ВИРТУАЛЬНОЕ ПРОТОТИПИРОВАНИЕ

ДЛЯ АППАРАТНО-ПРОГРАММНОЙ ВЕРИФИКАЦИИ СБИС

В.Кравченко,
Д.Радченко
vitaly.kravchenko@alt s.com

Время – деньги. Этот тезис особенно весом в современной электронике при разработке аппаратуры. Но до недавнего времени все были вынуждены мириться с тем, что отладко встроенного программного обеспечения начинается о завершению проектирования аппаратной части системы – а как же иначе? Но с развитием концепции виртуального прототипирования ситуация кардинально изменилась.

ПРОБЛЕМЫ СОВМЕСТНОЙ ВЕРИФИКАЦИИ ПРОГРАММНОЙ И АППАРАТНОЙ ЧАСТЕЙ СБИС

В 2003 году около 80% всех вновь разработанных СБИС включали по крайней мере одно процессорное ядро. Следовательно, неотъемлемой частью всех этих систем на кристалле (SoC) является встроенное программное обеспечение (ПО). И первая проблема, с которой сталкивается разработчик, – это выбор необходимого баланса между аппаратной и программной частями системы: какие функции лучше реализовать программно, какие – аппаратно. Причем этот баланс может изменяться в процессе работы. То есть процессы создания аппаратуры и ПО – взаимозависимы.

Также очевидно, что ПО должно быть написано, отлажено и верифицировано с учетом особенностей аппаратной реализации всей системы. Например, необходимо учитывать, что время доступа и задержки обращения к ОЗУ могут существенно меняться в зависимости от загруженности шины, приоритетов, назначенных процессорному ядру, арбитражу на шине и т.п.

Кроме того, сжатые сроки проектирования, диктуемые рынком и заказчиками, требуют, чтобы разработка и верификация ПО не задерживала работы над всем кристаллом, т.е. выполнялась не после завершения работ над аппаратной частью системы, а параллельно. Однако зачастую параллельная разработка и верификация аппаратной и программной частей SoC сопряжена с труднопреодолимыми проблемами.

Как правило, разработчик ПО работает в некоей среде разработки и отладки – в специальном отладчике, содержащем модель процессорного ядра. В идеальном случае при переносе ПО с модели на реальный процессор разрабатываемой SoC никаких коллизий происходить не должно. Однако это возможно, если:

- уже существует (разработана) модель аппаратной части SoC, включающая процессорное ядро и периферию;
- работа модели в отладчике с точностью до цикла соответствует аппаратной модели ядра процессора с учетом его взаимодействия с остальными элементами SoC;
- при верификации ПО в отладчике моделируется столько возможных вариантов поведения и состояния системы, сколько может возникнуть при работе реальной SoC;
- скорость исполнения модели высока настолько, что ПО можно проверить за приемлемое время.

Рассмотрим, как реализуются эти условия в существующих САПР СБИС.

ТРАДИЦИОННЫЙ ПОДХОД К МОДЕЛИРОВАНИЮ ПО И АППАРАТНОЙ ЧАСТИ

Существующие подходы к моделированию аппаратной части SoC предполагают ее описание на уровне регистровых передач (RTL), как правило – на языках описания аппаратуры (HDL): VHDL или Verilog. RTL-модели представляют собой описание аппаратуры как совокупности регистров и логических связей между ними. Взаимодействие между модулями детализировано до описания отдельных сигналов и выводов. Естественно, RTL-модели характеризуют аппаратуру с точностью до цикла. Фактически RTL-описание цифровых ИС – это законченный проект схемы в обобщенной форме, т.е. без привязки к библиотекам конкретных производителей СБИС.

Однако применение RTL-моделей для разработки и верификации ПО проблематично, поскольку:

- скорость моделирования ограничена, так как HDL-симулятор должен обрабатывать множество событий, связанных с изменениями сигналов. Это расходится с требованием высокой скорости моделирования системы.
- создание RTL-модели – это весьма существенный этап разработки аппаратной части проекта, фактически финальная часть схемотехнических работ (благодаря современным САПР). На это требуется немало времени и средств. А с ростом сложности аппаратуры начало процесса совместной верификации ПО и аппаратной части SoC отодвигается на поздние этапы проектирования (см. рис.1), что противоречит требованиям минимизации сроков разработки.

