МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ
УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ПРИБОРЫ АВТОМАТИКИ

Всеукраинский межведомственный научно-технический сборник

Основан в 1965 г.

Выпуск 134

Харьков
2006

УДК 62-50

В сборнике представлены результаты исследований, касающихся компьютерной инженерии, управления, технической диагностики, автоматизации проектирования, оптимизированного использования компьютерных сетей и создания интеллектуальных экспертных систем. Предложены новые подходы, алгоритмы и их программная реализация в области автоматического управления сложными системами, оригинальные информационные технологии в науке, образовании, медицине.

Для преподавателей университетов, научных работников, специалистов, аспирантов.

У збірнику наведено результати досліджень, що стосуються комп'ютерної інженерії, управління, технічної діагностики, автоматизації проектування, оптимізованого використання комп'ютерних мереж і створення інтелектуальних експертних систем. Запропоновано нові підходи, алгоритми та їх програмна реалізація в області автоматичного управління складними системами, оригінальні інформаційні технології в науці, освіті, медицині.

Для викладачів університетів, науковців, фахівців, аспірантів.

Свидетельство о государственной регистрации
печатного средства массовой информации КВ № 4619 от 18.10.2000г.

Адрес редакционной коллегии: Украина, 61166, Харьков, просп. Ленина, 14, Харьковский национальный университет радиоэлектроники, комн. 321, тел. 70-21-326

СОДЕРЖАНИЕ

УДК 519.713

А.С. ШКИЛЬ, Е.Е. СЫРЕВИЧ, А.Л. КАРАСЕВ, Д.И. ЧЕГЛИКОВ

ТЕСТОВАЯ ВЕРИФИКАЦИЯ ПОВЕДЕНЧЕСКИХ ЯЗЫКОВЫХ МОДЕЛЕЙ ЦИФРОВЫХ УСТРОЙСТВ

Предлагается процедура верификации моделей цифровых устройств (ЦУ), описанных с помощью языков описания аппаратуры. Основная идея состоит в генерации различающих тестов для отдельных функциональных элементов, суперпозиции этих тестов и в интерактивном вычислении эталонных реакций.

1. Введение

Необходимость исследований в данном направлении обусловлена отсутствием эффективных методов и средств функциональной верификации моделей ЦУ на стадии их описания на поведенческом уровне.

Мировые компании по производству цифровых систем вынуждены постоянно сокращать время до попадания их продукции на рынок. А по оценкам производителей верификация, в том числе и функциональная, занимает до 80% трудоёмкости в цикле проектирования [1]. Существует спрос на средства функциональной верификации моделей цифровых устройств на стадии их описания на поведенческом уровне с помощью конструкций языка VHDL.

Несмотря на обилие научных работ, связанных с верификацией и диагностикой цифровых устройств, в настоящее время весьма актуальны и востребуемы рынком инструментальные средства автоматической генерации тестов для функциональных моделей сложных ЦУ. Дефицит автоматических генераторов тестов нового поколения испытывают и известные фирмы: Aldec, Altera, Actel, Xilinx, Synopsis.

Таким образом, *цель* данного исследования – разработка стратегии верификации моделей ЦУ на языках описания аппаратуры, которая позволит уменьшить временные затраты на проектирование ЦУ путем использования методов сужения области возможных тестовых значений при автоматической генерации тестов.

2. Постановка задачи

Дана модель цифрового устройства на синтезируемом подмножестве языка описания аппаратуры. Синтаксические ошибки в модели исправлены компилятором. Дана спецификация на устройство, на основании которой строилась модель. Исходя из поставленной цели, для ее достижения необходимо решить следующие задачи:

1) разработать внутреннюю модель цифрового устройства, представленного на ЯОА VHDL в целях последующей верификации, а также операции прямой и обратной импликации на этой модели;

2) разработать методики построения различающих последовательностей для различных типов арифметических и логических функций;

3) разработать стратегии функциональной верификации моделей цифровых устройств на этапе ввода и моделирования проекта.

3. Описание стратегии верификации

На сегодняшний день самый распространенный подход к верификации состоит в следующем [2]. Имеются модели M1 и M2, причем M1-эталонная модель, а M2-верифицируемая. Генерируются тесты для модели M1, и в результате моделирования M1 на полученных тестовых наборах определяются эталонные значения (аналогично процессу тестового диагностирования). Затем эти же тесты подаются на M2, их также моделируют и получают экспериментальные ответные реакции. Реакции моделей M1 и M2 сравниваются. Модель M1 эквивалентна M2, если реакции совпадают, и содержит ошибку в противном случае.

