Для цитирования: Жезлов К. А., Колбасов Я. С., Николаев А. В., Путря Ф. М., Фролова С. Е. Этапы маршрута верификации и валидации системы, ориентированные на получение модели СнК, способной гарантированно исполнять целевые задачи и алгоритмы с заданными характеристиками // Вопросы радиоэлектроники. 2018. № 8. С. 64–72.
DOI 10.21778/2218-5453-2018-8-64-72
УДК 004.415.538

К. А. Жезлов[1], Я. С. Колбасов[1], А. В. Николаев[1], Ф. М. Путря[1], С. Е. Фролова[1]

[1] АО «Научно-производственный центр “ЭЛВИС”»

ЭТАПЫ МАРШРУТА ВЕРИФИКАЦИИ И ВАЛИДАЦИИ СИСТЕМЫ, ОРИЕНТИРОВАННЫЕ НА ПОЛУЧЕНИЕ МОДЕЛИ СНК, СПОСОБНОЙ ГАРАНТИРОВАННО ИСПОЛНЯТЬ ЦЕЛЕВЫЕ ЗАДАЧИ И АЛГОРИТМЫ С ЗАДАННЫМИ ХАРАКТЕРИСТИКАМИ

Для обеспечения сходимости проекта СнК по критерию эффективности на реальных задачах весь маршрут разработки СнК, начиная с архитектурного проектирования и заканчивая квалификационными тестами топологического списка цепей, должен быть ориентирован на раннюю локализацию проблем с производительностью и подтверждение потребительских характеристик СнК перед ее выпуском на фабрику (tapeout). В статье раскрываются некоторые детали маршрута верификации СнК, ориентированного на сходимость по требованиям задания по производительности, и приводятся результаты работы инструментария, используемого при исследованиях СнК и ее компонент на различных уровнях абстракции. В основе маршрута лежит использование шаблонных тестовых окружений и унифицированных средств анализа производительности. В статье предложен метод описания входных воздействий на основе графов, позволяющий точно восстановить сценарии, при которых возникла ошибка, а также смоделировать реальный трафик. Описаны этапы маршрута верификации и валидации системы, ориентированные на получение модели СнК, способной гарантированно исполнять целевые задачи и алгоритмы с заданными характеристиками, и приведен пример системы, верифицированной по принципу контроля метрик производительности.

Ключевые слова: *система на кристалле (СнК), верификация, анализ производительности, сценарии использования, верификация на системном уровне, автономная верификация, прототип, высокоуровневая модель*

Введение

Современные СнК – крайне сложные устройства, состоящие из множества разнотипных блоков (вычислительные ядра различной архитектуры, специализированные ускорители, большая номенклатура периферийных блоков, далеко не тривиальная подсистема коммутации, объединяющая все элементы между собой). Перед разработчиками таких систем стоит задача гарантировать на этапе разработки ИС СнК, что после ее изготовления у потребителя не возникнет никаких проблем с ее использованием. Для анализа всех потенциальных проблем и сценариев использования СнК требуются люди с различным набором компетенций и серьезная работа архитектурной команды. На этапе проработки архитектуры возможны просчеты, связанные с недостаточным учетом как специфики схемотехнической реализации аппаратуры, так и особенностей работы реального ПО. В свою очередь, разработчики аппаратуры могут допускать ошибки в своих блоках, не влияющие на функцию устройства, но сказывающиеся на эффективности его работы в составе СнК. В условиях сжатых сроков разработки современных СнК обнаружение подобных критических проблем с эффективностью на конечной стадии сборки RTL всей СнК может привести к серьезным временным и финансовым потерям. Поэтому весь маршрут разработки СнК, начиная с архитектурного проектирования и заканчивая квалификационными тестам топологического списка цепей, должен быть ориентирован на раннюю локализацию проблем с производительностью и подтверждение потребительских характеристик СнК перед ее выпуском на фабрику.

