П.В. Фролов (ОАО «ИНЭУМ им.И.С. Брука», ЗАО «МЦСТ»)

P. Frolov

ГЕНЕРАЦИЯ СЛУЧАЙНЫХ ТЕСТОВ СИСТЕМНОГО УРОВНЯ ДЛЯ МИКРОПРОЦЕССОРОВ С АРХИТЕКТУРОЙ «ЭЛЬБРУС»

RANDOM SYSTEM-LEVEL TEST GENERATION FOR ELBRUS ARCHITECTURE MICROPROCESSORS

Рассмотрена структура для разработки генераторов тестов системного уровня на основе функциональной модели тестируемой системы. Описан механизм функционирования генератора на её основе. Показана применимость на примере реализованных генераторов.

A model-based software framework for development of random system-level test generators is considered. The algorithm of framework-based generator functioning is described. Applicability of the framework is demonstrated with generators have been designed.

Ключевые слова: системная верификация, генерация тестов, тестирование на основе моделей.

Key words: system-level verification, test generation, model-based testing.

Введение

Современные микропроцессоры с архитектурой «Эльбрус» имеют многоядерную структуру, содержат ряд различных по функциональности устройств северного моста (контроллеров прерываний, модулей трансляции адресов в подсистеме ввода/вывода, межъядерных и межпроцессорных коммутаторов) и подсистему памяти, которая обеспечивает поддержку различных адресных пространств с трансляцией адресов и включает кэш-памяти различных уровней, средства обеспечения когерентности данных, многочисленные буферы и коммутаторы [1]. Всё это определяет высокую комбинаторную слож-

ность верификации, резко ограничивающую применение формальных методов отдельными узкими местами или устройствами.

Одновременное функционирование многих устройств в составе системы создает самые различные динамические ситуации, вероятность воспроизведения которых растет с увеличением числа тестов. Для увеличения тестового покрытия активно применяется автоматическая генерация тестов [2], при которой формирование исходного кода тестовой программы осуществляется случайным образом с учётом заданных ограничений. Ограничения позволяют нацеливать тесты на воспроизведение определённых ситуаций, но при различных наборах случайных параметров.

Развитие архитектуры «Эльбрус», связанное с появлением новых инструкций, устройств и конфигураций вычислительных комплексов, ставит вопрос о разработке и расширении функциональности генераторов псевдослучайных тестов, направленных на верификацию системы в целом. К настоящему времени в ЗАО «МЦСТ» созданы различные генераторы псевдослучайных тестов для верификации устройств процессорного ядра: целочисленной и плавающей арифметики, устройства управления, декодирования, MMU и других. Помимо этого, реализованы генераторы многоядерных тестов, функционирование которых, однако, ограничено проверкой подсистемы памяти.

Для генерации тестов системного уровня необходимо объединение существующих реализаций, что позволит осуществлять работу с различными устройствами системы в рамках одного теста. Серьезным препятствием для этого является то, что используемые сейчас генераторы имеют независимые коды и написаны на разных языках.

Таким образом, необходимо спроектировать структуру, позволяющую применить существующие генераторы для создания случайных тестов системного уровня и предоставляющую возможность дальнейшего развития и расширения тестовой функциональности. В статье представлен принципиальный подход, который положен в основу инициированной работы по решению проблемы.

1. Общие требования к тесту системного уровня

В результате анализа задачи были сформулированы следующие требования.

Перед исполнением теста с псевдослучайным кодом необходимо инициализировать тестируемую систему. При этом с целью проверки работы системы в разных режимах следует обеспечить возможность для параметризации процедуры инициализации, поскольку многие параметры невозможно изменить после её окончания. Например, изменение границ диапазона физических адресов оперативной памяти из теста, код которого размещён в этом диапазоне, может привести к неопределённому поведению системы.

После инициализации системы тест, рассчитанный на проверку одного ядра, выполняет случайную тестовую последовательность инструкций, заканчивающуюся процедурой проверки, которая обычно контролирует корректность данных в отдельных областях памяти и значений регистров в процессоре и тестируемых устройствах. Для генерации кода этой процедуры удобно использовать функциональную модель тестируемой системы.

