УДК 004.052.42

Возможности использования виртуальных платформ для верификации RTL-моделей сложно-функциональных блоков в составе «Систем на кристалле»

И.И. Шагурин, Е.И. Литвинов, Г.Ю. Жихарев

Национальный исследовательский ядерный университет «МИФИ»

НИЯУ «МИФИ», litvinov.ei@gmail.com

***Аннотация* — В докладе описывается использование современных виртуальных платформ для проектирования и моделирования цифровых СБИС класса «система на кристалле» (СнК). Показано, что виртуальные платформы могут не только использоваться в архитектурном моделировании разрабатываемых цифровых систем и создании их программного обеспечения, но и дополнять современные средства верификации моделей сложно-функциональных блоков (СФ-блоков), входящих в состав СнК. Предлагается методика верификации RTL-моделей СФ-блоков с помощью виртуальных платформ. Дается пример применения предложенной методики.**

***Ключевые слова* — виртуальная платформа, система на кристалле, сложно-функциональный блок, функциональное тестирование, RTL-модель, верификация.**

I. Введение

Структура современных систем на кристалле (СнК) обычно содержит одно или несколько процессорных ядер, специализированные сопроцессоры, ориентированные на эффективное выполнение определенных алгоритмов и процедур, и набор необходимых периферийных устройств (контроллеры памяти, интерфейсные блоки, таймеры и другие). Сокращение сроков, снижение стоимости и повышение качества проектирования СнК достигается путем применения в процессе разработки готовых сложно-функциональных блоков (СФ-блоков), представленных в виде синтезируемой RTL-модели, схемы соединения логических ячеек (netlist) или топологического фрагмента СБИС. Разработчик СнК может использовать готовые СФ-блоки, имеющиеся в свободном доступе или предлагаемые на коммерческой основе, и вводить в проект СФ-блоки собственной разработки. При этом одним из наиболее ответственных и трудоемких этапов разработки СнК является верификация проекта на разных этапах его выполнения [1]. В данной статье описываются возможности использования виртуальных платформ на этапе верификации аппаратной реализации проекта, что позволяет сократить сроки и снизить трудоемкость выполнения этого этапа.

Виртуальная платформа содержит набор моделей, которые можно использовать для представления различных СФ-блоков тестируемого проекта и необходимого тестового окружения, а также комплекс правил (сценариев) для создания новых моделей, генерации тестов, оценки результатов тестирования. Для сокращения времени тестирования используются поведенческие модели СФ-блоков на языке C/C++, модели тестового окружения обычно описываются на языках SystemC, SystemVerilog. Виртуальные платформы широко используются для разработки и отладки программного обеспечения проектируемых СнК, обеспечивая симуляцию аппаратной реализации выполняемого проекта до его фактического изготовления [2]. Ряд виртуальных платформ (Simics [3, 4], Seamless [5], OVP [6] и некоторые другие) содержат средства, позволяющие не только создавать и отлаживать программное обеспечение, но и моделировать его выполнение с использованием RTL-моделей СФ-блоков, входящих в состав СнК. В данном докладе эти возможности рассматриваются на примере использования платформы Open Virtual Platform (OVP) компании Imperas, в составе которой имеется свободно распространяемый симулятор OVPsim [7].

Платформа OVP обеспечивает функциональное моделирование цифровых систем с точностью до инструкций (instruction accurate), при котором в память и регистры модели заносятся корректные данные по окончанию исполнения каждой инструкции и реализуются корректные операции, вызываемые данной инструкцией. Поддерживается моделирование многоядерных/многопроцессорных систем в однотактном режиме, который предполагает, что каждый процессор или ядро исполняет одну инструкцию за «такт». При использовании общих ресурсов, например, памяти, обеспечивается доступ нескольких устройств к запрошенному ресурсу и успешное выполнение транзакций, однако не гарантируется требуемая последовательность доступа разных устройств к этому ресурсу.

II. Модельное представление СнК средствами платформы OVP

Платформа OVP представляет разрабатываемую систему как совокупность моделей отдельных устройств, связанных интерфейсами API. На рис. 1 показан пример модели двухпроцессорной системы с разделенной памятью и набором периферийных устройств, представленных в виде поведенческого описания. Для формирования общей модели системы ис-

пользуются стандартизированные интерфейсы [8]:

1) ICM (Innovative CPU Manager). Служит для взаимосвязи моделей отдельных устройств.