Если M1 задана неформальным или частично формальным способом, то описанный выше подход к верификации становится малоэффективным.

В современных технологиях проектирования вместо реального физического устройства используется его представление на ЯОА. Такое представление обычно базируется на данных, полученных из спецификации, т.е. в качестве М1 выступает спецификация, в качестве М2 — описание на ЯОА. Данные о разрабатываемом устройстве в спецификации могут быть представлены разными способами, чаще всего неформальными (текстовые пояснения) или частично формализованными (временные зависимости для отдельных сигналов, алгоритмы работы блоков и т.д.).

Для успешеного проведения верификации обычно составляется план, который определяет порядок проверки правильности функционирования модели устройства. Он призван определить, как проект будет верифицироваться, какие для него будут написаны тесты и как будет решаться задача получения эталонов. Таким образом, план верификации определяет стратегию верификации модели устройства.

В следующем разделе рассматривается стратегия верификации, основанная на графовой модели верифицируемого устройства, функциональных моделях ошибок проектирования и различающих последовательностях. Определим общие положения, на которых строится предлагаемая стратегия:

1) в ходе кодирования спецификации разработчик может внести ошибки не алгоритмического характера, а связанные с «человеческим фактором»;

2) тестовые различающие последовательности активизируют функциональные вершины-операторы;

3) активизация оператора позволяет после сравнения значений в КТ с полученными от разработчика сделать вывод о правильности применения данного оператора;

4) совокупность проверок операторов на правильность их применения позволяет сделать вывод о правильности модели на ЯОА.

4. Построение тестов верификации

Процесс построения тестов верификации основывается на следующих составляющих: внутренняя модель, позволяющая выполнять процедуры прямой и обратной импликации; модели ошибок проектирования; различающие последовательности и алгоритм генерации тестов.

Для построения внутренней модели исходное описание преобразуется в графовую модель, представляющую собой композицию двух графов. Первый – информационный – описывает потоки данных и их преобразование (подобно операционному автомату в классической композиционной модели с микропрограммным управлением) без учета условных ветвей. Второй граф соответствует цепочке условий [3].

Информационный I-граф содержит два типа вершин: операнды и функции. Типы операндов: целые числа integer, беззнаковые векторы типа bit и std_logic. Типы функций ограничены синтезируемым подмножеством VHDL. Дуги соединяют вершины следующим образом: вершина, соответствующая входному операнду и ставшая источником, соединяется с функциональной вершиной, затем дуга выходит из функциональной вершины и входит в вершину, соответствующую выходному операнду и ставшую таким образом приемником. Дуги, исходящие из функциональных вершин и входящие в вершины-приемники, могут быть условные и безусловные. Условные дуги соответствуют операторам, которые находятся внутри условных конструкций VHDL. Они содержат метки, кодирующие условие срабатывания перехода по дуге.

В свою очередь, второй граф (управляющий С-граф) содержит условные конструкции (например, *case*, *if...then...*, *with ... select*) из исходного описания цифрового устройства. С-граф – содержит два типа вершин: условия и метки. Вершины условий содержат вычисляемые условия. Вершины меток – конечные, не имеющие входной дуги. Эти вершины содержат имя метки. Результат моделирования С-графа – набор меток (метка), по которым необходимо выполнять переходы в I-графе.

Для формирования стратегии верификации используем подходы структурного тестирования, основывающиеся на активизации путей в модели устройства, которые после адаптации могут быть использованы и для функциональной верификации. Для этого сформируем систему доказательств, состоящую из определений, утверждения, лемм и теоремы.

Определение 1. Различающими будем называть последовательности, для которых на разных функциях в пределах заданного подмножества один и тот же входной набор (или наборы) будет давать различный результат.

Для того чтобы различить операции f_i от f_j (одинаковой разрядности), необходимо и достаточно, чтобы существовал хотя бы один входной набор (входная последовательность S), на котором они давали бы разные результаты. Такие входные последовательности S будут принадлежать различающим последовательностям.

Определение 2. Идентификация функционального элемента – это определение типа функционала в результате подачи на него различающих последовательностей.

