Для цитирования: Лебедев Д. А., Стотланд И. А. Автономная верификация контроллеров сопряжения интерфейсов на основе эталонных функциональных моделей // Вопросы радиоэлектроники. 2018. № 2. С. 81–86.
УДК 004.052.4

Д. А. Лебедев[1], И. А. Стотланд[1]

[1] АО «МЦСТ»

АВТОНОМНАЯ ВЕРИФИКАЦИЯ КОНТРОЛЛЕРОВ СОПРЯЖЕНИЯ ИНТЕРФЕЙСОВ НА ОСНОВЕ ЭТАЛОННЫХ ФУНКЦИОНАЛЬНЫХ МОДЕЛЕЙ

Рассмотрены вопросы автономной верификации контроллеров сопряжения интерфейсов, участвующих в передаче и преобразовании потока данных в современных микропроцессорных системах. Для корректного функционирования системы такие преобразования должны проходить быстро и без потерь. Точность функционирования контроллера подтверждается при проведении верификации. Приведена классификация методов верификации модулей микропроцессорных систем. Описан подход к построению автономного тестового окружения для контроллеров сопряжения интерфейсов при помощи методологии UVM. Обоснован выбор проверяющего модуля тестового окружения. Рассмотрены основные особенности автономной верификации аппаратных контроллеров сопряжения с использованием функциональных эталонных моделей, позволяющих проводить комплексную проверку этих устройств. Описаны сложности, возникшие в процессе разработки тестовой системы на основе методологии UVM, и способы их разрешения. Приведены результаты применения таких решений при верификации контроллеров микропроцессоров и дальнейший план наращивания тестовой системы.

Ключевые слова: *автономная верификация, тестовая система, контроллер сопряжения интерфейсов, микропроцессор «Эльбрус», UVM.*

Введение

В состав современных микропроцессоров входят различные аппаратные контроллеры, количество типов, сложность, скорость и объем передаваемых данных которых со временем непрерывно увеличиваются. При этом растут затраты на верификацию, т.к. возможности ее методов существенно отстают от развития микропроцессоров и, соответственно, проверка корректности требует все бóльших ресурсов [1].

Каждый из контроллеров периферийных устройств может иметь свой формат представления данных. Устройства, преобразующие компьютерные данные из одного формата в другой, называют контроллерами сопряжения интерфейсов (КСИ). Эти преобразования должны проходить быстро и без потерь данных. Поэтому верификация контроллеров сопряжения интерфейсов является ответственным этапом разработки микропроцессора в целом.

В статье рассмотрены подходы к построению проверяющих модулей при автономной верификации контроллеров сопряжения интерфейсов на примере Host-Bridge (HB) контроллера микропроцессора «МЦСТ R2000», разрабатываемого в АО «МЦСТ».

Функциональное тестирование контроллера сопряжения интерфейсов

При автономной верификации контроллера необходимо имитировать работу всего его окружения. Для этого необходима тестовая система, разработку которой лучше начать на самых ранних стадиях проектирования микропроцессора, как только станут доступными спецификация модуля и его RTL-модели. Это позволит обнаруживать ошибки на ранних стадиях, а также создавать труднореализуемые, критические и некорректные для модуля ситуации, формирование которых при системной верификации в составе модели всего микропроцессора требует много ресурсов. При этом также важно, что локализация ошибки происходит быстрее, позволяя тем самым уменьшить сроки отладки контроллера.

В силу своего расположения на стыке процессора и контроллера периферийных интерфейсов КСИ, помимо исполнения своей основной функции преобразования формата данных, может содержать копии регистров, буферы, очереди FIFO, части распределенных управляющих систем, а также выполнять другие дополнительные функции. При автономной верификации КСИ необходимо учитывать ряд этих особенностей.

Существенно, что согласно предложенной в [2] классификации к свойствам КСИ относят отсутствие конвейера, жестких временны́х (в тактах системной частоты) ограничений на обработку транзакций и теговую маркировку данных. В связи с этим при верификации таких устройств возможно применение событийных модулей проверки.

