УДК 004.415.532.3

Средство автоматизированного тестирования вычислительного блока рекуррентного операционного устройства

В.Н. Волчек, Р.А. Зеленов, А.А. Прокофьев

Учреждение Российской академии наук Институт проблем информатики РАН,
{VVolchek, RZelenov, AProkofyev}@ipiran.ru

***Аннотация* — В статье обозначены общие проблемы, связанные с тестированием рекуррентного обработчика сигналов, а также частные проблемы процесса тестирования вычислительного блока рекуррентного операционного устройства. Описывается разработанное средство автоматизации тестирования вычислительного блока рекуррентного операционного устройства, представлены преимущества использования данного подхода.**

***Ключевые слова* — автоматизация тестирования, вычислительный блок, потоковая архитектура, рекуррентность, ModelSim, VHDL.**

I. Введение

Достоверно известно, что этап тестирования – один из самых важных этапов в разработке. По объекту тестирование разделяется на различные подвиды: функциональное тестирование, тестирование производительности, тестирование совместимости и т.д. Функциональное тестирование является слабым звеном в технологической цепочке проектирования сложного аппаратного обеспечения. Согласно Бергерону (Janick Bergeron) функциональная верификация занимает около 70% общего объема трудозатрат, число инженеров, занимающихся верификацией, примерно вдвое превосходит число проектировщиков, а размер исходного кода тестов (testbenches) достигает 80% общего размера кода проекта [1]. Во многом это связано с необходимостью тестирования на различных этапах производства [2].

Цель любого функционального тестирования – это выявление ошибок. Ошибка, в данном контексте, это несоответствие ожидаемого поведения устройства (описанного в спецификации) фактическому. Если процессу функционального тестирования уделить недостаточно внимания, то, вероятнее всего, существующие в системе ошибки будут обнаружены пользователями. Последствия подобного развития сценария могут быть самыми разными и, как правило, всегда довольно неприятными. Причем чем раньше будет обнаружена ошибка, тем меньше затрат потребуется на ее устранение.

В отделе архитектур перспективных компьютерных систем Института Проблем Информатики РАН (ИПИ РАН) ведется работа по созданию рекуррентного обработчика сигналов (РОС), в рамках которой особое внимание уделяется процессу тестирования. Платформой, на которой создается РОС, как уже неоднократно отмечалось в [3, 4], является ПЛИС Stratix III фирмы Altera с использованием САПР Quartus II и языка описания аппаратуры VHDL. РОС представляет собой технически сложный аппаратный комплекс, базирующийся на новой нетрадиционной (гибридной) архитектуре, сочетающей в себе свойства классических фон Неймановских архитектур со свойствами архитектур потока данных. Архитектуры потока данных (потоковые архитектуры, *dataflow architecture*) относятся к классу нетрадиционных архитектур [5]. В отличие от традиционных архитектур, берущих свое начало от архитектуры фон Неймана, где активным (инициирующим процесс вычисления) является поток команд (*control flow*), в архитектурах потока данных вычисление полностью зависит от доступности входных аргументов, т.е. поток данных является активным.

Учитывая уникальность разработки целесообразно кратко рассмотреть её особенности. Ключевыми особенностями новой архитектуры являются:

1) объединение потока команд и потока данных в единый поток самодостаточных данных;

2) управляющий уровень разрабатываемой архитектуры управляется потоком команд, как в традиционных фон Неймановских системах;

3) поток самодостаточных данных рекуррентно свернут на стадии компиляции и автоматически разворачивается в ходе процесса выполнения задач при помощи специализированного устройства преобразования тегов (ПТ).

Сама по себе архитектура является универсальной и может быть реализована в процессорных устройствах разного назначения. РОС ориентирован на параллельную обработку речевых сигналов в режиме реального времени. Для того, чтобы удовлетворить этим требованиям, РОС (в текущем исполнении) содержит в своем составе четыре вычислительных ядра, что относит этот процессор к классу многоядерных процессоров, специализированных на цифровой обработке сигналов. К настоящему моменту была подана заявка на изобретение в РОСПАТЕНТ. Заинтересованный читатель может более глубоко ознакомиться с разработкой ИПИ РАН в [3, 4].

