УДК 658.5:681.3

РЕАЛИЗАЦИЯ МОДИФИКАЦИЙ ЭФФЕКТИВНЫХ ГЕНЕРАТОРОВ ТЕСТОВ

В.А. Андриянов, А.Н. Мартынюк

Одесский национальный политехнический университет
65044, Одесса, проспект Шевченко, 1,
тел. 8 (048) 734 8322
anmartynyuk@ukr.net

Работа посвящена исследованию структурных алгоритмов генерации тестов. Целью исследования является модификация существующего алгоритма генерации тестов за счет сокращения числа откатов выполненных алгоритмом. В работе выполнен анализ алгоритмов, а так же разработана модификация алгоритма FAN за счет улучшения эвристики, использованной для управления обратной трассировкой и прототип системы генерации тестов на базе исследуемых (D, PODEM, FAN).

This work is devoted research of structural algorithms of generation of tests. The purpose of the research is modification of existing algorithm of test generation by reducing the number of rollbacks the algorithm, This work made algorithm analyze, modification of FAN algorithm by the expense of improvement the heuristics used to control backtrace and a prototype of system of generation of tests on the basis of investigated (D, PODEM, FAN).

Введение

В современных компьютерных технологиях по-прежнему большое значение уделяется разработке цифровых устройств. Требования, предъявляемые к надежности таких устройств высоки, так как ошибки и сбои в них могут привести к некорректной работе систем, в которые эти устройтсва входят. В то же время число критичных применений самих компьютерных систем высоко.

Как правило, цифровые устройства достаточно сложны и их производство возможно начинать только после тщательного проектирования. Так как исправление ошибок в готовом устройстве требует больших затрат, ошибки важно выявить и исправить на этапе проектирования. Вместе с тем, всевозможные виды моделирования, верификации и тестирования на этом этапе не исчключают возможности появления бракованных изделий. Кроме того, в процессе эксплуатации ошибки появляются в результате целого ряда причин, начиная от естественной деградации материала микросхем и заканчивая экстремальными условиями использования. Все это обуславливает необходимость развития и эффективной реализации разнообразных методов тестирования и, следовательно, синтеза самих тестов.

В общем случае обычно определяют два вида тестов: параметрические и функциональные. Функциональное тестирование использует входны есигналы для получения выходных реакций. Генерация тестовых векторов и сценариев – одна из наиболее важных составляющих классических задач анализа, синтеза, контроля и диагноза вычислительных систем. Вместе с тем, она до сих пор не является в полной мере разрешенной, в частности, из-за того, что темпы развития аппаратуры существенно превышают темпы развития генерации тестов. Проблема состоит в том, что все алгоритмы генерации тестов относятся к классу NP-полных задач и не всегда возможно ответить на вопрос сколько времени требуется для построения теста для трудно проверяемых ошибок и сбоев. В этой связи, в частности, актуальна задача сокращения времени поиска тестового решения, выполняемого за счет уточнения промежуточных решений, которые принимаются алгоритмами генерации тестов.

Анализ основных составляющих алгоритмов генерации тестов

Генераторы тестов решают задачу поиска наборов входных сигналов, организованных в вектора и структуры (сценарии), которые выявляют разницу в работе исправных и неисправных устройств. Кроме поискатестов, алгоритмы генерации тестов имеют еще несколько применений, а именно:

- нахождение избыточных частей в цифровых устройствах;
- доказательство соответствия разных реализаций цифрового устройства друг другу.

Для решения задачи поиска теста генератор теста должен иметь описание тестируемого устройства с заданным уровнем детализации. Чем выше уровень детализации, тем больше разных ошибок и сбоев могут быть промоделированы. Для моделирования, в частности, «const1» и «const0» необходимо описать утройство в виде набора элементов и связей между ними. Если тестируютс явременные сбои, описание расширяется информацией о задержках распространения сигналов. Эти описания могут быть прямыми и непрямыми.

Простейший тест – исчерпывающий. В этом случае поиск тест-векторов не тебуется, так как тест включает все возможные входные вектора (и их последовательности для устройств с памятью). Однако исчерпывающий тест обладает избыточностью, которую следует уменьшить. Например, предположение одиночности ошибок и сбоев позволяет выполнить их сжатие на основе их оптимального покрытия тестовыми векторами с учетом их различимости (эквивалентности).

Все генераторы тестов используют структуры данных, представляющих пространство поиска тестовых векторов. Например, часто используются бинарные деревья поиска. При этом полный генератор может при поиске тестового вектора обойти все дерево. Нетестируемый сбой не вывляется даже после полного обхода дерева, это означает, устройтсво работает корректно даже в присутствии сбоя.