Исследование модели или прототипа СнК

Идеальным и самым точным способом оценки характеристик проектируемой системы (до момента выпуска кристалла) является запуск реальных приложений на RTL-модели, аппаратном прототипе или эмуляторе всей системы. Такие эксперименты обеспечили бы достаточно высокую степень уверенности в достижимости заданных параметров производительности на реальных приложениях. Однако исследования на данном уровне сопряжены с целым рядом проблем:

1. Моделирование RTL слишком медленное и не позволяет в разумные сроки выполнить исследование на реальных приложениях.
2. Эмуляторы являются весьма дорогостоящими, а для СнК с высокой степенью интеграции емкости даже современных эмуляторов может быть недостаточно, и на практике для СнК скорость моделирования на несколько порядков отличается от реальной СнК, и этого также недостаточно для отладки тестов производительности на базе реального ПО.
3. Создание аппаратного прототипа сопряжено с внесением изменений в проект, после которых прототип становится лишь похожим, но не эквивалентным исходному проекту (периферия, макроблоки памятей и т.п.), кроме того, для уровня СнК создание полного прототипа зачастую бывает просто невозможным (объем проекта, наличие закрытых макроблоков и т.п.).
4. RTL всей системы появляется ближе к концу проекта, когда уже бывает поздно вносить принципиальные коррективы в архитектуру и реализацию СнК и ее компонент.
5. Получить программное обеспечение, которое можно хотя бы запустить на новой СнК к моменту появления RTL, – задача весьма трудоемкая, не говоря уже о необходимости его оптимизации для получения более адекватных результатов исследования.

Тем не менее на системном уровне имеется возможность запуска относительно коротких синтетических тестов и тестов, имитирующих определенные краевые ситуации реальных сценариев использования. Для выявления потенциальных проблем с производительностью в маршруте верификации используется предложенный авторами подход к разработке параметризированных комплексных тестов, создаваемых по принципу конструктора. Такой подход позволяет быстро создавать различные исследовательские тесты, имитирующие краевые случаи сценариев использования СнК и при этом исполняющиеся за приемлемое время даже на RTL-модели.

Тем не менее для обеспечения сходимости проекта по параметрам производительности большая часть работы выполняется на других, более ранних этапах проектирования СнК.

Модели высокого уровня абстракции

Виртуальные прототипы [1], повторяющие блочную структуру СнК и состоящие из TLM-моделей СФ-блоков, позволяют разрабатывать ПО под выбранную архитектуру СнК параллельно с разработкой RTL, выполнять первоначальную (грубую) оценку производительности на базе анализа межблочных потоков транзакций, выставлять требования к блокам СнК и, при необходимости, вносить коррективы в архитектуру СнК на основе проведенного анализа.

Проблемы, характерные для данного этапа:

- Далеко не все поставщики СФ-блоков поставляют TLM-модели блоков, что не позволяет собрать полную модель СнК.
- Разработка недостающих моделей покупных и собственных СФ-блоков с нуля – трудоемкая задача, иногда даже сопоставимая с разработкой первого варианта RTL.

В маршруте верификации нами используются как TLM-модели СнК, когда имеются в наличии все требуемые для исследования модели блоков, так и модели еще более высокого уровня абстракции. Во втором случае создается смешанная модель, для некоторых частей системы по уровню абстракции близкая к алгоритмической. Степень подробности такой модели выбирается минимальной для получения требуемых характеристик будущей системы (потоки данных, требования к вычислительной мощности процессора), но таким образом, чтобы ее можно было разработать в кратчайшие сроки.

Средства автономного исследования СФ-блоков

Автономные окружения блоков являются наиболее удобными средствами исследования компонент системы в силу скорости моделирования и наблюдаемости внутренних событий. К проблемам данного уровня можно отнести следующее:

- Число блоков в составе СнК составляет десятки, т.е. для полноценного исследования требуется создание множества окружений разной степени сложности.
- В составе системы СФ-блок находится в определенных условиях и при определенном характере воздействий на блок, которые определяются как архитектурой СнК, так и особенностями ПО. Для

подтверждения характеристик блока именно в тех условиях, в которых он будет работать в составе системы, их необходимо воспроизвести на автономном окружении СФ-блока.

Для снижения трудозатрат на исследование большого числа СФ-блоков в маршруте используется подход на базе шаблонных окружений и унифицированных средств анализа производительности (модули сбора трасс, статистики, средства построения метрик) [2, 3].

В начале процесса проектирования, когда RTL СнК еще не собран, условия, создаваемые для блока на автономном окружении, формируются исходя из данных, полученных при исследовании модели системы высокого уровня абстракции (грубое приближение характера системного трафика) и автономного исследования других блоков, входящих в систему (уточнение на RTL-моделях характера трафика, формируемого блоками в заданных условиях). Например, весьма актуален обмен информацией о характере трафика между окружениями специализированных процессоров, контроллеров памяти и коммутаторов. В частности, речь может идти об исследовании устойчивости производительности специализированного вычислителя к времени доступа к памяти.