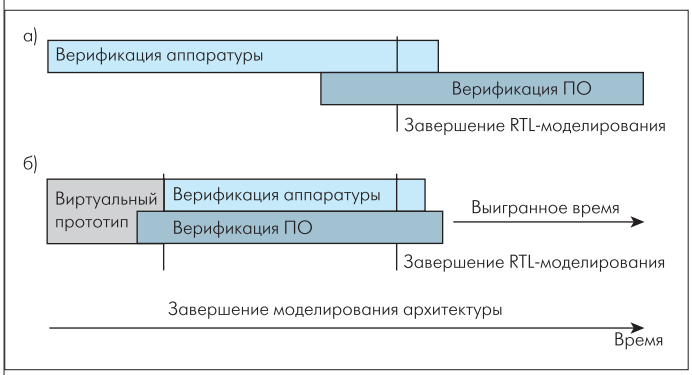

Рис.1. Сроки проектирования SoC при традиционном подходе (а) и в рамках методологии виртуального прототипирования (б)

Существенно, что названные проблемы зависят не от языка моделирования, а от уровня абстракции модели, а именно от RTL..

Почему же сегодня все еще используют RTL-описания для верификации ПО? Действительно, на системном уровне может быть с точностью до цикла проработана спецификация внешнего поведения аппаратуры (функциональная нагрузка, внешний обмен данными). Но к сожалению, в традиционном маршруте проектирования спецификация всей системы становится исполняемой моделью только тогда, когда создано ее RTL-описание.

Вместе с тем, для моделирования ПО традиционно используют командные эмуляторы, обычно оснащенные программой-отладчиком. Использование таких эмуляторов достаточно точно отражает поведение реального процессорного ядра в SoC, поскольку сами эмуляторы поставляются разработчами процессорных ядер. Однако это не снимает проблему совместной верификации ПО и аппаратуры SoC в целом, поскольку для остальной аппаратуры, не входящей в процессорное ядро, по-прежнему необходима RTL-модель.

Таким образом, при традиционном подходе верификация ПО начинается по завершении разработки RTL-описания системы. В результате не только увеличиваются сроки проектирования, но и вероятно обнаружение несоответствия между аппаратурой и ПО после завершения разработки аппаратной части. А это, в свою очередь, означает лишнюю итерацию в процессе проектирования SoC. Более того, из-за достаточно низкой общей скорости моделирования RTL-описаний (порядка 1000 инструкций в секунду) невозможна отработка сложных сценариев обмена данными, да и вообще большого числа сложных тестов. Поэтому до появления физического прототипа (например, на FPGA) взаимодействие аппаратуры и ПО может быть верифицировано весьма фрагментарно. Собственно, основная работа по верификации ПО начинается только с появлением физического прототипа, и хотя прототипы на FPGA обладают высоким быстродействием, отладка такой системы сильно затруднена.

Самое неприятное следствие перечисленных трудностей в том, что “узкие места” в архитектуре выявляются слишком поздно. И проблемы, связанные с архитектурой системы, могут остаться нераскрытыми вплоть до ее физической реализации, что таит угрозу для всего проекта. И чем сложнее аппаратура и ПО, тем пагубнее влияние позднего обнаружения ошибок.

Поэтому назрела необходимость в новом подходе к совместной разработке и верификации аппаратуры и ПО. Он называется “виртуальное прототипирование”.

ВИРТУАЛЬНОЕ ПРОТОТИПИРОВАНИЕ

В основу “виртуального прототипирования” положен простой принцип: необходимо создать поведенческую модель архитектуры до начала детальной разработки аппаратуры и ПО. При этом виртуальный прототип должен представлять из себя точную до цикла поведенческую модель всей SoC, включая и аппаратную, и программную ее части. Быстродействие обработки этой модели не может быть менее 100 тыс. инструкций в секунду, что гораздо выше, чем для RTL-описаний. Сама модель должна быть создана на как можно более раннем этапе проектирования.

