КОМПЛЕКСНАЯ ФУНКЦИОНАЛЬНАЯ ВЕРИФИКАЦИЯ СБИС

СИСТЕМА QUESTA КОМПАНИИ MENTOR GRAPHICS

А.Лохов, А.Рабоволюк

По статистике, сегодня в семидесяти пяти процентах случаев необходимость доработки проекта СБИС и повторного запуска в производство вызвана ошибками логического функционирования. И доля возвратов, причины которых лежат в функционально-логической области, продолжает расти. Чтобы как-то переломить негативную тенденцию, необходимы новые подходы к функциональной верификации СБИС. Для решения этой задачи компания Mentor Graphics предлагает использовать комплексную методологию верификации, базирующуюся на платформе Questa.

Задача функциональной верификации – доказать, что разрабатываемое устройство будет работать именно так, как задумано разработчиком. Эта парадигма определяет развитие средств функциональной верификации. Современные проекты настолько велики, что добиться полного анализа всех возможных ситуаций, используя традиционные системы логического моделирования на базе языков VHDL и Verilog, очень трудно. Поэтому в последнее время активно развиваются новые технологии, позволяющие повысить эффективность проверки правильности логического функционирования СБИС. В список наиболее значимых входят технологии автоматизации генерации тестов (testbench automation, TBA), технологии верификации на основе анализа встроенных утверждений (assertion-based verification, ABV), технологии верификации на основе анализа тестового покрытия (coverage-driven verification, CDV), а также моделирование на уровне транзакций (transaction-level modeling, TLM). Продолжают развиваться средства моделирования и эмуляции, формальной верификации (как традиционные, так и с учетом случайного варьирования параметров в заданном диапазоне). Однако любое отдельно взятое средство или технология не в состоянии обеспечить стопроцентную проверку проекта. Решить эту задачу можно только в рамках комплексной методологии верификации.

На этапе функциональной верификации разработчик преследует две цели – во-первых, удостовериться, что в проектируемом устройстве не возникает запрещенных ситуаций и, во-вторых, – все остальные ситуации отрабатываются именно так, как он ожидает. Средства проверки, следовательно, должны обеспечить анализ возможности возникновения запрещенных состояний и правильности поведения проектируемого устройства. Кроме того, разработчику должна быть постоянно доступна информация о том, насколько далеко достижение обеих целей. Следуя этим требованиям, компания Mentor Graphics разработала собственную методологию комплексной функциональной верификации (advanced functional verification, AFV), которая объединяет все доступные средства верификации в рамках платформы Questa.

Обобщенная блок-схема процесса комплексной верификации представлена на рис.1. Средства автоматической генерации тестов (TBA), с одной стороны, дают возможность увеличить число анализируемых вариантов благодаря возможности описания в рамках одного теста различных сценариев поведения, с другой, – служат альтернативным способом описания ожидаемого поведения. Использование встроенных утверждений (ABV) позволяет обнаружить ошибки по месту именно там, где они возникают. Кроме того, использование встроенных утверждений расширяет возможности средств анализа функционального покрытия при выявлении трудно диагностируемых ситуаций. Оценить, соответствует ли работа устройства тому, что ожидает разработчик, позволяют средства моделирования/эмуляции (динамически) и формального анализа (статически). Данные моделирования и формального анализа служат для оценки функционального покрытия, на базе которой формируются рекомендации для средств автоматической генерации тестов.

Если раньше быстродействие системы логического моделирования рассматривалось в качестве основного критерия оценки эффективности, то сегодня очевидно, что более важно то, насколько быстро система позволяет спроектировать полностью функционально работоспособный проект. С этой точки зрения уменьшение числа избыточных тестов наибо-

лее важный фактор повышения эффективности. И именно интеграция разрозненных средств верификации в рамках единой комплексной методологии помогает достичь такого повышения.

