В.С. Буренков (ЗАО «МЦСТ», МГТУ им. Н.Э. Баумана)

V. Burenkov

ГЕНЕРАТОР ТЕСТОВ ДЛЯ ВЕРИФИКАЦИИ ПРОТОКОЛА КОГЕРЕНТНОСТИ КЭШ-ПАМЯТИ

A TEST GENERATOR FOR CACHE COHERENCE PROTOCOL VERIFICATION

Рассматривается задача проверки аппаратной реализации протоколов когерентности кэш-памяти современных микропроцессорных систем. Описывается генератор тестов, основанный на формальной модели протокола. Показывается целесообразность применения генератора наряду со средствами верификации, создающими псевдослучайные тесты.

Ключевые слова: генератор тестов, model checking, Spin, протокол когерентности памяти, верификация.

This article examines correctness checking of hardware implementation of cache coherence protocols. The paper describes a test generator based on formal protocol model and shows the generator to be a practicable addition to widely used random test generators.

Keywords: test generator, model checking, Spin, cache coherence protocol, verification.

Введение

Современные высокопроизводительные вычислительные системы состоят из нескольких многоядерных процессоров, каждый из которых имеет доступ к общему адресному пространству и при этом физически обладает собственной памятью. Время доступа к различным участкам памяти в таких системах неодинаково, и для сокращения задержки при обращениях к памяти вычислительные ядра снабжаются локальной кэш-памятью. Более того, многие архитектуры добавляют еще один уровень кэш-памяти, разделяемый

всеми ядрами. Это приводит к проблеме обеспечения согласованного доступа к блокам кэш-памяти различных ядер, или когерентности.

В решении данной проблемы зачастую применяются аппаратные механизмы, реализующие протоколы когерентности кэш-памяти. Одновременная работа множества агентов (контроллеров кэш- и основной памяти), обменивающихся информацией в соответствии с протоколом, определяет колоссальную размерность пространства состояний протокола. Это, в свою очередь, обусловливает высокую сложность задачи проверки корректности протоколов (верификации).

1. Верификация протоколов когерентности

При верификации реализаций когерентного доступа к памяти важными являются два аспекта:

- правильность разработки самих протоколов,
- корректность аппаратной реализации этих протоколов, т.е. RTL-описания микропроцессора.

Прежде чем проверять аппаратную реализацию, следует быть уверенным в корректности самого протокола. Распространенная практика – анализ протокола вручную, а затем проверка его реализации тестами со случайными воздействиями – не дает гарантии корректности как протокола, так и системы. Случайные тестовые последовательности не обеспечивают полноты покрытия пространства состояний протокола за приемлемое время. Несмотря на то, что данные методы позволяют найти большое количество ошибок, некоторые сложные ошибки, связанные с передачей сообщений между частями верифицируемой системы, выполняемой в определенном порядке, могут оказаться необнаруженными.

Для преодоления этой проблемы уже несколько десятилетий ведутся исследования в направлении формальных методов, позволяющих проверить соответствие абстрактной

модели протокола (конечного автомата) его спецификации, т.е. набору свойств, которым должен отвечать протокол. Таким образом, формальные методы позволяют получить математическое доказательство корректности протокола (в пределах уровня детализации модели).

Распространенным формальным методом верификации является проверка моделей (model checking) [1], допускающая удобные способы спецификации свойств, описывающих желаемое поведение модели, полную автоматизацию проверки наличия этих свойств и генерацию контрпримеров, позволяющих отыскать источник ошибки. Данная статья посвящена использованию формальных моделей для проверки аппаратной реализации протоколов когерентности. В ней демонстрируется, каким образом наличие формальной модели может содействовать получению тестов, с помощью которых можно испытывать RTL- и другие модели микропроцессорной системы.

2. Формальная модель протокола когерентности

В предыдущих работах [2] показана применимость инструментального средства Spin [3], нацеленного на проверку корректности взаимодействующих параллельных асинхронных процессов, к написанию и проверке моделей протоколов когерентности памяти.

Spin предоставляет язык Promela, с использованием которого автором разработана формальная модель протокола выполнения когерентных обращений к памяти четырехъядерного микропроцессора «Эльбрус-4С». Поскольку протокол рассматривает состояние одной строки памяти и не затрагивает взаимодействие между операциями над различными кэш-блоками, в модели представлена одна кэш-строка. Обращения к памяти, помимо их подразделения на считывание (load) и запись (store), характеризуются *типом памяти*, определяющим, как они будут обработаны в системе. В формальной модели реализованы алгоритмы для типов памяти write back (определяющего режим отложенной записи в память) и write through (определяющего режим сквозной записи) [4].