Для построения тестового окружения и проведения автономной проверки устройств микропроцессора существует ряд методов, в том числе созданный в АО «МЦСТ» инструмент Alone-env, разработка ИСП РАН под названием C++TESKHW и методология UVM [3]. UVM является широко распространенной универсальной методологией верификации (Universal Verification Methodology), разработанной компанией Accellera Systems [4]. UVM представляет собой библиотеку с хорошо описанными средствами построения переносимых и повторно применяемых тестовых систем и их компонентов. Тестовая система, построенная на основе UVM, может генерировать псевдослучайные ограниченные входные запросы для охвата всех возможных состояний проверяемого устройства.

Принципы реализации функционального тестирования КСИ

Автономная верификация КСИ может проводиться с использованием имитационных эталонных моделей, входящих в состав тестовых систем (или testbenches) – программного окружения, специально реализованного относительно верифицируемого устройства. К его функциям относятся:

- генерация входных воздействий;
- отслеживание реакций от верифицируемого устройства и эталонной модели;
- сравнение реакций;
- формирование вывода о полноте тестирования.

Для генерации псевдослучайных ограниченных воздействий определяются классы-расширения uvm_sequence_item и uvm_sequence. В первом задается набор переменных, впоследствии необходимый для сериализации множества воздействий в последовательный битовый формат. Во втором производится единичная или множественная генерация набора переменных для передачи запроса. Сгенерированный объектом item запрос обрабатывается специальным классом uvm_sequencer и передается в класс uvm_driver. Драйвер производит преобразование сгенерированных случайных запросов в последовательные бит-векторы в соответствии с принятым протоколом обмена. Класс uvm_monitor – пассивный; монитор отслеживает изменения в интерфейсе верифицируемого устройства, свидетельствующие о поступлении данных, упаковывает последовательные битовые сигналы в формат, принятый в uvm_sequence_item, и передает для дальнейшего анализа в проверяющий блок. Для упрощения восприятия структуры тестового окружения драйвер, монитор и секвенсор объединяются в классе uvm_agent (рис. 1).

Проверку реакций верифицируемого устройства можно осуществить внутренними средствами библиотеки, однако, если проверяемое устройство имеет сложную структуру и множество состояний, проверяющий модуль строится на основе внешней по отношению к окружению контроллера эталонной модели верифицируемого устройства на языке C++. Обобщенная тестовая структура для проверки устройств типа КСИ, сопряженная с эталонной моделью при помощи интерфейса DPI SystemVerilog, представлена на рис. 2. Здесь DUV (Design Under Verification) – RTL-модель тестируемого устройства, ENV (Environment) – тестовое окружение, число агентов определяется количеством интерфейсов верифицируемого устройства (отслеживающий реакции объект monitor может быть вынесен из агентов). Эталонная модель при подаче воздействий от тестового окружения генерирует эталонные реакции. Модуль проверки – uvm_scoreboard – сравнивает реакции от верифицируемого устройства и эталонной модели и делает вывод о корректности работы. Использование интерфейса DPI необходимо для согласования типов и классов языка описания тестовой системы SystemVerilog с языком C++, на котором разрабатывается эталонная модель.