Определение 3. Предшественники – вершины (функциональные или операндовые) информационного графа, являющиеся источниками информации для рассматриваемой вершины.

Операндовые вершины, не имеющие предшественников, будем рассматривать как внешние входы графа.

Определение 4. Преемники – вершины (функциональные или операндовые), получающие информацию от рассматриваемой вершины.

Операндовые вершины, не имеющие преемников, будем рассматривать как внешние выходы графа.

Определение 5. Путь в I-графе – это непрерывная последовательность функциональных и операндовых вершин от выбранного функционала до некоторой контрольной точки информационного графа, реализующая фрагмент режима обработки данных.

Определение 6. Проведение последовательностей значений сигналов через неактивизируемый функционал (как вперед, так и назад) будем называть транспортированием.

Определение 7. Транспортирование различающих последовательностей от рассматриваемого функционала до внешних входов назовем обеспечением. При этом значения сигналов на операндовых вершинах- предшественниках рассматриваемого функционала будем называть условиями активизации.

Определение 8. Функционал является активизированным, если на его входы поданы различающие последовательности, реакции на них выведены на выход функционала, а на линиях, обеспечивающих транспортирование различающих последовательностей от внешних входов графа до входов рассматриваемого функционала, сформированы условия активизации. Результат активизации – идентификация функционала.

Определение 9. Активизированный путь – последовательность преемников от активизированного функционала до контрольной точки либо внешнего выхода, по которому обеспечивается транспортирование реакций функционального элемента на различающие последовательности.

Определение 10. Встроенные операторы – это операторы языка описания аппаратуры, которые являются базовыми для данного ЯОА или находятся в пакетах подпрограмм, поставляемых с программной средой разработки.

Лемма 1. *Если условия активизации и обеспечения функционала* (*активизированного пути*) *существуют, то они могут быть построены с применением процедуры Р* (*прямой и обратной импликации* [4]) *для графа.*

Доказательство. Рассмотрим проблему получения хотя бы одного решения на внешних входах. Примем, что все функциональные элементы – встроенные операторы, т.е. не содержат внутри себя ошибок и работают исправно. Если бы это было не так, то верификация ЯОА-модели включала бы в себя проверку системы моделирования. Кроме того, функциональные элементы позволяют получить реакции на все значения из области определения операндов. При этом при прохождении через функциональный элемент для любого выхода есть хотя бы одна пара входных значений, обеспечивающих заданный выход (для операций «+» и «-» это очевидно; для «*» и «/» на один вход ‘1’, на второй – значение выхода). Это справедливо для входов, на которых не заданы значения. При отсутствии ограничений на линиях - предшественниках всегда присутствует решение – значения на входах графа, позволяющие подвести требуемую различающую последовательность.

Следствие 1. Если для функционала существуют различающие последовательности и соблюдены условия обеспечения и транспортирования, то он может быть активизирован.

Лемма 2. *Активизация всех функционалов I-графа является проверкой механизма обработки данных в ЯОА-модели*.

Доказательство. Совокупность функционалов на активизированном пути определяет закон обработки данных, поступивших от внешних входов. Это очевидно, так как данные от внешних входов до внешних выходов проходят через функциональные элементы-операторы. С другой стороны, можно сказать, что совокупность законов обработки данных соответствует перечню режимов работы устройства, т.е. его спецификации. Аналогичное положение было доказано в [5, 6] по отношению к механизмам обработки данных микропрограммно-управляемых устройств.

Утверждение. *Если условия транспортирования через функциональный элемент f_i совпадают (принадлежат) с различающими последовательностями для функционала f_j (на активизированном пути), то тип функционала* f_j *также можно считать идентифицированным. Такие условия транспортирования назовем* $C^т_р$.

Доказательство. Очевидно, что, если различающие последовательности для i+1 элемента включают в себя результаты прямой импликации различающих последовательностей на i – м элементе, то эти результаты могут быть использованы для активизации i+1 элемента.

Лемма 3. *Все функционалы на активизированном пути идентифицируемы*.

Доказательство. Если при активизации пути использовались условия, включающие $C^т_р$, то все функционалы на активизированном пути можно считать идентифицированными. Для функциональных двухместных элементов, у которых входы обеспечения на момент начала активизации не заданы, условие активизации всегда может быть построено. Если решение существует, то оно может быть получено путем полного перебора возможных значений.