На первый взгляд, задача кажется неразрешимой. Но, с другой стороны, разработчику ПО при верификации требуется с точностью до цикла проанализировать процессы передачи между блоками данных и команд, выявить моменты начала и продолжительность обмена, особенно когда время передачи зависит от параметров передаваемых пакетов. При таком анализе совершенно излишне рассматривать процессы формирования отдельных сигналов на шинах,

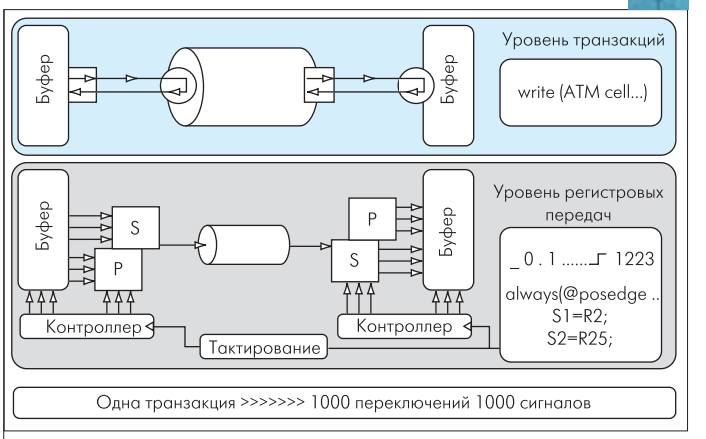

Рис.2. Различие между представлением схемы на уровне регистровых передач (RTL) и на уровне транзакций (TL)

поскольку с точки зрения ПО взаимодействие с аппаратурой сводится к обмену пакетами данных с учетом временных параметров. Поэтому все события переключения сигналов (в случае шины – до 100 отдельных сигналов) можно заменить одним событием чтения или записи, называемым транзакцией. Более того, одной транзакцией можно обозначить несколько команд чтения или записи, например – запись данных в последовательно расположенные ячейки памяти. Так как скорость моделирования обратно пропорциональна числу событий, то при снижении их интенсивности в 100–1000 раз типичная модель уровня транзакций исполняется в 100 раз быстрее соответствующей RTL-модели (рис.2). Отметим, что уровень транзакций – это фактически уровень разработки архитектуры системы.

Таким образом, эффективный путь для совместной разработки и верификации ПО и аппаратуры SoC – моделирование на уровне транзакций (TL). Существенно, что TL-модель системы – ее виртуальный прототип – после успешной верификации может служить эталоном для остальных этапов проектирования СБИС (вместо описательной спецификации).

Однако для успешного виртуального прототипирования необходимы, помимо TL-моделей отдельных блоков, средства разработки и совместной верификации ПО и TL-моделей аппаратуры. Причем средства эти должны включать язык описания моделей, среду разработки с удобным интерфейсом и механизмы автоматической кросс-компиляции описания моделей с уровня транзакций на более низкие уровни иерархии.

ОПИСАНИЕ ПРОГРАММНО-АППАРАТНЫХ МОДЕЛЕЙ НА УРОВНЕ ТРАНЗАКЦИЙ

Сегодня фактически стандартным языком описания поведенческих моделей СБИС стал SystemC. Его развитием и документированием занимается международная группа Open SystemC Initiative (OSCI) (www.systemc.org).

Язык SystemC основан на C++ и, благодаря использованию дополнительных классов, предоставляет новые возможности – описание статических и динамических событий, параллелизм, учет временных параметров, а также поддерживает специальные типы данных, необходимые при проектировании SoC (рис.3). В принципе, SystemC позволяет описывать систему на уровне RTL, однако он наиболее эффективен и популярен при работе с поведенческими моделями, прежде всего – на уровне транзакций.

В SystemC обмен данными (командами) между отдельными блоками на уровне транзакций описывается достаточно просто (рис.4). Взаимодействие модулей происходит посредством вызовов специ-

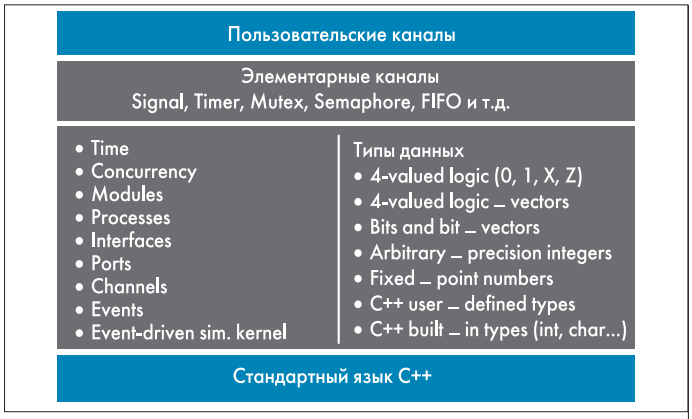

Рис.3. Структура языка SystemC

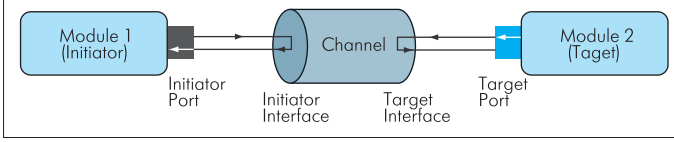

Рис.4. Модель взаимодействия между отдельными блоками на уровне транзакций в SystemC