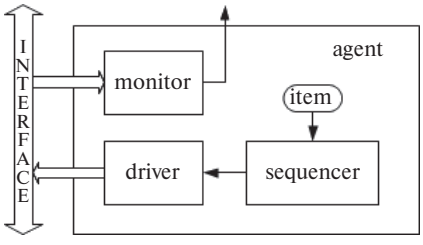

Рисунок 1. Пример структуры агента для интерфейса верифицируемого устройства

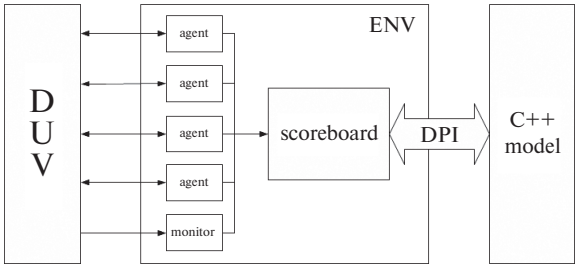

Рисунок 2. Обобщенная тестовая структура для проверки устройств типа КСИ

Эталонные модели можно разделить на три типа: потактовые, дискретно-событийные с учетом времени и событийные [5]. Выбор типа модели зависит от типа верифицируемого устройства, его архитектуры и сложности в разработке тестового окружения. Как указано ранее, для устройств типа КСИ обосновано использование событийных моделей, т.к. они требуют меньше времени на разработку и поддержание изменений и могут полностью имитировать работу такого контроллера.

Функциональная верификация контроллера Host Bridge

Контроллер сопряжения интерфейсов Host Bridge (HB), расположенный в северном мосте, связывает систему с внешними устройствами, принимая запросы от системы и пространства ввода-вывода и поддерживая при этом принятые в системе и пространстве ввода-вывода форматы транзакций. Запросы из системы поступают в HB из системного коммутатора, связь с пространством ввода-вывода осуществляют два контроллера каналов ввода-вывода (IO-линка). Также через контроллер происходят доступ к системным регистрам, регистрам межпроцессорных линков и контроллеров памяти, передача сообщений о прерываниях и сбор snoop-ответов. Регистры каждого типа имеют собственный интерфейс для связи с устройством назначения. При верификации HB нужно учитывать все эти особенности.

В процессе автономной верификации был реализован ряд рассмотренных ниже решений.

Рандомизация параметров нескольких синхросигналов

Основной задачей КСИ является согласование запросов и данных нескольких устройств микропроцессорной системы, работающих на разных частотах синхросигнала. Части КСИ, в которых происходит взаимодействие нескольких синхросигналов, должны быть проверены с особенной тщательностью. Для этого используется генерация случайных периодов синхросигналов и их сдвигов друг относительно друга. В спецификации контроллера определяются диапазоны работы каждого из синхросигналов, а в начале каждого теста предварительно рассчитываются частоты и время старта случайных генераторов синхросигналов (СГС), в результате чего становится возможным обнаружение ошибок в синхронизации внутренних блоков: определение неопределенных состояний интерфейсов, несогласованная выдача запросов и данных, невыдача данных из некоторых позиций буферов.

Поддержка кредитных механизмов обмена

Для управления потоком запросов и контролем свободных позиций в буферах транзакций контроллера HB используется кредитный механизм, который представляет собой передачу однотактного сигнала, информирующего о наличии свободного места в буферах соединенных устройств, в другие соединенные с КСИ устройства. Управление этим механизмом позволяет создавать тестовые сценарии с полным заполнением всех позиций в буферах устройства и необходимостью ожидания освобождающегося места для обработки новых запросов или, наоборот, такие сценарии, когда освобождение позиций происходит очень быстро, и запросы выполняются почти мгновенно. В результате возможно создание тестовых сценариев, которые трудно ввести в системные тесты.

Верификация контроллера трансляции адресов

Одной из составляющих HB является контроллер трансляции адресов – IOMMU. Он обеспечивает преобразование виртуальных адресов запросов, поступающих из подсистемы ввода-вывода, в физические адреса. Контроллер посылает в специальную область памяти запрос информации о физическом адресе для проведения трансляции. Информация о соответствии виртуальных адресов физическим хранится в специальном буфере контроллера – IOTLB (Input-Output Translation Lookaside Buffer). Если буфер заполнен, то производится замещение самого старого элемента. Алгоритм проведения трансляции можно представить в виде последовательных шагов:

1. Поступление DMA-запроса.
2. Анализ входящего адреса, поиск соответствия в кэше IOMMU со следующими возможными вариантами:
   - соответствие найдено (попадание IOMMU) – производятся преобразование адреса и выдача запроса с оттранслированным адресом;
   - соответствие не найдено (промах IOMMU) – производятся выдача запроса информации о физическом адресе, получение ответа с данными и только после этого – преобразование адреса и выдача запроса с оттранслированным адресом.