Инструменты исследования коммутационной логики

Одной из проблем, возникающих при тестировании, в частности коммутационной логики, является восстановление сценария, при котором возникла ошибка. Особенно актуально это для полностью случайных тестов. Для упрощения поиска сценариев, вызывающих ошибку, было предложено использовать тесты с графовым представлением трафика. Кроме того, при использовании тестов на основе графов повышается «естественность» трафика, т.е. при верификации модели устройства тестировщик приближается к работе реального устройства в кремнии; так же можно описывать и полностью реальные сценарии, снятые, например, с устройства схожего класса.

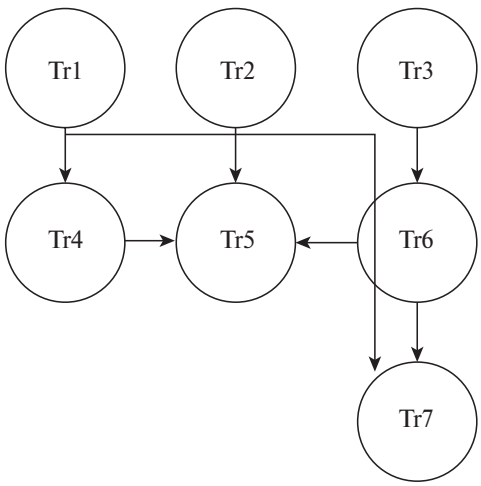

Рисунок 1. Тест в графовом представлении

На рис. 1 приведен пример теста в графовом представлении. Вершинами графа являются транзакции, а ребрами – зависимости между ними, то есть зависимая транзакция не запустится до тех пор, пока не будут исполнены все ее родительские транзакции. Цветом обозначены транзакции, проходящие через один ведущий блок.

Если рассматривать коммутационную логику как класс тестируемых устройств, то можно прийти к выводу, что любое устройство передачи информации внутри кристалла можно аппроксимировать коммутатором (в вырожденном случае интерфейса), а значит, существует возможность для автоматической генерации теста на основе графа из матрицы коммутации и описания портов. Подобный подход облегчает и ускоряет процесс верификации.

Базовым является случайный тест, он необходим для покрытия большей части кода и обнаружения «неожиданных» ошибок, и именно при возникновении такой ошибки будет удобен графовый подход, так как для воссоздания ситуации ее возникновения достаточно из большого графа выделить подграф, в котором произошла ошибка.

Помимо случайных тестов существует огромное множество вариантов для генерации сценариев и воссоздания критических ситуаций. Одним из них является тест карты памяти. Соответствие карты памяти устройства документации является одним из критических аспектов при верификации коммутационной логики, и при использовании генератора можно быстро совершить проверку при изменении карты памяти. Преимущества появляются и при нагрузочном тестировании, благодаря детальному контролю над потоками данных, проходящих через тестируемое устройство.

Для проверки вероятных «узких мест» существуют синтетические тесты, направленные на симуляцию конкретного класса ситуаций (переполнение FIFO, несколько потоков через один мастер и т.д.). С их помощью можно выявлять проблемы, упущенные при составлении списка рабочих сценариев устройства. Информация о них является полезной в том числе и для конечного пользователя СнК, для понимания потенциального спектра применимости системы. Итак, основными преимуществами использования универсального генератора тестов на основе графов являются:

- генерация теста, конфигурируемого с учетом ограничений коммутационной логики;

- графовое представление потока данных;
- дополнительный инструмент исследования производительности коммутации, характерной для реальных сценариев проектируемой системы;
- использование графа для быстрого поиска сценария (подграфа), который привел к ошибке;
- универсальность для коммутационной логики;
- возможность изменения графа вручную для более детальной его настройки.

Принцип использования метрик, с помощью которого проводится анализ, описан в статьях [4, 5]. Среди наиболее значимых метрик можно отметить следующие:

- средняя/максимальная скорость передачи данных;
- среднее/максимальное время исполнения транзакции;
- средняя/минимальная/максимальная пропускная способность.

Для автоматизации поиска проблем с коммутационной логикой на автономном уровне и на уровне системы на кристалле введены новые метрики:

- среднее/минимальное время пересечения транзакций;
- средняя/максимальная/минимальная скорость исполнения транзакции.

Прототипирование СФ-блоков

При работе с аппаратным прототипом на задачу использования целевых алгоритмов накладываются ограничения использования алгоритмов, связанные прежде всего со структурой существующей аппаратной платформы. Типовая платформа прототипирования для процессорных СнК, как правило, имеет структуру, представленную на рис. 2.