БАЗОВЫЙ ПРОЦЕСС И ИНСТРУМЕНТЫ ФУНКЦИОНАЛЬНОЙ ВЕРИФИКАЦИИ

Схема на рис.2 отражает общепринятый процесс верификации. Исходный документ – функциональная спецификация. Отталкиваясь от функциональной спецификации, одна команда разработчиков занимается непосредственно устройством (правая ветвь), другая – созданием плана тестирования (левая ветвь). В центре этого процесса – сравнение результатов (традиционно, для сравнения используются системы логического моделирования). Сравнивается желаемое поведение, описанное в плане тестирования, и то, которое получено в результате моделирования проекта. Если обнаружено расхождение, ошибку надо обнаружить и исправить (здесь использование механизмов ABV позволяет улучшить работу правой ветви, благодаря более эффективной локализации ошибок). Если расхождений нет, возникает вопрос, все ли режимы работы, определенные тестовым планом, проверяются разработанной системой тестов. Использование средств анализа функционального покрытия помогает решить этот вопрос, определить, необходимо ли создание дополнительных тестов или модификация существующих. Применение средств автоматической генерации тестов позволяет создавать новые тесты и сценарии с минимальными затратами.

Эффективность организованного таким образом процесса верификации во многом определяется программными средствами. Для того, чтобы процесс верификации был эффективным, необходимо, чтобы программные инструменты использовали современные технологии, единую среду верификации, которая бы обеспечивала тесную интеграцию и программ, и различных технологий в рамках единой методологии. Такой подход лежит в основе платформы Questa. Рассмотрим подробнее технологии верификации, объединенные в рамках этой платформы.

СОЗДАНИЕ ТЕСТОВОГО ОКРУЖЕНИЯ

Процесс верификации предполагает, что должна быть сформирована тестовая инфраструктура, которая обеспечивает управление воздействиями, запись и проверку полученных результатов. Создание такого тестового окружения требует много времени и обычно находится на критическом пути проекта. Средства автоматизации тестирования помогают ускорить процесс формирования тестовой инфраструктуры. Их качество определяется тем, как быстро может быть создано тестовое окружение, позволяющее полностью проверить необходимую функциональность.

Увеличение числа различных воздействий, очевидно, улучшает качество тестирования. Возможность автоматической генерации тестов упрощает и ускоряет процесс создания новых тестов. Важно, чтобы процесс автоматической генерации был гибок, управляем и не приводил к избыточности. Сегодня эти задачи решаются в рамках механизма ограниченно случайного (constrained-random) моделирования. В тестовом окружении сразу задается множество возможных сценариев, а система моделирования выбирает допустимый сценарий для каждой проверки. Языки описания тестов, такие как SystemVerilog и SystemC, позволяют описать сценарии в терминах ограничений допустимых воздействий. Система моделирования генерирует случайные воздействия, средства контроля ограничений проверяют их допустимость, а система контроля функционального покрытия проверяет, был ли такой сценарий ранее опробован.

Другой важный способ увеличения производительности тестирования – переход на более высокий уровень абстракции, уровень транзакций. Он в первую очередь удобен для