альных функций (коммуникационных методов). Устойчивая и регулярно используемая совокупность коммуникационных методов называется *интерфейсом*. Компонент, реализующий интерфейс, называется *каналом*. Два взаимодействующих модуля подсоединяются к каналу через порты и обмениваются данными через вызовы функций, объявленных в интерфейсах и реализуемых соответствующим каналом.

Возможны два способа взаимодействия ПО и исполняемой модели аппаратуры на SystemC. Первый метод актуален при анализе архитектуры будущей SoC в целом, когда очень важно оценить загруженность шин и выбрать правильное соотношение между аппаратно и программно реализуемыми функциями. ПО пишется (моделируется) на C++ и выполняется независимо от модели аппаратуры, а связь с TL-моделью аппаратной части SoC поддерживается через коммуникационные порты. При этом все временные параметры работы аппаратуры задаются в ее SystemC-описании.

Другой способ совместного моделирования ПО и аппаратуры – это выполнение ПО на эмуляторе процессора, интегрированном в SystemC-модель системы уровня транзакций. Технически такая интеграция несложна, поскольку обычно эмуляторы процессорных ядер написаны на C/C++. Далее все – как в "традиционном" подходе: ПО компилируется и выполняется на эмуляторе процессора, в привычной разработчику программме-отладчике. Только скорость моделирования возрастает в 100 раз.

СРЕДА РАЗРАБОТКИ ВИРТУАЛЬНЫХ ПРОТОТИПОВ

Концепция виртуального прототипа не реализуема без интегрированной среды разработки, позволяющей:

- использовать существующие IP-блоки, независимо от языка или уровня абстракции их описания, и разрабатывать новые IP-блоки;
- собирать виртуальный прототип из IP-блоков;
- взаимодействовать с виртуальным прототипом;
- анализировать поведение всей системы.

Кроме того, поскольку стандартным языком описания системных моделей выступает SystemC, крайне важно, чтобы среда виртуального прототипирования избавляла разработчиков от необходимости вручную переписывать проект системы с SystemC на языки описания аппаратуры (например, Verilog или VHDL).

Все эти задачи успешно решает среда разработки **CoCentric System Studio** компании Synopsys. System Studio – это идеальная среда для разработки и аппаратно-программной верификации на системном уровне. Она включает программу моделирования на SystemC, которая позволяет использовать SystemC-модели любого происхождения на любом уровне абстракции. При необходимости на уровне транзакций легко могут быть подключены и RTL-модели на Verilog /VHDL. Для этого предназначены лучшие в своем классе программы моделирования на языках описания аппаратуры (HDL) от Synopsys – VCS и Sirocco. Можно использовать и HDL-симуляторы третьих производителей, связь с ними происходит через программируемый логический интерфейс системы, а также модели Mathlab.

Если необходимые IP-блоки отсутствуют в библиотеках, в среде CoCentric System Studio удобно разрабатывать и отлаживать новые IP-блоки на языке SystemC, причем на всех уровнях абстракции. Отметим, что при работе в CoCentric System Studio для создания новой модели на SystemC не нужно быть специалистом в C++. CoCentric System Studio сама сгенерирует шаблон на SystemC из описания интерфейса.

CoCentric System Studio оснащена мощным и удобным интерактивным графическим интерфейсом, позволяющим легко собирать виртуальный прототип из подготовленных IP-блоков (рис.5). При этом автоматически генерируется SystemC-код, соответствующий графическому представлению системы. Кроме того, CoCentric System Studio автоматически формирует и документации в формате HTML. Таким образом, один и тот же исходный код используется и для описания исполняемой спецификации-модели, и для документирования этой модели.

CoCentric System Studio позволяет управлять процессом моделирования системы. Более того, в этой среде можно контролировать моделирование и с помощью мощного TCL-интерфейса (Tool Command Language – разработка компании SUN, ставшая фактически международным стандартом). Для моделирования ПО предусмотрен программный отладчик, позволяющий в пошаговом режиме выполнять встроенную программу, в то время как CoCentric System Studio в таком же пошаговом режиме моделирует работу всей системы.

