В.Н. Куцевол, к.т.н. А.Н. Мешков, Е.М. Николаева, С.В.Черных
(ЗАО «МЦСТ»)

V. Kutsevol, A. Meshkov, E. Nikolaeva, S. Chernyh

ВЕРИФИКАЦИИ ГРАФИЧЕСКОЙ ПОДСИСТЕМЫ, ИНТЕГРИРОВАННОЙ В ПРОЦЕССОР

PROCESSOR-INTEGRATED GRAPHICS SYSBSYSTEM VERIFICATION

Рассматриваются задача верификации интегрированного графического контроллера и общий подход к её решению. Даётся краткое описание устройства. Описываются решения, применяемые при разработке программной модели и тестовой системы, которая сочетает элементы системной и автономной верификации. Приводятся предварительные результаты работы и пути дальнейшего развития.

The article describes methods of integrated graphics adapter verification. A brief description of such device is given. The solutions having been applied in the reference model and testbench are described. The testbench combines both system and standalone verification elements. The preliminary results and further possible improvements are given.

Ключевые слова: верификация графического адаптера, моделирование, функциональная верификация, тестирование на основе моделей.

Keywords: graphics adapter verification, hardware simulation, functional verification, model-based testing.

Ведение

Развитие микроэлектронной индустрии идет по пути интеграции различных подсистем на одном кристалле. Большое число используемых в настоящее время приложений включают графическое отображение, и, как следствие, одной из тенденций развития вычислительных систем является включение устройств, предназначенных для обработки

графики, в состав кристалла, содержащего основные вычислительные мощности [1]. Отличительной особенностью графических контроллеров являются высокие требования к пропускной способности подсистемы памяти, что может привести к значительным изменениям существующей архитектуры. Помимо этого, включение вышеупомянутых устройств в состав микропроцессора требует существенных доработок процесса верификации, который должен проверять не только корректность работы встроенного устройства, но и его быстродействие.

Система на кристалле «Эльбрус-1С+», разрабатываемая ЗАО «МЦСТ», включает в себя одно ядро общего назначения с архитектурой «Эльбрус» и интегрированный графический контроллер, содержащий устройства обработки двухмерной и трехмерной графики [2, 3]. Система обработки двухмерной графики является собственной разработкой, и статья посвящена её верификации.

1. Библиотека переносимых тестов

При верификации встроенного графического контроллера использовалась методика построения тестов, обеспечивающая возможность их запуска в различных тестовых окружениях без модификации. Также была обеспечена совместимость с эталонной функциональной моделью вычислительного комплекса, прототипами, основанными на ПЛИС, и с x86-совместимыми системами.

Исходный код теста представлен на языке C++, что позволяет компилировать его как для архитектуры разрабатываемой системы («Эльбрус»), так и для других платформ, на которых возможно исполнение (x86). Доработанные в рамках данной работы тестовые окружения и библиотеки обеспечили унифицированный программный интерфейс, позволяющий обеспечить переносимость теста между различными целевыми конфигурациями. Этот интерфейс содержит средства управления оперативной памятью и памятью устройства, трассировки, обращения к подсистеме прерываний и другими объектами.

При использовании тестового окружения для автономной верификации на RTL-модели вычислительного устройства тест компилируется как подключаемый к ней DPI-модуль. Тестовое окружение обеспечивает взаимное преобразование библиотечных вызовов в пакеты интерфейса между RTL-моделью и тестом.

Для системной верификации окружение генерирует образы тестов для требуемой архитектуры, пригодные для исполнения в условиях недоступности операционной системы. Разработанные тесты также возможно исполнять на прототипе вычислительного комплекса и программной модели.

В рамках работ по верификации графической подсистемы тестовое окружение было доработано с учетом накопленного опыта и новых требований, вызванных необходимостью поддержки новых устройств.

2. Структура графического контроллера и особенности реализации его программной модели

Структура контроллера

Общая структура встраиваемого графического контроллера MGA2 приведена на рис. 1. Графический контроллер управляется процессорным ядром через внешний интерфейс, который можно разбить на следующие секции: VGA IO и MemBar – доступ к управляющим регистрам контроллера, VGA Mem – доступ к видеопамяти.