Подобная сложность аппаратного обеспечения не позволяет за разумное время вручную разрабатывать наборы тестов, обеспечивающие оптимальное тестовое покрытие. Учитывая вышеперечисленные факторы можно утверждать, что процесс функционального тестирования и отладки VHDL-модели РОС представляет собой нетривиальную задачу.

Процесс тестирования, как правило, состоит из нескольких этапов [6]. Перед тестированием всей системы всегда происходит тестирование подсистем. В данном случае это тестирование различных уровней: управляющего (реализован на базе традиционного фон Неймановского процессора) и рекуррентного операционного уровня (РОУ). Каждый уровень соответственно состоит из различных функциональных блоков (ФБ), отдельному тестированию которых также необходимо уделить особое внимание.

В настоящий момент разработчики РОС проводят тестирование РОУ, которое состоит из различных функциональных блоков: буферная память, распределитель, память совпадений, вычислительный блок (ВБ, вычислитель), преобразователь тэгов, коммутатор. В данной статье речь пойдет о решении задачи функционального тестирования VHDL-модели вычислительного блока. Несмотря на то, что данный функциональный блок по своей сути представляет классическое вычислительное ядро цифрового сигнального процессора (ЦСП), некоторые заложенные решения являются сугубо специализированными для рекуррентной потоковой архитектуры и в определенном смысле уникальными для всех цифровых сигнальных процессоров (ЦСП) [3, 4].

II. Постановки задачи

Для подтверждения корректной работы вычислительного блока рекуррентного операционного устройства (ВБ РОУ) необходимо удостовериться в соответствии фактических характеристик, полученных в результате разработки устройства, заявленным требованиям, указанным в спецификации. Ниже перечислены основные характеристики ВБ РОУ, которые необходимо подвергнуть проверке:

1) система команд ВБ РОУ;
2) три режима работы:
    a) обычный (последовательный) режим,
    b) суперскалярный режим I типа,
    c) суперскалярный режим II типа (также именуемый псевдосуперскалярным режимом);
3) внутренние регистры;
4) обработка исключительных и неспецифицированных ситуаций.

Самым трудоемким процессом в задаче тестирования ВБ РОУ является проверка корректности работы системы команд, которая в данный момент включает в себя:

1) 17 арифметических команд,
2) 3 логические команды,
3) 6 команд сдвига,
4) 8 управляющих команд,
5) 6 команд передачи данных.

Каждую команду необходимо проверить на различных диапазонах значений входных данных в соответствии с требованиями ГОСТа [7]. Соответственно каждую команду необходимо также протестировать во всевозможных существующих режимах.

Очевидно, что ручное тестирование одной только системы команд окажется весьма трудоемкой задачей, не говоря уже о тестировании всего ВБ РОУ. В настоящее время на рынке представлены различные технологии и средства автоматизированного тестирования поведенческих моделей на языках описания аппаратуры: AVM компании Mentor Graphics, OVM – совместный продукт компаний Mentor Graphics и Cadence Design Systems, UniTESK – разработка Института Системного Программирования РАН (ИСП РАН) и др. Однако внедрение вышеперечисленных инструментов тестирования не всегда целесообразно.

Большинство промышленных средств автоматизированного тестирования имеют высокую стоимость, поэтому прежде всего необходимо оценить экономическую целесообразность внедрения данного инструментария. Согласно [8], время, затрачиваемое на внедрение автоматизированного аналога не должно превышать 10-12 кратного времени на ручное тестирование. В противном случае автоматизация тестирования не будет экономически оправдана. Первоначальные затраты могут в 8-12 раз превосходить затраты на полноценное ручное тестирование [8].

Кроме того, трудоемкость их внедрения оставляет желать лучшего, особенно в условиях ограниченных финансовых и людских ресурсов. Использование промышленных средств автоматизации тестирования в большинстве случаев требует от пользователей специальных знаний и навыков. Как правило, для этих целей организовывают платные курсы и различные тренинги. Все это требует времени, которое с лихвой окупается в долгосрочной перспективе при успешном выходе на рынок. Однако, если разговор о внедрении в эксплуатацию еще не ведется, использование промышленных средств автоматизированного тестирования не всегда целесообразно.