Плата с ПЛИС (FPGA board) содержит проверяемый проект. Плата управляющего процессора (CPU board) служит для задания режимов проекта, для загрузки данных в общую память, а также для целей управления процессом отладки и сбором служебной информации. Материнская плата (Motherboard) объединяет две платы интерфейсами и содержит интерфейсы ввода-вывода, доступные со стороны проверяемого проекта. Состав имеющихся интерфейсов ввода-вывода, размер доступной памяти, пропускная способность интерфейсов – это элементы, ограничивающие возможности реализации штатных алгоритмов на аппаратном прототипе. Соответственно, целевой алгоритм должен быть переработан с учетом этих ограничений.

С другой стороны, несмотря на ограничения, использование прототипа предоставляет целый ряд

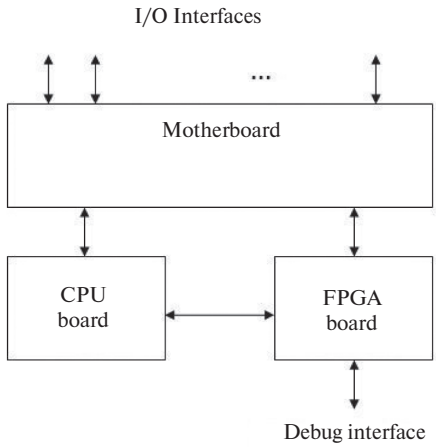

Рисунок 2. Типовая структура платформы прототипирования

преимуществ при отладке проекта и анализе его с точки зрения реализации целевых алгоритмов.

Для оценки производительности будущей СнК и отладки целевых алгоритмов при работе с прототипом процессорных СнК применяются следующие средства:

- модули трассы;
- мониторы интерфейсов;
- мониторы заданных событий.

Модуль трассы

Под трассой процессора подразумевается поток данных по специализированному интерфейсу, встроенному в тестируемую аппаратуру. К примеру, для процессорного ядра в трассе передается информация об изменении программного счетчика вместе с программно и аппаратно устанавливаемыми временными метками, которые могут быть использованы для профилирования и оптимизации программно-аппаратного комплекса. Модуль трассы представляет собой блок, внедряемый в ПЛИС и служащий для сохранения трассы проверяемого процессорного блока в памяти или выдаче данных трассы на внешний интерфейс. При сохранении трассы в памяти модуль при заполнении текущего выдает прерывание на управляющий процессор, по которому начинается выгрузка данных из памяти.

Мониторы интерфейсов

Монитор интерфейсов – это блок, который подсоединяется к проверяемому интерфейсу проверяемого блока. В случае задачи отработки целевого алгоритма в целях изучения производительности СнК мониторы необходимо подсоединить к каждому блоку – инициатору обмена и пассивному

устройству, находящемуся на интерфейсе. Монитор выполняет следующие функции:

- мониторинг протокола интерфейса, контроль сигналов, формирование прерывания по зависанию;
- подсчет производительности интерфейса на базе сигналов интерфейса и встроенных счетчиков (подсчет латентности, разных видов задержек, подсчет продолжительности цикла);
- генерация прерываний по заданным событиям (можно настроить по фильтру);
- фиксация данных по заданным событиям интерфейса.

Монитор интерфейса проектируется отдельным блоком однократно для заданного вида интерфейса и может быть повторно использован в разных проектах, использующих данный интерфейс.

Мониторы заданных событий

При необходимости отслеживания наступления какого-либо события или последовательности событий в прототип внедряется блок, который по наступлении события может генерировать прерывание на центральный процессор, делать сброс отдельных модулей, останавливать процессор. Такой блок проектируется заказным под конкретную задачу для заданного прототипа СнК.

Пример системы, спроектированной по принципу отслеживания метрик производительности на всех этапах разработки

Наряду с требованиями к радиационной стойкости и малому потреблению чипа для обеспечения его применимости в космических аппаратах для вновь разрабатываемого проекта контроллера накопителя на базе NAND памяти ставилось дополнительное амбициозное требование обеспечения скорости обмена с накопителем в 2 Гбит/с.

