УДК 004.052.42 DOI 10.31114/2078-7707-2018-2-52-58

Подход к стохастическому тестированию RTL-моделей многоядерных микропроцессоров

Н.А. Гревцев[1,2], П.А. Чибисов[1]

[1]ФГУ ФНЦ НИИСИ РАН, г. Москва, chibisov@cs.niisi.ras.ru

[2]МФТИ ГУ, г. Долгопрудный

***Аннотация* — В статье предложен маршрут раннего тестирования моделей многоядерных микропроцессоров без создания отдельных генераторов тестов, направленных исключительно на многоядерное тестирование. В данной работе рассматривается способ адаптации имеющихся средств одноядерного стохастического тестирования под полноценный инструмент многоядерного тестирования. Построение многопоточных тестовых программ осуществляется генератором, основное предназначение которого заключается в генерации одноядерных тестовых программ. При этом сам генератор тестов не требует внесения значительных исправлений и доработок. Предложенный метод был успешно применен для тестирования RTL-модели разрабатываемого в ФГУ ФНЦ НИИСИ РАН двухъядерного микропроцессора с SMP.**

***Ключевые слова* — функциональная верификация, многоядерные микропроцессоры с общей памятью, когерентность кэш-памяти, стохастическое тестирование, генерация псевдослучайных тестов, MOESI-протокол.**

I. Введение

Стратегия верификации RTL-модели многоядерного микропроцессора, рассматриваемая в данной работе, состоит в том, чтобы осуществить как можно более полное тестирование проекта на ранних этапах цикла проектирования с использованием уже готовых инструментов тестирования одноядерного микропроцессора. Цель рассматриваемого системного подхода — верификация межъядерных взаимодействий между отдельными блоками подсистемы памяти, и самих блоков в масштабе всей системы непосредственно в процессе разработки модели. Следует подчеркнуть, что генератор одноядерных тестовых программ при этом не требует внесения значительных исправлений и доработок.

Многоядерная архитектура микропроцессора с симметричным доступом к памяти (SMP) позволяет использовать ресурсы дополнительных процессорных ядер для распараллеливания ресурсоемких вычислений или одновременного выполнения нескольких задач. Общий системный контроллер в такой системе обеспечивает доступ к ОЗУ для каждого микропроцессорного ядра, а также подключение периферийных устройств. Пример структурной схемы многоядерной системы на кристалле с архитектурой с симметричным доступом к памяти SMP показан на рис. 1.

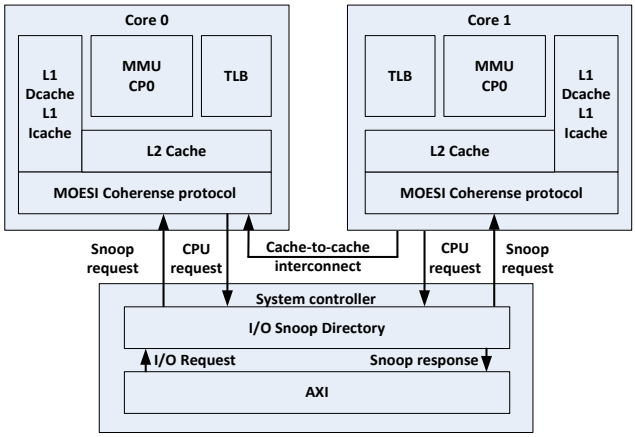

Рис. 1. Многоядерная архитектура с симметричным доступом к памяти SMP

Проблема целостности данных (иначе говоря, когерентности данных) может возникнуть в случае одновременного обращения двух или более вычислительных ядер к одним и тем же данным, если одно или более из этих обращений осуществляется на запись. Таким образом, наиболее актуальные данные могут оказаться в кэш-памяти одного из ядер и будут недоступны остальным устройствам в системе. Эта проблема сохранения целостности данных в многоядерных микропроцессорах решается с помощью протокола когерентности, согласно которому обеспечивается синхронизация данных между процессорными ядрами и периферийными устройствами.

Статья является продолжением исследований [1], [2], посвященных стохастическому тестированию моделей микропроцессоров. В последующих разделах статьи приводится описание предлагаемого нового подхода к раннему тестированию RTL-моделей многоядерных микропроцессоров, ключевыми моментами которого можно назвать акцент на раннем этапе проектирования, а также применимость однопоточных средств тестирования.