CoCentric System Studio обеспечивает широчайшие возможности по проведению анализа работы всей системы, причем независимо от того, на какой стадии находится разработка. Например, для

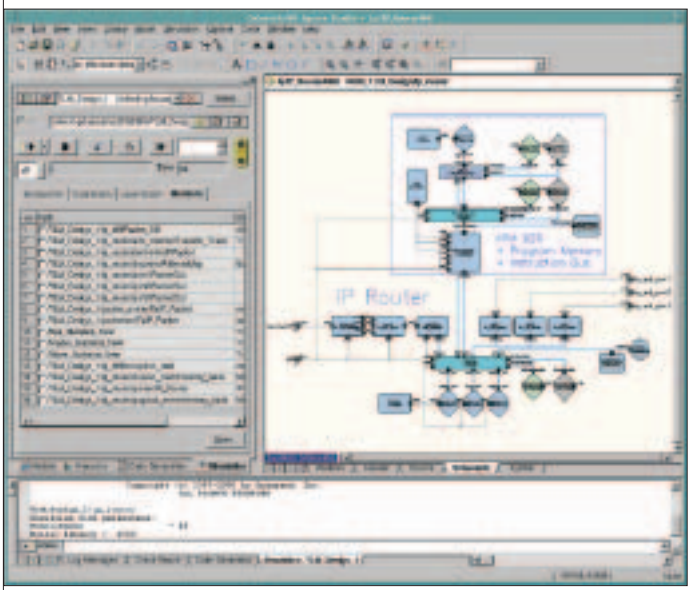

Рис.5. Создание (сборка) виртуального прототипа в System Studio

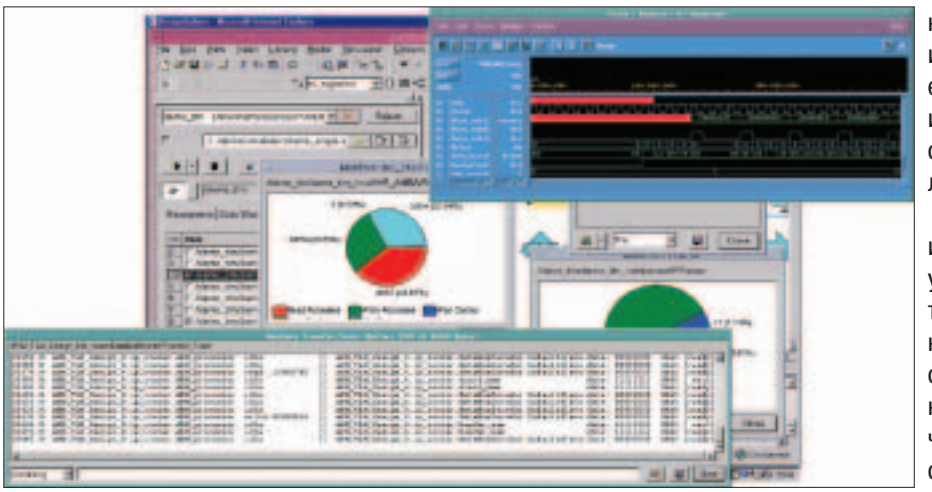

Рис.6. Отладка и анализ в System Studio

анализа сигналов предусмотрен визуализатор сигналов, позволяющий при отладке моделей системы выводить на экран все сигналы всех модулей, независимо от языков их описания (Verilog/VHDL, SystemC и др.) (рис.6).

Монитор шины позволяет отслеживать ее активность цикл за циклом. Причем если при совместной программно-аппаратной отладке нужно знать, например, как шина отрабатывает обращение к памяти и почему системе потребовалось больше циклов на отклик, чем ожидалось, то необходимо лишь активировать соответствующий монитор шины. Кроме того, специальный монитор в виде круговой диаграммы отображает статистику использования шины, позволяя анализировать причины снижения общей производительности системы.

Предоставляемые CoCentric System Studio средства мониторинга свободно подключаются к любой части проекта. Причем кроме стандартных мониторов, можно создать свои собственные, используя встроенные шаблоны. А так как в базовой для CoCentric System Studio системной библиотеке Synopsys DesignWare для SystemC-моделей предусмотрено множество встроенных мониторов, разработчик может сразу приступать к анализу создаваемой им системы.

