# Сравнительный анализ современных технологий разработки тестов для моделей аппаратного обеспечения

*Я.С. Губенко, А.С. Камкин, М.М. Чупилко*
*{jacob, kamkin, chupilko}@ispras.ru*

**Аннотация**. Работа представляет собой сравнительный анализ современных подходов к разработке функциональных тестов для моделей аппаратуры: AVM (Advanced Verification Methodology) от компании Mentor Graphics, OVM (Open Verification Methodology) — совместной разработки Mentor Graphics и Cadence Design Systems — и технологии UniTESK (Unified TEsting and Specification tool Kit), разработанной в Институте системного программирования РАН. В статье анализируются сильные и слабые стороны различных подходов, сопоставляются архитектуры тестовых систем, даются рекомендации по развитию технологии UniTESK и ее унификации с методологией OVM, набирающей все большее распространение и претендующей на то, чтобы стать стандартом в области тестирования моделей аппаратного обеспечения.

## 1. Введение

Увеличение сложности аппаратного обеспечения неизбежно влечет за собой развитие *технологий разработки тестов*. В настоящее время сложность аппаратуры достигла такого уровня, что ее тестирование немыслимо без применения *методов автоматизации*. Такие методы позволяют разрабатывать тестовые системы, которые автоматически генерируют тестовые последовательности, оценивают правильность поведения системы и собирают информацию о достигнутом уровне тестового покрытия. Однако следует понимать, что автоматизация это далеко не все, что требуется от современной технологии разработки тестов.

В условиях постоянного изменения требований и непрерывной доработки проектов огромное значение приобретают такие характеристики технологии, как *возможность повторного использования тестов* и *возможность создания тестов, устойчивых к изменениям реализации*. Существующие подходы к построению тестовых систем, такие как AVM (Advanced Verification Methodology), URM (Universal Reuse Methodology), OVM (Open Verification Methodology) и UniTESK (Unified TEsting and Specification tool Kit), в той или иной степени решают обозначенные задачи, но многообразие



существующих подходов усложняет взаимодействие между разными группами разработчиков.

*Стандартизация технологий разработки тестов* является необходимым шагом в развитии полупроводниковой индустрии. Появление стандартов создаст предпосылки для отчуждения тестов, что окажет стимулирующее воздействие на рынок готовых компонентов (IPs, Intellectual Property Cores). Если тесты к компоненту аппаратного обеспечения разработаны с использованием общепринятого подхода, сторонние инженеры могут без труда разобраться в них и при необходимости дополнить.

В последнее время в области тестирования моделей аппаратуры все большее распространение получает технология OVM, совместно разработанная компаниями Mentor Graphics и Cadence Design Systems. Предшественниками OVM являются два основных подхода: AVM (от Mentor Graphics) и URM (от Cadence Design Systems). В данной работе технология OVM вместе со своими предшественниками сравнивается с технологией UniTESK, разработанной в Институте системного программирования РАН. В статье анализируются сильные и слабые стороны указанных подходов, сопоставляются архитектуры тестовых систем, даются рекомендации по развитию инструментов UniTESK и их унификации с подходом OVM.

Статья состоит из семи разделов, включая введение и заключение. Во втором разделе вводятся основные понятия, используемые в различных технологиях. Третий, четвертый и пятый разделы посвящены обзору технологий AVM, OVM и UniTESK соответственно. В этих разделах описываются основные принципы указанных подходов и архитектуры тестовых систем. Шестой раздел содержит сравнительный анализ рассматриваемых технологий. В седьмом разделе делается заключение, в котором даются рекомендации по развитию технологии UniTESK.

## 2. Основные понятия

Под *моделями аппаратного обеспечения* в данной статье понимаются программные модели цифровых устройств, описанные на специальных языках. Такие языки можно разбить на две разновидности:

- *языки описания аппаратуры* (*HDLs*, *Hardware Description Languages*), с помощью которых разрабатываются *модели уровня регистровых передач* (*RTL*, *Register Transfer Level*). Среди таких языков можно выделить Verilog [1] и VHDL [2];
- *языки проектирования системного уровня* (*system-level design languages*), позволяющие описывать аппаратную систему на более высоком уровне абстракции и поддерживающие разработку программной части системы. К таким языкам относятся SystemC [3] и SystemVerilog [4].

