КОМПЬЮТЕРНАЯ ИНЖЕНЕРИЯ И ТЕХНИЧЕСКАЯ ДИАГНОСТИКА

УДК 681.324:519.613

# МЕТОДЫ РАНЖИРОВАНИЯ ТЕСТОВ ДЛЯ МОДЕЛЕЙ ЦИФРОВЫХ СИСТЕМ ПО ГЕТЕРОГЕННЫМ МЕТРИКАМ ПОКРЫТИЯ

*ЗАЙЧЕНКО С.А., АЛЕКСАНДРОВ В.И., БЕЛОУС В.В., БЕРЕЗИН Н.П.*

Рассматривается проблема неэффективности процесса автоматизированного регрессионного тестирования моделей цифровых устройств на этапе функциональной верификации. Предлагается подход к уменьшению стоимости вычислительных затрат на выполнение наборов регрессионных тестов, опирающийся на устранение избыточности тестовой нагрузки, выявляемой сравнением времени работы индивидуальных тестов и степенью их влияния на совокупность автоматических и функциональных метрик покрытия.

## 1. Введение

Задача функциональной верификации состоит в демонстрации, что разрабатываемое устройство функционирует в соответствии с требованиями спецификации и намерениями разработчика. Современные проекты настолько велики, что добиться полного анализа всех возможных ситуаций, используя традиционные системы логического моделирования на базе языков VHDL и Verilog, очень трудно. В последнее время активно развиваются новые технологии, позволяющие повысить эффективность проверки корректности логического функционирования СБИС. В список наиболее значимых входят технологии автоматизации генерации тестов (testbench automation, TBA), технологии верификации на основе анализа асерций (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). Любая отдельно взятая технология не в состоянии на 100% обеспечить валидацию проекта. Решить эту задачу можно только в рамках комплексной методологии верификации.

При использовании псевдослучайной генерации стимулов необходимо организовать сбор и анализ информации о том, какие сценарии уже были проверены. Корректирующую обратную связь для генераторов стимулов может предоставить система анализа функционального покрытия. Функциональное покрытие – некоторая общая оценка, характеризующая завершенность процесса верификации относительно заданного плана тестирования. Основной смысл технологии верификации на основе анализа функционального покрытия заключается в использовании различных метрик для прогнозирования и управления процессом оптимизации формирования и применения тестовых сценариев. Современная технология CDV применяет методы псевдослучайной генерации тестов в рамках заданной системы ограничений. После сбора и анализа информации о функциональном покрытии система ограничений может быть скорректирована (автоматически или вручную) таким образом, чтобы обеспечить проверку всех определенных верификационным планом ситуаций.

Регулярное регрессионное тестирование проектов современных цифровых систем промышленного уровня требует сверхбольших вычислительных ресурсов. Автоматизированное тестирование использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс. Одной из главных проблем автоматизированного тестирования является его трудоемкость: несмотря на то, что оно позволяет устранить часть рутинных операций и ускорить выполнение тестов, большие ресурсы могут тратиться на обновление самих тестов. В ранее проводившихся исследованиях основное внимание уделялось либо достижению более высоких параметров производительности конкретных вычислительных процедур в цикле верификации, либо повышению уровня абстракции модели, либо созданию обратных связей между методами верификации.

Эффективность функциональной верификации может быть достигнута путем ликвидации непроизводительных затрат на создание, поддержку и регулярные запуски избыточных тестов. В качестве критерия оценки избыточности тестовых наборов предполагается использование различных метрик покрытия. Однако в существующих подходах недостаточно проработаны методологические вопросы комбинирования различных метрик покрытия при оценке эффективности тестовых наборов. Несмотря на интенсивное развитие GRID-систем, в том числе в виде облачных кластеров, минимизация нерациональной утилизации вычислительных ресурсов в процессе верификации является чрезвычайно актуальной задачей.

Основной *целью* данного исследования является разработка методологии уменьшения избыточности тестовых наборов на основе систематического анализа эффективности каждого теста для формирования общего уровня покрытия относительно других тестов. Для достижения цели требуется решить ряд *задач*:

1. Формально определить критерии удовлетворения тестирования с учетом всех интересующих автоматических и функциональных метрик покрытия.

2. Создать математическую модель процесса тестирования, как совокупности набора тестов, результатов их запуска и влияния на пространство покрытия.

3. Разработать процедуры слияния результатов тестирования с учетом временных, пространственных составляющих, а также различных видов метрик покрытия.

4. Сформулировать методы ранжирования тестов по соотношению между степенью их влияния на покрытие, достижение уникального покрытия, а также стоимостью вычислительных затрат на их запуск.

**2. Кумулятивное гетерогенное покрытие**

Элементарной единицей анализа покрытия является точка покрытия $P$ (coverage point) – некоторое состояние, переход, событие внутри верифицируемой системы, оказывающее, по мнению проектировщика, существенное влияние на ее функционирование. Для подтверждения полноты процесса функциональной верификации необходимо при помощи измерений во время моделирования зафиксировать количество активизаций каждой из выбранных точек покрытия и сравнить их с заданными пороговыми значениями:

$$P = \{c, g, w\},\ C(P) = \begin{cases} 1, c \geq g \\ 0, c < g \end{cases}, \qquad (1)$$

где c – счетчик (counter) активизаций точки покрытия; g – заданное пороговое значение счетчика (goal); $C(P)$ – предикат, определяющий достаточную степень активизации точки покрытия P, а w – относительный безразмерный весовой коэффициент точки покрытия в общем покрытии.

Совокупность всех заданных некоторой метрикой точек покрытия формирует пространство покрытия (coverage space):

$$S = (P_1, \ldots, P_N). \qquad (2)$$

Анализ покрытия состоит в измерении доли $C(S)$ точек среди всех заданных точек, для которых во время моделирования зафиксировано достаточное количество активизаций:

$$C(S) = \frac{\sum_{i=1}^{N} C(P_i) \times w_i}{\sum_{i=1}^{N} w_i} \times 100\%, \qquad (3)$$

где числитель представляет собой взвешенное количество всех точек покрытия, признанных активизированными, а знаменатель – общее число точек покрытия.

Достижение желаемой доли $C(S)$ наблюдаемых точек называют целью покрытия (coverage goal). Значение цели покрытия редко устанавливается на уровень 100%, поскольку достижение абсолютного покрытия выходит за рамки экономически приемлемых затрат.