Работу протокола когерентности для обращений с типом памяти write back можно кратко описать следующим образом. Получив от ядра исходную команду считывания или записи в память, кэш-память отправляет в системный коммутатор того процессора, к чьей памяти планируется обращение, соответствующий запрос (считывание и/или уничтожение). По приему этого запроса системный коммутатор формирует и рассылает снуп-запросы, отслеживающие состояние кэшей системы, и, возможно, запросы на считывание в память. Ответы на снуп-запросы (подтверждения или данные) отправляются запросчику, который, получив всю необходимую информацию, извещает системный коммутатор о завершении операции.

Порядок исполнения записей с типом write through более сложен и предполагает исполнение запросов других типов, например, запроса на запись в память. Обработка такого запроса требует от системного коммутатора выполнения следующих действий: приема исходного запроса, рассылки снуп-запросов, отправки запроса за данными инициатору исходного запроса, приема этих данных, сбора когерентных ответов, отправки данных в контроллер памяти и информирования запросчика о завершении операции. При этом остальные устройства взаимодействуют с системным коммутатором, отвечая на запросы.

В разработанной формальной модели функции частей микропроцессора, задействованных при обращении к памяти (ядер, кэш-памяти второго уровня L2, устройства обращения к памяти MAU, системного коммутатора процессора-владельца адреса, контроллера основной памяти), представлены Promela-процессами.

3. Архитектура генератора тестов

Задачей генератора является создание тестов на языке ассемблера Эльбрус, направленных на верификацию когерентных обращений к памяти и готовых для запуска на процессоре, процессорной системе, а также на различных моделях этой системы, включая RTL-модель. Содержательная часть тестов должна представлять собой последователь-

ность инструкций считывания и записи. В формальной модели протокола когерентности отражены события, соответствующие обращениям к памяти со стороны ядра. Путем трассировки этих событий, можно получить искомую последовательность инструкций.

Трассировка реализована с помощью операторов printf, которые вызываются всякий раз, когда происходит обращение к памяти. Для получения конкретного теста можно воспользоваться определенным в системе Spin режимом симуляции, при котором отражается и запоминается в журнале одна из возможных последовательностей исполнения операторов модели. Из журнала впоследствии можно извлечь информацию о пересылках сообщений между процессами модели, а также результаты индикации операторов printf.

Переход от формальной модели к тесту происходит путем анализа (и, возможно, модификации и уточнения) журнала работы системы Spin. Поскольку в формальной модели заданы все процессорные ядра верифицируемой системы, одного журнала достаточно для получения кода для всех ядер.

Разработанный генератор тестов состоит из нескольких функциональных компонентов (рис. 1) и осуществляет описанный переход от Promela-модели к ассемблерному тесту.

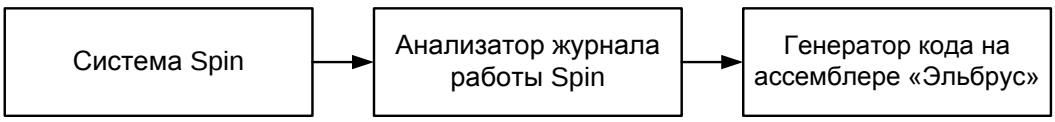

Рис. 1

Архитектура генератора тестов

В качестве основных параметров генератора пользователю доступны:

- маска ядер, указывающая, какие ядра узлов процессорной системы будут задействованы тестом. Предполагается работа с кластером из четырех процессоров «Эльбрус-4С» или «Эльбрус-8С», содержащим 16 и 32 ядра соответственно;
- параметр, определяющий количество обращений к памяти и коррелирующий с числом шагов симуляции модели.

Система Spin исполняет Promela-модель и фиксирует события в журнале в обобщен-

ном виде, позволяющем расширить набор тестов, формируемых генератором, в частности, увеличить диапазон адресов обращений к памяти.

Анализатор журнала работы Spin, использующий результаты предыдущих работ автора [5], извлекает необходимую информацию и, подставляя конкретные значения данных и адресов обращений к памяти, выдает код, пригодный для запуска на детальной программной модели подсистемы памяти Эльбрус, используемой в ЗАО «МЦСТ». В результате предоставляется возможность создания воздействий для тестовой системы, объединяющей RTL-модель процессора и программную модель подсистемы памяти, призванной облегчить и ускорить отладку процессора [6].