Если при построении активизированного пути условия обеспечения окажутся противоречивыми, то в ЯОА-модели необходимо ввести дополнительную контрольную точку.

Определение 11. Покрытие путями графа – это множество W такое, что каждый функциональный элемент из множества функциональных элементов принадлежит одному из путей W.

Определение 12. Минимальное покрытие путями графа – это нахождение минимального числа путей графа, проходящих по всем функциональным элементам хотя бы один раз.

Теорема. *Для того чтобы идентифицировать все функционалы в информационном I-графе, необходимо и достаточно активизировать все пути в графе, начиная от функциональных элементов 1-го ранга, до внешних выходов или контрольных точек.*

Доказательство. На основании лемм 1 - 3 и следствия 1 очевидно, что активизируя путь от функционального элемента 1-го ранга, активизируются все функционалы, лежащие на этом пути, а значит, они идентифицируемы. Аналогичные результаты для неизбыточных цифровых схем логического уровня получены в [7]. Вопрос избыточности ЯОА-кода, а следовательно I-графа, решается путем введения дополнительных контрольных точек.

Следствие 2. *Идентификация всех функционалов ЯОА-модели проверяет механизм обработки данных модели*.

На основании леммы 2 активизация всех функционалов является проверкой механизмов обработки данных. А совокупность проверенных механизмов обработки данных отвечает проверке на соответствие графовой модели спецификации.

Исходя из введенных определений и доказанных лемм и теоремы, определим общую стратегию верификации, основанную на активизации функциональных элементов, начиная с элементов 1-го ранга. Она состоит в следующем.

1. Активизируется i-й функциональный элемент 1-го ранга. Различающие последовательности подаются напрямую с внешних входов графа. Внешние входы (выходы) информационного графа – это операндовые вершины, являющиеся портами модели на ЯОА.

2. Формируется путь (пути) активизации от активизированного элемента до внешних выходов либо до контрольной точки.

3. Реакции на различающие последовательности транспортируются через функциональные элементы – преемники. Активизация элемента происходит в том случае, если

условия транспортирования через функциональный элемент совпадают с различающими последовательностями для этого элемента.

4. Активизация считается завершенной, если достигнута заданная операндовая вершина или активизация пути далее невозможна.

5. Пункты 1-4 необходимо повторить для всех функционалов 1-го ранга.

6. После окончания активизации функционалов 1-го ранга следующим для активизации выбирается k-й функционал p-го ранга ($p>1$, $p=\overline{1,n}$), не принадлежащий ни одному уже активизированному пути.

7. Активизация идет до тех пор, пока все функциональные элементы не будут активизированы.

Согласно лемме 3 для того, чтобы идентифицировать все функционалы в информационном I-графе, необходимо и достаточно активизировать все пути в графе, начиная от функционалов 1-го ранга до внешних выходов или контрольных точек таким образом, чтобы каждый функциональный элемент из множества функциональных вершин принадлежал одному пути. Активизация пути в свою очередь подразумевает обеспечение условий транспортирования. Условия транспортирования включают в себя входную последовательность, условие обеспечения и закон изменения выхода.

При обеспечении возможно несколько типичных ситуаций.

Ситуация 1 (идеальная) – выходная последовательность полностью совпадает с входной. Это выполняется как для арифметических операций, так и для логических.

Ситуация 2 – закон изменения выхода относительно входа легковычисляем. Например, если для функционала исключающее ИЛИ на обеспечивающем входе задать значение ‘1’ по всем разрядам, то выходная последовательность будет представлять собой инвертированную входную последовательность по всем разрядам.

Ситуация 3 – закон изменения выхода жестко связан с типом функционала и значениями на обеспечивающих входах. При этом выход может быть определен в результате несложных вычислений в ходе прямой импликации. Например, если на обеспечивающем входе арифметической операции сложения – константа, то выход – это значение на входе плюс эта константа.

При формировании алгоритма активизации путей изберем следующую стратегию.

1. Выбирается один из проверяемых операторов.

2. Строится активизированный путь от одного проверяемого оператора и от него до внешнего выхода либо контрольной точки графа таким образом, чтобы покрытие графа путями было минимальным.

3. Пункты 1-2 повторяются до тех пор, пока все функциональные элементы не будут покрыты активизированными путями.