С функциональной точки зрения основной задачей графического контроллера является представление на экране изображения, сохраненного в видеопамяти процессорным ядром. Кроме того, MGA2 содержит графический сопроцессор – блиттер BitBlt, осуществляющий потоковую обработку двухмерной графики и поддерживающий генерацию изображения на основе лежащих в оперативной или видеопамяти образов.

Графический контроллер поддерживает два режима работы – основной и режим совместимости со стандартным видеоинтерфейсом VGA [4]. В основном режиме изображе-

ние формируется либо процессорным ядром, либо блиттером BitBlt. Дисплейные контроллеры DC0 и DC1, используя сформированное изображение, создают образ экрана и передают его на видеовыходы VideoOuts. Управление функционированием контроллера в основном режиме осуществляется через блок регистров MGA2 Regs. Дополнительный режим необходим для поддержки совместимости с приложениями, использующими VGA. В этом режиме все функции видеоадаптера осуществляются устройствами, входящими в VGA-модуль, а управление ведется через блок регистров VGA IO Regs. Кроме того, в графическом контроллере присутствует ряд вспомогательных устройств. Блок Memory осуществляет доступ устройств к оперативной и видеопамяти. Модуль IntControl контролирует выдачу устройством прерываний, а IntIFace передает их адресатам. Модуль AutoControl осуществляет автоматическое управление функционированием контроллера.

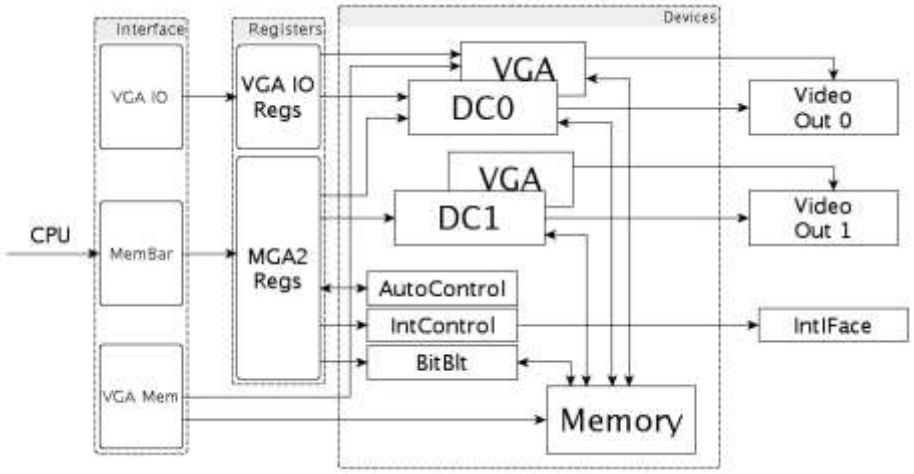

Рис. 1

Общая структура графического контроллера

Особенности реализации программной модели

Программная модель предназначена для использования в тестовой системе как эталон и для отладки тестов. Поэтому в рамках поставленной задачи нет необходимости в точном моделировании механизмов развертки, а следует достаточно корректно отображать видеопамять на экран в соответствии с заданными параметрами. Пользователю программной модели предоставляются интерфейсы, аналогичные интерфейсам видеоадаптера:

VGA IO, MemBar и Vga Mem. Общая схема генерации модельного изображения на основе образа видеопамяти представлена на рис. 2.

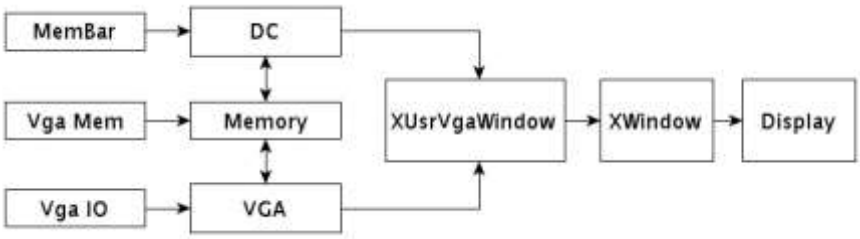

Рис. 2

Общая схема функционирования модели

Операционная система Linux, используемая на исполняющих код программной модели инструментальных машинах, предоставляет оконную систему XWindow [5] для построения графического интерфейса. XWindow позволяет формировать дисплейное изображение на основе массива пикселей, сохраненных в формате BxVgaRgb [7]; взаимодействие с XWindow осуществляется через интерфейсный модуль XUsrVgaWindow, который включен в модели дисплейного контроллера и VGA-модуля.