2) VMI (Virtual Machine Interface). Предназначается для связи моделируемых процессоров с симулятором.

3) BHM (Behavioral Hardware Model). Используется для подключения устройств, представленных поведенческим описанием.

4) PPM (Peripheral Programming Model). Необходим для подключения к модели периферийных устройств.

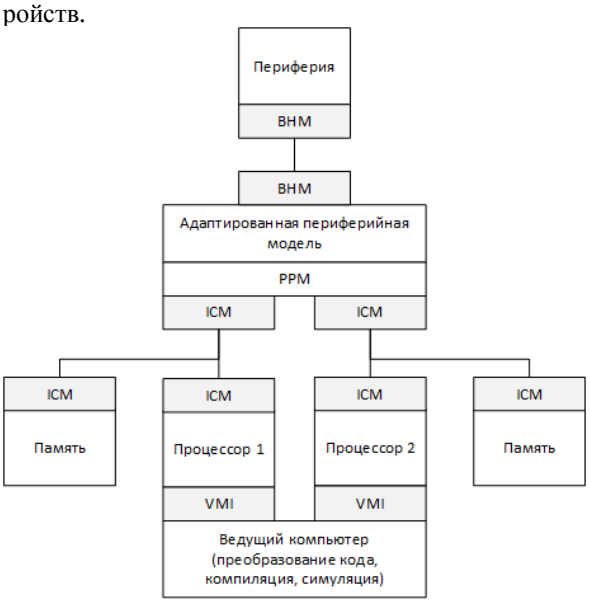

Рис. 1. Интерфейсы платформы OVP

Интерфейс ICM служит для формирования общей модели системы, реализуя взаимосвязь и контроль моделей отдельных устройств. При этом обеспечивается также необходимое отображение адресного пространства и загрузка кода программы в память системы. С помощью данного интерфейса формируется модель проектируемой системы, которая может состоять из различных процессоров, внутренней и внешней памяти, мостов и периферийных устройств. Затем на этой платформе можно запустить операционную систему или пользовательские приложения.

Интерфейс VMI обеспечивает связь модели процессора, написанной на языке С, с ведущим компьютером, реализующим симуляцию выполняемой программы. Модель процессора декодирует выбираемый из памяти программный код, предназначенный для выполнения системой, и преобразует его в инструкции ведущего компьютера. Результаты выполнения инструкций передаются модели процессора от ведущего компьютера по интерфейсу VMI и далее используются для моделирования поведения проектируемой системы в соответствии с выполняемой программой. Интерфейс VMI позволяет моделировать 8, 16, 32 и 64-разрядные процессорные архитектуры.

Интерфейсы BHM, PPM служат для подключения к модели системы периферии и других устройств, представленных в виде поведенческих моделей, описанных на языке С. Отличия данных интерфейсов в том, что BHM отвечает за управление процессами и событиями, а PPM за взаимодействие с платформой.

Интерфейсы BHM выполняют функции арбитров при обслуживании запросов различных периферийных устройств, обеспечивая добавление и контроль необходимых задержек или организацию параллельного выполнения задач.

Интерфейс PPM предоставляет доступ к компонентам платформы, шинам, проводам и портам. Так как периферийные компоненты работают в изолированном окружении с использованием дополнительного симулятора PSE (Peripheral Simulation Engines) и имеют своё адресное пространство, то данный интерфейс отображает адресное пространство периферийных моделей в реальное адресное пространство платформы. При этом сначала с помощью интерфейса BHM производится формирование адаптированной модели периферии (добавление обратных вызовов, исполняемых по указанным событиям), которая связывается с общей моделью системы посредством интерфейса PPM.

III. Методика использования виртуальных платформ для верификации аппаратных средств СнК

Для верификации модели цифрового устройства, представленной на уровне регистровых передач (RTL), необходимо интегрировать виртуальную платформу в тестовое окружение для получения общей системы тестирования (рис. 2).