Способ представления данных покрытия в виде точек и пространств, механизмы извлечения данных покрытия из входного HDL-описания, принцип измерения, лежащий в основе процедуры инкрементирования счетчиков, полностью зависят от конкретных метрик и их реализации в рамках конкретной системы моделирования. Рассматриваемая в данной работе модель полностью абстрагируется от деталей реализации сбора статистики покрытия, особенностей преобразования одних видов покрытия в другие, от семантики конкретных метрик покрытия.

В целях удобства анализа пространство покрытия $S$ может быть декомпозировано на некоторое множество субпространств $(S'_1, \ldots S'_M)$ (coverage scope). В таком случае их покрытие анализируется отдельно, а общий результат является средним арифметическим с учетом весов:

$$C(S) = \frac{\sum_{k=1}^{M} (C(S'_k) \times \sum_{i=1}^{N_k} w_{ki} \times W_k)}{\sum_{k=1}^{M} (\sum_{i=1}^{N_k} w_{ki} \times W_k)} \times 100\%, \qquad (4)$$

где $C(S'_k)$ – покрытие субпространства $k$; $N_k$ – число точек покрытия в субпространстве $k$; $w_{ki}$ – весовой коэффициент точки покрытия $i$ в $k$; $W_k$ – вес субпространства относительно других субпространств.

Формула (4) может быть применена к любому уровню иерархии проекта рекурсивным образом, соответственно, глубина декомпозиции не ограничивается. Типично информация о покрытии декомпозируется в соответствии с естественной иерархией компонентов проекта. В первую очередь вычисляется покрытие субпространств самых нижних уровней иерархии, а затем происходит композиция результатов на вышестоящих уровнях иерархии. Вышестоящий уровень использует при вычислении и итоговый уровень покрытия конкретного субпространства $C(S'_k)$, и подсчитанные множители, представляющие суммы весов.

Покрытие (4), вычисленное для высшего уровня иерархии для некоторой отдельно взятой метрики X, обозначим как $C(S^{REC})$ и будем называть рекурсивным общим покрытием типа X.

Поскольку отдельные блоки в иерархии могут быть использованы более 1 раза, сохраняя при этом идентичную реализацию, возникает потребность в вычислении единого агрегированного пространства покрытия блока $C(\overline{S})$ за счет суммирования счетчиков из экземпляров:

$$C(\overline{S}) = \frac{\sum_{i=1}^{N} C(\overline{P_i}) \times w_i}{\sum_{i=1}^{N} w_i} \times 100\%, \qquad (5)$$

$$C(\overline{P}) = \begin{cases} 1, (\sum_{j=1}^{N_{inst}} c) \geq t, \\ 0, (\sum_{j=1}^{N_{inst}} c) < t, \end{cases} \quad (6)$$

где $C(\overline{P})$ – предикат, задающий достаточность активизации агрегированной точки покрытия по сумме счетчиков точек из отдельных экземпляров блока, а $N_{inst}$ – число экземпляров блока.

Формула (4) может быть также применена и к агрегированному покрытию, однако в отличие от рекурсивного покрытия глубина декомпозиции общего пространства покрытия на субпространства всегда равна 1. Некоторые языки описания аппаратуры, например SystemVerilog, допускают наличие вложенных модулей. С точки зрения анализа агрегированного покрытия такие модули имеет смысл учитывать отдельно от родительских модулей.

Фактически, покрытие (4) в применении к агрегированным данным представляет собой средневзвешенное покрытие списка всех блоков проекта. Обозначим такую величину, вычисленную для конкретной метрики X, как $C(S^{AGGR})$ и назовем общим агрегированным покрытием типа X.

Агрегированное покрытие является менее точной оценкой по сравнению с рекурсивной, поскольку в вычислении участвует меньшее количество точек покрытия. Однако процесс его измерения требует меньших вычислительных затрат. Можно утверждать, что при идентичных весах конкретных точек покрытия, относительных весах субпространств общее агрегированное покрытие не может быть меньше, чем общее рекурсивное покрытие:

$$C(S^{AGGR}) \geq C(S^{REC}). \quad (7)$$

Показатели $C(S^{AGGR})$ и $C(S^{REC})$ могут быть получены отдельно для любого вида покрытия, используемого в верификационном процессе. Если предположить, что собранные данные нескольких метрик согласованы и сбалансированы между собой системой моделирования при помощи весовых коэффициентов, представляет интерес вычисление специального показателя, объединяющего результаты всех метрик покрытия в единую оценку. Такую оценку будем называть кумулятивным гетерогенным покрытием:

$$K^{AGGR} = \frac{\sum_{i=1}^{N} C(S_i^{AGGR}) \times \Omega_i}{\sum_{i=1}^{N} \Omega_i}, \quad (8)$$

$$K^{REC} = \frac{\sum_{i=1}^{N} C(S_i^{REC}) \times \Omega_i}{\sum_{i=1}^{N} \Omega_i}, \quad (9)$$

где $K^{AGGR}$ и $K^{REC}$ – агрегированное и рекурсивное кумулятивное гетерогенное покрытие соответственно; $C(S_i^{AGGR})$ и $C(S_i^{REC})$ – общее агрегированное и рекурсивное покрытие некоторой метрики i, а $\Omega_i$ – вес метрики покрытия $i$ относительно других метрик.

Разумной стратегией является установка большего веса $\Omega_i$ для высокоуровневых функциональных метрик (ассерционное покрытие, покрытие covergroup), для покрытия конечных автоматов и меньшего веса для низкоуровневых менее информативных автоматических метрик покрытия (покрытие инструкций, разветвлений, термов управляющих выражений, переключений). Если учет тех или иных метрик в кумулятивном гетерогенном покрытии, по мнению инженера, не имеет смысла, метрике должен быть поставлен в соответствие нулевой вес, а вычисление соответствующих индивидуальных показателей общего покрытия можно отключить наряду с предшествующими процессами сбора информации о покрытии и предварительного инструментирования HDL-модели.