II. ОСОБЕННОСТИ ТЕСТИРОВАНИЯ ПОДСИСТЕМЫ ПАМЯТИ

При проектировании новых микропроцессоров может потребоваться исследование влияний новых микроархитектурных решений. Для этого создается ряд экспериментальных моделей, на которых оценивается эффективность решений на множестве ключевых критериев, таких как: сложность реализации в HDL-коде, временные затраты на реализацию, производительность, размер занимаемой площади на кристалле, энергопотребление. Разные подходы к разработке требуют итерационного создания экспериментальных моделей для оценки работоспособности и производительности принимаемых решений.

При каждом новом изменении работоспособность модели должна быть протестирована за ограниченное время. Например, в процессе разработки потребовалось оценить применимость одного из двух вариантов (инклюзивного и эксклюзивного) взаимодействия между кэш-памятью первого и второго уровня. Инклюзивная организация предполагает дублирование информации, находящейся в L1-кэше и L2-кэше, в то время как эксклюзивная кэш-память предполагает уникальность информации, находящейся в L1-кэше и L2-кэше. Для оценки оптимальности подобных принимаемых архитектурных решений необходима тестовая система, в которую входят тесты аттестации архитектуры и тесты производительности.

В статье предлагается маршрут раннего тестирования моделей многоядерных микропроцессоров и получение полноценного инструмента многоядерного тестирования за счет адаптации имеющихся средств одноядерного тестирования. Одним из методов тестирования моделей на системном уровне является метод направленного стохастического тестирования. При этом в условиях крайне ограниченных временных рамок и человеческих ресурсов требовалось осуществить тестирование разрабатываемой RTL-модели многоядерного микропроцессора без создания новых инструментов тестирования, направленных исключительно на верификацию протокола когерентности.