В условиях динамических тестов возможны ситуации, когда в RTL-модели строка в кэше IOMMU еще не вытеснена и допустима трансляция адреса без проведения запроса в память, а в эталонной модели эта строка отсутствует. В таком случае эталонная модель будет выдавать лишние запросы в тестовую систему. Для выхода из этой ситуации введен дополнительный модельный глобальный счетчик транзакций, необходимый для идентификации исходных запросов. Кроме того, проводится анализ ответов, сгенерированных тестовой системой. В случае, когда запрос успешно оттранслирован на стороне RTL-модели, а эталонная модель

уже выдала лишний запрос информации о физическом адресе, тестовое окружение генерирует ответ, который при передаче в эталонную модель помечается специальным идентификатором. Модель, обрабатывая этот ответ, делает вывод о том, что трансляция не производилась, вычисляет нужный номер транзакции и отправляет его в специальный буфер отмененных запросов информации о физическом адресе. При проведении проверки буферов обмена с моделью после завершения исполнения тестового сценария значения идентификаторов в буфере отмененных запросов сравниваются с идентификаторами оставшихся необработанными запросов информации о физическом адресе от эталонной модели. Такие оставшиеся запросы не рассматриваются как ошибочные и удаляются. Псевдокод алгоритма представлен ниже.

```
while true do
    wait p ← start(x) // ожидание DMA запроса x, старт процесса p обработки трансляций
    if x' then // если RTL выдает запрос за поиском страницы физ. адреса x'
        p.ncheck(x') // стандартная обработка запроса
    else if x" then // если RTL выдает оттранслированный запрос x"
    begin
        wait p.receive(y) // ожидаем ответы y от системы, помечаем их id и посылаем в модель
        p.model_check(y) // проверка типа ответа и извлечение id
        Q_id ← y.id // сохраняем id отмененного запроса о физ. адресе в буфере
        p.finish() // окончание обработки
    end
end
for i∈Qred do // для всех оставшихся элементов в буфере запросов инф. о физ. адр.
    if c.check(req.id, Qid) then // произвести проверку соответствия id с элементами очереди id
        delete(req.id) // удалить запрос, если соответствие найдено
    else report(req.id) // сообщить об ошибке, если нет соответствия
end
```

Корректная организация порядка обмена в условиях неизвестной очередности выдачи запросов

В условиях высоконагруженных динамических тестов с множеством входных запросов и ответов маркировка этих запросов и ответов тэгами, соответствующими позиции в буфере, а также выбор в событийной модели проверяющего устройства приводят к расхождению значений тэгов в модели и верифицируемом устройстве. Это происходит из-за невозможности спрогнозировать время освобождения позиции буфера в моделях событийного типа, поэтому важно определить соответствия входных и выходных запросов. Каждый запрос, будь то запрос из области ввода-вывода или процессорный запрос, имеет несколько стадий исполнения.

Для обеспечения корректного функционирования тестовых сценариев необходимо использовать ассоциативные памяти соответствия запросов, или мапперы (mappers). Функция маппера состоит в сохранении соответствия маркировочных тэгов запросов RTL и эталонной модели при условии сравнения остальных полей данных запроса, таких как: адрес, код устройства назначения, номер процессора и других. В дальнейшем, при получении ответа на запрос, необходимо передать в модель тот же тэг, который был выделен моделью на этапе формирования запроса.

КСИ в многоядерных системах могут участвовать в протоколе поддержки когерентности и принимать snoop-ответы. В зависимости от режима работы и содержания полей первого полученного ответа на один запрос может прийти как один, так и несколько ответов. Такой механизм имитирует работу подсистемы поддержки когерентности памяти. Для корректного завершения проверки запроса в когерентном режиме необходимо также передавать в модель выделенные ею тэги.