В дальнейшем для удобства будем называть *языками описания аппаратуры* или *HDL-языками* обе указанные разновидности языков.

Рассматриваемые в статье технологии нацелены на автоматизацию разработки тестов для моделей аппаратного обеспечения и создание тестовых систем с высоким уровнем повторного использования и устойчивых к изменениям реализации.

Перед описанием технологий AVM, OVM и UniTESK (технология URM, упомянутоя во введении, не описывается в виду почти полного отсутствия документации по ней) отметим следующее. В названии подходов AVM и OVM содержится слово «*методология*», в то время как UniTESK, по мнению ее создателей, является *технологией*. В рамках данной статьи мы не различаем эти два термина и понимаем под ними совокупность принципов, методов и инструментов для разработки набора тестов.

## *3. Обзор технологии AVM*

Технология AVM (от англ. *advanced verification methodology — передовая методология верификации*) была создана в компании Mentor Graphics и распространяется под лицензией Apache 2.0. На настоящий момент она никак не стандартизирована и вся информация по ней находится в «книге рецептов» [5] и пользовательской инструкции по эксплуатации.

### 3.1. Основные принципы AVM

С помощью технологии AVM реализуется возможность разработки сложных тестовых систем на SystemVerilog или SystemC. Основным средством AVM являются библиотеки классов, разработанные на этих языках. При разработке тестовых систем используются следующие базовые принципы: *объектно-ориентированный подход*, *моделирование на уровне транзакций* (*TLM*, *Transaction-Level Modeling*) и *генерация случайных стимулов на основе ограничений* (*constraint-random generation*).

TLM — это подход к моделированию пересылки данных в цифровых системах, при котором организация связи между модулями системы отделена от их реализации. Под *транзакцией* понимается единичная пересылка управляющей информации или данных между двумя модулями. Например, в проектах, использующих шины передачи данных, чтение и запись в шину могут быть описаны транзакциями; в системах с коммутацией пакетов транзакцией является посылка или прием пакета.

Использование объектно-ориентированного подхода и моделирования на уровне транзакций позволяет технологии AVM добиться высоких показателей повторного использования тестов. Использование механизмов наследования и перегрузки методов позволяет переопределять поведение компонента тестовой системы. За счет этого сокращается время, затрачиваемое на разработку и отладку тестов.



### 3.2. Архитектура тестовой системы AVM

Тестовые системы, создаваемые с использованием технологии AVM, должны быть разбиты на несколько слоев (см. Рис. 1). Самый нижний уровень — это *RTL-модель тестируемого устройства* (*DUT*, *Design Under Test*). Над ним находится прослойка *транзакторов* (*transactors*) — компонентов тестовой системы, осуществляющих преобразование потока входных транзакций во входные сигналы тестируемой модели и наоборот преобразование выходных сигналов тестируемой модели в поток выходных транзакций.

Примерами транзакторов являются:

- *драйвер* (*driver*) — конвертирует поток транзакций во входные сигналы тестируемой модели;
- *ответчик* (*responder*) — отвечает на запросы со стороны тестируемой модели относительно подачи дополнительных данных;
- *монитор* (*monitor*) — осуществляет наблюдение за выходами тестируемой модели, не оказывая на нее влияния.

Следующим слоем является так называемый *слой операций* (*operational layer*). Компоненты этого уровня образуют тестовое окружение необходимое тестируемой модели. К ним относятся:

- *генератор стимулов* (*stimulus generator*) — создает поток входных транзакций (стимулов) на тестируемую модель. Для генерации стимулов используется подход на основе разрешения ограничений с использованием рандомизации (CRV, Constraint-Random Verification). Связь генератора с тестируемой моделью осуществляется через драйвер;
- *главный модуль* (*master*) — элемент окружения тестируемой модели, инициирующий взаимодействие с ней. Главный модуль посылает последовательности транзакций (случайные или направленные на достижение какой-либо ситуации) и может использовать обратную связь для определения своих следующих действий. Связь главного модуля с тестируемой моделью осуществляется через драйвер;
- *подчиненный модуль* (*slave*) — как и главный модуль моделирует окружение тестируемой модели, но, в отличие от него, является пассивным компонентом, отвечающим на запросы тестируемой модели. Связь подчиненного модуля с тестируемой моделью осуществляется через ответчик.