Алгебра генераторов тестов обычно имеет расширенную булеву нотацию, единую для корректной и дефектной схемы, что обуславливает один проход для определения значений. Простейшая алгебра – пятизначная $\{0, 1, X, D, \overline{D}\}$.

Используемые алгоримы генераторов тестов для цифровых схем обычно псевдослучайные и детерминированные – на основе активизации путей.

В генераторах на основе активизации путей для константных неисправностей активизация и распространение обычно выполняются с помощью тех или иных процедур

- импликации;
- определения целей;
- обратной трассировки;
- определения возможности продолжения поиска;
- смены последнего решения.

Вместе эти процедуры образуют алгоритмы «согласования и связи» (“branch and bound” algorithm), предназначенные для поиска и его прерывания на возможно более ранних стадиях, как только становится ясно, что путь не ведет к желаемому результату. Как следствие, уменьшаются вычислительные затраты.

Прямая или обратная импликация обеспечивает расчет полностью определенных сигналов в схеме. Удобная структура для импликации стек, где записываются наборы выбранных генератором сигналов в выбранных узлах, например, входах, схемы.

Остальные процедуры реализуют «согласования и связи».

Так процедура определения целей выбирает промежуточные цели, существенно влияющие на объем вычислений, процедура обратной трассировки преобразует прмежуточные цели в узлы дерева выбора, используемого для поиска.

Процедура смены последнего решения меняет одно из решений, принятых ращее предыдущими процедурами, если формируется вывод о невозможности достижения требуемых результатов. От процедуры смены последнего решения также существенно зависит объем вычислений. Обычно смена решения происходит, если:

- фронт стал пустым (нет распространения);
- сигнал обновременно и «0», и «1», что невозможно.

Для поиска тестового вектора в прямом или обратном виде используются активизация и распространение сбоя. Активизация обеспечивает условия проявления сбоя, распространение обеспечивает возможность наблюдения этого проявления на выходах схемы.

Постановка задачи исследования

Основная задача определяется как модификация эффективных, программно реализуемых алгоритмов генерации тестов с целью возможно более раннего сокращения числа неверных решений этих алгоритмов.

Для решения этой задачи следует решить подзадачи:

- анализа эффективных, программно реализуемых алгоритмов генерации тестов;
- разработки методики уточнения меры контролируемости на основе алгоритма SCOAP;
- модификации основных процедур, ориентированной на эффективную программную реализацию;
- разработки среды сопоставления алгоритмов и их реализаций, в частности PODEM и FAN.

Представление программной реализации эффективных генераторов тестов

Разработанная в результате исследования алгоритмов генераторов тестовых векторов программная среда генерации тестов определяет:

- собственные структуры данных для описания цифровых схем;
- собственный язык описания цифровых схем на базе XML, а также синтаксически управляемый транслятор этих описаний в динамический код для сборки соответствующих цифровых схем;
- графический редактор цифровых схем на базе тонкого клиента;
- алгоритмы моделирования работы цифровых схем;
- алгоритмы для поиска и устранения эквивалентных сбоев;
- алгоритмы для нахождения минимального набора тестовых векторов;
- d-алгоритм в качестве базового алгоритма;
- утилита для создания комбинационных схем с определенными параметрами;
- утилиты для сбора статистики, подсчета затраченного времени, а также расчета текущего покрытия;
- алгоритмы PODEM и FAN;

Собственные структуры данных для описания схем включают в себя:

- простой элемент;
- сложный элемент;
- связь.

Сложный элемент может быть представлен как композиция более простых элементов, либо в виде «черного ящика».

Собственный язык описания схем на базе XML дает возможность добавления в систему описаний новых устройств. При синтезе схем вместо интерпретации XML описания выполняется его трансляция в машинно-независимый язык низкого уровня CIL. Это позволяет значительно сократить время синтеза схемы и увеличить производительность системы в целом.

На рис.1 показана обобщенная схема взаимодействия между предлагаемой системой генерации тестов и библиотекой описания устройств. Описание устройства проходит этапы лексического, синтаксического, семантического анализа, и этап трансляции перед занесением в библиотеку «методов синтеза» устройств.

