К.т.н. И.А. Стотланд, А.А. Лагутин (ЗАО «МЦСТ»)

I. Stotland, A. Lagutin

ПРИМЕНЕНИЕ ЭТАЛОННЫХ СОБЫТИЙНЫХ МОДЕЛЕЙ ДЛЯ АВТОНОМНОЙ ВЕРИФИКАЦИИ МОДУЛЕЙ МИКРОПРОЦЕССОРОВ

APPLICATION OF REFERENCE TIME-ABSTRACT MODELS TO STANDALONE VERIFICATION OF MICROPROCESSOR UNITS

Рассмотрены основные способы построения тестовых систем и эталонных программных моделей для автономной верификации модулей микропроцессоров. Предложены подходы к организации взаимодействия тестовых систем и эталонных событийных моделей, а также методы адаптации компонентов программных моделей микропроцессоров. Описан опыт разработки и применения высокоуровневой событийной модели для автономной верификации неконвейерного модуля микропроцессора.

Ключевые слова: автономная верификация, тестовая система, событийная модель, верификация на основе моделей.

The main methods of testbenches and reference models design for standalone verification of microprocessor units are considered in this paper. The approaches to communication of testbenches and reference models and adaptation parts of microarchitecture simulators are proposed. The experience of standalone verification based on time-abstract model of the microprocessor unit is described.

Keywords: stand-alone verification, testbench, time-abstract model, model-based verification.

Введение

Прогресс в проектировании микропроцессоров характеризуется постоянным ростом их сложности и быстродействия наряду с всё возрастающими требованиями к корректности функционирования и сокращению сроков проектирования. Согласно данным Дж. Бер-

герона [1] 70% ошибок вносятся в начале проектирования, и некоторые из них удается устранить только на этапе отладки изготовленной системы. Значительные усилия прилагаются для изменения этой тенденции путем применения различных методов и средств верификации, на которую по данным различных исследований [1, 2, 3] приходится более половины от общего времени разработки микропроцессора.

Одним из основных способов контроля корректности при проектировании микропроцессоров является *динамическая верификация*, в основе которой лежат генерация входных воздействий и проверка правильности реакции на них в процессе моделирования. Выделяют два основных уровня динамической верификации микропроцессоров – модульный и системный. Объектом системной верификации является RTL-модель микропроцессора или микропроцессорной системы в целом. При модульной верификации функциональные блоки, входящие в состав микропроцессора, проверяются автономно, поэтому ее также называют *автономной* (standalone verification [2]). Основным преимуществом автономной верификации является возможность проводить тестирование на самых ранних стадиях разработки, не дожидаясь спецификации и реализации всей системы в целом, по мере готовности RTL-модели и спецификации модуля. На модульном уровне также возможно создание необходимой динамики работы RTL-модели для достижения критических ситуаций (переполнение буферов, блокировок и других редких состояний). Кроме того, время локализации и исправления ошибок при автономной верификации значительно меньше, чем при системной, что дает возможность сократить сроки отладки RTL-модели.

В статье рассмотрены подходы к автономной верификации модулей микропроцессоров с использованием эталонных программных моделей на примере хост-контроллера микропроцессора «Эльбрус-8С», разрабатываемого в ЗАО «МЦСТ».

1. Тестовые системы автономной верификации микропроцессоров

Тестовая система, применяемая при верификации сложных промышленных RTL-

моделей, в общем случае решает три задачи: генерацию входных воздействий, проверку правильности выходных реакций и оценку полноты тестирования. Соответственно, ее основными компонентами являются генератор входных воздействий (stimulus generator), устройство проверки (checker) и анализатор (scoreboard).