Далее следует *слой анализа* (*analysis layer*), компоненты которого проверяют корректность поведения тестируемой модели, оценивают полноту тестирования и выполняют некоторые другие функции. Основными компонентами этого уровня являются:

- *компонент проверки* (*scoreboard*) — проверяет корректность поведения тестируемой модели в ответ на входные транзакции;

- *сборщик покрытия* (*coverage collector*) — оценивает полноту тестирования путем подсчета числа событий заданных типов, возникающих в процессе тестирования.

На самом верху находится *слой управления*, включающий в себя один единственный компонент — *контроллер теста* (*test controller*), который осуществляет координацию работы других компонентов тестовой системы.

Рис. 1 иллюстрирует многослойную структуру тестовой системы. Каждый прямоугольник соответствует определенному слою. В левой части указано его название, в правой — основные компоненты тестовой системы, образующие данный слой. Стрелки показывают взаимодействие между слоями.

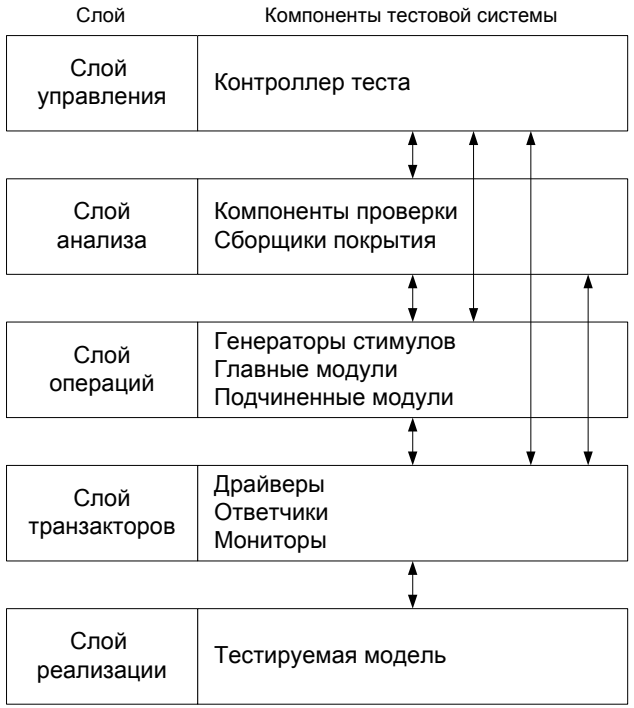


*Рис. 1. Слои тестовой системы AVM*

## *4. Обзор технологии OVM*

Технология OVM (от англ. *open verification methodology — открытая методология верификации*) является совместной разработкой компаний Mentor Graphics и Cadence Design Systems. OVM представляет собой первый *открытый* промышленный метод разработки тестов на основе языка SystemVerilog [6]. Предшественниками OVM являются подходы AVM (от Mentor Graphics) и URM (от Cadence Design Systems). Технология OVM пока



не стандартизирована, но является главным претендентом на звание стандарта.

### 4.1. Основные принципы OVM

Базовые принципы технологии OVM во многом схожи с AVM. Основным средством OVM является библиотека классов на языке SystemVerilog (в дальнейшем планируется поддержка SystemC), позволяющая создавать модульные тестовые системы, допускающие многократное использование. Одним из достоинств технологии является *нацеленность на тестовое покрытие* (*CDV*, *Coverage-Driven Verification*), что позволяет четко задавать цели тестирования и лучше управлять процессом разработки тестов.

### 4.2. Архитектура тестовой системы OVM

В рамках технологии OVM тестовая система собирается из специальных блоков, называемых *OVC* (*Open Verification Component*). Каждый блок инкапсулирует тестовое окружение для отдельного модуля аппаратуры и состоит из полного набора компонентов, обеспечивающих генерацию стимулов, проверку правильности поведения и оценку полноты тестирования. Если тестируемая модель аппаратного обеспечения получается путем интеграции нескольких модулей, тестовая система для нее строится путем объединения соответствующих OVC-блоков и построения над ними специального синхронизирующего контроллера (virtual sequencer).