4. Если на i-м шаге активизация следующего элемента невозможна, то необходимо вводить дополнительные контрольные точки на шаге активизации (i-1) функционального элемента.

5. Построение различающих последовательностей.

Если рассматривать описание модели на ЯОА как программу [8], то, с одной стороны, необходимо выполнять верификацию программного обеспечения, а с другой – это не всегда оптимально. Проверка программного кода подразумевает проверку всех режимов на всех данных. Для проверки соответствия кода, описывающего устройства, спецификации на всех возможных данных для всех доступных входов провести верификацию за приемлемое время со 100% полнотой не представляется возможным.

Предположим, что все встроенные операторы являются комбинационными элементами. Это значит, что их проверка потребует подачи 2^n значений, где n – суммарная разрядность входов. Для получения высокого качества верификации за приемлемое время необходимо уменьшить количество подаваемых тестовых наборов (сузить область возможных тестовых наборов).

При построении тестов для функционального элемента необходимо учесть следующее. Как уже было сказано в разделе 2, в качестве функциональных элементов выбраны операторы ЯОА. Учитывая, что операторы языка не содержат внутри себя ошибок,

очевидно, что подача на функциональные элементы тестов проверки исправности является нецелесообразной. Поэтому смысл тестирования примитива состоит не в проверке функционирования, а в идентификации его типа. Таким образом, на примитив необходимо подать такие тестовые наборы, чтобы после анализа реакций на них можно было идентифицировать тип (функцию) примитива и отличить его от других функционалов.

Такие различающие последовательности позволяют найти ошибки, связанные с подменой операторов в ЯОА-коде, причем с подменой логических операторов логическими, арифметических арифметическими, так как перекрестную замену операторов можно выявить в процессе компиляции ЯОА-кода.

Различающие последовательности позволяют идентифицировать тип функционального элемента по его реакциям на заранее определенную входную последовательность.

Рассмотрим построение различающих последовательностей для логических операторов. Пусть количество входов – n. Количество разрядов каждого входа – m. Все операторы ЯОА (кроме инверсии, взятия знака и модуля и возведения в степень) имеют $n \geq 2$. Будем рассматривать следующий набор логических операций – {**and, or, nand, nor, xor, xnor**}. В зависимости от разрядности операндов для логических операций в различающей последовательности может быть от 1-го до 3-х векторов. В качестве примера рассмотрим тесты проверки исправности (ТПИ) для логических элементов, приведенные на рис. 1.

ЗИ

1	2	3	4
0	1	1	0
1	0	1	0
1	1	0	0
1	1	1	1

ЗИ-НЕ

1	2	3	4
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

ЗИЛИ

1	2	3	4
1	0	0	1
0	1	0	1
0	0	1	1
0	0	0	0

ЗИЛИ-НЕ

1	2	3	4
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Рис. 1. ТПИ основных логических элементов

Логические операции над каждым битом выполняются независимо. Можно выделить 3 возможных условия, влияющих на количество векторов в различающих последовательностях: 1) n>=2, m=1; 2) n>=2, m=2; 3) n>=2, m>2.

В первом случае максимальное количество векторов – 3, во втором – 2, в третьем –1.

Различающая последовательность для логических элементов строится после анализа ТПИ функциональных элементов. Различающая последовательность должна удовлетворять требованиям: она должна различать типы логических функциональных элементов друг от друга, количество векторов в различающей последовательности должно быть минимальным.

Эти требования сформулированы на основе определения различающей последовательности и требования к минимальности количества тестов верификации. Рассмотрим один из вариантов построения различающей последовательности для случая n=2, m=1 (частный случай 1-го условия из списка условий, влияющих на количество векторов).

Комбинация “00” различает множество {and, or, xor} от {nand, nor, xnor}.

Комбинация “01” в каждом из подмножеств различает:

– для {and, or, xor} элемент {and} от {or, xor};

– для {nand, nor, xnor} элемент {nand} от {nor, xnor}.

Комбинация “11” в каждом из подмножеств различает:

– для {or, xor} элемент {or} от {xor};

– для {nor, xnor} элемент { nor } от {xnor}.

Таким образом, последовательность {00, 01, 11} идентифицирует тип логического функционального элемента n=2, m=1. Покажем, что эту же последовательность можно использовать для случая n>2, m=1. При этом различающие последовательности необходимо подать на любые два входа. А на остальные входы необходимо установить значения, которые не затрудняли бы анализ выходных значений для данного типа логического элемента, например, “00…00”.