Рис.1. Процесс комплексной функциональной верификации

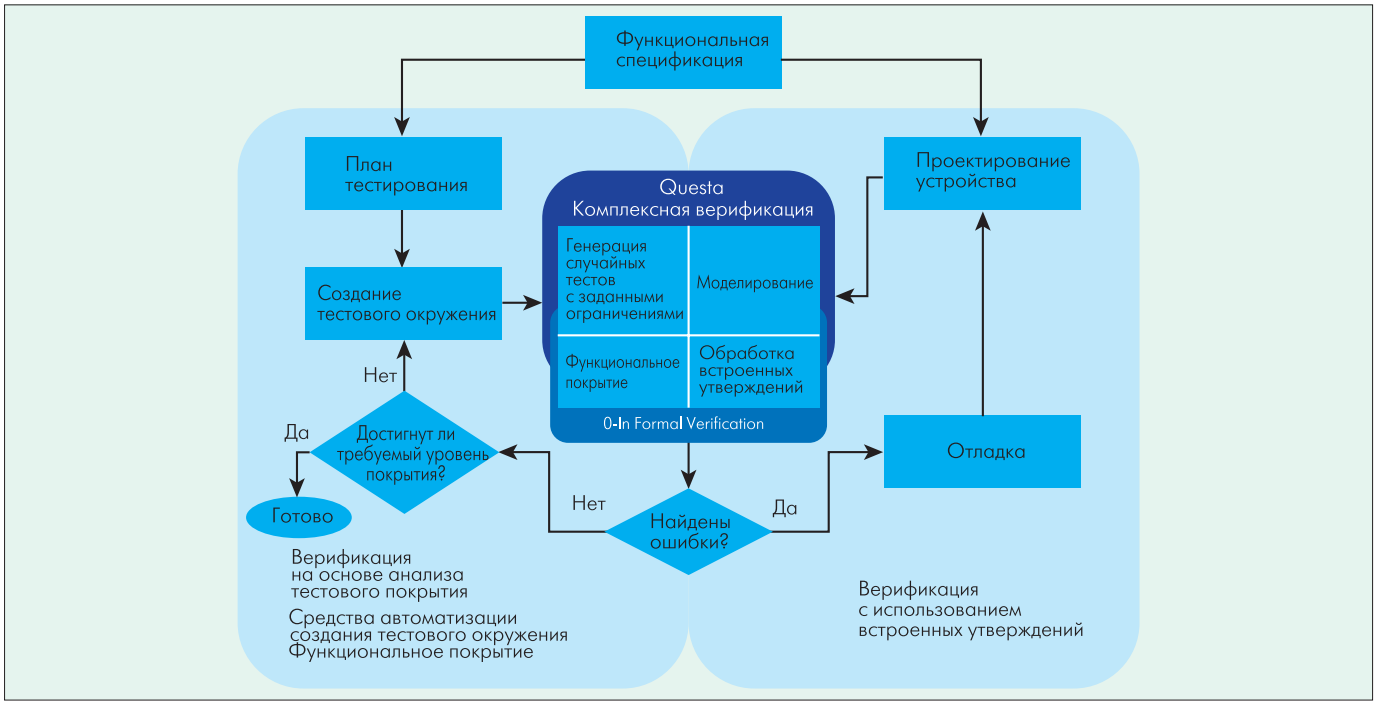

Рис.2. Общепринятая организация процесса верификации СБИС

проектов, ориентированных на поддержку стандартных шинных архитектур и интерфейсов. Тестовое окружение должно поддерживать генерацию и проверку как случайных транзакций, так и транзакций, соответствующих определенному протоколу, а также генерацию набора транзакций в рамках заданных ограничений. Используя средства ограниченно случайного моделирования, можно, например, внутри одного теста проверить, все ли режимы шины работают правильно (включая редкие ситуации, такие как пакетное чтение непосредственно за одиночной записью).

МЕХАНИЗМ ВСТРОЕННЫХ УТВЕРЖДЕНИЙ

Встроенное утверждение (assert) – это механизм, который позволяет проверить заданное разработчиком условие. Например, в блоке HDL-кода можно вставить проверку состояния внутренних переменных, которая постоянно будет выполняться в процессе моделирования. Использование встроенных утверждений позволяет увеличить наблюдаемость проекта при моделировании, быстрее и легче локализовать источник ошибки, поскольку проверки делаются непосредственно по месту. Кроме того, встроенные утверждения очень полезны в методах формальной верификации.

Встроенные утверждения отражают то, что мы ожидаем от устройства, как оно должно работать. Непосредственно в утверждении может быть определено как правильное, так и неправильное поведение компонентов устройства (RTL-блоки, интерфейсные блоки и блоки управления). Таким образом, можно контролировать и правильность поведения схемы, и возникновение запрещенных ситуаций. Для формирования утверждений применяются специальные конструкции таких языков, как, например, System Verilog Assertions (SVA) и PSL. Можно также использовать готовые библиотеки (OVL или CheckerWare Library). Язык обеспечивает возможность создания собственных уникальных проверок, в то время как использование библиотек значительно упрощает организацию сложных проверок при наличии в проекте стандартных интерфейсных блоков. Преимущества и недостатки этих способов задания очевидны. Чтобы обеспечить гибкость использования механизма встроенных утверждений, средства верификации должны поддерживать оба варианта.