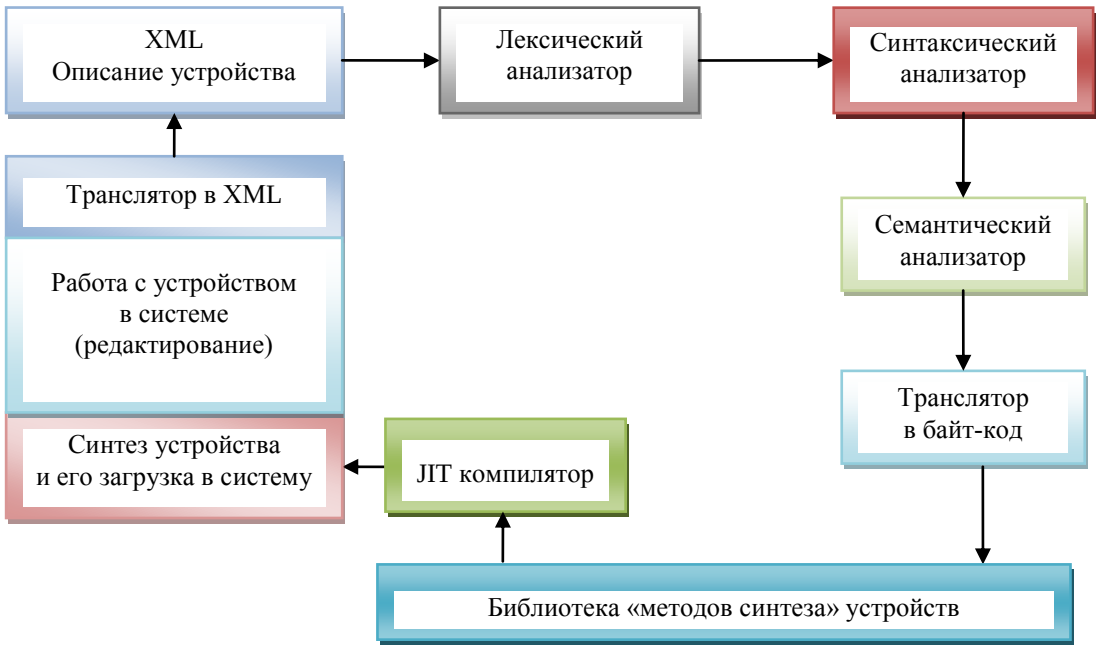

Рис. 1. Взаимодействие системы и библиотеки описаний устройств

Следует отметить, что существует возможность внедрения в систему VHDL-интерпретатора.

Существующую реализацию системы можно рассматривать как готовую среду разработки тестов. Она реализована в виде тонкого клиента на базе технологии Silverlight, но может быть легко преобразована в приложение для рабочего стола на базе технологии WPF.

Система позволяет выполнять моделирование цифровых схем. Для моделирования работы простых логических элементов используется таблицы или альтернативные графы их функционирования. Для сложных элементов представленных в виде «черного ящика» моделирование выполняется с помощью специальных процедурных описаний.

В систему введены алгоритмы для обнаружения и устранения эквивалентных сбоев, что позволяет сократить время поиска тестов. В системе используются алгоритмы поиска минимального набора тестовых векторов, что позволяет устранить поглощаемые тестовые вектора. Система позволяет собирать статистику и контролировать процесс поиска тестов с помощью различных алгоритмов. Статистика представляется в системе в виде следующих графиков зависимости:

- количества откатов от параметров устройства;
- покрытия от разрешенного количества откатов на один сбой;
- времени работы от требуемого покрытия;

Система предоставляет информацию о среднем времени, затраченном на один сбой, а также информацию о приблизительном времени нахождения теста.

Система может выполнять генерацию случайных тестовых векторов, а также использует различные детерминированные алгоритмы для поиска тестов, в том числе:

- d-алгоритм;
- PODEM;
- FAN;
- TRAN.

В системе используется так называемые Е-фронты, которые разделяют тестируемую схему на две части. Первая часть состоит из определенных значений сигналов, а вторая часть из безразличных значений. Новый Е-фронт генерируется на каждом шагу алгоритма и сохраняется в базе Е-фронтов для этой схемы. На каждом шаге выполняется поиск соответствия Е-фронта, Е-фронтам полученным на предшествующих шагах. Если соответствие найдено, то это означает, что тестовый вектор для этого сбоя был найден на предшествующих шагах.

Одна из возможных и реализуемых в настоящее время возможностей системы – ее статическое и динамическое обучение.

Выводы

В результате проведённого исследования выполнен анализ алгоритмов генерации тестовых векторов, разработана модификация алгоритмов PODEM и FAN с улучшенной эвристикой, использованной для управления обратной трассировкой, реализован программный прототип системы генерации тестов на базе исследуемых (D, PODEM, FAN), а также модифицированного алгоритма FAN.

Предложенный прототип программной системы генерации тестов, позволяет выполнять сравнения различных алгоритмов генерации, за счет незначительного увеличения времени подготовки к поиску следующего теста.