Одной из наиболее важных проблем, решаемых при разработке тестовых систем, является проверка правильности результатов моделирования. С этой точки зрения они подразделяются на два основных типа – детерминированные тестовые системы и тестовые системы на основе эталонных моделей [3]. В первом случае при проверке не только анализируются выходные сигналы, но и описывается правильная функциональность верифицируемого модуля, что связано со значительными трудозатратами и расходом времени. В связи с этим определенное преимущество имеет проверка эталонной модели, которая разрабатывается одновременно с тестовой системой, или в данном качестве используется уже готовая программная модель верифицируемого модуля. Это существенно сокращает сроки автономной верификации и дает возможность повысить ее качество, т.к. модель и ее тесты разрабатываются разными специалистами, что дает возможность избежать ошибок при интерпретации проверяемой спецификации.

Существует ряд стандартных подходов к построению тестовых систем: расширенная методология верификации (AVM, Advanced Verification Methodology); методология верификации, разработанная компанией Synopsys (VMM, Verification Methodology Manual); открытая методология верификации (OVM, Open Verification Methodology); универсальная методология верификации (UVM, Universal Verification Methodology). Каждая из методологий предлагает дополненный документацией набор библиотечных средств для построения повторно применяемых тестовых систем и их компонентов. Однако в любом варианте приведенные методологии не включают построение эталонных моделей и организацию их взаимодействия с тестовыми системами. Этим обусловлена актуальность формирования и реализации подходов, решающих данные проблемы.

2. Подходы к построению эталонных моделей

Под «эталонной» будем подразумевать программную модель верифицируемого модуля на языке C++, созданную на том же или более высоком уровне абстракции, что и его RTL-модель. Существуют также другие решения – поведенческие модели на языках описания аппаратуры, таких как SystemVerilog и «e», или модели, построенные в соответствии с концепцией TLM-2.0 на языке SystemC. Однако программные модели на C++ обладают рядом преимуществ: простотой разработки, широким распространением языка, удобством компиляции и отладки, быстродействием, возможностью повторного применения в программных моделях микропроцессоров (архитектурных симуляторах). Такие эталонные модели также называют дискретно-событийными [4].

В дискретно-событийных моделях значения переменных изменяются мгновенно, отражая состояние системы в определенные моменты времени. Обновление состояния дискретно-событийной модели выполняется в процессе обработки каждого события. Можно выделить три типа дискретно-событийных моделей: потактово-точные (потактовые), дискретно-событийные с учетом времени и событийные [4].

При моделировании цифровых устройств событием для *потактовых моделей* является сигнал синхроимпульса. Потактовые модели учитывают временны́е особенности работы устройства и моделируются с учетом реальных временны́х задержек [5].

Дискретно-событийные модели с учетом времени более абстрактны, чем потактовые, но значительно проще в реализации и не требуют подробной (с точностью до такта синхроимпульса) спецификации. При таком подходе применяется специальный планировщик, который вычисляет имитационное время следующего события, после чего программа моделирует часть работы системы в дискретном промежутке между текущим и следующим событием.

При *событийном моделировании* предполагается, что все события и связанные с ни-

ми изменения в состоянии происходят мгновенно. Модель, способную инициировать события, будем называть *активной событийной моделью*. Если же модель меняет свое состояние в результате внешних событий и выдает реакции только по внешнему вызову соответствующей функции, то она рассматривается как *пассивная событийная модель*. Пассивные модели проще в реализации, но не могут самостоятельно вырабатывать воздействия на другие модели, что усложняет их применение в составе сложных многомодульных систем.

От типа эталонной модели зависит архитектура и сложность тестовой системы: чем выше уровень абстракции эталонной модели, тем ниже затраты на ее разработку и поддержку и тем сложнее тестовая система. Таким образом, выбор типа модели – одна из ключевых задач автономной верификации, от правильного решения которой зависит эффективность верификации в целом.

3. Автономная верификация на основе событийных моделей

Несмотря на существенные различия в архитектуре и функциональности модулей, входящих в состав микропроцессоров, с точки зрения динамической верификации их можно разделить на два основных класса: *операционные модули*, обладающие конвейерной архитектурой, и *модули системного обмена* [6]. Наличие конвейера определяет жесткие ограничения на время выполнения операций, что значительно усложняет тестовую систему и накладывает ряд ограничений на эталонную модель. Для проверки конвейеризированных модулей зачастую необходима разработка эталонной модели с потактовой точностью.