Смысл показателей покрытия (8) и (9) состоит в комплексной высокоуровневой оценке завершенности процесса верификации. Кумулятивное покрытие учитывает влияние всех выбранных в верификационном процессе метрик покрытия одновременно, принимая во внимание их значимость при помощи весов. Этот показатель менее точен по сравнению с индивидуальными показателями метрик, и его следует использовать лишь для грубой оценки покрытия в целом. Безусловно, достижение желаемых уровней $C(S_i^{AGGR})$ или $C(S_i^{REC})$ для конкретных метрик является немаловажным. Однако если такие показатели достигнуты для одной из применяемых метрик, например, для ассерционного покрытия, но не достигнуты для других, например, покрытия инструкций, то вывод о достаточности верификации следует считать преждевременным.

### 3. Наборы тестов в контексте анализа покрытия

Динамические методы функциональной верификации, в основе которых стоит использование имитационного моделирования (симуляции), предполагают создание набора тестов для проверки корректности поведения проектируемой системы. Под термином “тест” в широком смысле следует понимать некоторый автоматизированный самостоятельный, самодостаточный, самопроверяющийся сценарий проверки корректности использования проектируемой системы либо ее отдельных элементов на интересующем уровне абстракции.

В общем случае для теста $T$ характерны следующие составляющие:

$$T = \{O, G, I, R, A\}, \quad (10)$$

– определенный объект тестирования O (object) в виде конкретного блока, группы блоков, образующих более крупные компоненты проектируемой системы либо проекта в целом;

– цель тестирования G (goal), связанная с пунктом (пунктами) верификационного плана, задается в виде некоторого предиката, требующего удовлетворения;

– тестовые воздействия I (inputs), заданные либо в явном виде в форме последовательности тест-векторов, либо генерируемые программно во время моделирования детерминированным либо ограниченным псевдослучайным способом;

– ожидаемые реакции R (responces), заданные либо в явном виде при помощи заготовленной последовательности значений наблюдаемых выходов либо в виде программного кода, реализующего некоторый предикат, который определяет корректность реакции;

– верификационная избыточность в виде самопроверяющихся ассерций A (assertions), ни одна из которых не должна быть нарушена за время моделирования.

Тесты, связанные общей тематикой, объектом тестирования и близкими целями, объединяют в наборы (test suits), организованные по иерархическому принципу. Зачастую программная составляющая связанных тестов в существенной мере обобщена и повторно используется с отличием в виде настроек, параметров, связей с конкретными сигналами.

$$S = \{O, T_1, ..., T_n\} \quad O \in S \Leftrightarrow O \in T_1 \Leftrightarrow O \in T_n . \quad (11)$$

Набор тестов для современных цифровых систем может оказаться значительным по размеру, и для их регулярного запуска с сохранением быстроты отклика верификационного процесса часто применяют вычислительные кластеры. В качестве кластера может выступать как внутренняя GRID-система проектной организации, так и публичные облачные системы.

С течением времени и реализация объекта тестирования, и компоненты самого теста могут изменяться. Если в связи с исправлением ошибок, уточнением требований изменяется тест T, то требуется его перезапуск на неизменившейся версии объекта O. Если же изменяется реализация объекта тестирования O, то должно быть повторно выполнено тестирование всех тестовых наборов S, объектом тестирования которых является O либо другой объект, в состав которого входит объект O.

Очевидно, с точки зрения уменьшения вычислительных затрат, необходимо выявить минимальный набор тестов, который необходимо запустить при изменении объекта тестирования. Для упрощения поиска минимального тестового набора должна быть построена функция трассировки $TRACE(P \in O, T) \in \{0,1\}$, основанная на применении запросов к некоторому постоянному хранилищу, содержащему проектные статистические данные. Отказ от использования функции трассировки означает необходимость запуска всех имеющихся тестов при изменении любого объекта тестирования. Такая стратегия подтвердит отсутствие регрессий, однако является крайне нерациональной с точки зрения вычислительных затрат.

Запуском L текущей версии теста T на текущей версии объекта тестирования O называется зафиксированный факт выполнения теста с результирующим статусом, набором внутренних атрибутов моделирования, характеризующих ход выполнения сценария:

$$L_i(t) = T_i(t) \times (O(t) \in T_i), \; L_i(t) = \{s \in (0,1,X), \rho\}, \quad (12)$$

где t – момент времени запуска тестирования; $T_i(t)$ – версия теста с порядковым номером i, а $O(t)$ – версия объекта тестирования, актуальные на момент времени запуска; s – результирующий статус (fail/pass/unknown) запуска; $\rho$ – множество любых атрибутов в виде пар ключ-значение, присвоенных запуску системой моделирования.

Большинство атрибутов транспортирует информацию технического характера, включая версию симулятора, путь к тесту в системе контроля версий, адрес компьютера, выполнявшего тест в вычислительном кластере, данные пользователя, инициировавшего моделирование, длительность тестирования, потребление памяти.

В зависимости от настроек верификационного процесса, в результате запуска индивидуального теста на конкретной версии объекта тестирования может формироваться база данных покрытия $D_i(t)$, содержащая следующую информацию:

– данные о единственном проанализированном тесте $T_i$;

– данные, касающиеся непосредственно запуска $L_i(t)$;

– структура пространств и субпространств покрытия, соответствующая внутренней иерархии объекта тестирования $O(t)$, включая список блоков, иерархию экземпляров блоков, точки покрытия, входящие в конкретные пространства;

– данные о местоположении извлеченных элементов проекта в исходных файлах HDL-описания – для последующего аннотирования и обратной трассировки;

– данные о зафиксированных значениях счетчиков каждой точки покрытия.

На основе сохраненных в базе покрытия данных в любой момент времени для любого блока без повторного запуска моделирования могут быть подсчитаны показатели покрытия (4) и (5), а также общие кумулятивные показатели (8) и (9).

Разделение вычисления показателей покрытия от процесса измерения позволяет без дорогостоящего повторного моделирования модифицировать такие данные базы, как весы конкретных элементов. Наличие

базы также упрощает вычисление кумулятивных гетерогенных показателей с примененными фильтрами по типу покрытия.

На начальном этапе анализа для каждого запущенного теста формируется индивидуальная база покрытия. В дальнейшем возникает необходимость оценки совокупного покрытия, формируемого всеми имеющимися тестами в наборе. Совокупный анализ предполагает наличие процедур слияния данных покрытия в общие "склеенные" базы. По таким базам данных могут формироваться общие проектные отчеты о покрытии, на основе которых принимаются решения о завершенности верификационного процесса, о необходимости создания дополнительных тестов, об избыточности элементов реализации системы и т.п.