На рис. 2 представлена схема формирования различающих последовательностей для одноразрядных многовходовых логических функциональных элементов. Здесь же показа-

на следующая последовательность действий: на все одноразрядные входы логического элемента подается значение «0». В зависимости от реакции формируются два подмножества. Затем на функционал подается последовательность такая, что один из разрядов (входов) равен «1», а остальные «0». Формируются еще две пары подмножеств. И третья комбинация – это две единицы в любых двух разрядах, и опять формируются две пары подмножеств. В конце концов после подачи последовательности тестов и анализа результатов сформированы 6 подмножеств, содержащих по одному элементу.

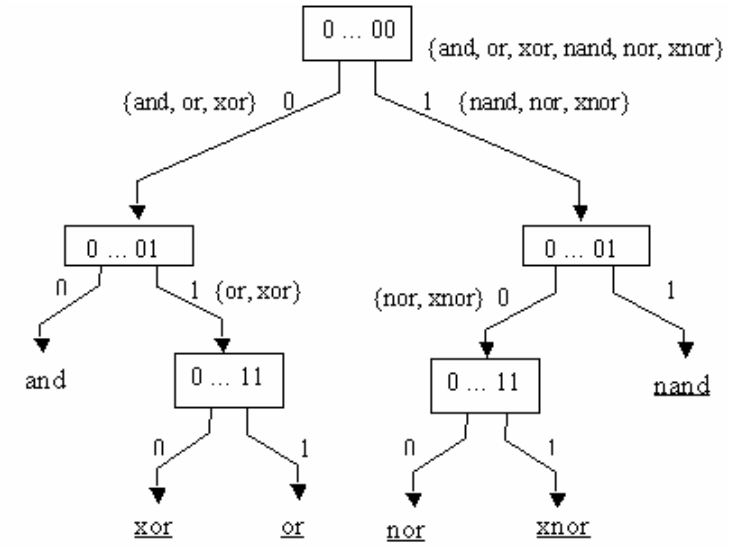

Рис. 2. Различающие последовательности для логических операторов (m=1, n>2)

Многоразрядные логические функциональные элементы оперируют с каждым разрядом операндов независимо от остальных разрядов. Поэтому многоразрядный функциональный элемент можно представить совокупностью одноразрядных. В таком случае можно использовать ту же последовательность {00, 01, 11}. Но в зависимости от разрядности операндов значения из последовательности можно подать, используя 2 (m=2) или 1 (m>2) вектор. На рис. 3 приведены различающие последовательности для m-произвольного, n>2.

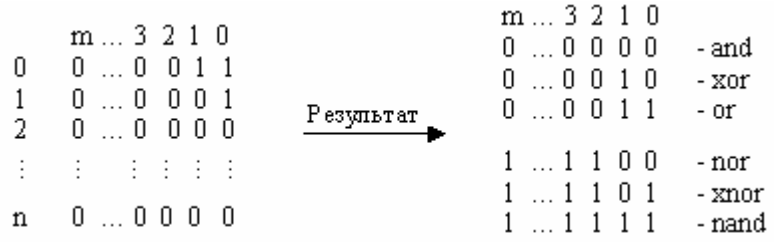

Рис. 3. Различающие последовательности для логических операторов ($m \geq 2$, n>2)

В этом случае формирование последовательности происходит при подаче трех наборов. Слева представлены входные наборы, справа – реакции на них и соответствующие логические операторы.

Рассмотрим алгоритм построения различающих последовательностей для арифметических операций по типу «все от всех» из заданного подмножества арифметических операций {add (+), sub (-), mul (*), div (/)}(рис.4). Арифметические операции можно различить при условии, что разрядности операндов хватит для представления чисел, больших единицы.