Отличительными особенностями модулей системного обмена являются отсутствие в них конвейера и жестких временны́х ограничений на исполнение операций, а также маркированность транзакций (однозначность отображения множества воздействий во множество реакций). Благодаря этому при автономной динамической верификации таких моду-

лей возможны переход на более высокий уровень абстракции (уровень транзакций) и применение высокоуровневых событийных моделей без потери качества верификации. Для организации взаимодействия тестовой системы, верифицируемой RTL-модели и эталонной модели в таком случае предлагается применять сериализаторы входных воздействий и десериализаторы реакций. Сериализацией называется отображение множества воздействий, представленных на уровне транзакций, во множество воздействий, представленных на битовом уровне в виде последовательности изменений интерфейсных сигналов RTL-модели. Десериализация, в свою очередь, является обратным сериализации отображением.

На рис. 1 представлена схема тестовой системы и тестового окружения для автономной верификации на уровне транзакций. Тестовая система включает в себя:

- генератор входных воздействий (GENERATOR), содержащий генератор высокоуровневых тестовых последовательностей (TRANSACTOR) и сериализаторы базовых транзакций;
- устройство проверки (CHECKER), в состав которого входят десериализаторы воздействий, поданных на верифицируемую RTL-модель (DUV), и реакций, полученных от RTL-модели;
- компаратор реакций (COMP);
- адаптер эталонной модели (ADAPTER), преобразующий транзакции в формат, принятый в эталонной событийной пассивной модели (CMODEL).

Функции получения модельных реакций вызываются на каждом такте и, если в модели есть готовые реакции, они сохраняются в буферах модельных реакций компаратора в том порядке, в котором были получены от модели. При получении реакции от верифицируемой RTL-модели в компараторе реакций запускается функция поиска модельной реакции. Поиск может быть проведен как с учетом очередности выдачи реакций, так и без него, что дает возможность проверять модули системного обмена с очередным и внеочеред-

ных исполнением операций соответственно.

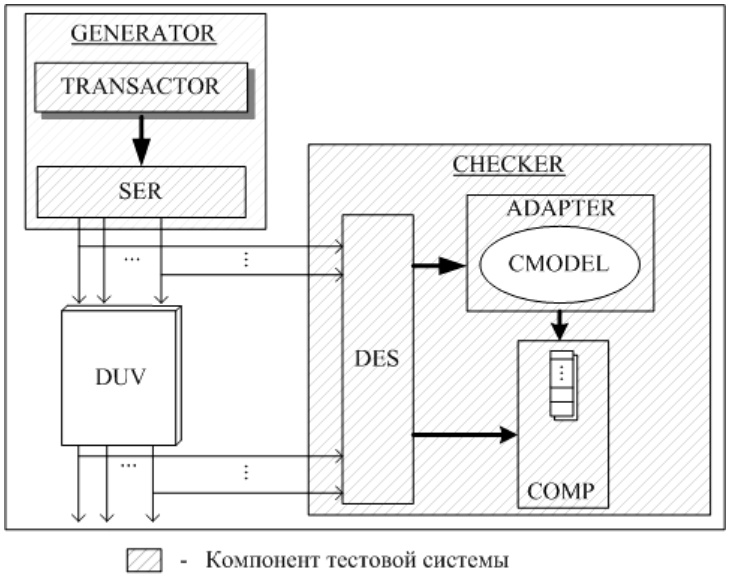

Рис. 1

Тестовая система для автономной верификации на уровне транзакций

4. Опыт практического применения

Описанный выше подход был применен при автономной верификации хост-контроллера микропроцессора «Эльбрус-8С». Хост-контроллер (НС) входит в состав северного моста микропроцессора и обеспечивает взаимодействие ядер с каналом ввода-вывода, а также со встроенным программируемым контроллером прерываний (APIC) и блоком конфигурационных регистров северного моста. В состав хост-контроллера входят: контроллеры битовой защиты (BSC); контроллер, осуществляющий трансляцию виртуальных адресов в физические (IOMMU); DMA-кэш, обеспечивающий упорядоченность выполнения DMA-операций в системе [7].