Кроме того, согласно [9], уникальность разработки может оказаться дополнительным (возможно даже решающим) фактором в принятии решения о внедрении готовых универсальных средств автоматизированного тестирования. Сам научно-исследовательский характер работы требует постоянного внесения изменений в спецификацию проекта, что приводит к постоянным обновлениям программы испытаний, а, следовательно, и обновлениям так называемых тест-кейсов (test cases) в средстве автоматизированного тестирования.

Как результат, учитывая все вышесказанное, разработчики вынуждены были решить задачу создания

собственных специализированных средств автоматизированного тестирования ВБ РОУ с частичным применением ручного тестирования.

III. Реализация задачи

Программы, подготовленные для исполнения на РОУ, называются капсулами. Капсула представляет собой набор элементов самодостаточных данных (операндов), содержащих данные и инструкции для их обработки, а также инструкции настройки режимов работы РОУ. Каждая капсула хранит рекуррентно свернутый алгоритм решения конкретной задачи, который разворачивается в ходе выполнения в РОУ. Процедура рекуррентной свертки представляет собой архивирование поля кода операции в потоке самодостаточных данных. Таким образом, каждый код операции несет в себе информацию о последующем коде (последующих кодах).

Для поддержки задач программирования и отладки модели РОУ была разработана система капсульного программирования и отладки (СКАТ) [10, 11]. СКАТ написан на языке C# с использованием платформы .NET. Данная система позволяет не только программировать и отлаживать капсулы, но и также изначально задумывалась в качестве средства отладки VHDL-модели РОС. Соответственно, вполне логичным выглядит решение дополнить СКАТ средствами автоматизированного тестирования ВБ РОУ.

Согласно [10], формат всех возможных элементов самодостаточных данных – операндов капсулы – подробно описан в виде XML-представления, используемого в среде СКАТ для формирования графического интерфейса конструктора капсул, а также для интуитивно-понятного (user-friendly) отображения результатов моделирования РОС, полученных в системе ModelSim. Это же XML-представление при помощи технологии XSLT преобразуется в HTML страницу, позволяя одновременно использовать внутреннее представление операндов в СКАТ в качестве внешней спецификации на операнды для разработчиков и пользователей системы. Данный подход очень удобен и перспективен, так как позволяет снизить возможность распространения ошибок в форматах операндов в СКАТ или в спецификации.

```
73 <field name="Oc" length="5">
74    <value name="MACac code="01100"
   symbol="*+c" model="L*R+regC"
   result="regC">
75
76       <subDescription id="1">
77 Умножение с накоплением в
78 регистре C
79       </subDescription>
80
81    </value>
82
```

Рис.1. Фрагмент описания системы команд в виде XML-представления для СКАТ

Возможности, предоставляемые технологией XML и системой капсульного программирования и отладки СКАТ, позволили разработчикам РОУ придумать оригинальную реализацию средства автоматизированного тестирования ВБ РОУ.

Основная идея заключается в том, чтобы расширить XML-представление операндов, в котором уже присутствует описание системы команд ВБ РОУ, математическими моделями, описывающими каждую команду ВБ РОУ в терминах внутреннего представления СКАТ. На рис. 1 на примере одной команды показано, как XML-представление выглядит в конечном итоге.

Тэг «field» обозначает поле операнда, формат описания которого представлен внутри данного тэга. В данном случае это поле «Oc» – код операции длиною в 5 разрядов. Тэг «value» обозначает возможные значения данного поля операнда. В примере выше описана команда умножения с накоплением в регистре C. Мнемоника этой команды – «MACac», код операции - «01100», а символьное представление, используемое в графе алгоритма программистом капсулы - «*+c».

Для задачи создания системы автоматизированного тестирования ВБ РОУ в изначальное XML-представление были добавлены атрибуты «model» и «result». Каждый атрибут представляет собой фрагмент кода на языке C#. Атрибут «model» указывает математическую операцию, аналогичную команде ВБ РОУ, а атрибут «result» показывает результирующую переменную внутреннего представления ВБ РОУ в среде автоматизированного тестирования, аналогичную результирующему регистру или сигналу VHDL-модели ВБ РОУ.