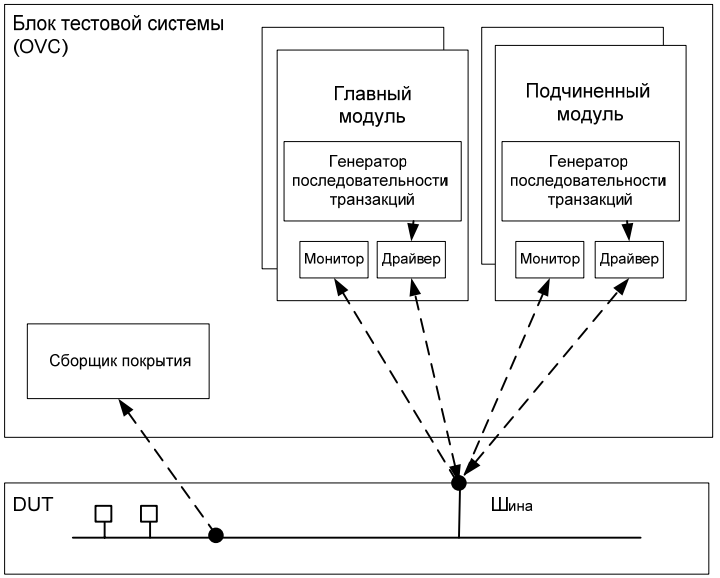


*Рис. 2. Блок тестовой системы*

Структура OVC-блоков практически полностью соответствует архитектуре тестовой системы AVM. Пример блока тестовой системы приведен на Рис. 2.

## 5. Обзор технологии UniTESK

Технология UniTESK (от англ. *unified specification and testing tool kit — унифицированный инструментарий для спецификации и тестирования*) [7] разработана в Институте системного программирования РАН. В отличие от рассмотренных ранее технологий AVM и OVM, созданных в коммерческих организациях, UniTESK является академической разработкой, которая, тем не менее, основана на реальном опыте тестирования программного и аппаратного обеспечения.

### 5.1. Основные принципы UniTESK

При разработке тестовых систем по технологии UniTESK используются следующие базовые принципы:

- определенность набора компонентов с ясным разделением функций и четкими интерфейсами;
- использование формальных спецификаций в форме пред- и постусловий интерфейсных операций и инвариантов типов данных для автоматической генерации тестовых оракулов (компонентов проверки);
- применение конечно-автоматных моделей для построения последовательностей тестовых воздействий (стимулов).

В качестве основного средства тестирования моделей аппаратного обеспечения используется инструмент CTESK, использующий для разработки тестовой системы язык SeC (Specification extension of C) — расширение языка программирования C.

### 5.2. Архитектура тестовой системы UniTESK

Тестовая система UniTESK состоит из следующих основных компонентов (см. Рис. 3):

- *обходчик* является библиотечным компонентом тестовой системы. В его основе лежит алгоритм обхода графа состояний конечного автомата, моделирующего целевую систему на некотором уровне абстракции;
- *итератор тестовых воздействий* работает под управлением обходчика и предназначен для перебора в каждом достижимом состоянии конечного автомата допустимых тестовых воздействий. Он автоматически генерируется из тестового сценария, который представляет собой описание конечно-автоматной модели тестируемой системы;
- *тестовый оракул* оценивает правильность поведения тестируемой системы в ответ на единичное тестовое воздействие. Он автоматически



генерируется на основе формальных спецификаций, описывающих требования к системе в виде пред- и постусловий интерфейсных операций и инвариантов типов данных;

- *медиатор* связывает формальные спецификации с конкретной реализацией тестируемой системы. Медиатор преобразует единичное тестовое воздействие из спецификационного представления в реализационное, а полученную в ответ реакцию — из реализационного представления в спецификационное. Также медиатор синхронизирует состояние спецификации с состоянием тестируемой системы;
- *трасса теста* отражает события, происходящие в процессе тестирования. На основе трассы можно автоматически генерировать различные отчеты, помогающие в анализе результатов тестирования.