В случае тестов, рассчитанных на многоядерную структура микропроцессора, необходимо добавить в код процедуры синхронизации, которые обеспечивают корректность работы различных ядер с общими ресурсами, такими как память или периферийные устройства. В частности, при тестировании механизма обеспечения когерентности памяти следует сформировать поток параллельных обращений в одни лайны кэш-памятей из разных ядер. Для этого необходимо синхронизировать начало исполнения кода, инициирующего эти обращения на разных ядрах. Тестовая последовательность инструкций между двумя соседними вызовами процедур синхронизации может быть получена с помощью уже разработанных генераторов тестов, рассчитанных на одно ядро, при условии корректного распределения разделяемых ресурсов.

2. Структура генерации тестов

Обобщенная структура, позволяющая решить поставленную задачу, приведена на рис. 1. Кодогенераторы осуществляют формирование случайного исполняемого кода, сохраняемого в образе памяти; функциональная модель исполняет этот код, обеспечивая таким образом возможность получения текущего состояния тестируемой системы. Модуль управления кодогенераторами задаёт ход генерации теста путём настройки диспетчера запросов, который, в свою очередь, обеспечивает обработку обращений функциональной модели в память за исполняемым кодом и данными.

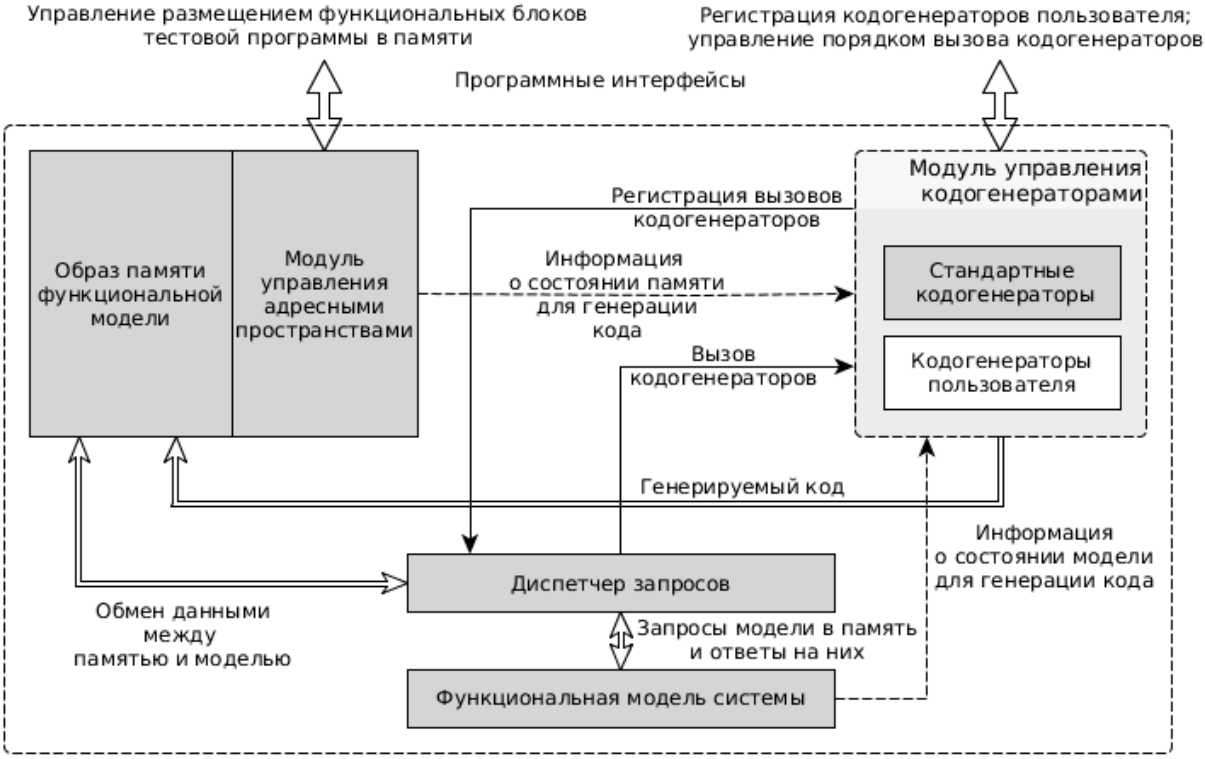

Рис. 1

Структура, обеспечивающая генерацию случайных тестов системного уровня

Разработка генератора тестов ведётся на основе описываемой структуры: пользователь реализует и регистрирует в модуле управления набор кодогенераторов, вызываемых в ходе генерации теста для получения отдельных блоков исполняемого кода.