Когда сегодня говорят о ABV, обычно имеют в виду дополнительные проверки в процессе моделирования. Однако более корректно ставить вопрос об использовании утверждений и в средствах моделирования, и в средствах формальной верификации. И для того, и для другого важно, что утверждение – это четкое, краткое, математически точное описание поведения или ограничений, которые должны быть соблюдены блоком. При моделировании, помимо локализации ошибок по месту (без необходимости транспортировать реакцию внутренних блоков к внешним портам), утверждения могут помочь в анализе трудно диагностируемых ситуаций и критической функциональности.

Формальная верификация – инструмент оценки соответствия RTL-описания блока заданной формальной логической модели. Статический анализ на основе математических методов теории доказательств дает возможность провести исчерпывающую проверку логики работы блока и сравнить ее с формальными требованиями.

При обнаружении несоответствия пользователь может получить расшифровку ситуации, при которой возникает рассогласование. Использование механизма утверждений позволяет встроить описание модели непосредственно

в описание блока, повысив эффективность формальной верификации.

Конструируя систему утверждений, разработчики устройства активно участвуют в процессе верификации. Кроме утверждений, описывающих внешнее поведение блока и не зависящих от способа его исполнения, разработчик может формировать утверждения, которые обеспечивают контроль внутренних состояний для конкретного варианта исполнения. В этом процессе важно понять, когда наступает момент, при котором сформированная система проверок уже достаточна для проведения верификации. На этот вопрос можно посмотреть с разных точек зрения. Например, можно считать, что достаточно гарантировать наличие встроенных проверок для ситуаций, определенных в плане тестирования. Другой подход – обеспечить плотность утверждений, при которой каждый блок с их помощью проверяется за два-три такта.

ДИНАМИЧЕСКАЯ ФОРМАЛЬНАЯ ВЕРИФИКАЦИЯ

Применение традиционной технологии формальной верификации на практике лимитировано рамками статического анализа небольших блоков устройства. Это вызвано естественными ограничениями памяти и производительности современных компьютеров. Новая технология динамической формальной верификации (dynamic formal verification), впервые появившаяся в системе 0-In Formal Verification, расширяет рамки методов формальной верификации. Особенно полезно применение динамической формальной верификации в сочетании с технологиями ABV и CDV. Анализ встроенных утверждений и обработка/накопление данных о состоянии функционального покрытия происходит непосредственно в процессе динамического моделирования. Исходными данными служат промежуточные внутренние состояния системы в процессе прохождения входного воздействия. Отталкиваясь от этих данных, методы формального анализа позволяют рассмотреть все возможные варианты работы в некоторой локальной области состояний. При этом облегчается проверка ситуаций, возникновение которых трудно обеспечить с помощью задания внешних воздействий.

Допустим, для примера, что ошибка возникает только в случае специфического сочетания внутренних состояний и при полном заполнении определенного блока FIFO(очереди). Чтобы проверить эту ситуацию с помощью моделирования, надо, во-первых, провести все циклы, необходимые для заполнения FIFO. Во-вторых, как-то предусмотреть создание необходимого сочетания внутренних состояний. Вероятность того, что подобное сочетание возникнет в результате случайной или псевдослучайной генерации тестов, очень низкая. Чтобы возникло нужное сочетание, может потребоваться слишком большой набор тестов. Если же в ситуации заполнения очереди инициировать работу средств формальной верификации, то можно проверить все возможные последующие состояния. Сообщение об ошибке будет содержать описание ситуации, в которой она возникает. Таким образом, динамическая формальная верификация может увеличить эффективность моделирования более чем в десять тысяч раз. И тесты при этом будут гораздо более простыми.

Важно учитывать некоторые особенности взаимодействия формальной верификации и моделирования. Большинство средств моделирования работают в событийной семантике, а средства формальной верификации – потактово. В результате возможно расхождение результатов. Чтобы избежать этого, в системе 0-In Formal Verification предусмотрена автоматическая перепроверка результатов формального анализа методами событийного моделирования.