При слиянии баз данных представляется возможным отслеживание влияния на конкретные точки и пространства покрытия индивидуальных тестов. Наличие возможности трассировки между элементами реализации и спецификации и подтверждающими их корректность тестами открывает путь к уменьшению вычислительных затрат за счет исключения избыточных тестов, группирования тестов в наборы, запускаемые с различной частотой, получения точного списка тестов, активизирующих конкретную интересующую часть проектируемой системы.

Также актуальной задачей в цикле регрессионного анализа является выявление изменений в показателях покрытия между различными запусками одного и того же теста. Незапланированное ухудшение показателей покрытия является важным сигналом обратной связи для команды.

### 4. Процедуры слияния результатов тестирования в единую базу

В зависимости от цели объединения двух и более баз данных покрытия в единую базу различают 3 вида слияния данных покрытия:

темпоральное слияние (temporal merge) – суммирование/сравнение данных покрытия, вытекающих из одного и того же верификационного процесса, примененного к модификациям (версиям) модели тестируемой системы в различные моменты времени жизненного цикла;

пространственное слияние (spatial merge) – объединение результатов одного или нескольких верификационных процессов, примененных к различным подсистемам одной системы с возможностью корректировки иерархии;

гетерогенное слияние (heterogeneous merge) – объединение данных различных верификационных процессов, примененных к одной и той же версии системы.

На практике все 3 вида слияния могут быть инициированы одновременно. В таком случае для совершения процедуры слияния описанные ниже алгоритмы следует применять последовательно.

*Темпоральное слияни*е имеет существенное значение в цикле регрессионного тестирования модели, поскольку по мере эволюции проекта ранее достигнутое покрытие не должно беспричинно уменьшаться. Регулярное сравнение статистики позволяет своевременно выявлять неожиданные колебания в покрытии, вызванные изменениями в реализации и тестах.

Пусть в рамках жизненного цикла проектируемой системы имеется два момента времени $t_1$ и $t_2$, для которых совершались запуски одного и того же набора тестов $S = \{O, T_1, ..., T_n\}$ на различных версиях объекта тестирования $O(t_1)$ и $O(t_2)$. В результате моделирования получено два вектора данных о запусках тестирования $L(t_1) = (L_1(t_1), ...L_n(t_1))$ и $L(t_2) = (L_1(t_2), ...L_n(t_2))$, а также два набора баз данных покрытия $D(t_1) = (D_1(t_1), ...D_n(t_1))$ и $D(t_2) = (D_1(t_2), ...D_n(t_2))$ для каждого запущенного теста. Применив к каждой точке покрытия операцию суммирования счетчиков, получим темпоральное слияние данных покрытия:

$$L_i(t_1 + t_2) = \{s_1, s_2, \rho_1, \rho_2\},$$

$$D_i(t_1 + t_2) = D_i(t_1) \bigcup D_i(t_2),$$

$$\forall P \in O(t_1) \cup O(t_2), C(P) = C^{t1}(P) \vee C^{t2}(P). \quad (13)$$

Отметим, что во всех видах слияния информация о запусках тестов, участвующих в слиянии, никогда не преобразовывается, а лишь накапливается.

Если структура точек покрытия между версиями объекта тестирования отличается, то в результирующей базе будут представлены объединенные списки точек из обеих входных баз покрытия. Это может быть неожиданным для пользователя в случае удаления из проверяемой модели ряда точек покрытия по мере эволюции реализации системы.

Если весы одних и тех же пространств покрытия отличаются между базами, при темпоральном слиянии выбираются максимальные.

Применяя алгоритм (13) ко всем запускам тестового набора, в истории можно получить так называемое "лучшее покрытие" (best coverage) для выявления разницы между статистикой покрытия текущей версии проекта и максимально достигнутой агрегированной статистикой за всю историю. Наличие такой разницы свидетельствует о проигнорированных командой ухудшениях в уровне покрытия верифицируемой системы.

Больший интерес представляет темпоральное сравнение двух запусков набора тестов, составляющее основу регрессионного анализа покрытия (14). Множество всех точек покрытия, для которых вычитание статистики активизации в более позднем запуске из статистики более раннего запуска дает положительное значение счетчика, выявляет регрессию покрытия. Напротив, множество точек покрытия, для которых упомянутое вычитание дает отрицательное значение счетчика, представляет собой прогрессию покрытия:

$$D_i(t_1 - t_2) = D_i(t_1) \setminus D_i(t_2),$$

$$\forall P \in O(t_1) \cap O(t_2), C(P) = \overline{C^{t1}(P)} \vee C^{t1}(P) \wedge C^{t2}(P), \quad (14)$$

Регрессия на конкретной точке покрытия наблюдается при соблюдении условия:

$$C^{t1}(P) \wedge \overline{C^{t2}(P)} = 1. \quad (15)$$

Применение алгоритма (14) для сравнения двух соседствующих по времени запусков способно выявить регрессионные изменения покрытия в момент, близкий к моменту внесения нежелательных изменений в реализацию модели либо в тестовые наборы. В свою очередь, применение (14) для сравнения наиболее позднего запуска с базой "лучшего покрытия" отвечает на вопрос, имеются ли в текущей версии неустраненные регрессии покрытия. Список прогрессий можно получить поменяв порядок запусков-операндов при темпоральном сравнении. Такой список может быть интересным для подтверждения влияния произведенных модификаций на покрытие. Изменения в структуре точек покрытия между запусками не являются ни регрессиями, ни прогрессиями, и игнорируются при темпоральном сравнении.

*Пространственное слияние* необходимо для обеспечения возможности горизонтального масштабирования верификации. Современные проекты достигают гигантских размеров и способны превысить вычислительные возможности средств автоматизации и компьютерных систем даже при обработке на высоком уровне абстракции. Соответственно, оправдан подход, основанный на декомпозиции тестовых наборов по ключевым подсистемам, верификация которых происходит при помощи отдельных вычислительных ресурсов. Пространственное слияние статистики покрытия позволяет осуществить обратную композицию на уровне результатов.