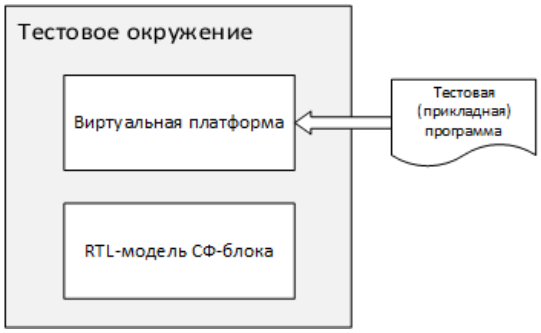

Рис. 2. Общая структура системы тестирования

Виртуальная платформа – это модель цифровой системы, управляемая одним или несколькими процессорами стандартной архитектуры (IA-32, ARM, MIPS или другая), которая сформирована в соответствии с технологией OVP. Для её создания используются описанные выше интерфейсы компании Imperas и симулятор OVPsim, представленный в виде динамически подключаемой библиотеки (DLL/SO), в которой содержатся реализации функций интерфейсов. К системе подключается специализированный СФ-блок, представленный в виде RTL-модели на языке Verilog.

Получаемая комплексная модель, состоящая из поведенческого описания на языках C/C++ и RTL-описания СФ-блока, далее называется гибридной моделью СнК. Верификация гибридной модели СнК

осуществляется путем контроля корректности выполнения прикладной программы, написанной на языке C или C++. В процессе тестирования модели в качестве прикладной программы может использоваться набор специальных тестовых программ. Кроме того достигается заметное ускорение исполнения тестовых сценариев благодаря отсутствию необходимости моделирования поведения основных частей системы на уровне RTL.

Чтобы виртуальная платформа взаимодействовала с тестируемым СФ-блоком, все компоненты системы запускаются с помощью тестового окружения, написанного на языке SystemVerilog с использованием современной библиотеки Universal Verification Methodology (UVM) [9]. Тестовое окружение взаимодействует с виртуальной платформой посредством вызовов функций, которые осуществляются с помощью стандартного интерфейса Direct Programming Interface (DPI) языка SystemVerilog. Шаблоны функций, написанные на языке SystemVerilog, входят в состав создаваемого тестового окружения, а реализация этих функций обеспечивается библиотекой, имеющейся в составе используемой виртуальной платформы. При этом используются следующие функции OVP и функции собственной разработки:

1) «platform_init». Инициализирует платформу, создавая процессор с указанными атрибутами, память, шину передачи данных, необходимые буферы для передачи информации в тестовое окружение, а также устанавливает диапазон адресов памяти для выполнения обратных вызовов.

2) «platform_load_app». Обеспечивает загрузку тестовых (прикладных) программ в память системы.

3) «platform_simulate». Запускает процесс моделирования выполнения инструкций тестовых (прикладных) программ в пошаговом режиме или в режиме прогона.

4) «platform_exit». Завершает работу виртуальной платформы, осуществляя очистку памяти, выводит статистику и тому подобное.

5) «platform_rtl_take_control». Передает управление системой виртуальной платформе.

6) «platform_rtl_get_address». Устанавливает адрес, по которому RTL-модель СФ-блока может записывать данные. Виртуальная платформа запоминает этот адрес и осуществляет обратный вызов при записи данных по указанному адресу.

7) «platform_rtl_get_data». Осуществляет по указанному адресу запись данных, поступающих от RTL-модели СФ-блока. При обратном вызове виртуальная платформа считывает записанные данные.

8) «platform_bd_wr_wd». Обеспечивает запись в память данных, поступающих от процессора.

9) «platform_bd_rd_wd». Обеспечивает чтение процессором данных из памяти.

На рис. 3 показан тестовый комплекс, в состав которого входит тестовое окружение, написанное на языке SystemVerilog. В соответствии с методологией UVM в процессе тестирования используются различные тестовые программы, каждая из которых содержит свои настройки и набор тестовых последовательностей данных. Модель тестируемого СФ-блока подключается к тестовому окружению через интерфейсы, обеспечивающие прием и передачу управляющих сигналов и данных.