Проверки после завершения тестового сценария

Корректное поведение устройства КСИ заключается в выдаче определенного количества реакций на запросы и ожидании соответствующего количества ответов на них. Факт нарушения его правильной работы можно установить путем подсчета количества пришедших запросов, соответствующих им запросов в систему, преобразованных в другой формат, и принятых ответов. Для этого в тестовой системе предусматриваются счетчики транзакций, которые регистрируют все виды транзакций во время работы тестового сценария.

После завершения тестового сценария необходимо также проверить отсутствие не получивших ответа запросов в буферах, связывающих эталонную модель и тестовое окружение. Наличие таких запросов сигнализирует об ошибке либо верифицируемого устройства, либо эталонной модели.

Опыт применения

Описанные выше методы нашли применение при автономной верификации устройства Host Bridge микропроцессора «МЦСТ R2000». С помощью НВ реализуются:

- взаимодействие процессорных ядер с двумя каналами ввода-вывода;

- поддержка формата сообщений в каналах ввода-вывода и на системном уровне;
- трансляция виртуальных адресов в физические;
- передача сообщений между частями распределенной системы прерываний;
- организация доступа к системным регистрам, регистрам межпроцессорных линков и контроллеров памяти;
- передача новых значений регистров при изменениях в локальных копиях регистров по выделенным интерфейсам;
- передача и генерация асинхронных ошибок и статусных сигналов.

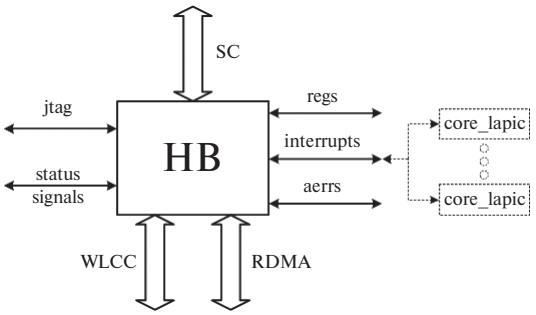

Рисунок 3. Обобщенная структура Host Bridge с подключением контроллеров прерываний

Для контроллера, в силу его функциональных и структурных особенностей относящегося к классу КСИ, разработано тестовое окружение с проверяющим устройством на основе пассивной событийной модели с некоторыми активными функциями, необходимыми для корректной работы механизма трансляции адресов. При автономной верификации устройства было найдено и исправлено 67 ошибок, что свидетельствует о ее эффективности.

Дальнейшее развитие разработки тестового окружения возможно при изменении сборки проекта и подключении части системы прерываний, передающей сигналы непосредственно в ядра. На рис. 3, представляющем обобщенную структуру Host Bridge, пунктиром приведена схема такого наращивания.

Заключение

Контроллеры сопряжения интерфейсов относятся к числу важных элементов многоядерных микропроцессорных систем, подлежащих тщательной проверке. Описанные в статье принципы в основном не зависят от реализации этих контроллеров и позволяют проводить их всестороннюю автономную верификацию. В статье предложены способы организации взаимодействия тестовой системы и событийной эталонной модели при построении модулей проверки, а также способы разрешения трудностей, возникших при разработке тестовой системы.

Предложенные подходы были применены при верификации контроллера сопряжения интерфейсов Host Bridge в составе восьмиядерного микропроцессора, разрабатываемого в АО «МЦСТ». Разработанные тестовые системы и сценарии позволили обнаружить и исправить ряд логических ошибок, которые не были выявлены с помощью других методов верификации. Дальнейшая работа предполагает подключение локальных контроллеров прерываний, являющихся частью ядра, и проверку объединенной системы прерываний, связанной с контроллером сопряжения интерфейсов.