Во время предварительного анализа архитектуры были выявлены следующие критические точки, способные повлиять на конечные характеристики системы в целом:

1. Скорость работы портов SpaceFiber (цифровая часть + PHY).
2. Производительность NAND-контроллеров.
3. Производительность порта памяти DDR.
4. Производительность коммутационной логики.
5. Производительность CPU, в частности:
   - загрузка при обработке запросов, приходящих от SpaceFiber по RMAP-протоколу;
   - загрузка при управлении NAND (обработка очереди запросов, переупорядочивание операций обращения к памяти, обслуживание событий от NAND-контроллеров);
   - загрузка при отработке алгоритма распределения запросов между восемью очередями NAND-контроллеров и управления программным кэшем для повышения общей пропускной способности и времени жизни микросхем памяти.

Каждый из приведенных пунктов может стать «узким местом», критически влияющим на характеристики системы. Чтобы избежать лишних циклов перепроектирования, был применен подход Requrement Driven Verification [3], подразумевающий отслеживание факта выполнения [4] требований технического задания на всех этапах проектирования начиная с самого первого этапа, на котором имеет место неопределенность архитектуры системы.

В маршруте использовались:

- анализ требований к элементам системы со стороны алгоритма распределения запросов и управления программным кэшем с помощью TLM-модели высокого уровня абстракции;
- декомпозиция задачи анализа эффективности системы (автономное исследование блоков);
- анализ эффективности системы на RTL-модели всей системы (комплексные тесты системного уровня).

Принципиальной особенностью маршрута была обратная связь от каждого направления исследования на все остальные. Таким образом, при появлении более детализированных моделей блоков обновлялась информация для окружения TLM-модели, в котором отлаживался алгоритм (например, число прерываний от NAND-контроллера). Точно так же при обновлении алгоритма уточнялись требования к пропускной способности интерфейсов и производительности CPU, в частности, принципиально важным было получение информации о служебном трафике от CPU в процессе выполнения алгоритма распределения запросов по NAND-контроллерам.

Далее приведены некоторые особо значимые результаты исследований, выполненных в процессе проектирования системы на кристалле.

Исследование 1 (инструменты: автономное окружение NAND-контроллера, высокоуровневая модель всей системы): 8 контроллеров NAND в совокупности создают огромный поток прерываний (8 прерываний при передаче одной 4K страницы), который полностью отвлекает на себя ресурсы одного процессора выбранной для проекта архитектуры.

По этой причине было принято сразу два решения:

1. Ввести в систему дополнительный CPU для управления восемью контроллерами NAND.

2. Внедрить в аппаратуру контроллера NAND автомат, берущий на себя управление контроллером при пересылке страниц в режиме автоинкремента адреса (блочный режим работы контроллера).

В совокупности эти решения привели к высвобождению целого CPU для задач управления кэшем и SPFR, а второй CPU в типовом режиме работы получил запас для выполнения задачи оптимизации и распараллеливания потока транзакций записи и чтения в NAND-контроллер. На рис. 3 приведен график загруженности CPU при одновременной передаче пакетов в блочном режиме.

Исследование 2 (инструменты: автономное окружение DDR-контроллера): для балансировки скорости вешнего порта DDR и внутреннего коммутатора в DDR-контроллер был добавлен дополнительный AXI-порт. Однако исследование показало, что при работе в рабочем диапазоне частот второй порт дает преимущество лишь в случае работы только с уже открытыми страницами. В реальной ситуации на DDR работает более 10 мастеров одновременно, что приводит к частому открытию/закрытию страниц, а в такой ситуации разницы между одним и двумя портами практически нет. На рис. 4 и 5 приведены графики зависимости пропускной способности DDR от числа одновременно используемых страниц при различных частотах.