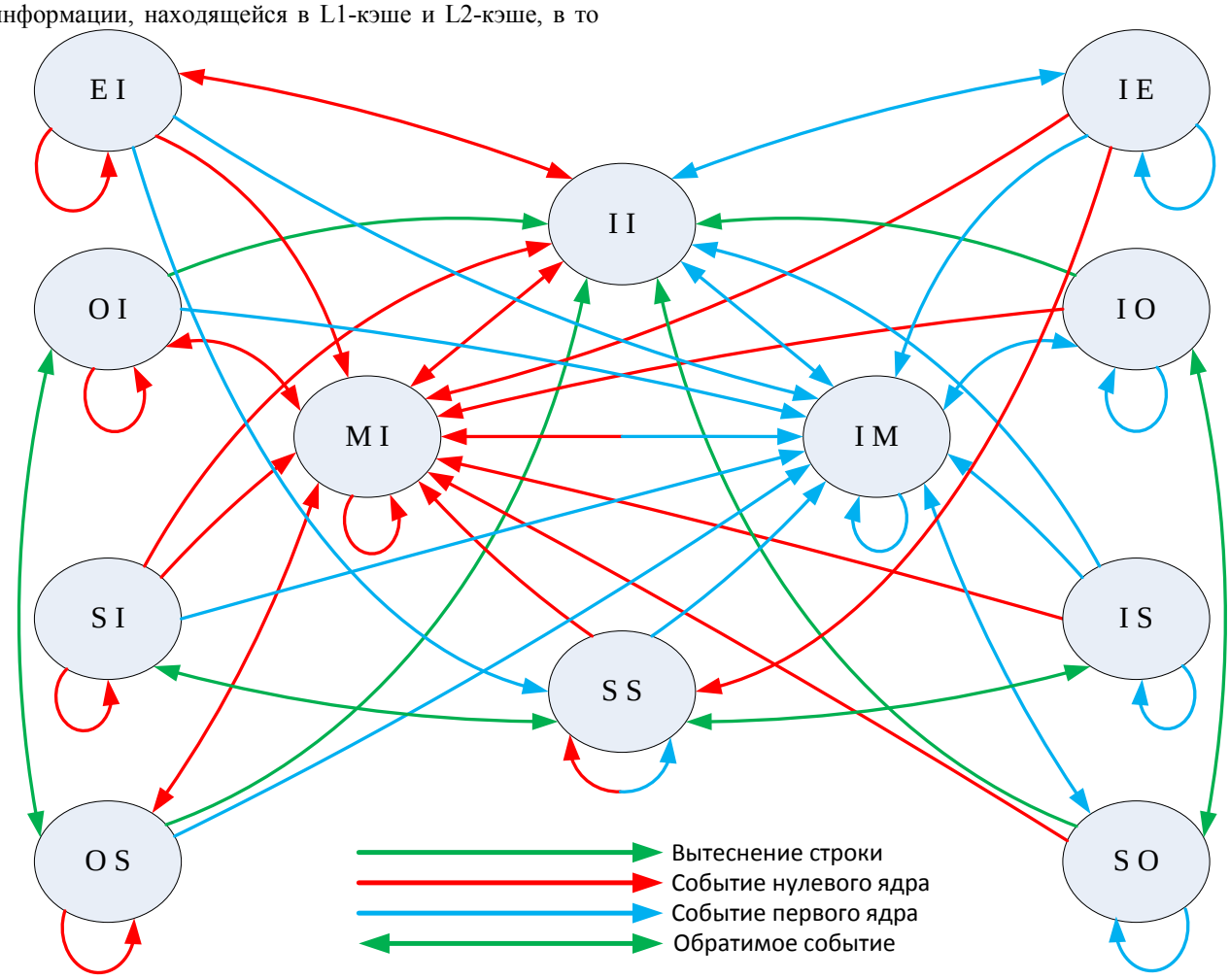

Рис. 2. Протокол когерентности кэш памяти MOESI, спроецированный на два ядра

Подсистема памяти современных многоядерных микропроцессоров представляет собой совокупность множества компонентов на одном кристалле: MMU, TLB, кэш-памяти всех уровней и их контроллеры, механизмы поддержки когерентности данных, системный контроллер, буферы предварительной подкачки данных, буферы упорядочивания записи данных. Наличие множества вычислительных ядер повышает комбинаторную сложность тестирования подсистемы памяти.

Протокол когерентности описывает состояние одной строки кэш-памяти относительно аналогичной строки в другом ядре и не затрагивает операции над другими строками кэш-памяти. Под состоянием строки кэш-памяти понимается состояние соответствующего контроллера кэш-памяти. Все состояния можно поделить на основные и промежуточные. Основные состояния определяются подмножеством состояний Modified, Owned, Exclusive, Shared, Invalid (MOESI). Переходы из одного основного состояния в другое в современных протоколах когерентности кэш-памяти происходят не мгновенно, а посредством переходных состояний [3].

Высокая сложность задачи проверки (верификации) корректности протоколов когерентности обуславливается тем, что простой перебор состояний быстро приводит к комбинаторному взрыву. В работе [4] показано, что протокол когерентности можно верифицировать отдельно по его спецификации, применяя метод верификации моделей (*model checking*). Однако для исследуемых в рамках настоящей статьи моделей стояла задача проверить не только проект и реализацию выбранного протокола когерентности, но и реализацию всей подсистемы памяти в совокупности с прочими микроархитектурными усовершенствованиями, упомянутыми выше.

На рис. 2 показаны все возможные переходы от операций загрузки-сохранения данных из памяти, осуществляемых вычислительными ядрами микропроцессора, на примере двух ядер. Большинство таких операций, приводящих к смене состояний конечного автомата MOESI, отображено на рис. 2. Однако в реальной системе они задействуют множество прочих блоков подсистемы памяти.

Одним из методов тестирования моделей на системном уровне является метод направленного стохастического тестирования, показавший высокую эффективность [2].

III. Применимость имеющихся инструментов

До разработки узконаправленного генератора случайных тестов для тестирования подсистемы памяти многоядерных систем предлагается доработать имеющиеся средства тестирования и создавать отдельные тесты для каждого ядра, а затем запускать их параллельно.

При этом ответственность за корректное распределение памяти между ядрами лежит на разработчике шаблона.

Для создания тестового кода для каждого ядра необходим собственный шаблон и один запуск одноядерного генератора псевдослучайных тестов. При построении шаблонов учитывается распределение памяти между ядрами, инструкции обращения к памяти выбираются случайно в пределах заданных ограничений. Пример маршрута стохастического тестирования для верификации двухъядерной системы показан на рис. 3.

