В.Н. Куцевол (ЗАО «МЦСТ»)**, к.т.н. А.Н. Мешков, М.П. Рыжов,**
П.В. Фролов (ЗАО «МЦСТ», ПАО «ИНЭУМ им. И.С. Брука)

V. Kutsevol, A. Meshkov, M. Ryzhov, P. Frolov

МЕТОД ВЕРИФИКАЦИИ ПОДСИСТЕМЫ ПРЯМОГО ДОСТУПА ПЕРИФЕРИЙНЫХ УСТРОЙСТВ К ОПЕРАТИВНОЙ ПАМЯТИ МИКРОПРОЦЕССОРОВ СЕМЕЙСТВА «ЭЛЬБРУС»

A METHOD OF DIRECT MEMORY ACCESS SUBSYSTEM VERIFICATION FOR ELBRUS SERIES MICROPROCESSORS

Описан метод верификации подсистемы прямого доступа к памяти, используемый при разработке микропроцессоров семейства «Эльбрус». Разработан имитатор контроллера периферийных устройств, позволяющий снизить накладные расходы при верификации. Модель имитатора включена в состав функциональной модели вычислительного комплекса. Реализован опирающийся на функциональную модель генератор псевдослучайных тестов, направленных на верификацию подсистемы прямого доступа к памяти.

A method of direct memory access subsystem verification used for Elbrus series microprocessors has been described. A peripheral controller imitator has been developed in order to reduce verification overhead. The model of imitator has been included into the functional machine simulator. A pseudorandom test generator for verification of the direct memory access subsystem has been based on the simulator.

Ключевые слова: системная верификации, функциональная модель, прямой доступ к памяти, генератор псевдослучайных тестов.

Keywords: system verification, functional model, direct memory access, pseudorandom test generation.

Введение

Современные вычислительные системы характеризуются высокой интенсивностью потоков данных между периферийными устройствами и оперативной памятью. В подавляющем большинстве случаев этот обмен осуществляется через подсистему прямого доступа к оперативной памяти. Возрастающие требования к производительности подсистемы приводят к ее усложнению и, соответственно, необходимости разработки эффективных подходов к верификации [1, 2].

Материал данной статьи сформирован по результатам комплексного проекта, в процессе которого были реализованы совместно используемые методы верификации, основанные на последовательной проверке подсистемы с использованием одной из трех моделей: 1) функциональной модели на языке C++, соответствующей работе подсистемы в окружении, определяемом реальной конфигурацией ВК; 2) RTL-модели на языке Verilog; 3) ПЛИС-прототипа. Статья характеризует первый метод, позволяющий в начальной стадии верификации оперативно приступить к проверке корректности разработки и получить значимые результаты с использованием простых тестов.

Важнейшей проблемой, во многом влияющей на качество тестирования подсистемы, является полнота представления связанных с ней периферийных устройств и генерируемых ими воздействий. В данном случае задача решалась введением в функциональную модель имитатора внешних устройств, позволяющего в широком диапазоне изменять нагрузку на тракт прямого доступа к памяти. Основные аспекты его реализации приведены в разделе 1.

Полнота верификации рассматриваемой подсистемы достигается с помощью генератора, позволяющего создавать необходимые воздействия с использованием имитатора. Результатом работы генератора является тестовая программа,

осуществляющая задание сценариев для имитатора, запуск его работы, сопровождение работы обращениями со стороны процессорных ядер, а также проверку корректности конечного состояния памяти. Принципу работы генератора посвящен раздел 3.

Генерация кода, проверяющего корректность конечного состояния оперативной памяти, вызывает необходимость интеграции эталонной модели подсистемы памяти в генератор. В этом качестве была использована ранее разработанная библиотека, которая реализует функциональную модель вычислительного комплекса (ВК), рассмотренную в разделе 2.

1. Имитатор внешних устройств

Рассматривая ВК, в составе которого функционирует исследуемая подсистема (рис. 1а), следует отметить, что при ее верификации можно в значительной мере избежать проблем, связанных с буквальным моделированием периферийных устройств южного моста и, как следствие, использованием драйверов сложных устройств ввода/вывода, путем имитации реальных операций обращения к оперативной памяти. В качестве базовой операции выбрано DMA-копирование с маской, позволяющее реализовать большинство актуальных сценариев прямого доступа. Для обеспечения высокой скорости исполнения тестов имитатор интегрируется в канал связи, соединяющий северный и южный мосты (рис. 1б). Расположение имитатора на позиции стандартного контроллера ввода-вывода позволяет применить модель к любой современной версии микропроцессора семейства «Эльбрус».