Исследование 3 (инструменты: окружение системного уровня СнК): анализ подсистемы

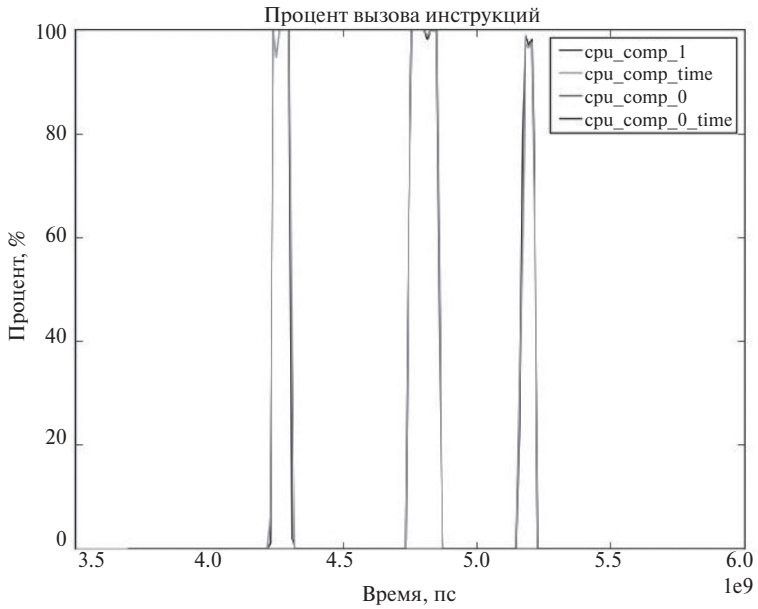

Рисунок 3. Загруженность CPU при отработке прерываний от 8 контроллеров NAND при одновременной передаче пакетов в блочном режиме

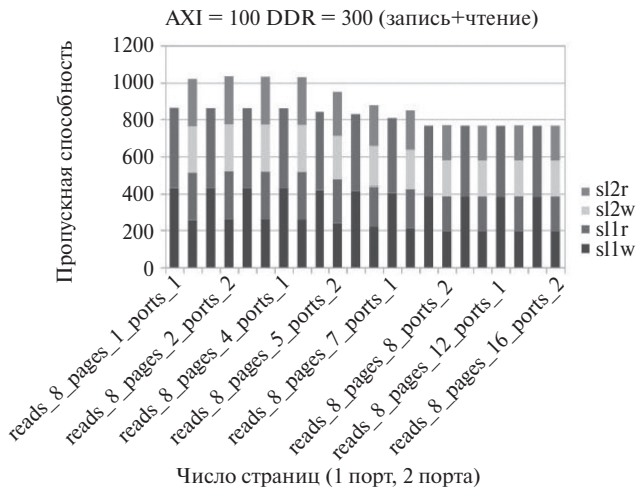

Рисунок 4. Производительность DDR для различного соотношения числа одновременно используемых страниц при частотах AXI = 100 МГц, DDR = 300 МГц

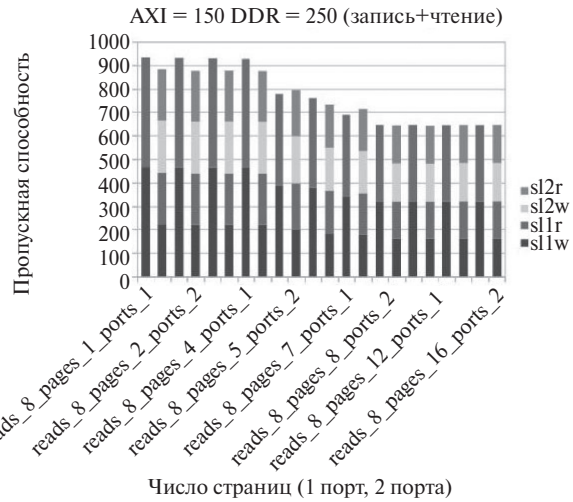

Рисунок 5. Производительность DDR для различного соотношения числа одновременно используемых страниц при частотах AXI = 150 МГц, DDR = 250 МГц

коммутации и памяти на комплексных тестах показал оттеснение трафика CPU (произвольный трафик из коротких транзакций) при непрерывных обменах крупными пачками по портам NAND и SPFR. Кроме того, достигаемая на выбранной технологии пропускная способность DDR является «узким местом» для работы всей системы, поэтому даже изменение системы арбитража и буферизации в DDR-контроллере привело бы к просадке производительности NAND-потоков, что также снижало бы производительность, но уже в другом месте.

Наиболее простым решением оказалось увеличение внутренней памяти CRAM и размещение в ней программы и наиболее часто используемых алгоритмов данных. В табл. 1 приведено среднее время передачи данных, подтверждающее оттеснение запросов от CPU трафиком с более высокой плотностью.

В табл. 2 приведены интегральные оценки производительности NAND-контроллера (размер страницы – 4 КБ, размер блока памяти – 256 КБ).

Пропускная способность для полезных данных, полученная при оценке производительности системы приведена в табл. 3.