Пусть имеются текущие версии двух объектов тестирования $O_1(t)$ и $O_2(t)$, которые были протестированы при помощи двух различных тестов $T_1$ и $T_2$, в результате чего получены две базы покрытия $D_{O_1}(t)$ и $D_{O_2}(t)$. Предположим, существует больший объект тестирования $O'(t)$, для которого соблюдаются условия $O_1(t) \in O'(t)$ и $O_2(t) \in O'(t)$. В результате пространственного слияния может быть получена база покрытия для объекта $O'(t)$:

$$D_{O_1+O_2}(t) = D_{O_1}(t) \bigcup D_{O_2}(t),$$
$$S_{O_1} \cap S_{O_2} = \varnothing. \quad (16)$$

С точки зрения рекурсивного покрытия, пространства покрытия верхнего уровня каждой из баз не имеют точек пересечения, поскольку объединяются разные непересекающиеся в пространстве иерархии. Общее рекурсивное покрытие объединенной базы вычисляется следующим образом:

$$C(S^{REC}_{O'}) = \frac{\sum_{i=1}^{N_1} C(S_i^{REC-O_1}) \times \Omega_i^{O_1} + \sum_{k=1}^{N_2} C(S_k^{REC-O_2}) \times \Omega_k^{O_2}}{\sum_{i=1}^{N_1} \Omega_i^{O_1} + \sum_{i=1}^{N_2} \Omega_i^{O_2}}, \quad (17)$$

где $N_1$ и $N_2$ – количество субпространств на верхнем уровне иерархии первой и второй базы.

При подсчете агрегированного покрытия пространства могут пересекаться, так как каждая из иерархий может содержать внутренние блоки одинакового типа. Слияние агрегированных данных для пересекающихся блоков должно происходить аналогично темпоральному слиянию (13):

$$C(S^{AGGR}_{O'}) = $$
$$= \Big( \sum_{i=1}^{N_1 - N_{1\cap 2}} C(S_i^{AGGR-O_1}) \times \Omega_i^{O_1} +$$
$$+ \sum_{k=1}^{N_2 - N_{1\cap 2}} C(S_k^{AGGR-O_2}) \times \Omega_k^{O_2} +$$
$$+ \sum_{j=1}^{N_{1\cap 2}} C(S_j^{AGGR-(O_1+O_2)}) \times \Omega_j^{(O_1+O_2)} \Big) /$$
$$/ \Big( \sum_{i=1}^{N_1 - N_{1\cap 2}} \Omega_i^{O_1} + \sum_{i=1}^{N_2 - N_{1\cap 2}} \Omega_i^{O_2} + \sum_{j=1}^{N_{1\cap 2}} \Omega_j^{(O_1+O_2)} \Big), \quad (18)$$

где $N_{1\cap 2}$ – число общих агрегированных блоков между двумя входными базами; $C(S_j^{AGGR-(O_1+O_2)})$ – подсчитанное по алгоритму (13) агрегированное покрытие общих блоков, а $\Omega_j^{(O_1+O_2)}$ – вектор средних арифметических между весами общих блоков в каждой из баз.

Пространственное слияние также допускает коррекцию иерархии, когда в первой базе имеется некоторая надсистема $O'(t)$ с подсистемой $O_1(t)$, а вторая база содержит информацию о другой подсистеме $O_2(t)$, но не содержит сведений о надсистеме. В таком случае алгоритм (17) для рекурсивного покрытия применяется к более низкому уровню пространств покрытия, а затем покрытие верхнего уровня пересчитывается по формуле (4). Коррекция иерархии не оказывает никакого влияния на алгоритм (18) для агрегированного покрытия.

*Гетерогенное слияние* предполагает объединение данных покрытия разного типа, полученных на одном и том же тесте и объекте тестирования. Если в пространственном слиянии в модель добавляются новые субпространства и точки покрытия одной и той же метри-

ки, представляющие соседствующие иерархии, то при гетерогенном слиянии одна и та же иерархия дополняется "слоями", соответствующими другим метрикам покрытия.

Простейший подход состоит в тривиальном объединении списков субпространств покрытия разного типа в рамках единых пространств, отражающих структурную иерархию проекта. Если в подвергаемых слиянию базах покрытия не имеется пересечений по типу примененных метрик, такое слияние не влияет на показатели общего рекурсивного и агрегированного покрытия (4) и (5). Изменениям подвергаются только кумулятивные показатели (8) и (9). При наличии пересечений по типу для общих метрик применяется алгоритм (13), характерный для темпорального слияния.

Однако простейший подход не учитывает факторы пересечения семантики различных метрик, состоящие в зависимости одних метрик покрытия от других. При наличии подготовленной структурной информации представляется возможным преобразовывать данные одних метрик покрытия в другие в целях получения более точных оценок кумулятивного покрытия (8) и (9).

Пусть для некоторого теста $T$, примененного к объекту тестирования $O(t)$, имеются две базы покрытия $D^A(t)$ и $D^B(t)$ для метрик покрытия A и B соответственно. Пусть целью слияния является получение результирующей базы покрытия исключительно с использованием метрики $A$. Для достижения данной цели необходимо наличие подготовленной структурной информации о зависимостях между счетчиками точек покрытия обеих метрик в виде контекстных функций преобразования $F_i$:

$$\forall i, 0 < i < N^B, C(P_i^B) = F_i(C(P_1^A),...,C(P_N^A)), N < N^A, \quad (19)$$

где $N^A$ и $N^B$ – количество точек покрытия в объекте тестирования с точки зрения соответствующих метрик, а N – число точек покрытия в метрике A, непосредственно влияющих на значение покрытия точки $P_i^B$. Тогда данные по метрике B могут быть приведены к метрике A согласно алгоритму (20), основанному на подстановке (19) в общий алгоритм (4):

$$C(S^{B\to A}) = \frac{\sum_{i=1}^{N^B} C(P_i^B)\times w_i}{\sum_{i=1}^{N} w_i}\times 100\% = \frac{\sum_{i=1}^{N^B} F_i(C(P_1^A),...,C(P_N^A))\times w_i}{\sum_{i=1}^{N} w_i}. \quad (20)$$

Конкретная функция преобразования зависит от типа приводимых друг к другу метрик. Однозначные функции преобразования могут быть построены только между ограниченным набором пар метрик. Например, покрытие разветвлений можно выразить через покрытие инструкций либо покрытие термов управляющих выражений. Покрытие переходов в конечных автоматах можно выразить через покрытие разветвлений. Однако, во-первых, такие преобразования существуют не между каждой парой метрик, во-вторых, может не существовать функции, обратной (19), поскольку происходит потеря данных.