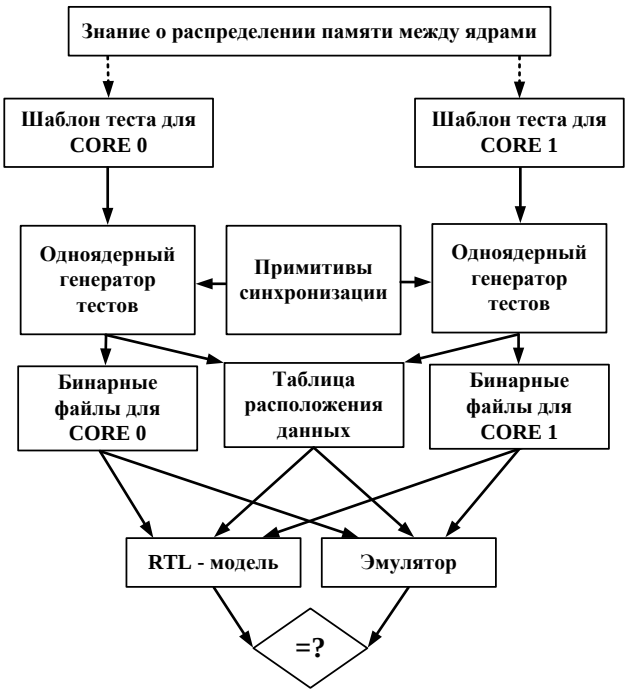

Рис. 3. Схема применения одноядерного генератора тестов для верификации многоядерных систем

Достоинства метода:

- возможность сразу начать тестирование, не тратя ресурсы на разработку отдельного генератора,
- масштабируемость: легко расширить подход на любое число ядер (2-16),
- возможность точного задания тестовых ситуаций с разными степенями свободы,
- отсутствие необходимости видоизменять процесс генерации при изменении различных архитектурных решений.

Недостатки:

- необходимость дополнительного контроля над распределением памяти между ядрами со стороны разработчика шаблонов,
- необходимость контроля синхронизации потоков,
- область тестирования ограничена подсистемой памяти.

IV. ПРОЦЕСС ГЕНЕРАЦИИ ТЕСТОВ ДЛЯ МНОГОЯДЕРНЫХ МИКРОПРОЦЕССОРОВ

В данной главе описаны основные этапы тестирования многоядерных подсистем памяти, соответствующие разным уровням завершенности функциональной разработки RTL-модели проектируемого микропроцессора.

A. Рукописные тесты, направленные на протокол

Процесс тестирования начинается с создания простых рукописных тестов на языке ассемблера, направленных на проверку протокола когерентности MOESI. В таких тестах отдельно проверяются все переходы между состояниями MOESI (рис. 2), формализуются и проверяются алгоритмы синхронизации ядер, которые будут использованы при дальнейшем тестировании. На примере таких тестов отлаживается механизм проверки корректности выполнения теста. Для этого используется система сравнения логов эталонного эмулятора и RTL-модели.

B. Рукописные тесты с самопроверкой

Исходная идея заключается в том, что ситуации, связанные с ложным разделением данных между потоками (*false sharing*) при запуске многопоточных приложений, приводят к потере производительности вычислений. Модификация даже одного байта приводит к обновлению целой строки кэш-памяти, и в этом случае могут возникать ситуации, когда потоки, параллельно выполняющиеся на разных ядрах, поочередно обновляют разные переменные, попадающие в одну и ту же строку кэш-памяти. В таких случаях обновление строки на одном ядре микропроцессора будет приводить к вытеснению соответствующей строки из кэш-памяти другого ядра [5].

В работах [6], [7] также исследуются способы обнаружения ложного совместного использования данных, и анализируется влияние на производительность конфликтов, связанных с использованием одной кэш-линии разными потоками. При проектировании параллельных программ рекомендуется избегать описанных выше конфликтов. Однако для того, чтобы протестировать межъядерные взаимодействия, имеет смысл намеренно (искусственно) создавать задачи, в которых память между ядрами распределена таким образом, что оба потока начинают бороться (итерационно обращаться) за одну кэш-линию. За эталонный результат таких тестов принимается результат той же задачи, реализованный с другой схемой распределения данных между потоками (не требующей пересылок данных между кэш-линиями разных ядер).

Самым простым примером рукописного теста со встроенной самопроверкой, основанного на ложном распределении данных, является классическая задача программирования — сложение двух массивов (рис. 4). Для организации самопроверки массивы складываются двумя различными способами.

Способ 1. Правильное с точки зрения распределения памяти между потоками разделение по кэш-линиям. Каждое ядро осуществляет обработку данных в пределах выделенных ему кэш-линий, например, нулевое ядро обрабатывает четные кэш-линии, первое — нечетные.

Способ 2. Применяется ложное разделение данных. Каждое ядро имеет доступ к различным местам совпадающих кэш-линий, что приводит к постоянным пересылкам данных между ядрами.