В результате анализа спецификации было установлено, что хост-контроллер является модулем системного обмена с неупорядоченным исполнением операций. Таким образом, для его автономной верификации была разработана событийная модель и тестовая система с архитектурой, описанной в разделе 3.

4.1. Особенности реализации эталонной модели хост-контроллера

Согласно спецификации в состав хост-контроллера входят:

- IORE (IO Requests Executor) – устройство выполнения процессорных запросов;
- URCE (Upward Requests and Completions Executor) – устройство обработки DMA-запросов, ответов на процессорные запросы, snoop-запросов;
- APICME (APIC Messages Executor) – устройство передачи прерываний;
- Coherence analyzer – устройство обработки snoop-запросов;
- IOMMU – контроллер трансляции адресов DMA-запросов;
- BSC (Bit Scale Controller) – контроллер битовой шкалы;
- WLCC_in_buf – приемный буфер входных транзакций из южного моста.

Для поддержки функционального соответствия было принято решение о реализации аналогичных модулей в программной модели хост-контроллера. Общая архитектура модели представлена на рис. 2. Все модули унаследованы от абстрактного класса (AbstractDevice), который представляет удобные средства вывода служебной информации. Такая архитектура обеспечивает гибкость разработки и позволяет быстро локализовать ошибки. Ввиду того, что модель предназначена для автономного применения, она реализована как пассивная событийная модель, реакции которой выдаются при вызове ее интерфейсных функций из адаптера модели тестовой системы.

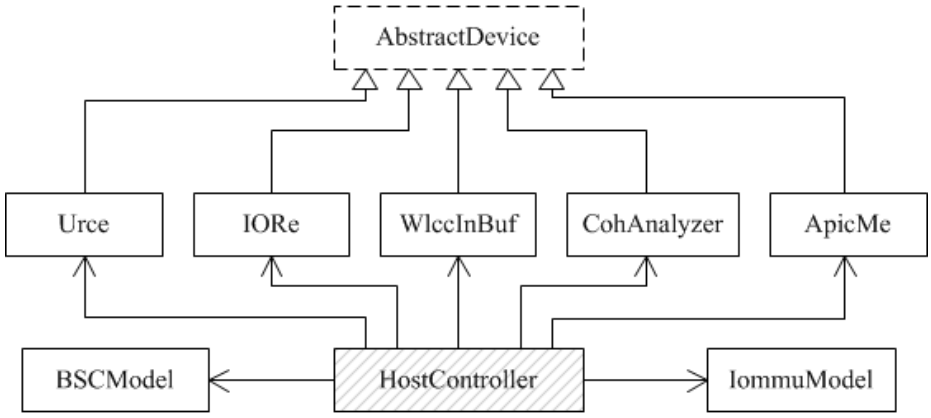

Рис. 2

Диаграмма классов программной модели хост-контроллера

Модели IOMMU и BSC, которые были ранее реализованы и отлажены в программной модели предшествующего процессора «Эльбрус-2S», интегрированы в полную модель хост-контроллера. В отличие от нее они являются активными событийными моделями, способными инициировать операции чтения данных из памяти и выдачу реакций. Для организации взаимодействия модели хост-контроллера и моделей IOMMU и BSC был разработан промежуточный класс, позволяющий скрыть реализацию подключаемых модулей и унифицировать интерфейс взаимодействия модулей внутри модели.

Для обеспечения гибкости разработки и отладки, а также для поддержки совместимости разрабатываемой модели с программной моделью процессора «Эльбрус-8С» были разработаны новые и использованы существующие средства:

- программная библиотека SimTracer, предоставляющая инструменты трассировки;
- библиотека MaskedValue, дающая возможность эмулировать регистровую организацию данных;
- средства описания базовых элементов (буферов, очередей, компараторов);
- инструменты межмодульного взаимодействия.

