процессорных тактов. Кроме того, все нереализованные функции, являющиеся частью интерфейса теста и не входящие в состав тестирующего цикла, можно исключить. Также для адаптации тестов можно рассмотреть возможность уменьшения количества проходов тестового цикла. Это может привести к понижению точности оценки производительности, но дает возможность получить результат за разумное время.

Также на этапе разработки RTL-модели осуществляется контроль значения IPC (IPC=1/CPI) для всех тестов, предназначенных для верификации и валидации микропроцессора. Обычно после прохода данных тестов выдается отчет не только о содержащихся в проекте ошибках, но и отчет о производительности, содержащий название и значение IPC тех тестов, которые показали худший результат оценки производительности по сравнению с предыдущей версией RTL-модели.

Чтобы разрабатываемый микропроцессор имел наибольшую производительность на каком-то определенном классе задач, можно оценить приложения, которые наиболее часто будут исполняться на реальной микросхеме (если таковые известны), с помощью специальных средств, называемых средствами профилирования. Такие средства выделяют функции, наиболее часто выполняемые микропроцессором. Эти функции в дальнейшем могут быть включены в тесты производительности для RTL-модели микропроцессора.

D. Средства повышения скорости моделирования

Для повышения скорости моделирования используют ПЛИС-прототипы. Достоинством использования прототипа являются максимально приближенные к реальным условия, в которых функционирует тестируемая микросхема, а также достаточно высокая производительность, что позволяет запускать программы и приложения, недоступные для запуска на RTL-модели из-за большого времени выполнения. ПЛИС-прототип позволяет запустить одну или несколько операционных систем, а также запустить и за разумное время получить результаты тестов под операционные системы. Кроме того, ПЛИС-прототип позволяет изучать диаграммы внутренних сигналов в графической оболочке, что, с одной стороны, используется для нахождения ошибок, а с другой – позволяет выявить узкие места в выполнении отдельных наборов инструкций. Обнаруженные проблемные наборы команд в дальнейшем используются для регрессионного тестирования RTL-модели.

К недостаткам ПЛИС-прототипа относится невозможность установки тактовой частоты, на которой планируется эксплуатация готовой микросхемы, поэтому результаты измерения необходимо нормировать.

E. Тестовый запуск

Тестовый запуск позволяет получить готовую микросхему с конечной производительностью без запуска ее в серийное производство. На этом этапе могут быть использованы счетчики производительности, которые дают возможность разработчикам аппаратного обеспечения оценивать производительность микропроцессора на сложных приложениях, которые невозможно было запустить на стадии разработки. Становится возможным запуск тестов производительности на реальной системе и сравнение их результатов со значениями,

полученными на микропроцессоре со схожей архитектурой и компиляторами.

Тестовый кристалл может быть использован группой разработки компилятора для проверки создаваемых планировщиком последовательностей инструкций с точки зрения заполнения конвейера микропроцессора.

Также для тестового кристалла может быть построена механистически-эмпирическая модель [21]. Это гибридная модель, заимствующая фундаментальное представление о системе от механистической модели и более простую реализацию от эмпирической. Модель позволяет построить CPI-стеки для разных приложений, тем самым делая возможным оптимизировать программное обеспечение под созданный микропроцессор.

К недостаткам тестовых запусков можно отнести высокую стоимость производства кристаллов и низкую локализацию проблем производительности. Тем не менее, именно благодаря тестовым запускам становится возможным оценка производительности на реальных задачах, выполняющихся на готовой микросхеме, и, в случае выявления проблем производительности, внесения в микропроцессор изменений перед запуском его в серийное производство.

V. Заключение

В статье предпринята попытка рассмотреть различные методы оценки и контроля производительности разрабатываемых микропроцессоров и их моделей. Рассмотрены преимущества ранней оценки производительности микропроцессора (до разработки RTL-модели), методы оценки производительности на стадии RTL-моделирования, ПЛИС-прототипирования. Также рассматривается возможность оценки производительности на реальной микросхеме, используя тестовый запуск.

Рассмотренные методы суммируют накопленный опыт измерения производительности разрабатываемых в НИИСИ РАН микропроцессоров и их моделей. Кроме того, по данным методам авторами создана система регрессионного измерения производительности, позволяющая контролировать производительность на всех этапах разработки.