Процесс генерации теста состоит из двух фаз: определения параметров режима работы системы и формирования случайного исполняемого кода. В первой фазе с учётом заданных пользователем ограничений случайным образом устанавливаются значения пара-

метров, используемых при инициализации системы. Определяется карта памяти системы (рабочие диапазоны физических адресов), и в образе памяти размещается исполняемый код библиотечных процедур (инициализации системы и работы со специализированными устройствами). Во второй фазе структура осуществляет последовательный вызов функции исполнения команды моделью, которая отправляет запрос в память для получения этой команды. Сначала из образа памяти читается код процедуры инициализации, заканчивающийся командой перехода по адресу начала случайной последовательности инструкций. При получении запроса модели первой инструкции этой последовательности вызывается предварительно зарегистрированный пользователем кодогенератор, который формирует участок исполняемого кода и помещает его в образ памяти по адресу запроса. Из образа памяти сгенерированная команда передаётся функциональной модели тестируемой системы, которая её исполняет и меняет своё внутреннее состояние. Следует отметить, что генерация теста идёт параллельно с его исполнением на функциональной модели, и это позволяет реализовывать кодогенераторы, формирующие последовательность инструкций на основе текущего состояния функциональной модели.

В описываемом проекте в качестве функциональной модели использовалась ранее разработанная в ЗАО «МЦСТ» модель вычислительного комплекса на основе микропроцессора «Эльбрус-4С» [3]. Она реализует систему команд микропроцессора и охватывает архитектуру вычислительного комплекса в целом. Модель компилируется в отдельную библиотеку, предоставляющую следующие программные интерфейсы:

- управление моделью (исполнение одной или нескольких команд, сохранение и восстановление состояний модели);
- доступ к внутреннему состоянию модели (получение текущих значений регистрового файла вычислительных ядер и системных регистров);
- регистрация функций, вызываемых при обращениях модели в память (обратный вызов).

Эти интерфейсы задействуются при последовательном вызове модельных функций исполнения команд и обращения к памяти вычислительного комплекса.

Обращения функциональной модели в память обрабатывает диспетчер запросов; атрибутами запроса являются физический адрес, размер обращения и идентификатор устройства, инициировавшего запрос. В случае записи в память запрос содержит данные, при чтении система должна сформировать ответ с данными. Идентификатор устройства позволяет различать запросы данных и исполняемого кода тестовой программы. Когда от модели поступает запрос случайной последовательности по адресу ее начала, диспетчер вызывает соответствующий зарегистрированный кодогенератор, который помещает сформированную им последовательность инструкций в образ памяти. После окончания работы кодогенератора диспетчер считывает требуемую команду из образа памяти и возвращает её модели.

Данные, полученные при моделировании системы в ходе генерации теста и помещённые в образ памяти, используются для формирования эталона при реализации процедуры самопроверки теста.

Модуль управления кодогенераторами предоставляет программный интерфейс для регистрации пользовательских кодогенераторов, параметры которой позволяют определять порядок и условия их вызова. Это позволяет реализовать различные сценарии тестирования. Кроме того, в рассматриваемой структуре могут использоваться и стандартные кодогенераторы, примером которых служат реализованные в данном проекте кодогенераторы для встраивания в тест процедур самопроверки, сравнивающих текущие значения регистров с эталонными, и кодогенераторы, которые осуществляют вызов библиотечных функций, предназначенных для работы со специализированными устройствами.