МОДЕЛИ И БИБЛИОТЕКИ ДЛЯ ВИРТУАЛЬНОГО ПРОТОТИПИРОВАНИЯ

Не менее важный, чем среда моделирования, аспект – это наличие библиотеки IP-блоков, описанных на соответствующих уровнях абстракции. Ведь в отличие от традиционного маршрута проектирования, создание виртуального прототипа требует промежуточного шага – построения модели архитектуры на уровне транзакций, что влечет дополнительные затраты на моделирование. Решить проблему можно с помощью библиотек IP-блоков, содержащих модели уровня транзакций. Причем наиболее ценны библиотеки, содержащие поведенческие и синтезируемые модели (TL- и RTL-уровней), работа которых верифицирована друг относительно друга с точностью до цикла. Они позволяют создавать виртуальный прототип системы с уверенностью, что после реализации SoC ее поведение не будет отличаться от прототипа.

Компания Synopsys поставляет семейство именно таких библиотек, называемых **DesignWare**. Они предоставляют разработчикам тщательно оттестированные, проверенные в кремнии IP-блоки, что критически важно при промышленной разработке современных СБИС. DesignWare включают широчайший набор IP-блоков. В частности, собственно библиотека DesignWare содержит более 18,5 тыс. моделей аирфметических устройств, трактов обработки данных, процессорных ядер, элементов памяти и т.д. Библиотека DesignWare Cores включает модели шин PCI-X, PCI, USB 2.0, Ethernet и т.п. В DesignWare Star IP входят процессорные ядра многих ведущих производителей (MIPS, NEC, Infineon и др.)

Если же в распоряжении разработчика имеется IP-блок, описанный только на RTL-уровне, в ряде случаев целесообразно потратить время на создание его модели на уровне транзакций, чтобы получить выигрыш в скорости моделирования всей системы. Однако при этом необходимо удостовериться, что поведение созданной TL-модели с точностью до цикла повторяет поведение RTL-модели. Для такой проверки лучше всего подойдет уже готовый набор тестов уровня регистровых передач. С другой стороны, среда CoCentric System Studio позволяет проводить смешанное моделирование, так что даже RTL-модули, описанные на Verilog и VHDL, могут стать частью виртуального прототипа. Однако общая скорость моделирования системы при этом снизится.

В особую статью можно выделить эмуляторы процессорных ядер – ключевые элементы виртуального прототипа. Обычно они поставляются в виде кода C/C++ производителями процессоров или специализированными компаниями. Synopsys поставляет такие модели как часть семейства продуктов DesignWare. Важно отметить, что с ростом популярности SystemC все больше процессорных моделей снабжают SystemC-интерфейсом, тем более что модель из C/C++-эмулятора можно получить без труда. Например, компания ARM, известный поставщик процессорных ядер, объявила, что все ее модели с вызывами по циклам (cycle-callable models) будут снабжены SystemC-интерфейсом. Такие модели процессоров ARM92EJ-S и ARM946E-S вошли в библиотеку DesignWare. Причем SystemC-интерфейс моделей процессоров соответствует мастер-интерфейсу AHB-шины (входит в спецификацию стандартной внутрисистемной шины AMBA), так что они могут быть соединены напрямую, без адаптации протоколов обмена.

КРАТКИЕ ВЫВОДЫ

При все возрастающей сложности аппаратного и программного обеспечения SoC традиционные концепции верификации уже не справляются с задачей совместной верификации аппаратуры и ПО, поскольку при этом слишком мала скорость моделирования и высоки риски срыва выполнения проекта. Концепция виртуального прототипа устраняет эти недостатки, предоставляя быстро исполняемую, точную до цикла модель всей системы еще до начала ее детальной реализации. Моделирование на уровне транзакций увеличивает быстродействие до 100 раз.

Компания Synopsys предоставляет все необходимые составляющие для виртуального прототипирования совместной верификации аппаратной и программной частей на основе виртуального прототипа. Причем маршрут проектирования Synopsys предусматривает автоматический переход с уровня транзакций к синтезируемым моделям, напрямую из SystemC-кода. Эту возможность обеспечивают прежде всего такие продукты компании, как среда проектирования CoCentric System Studio, семейство библиотек DesignWare и компилятор CoCentric SystemC Compiler. Пожалуй, сегодня подобной возможности не предоставляют продукты ни одного другого разработчика САПР СБИС. ❍