Таблица 1. Эффект оттеснения запросов от CPU трафиком NAND и SPFR с более высокой плотностью

Поток	Среднее время передачи байта, пс
nand3_w	3318,68489583
nand2_w	3385,41666667
nand4_w	3447,265625
…	–
cpu1_r	29436,7913148
cpu1_w	32159,8223481

Анализ архитектуры и реализации ИС показал, что в зависимости от характера внешнего трафика система будет показывать производительность от 5 МБ/с (трафик из пакетов минимального размера по случайным адресам, при котором ограничивающим фактором является производительность NAND) до 200 МБ/с (200 – практически достижимый максимум для SPFR, при большом объеме пакетов производительность может падать до 140–180 МБ/с, ограничением при этом является пропускная способность DDR, в которых идут одновременные потоки от всех абонентов).

Два CPU оставляют запас прочности с точки зрения потенциальной оптимизации ПО таким образом, нижняя оценка по производительности может быть улучшена на практике.

Негативное влияние активности CPU на эффективность работы системы и большое время вычитки из памяти при высокой нагрузке системы приводят к необходимости размещения всей программы и временных данных во внутренней памяти процессора.

Научная новизна

По нашему мнению, критериям научной новизны соответствуют архитектура генератора тестов, ориентированного на воспроизведение сценариев использования СнК на окружении коммутационной логики, и система автоматизированного анализа результатов моделирования с использованием метрик эффективности коммутационной логики.

Заключение

Внедренный в маршрут набор инструментов позволяет добиться сходимости проекта с точки зрения удовлетворения требования технического задания (и предъявляемых со стороны целевых задач) по производительности и эффективности.

Таблица 2. Оценки производительности NAND-контроллера

Режим	Пропускная способность ONFI, МБ/с	Производительность в режиме SLC при чтении, МБ/с	Производительность в режиме SLC при записи, МБ/с
16-bit	400	304	160
8-bit	200	152	80

Таблица 3. Оценки производительности системы

Трафик	Требуемая средняя пропускная способность на кристальной коммутации, МБ/с	Пропускная способность для полезных данных (пакетов, приходящих по SpaceFiber), МБ/с
Трафик пакетами по 4 КВ	375	5–7
Трафик пакетами по 256 КВ	350	До 180
Смешанный трафик пакетами 4/256 КВ	308	50–60

Апробация маршрута, включающего все изложенные в статье решения, была проведена на проекте накопителя NAND-памяти с интерфейсом SpaceFiber и показала практическую применимость и ценность маршрута, ориентированного на достижение заданных параметров СнК, а также зрелость разработанных утилит и инструментария, использованного при анализе узких мест системы.

СПИСОК ЛИТЕРАТУРЫ

1. Wilcox P. Professional Verification: A Guide to Advanced Functional Verification. Kluwer Academic Publishers, 2004, 225 с.
2. Автоматизация верификации СНК на основе платформенного подхода / К.А. Жезлов, Я.С. Колбасов, А.В. Николаев, Ф.М. Путря, С.Е. Фролова // Электронные компоненты, 2016. С. 30–35.
3. Lim Z.N., Loh S.H., Lee S.W., Yap V.V., Ng M.S., Tang C.M. A Reconfigurable and Scalable Verification Environment for NoC Design. New Media Studies, 2013, pp. 1–5.
4. Requirements-driven Verification Methodology (for Standards Compliance). Available at: www.accellera.orq (accessed 01.05.2018)
5. Мешков А.Н., Рыжов М.П., Шмелев В.А. Развитие средств верификации микропроцессора «ЭЛЬБРУС» // Вопросы радиоэлектроники. 2014. Т. 4. № 3. С. 5–17.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Жезлов Кирилл Александрович, инженер, АО «Научно-производственный центр “ЭЛВИС”», 124498, Москва, Зеленоград, проезд № 4922, д. 4, стр. 2, тел.: 8 (985) 272-44-38, e-mail: kzhezlov@elvees.com.
Колбасов Ярослав Сергеевич, инженер, АО «Научно-производственный центр “ЭЛВИС”», 124498, Москва, Зеленоград, проезд № 4922, д. 4, стр. 2, тел.: 8 (926) 100-39-59, e-mail: kolbasov@elvees.com.
Николаев Артем Валерьевич, к.т.н., старший научный сотрудник, АО «Научно-производственный центр “ЭЛВИС”», 124498, Москва, Зеленоград, проезд № 4922, д. 4, стр. 2, тел.: 8 (906) 078-01-29, e-mail: anikolaev@elvees.com.
Путря Федор Михайлович, к.т.н., начальник лаборатории, АО «Научно-производственный центр “ЭЛВИС”», 124498, Москва, Зеленоград, проезд № 4922, д. 4, стр. 2, тел.: 8 (916) 572-60-52, e-mail: fputrya@elvees.com.
Фролова Светлана Евгеньевна, руководитель группы, АО «Научно-производственный центр “ЭЛВИС”», 124498, Москва, Зеленоград, проезд № 4922, д. 4, стр. 2, тел.: 8 (910) 453-18-34, e-mail: svetlanaf@elvees.com.