4.2. Особенности реализации тестовой системы

Тестовая система для автономной верификации хост-контроллера реализована на основе методологии UVM, поддерживаемой всеми применяемыми в ЗАО «МЦСТ» симуляторами (ModelSim, VCS) [8]. С использованием компонентов библиотеки UVM построены: сериализаторы воздействий (uvm_driver), десериализаторы воздействий и реакций (uvm_monitor), устройство проверки (uvm_scoreboard), генератор высокоуровневых воздействий (uvm_virtual_sequencer).

Для организации взаимодействия активных событийных моделей IOMMU и BSC, входящих в состав эталонной модели хост-контроллера, был реализован обратный интерфейс между моделями и адаптером модели тестовой системы (callback). Данные из опера-

тивной памяти, получаемые IOMMU и BSC, сохраняются в ассоциативной памяти устройства проверки. Таким образом, при DMA-запросах от RTL-модели генератор воздействий анализирует ассоциативную память устройства проверки и при наличии в ней данных направляет их в качестве ответа RTL-модели. Такой подход позволил адаптировать модели IOMMU и BSC, не внося в них изменений.

Наряду со сравнением реакций, получаемых от RTL-модели, с модельными, в устройстве проверки тестовой системы также были предусмотрены дополнительные механизмы:

- проверка корректности данных с помощью утверждений на отсутствие неопределенных значений в интерфейсах RTL-модели;
- проверка завершенности операций (сравнение числа отправленных запросов и полученных ответов и сообщений об освобождении позиций в очередях);
- проверка завершенности тестовых последовательностей.

Заключение

Описанный в статье подход к применению событийных программных моделей в качестве эталонных дает возможность существенно сократить сроки автономной верификации модулей микропроцессоров, обладающих неконвейеризированной архитектурой, не ухудшая при этом результаты и достоверность верификации. В статье предложены способы организации взаимодействия тестовых систем и событийных моделей, дающие возможность использовать модели различных типов и адаптировать существующие компоненты программных моделей микропроцессоров без внесения в них изменений.

Разработанные подходы были успешно применены при автономной верификации хост-контроллера микропроцессора «Эльбрус-8С», в результате которой была разработана и отлажена высокоуровневая событийная модель хост-контроллера, выявлено и исправлено 73 ошибки в RTL-модели устройства и 11 ошибок в спецификации, что по приближен-

ным оценкам составляет более половины от общего числа обнаруженных в хост-контроллере ошибок. Однако в процессе работы был выявлен ряд проблем, связанных с невозможностью полноценного моделирования DMA-кэша и внутренних очередей запросов и данных из-за зависимости их состояний от внутренних изменений в хост-контроллере и динамических изменений, не зависящих от внешних воздействий. Решение данных проблем определяет направление дальнейших работ.

Литература

1. Bergeron J. Writing testbenches: functional verification of HDL models. – Boston: Kluwer Academic Publishers, 2003.

2. Lam W. Hardware design verification: simulation and formal method-based approaches. – New Jersey: Prentice Hall, 2005.

3. Камкин А.С., Чупилко М.М. Механизмы поддержки функционального тестирования моделей аппаратуры на разных уровнях абстракции – Труды Института системного программирования РАН, 2011, т. 20, с. 143-160.

4. Кельтон В., Лоу А. Имитационное моделирование. Классика CS. 3-е изд. – СПб.: Питер, 2004.

5. Мешков А.Н. Методы и средства программного моделирования для обеспечения процесса проектирования микропроцессорных систем: Автореф. дис. канд. тех. наук. Москва, 2013. 22 с.

6. Стотланд И.А. Метод динамической верификации модулей системного обмена микропроцессорных вычислительных комплексов – «Научно-технический вестник Поволжья», 2012, № 4, с. 191-196.

7. Перов Д.Ю., Поляков Н.Ю. Обеспечение упорядоченности выполнения DMA-операций в NUMA-системах методом предварительного кэширования – «Вопросы радиоэлектроники», сер. ЭВТ, 2013. вып. 3, с. 38-47.

8. Universal Verification Methodology [Электронный ресурс] // Accellera. URL: http://www.accellera.org/community/uvm/ (дата обращения 30.11.13).