Верхний уровень иерархии тестового окружения представляет собой модуль (top_module), который формирует тактовый сигнал, обеспечивает управление начальным запуском RTL-модели тестируемого СФ-блока, создает необходимые компоненты и интерфейсы, а также осуществляет запуск выбранного теста. В рамках каждого теста запускается одна рабочая тестовая последовательность и загружается тестовое приложение в машинном коде во внутреннюю память процессора, представленного моделью OVP (виртуальная платформа). Тестовые последовательности содержат комбинации транзакций, которые создают требуемый сценарий возникновения событий.

Каждый тест uvm_test представляет собой контейнер, содержащий необходимые настройки и наборы тестовых последовательностей. Кроме этого, каждый тест создает дополнительные компоненты тестового окружения, такие как uvm_environment.

Модуль uvm_environment обеспечивает создание всех главных компонент, требуемых для выполнения тестов. В состав этих компонент входят:

1) Uvm_agent. Модуль-контейнер, служащий для корректного обслуживания конкретных интерфейсов тестируемого СФ-блока. Содержит в своем составе компонент uvm_sequencer, который формирует необходимый набор логических сигналов, подаваемых на входы тестируемого СФ-блока, и компоненты uvm_driver, uvm_monitor, выполняющие, соответственно, передачу этих сигналов и прием сигналов, получаемых с выходов СФ-блока.

2) Virtual_sequencer. Обеспечивает централизованный контроль подачи различных тестовых последовательностей на соответствующие интерфейсы тестируемого СФ-блока, определяет порядок работы имеющихся модулей uvm_agent.

3) Uvm_scoreboard. Проверяет корректность функционирования тестируемого СФ-блока путем сравнения получаемых данных с ожидаемым результатом.

Формирование тестового комплекса осуществляется с помощью стандартной утилиты make и соответствующего файла с набором инструкций - Makefile.

Для сборки всего комплекса, включающего гибридную модель СнК и необходимые компоненты тестового окружения, а также запуска процесса тестирования, требуется выполнить следующие процедуры.

1) Скомпилировать файлы тестируемого устройства и тестового окружения.

2) Для использования UVM макросов создать динамически-подключаемую библиотеку.