Имитатор представляет собой урезанную версию южного моста. Он включает параметризуемое число равноправных агентов (рис. 2), каждый из которых может работать в обычном и табличном режимах. Во втором режиме, существенно упрощающем

формирование различных сценариев доступа, параметры операций копирования задаются в расположенных в памяти таблицах.

Агент может выполнять следующие действия:

●копирование данных из одной области оперативной памяти в другую в обычном и табличном режимах работы;

●чтение из памяти таблиц с параметрами операций копирования;

●преобразование данных.

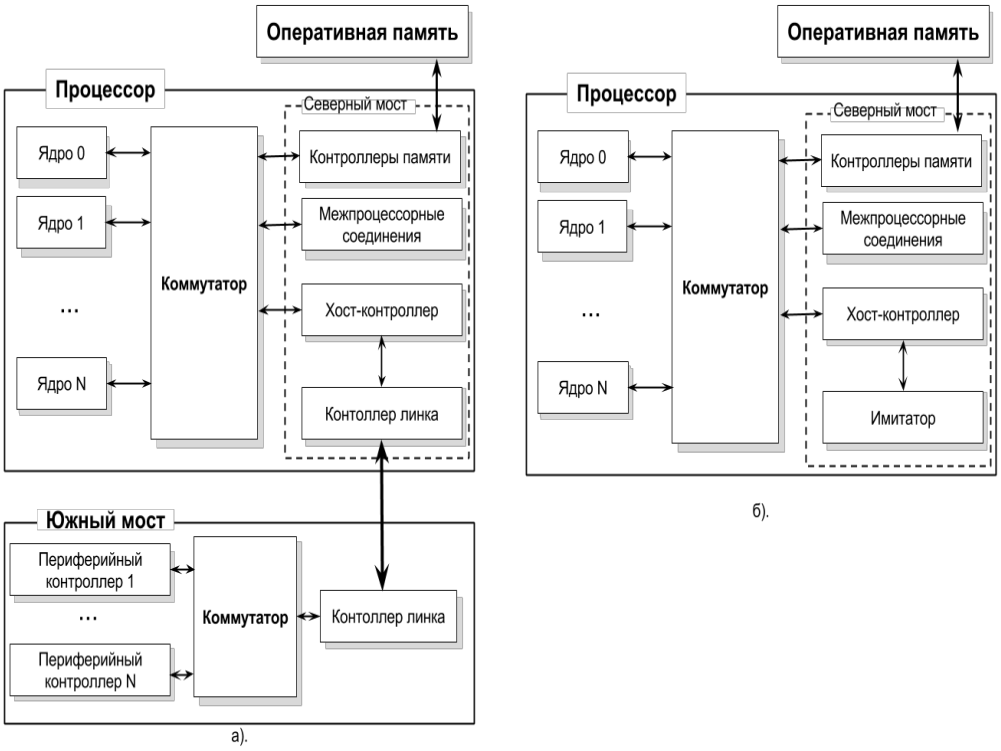

Рис. 1. Рассматриваемая структура вычислительного комплекса (а – реальная конфигурация, б – модельная конфигурация (интеграция имитатора в северный мост)

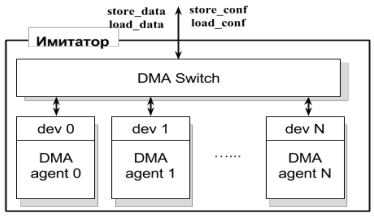

Рис. 2. Структурная схема DMA-имитатора

Имитатор реализован в соответствии со стандартом PCI, соответственно, каждый агент задается как независимое устройство (dev), управление которым осуществляется через общий канал операциями чтения и записи системных пакетов в конфигурационное пространство, а обмен с памятью – операциями чтения и записи пакетов данных. Коммутацию пакетов между агентами выполняет модуль DMA Switch.

Схема DMA-агента изображена на рис. 3. Здесь ConfigRegisters – блок регистров config-пространства, через который задаются режимы работы, базовые адреса и прочие параметры. В базовом режиме работы адреса, записанные в ConfigRegisters, используются для доступа в оперативную память, а в табличном режиме (TM) – для получения таблиц с адресами по чтениям и записям, которые обрабатывает модуль TMHandler. Модуль Format отвечает за наложение масок и корректное склеивание разрозненно считываемых данных в табличном режиме. Модуль DMA Engine, в основе которого лежит FIFO-буфер с данными, осуществляет чтение и запись данных по предоставляемым функциям DMA-чтения и DMA-записи.