По окончании теста массивы, полученные разными способами, подлежат сравнению. Расхождение элементов массивов свидетельствует о потере данных при пересылке между кэш-памятью ядер или о получении доступа к устаревшей копии данных.

C. Псевдослучайная генерация тестов

Для повышения вероятности нахождения редких и труднообнаружимых ошибок, возникающих в результате взаимодействия нескольких инструкций в конвейере, а также в результате множества одновременных запросов в кэш-память, применяется метод стохастического тестирования, заключающийся в генерации псевдослучайных тестов по заданному шаблону инструкций [2].

Ниже будут рассмотрены три ступени псевдослучайного тестирования, разработанные для тестирования RTL-модели на разных стадиях функциональной разработки.

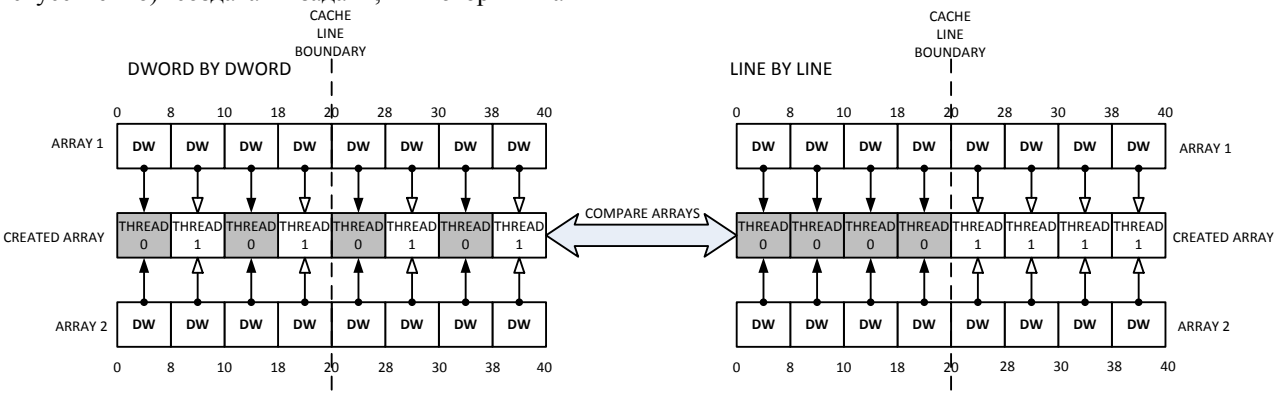

Рис. 4. Сложение массивов двумя способами как пример теста с самопроверкой

D. Изолированные друг от друга линии кэш-памяти

На начальной стадии тестирования RTL-модели многоядерного микропроцессора вычислительные ядра могут быть протестированы совместно, но с минимальным межъядерным обменом. В таком случае шаблоны для генератора псевдослучайных тестов устроены так, что каждое из ядер использует области данных, отображение которых на кэш-память обоих уровней будет являться персональным для каждого ядра. Ниже приведен пример описания физической памяти, задаваемой в настройках генератора тестов.

Name	Lower	Upper
data1_core0,	0x0020000,	0x00207FF,
data2_core0,	0x0030000,	0x00307FF,
data1_core1,	0x0020800,	0x0020FFF,
data2_core1,	0x0030800,	0x0030FFF,

В приведенном примере видно, что области данных для нулевого и первого ядра занимают различные части кэш-памяти первого и второго уровней и не пересекаются между собой.

Данное приближение было сделано для того, чтобы инструкции загрузки и сохранения от разных ядер в пределах одной итерации (между синхронизациями ядер) теста, во-первых, не могли писать данные в одно место, во-вторых, не обращались к совпадающей линии кэш-памяти. Иначе нельзя гарантировать, что обращения к одной области памяти (или одной линии кэш-памяти) от разных ядер на RTL-модели будут происходить в той же последовательности, что и на эталонном эмуляторе. В первом случае возникает неопределенность (*race condition*), какое из ядер успело первым изменить данные по совпадающему адресу, а во втором случае, даже при разделении памяти между ядрами, данные из разных областей попадают в разные секции одной линии кэш-памяти. Это приводит к невозможности добиться полного совпадения лог-файлов RTL-модели и эмулятора.