В семантике последних версий языков описания встроенных утверждений System Verilog Assertions и PSL были сделаны изменения, направленные на то, чтобы гарантировать

совместимость моделирования и формальной верификации. Это открывает новые возможности для моделирования больших систем. В самом деле, если отдельный блок схемы прошел формальную верификацию в рамках системы правил, заданных встроенным утверждением, то это же утверждение при моделировании можно использовать для мониторинга входных воздействий блока. Если входные воздействия соответствуют системе ограничений, для которой была проведена формальная верификация блока, то моделирование блока можно не проводить, поскольку правильность функционирования блока в этих условиях уже была доказана. Будучи уверенным, что блоки работают правильно, можно сфокусировать внимание на поведении системы на уровне их входов/выходов и межблочных связей.

Некоторые специалисты уже сегодня ставят вопрос о возможности проведения верификации всего устройства без моделирования, используя лишь формальные методы. Теоретически можно гарантировать то, что если блок A работает правильно в определенной системе ограничений, а блок B, определяющий входные воздействия для блока A, не нарушает этих ограничений, система из блоков A и B работает корректно. На практике, правда, это сделать трудно из-за того, что система взаимосвязей между блоками слишком сложна.

В целом парадигма формальной верификации оказалась весьма плодотворной. Число разработчиков, осознавших возможности этих средств, быстро растет. В первую очередь формальные методы сегодня используются при проектировании критических блоков, в которых очень важно найти все несоответствия между описанием, заданным в утверждении, и реальным поведением блока. Анализ расхождений может выявить и устранить как ошибки проектирования блока, так и ошибки, вызванные неправильной трактовкой спецификации во встроенных утверждениях.

ВЕРИФИКАЦИЯ НА ОСНОВЕ АНАЛИЗА ФУНКЦИОНАЛЬНОГО ПОКРЫТИЯ

При использовании элементов случайного тестирования необходимо уметь собирать и анализировать информацию о том, какие сценарии уже были проверены. Работа эта обычно называется анализом функционального покрытия. Функциональное покрытие – некоторая общая оценка, характеризующая успешность процесса верификации. Использование различных метрик функционального покрытия для прогнозирования и управления процессом оптимизации формирования и применения тестовых сценариев – основной смысл технологии верификации на основе анализа функционального покрытия (CDV). Современная технология CDV использует методы случайной генерации тестов в рамках заданной системы ограничений. После сбора и анализа информации система ограничений может быть изменена (автоматически или вручную) таким образом, чтобы обеспечить проверку всех определенных разработчиком ситуаций – точек покрытия. Для задания точек могут использоваться как встроенные утверждения, так и более традиционные механизмы.