Изменения в изображении, как правило, носят локальный характер, и постоянное формирование всего экрана нерационально. Чтобы упростить отслеживание изменений в состоянии дисплея и вывод актуального изображения на экран, последний разбивается на прямоугольные области Tiles, изображение которых создается в дисплейном контроллере или в VGA-модуле и затем передается в XUsrVgaWindow вместе с координатами области.

На рис. 3 показана последовательность формирования образа каждой области Tile в дисплейном контроллере. Модуль ScreenState отслеживает изменения в видеопамяти и по координатам области на экране определяет необходимость её перерисовки. DrawUnit определяет положение образа области Tile в видеопамяти, осуществляет чтение данных и передает их в устройства формирования цвета – Pallette, GammaCorrection, Dither. Реализация этих устройств в модели полностью отражает аппаратуру. В Palette хранится палитра, содержащая 256 цветов, для формирования восьмибитного изображения. Модуль Dither

осуществляет дизеринг (сглаживание) изображения [6]. В модуле GammaCorrection хранятся три палитры по 256 цветов для коррекции каждого из трех цветовых каналов RGB. Полученное в результате изображение области Tile передается в систему XWindow посредством XUsrVgaWindow.

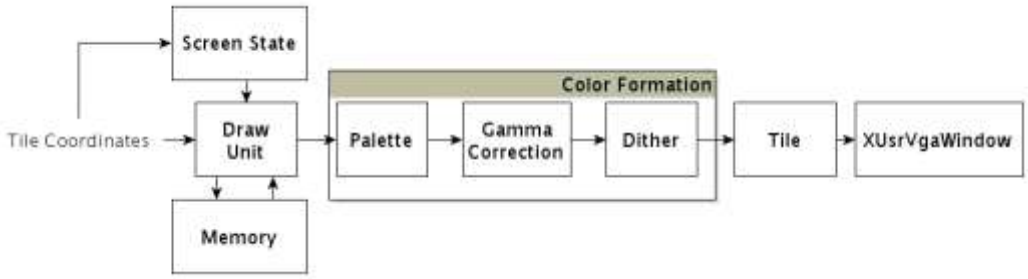

Рис. 3

Алгоритм обработки изображения

Модель VGA-совместимой части, структурная схема которой изображена на рис. 4, представляет собой переработанную модель видеоадаптера VGA. Управление режимами функционирования VGA-модуля осуществляется блоком программируемых регистров VGA Subctrl Registers. VGA Screen Manager интерпретирует состояние регистров управления VGA, читает данные из памяти посредством Vga Mem Interface и формирует дисплейное изображение.

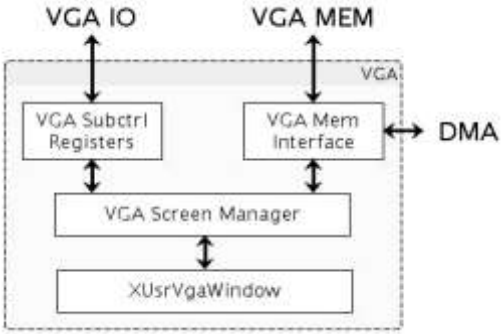

Рис. 4

Схема модуля VGA

3. Тестовая система

Для проверки функционирования графического контроллера была разработана те-

стовая система, позволяющая проверять следующую функциональность:

- функционирование регистров VGA- и MGA2-блоков;
- запись и чтение видеопамяти в различных режимах;
- запись регистров дисплейных контроллеров с буферизированным доступом;
- работу блока копирования и наложения фрагментов изображений (BitBlt);
- корректность работы подсистемы прерываний;
- функционирование контроллера автоматического управления;
- работу блоков формирования тактового сигнала.