E. Перекрестный вторичный запуск

В случае изолированных строк кэш-памяти в конце теста все строки переходят в случайное модифицированное состояние. Это состояние может быть использовано в качестве отправной точки для построения более содержательного теста. Во второй половине задачи тестовый код, выполненный на нулевом ядре, передается на выполнение первому и наоборот. Тем самым гарантируется, что каждая запрошенная строка кэш-памяти этого ядра находится в модифицированном состоянии в кэш-памяти другого ядра. Этот прием дает возможность повысить эффективность тестирования механизмов межъядерных взаимодействий (временные взаимоотношения между ядрами и кэш-памятью ядер) за счёт большего разнообразия тестовых ситуаций. Структура теста, основанного на перекрестном вторичном запуске, с примерами переходов состояний MOESI показана на рис. 5.

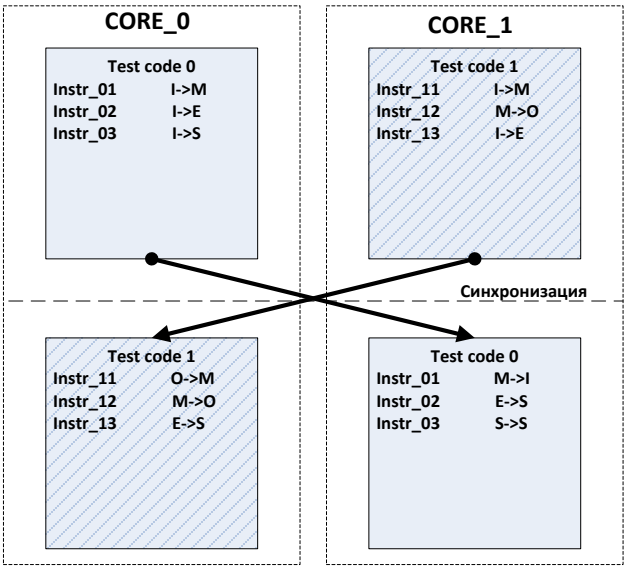

Рис. 5. Структура теста, основанного на перекрестном вторичном запуске

Первая половина теста на этой стадии по уровню покрытия конечного автомата протокола MOESI дает такой же результат, как и тесты из предыдущей стадии. Однако при перекрестном запуске (вторая половина теста) пространство достижимых состояний расширяется за счёт переходов MOESI от запросов другого ядра.

F. Структуры данных с чередованием

Из-за приближений, сделанных в предыдущих пунктах, упускается из рассмотрения целый класс потенциальных ошибок, связанных с одновременным обращением обоих ядер к одной области памяти.

Одно из возможных решений проблемы — структуры данных с чередованием (*interleaved memory structure*).

Для того чтобы обеспечить полноту тестирования протокола когерентности кэш-памяти, необходимо реализовать в тестовой системе возможность одновременного (в пределах зоны синхронизации) доступа к одной ячейке кэш-памяти обоими ядрами. Для этого на выбранную область памяти накладывается периодическая маска с размером ячейки меньше размера строки кэш-памяти. При этом нулевое ядро получает доступ ко всем нечетным элементам, а первое — к четным.

Важным следствием такой организации данных в памяти является тот факт, что оба ядра получают доступ к одной строке кэш-памяти одновременно и на запись, и на чтение, не нарушая целостности данных (обращения происходят в разные байты).

Благодаря такому разделению данных внутри любой отдельно взятой кэш-линии, каждая общая область, в которую разрешено писать более чем одному ядру, будет содержать детерминированные значения, и, таким образом, в любой момент времени все данные этой области являются предсказуемыми.

Чтобы протестировать оставшуюся непокрытой группу ситуаций, связанных с одновременным (в пределах нескольких тактов) доступом обоих ядер к одному и тому же адресу в памяти, создаются отдельные области «только на чтение» и «только на запись». Операции загрузки и сохранения в пределах этих областей происходят бесконтрольно из-за ситуаций, связанных с несинхронностью потоков обращений к памяти, работающих с общими данными (*race condition*). Такие обращения не подлежат проверке, так как направлены исключительно на поиск зависаний конвейера (*deadlock*).