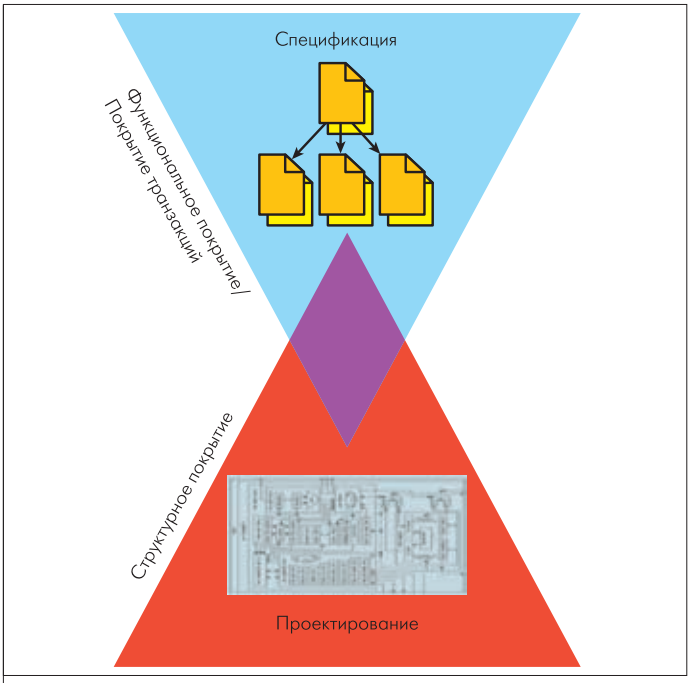

Рис.3. Обобщенная модель покрытия

Значение термина "функциональное покрытие" зависит от контекста. Базовое значение – метрика, которая показывает, какое число требуемых сценариев было протестировано. Даже таблица (например, сделанная в программе Excel) с галочками напротив выполненных тестов – это форма оценки функционального покрытия. Так же, как существуют различные виды оценки покрытия кода (активизация строк, переключателей, путей, выражений), в зависимости от того, что измеряется, есть различные подклассы функционального покрытия. Тестовое покрытие и покрытие спецификации говорят о том, были ли проверены черты, описанные в тестовом плане и спецификации, соответственно. Эти типы как раз обычно используют табличную модель. Важно, чтобы эта модель была встроена в процесс верификации.

Обобщенная модель покрытия – гораздо более тонкий подход, призванный объединить усилия инженеров-разработчиков и команду, занятую тестированием (рис.3). Снизу осуществляется процесс формирования структурного покрытия.

Структурное покрытие – усовершенствованный тип функционального покрытия, учитывающий особенности реализации. Оно ориентировано на то, чтобы проверить все трудно диагностируемые ситуации и гарантировать, что устройство никогда не делает того, что от него не ожидают. Специфические компоненты (например, схемы разрешения конфликтов, FIFO) предварительно распознаются. Поскольку требования к работе подобных компонентов обычно известны, для них используются специализированные метрики. Оценки функционального покрытия, которые идут сверху,

от команды тестирования, сообщают, все ли функции устройства проверены. Они подтверждают, что устройство работает так, как от него требуется.

Помимо самой метрики, говоря о функциональном покрытии, часто имеют в виду процесс сбора информации о работе устройства. В этом контексте принято рассматривать покрытие данных (data-oriented coverage) и покрытие сценариев (control-oriented coverage). В первом случае аккумулируется информация о состоянии в заданных точках схемы (в зависимости от проекта это могут быть логические состояния, содержание сетевых пакетов, адреса и режимы работы шины). Она служит для оценки полноты набора входных воздействий и контроля работы устройства. Сбор этой информации обычно осуществляется средствами тестового окружения. Результаты представляются в виде матрицы, значения которой показывают, сколько раз возникала каждая уникальная комбинация данных. Во втором случае сбор информации обычно реализуется с помощью встроенных утверждений. Сценарии, определенные в утверждениях, отслеживаются, данные о них сохраняются и анализируются. Средства описания сценариев достаточно гибкие, и разработчик в результате может получить разнообразную статистику – когда, как часто и в какой последовательности возникали определенные сценарии.

МОДЕЛИРОВАНИЕ НА УРОВНЕ ТРАНЗАКЦИЙ

Переход при верификации на более высокий уровень абстракции позволяет быстро провести оценку правильности функционирования устройства в целом на уровне взаимодействия крупных блоков, оставив прояснение деталей на потом. Тестовые воздействия для такой высокоуровневой верификации описываются на уровне транзакций. Обмен данными между тестовым окружением и верифицируемым проектом задается протоколом обмена, который управляет потоком транзакций. Использование мониторов протокола (модель на базе встроенных утверждений) позволяет контролировать нарушения на входах и выходах блоков, гарантируя правильность генерации и обработки транзакций между блоками. Нарушения выявляются сразу в источнике их возникновения.