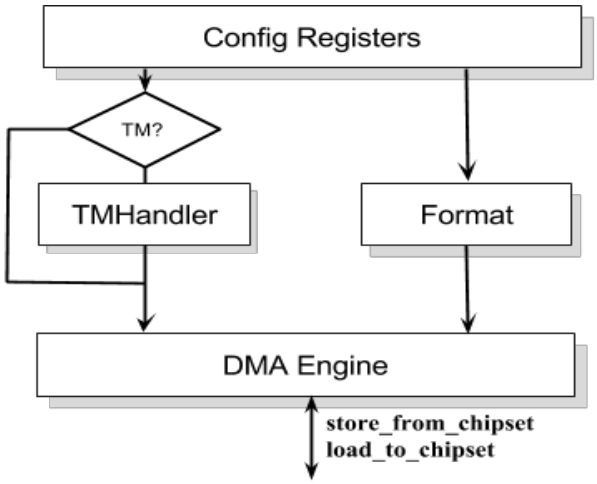

Рис. 3 DMA-агент

Рис. 3. DMA-агент

2. Функциональная модель подсистемы прямого доступа к памяти

Подход к проблеме основан на представлении процесса прямого доступа к оперативной памяти через взаимодействие двух самостоятельных программных модулей – симулятора, имитирующего работу непосредственно занятых в процессе объектов архитектуры ВК, и генератора, который, формируя режимы и параметры обращения к памяти, задает логику этих объектов и контролирует ее корректность (рис. 4). Структурная и функциональная автономность этих модулей существенно расширяет гибкость всей системы моделирования относительно состава и взаимосвязи исследуемых объектов, спектра генерируемых воздействий и контроля их результатов.

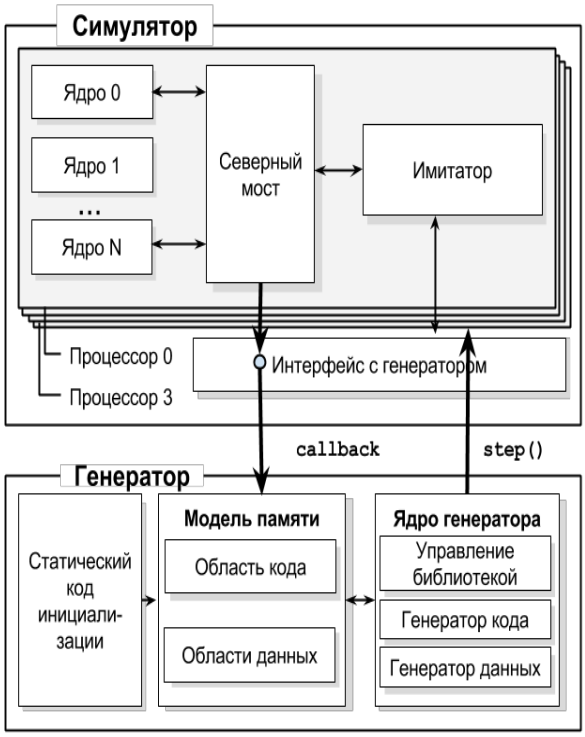

Рис.4. Объекты функционального моделирования подсистемы прямого доступа к памяти

В разработанной версии симулятора в конфигурацию ВК входят четыре процессора, каждый из которых содержит вычислительные ядра, северный мост и имитатор, представляющий набор периферийных устройств и их интерфейсов [3]. В соответствии с разделом 1 взаимодействие имитатора и северного моста осуществляется посредством функций программной модели, описанной в спецификации PCI.

Симулятор работает по принципу интерпретации. В каждом виртуальном такте происходит исполнение одной команды каждого процессорного ядра, кроме того, производятся различные асинхронные по отношению к исполнению команд действия, такие как переключение счетчиков и таймеров и обработка внешних прерываний [4]. Для того чтобы обеспечить взаимодействие симулятора с генератором, было принято решение реализовать работу симулятора в виде набора образующих его библиотечных процедур.

3. Генератор тестов

Генератор включает в себя статический код инициализации, модель памяти и ядро генератора. Код инициализации это последовательность инструкций, которая позволяет выполнить начальную настройку аппаратуры с помощью тестов.