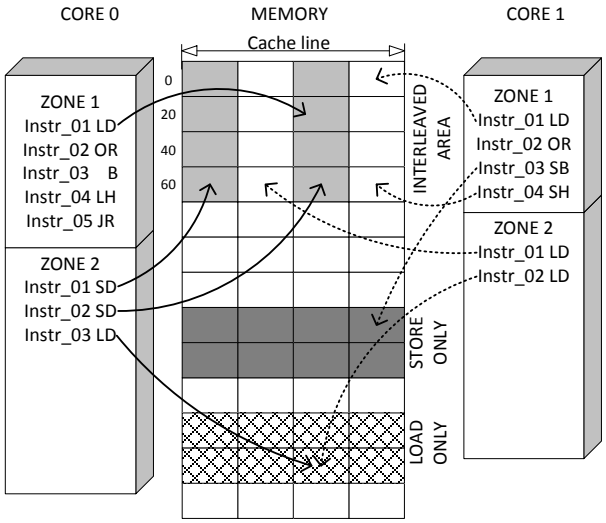

Рис. 6. Структурная схема распределения памяти между ядрами с использованием чередующейся маски

В приведенном примере (рис. 6) оба ядра имеют общие области памяти, но при этом одно ядро будет иметь доступ только к четным двойным словам, а другое — к нечетным. Таким образом, тесты позволяют охватить гораздо большее число сложно достижимых ситуаций, потенциально приводящих к ошибкам в подсистеме памяти. Пример распределения адресов в тестовых шаблонах может выглядеть следующим образом:

валидные адреса для ядра 0: XXXX0-XXXX7,

валидные адреса для ядра 1: XXXX8-XXXXF.

При использовании предложенной модели распределения памяти необходимо исключить из сравнения с эталонным эмулятором лог-файлы кэш-памятей из-за невозможности гарантировать одинаковую очередность обращений. При этом заведомо корректное состояние памяти и регистров общего назначения позволяет с высокой вероятностью находить ошибки RTL-модели.

В усовершенствованной системе тестирования одновременно применяются все вышеперечисленные методы, а также в ходе выполнения теста несколько раз может меняться периодичность разбиения interleaved-областей.

V. Оценка качества создаваемых тестов

Для оценки качества создаваемых псевдослучайных тестов была задана обобщенная метрика функционального покрытия, основанная на пространстве состояний, построенном на комбинации тестовых ситуаций, которые задаются такими событиями, как: тип операции, попадание или промах в кэш-память любого уровня, замещение модифицированных строк в кэш-памятях, виды переходов конечного автомата MOESI. На основании эвристического анализа вводится понятие условного тестового покрытия и определяется его максимальное значение, которое соответствует 100% тестового покрытия.

Введение метрики функционального покрытия позволяет количественно измерить степень завершенности работ по тестированию, а также оценить качество создаваемых генератором тестов.

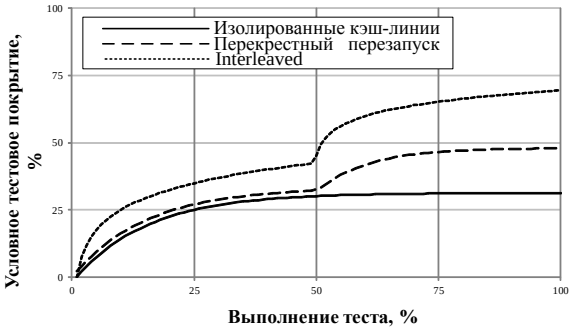

Рис. 7. Графическое отображение функционального покрытия для трех видов тестов

На рис. 7 показан рост функционального покрытия в ходе выполнения теста для трех описанных выше техник построения тестов. Резкий скачок в середине тестов (50% по оси абсцисс) обусловлен тем, что в этой точке происходит перекрестный вторичный перезапуск.

VI. Заключение

Предложенный в статье подход первоначально рассматривался как этап тестирования RTL-модели проектируемого микропроцессора на ранних стадиях, однако возможности такого подхода также представляют интерес и для тестирования модели многоядерного микропроцессора и на поздних этапах завершенности ее функциональной разработки.

Процесс тестирования начинается с создания простых тестов, направленных на проверку протокола когерентности MOESI. Для повышения вероятности нахождения редких и трудно обнаруживаемых ошибок в подсистеме памяти многоядерного микропроцессора предложен метод стохастического тестирования, основанный на использовании развитых одноядерных инструментов псевдослучайной генерации тестов, адаптированных под модели микропроцессоров с произвольным числом вычислительных ядер.

ЛИТЕРАТУРА

[1] Хисамбеев И.Ш., Чибисов П.А. Об одном методе построения метрик функционального покрытия в тестировании микропроцессоров // Проблемы разработки перспективных микро- и наноэлектронных систем - 2014. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2014. Часть2. С. 63-68.