Чтобы обеспечить свободный переход от высокоуровневого моделирования к детальной верификации, например на RTL-уровне, необходимо, чтобы тестовое окружение поддерживало различные представления для одних и тех же транзакций (без учета времени, потактовое, временные диаграммы). Инструменты, ответственные за корректное преобразование транзакций к нужному виду, называют транзакторами. Если средства верификации поддерживают наборы совместимых между собой транзакторов для различных уровней моделирования, то один и тот же тест может управлять и моделью на уровне транзакции, и моделью на RTL-уровне. При использовании механизма ограниченно случайной генерации один

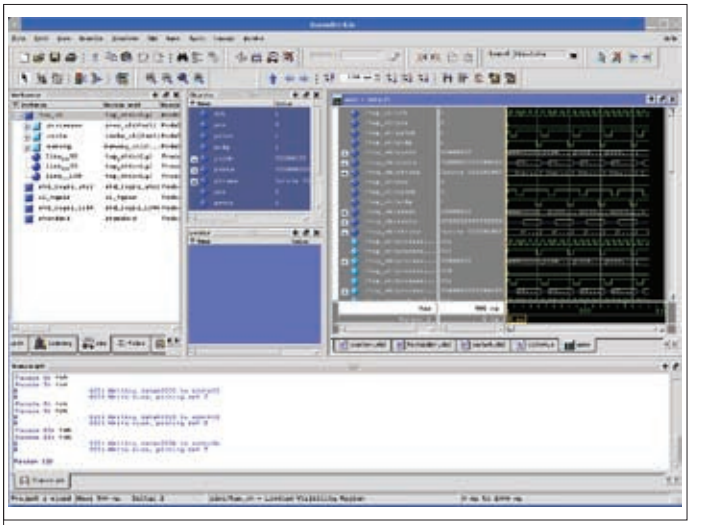

Рис.4. Система Questa компании Mentor Graphics

и тот же тест на уровне транзакций может порождать целый набор детализированных тестов, соответствующих исходному тесту. Встроенные утверждения также должны учитывать уровень абстракции. При моделировании на уровне транзакций применяется описание, ответственное за представление блока как "черного ящика". Детализация, учитывающая RTL-реализацию, используется при моделировании на RTL-уровне и для формального анализа.

ВСЕ ВМЕСТЕ В QUESTA

В системе Questa компании Mentor Graphics (рис.4) вокруг единого ядра моделирования объединены такие технологии, как HDL-моделирование, ABV, CDV, TLM, TBA. Весь процесс проектирования основан на использовании стандартных языков. Обеспечивается гибкий подход к организации процесса проектирования и верификации. В рамках единой платформы могут быть реализованы различные стили работы, система легко встраивается в любые маршруты проектирования СБИС.

Комплексная верификация на базе современных стандартов языков SystemVerylog, PSL, SystemC, Verilog, VHDL делает доступным для разработчиков, например, задание системы ограничений для средств случайной генерации тестов. Также появилась возможность подключения описаний встроенных утверждений из внешних файлов, что позволяет проводить разработку системы проверок независимо от описания устройства. Поддержка команды *cover* языков SystemVerylog и PSL и команды *covergroups* упорядочивает организацию анализа функционального покрытия. Встроенная поддержка высокоуровневого и HDL-описания в едином ядре увеличивает производительность при отладке проекта. Общий пользовательский интерфейс упрощает миграцию от одного языка к другому, делает использование нескольких языков в одном проекте прозрачным и удобным. Например, тесты, написанные на SystemVerylog и SystemC, можно легко использовать и для других языков.