Типичные функции (19) представляют собой суммы значений покрытия для подмножеств точек, из чего следует возможность дальнейшей минимизации формулы (20) за счет группировки слагаемых с одинаковыми множителями при подстановке в формулы вычисления кумулятивного покрытия (8) и (9).

После приведения данных от одной метрики к другой следует применить алгоритм (13), автоматически исключающий дупликации. Таким образом, общее покрытие будет обладать более высокой точностью благодаря уменьшению количества эквивалентных точек покрытия между различными метриками.

### 5. Процедуры ранжирования тестов

Во время осуществления процедур слияния баз покрытия представляется возможным организовать сбор информации о взаимном влиянии между тестами и точками покрытия. На основе данной ассоциативной информации для осуществления дальнейшего анализа эффективности тестового набора должна быть построена специальная табличная функция трассировки $TRACE(P,T) \in \{0,1\}$.

Пусть в проекте имеется пространство покрытия, состоящее из 4 точек $S = \{P_1,...,P_4\}$. Допустим, некоторый тест $T_1$ покрывает точку $P_1$, другой тест $T_2$ – точки $P_1$ и $P_3$, а третий тест $T_3$ – точки $P_2$, $P_3$ и $P_4$. Осуществим слияние баз покрытия, сгенерированных данными тестами, и определим на результирующих данных функцию трассировки:

$$\begin{array}{lll} TRACE(P_1,T_1)=1 & TRACE(P_1,T_2)=1 & TRACE(P_1,T_3)=0 \\ TRACE(P_2,T_1)=0 & TRACE(P_2,T_2)=0 & TRACE(P_2,T_3)=1 \\ TRACE(P_3,T_1)=0 & TRACE(P_3,T_2)=1 & TRACE(P_3,T_3)=1 \\ TRACE(P_4,T_1)=0 & TRACE(P_4,T_2)=0 & TRACE(P_4,T_3)=1 \end{array} \quad (21)$$

Дизъюнкция значений по строкам в функции трассировки определяет покрытие конкретной точки на всем тестовом наборе:

$$C(P_i) = \bigcup_{k=1}^{N_T} TRACE(P_i, T_K), i \in [1, N_P], \quad (22)$$

где $N_t$ – число тестов в базе покрытия; $N_p$ – общее число точек покрытия. Покрытие пространства (3) может быть выражено через функцию трассировки:

$$C(S)=\frac{\sum_{i=1}^{N_P}\left(\bigcup_{k=1}^{N_T}TRACE(P_i,T_K)\right)\times w_i}{\sum_{i=1}^{N_P}w_i}. \quad (23)$$

В примере (21) результирующее покрытие достигает 100%, поскольку значение (22) для каждой точки покрытия равно 1. Однако,несмотря на достижение целевого покрытия, набор тестов является избыточным.

Подсчитаем взвешенную сумму столбцов в функции трассировки для каждого теста и назовем ее рангом теста:

$$R(T_k)=\frac{\sum_{i=1}^{N_P}(TRACE(P_i,T_K)\times w_i)}{\sum_{i=1}^{N_p}w_i},k\in[1,N_T]. \quad (24)$$

Упорядочим тесты в соответствии с рангом (24):

$$\begin{aligned}R(T_3)&=(0+1+1+1)/4=0.75,\\ R(T_2)&=(1+0+1+0)/4=0.5,\\ R(T_1)&=(1+0+0+0)/4=0.25.\end{aligned} \quad (25)$$

Далее, начиная с двух тестов с наибольшими рангами, сформируем минимальный набор, при котором будет достигнута цель покрытия. На каждом шаге необходимо оценивать, добавляет ли очередной тест что-либо к общему покрытию. Этого можно достигнуть без процедур слияния для каждой пары тестов, используя лишь функцию трассировки. Тесты, представляющие собой, с точки зрения результирующего покрытия, подмножества предыдущих тестов, должны быть исключены. Если на очередном шаге соблюдается критерий (26), очередной тест следует добавить в набор, а в противном случае – исключить:

$$\frac{\sum_{i=1}^{N_P}\left(\overline{TRACE(P_i,T_{PREV})}\wedge TRACE(P_i,T_{NEXT})\right)\times w_i)}{\sum_{i=1}^{N_p}w_i}>0 \quad (26)$$

где $T_{PREV}$ – агрегация всех уже включенных в результирующий набор тестов; $TRACE(P_i,T_{PREV})$ – дизъюнкция значений функции трассировки на всех включенных в $T_{PREV}$ тестов, а $T_{NEXT}$ – следующий по рангу тест. Применив (26) к текущему примеру, получим, что для обеспечения целевого покрытия достаточно использовать тесты $T_3$ и $T_2$.

Точность рангов (24) можно существенно повысить, учитывая нормированное время выполнения теста. Для каждого теста в базе покрытия имеется атрибут длительности выполнения $\rho^{DUR}$. Обозначим самое меньшее значение длительности из имеющихся данных как $\rho^{DUR}_{MIN}$. Уточним оценку (24) при помощи атрибута длительности:

$$R(T_k)^{DUR}=R(T_k)\times\frac{\rho^{DUR}}{\rho^{DUR}_{MIN}}=\frac{\sum_{i=1}^{N_P}(TRACE(P_i,T_K)\times w_i)}{\sum_{i=1}^{N_p}w_i}\times\frac{\rho^{DUR}_{MIN}}{\rho^{DUR}},k\in[1,N_T] \quad (27)$$