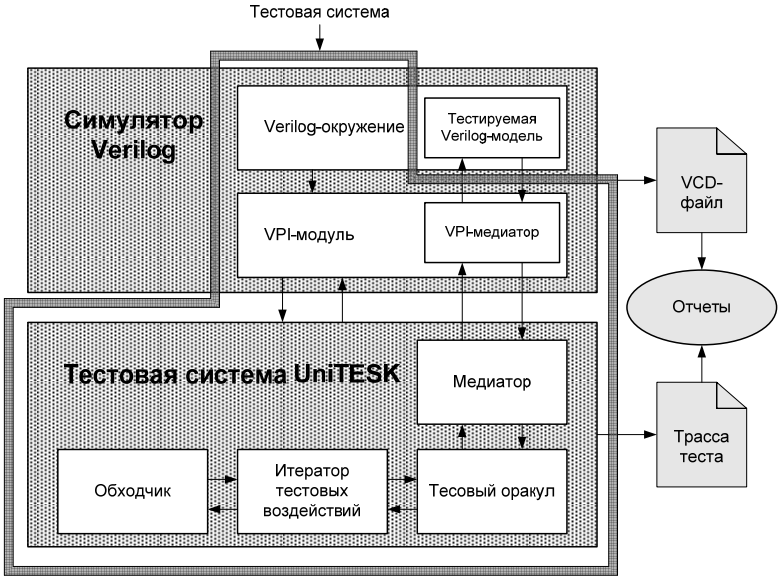


*Рис. 3. Архитектура тестовой системы UniTESK*

Помимо основных компонентов в тестовой системе выделяются части, отвечающие за интеграцию с конкретным HDL-языком. Например, для языка Verilog такими компонентами являются *VPI*[1]*-модуль* (включая *VPI-медиатор*) и *Verilog-окружение*.

[1] VPI (Verilog Procedural Interface) — стандартный интерфейс, предназначенный для вызова из Verilog-модулей функций, написанных на языке программирования C и других языках программирования.

## 6. Сравнение технологий разработки тестов

Сравним технологии AVM, OVM и UniTESK. Подходы AVM и OVM разработаны коммерческими организациями с учетом большого опыта тестирования моделей аппаратного обеспечения. Основной акцент в этих технологиях сделан на повторном использовании тестов, которое достигается средствами объектно-ориентированного программирования и моделирования на уровне транзакций. Дополнительно к этому в подходе OVM заложен механизм построения тестовых систем из готовых блоков, отвечающих за тестирование отдельных модулей аппаратной системы. Эта концепция хорошо отражает тенденции современной индустрии, когда основная работа сосредотачивается не на проектировании принципиально нового аппаратного обеспечения, а на интеграции аппаратуры из готовых модулей.