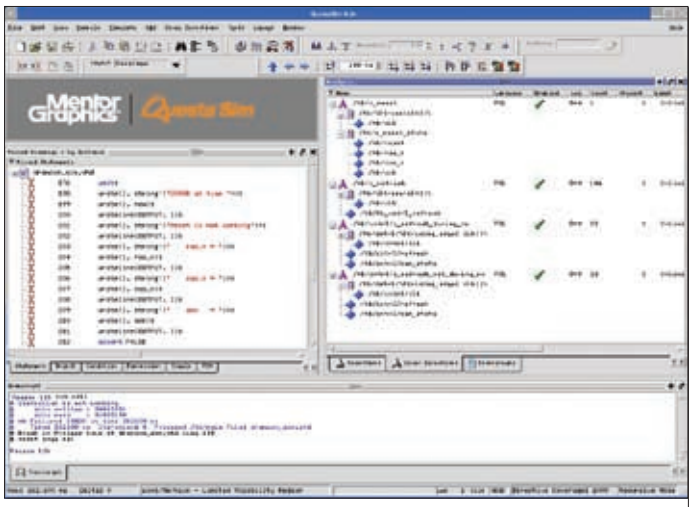

Рис.5. Анализ функционального покрытия и покрытия кода в системе Questa

Средства системы Questa позволяют автоматически создавать сложные комбинации входных воздействий, которые практически невозможно описать вручную. Сценарии для автоматической генерации тестов могут быть описаны в терминах ограничений с помощью конструкций языка SystemVerylog и библиотеки SCV(SystemC Verification). Есть возможность переиспользования и переопределения ранее подготовленных тестов путем изменения системы ограничений. Тестовое окружение системы Questa динамически реагирует на результаты анализа функционального покрытия. Система ограничений модифицируется таким образом, чтобы повышалась вероятность генерации проверочных воздействий для еще непродиагностированных ситуаций. Случайный характер генерации тестов позволяет также проверить множество ситуаций, не предусмотренных разработчиками.

Платформа Questa поддерживает описания встроенных утверждений с помощью стандартных языков PSL и SystemVerilog. Встроенные утверждения интегрированы со средствами автоматической генерации тестов, анализа функционального покрытия, моделирования, формальной верификации и эмуляции. При обнаружении нарушения во встроенном утверждении средства отладки позволяют быстро найти его причину, понять, какие сигналы на входе инициировали ошибку. На реальных примерах Questa ABV сокращает время анализа и устранения дефекта с двух дней до двух часов. Использование встроенных утверждений и специальных команд языков SystemVerylog и PSL (например, команды *coverpoints*) в результате позволяет получить оценки всех видов покрытия, включая покрытие кода, структурное, функциональное покрытие и покрытие транзакций (рис.5).

Библиотека CheckerWare, встроенная в Questa ABV, расширяет сферу применения встроенных утверждений благодаря возможности использования стандартных IP-

решений для верификации. Кроме проверки нарушений во время моделирования, средства CheckerWare и встроенные в библиотеку мониторы протоколов позволяют собирать статистические данные для анализа структурного покрытия и служат в качестве базовой модели для формальной верификации. Библиотека поддерживает большое количество стандартных IP. Она совместима с такими языками описания утверждений, как OVL, PSL и SystemVerolog Assertion. Компоненты CheckerWare легко добавляются в проект и могут без модификации использоваться во всех инструментах (включая моделирование, эмуляцию и формальную функциональную верификацию). Многократно проверенные стандартные IP-решения уменьшают риски и позволяют инженерам с минимальными затратами воспользоваться всеми преимуществами ABV-технологии.

Средства формальной верификации представлены в платформе Questa системой 0-In Formal Verification. Она поддерживает как традиционный статический анализ, так и динамическую формальную верификацию с использованием технологий ABV и CDV. Статические формальные методы обычно используется для анализа отдельных блоков, а динамические – при моделировании всего кристалла. Включение компонентов CheckerWare при работе с системой 0-In Formal Verification облегчает использование средств формальной верификации и расширяет их возможности.

Использование встроенных средств моделирования на уровне транзакций Questa TLM позволяет ускорить тестирование в тысячу раз за счет того, что на начальных этапах многие низкоуровневые детали могут быть опущены. Для реализации методологии TLM в систему были добавлены такие новые возможности, как создание протоколов и транзакторов. Средства TLM также служат базой для формирования библиотек верифицированных модулей, использование которых упрощает и ускоряет создание тестового окружения.

Итак, некоторые итоговые тезисы. Задача функциональной верификации СБИС – показать, что поведение устройства соответствует ожиданиям разработчика. Эффективность средств верификации определяется не тем, как быстро работает отдельная программа моделирования, а тем, насколько быстро они позволяют достичь полной функциональной проверки проекта. Лучший способ повышения эффективности верификации – интеграция всех современных технологий в рамках комплексной методологи. Использование стандартных языков способствует распространению комплексной методологии и уменьшает риски инвестиций в средства верификации. Платформа Questa органично объединяет современные средства и технологии верификации и позволяет на практике применить комплексную методологию верификации.