Пусть длительность выполнения тестов в рассматриваемом примере имеет значения

$$\rho^{DUR}(T_1)=12,\ \rho^{DUR}(T_2)=38,\ \rho^{DUR}(T_3)=45.$$

Соответственно, находим показатель минимальной длительности $\rho^{DUR}_{MIN}=\rho^{DUR}(T_1)=12$. Пересчитав ранги тестов с учетом длительности выполнения, увидим существенные отличия в результирующем ранжировании (28). Применив алгоритм (26) к новым значениям рангов, получим минимальный набор тестов, достигающих целевого покрытия, который состоит из тестов $T_1$ и $T_2$. Общее время выполнения такого набора составит $\rho^{DUR}(T_1)+\rho^{DUR}(T_2)=50$, что существенно эффективней ранее полученного набора без учета длительности выполнения, для которого общее время составит $\rho^{DUR}(T_2)+\rho^{DUR}(T_3)=83$:

$$\begin{aligned}R(T_1)^{DUR}&=R(T_1)\times\frac{\rho^{DUR}(T_1)}{\rho^{DUR}_{MIN}}=0.25\times\frac{12}{12}=0.25,\\ R(T_3)^{DUR}&=R(T_3)\times\frac{\rho^{DUR}(T_3)}{\rho^{DUR}_{MIN}}=0.75\times\frac{12}{45}=0,2,\\ R(T_2)^{DUR}&=R(T_2)\times\frac{\rho^{DUR}(T_2)}{\rho^{DUR}_{MIN}}=0.5\times\frac{12}{38}\approx 0,15789.\end{aligned} \quad (28)$$

Поскольку длительность фактического выполнения может колебаться в зависимости от компьютерной системы и ее загрузки в конкретный момент времени, для получения еще более точных оценок следует учитывать количество тактов реальной загрузки процессора, либо использовать среднестатистическое время, измеренное на протяжении нескольких запусков.

Описанная процедура ранжирования имеет вычислительную сложность $N\times\log_2(N)$ в типичном случае относительно количества тестов. В худшем случае вычислительная сложность возрастает до порядка $N^2$, если между тестами не имеется пересечений по покрытию. Данный алгоритм относится к классу "жад-

ных алгоритмов", поскольку на каждом шаге принимается локальное оптимальное решение. Однако получаемый в результате такого ранжирования набор тестов, в общем случае, не будет оптимальным глобально.

Глобальная оптимальность может быть достигнута алгоритмом итеративного ранжирования за счет больших вычислительных затрат. На первой итерации требуется подсчитать покрытие каждой пары тестов в виде матрицы:

$$\begin{array}{ccccc} C(T_1+T_2)=C(T_1)\cup C(T_2) & .. & C(T_i+T_2)=C(T_i)\cup C(T_2) & .. & C(T_N+T_2)=C(T_N)\cup C(T_2) \\ ... & .. & ... & .. & ... \\ C(T_1+T_k)=C(T_1)\cup C(T_k) & .. & C(T_i+T_k)=C(T_i)\cup C(T_k) & .. & C(T_N+T_k)=C(T_N)\cup C(T_k) \\ ... & .. & ... & .. & ... \\ C(T_1+T_N)=C(T_1)\cup C(T_N) & .. & C(T_i+T_N)=C(T_i)\cup C(T_N) & .. & C(T_{N-1}+T_N)=C(T_{N-1})\cup C(T_N) \end{array} \quad (29)$$

Далее следует получить матрицу изменений покрытия при слиянии:

$$\forall i,k\in[1,N],\Delta C(T_i+T_k)=C(T_i+T_k)-C(T_i)\,. \quad (30)$$

В том случае, когда для некоторого теста $T_k$ значение всех ячеек $\Delta C(T_i+T_k)$ равно 0, либо ниже заданного пользователем минимального порога, то тест $T_k$ следует исключить из анализа как очевидно избыточный.

Затем следует выбрать в матрице (30) ячейку с наибольшим значением, и объединить данные покрытия тестов, соответствующих конкретному столбцу и строке матрицы при помощи алгоритма (13). В дальнейшем объединение двух тестов будет обрабатываться как единый тест.

Алгоритм следует повторять циклически, удаляя очевидно избыточные тесты на каждом шаге по принципу (30), а затем объединяя группы тестов, дающие наибольший прирост совместного покрытия. По завершению нескольких итераций алгоритма будет получен набор тестов, для которого либо будет достигнут целевой уровень покрытия, либо разницы (30) относительно всех оставшихся тестов будут меньше заданного минимального порога.

Учет длительности времени выполнения тестов также способен улучшить точность итеративного ранжирования. Для этого матрицу (30) следует видоизменить с учетом относительной вычислительной стоимости пары тестов:

$$\forall i,k\in[1,N],\Delta C(T_i+T_k)^{DUR}=$$

$$=(C(T_i+T_k)-C(T_i))\times\frac{\rho^{DUR}(T_i)+\rho^{DUR}(T_k)}{\rho^{DUR}_{MIN}}. \quad (31)$$

Очевидно, после объединения тестов в группу длительность их выполнения следует суммировать для следующих итераций.

Вычислительная сложность итеративного ранжирования оценивается как $N^2\times\log_2(N)$ в типичном случае и $N^3$ – в худшем случае. Аналогично "жадному" алгоритму ранжирования, худшим случаем является ситуация, когда в покрытии всех тестов не имеется пересечений.

По результатам ранжирования также можно выявить минимальный с точки зрения времени выполнения набор тестов, обладающий максимальным покрытием. Суммарное покрытие данного набора может не достигать целевого покрытия, однако такой набор может быть использован как быстрый набор для частых запусков при небольших изменениях (smoke test). Полный набор тестов в таком случае можно запускать реже, поскольку вероятность появления ошибок, не выявляемых быстрым набором, существенно падает.

**6. Выводы**

В ходе исследования были решены ключевые поставленные *задачи*:

1. Формально определены критерии удовлетворения тестирования на основе гетерогенной модели покрытия, учитывающей все интересующие автоматические и функциональные метрики с возможностью коррекции влияния наиболее важных факторов при помощи весовых коэффициентов.

2. Создана математическая модель процесса тестирования, описывающая наборы тестов и результаты их запуска с точки зрения влияния на различные метрики покрытия в субпространствах проекта.

3. Разработы процедуры темпорального, пространственного и гетерогенного слияния результатов тестирования, являющиеся основой для алгоритмов ранжирования тестов.

4. Сформулированы два метода ранжирования тестов по соотношению между степенью их влияния на покрытие, достижение уникального покрытия, а также стоимостью вычислительных затрат на их запуск.

Главный *научный результат* исследования состоит в разработке методологии уменьшения избыточности набора регрессионных тестов для функциональной верификации цифровых устройств на основе систематического анализа эффективности каждого теста в формировании уровня кумулятивного гетерогенного покрытия относительно других тестов.