В первую очередь был разработан набор базовых тестов, которые выполняет VGA-BIOS. Использование библиотеки для переносимых тестов позволило производить запуск тестов как на микропроцессоре с системой команд «Эльбрус», так и на вычислительных машинах с архитектурой x86 под управлением операционной системы DOS. Кроссплатформенность тестового окружения позволила отладить тесты на программной модели и собрать информацию о значениях регистров VGA в базовых режимах работы, используя видеокарты ATI Radeon 7500, ATI rage 128, S3 Trio 64, Nvidia Gf4MX-440, Nvidia Riva TNT2. На основе полученных данных проведена верификация модели VGA-совместимой видеокарты, подключенной через PCI-интерфейс программной модели микропроцессора «Эльбрус-4С». Отлаженная модель VGA была встроена в программную модель MGA2 в составе микропроцессора «Эльбрус-1С+».

На большую часть внешних воздействий графический контроллер реагирует изменением изображения на экране. Изображение анализируется в тестировании палитры, регистров дизеринга, гамма-коррекции, управления курсором и других модулей, влияющих на изменение состояния дисплея и формирование изображения в различных предусмотренных спецификацией режимах. На этом этапе была поставлена задача сведения к минимуму неавтоматизированного зрительного анализа результатов тестирования.

Программная модель предоставляет возможность автоматизировать визуальные те-

сты при помощи сохранения состояния дисплея в определенный момент времени в графический файл и сравнения с эталоном. Для RTL-описания микропроцессора «Эльбрус-1С+» в тестовое окружение также были включены блоки, работающие с графическими файлами. Для проверки корректности формирования изображений на RTL-описании микропроцессора был предложен следующий алгоритм:

1) bmp-конвертер преобразует заданный bmp-файл в массив данных в указанном формате;

2) тестовое окружение, использующее библиотеку для переносимых тестов, записывает значения массива в видеопамять;

3) модуль, подключенный к видеовыходам MGA2, сохраняет полученные данные в bmp-файл;

4) с необходимой точностью сравниваются активные области в исходном и полученном bmp-файлах.

Для полного охвата функционала устройства было принято решение наряду с системной верификацией проводить также автономную. Непосредственное взаимодействие тестового окружения с интерфейсами MGA2 позволяет сократить время работы теста и более точно воссоздать тестовую ситуацию, приводящую к возникновению ошибки.

Для проведения автономной верификации в MGA2 выделены блоки:

- мультиплексор внутренних шин запросчиков на единую внешнюю шину AXI;
- контроллер автоматического управления;
- блок копирования и наложения фрагментов изображений;
- управляющие блоки регистров MGA2 и VGA.

Тестовое окружение разработано по методологии OVM. Оно состоит из ряда статических компонент, представляющих собой гибкие, легко конфигурируемые блоки, реализующие выбранную архитектуру тестового окружения на протяжении всего процесса моделирования, и динамических компонент (транзакций), представляющих собой минималь-

ную единицу обмена информацией между статическими компонентами, выбранную с учетом интерфейса или протокола передачи. Архитектура тестового окружения OVM для автономной верификации RTL-описания MGA2 представлена на рис. 5.