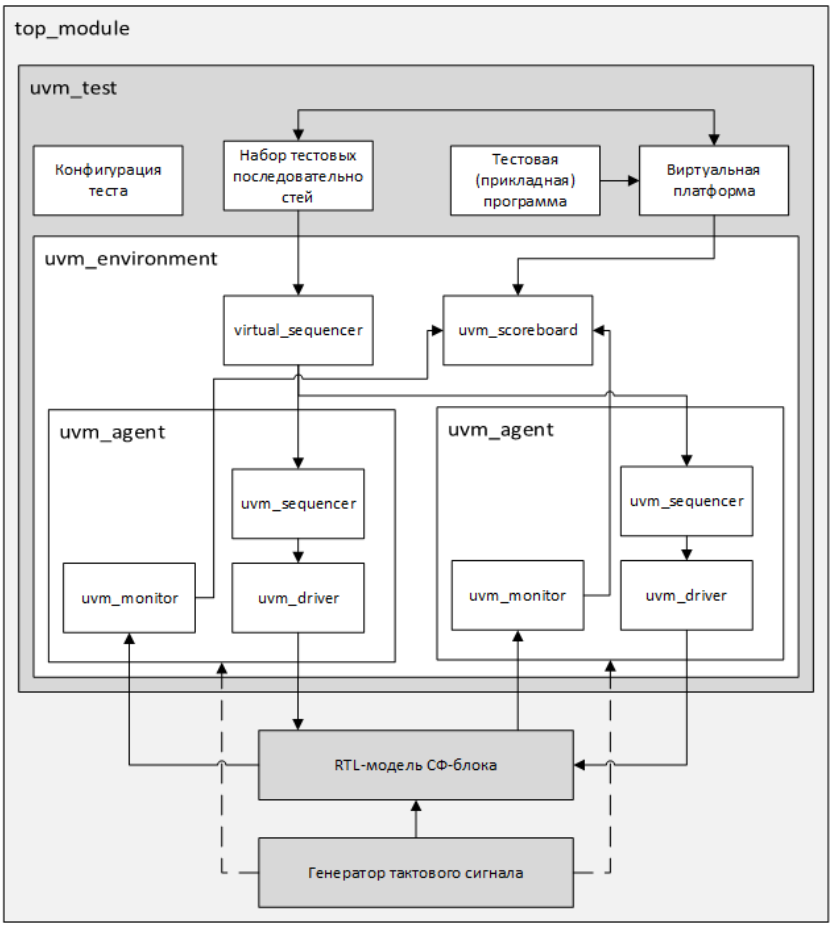

Рис. 3. Состав тестового комплекса

3) Получить исполняемый файл тестовой (прикладной) программы, который будет запускаться на виртуальной платформе.

4) Создать динамически-подключаемую библиотеку виртуальной платформы (включает в себя симулятор и используемую виртуальную платформу).

5) Скомпилировать тестовую (прикладную) программу для ее выполнения центральным процессором.

6) Выполнить линковку тестовой программы.

7) Запустить процесс моделирования с помощью стандартных средств моделирования RTL-описаний.

IV. Применение предлагаемой методики

Данная методика использовалась для верификации RTL-модели СФ-блока специализированного арифметико-логического сопроцессора, работающего в составе СнК (рис. 4) под управлением центрального процессора (ЦП). В составе виртуальной платформы использовались готовые поведенческие модели 32-разрядного процессора OR1K, общей оперативной памяти емкостью 4 ГБ и 32-разрядной шины, написанные на языке C, которые представлены в документации на OVP [6]. Поэтому создание виртуальной платформы потребовало значительно меньших трудозатрат, чем написание собственных моделей без применения технологии OVP. Модель тестируемого сопроцессора описана на языке Verilog HDL. Все компоненты тестового окружения, такие как: драйвер, монитор, генератор тестов и другие, написаны на языке SystemVerilog.

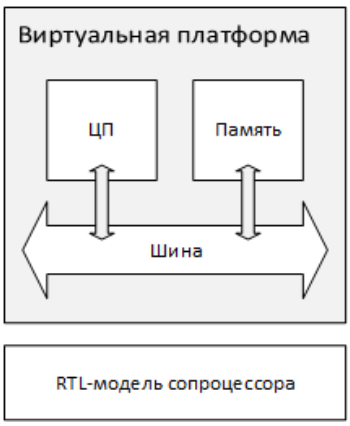

Рис. 4. Структура тестируемого фрагмента СнК

Для обработки файлов, написанных на языках Verilog HDL/SystemVerilog, использовались симуляторы Questasim/Modelsim. На рис. 5 показаны файлы верхнего уровня, используемые в процессе тестирования проекта.

SSRC – исходные файлы тестового окружения, global_inc.sv – включает все файлы UVM компонент, интерфейсов и т.п., ovp_tb_top.sv – модуль верхнего уровня (top_module на рис. 3).

RSRC – исходные файлы RTL-модели СФ-блока, dut.inc.v – файл включает все исходные RTL- файлы.

uvm_dpi.cc – исходный код UVM макросов, используемых при описании тестового окружения (из этого файла формируется динамически подключаемая библиотека uvm_dpi.so).

ASRC – исходные файлы тестовой (прикладной) программы application.c. На начальном этапе получаем объектный файл этой программы, затем исполняемый файл для конкретной платформы.

PSRC – исходные файлы платформы. Содержат все необходимые функции для создания и работы с платформой, которые экспортируются с использованием интерфейса DPI в тестовое окружение. Для экспорта создаётся динамически подключаемая библиотека.

На рис. 5 не показаны файлы интерфейсов OVP, которые используются в составе проекта при создании динамически-подключаемой библиотеки виртуальной платформы (см. пункт 4 раздела III).

Для каждой группы файлов проекта в Makefile определена цель, которой является либо скомпилированный файл, либо динамически подключаемая библиотека.