Генератор кода на ассемблере Эльбрус формирует финальный тест на основе результатов анализатора журнала работы Spin. Этот процесс сопровождается созданием таблицы страниц (с целью использования в тестах виртуальной адресации) и получением кода сравнения данных по каждому адресу.

Виртуальная адресация позволяет расширить возможности тестов проверкой блоков устройства управления памятью, отвечающих за трансляцию виртуального адреса в физический. Более того, тип памяти write through может быть задействован только путем использования соответствующего атрибута виртуальной страницы. Чтобы использовать виртуальный режим, в код теста добавляется введенная архитектурой Эльбрус четырехуровневая таблица страниц, позволяющая выполнять преобразование как адресов обращений за данными, так и адресов, по которым расположен сам код теста. Таким образом, у пользователя есть возможность задания количества используемых физических и виртуальных адресов.

Основную трудность при написании генератора составила разработка формальной модели на языке Promela. При этом существенно, что задачи, связанные с добавлением в тесты таблицы страниц, синхронизацией процессорных ядер, инициализацией, являются общими для различных генераторов тестов на проверку подсистемы памяти, поэтому их

решение может быть использовано заново в других проектах.

Части генератора, осуществляющие обработку результатов работы Spin, представляют собой код небольшого объема на языке Perl. Поэтому создание первых версий генераторов на основе формальных моделей не является слишком трудоёмким и при этом дополняет генераторы, основанные на псевдослучайных воздействиях. Модель на Promela занимает более 1000 строк кода, а скрипты на Perl – несколько сотен строк. Также несколько сотен строк занимает вспомогательный код на ассемблере.

4. Результаты верификации

На момент написания статьи представленный генератор использовался в течение полутора месяцев для верификации четырехъядерного и восьмиядерного микропроцессоров серии «Эльбрус», разрабатываемых в ЗАО «МЦСТ». Он позволил найти в обоих процессорах ошибки, которые не были обнаружены другими средствами.

5. Направления дальнейших исследований

Важной проблемой дальнейшей работы является формулирование вердикта о прохождении теста.

Тест может считаться «не прошедшим» по нескольким причинам: при моделировании сработала проверка, встроенная в Verilog-описание процессора; исполнение теста не завершилось за отведенное время; программная проверка состояния процессора выдала отрицательный результат. Существенно, что применение эффективных схем сличения данных, необходимое в ассемблерных тестах многопроцессорной системы, создает нетривиальные проблемы из-за неопределенности в порядке исполнения инструкций обращения к памяти, свойственной архитектуре Эльбрус.

В основе созданного генератора лежит формальная модель протокола когерентности памяти микропроцессора «Эльбрус-4С». В силу того что этот микропроцессор был взят за

основу при проектировании восьмиядерного микропроцессора следующего поколения «Эльбрус-8С», многие решения предполагается использовать и в формальной модели последнего. В то же время, целесообразны и определенные усовершенствования, такие как замена или дополнение случайного выбора альтернатив, свойственного системе Spin, использование трасс, полученных направленными способами, или создание инструментов верификации, в отличие от текущего варианта учитывающих информацию о внутренних событиях модели и сопоставляющих с ними работу процессоров.

Заключение

Разработанный генератор тестов демонстрирует, что, имея формальную модель протокола когерентности памяти, можно построить и механизм создания тестов, направленных на проверку когерентности обращений к памяти, реализованных в конкретной микропроцессорной системе.

Генератор следует рассматривать как средство, дополняющее другие генераторы, основанные на псевдослучайных воздействиях, а также повышающее степень уверенности в корректности функционирования проектируемых микропроцессоров.

Литература

1. Clarke E.M., Grumberg O., Peled D. Model Checking. – MIT Press, 1999, 314 pp.

2. Буренков В.С. Анализ применимости инструмента Spin к верификации протоколов когерентности памяти. – «Вопросы радиоэлектроники», сер. ЭВТ, 2013, вып. 3, с 126-134.

3. Holzmann G.J. The Spin Model Checker: Primer and Reference Manual. – Addison-Wesley, 2003. 608 pp.

4. Intel 64 and IA-32 Architectures Software Developer’s Manual Volume 3A: System Programming Guide, Part 1, 2010.

5. Буренков В.С. Инструмент верификации протокола когерентности памяти. – Молодежный научно-технический вестник, 2013, № 1.

6. Куцевол В.Н., Мешков А.Н., Петроченков М.В. Методология верификации протокола когерентности микропроцессора «Эльбрус-2S». – «Вопросы радиоэлектроники», сер. ЭВТ, 2013, вып. 3, с. 107-117.