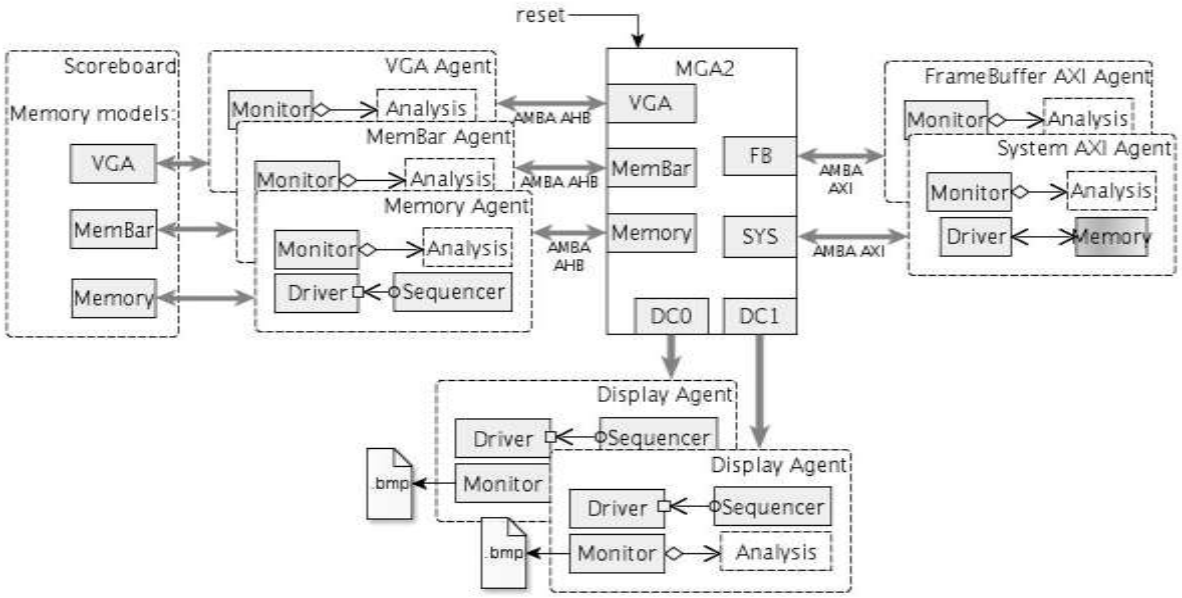

Рис. 5

Архитектура тестового окружения OVM для RTL-описания MGA2

Принцип автономного тестирования блоков, отвечающих за формирование изображения, аналогичен системному тестированию. Для конвертера изображения форматы выходных файлов были изменены с учетом особенностей OVM, в то время как модули сохранения в файл и сравнения изображений одинаковы для обоих тестовых окружений.

4. Предварительные результаты и дальнейшие разработки

На момент написания статьи были практически завершены разработки программных моделей, а также автономных и системных тестовых сценариев. С помощью тестов для MGA2 было обнаружено более 30 ошибок в RTL-описании графического контроллера MGA2 и около 10 ошибок межблочного взаимодействия компонентов микропроцессора «Эльбрус-1С+».

Во всех видах тестирования графической подсистемы ставится задача автоматизации

сравнения результатов с эталоном. Наиболее остро она проявляется для тестов с высокой частотой перерисовки изображения, что объясняется сложностью определения момента для сохранения эталонного состояния экрана.

Для программной модели одним из решений этой задачи может стать добавление функциональности, которая позволит сохранять текущее состояние дисплея по команде из теста. На момент написания статьи этот способ еще не был реализован, и основным подходом к системной верификации регистров, отвечающих за отображение видеопамяти для программной модели, являлась визуальная проверка.

Для сокращения времени исполнения тестов, запускаемых на RTL-модели микропроцессора, представляется целесообразным исключить вычисления, связанные с механизмом проверки эквивалентности изображений во время выполнения теста. Вместо этого предлагается формировать эталонные значения на программной модели, а на RTL-модели осуществлять только проверку эквивалентности полученных данных.

Заключение

Представленная работа явилась первым опытом компании ЗАО «МЦСТ» по направленной верификации графических контроллеров. В процессе верификации возникло определенное число проблем, среди которых можно отметить сложность разработки корректной программной модели графического адаптера, сложность создания универсальных тестов, исполняющихся как на архитектуре «Эльбрус», так и на архитектуре x86, и необходимость автоматизации проверки результатов работы устройств, формирующих изображение. Тем не менее, основные проблемы были преодолены, что позволило разработать многоступенчатую тестовую систему, которую можно применять к дальнейшим разработкам ЗАО «МЦСТ».

Литература

1. Manish Arora, The Architecture and Evolution of CPU-GPU Systems for General Purpose Computing. – Department of Computer Science and Engineering, UC San Diego, 2012.

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

3. http://www.mcst.ru/ekonomichnyj-mikroprocessor-s-arkhitekturoj-elbrus-i-vstroennym-graficheskim-yadrom.

4. IBM VGA Technical Reference Manual.

5. http://tronche.com/gui/x/xlib/utilities.

6. Kjell Lemstrom et all, Color Dithering with n-Best Algorithm. – The IV International Conference in Central Europe on Computer Graphics and Visualization, Plzen, 1996.

7. www.x.org.