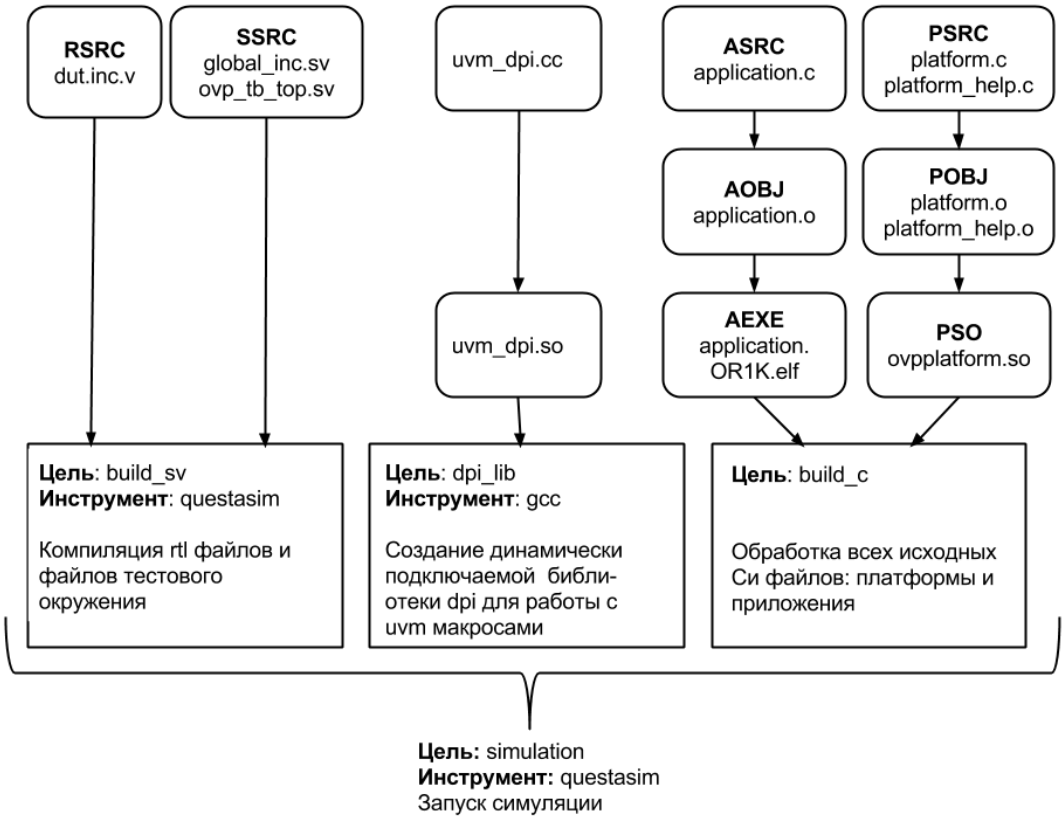

Рис. 5. Формирование тестового комплекса на базе виртуальной модели

Для достижения этой цели используются разные средства - компиляторы языков (например, gcc), средства работы с RTL-описаниями (например, Modelsim/Questasim). Процесс сборки тестового комплекса является полностью автоматизированной процедурой.

В соответствии с описанной методикой далее выполнялись следующие этапы создания тестового комплекса:

1) Компиляция файлов тестируемого СФ-блока и тестового окружения (компиляция файлов RSRC и SSRC).

2) Создание библиотеки для использования UVM макросов (компиляция uvm_dpi.cc).

3) Получение исполняемого файла тестовой (прикладной) программы для запуска на виртуальной платформе (файлы ASRC). Для компиляции используется набор утилит cross compilers:

```
$(PLATFORM)/%.o: $(PLATFORM)/%.c
$(V) echo "Compiling $<"
$(CC) -fPIC -c $<
-I$(IMPERAS_HOME)/ImpPublic/include/host
-I$(MTI_HOME) -o $@ \
$(SIM_CFLAGS)     \
```

```
-DTYPE_NAME=\"$(NAME)\"    \
-DMORPHER_FILE=\"$(PROC)\" \
-DMORPHER_ATTR=\"$(ATTR)\" \
-DSEMIHOST_FILE=\"$(SEMI)\"
```

4) Создание динамически-подключаемой библиотеки виртуальной платформы, которое производится с помощью процедуры (файлы PSRC):

```
$(PSO): $(POBJ)
$(V) echo "Linking $<"
$(V) $(CC) -shared -o $@ $^
$(PLATFORM_LDFLAGS) $(LDFLAGS)
$(IMPERAS_ICMSTUBS)
```