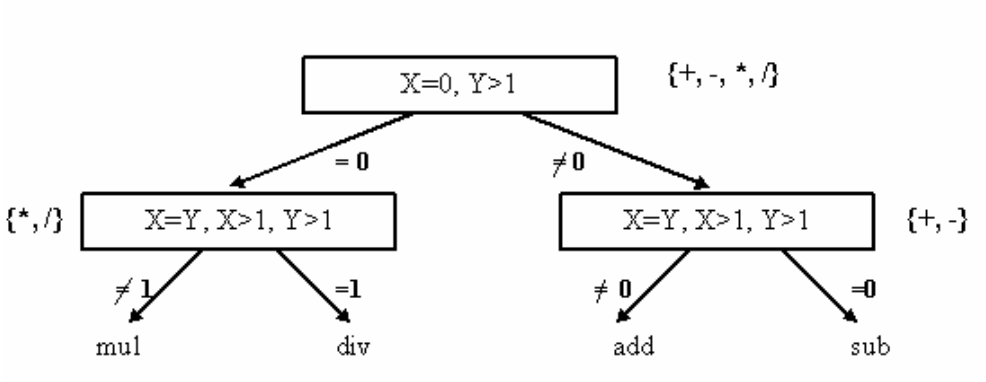

Рис. 4. Различающие последовательности для арифметических операторов

Чтобы различить операцию «плюс» из подмножества {плюс, минус, умножить, разделить}, необходимо на один из входов функционала подать ноль, а на другой – значение больше единицы. Далее разбиваем на подмножества {плюс, минус} и {умножить, разделить} следующим образом: если результат на предыдущем шаге равен 0, то отнесем тестируемый функционал в подмножество {умножить, разделить}; если не равен 0, то {плюс, минус}. Подаем на входы одинаковые значения, большие единицы. Если значение не равно 0, то функционал – операция сложения.

Если различающие последовательности для i+1 элемента включают в себя результаты прямой импликации различающих последовательностей на i – м элементе, то эти результаты могут быть использованы для активизации i+1 элемента, т.е. условие транспортирования через функционал может содержать в себе различающие последовательности.

5. Способы получения эталонных реакций

Проблема получения эталонов при функциональной верификации состоит в том, что спецификация обычно неформальная (нет однозначного соответствия между входными воздействиями и выходными реакциями) и неполная (не все режимы кода описаны). Именно эти две причины и приводят к тому, что в явном виде реакции часто получить оказывается невозможно.

Задача получения эталонов является одной из основных при тестовой верификации. Можно выделить три способа получения эталонов.

1. По спецификации.

В спецификации эталоны заданы явно или неявно. Спецификация на цифровое устройство задается в неформальном виде, обычно в словесной форме. На основании опыта инженер-проектировщик может выделить эталонные реакции или вычислить их, если они заданы в качестве алгоритма функционирования, временных диаграмм и т.д.

2. С использованием внешней моделирующей системы.

Если спецификация содержит алгоритм работы устройств, то можно создать программную моделирующую систему либо эмулятор, позволяющие получить эталонные реакции автоматически путем подачи полученных тестовых воздействий на такую систему. При этом необходимо сделать допущение, что такая внешняя система является идеальной.

3. Итеративный интерактивный процесс.

Основой данного подхода служит предположение о том, что инженер-разработчик для любых входных данных может вручную по спецификации или каким-либо другим способом получить реакции разрабатываемого устройства, если предлагаемые для вычисления данные соответствуют описанным в спецификации состояниям.

Рассмотрим подробнее принцип интерактивности при высчитывании эталонных реакций.

Необходимое условие – это возможность инженера, отвечающего за реализацию ЯОА–кода, просчитать (не на созданном коде, а каким – либо другим образом, например, вручную) эталонные реакции в выходных и некоторых внутренних контрольных точках. Контрольные точки задаются одновременно с составлением тестов. Моделирование тестов и вычисление эталонных значений происходит независимо друг от друга. Результаты моделирования и расчетов «вручную» сравниваются. Результат сравнения позволяет сделать вывод о соответствии ЯОА–модели спецификации.

Последовательность шагов при интерактивном процессе определяется следующей процедурой:

1) получить от инженера-разработчика откомпилированный ЯОА-код, описывающий целое устройство или его логически или функционально законченный блок;

2) выполнить построение тестов по полученному ЯОА-коду путем генерации различающих последовательностей, задать контрольные точки;

3) возвратить полученные тесты инженеру-разработчику для вычисления соответствующих им выходных значений и/или значений в ранее заданных контрольных точках;

4) выполнить моделирование сгенерированных тестов в целях получения экспериментальных значений;

5) сравнить значения, полученные при моделировании ЯОА-кода, и значения, полученные от инженера-разработчика;

6) при несовпадении сделать вывод о наличии ошибок проектирования в ЯОА-коде модели цифрового устройства.