СПИСОК ЛИТЕРАТУРЫ

1. Средства функциональной верификации микропроцессоров / А.С. Камкин, А.М. Коцыняк, С.А. Смолов, А.А. Сортов, А.Д. Татарников, М.М. Чупилко // Труды ИСП РАН. 2014. Т. 26. Вып. 1. С. 149–200.
2. Шмелев В.А., Стотланд И.А. Автономная верификация микропроцессоров на основе эталонных моделей разного уровня абстракции // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2012. № 1. С. 435–440.
3. Мешков А.Н., Рыжов М.П., Шмелев В.А. Развитие средств верификации микропроцессора «Эльбрус-2S» // Вопросы радиоэлектроники. 2014. Т. 4. № 3. С. 5–17.
4. Standard Universal Verification Methodology. Available at: http://accellera.org/downloads/standards/uvm (accessed 08.10.2017).
5. Кельтон В., Лоу А. Имитационное моделирование: 3-е изд. СПб.: Питер, 2004. 847 с.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Лебедев Дмитрий Алексеевич, инженер 2-й категории, АО «МЦСТ», 119334, Москва, ул. Вавилова, д.24, тел.: 8 (499) 135-44-61, e-mail: lebedev_d@mcst.ru.
Стотланд Ирина Аркадьевна, к.т.н., начальник сектора, АО «МЦСТ», 119334, Москва, ул. Вавилова, д.24, тел.: 8 (499) 135-44-61, e-mail: stotl_i@mcst.ru.

For citation: Lebedev D.A., Stotland I.A. Standalone verification of communication controllers based on reference functional models. Voprosy radioelektroniki, 2018, no. 2, pp. 81–86.

D.A. Lebedev, I.A. Stotland

STANDALONE VERIFICATION OF COMMUNICATION CONTROLLERS BASED ON REFERENCE FUNCTIONAL MODELS

Issues of standalone verification of communication controllers involved in the transfer and transformation of data flow in state of the art microprocessor systems are considered in the article. For the correct functioning of the system such transformations should take place quickly and without losses. The accuracy of the controller functioning is confirmed during the verification. The classification of verification methods of microprocessor systems modules is presented in the paper. The approach of building a standalone test environment for communication controllers with the UVM methodology is described. The choice of the checking module of verification environment is justified. The basic features of the standalone verification of hardware communication controllers with using reference functional models, enabling comprehensive testing of these devices are considered. The difficulties discovered in the process of developing a test system based on the UVM methodology and their resolutions are described. The results of using such solutions for verification of microprocessor controllers and further plan of the test system enhancement are considered.

Keywords: standalone hardware verification, test system, multi-core microprocessor, Elbrus microprocessor, UVM.

REFERENCES

1. Kamkin A.S., Kotsynyak A.M., Smolov S.A., Sortov A.A., Tatarnikov A.D., Chupilko M.M. Tools for functional verification of microprocessors. *Proceedings of ISP RAS,* 2014, vol. 26, no. 1, pp. 149–200 (In Russian).
2. Shmelev V.A., Stotland I.A. Standalone verification of microprocessors using reference models with various levels of abstraction. *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem (MES),* 2012, no. 1, pp. 435–440 (In Russian).
3. Meshkov A.N., Ryzhov M.P., Shmelev V.A. The development of the verification tools of the Elbrus-2S microprocessor. *Voprosy radioelektroniki,* 2014, no. 3, pp. 5–17 (In Russian).
4. Standard Universal Verification Methodology. Available at: http://accellera.org/downloads/standards/uvm (accessed 08.10.2017)
5. Kelton W., Law A. *Imitatsionnoe modelirovanie* [Simulation modeling]. 3-e izd. Saint-Petersburg, Piter Publ., 2004, 847 p. (In Russian).

AUTHORS

Lebedev Dmitriy, engineer 2nd category, JSC MCST, 24, ulitsa Vavilova, Moscow, 119334, Russian Federation, tel.: +7 (499) 135-44-61, e-mail: lebedev_d@mcst.ru.
Stotland Irina, PhD, head of Sector, JSC MCST, 24, ulitsa Vavilova, Moscow, 119334, Russian Federation, tel.: +7 (499) 135-44-61, e-mail: stotl_i@mcst.ru.