По умолчанию система моделирования ModelSim позволяет просматривать результаты выполнения VHDL-модели в виде временных диаграмм [12]. Такой подход считается универсальным, но при тестировании и отладке не всегда оказывается удобным, особенно в технически сложных устройствах с большими разрядностями внутренних сигналов и регистров. СКАТ способен агрегировать результаты моделирования в ModelSim для последующего анализа или преобразования в нужное интуитивно-понятное графическое представление [10]. Для этой задачи в системе СКАТ имеется внутреннее представление регистров РОУ, т.е. каждому физическому регистру РОУ соответствует своя программная переменная в СКАТ. Операции над этими переменными описываются в дополнительных атрибутах XML-представления, речь о которых шла выше.

Таким образом, появляется возможность запустить моделирование VHDL-модели ВБ РОУ в среде ModelSim, агрегировать результаты (моделирования), проанализировать их и выполнить фрагменты кода, указанные в XML-описании. Возможности динамического компилирования языка C# позволяют без труда реализовать эту задачу в рамках системы СКАТ.

По принципу мажоритарного алгоритма результаты выполнения команды в обеих средах сравниваются, и в случае несовпадения результатов моделирования ВБ РОУ в ModelSim с результатами моделирования в СКАТ тестировщику выдается соответствующее сообщение об ошибке. Выдача сообщений об ошибке настраивается опционально. В настоящий момент существуют два режима взаимодействия СКАТ с тестировщиком – режим мгновенной выдачи сообщения об ошибке (в момент обнаружения) с остановкой процесса тестирования и режим выдачи всех ошибок, накопленных в результате выполнения программы испытаний.

На рис. 2 представлена блок-схема алгоритма работы системы автоматизированного тестирования ВБ РОУ.

Генерация тестовых воздействий может происходить как в ручном режиме, при помощи непосредственного программирования капсулы, так и в автоматическом режиме. Данный подход обеспечивает гибкость при отладке как капсулы, так и VHDL-модели, и позволяет верифицировать результаты выполнения программы в РОУ. Наличие графического конструктора капсул в СКАТ позволяет значительно упростить

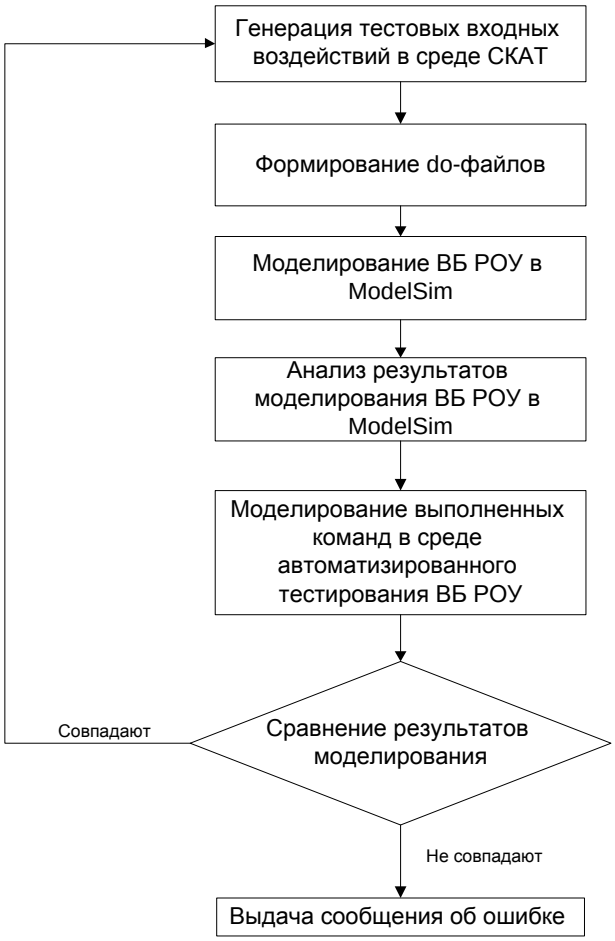

Рис. 2. Блок-схема алгоритма автоматизированного тестирования ВБ РОУ

процедуру создания do-файлов, использующихся для инициации тестовых воздействий в среде ModelSim.

Для организации тестирования в автоматическом режиме возможно применение различных алгоритмов генерации тестовых воздействий. Задача поиска оптимальной методики генерации тестовых программ для верификации ВБ РОУ представляет собой серьезный интерес и выходит за рамки данной статьи.