Ключевым *практическим результатом* исследования является выявление тестов, вычислительные затраты на проверку которых не оправданы полезностью получаемой в результате их запуска информацией. При этом полезность тестов оценивается с учетом их влияния одновременно на все применяемые в верификационном процессе метрики покрытия. Путь достижения более эффективного процесса функциональной верификации состоит в формировании набора тестов, требующего как можно меньших вычислительных затрат, при этом обеспечивающего приемлемый для выхода на рынок уровень качества выпускаемого изделия.

**Литература: 1.** *Rashinkar P.* System-On-A-Chip Verification. Norwell: Kluwer Academic, 2002. 372 p. **2.** *Piziali A.* Functional Verification Coverage Measurement And Analysis. Norwell: Springer, 2004. 213 p. **3.** *Wilcox P.* Professional Verification: A Guide to Advanced Functional Verification. Norwell: Springer, 2004. 191 p. **4.** *Adler Y., Blue D., Conti T., Prewitt R., Ur S.* Evaluating Workloads Using Comparative Functional Coverage. Haifa Verification Conference. 2008. P. 84-98. **5.** *Kwon Y.-S., Kim Y.-I., Kyung C.-M.* Systematic Functional Coverage Metric Synthesis from Hierarchical Temporal Event Relation Graph // Design Automation Conference, 41st Conference on (DAC'04). 2004. C. 45-48. **6.** *Molnar C.* Regression Testing: What to test and When // Aerospace and Electronic Systems Magazine, IEEE. 2001. Volume: 16 ,Issue: 6. P. 25-30. **7.** *Bailey B.* The Multiple Dimensions of Scalability that Comprise a System Verification Strategy. Mentor Graphics Scalable Verification White Paper. 2004. P. 47-50.

Поступила в редколлегию 14.12.2013

**Рецензент:** д-р техн. наук, проф. Хаханов В.И.

**Зайченко Сергей Александрович**, канд. техн. наук, директор ООО "Алдек-КТС" (Харьков), доцент кафедры АПВТ ХНУРЭ. Научные интересы: автоматизированное проектирование и верификация цифровых систем. Увлечения: технологии онлайн-образования. Адрес: Украина, 61045, Харьков, ул. Космическая, 23а, тел. (057)-760-47-25.

**Александров Владислав Игоревич**, инженер-программист ООО "Алдек-КТС" (Харьков). Научные интересы: автоматизированное проектирование и верификация цифровых систем. Увлечения: футбол. Адрес: Украина, 61045, Харьков, ул. Космическая, 23а, тел. (057)-760-47-25.

**Белоус Василий Виталиевич**, студент 5 курса ХНУРЭ, факультет компьютерной инженерии и управления, кафедра АПВТ, специальность СКС. Научные интересы: автоматизированное проектирование и верификация цифровых систем. Увлечения: международный туризм. Адрес: Украина, 61166, Харьков, пр. Ленина 14, тел. (057)-702-13-26.

**Березин Никита Петрович**, студент 5 курса ХНУРЭ, факультет компьютерной инженерии и управления, кафедра АПВТ, специальность СКС. Научные интересы: автоматизированное проектирование и верификация цифровых систем. Увлечения: методики самоорганизации. Адрес: Украина, 61166, Харьков, пр. Ленина 14, тел. (057)-702-13-26.

УДК 658.512.011:681.326:519.713

# ДИАГНОСТИРОВАНИЕ HDL-МОДЕЛЕЙ СИСТЕМ НА КРИСТАЛЛАХ

*BAGHDADI AMMAR AWNI ABBAS, ХАХАНОВ В.И., ЛИТВИНОВА Е.И., ЗАЙЧЕНКО С.А.*

Предлагается технология диагностирования HDL-моделей систем на кристаллах, которая базируется на использовании транзакционного графа. Описывается метод диагностирования, направленный на уменьшение времени обнаружения неисправностей и памяти для хранения диагностической матрицы за счет формирования тернарных отношений между тестом, монитором и функциональным компонентом. Разрабатывается модель цифровой системы в виде транзакционного графа и мультидерева таблиц неисправностей, а также тернарные матрицы активации функциональных компонентов выбранного набора мониторов с помощью тестовых последовательностей. Предлагается метод анализа матрицы активации в целях обнаружения неисправных блоков с заданной глубиной и синтеза логических функций для последующего встроенного аппаратного диагностирования неисправностей.

## 1. TAB-модель диагностирования неисправных компонентов SoC

*Цель* исследования – разработка матричной модели TAB (Tests – Assertions – Blocks) и метода диагностирования, позволяющих уменьшить время тестирования и объем памяти для хранения диагностической информации за счет формирования тернарных отношений (тест – монитор – функциональный компонент) в одной таблице.

*Задачи* исследования: 1) разработка HDL-модели цифровой системы в форме транзакционного графа для диагностирования функциональных блоков с использованием набора ассерций [1-6,15]; 2) разработка метода анализа TAB-матрицы в целях обнаружения минимального набора неисправных блоков [4-7,13]; 3) синтез логических функций для встроенной процедуры диагностирования неисправностей [8-11,14].

Модель тестирования HDL-кода цифровой системы представлена следующими xor-отношениями параметров <тест – функциональность – неисправные блоки B*>:

$$T \oplus B \oplus B^* = 0;$$
$$B^* = T \oplus B = \{T \times A\} \oplus B,$$

которые преобразуются в отношения компонентов TAB-матрицы:

$$M = \{\{T \times A\} \times \{B\}\},$$
$$M_{ij} = (T \times A)_i \oplus B_j.$$

Здесь координаты матрицы равны 1, если пара тест-монитор $(T \times A)_i$ проверяет или активирует неисправности функционального блока $B_j \in B$.

Аналитическая модель верификации с использованием темпоральных ассерций (дополнительная наблюдаемость операторов или линий) ориентирована на достижение заданной глубины диагностирования:

$$\Omega = f(G, A, B, S, T),$$
$$G = (A * B) \times S; S = f(T, B);$$
$$A = \{A_1, A_2, ..., A_i, ..., A_h\};$$
$$B = \{B_1, B_2, ..., B_i, ..., B_n\};$$
$$S = \{S_1, S_2, ..., S_i, ..., S_m\};$$
$$T = \{T_1, T_2, ..., T_i, ..., T_k\}.$$