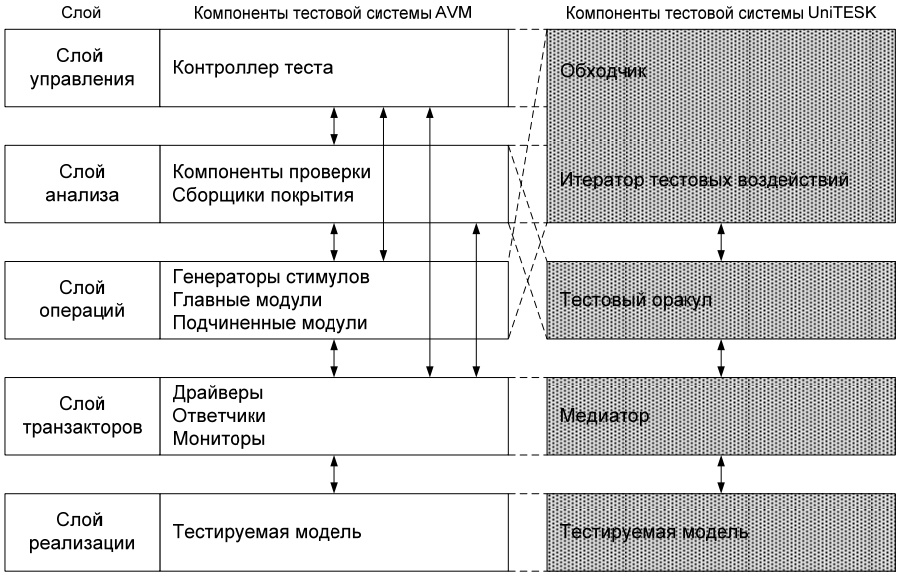


*Рис. 4. Сопоставление архитектур тестовых систем*

В технологии UniTESK, являющейся академической разработкой, основной упор сделан на автоматизацию создания тестов. В UniTESK используются формальные спецификации поведения в форме пред- и постусловий операций и инвариантов типов данных, а также конечно-автоматные модели для построения последовательностей стимулов. Все это позволяет разрабатывать высококачественные тесты, тщательно проверяющие функциональность тестируемой модели. Из плюсов UniTESK можно также отметить возможность разработки спецификаций с потактовой точностью на основе пред- и постусловий стадий операций. Однако, что касается средств



повторного использования тестов, в технологии UniTESK они менее развиты по сравнению с AVM и OVM. Во многом это связано с тем, что инструмент CTESK, используемый для тестирования моделей аппаратного обеспечения, основан на языке программирования C. Реализация основных принципов UniTESK для языков SystemVerilog и SystemC позволила бы улучшить показатели повторного использования тестов.

Сопоставим архитектуру тестовой системы AVM (она такая же, как у OVC-блока) и тестовой системы UniTESK (см. Рис. 4).

Слой транзакторов в тестовой системе AVM приблизительно соответствует слою медиаторов в тестовой системе UniTESK. Соответствие в вышележащих слоях установить сложнее. Генератору стимулов AVM соответствует итератор тестовых воздействий UniTESK (в паре с обходчиком), компонентам проверки — тестовый оракул. Наиболее близким компонентом UniTESK, соответствующим контроллеру тестов AVM, является обходчик. Заметим, что для некоторых компонентов AVM нет явных соответствий в технологии UniTESK, например, там нет таких сущностей, как главные и подчиненные модули.

## 7. Заключение

Мы считаем, что в области разработки тестов для моделей аппаратного обеспечения в скором времени должны появиться стандарты. Главным претендентом на звание стандарта является технология OVM, которая продвигается такими крупными компаниями, как Mentors Graphics и Cadence Design Systems. Эта технология основана на хорошо зарекомендовавших себя подходах AVM и URM, поэтому ее начинают активно использовать в индустрии. Мы выступаем за унификацию технологии UniTESK (и других методов автоматизации разработки тестов для моделей аппаратного обеспечения) с подходом OVM. Лучше использовать один подход, который будет понятен разным группам разработчиков.

Под унификацией здесь понимается следующее. Основные принципы, на которых основана UniTESK, прежде всего, использование формальных спецификаций в форме пред- и постусловий для описания поведения системы и применение конечно-автоматных моделей для генерации тестовых последовательностей, хорошо зарекомендовали себя на практике. Ничто не мешает реализовать эти принципы в терминах технологии OVM, на основе библиотеки базовых классов. Это позволит создавать тесты по технологии UniTESK, но которые при этом приобретают новое качество — их можно отчуждать, повторно использовать при построении более крупных тестовых систем и т.д.

**Литература**


[1] IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-1987.

[2] IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language. IEEE Std 1364-1995.

[3] IEEE Standard SystemC Language Reference Manual. IEEE Std 1666-2005.

[4] *IEEE Standard for SystemVerilog – Unified Hardware Design, Specification, and Verification Language*. IEEE Std 1800-2005.

[5] Adam Rose, Tom Fitzpatrick, Dave Rich, Harry Foster. Advanced Verification Methodology Cookbook. Mentor Graphics Corporation, 2008. (http://www.mentor.com/go/cookbook)

[6] OVM Whitepaper. Mentor Graphics Corporation, Cadence, 2007. (http://www.ovmworld.org)

[7] В.П. Иванников, А.С. Камкин, В.В. Кулямин, А.К. Петренко. Применение технологии UniTESK для функционального тестирования моделей аппаратного обеспечения. Препринт ИСП РАН, 2005.(http://citforum.ru/SE/testing/unitesk_hard/)