При использовании различных кодогенераторов особый интерес представляет совместное влияние сформированных ими кодов на тестируемую систему, позволяющее получить различные зависимости по ресурсам в генерируемых тестовых программах. Харак-

тер зависимостей и механизм их проверки в общем виде не подлежат обобщению и определяются генератором. Однако для реализации управления памятью необходим инструмент распределения пространства физических и виртуальных адресов между блоками исполняемого кода и данных генерируемого теста. Эту функцию осуществляет модуль управления адресными пространствами. С учётом конфигурации системы (количество процессоров, объём и структура оперативной памяти, наличие южных мостов) он определяет границы используемых тестом диапазонов физических адресов системы. Полученная карта памяти используется для размещения в оперативной памяти системы рабочих блоков тестовой программы и отображения на пространство физических адресов программно видимых областей периферийных устройств. Адреса размещения и границы могут быть заданы явно или получены случайным образом с учётом заданных пользователем ограничений.

При генерации теста, использующего трансляцию адресов, возникает задача размещения рабочих блоков в пространствах виртуальных адресов и их корректного отображения на пространство физических адресов. Модуль управления адресными пространствами реализует модели трансляции MMU ядра (трансляция адресов для инструкций ядра и исполняемого кода) и IOMMU северного моста (трансляция адресов в подсистеме ввода/вывода) и хранит образы виртуальных пространств, используемых в тесте, а также предоставляет функциональность получения кода таблицы трансляции для образа виртуального пространства теста.

3. Реализация и применение

В результате работы по решению поставленной проблемы была спроектирована и реализована на языке C++ описанная выше обобщенная схема генерации тестов системного уровня. Разработан набор кодогенераторов, осуществляющих формирование стандартных процедур самопроверки. Поддержана возможность встраивания в тестовую програм-

му процедур, написанных на языке C, что позволило использовать уже применяющиеся в ручных тестах библиотечные функции для работы с периферийными устройствами. В качестве аргументов эти процедуры получают параметры тестируемых устройств, сгенерированные случайным образом с учётом заданных ограничений.

Процедура инициализации тестируемой системы, параметризованная с помощью сгенерированных случайным образом значений, была унифицирована с загрузочным кодом, также используемым при разработке ручных тестов.

К настоящему времени на данной платформе реализованы два генератора для тестирования подсистемы памяти. Один из них является адаптацией разработанного ранее на языке Python многоядерного генератора тестов, направленных на верификацию механизма поддержки когерентности. В результате его усовершенствования была значительно увеличена скорость генерации и ослаблены ограничения на размер тестовых программ. Второй генератор разработан на языке C++ уже на основе приведенной в статье обобщенной схемы. Тесты осуществляют проверку функциональности устройств, осуществляющих DMA-обмен, на фоне обращений вычислительных ядер в память и в условиях программного изменения режима питания ядра.

Заключение

На примере ряда разработок в настоящее время показана применимость описанной в статье обобщенной схемы для реализации на её основе генераторов случайных тестов системного уровня. Использование библиотеки функций, обеспечивающих работу со специализированными устройствами и инициализацию тестируемой системы, позволило уменьшить трудоёмкость сопровождения программного кода, а также дало возможность генерации тестов, рассчитанных на верификацию нескольких версий микропроцессоров ЗАО «МЦСТ». Дальнейшие работы связаны с использованием существующих генераторов одноядерных тестов для реализации кодогенераторов, позволяющих увеличить набор од-

новременно работающих подсистем микропроцессора в рамках одного теста.

Литература

1. Ким А.К., Перекатов В.И., Ермаков С.Г. Микропроцессоры и вычислительные комплексы семейства «Эльбрус»: Учеб. пособ. СПб.: Питер, 2013.

2. Adir A., Almog E., Fournier L., Marcus E., Rimon M., Vinov M., Ziv. A. Genesys-Pro: innovations in test program generation for functional processor verification. – «Design \& Test of Computers, IEEE», vol. 21, no. 2, pp. 84, 93, Mar-Apr 2004.

3. Гурин К.Л., Мешков А.Н., Сергин А.В., Якушева М.А. Развитие модели подсистемы памяти вычислительных комплексов серии «Эльбрус». – «Вопросы радиоэлектроники», сер. ЭВТ, 2010, вып. 3.