Ядро генератора включает в себя модули управления библиотекой, а также генераторы кода и данных [5]. Модуль управления, отвечающий за взаимодействие с библиотекой, вызывает функцию step(), которая реализует исполнение инструкций моделируемой аппаратуры и анализ результата ее выполнения. Генератор кода создает код, задающий работу отдельных DMA-агентов, а генератор данных формирует блоки пересылаемых данных. Гибкость параметризации DMA-имитаторов полностью поддержана путем генерации случайных тестов, осуществляющих формирование случайных параметров DMA-обмена, таких как: адреса буферов передачи, диапазоны размеров DMA-пакетов, различные режимы передачи.

Статический код инициализации и динамически-генерируемый код помещаются в область кода, которая является одним из компонентов модели памяти. При обращениях за кодом в процессе выполнения моделируемой программы имитатором вызовы перенаправляются в виде callback-функции в область кода, расположенную в генераторе. Похожим образом происходит и работа с областью данных, являющейся другим компонентом модели памяти. Обращения за данными – чтения и записи могут в процессе работы инициировать как процессорные ядра, так и DMA-агенты. Все они перенаправляются в область данных, содержащую числовые массивы, динамически создаваемые генератором данных. Пошаговый алгоритм взаимодействия основных программных модулей функциональной модели приведен на рис. 5.

Общий сценарий работы с DMA-имитаторами выглядит следующим образом: инициализация областей памяти (DMA-буферов) данными, предназначенными к приёму/передаче; конфигурация DMA-имитаторов и запуск DMA-обмена. Инициализация областей данных осуществляется процессорными ядрами, поэтому на момент начала DMA-обмена предназначенные к передаче данные находятся в разных кэшах общей иерархии памяти [6]. При конфигурации имитатора производится задание режима его работы и адреса таблицы, описывающей подлежащие копированию области. Сам DMA-обмен также проходит на фоне обращений ядер в память, осуществляемых в том числе и по адресам, находящимся в буферах обмена. После завершения работы имитатора производится генерация кода сличения, принимающего конечное состояние данных в оперативной памяти за эталон.

Рис. 5. Поток управления при работе функциональной модели подсистемы прямого

доступа к памяти

Заключение

В работе применительно к микропроцессорам семейства «Эльбрус» рассмотрена проблема верификации подсистемы прямого доступа к памяти. Для обеспечения исполнения необходимого количества тестов, ускорения разработки генератора тестов и разбора ошибок предложен метод верификации, основанный на замене реальных устройств, поддерживающих прямой доступ в память, устройством-имитатором с простым программным интерфейсом и возможностью обеспечить полную загрузку тракта доступа в память. Применение разработанной методики позволяет обеспечивать режимы работы, аналогичные реальным сценариям прямого доступа в память. Использование унифицированного интерфейса имитатора в RTL-модели, симуляторе вычислительной машины и прототипе, основанном на ПЛИС, позволяет ускорить разработку генератора тестов, использующих имитатор.

Литература

1. Grosso, M. et al. Functional Verification of DMA Controllers – Journal of Electronic Testing: Theory and Applications Volume 27 Issue 4, August 2011, p. 505-516.

2. Ким А.К., Михайлов М.С., Фельдман В.М. Подсистема ввода-вывода для систем на кристалле «МЦСТ-4R» и «Эльбрус-S» на основе микросхемы контроллера периферийных интерфейсов. – «Вопросы радиоэлектроники», сер. ЭВТ, 2012, вып. 3, с. 52-62.

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, с. 62-70.

4. Nohl, A., Braun, G., Schkiebusch, O., Leupers, R., Meyr, H., A Universal Technique for Fast and Flexible Instruction-Set Architecture Simulation, DAC2002, June 10-14, New Orleans, Louisiana, USA, 2002.

5. Фролов П.В. Генерация случайных тестов системного уровня для микропроцессоров с архитектурой «Эльбрус». – «Вопросы радиоэлектроники», сер. ЭВТ, 2014, вып. 3, с. 38-46.

6. Исаев М.В., Поляков Н.Ю. Применение кэша и справочника DMA-обменов в NUMA-системах для повышения производительности подсистемы ввода-вывода: Первая всероссийская научно-техническая конференция «Расплетинские чтения»: Тез. докл. – Москва, 2013. С. 169-170.