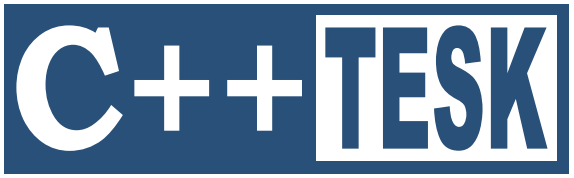

C++TESK Hardware Edition: **Основные возможности**

Версия 1.0, 02/06/2011

Инструмент C++TESK Hardware Edition входит в состав набора инструментов C++TESK Testing ToolKit, который можно скачать на странице http://forge.ispras.ru/projects/cpptesk-toolkit.

Вопросы по использованию C++TESK Hardware Edition и C++TESK Testing ToolKit в целом, а также описание обнаруженных проблем в инструментах и документации к ним отправляйте по адресу cpptesk-support@ispras.ru. Для этого также можно использовать форум http://hw-forum.ispras.ru.

Введение

В документе описываются основные возможности инструмента C++TESK Hardware Edition (далее C++TESK), предназначенного для автоматизированной разработки тестовых систем для HDL-моделей[1] аппаратуры с использованием *языка программирования C++*.

Тестовой системой называется специальная программа, которая путем подачи на *целевую систему* воздействий и анализа выдаваемых ей результатов оценивает ее функциональную корректность. Воздействия на тестируемую модель также называются *стимулами*, а результаты ее работы — *реакциями*. Тестовая система решает три основные задачи: (1) генерация стимулов, (2) *проверка правильности реакций* и (3) *оценка полноты тестирования*.

Более подробная документация по C++TESK входит в поставку инструмента и может быть скачана на странице http://forge.ispras.ru/projects/cpptesk-toolkit/files. По вопросам проведения тренинг-курса по инструменту обращайтесь по адресу cpptesk-support@ispras.ru.

Почему C++?

Разработка компонентов тестовой системы в C++TESK осуществляется на языке программирования C++. Важно отметить, что в инструменте используется «*чистый*» C++ без языковых расширений. Ядро инструмента представляет собой *библиотеку* классов, шаблонов и макросов на языке C++, с помощью которых можно определять различные сущности тестовой системы.

Почему для инструмента тестирования HDL-моделей аппаратуры был выбран именно C++?

Во-первых, C++ является широко распространенным языком программирования знакомым большинству инженеров.

Как правило, в процессе проектирования аппаратуры (в частности, микропроцессоров) помимо HDL-модели создается высокоуровневая модель на языке C/C++ (так называемый *симулятор*), которая может быть использована в качестве эталона при тестировании. Использование для разработки тестовой системы того же самого языка существенно упрощает использование симулятора (или его отдельных компонентов) для целей тестирования, сокращая трудозатраты на создание тестов.

Третьей причиной использования C++ является простая интеграция C++-модулей с HDL-симуляторами аппаратуры. В настоящее время имеется несколько стандартных интерфейсов, предназначенных для этих целей (прежде всего, VPI — Verilog Procedural Interface и DPI — Direct Programming Interface).

Проверка правильности реакций

Автоматическая проверка правильности поведения тестируемой системы имеет ключевое значение для автоматизации тестирования. Каждый раз, когда тестовая система получает реакцию от тестируемой HDL-модели, она должна определить, правильная эта реакция или нет. Чтобы иметь возможность это сделать, необходимы два компонента: (1) *эталонная модель* и (2) *адаптер эталонной модели*.

Эталонная модель работает совместно с тестируемой HDL-моделью, обрабатывая ту же самую последовательность стимулов. Результатом работы эталонной модели является множество эталонных реакций. При получении реакции от тестируемой модели она с

[1] HDL (Hardware Description Language) — класс языков, используемых для описания аппаратуры. Самыми известными примерами таких языков являются Verilog и VHDL.

помощью адаптера преобразуется в формат эталонной модели и сопоставляется одной из эталонных реакций, после чего происходит их сравнение пары реакций.

C++TESK предоставляет средства создания эталонных моделей и их адаптеров. Следует подчеркнуть, что инструмент поддерживает разработку эталонных моделей разного уровня детализации (от абстрактных функциональных моделей до моделей с потактовой точностью) и позволяет использовать в качестве эталонов уже разработанные симуляторы и их компоненты. Часть адаптера генерируется автоматически на основе анализа интерфейса тестируемой HDL-модели.

Генерация стимулов

От того, как генерируются стимулы, напрямую зависит качество тестирования. В C++TESK применяется подход к автоматизации разработки генераторов стимулов на основе *тестовых сценариев*. Тестовым сценарием называется высокоуровневая спецификация теста, в которой определяются доступные для тестирования стимулы и функция вычисления *обобщенного состояния* целевой системы. Тестовая последовательность строится путем интерпретации тестового сценария *обходчиком* — библиотечным компонентом C++TESK.

В текущей реализации инструмента имеются два обходчика: `fsm` и `rnd`. Первый из них осуществляет обход *графа обобщенных состояний*, заданного в неявной форме в тестовом сценарии. Критерием завершения тестирования в этом случае является посещение всех обобщенных состояний достижимых из начального. Обходчик `rnd` строит тестовую последовательность *случайным* образом — на каждом шаге выбирается случайный стимул или набор стимулов из числа описанных в тестовом сценарии. Тестирование прекращается при выполнении указанного при запуске числа шагов. Важно отметить, что обходчики имеют одинаковый интерфейс, что позволяет использовать разные обходчики для выполнения одного и того же тестового сценария.

Оценка полноты тестирования

Для оценки полноты (степени завершенности) тестирования в C++TESK реализованы средства описания *тестового покрытия*. Структура тестового покрытия описывается явным перечислением возможных при тестировании ситуаций (*тестовых ситуаций*). Для описания сложных ситуаций имеются средства композиции тестовых покрытий: сложная ситуация представляется в форме совокупности нескольких более простых ситуаций.

После выполнения теста можно построить отчет о проведенном тестировании, который, помимо обнаруженных ошибок, содержит информацию об уровне достигнутого тестового покрытия — какие тестовые ситуации были покрыты во время тестирования, а какие нет. Отчеты о тестировании позволяют оценить качество тестов и сфокусировать усилия на проверке еще не охваченных аспектов функционирования целевой системы.

Распараллеливание выполнения тестов

В виду большой сложности современной аппаратуры и огромного объема тестов для HDL-моделей аппаратуры, выполнение тестов может занимать значительное время. Для того чтобы сократить время тестирования и, соответственно, время, затрачиваемое на обнаружение ошибок, C++TESK имеет средства автоматического распараллеливания процесса тестирования на компьютерном кластере. Проведенные эксперименты показывают что, при использовании 100 компьютеров время выполнения тестов сокращается в 80-90 раз.

Следует отметить, что распараллеливание осуществляется прозрачно для разработчика тестовой системы. Другими словами, любая тестовая система, разработанная с помощью C++TESK, может быть запущена как на одном компьютере, так и на нескольких компьютерах, объединенных в сеть.