[2] Гревцев Н.А., Хисамбеев И.Ш., Чибисов П.А. Исследование способов повышения эффективности стохастического тестирования моделей микропроцессоров // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №2. С. 8-15.

[3] Sorin, D. A Primer on Memory Consistency and Cache Coherence / Morgan & Claypool, 2012. - 210 p.

[4] Камкин А.С., Буренков В.С. Метод масштабируемой верификации PROMELA-моделей протоколов когерентности кэш-памяти // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №2. С. 54-60.

[5] Велесевич Е. А. Обнаружение и оценка количества промахов когерентности на основе вероятностной модели, Труды ИСП РАН, 27:4 (2015), 39-48

[6] T. Liu et al., "PREDATOR: Predictive false sharing detection", PPoPP 2014.

[7] Tongping Liu , Xu Liu, Cheetah: detecting false sharing efficiently and effectively, Proceedings of the 2016 International Symposium on Code Generation and Optimization, March 12-18, 2016.

A Practical Approach to Verification of Multicore Microprocessor Models

N.A. Grevcev[1,2], P.A. Chibisov[1]

[1]Scientific Research Institute of System Analysis (SRISA RAS), chibisov@cs.niisi.ras.ru

[2]MIPT

***Abstract* — in the paper, the early stage of verification technology for multicore processor models testing is proposed. We demonstrate the applicability of the single core verification method extension where a creating new multicore test generator is not required. The solution scheme deals with the adaptation method of some available single core stochastic testing approaches to a fully functional multicore testing tool.**

The proposed technique has been successfully applied to test RTL-model of dual-core microprocessor with SMP developed in SRISA. The discussed approach was initially considered to be a first stage of RTL-model testing, but the possibilities of the approach are also of interest for testing the model at the later stages of its design and functional maturity.

The testing process begins by creating simple random tests that check the MOESI coherence protocol. New advanced random testing method based on the usage of proposed interleaved memory structures is developed to increase the probability of finding rare and hard to detect bugs in the memory subsystem.

The great advantage of the proposed memory allocation structure is that both cores gain access to the same cache-line simultaneously for read and write. Despite this accessibility, the methodology avoids losses of data consistency due to cores access to different bytes. Furthermore, each common area that is allowed to write to more than one core will contain deterministic values at every time.

The proposed approach aims to detect failures in cache coherence protocol, memory subsystem and memory buffers. Also, this testing system helps to find machine state errors that lead to dead lock.

***Keywords* — functional verification, multicore, stochastic testing, pseudorandom tests generation, memory subsystem, SMP, MOESI, cache coherence, pre-silicon verification.**

REFERENCES

[1] KHisambeev I.SH., CHibisov P.A. Ob odnom metode postroeniya metrik funktsional'nogo pokrytiya v testirovanii mikroprotsessorov (On a method for constructing functional coverage metrics in microprocessor testing) // Problemy razrabotki perspektivnykh mikro- i nanoehlektronnykh sistem. Sbornik trudov / M.: IPPM RAN, 2014. S. 63-68.

[2] Grevtsev N.A., KHisambeev I.SH., CHibisov P.A. Issledovanie sposobov povysheniya ehffektivnosti stokhasticheskogo testirovaniya modelej mikroprotsessorov (Methods to improve efficiency of microprocessor model stochastic tests) // Problemy razrabotki perspektivnykh mikro- i nanoehlektronnykh sistem (MES'2016).

[3] Sorin, D. A Primer on Memory Consistency and Cache Coherence / Morgan & Claypool, 2012. - 210 p.

[4] Kamkin A.S., Burenkov V.S. Metod masshtabiruemoj verifikacii PROMELA-modelej protokolov kogerentnosti ke`sh-pamyati (A method for scalable verification of PROMELA models of cache coherence protocols) // Problemy` razrabotki perspektivny`x mikro- i nanoe`lektronny`x sistem (MES). 2016. №2. S. 54-60

[5] Velesevich E. A. Obnaruzhenie i ocenka kolichestva promaxov kogerentnosti na osnove veroyatnostnoj modeli (Number of coherence misses detection based on the probabilistic model), Trudy` ISP RAN, 27:4 (2015), 39-48

[6] T. Liu et al., "PREDATOR: Predictive false sharing detection", PPoPP 2014.

[7] Tongping Liu , Xu Liu, Cheetah: detecting false sharing efficiently and effectively, Proceedings of the 2016 International Symposium on Code Generation and Optimization, March 12-18, 2016.