Литература

[1] John Paul Shen, Mikko H. Lipasti. Modern Processor Design - Fundamentals of Superscalar Processors // McGraw-Hill. – 2003. – 488 p.

[2] Linpack. URL: http://www.top500.org/lists/linpack.php (дата обращения: 11.01.2012).

[3] Standart Performance Evaluation Corporation. URL: http://www.spec.org (дата обращения: 11.01.2012).

[4] Embedded Benchmark Consortium. URL: http://www.eembc.org (дата обращения: 11.01.2012) .

[5] Kramer K., Stolze T., Oppelt A. Microprocessor Benchmarks - A Detailed Look at Techniques, Problems and Solutions // ICSENG '11 Proceedings of the 2011 21st International Conference on Systems Engineering. - 2011. - P. 337 - 340.

[6] Strohmaier E., Shan H. Apex-Map: A Synthetic Scalable Benchmark Probe to Explore Data Access Performance on Highly Parallel Systems // EuroPar2005. - 2005. – P. 114 – 123.

[7] NAS Parallel Benchmarks. URL: http://www.nas.nasa.gov/Software/NPB (дата обращения: 11.01.2012).

[8] Stanford Parallel Applications for Shared Memory. URL: http://www-flash.stanford.edu/apps/SPLASH (дата обращения: 11.01.2012).

[9] Sprunt B. The Basics of Performance-Monitoring Hardware // IEEE Micro. 2002. Vol. 22. No. 4. P. 64-71.

[10] Caprioli P., Chaudhry S., Yip S. Patent No. : US 7,757,068 B2. Method and apparatus for measuring performance during speculative execution. Date of Patent : Jul. 13, 2010.

[11] Joseph P. J., Vaswani K., Thazhuthaveetil M. J. A predictive performance model for superscalar processors // MICRO. - 2006. – P. 161 - 170.

[12] Lee B., Brooks D. Efficiency trends and limits from comprehensive microarchitectural adaptivity // ASPLOS. - 2008. – P. 36 - 47.

[13] Karkhanis T., Smith J. Automated design of application specific superscalar processors: An analytical approach // ISCA. - 2007. – P. 402 - 411.

[14] Taha T., Wills D. An instruction throughput model of superscalar processors // IEEE Transactions on Computers. - 2008. Vol. 57. No. 3. P. 389 - 403.

[15] Eyerman S., Eeckhout L., Karkhanis T., Smith J. E. A Top-Down Approach to Architecting CPI Component Performance Counters // IEEE Micro. 2007. Vol. 27. No. 1. P. 84-93.

[16] Слепов А. Б. Разработка потактовой поведенческой модели системы на кристалле на языке C++ / Сборник научных трудов «Проблемы разработки перспективных микроэлектронных систем – 2010» / Под ред. А.Л. Стемпковского. - М.: ИППМ РАН, 2010. - С. 450 – 453.

[17] Maturana G., Ball J. L., Gee J., Iyer A., O'Connor J. M. Incas: a cycle accurate model of UltraSPARC // IEEE International Conference on Computer Design (ICCD'95). - 1995. – P. 130 - 135.

[18] SystemC. URL: http://www.systemc.org (дата обращения: 11.01.2012).

[19] Chang S. F., Hu A. J. Fast Specification of Cycle-Accurate Processor Models // IEEE International Conference on Computer Design (ICCD'01). - 2001. – P. 144 - 148.

[20] Аряшев С.И., Краснюк А.А., Чибисов П.А. Адаптация тестов для оценки производительности 64-разрядного универсального суперскалярного микропроцессора / Сборник научных трудов «Проблемы разработки перспективных микроэлектронных систем – 2005» / Под ред. А.Л. Стемпковского. - М.: ИППМ РАН, 2005. - С. 263 – 268.

[21] Eyerman S., Hoste K., Eeckhout L. Mechanistic-empirical processor performance modeling for constructing CPI stacks on real hardware // IEEE International Symposium on Performance Analysis of Systems and Software. - 2011. – P. 216 - 226.