5) Компиляция тестовой (прикладной) программы для ее выполнения центральным процессором:

```
$(APPLICATION)/%.o: $(APPLICATION)/%.c
        $(V) echo "Compiling $<"
        echo "$(IMPERAS_CC)"
        $(IMPERAS_CC) -c -o $@ $<
```

6) Общая линковка тестовой программы:

```
$(AEXE): $(AOBJ)
        $(V) echo "Linking $<"
        $(V) echo "$(AOBJ)"
        $(V) echo "$(APPLICATION)/application.o"
        $(IMPERAS_LINK) -o $@ $^
        $(IMPERAS_LDFLAGS) –lm
```

7) Запуск процесса моделирования с помощью стандартных средств для моделирования RTL-описаний (средства Modelsim/Questasim).

В результате выполнения этих процедур формируется модель тестового комплекса (рис. 4), которая позволяет совмещать процесс отладки программного обеспечения проектируемой СнК с процессом моделирования и верификации RTL-моделей СФ-блоков, входящих в ее состав. После этого можно осуществлять тестирование полученной гибридной модели СнК с использованием единых средств моделирования.

V. Заключение

Предложенная методика позволяет эффективно производить верификацию моделей сопроцессорных СФ-блоков с помощью виртуальных платформ, позволяя получить существенное ускорение исполнения тестов.

Методика предполагает совместное использование мощных средств, доступных для виртуальных платформ (широкий набор готовых моделей для большого класса процессорных архитектур: MIPS, ARM, PowerPC, MicroBlaze и другие, наличие утилит для компиляции и отладки программного обеспечения) и современных методологий верификации аппаратуры (UVM). Создаваемый на базе платформы единый тестовый комплекс позволяет решать следующие задачи:

- верификация RTL-моделей разрабатываемых аппаратных средств (СФ-блоков) СнК;
- разработка (отладка) программного обеспечения;
- совместная верификация и комплексная отладка аппаратно-программных средств СнК.

Использование единого тестового комплекса позволяет избежать применения различных моделей и версий тестового окружения для решения задач тестирования, что обеспечивает сокращение трудозатрат, необходимых для разработки СнК.

Литература

[1] Баратов Р.А., Камкин А.С., Майорова В.М., Мешков А.Н., Якушева М.А., Сортов А.А. Трудности модульной верификации аппаратуры на примере буфера команд микропроцессора «Эльбрус-2s» // Вопросы радиоэлектроники. Серия ЭВТ. 2013. Вып. 3. С. 84-96.

[2] Using Virtual Platforms for Pre-Silicon Software URL: http://www.iqmagazineonline.com/IQ/IQ24/pdfs/IQ24_Using%20Virtual%20Platforms.pdf (дата обращения 20.01.2014).

[3] Popovicic K., Jerraya A. Virtual Platforms in System-on-Chip Design // Knowledge center article DAC’47, 2010.

[4] Engblom J. Simics System Modeling URL: http://www.isk.kth.se/~ablak/Documents/wp-modeling-2009-03-25-letter.pdf (дата обращения 20.01.2014).

[5] Seamless Hardware/Software Integration Environment. URL: http://www.mentor.com/seamless (дата обращения 20.01.2014).

[6] Open Virtual Platforms. URL: http://www.ovpworld.org/ (дата обращения 20.01.2014).

[7] OVPsim and Imperas CpuManager User Guide. URL: http://www.ovpworld.org/documents/OVPsim_and_CpuManager_User_Guide.pdf (дата обращения 20.01.2014).

[8] Bailey B. System Level Virtual Prototyping becomes a reality with OVP donation from Imperas // White Paper: System Level Virtual Prototyping & OVP, 2008. URL: http://www.ovpworld.org/documents/BrianBaileyWhitePaper_SLVP_and_OVP.pdf (дата обращения 20.01.2014).

[9] Ровнягин М.М. Использование UVM для автономной верификации цифровой аппаратуры // Проблемы разработки перспективных микро- и наноэлектронных систем – 2012. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2012. С. 129-132.