Выводы

Предложен подход к функциональной верификации моделей цифровых устройств на языках описания аппаратуры с использованием специальных различающих тестовых последовательностей. Рассмотрена общая стратегия верификации модели устройства и алгоритм построения тестов. Доказана полнота получаемых тестов и их различающие свойства. Отдельно рассмотрен вариант получения эталонных реакций по неформальной спецификации устройства.

Научная новизна полученных результатов состоит в использовании графовой модели цифрового устройства в виде совокупности сигналов (переменных) и функционалов при проведении функциональной верификации; в разработке методов построения различающих последовательностей для указанных функционалов; в разработке стратегии активизации путей в графовой структуре и доказательстве полноты получаемых тестов.

Практическая ценность работы определяется программной реализацией метода построения тестовых наборов для верификации на основе VHDL-модели верифицируемого устройства и их моделирования в ходе проведения диагностического эксперимента.

Список литературы. 1. *Cohen* B. Real Chip Design and Verification Using Verilog and VHDL, Kluwer Academic Publishers, 2002. **2.** *Bergeron J.* Writing Testbenches: Functional Verification Of Hdl Models, 2003, Kluwer Academic Publishers. 354 p. **3.** *Syrevitch Yev. Yef.* HDL-models verification. TCSET ‘2006 , February 28- March 4. 2006, Slavske, Ukraine. P. 570-573. **4.** *Рустинов В.А., Сыревич Е.Е., Сыревич А.В., Чегликов Д.И.* Процедуры импликации на арифметических операциях при синтезе тестов верификации// Радиэлектроника и информатика. Вып. 130, 2005. С. 4-13. **5.** *Чипулис В.П., Шаршунов С.Г.* Анализ и построение тестов цифровых программно - управляемых устройств. М.: Энергоатомиздат, 1992. **6.** *Шаршунов С.Г.* Разработка функциональных тестов RISC-микропроцессоров // Автоматика и телемеханика. 2004. N 11. С. 174-189. **7.** *Шкиль, А.С., Кизуб В.А., Кривуля Г.Ф.* Генерация тестов в системе автоматизированного проектирования диагностического обеспечения // Управляющие системы и машины. 1987. № 4. С. 44-48. **8.** *Соммервилл Иан.* Инженерия Программного Обеспечения, 6-Е Издание: Пер. с англ. М.: Издательский Дом ”Вильямс”, 2002. 624 с.

Поступила в редколлегию 03.05.2006

Шкиль Александр Сергеевич, доцент кафедры АПВТ ХНУРЭ. Научные интересы: верификация HDL-моделей, логическое моделирование, дистанционное образование. Адрес: Украина, 61726, Харьков, пр. Ленина, 14, тел. 702-13-26.

Сыревич Евгения Ефимовна, аспирантка кафедры АПВТ ХНУРЭ. Научные интересы: верификация HDL-моделей, проектирование на ПЛИС. Увлечения: иностранные языки. Адрес: Украина, 61726, Харьков, пр. Ленина, 14, тел. 702-13-26.

Карасев Андрей Леонидович, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: верификация HDL-моделей, проектирование на ПЛИС, тестирование программных продуктов, программирование на С++. Адрес: Украина, 61057, Харьков, пр. Ленина, 14, тел. 702-13-26

Чегликов Денис Игоревич, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: верификация HDL-моделей, проектирование на ПЛИС. Увлечения: футбол, баскетбол. Адрес: Украина, 61726, Харьков, пр. Ленина, 14, тел. 702-13-26.

Відповідальний випусковий В.І. Хаханов
Редактор О.П. Гужва
Комп'ютерна верстка Г.В. Хаханова, С.В. Чумаченко

Підп. до друку 27.03.2006. Формат 60x84 1/8. Умов. друк. арк. .
Обл.-вид. арк. . Тираж 300 прим.
Зам. № . Ціна договірна.

Харківський національний університет радіоелектроніки (ХНУРЕ).
Україна, 61166, Харків, просп. Леніна, 14.

Оригінал-макет підготовлено і збірник віддруковано
в навчально-науковому видавничо-поліграфічному центрі ХНУРЕ.
Україна, 61166, Харків, просп. Леніна, 14.