For citation: Zhezlov K.A., Kolbasov Ya.S., Nikolaev A.V., Putrya F.M., Frolova S.E. Steps of verification and validation route aimed at obtaining the SoC model able to perform target tasks and algorithms with specified characteristics. Voprosy radioelektroniki, 2018, no. 8, pp. 64–72.
DOI 10.21778/2218-5453-2018-8-64-72

K.A. Zhezlov, Ya.S. Kolbasov, A.V. Nikolaev, F.M. Putrya, S.E. Frolova

STEPS OF VERIFICATION AND VALIDATION ROUTE AIMED AT OBTAINING THE SOC MODEL ABLE TO PERFORM TARGET TASKS AND ALGORITHMS WITH SPECIFIED CHARACTERISTICS

In order to ensure the convergence of SoC design based on its performance on real applications the entire route of SoC development starting with the architectural design to the qualification tests for topological netlist should be focused on the early localization of performance issues and the confirmation of consumer characteristics of SoC before its tapeout. The article describes some of the details of of SoC verification flow, based on the convergence of performance requirements. Also the results of the work of tools used in research of SoC and its components at various abstraction levels are presented. Verification flow is based on the usage of templated test environments and unified performance analysis tools. Method of description of input stimuluses based on graphs is proposed. This method allows recovering the scenarios which discovered errors and simulateing of real traffic. Steps of system verification and validation route, oriented to the SoC model capable to perform targeted objectives and algorithms with preset characteristics are proposed. An example of a system verified by principle of performance metrics control is given.

Keywords: SoC, verification, performance analysis, usage scenarios, system level verification, standalone verification, prototype, high-level model.

REFERENCES

1. Wilcox P. Professional Verification: A Guide to Advanced Functional Verification. Kluwer Academic Publishers, 2004, 225 с.
2. Zhezlov K.A., Kolbasov Ya. S., Nikolaev A.V., Putrya F.M., Frolova S.E. Automation of SoC verification based on platform approach. *Elektronnye komponenty,* 2016, pp. 30–35 (In Russian).
3. Lim Z.N., Loh S.H., Lee S.W., Yap V.V., Ng M.S., Tang C.M. A Reconfigurable and Scalable Verification Environment for NoC Design. New Media Studies, 2013, pp. 1–5.
4. Requirements-driven Verification Methodology (for Standards Compliance). Available at: www.accellera.orq (accessed 01.05.2018)
5. Meshkov A.N., Ryzhov M.P., Shmelev V.A. Development of «ELBRUS» microprocessor verification means. *Voprosy radioelektroniki,* 2014, vol. 4, no. 3, pp. 5–17 (In Russian).

AUTHORS

Zhezlov Kirill, engineer, Joint Stock Company Research and Development Center ELVEES, 4/2, proezd № 4922, Zelenograd, Moscow, 124498, Russian Federation, tel.: +7 (985) 272-44-38, e-mail: kzhezlov@elvees.com.
Kolbasov Yaroslav, engineer, Joint Stock Company Research and Development Center ELVEES, 4/2, proezd № 4922, Zelenograd, Moscow, 124498, Russian Federation, tel.: +7 (926) 100-39-59, e-mail: kolbasov@elvees.com.
Nikolaev Artem, PhD, senior researcher, Joint Stock Company Research and Development Center ELVEES, 4/2, proezd № 4922, Zelenograd, Moscow, 124498, Russian Federation, tel.: +7 (906) 078-01-29, e-mail: anikolaev@elvees.com.
Putrya Fedor, PhD, head of laboratory, Joint Stock Company Research and Development Center ELVEES, 4/2, proezd № 4922, Zelenograd, Moscow, 124498, Russian Federation, tel.: +7 (916) 572-60-52, e-mail: fputrya@elvees.com.
Frolova Svetlana, team leader, Joint Stock Company Research and Development Center ELVEES, 4/2, proezd № 4922, Zelenograd, Moscow, 124498, Russian Federation, tel.: +7 (910) 453-18-34, e-mail: svetlanaf@elvees.com.