Существующие методики генерации тестовых воздействий подробно описаны в [13]. На текущий момент в системе СКАТ реализована возможность генерировать тестовые воздействия случайным образом на различных диапазонах значений данных.

При этом входные данные – операнды, подвергающиеся обработке и использующиеся в вычислениях, представляют собой случайные числа, генерируемые в среде автоматизированного тестирования ВБ РОУ. Эти операнды разбиваются на подмножества максимальных, средних и минимальных значений данных. В процессе функционального тестирования наборы полученных данных также случайным образом перебираются из всех подмножеств. В результате покрытие входных данных становится более полным, что позволяет параллельно протестировать обработку переполнений и переносов.

Несмотря на то, что метод генерации тестовых воздействий случайным образом нельзя назвать систематическим, он до сих пор является самым распространенным методом автоматического построения тестовых программ. Этот метод популярен в силу своей простоты. Он позволяет быстро обнаруживать простые ошибки, а также позволяет создать ситуации, которые сложно предугадать, но которые в то же время могут быть весьма интересными для тестирования [13].

Состояние всех внутренних регистров также подвергается сравнению, что позволяет в полном объеме оценить корректность их работы. Было принято решение протестировать возникновение исключительных ситуаций вручную, так как это не составляет большого труда в силу того, что в ВБ РОУ количество исключительных ситуаций невелико, а СКАТ облегчает взаимодействие с ModelSim. Для этих целей в СКАТ были созданы капсулы, содержащие ошибки (например, конфликтные коды операций в суперскалярном режиме или несовместимые коды операций в обычном режиме), на которые ВБ РОУ реагирует исключительной ситуацией. Путем просмотра результатов моделирования выявлялись исключительные ситуации.

После выполнения программы испытаний средство автоматизированного тестирования ВБ РОУ генерирует отчет в табличном виде, позволяющий при необходимости дополнительно проанализировать результаты выполнения тестирования.

IV. Пример использования

В качестве наглядного примера использования средства автоматизированного тестирования рассмотрим простейшую капсулу, выполняющую функцию

$$Ln(1+x) = Ln(2) - (\gamma + \frac{\gamma^2}{2} + \frac{\gamma^3}{3} + \frac{\gamma^4}{4} + \frac{\gamma^5}{5} + \frac{\gamma^6}{6} + \frac{\gamma^7}{7} + \frac{\gamma^8}{8}) = Ln(2) - G \text{, где } \gamma \in (-\frac{1}{2}, 0). \quad (1)$$

натурального логарифма (1). За один вычислительный такт на ВБ РОУ обрабатывается четыре значения логарифма (G_1-G_4).

В результате проверки данной капсулы на VHDL модели РОС в среде ModelSim в режиме «черного ящика» были получены результаты, близкие к оптимальным, но незначительно от них отличающиеся (см. табл. 1).

Таблица 1

Фрагмент протокола тестирования натурального логарифма на РОУ

Входные данные	**Ожидаемые результаты**	**Полученные Результаты**
$Gamma_1 = 6454$ $Gamma_2 = 16002$ $Gamma_3 = 6322$ $Gamma_4 = 13717$	$G_1 = 7187$ $G_2 = 21947$ $G_3 = 7023$ $G_4 = 17769$	$G_1 = 7187$ $G_2 = 21889$ $G_3 = 7021$ $G_4 = 17769$
$Gamma_1 = 13972$ $Gamma_2 = 11170$ $Gamma_3 = 11660$ $Gamma_4 = 15707$	$G_1 = 18211$ $G_2 = 13659$ $G_3 = 14410$ $G_4 = 21378$	$G_1 = 18205$ $G_2 = 13604$ $G_3 = 14410$ $G_4 = 21378$

Средство автоматизированного тестирования ВБ РОУ обнаружило несколько ошибок: в выполнении команды округления результата вычисления и в настройке 40-разрядного регистра блока умножения с накоплением.

В табл. 2 представлен фрагмент отчета, сгенерированный средством автоматизированного тестирования ВБ РОУ, который демонстрирует ошибку с выполнением команды MACac – умножение с накоплением в регистре C. Данная команда является базовой операцией в области цифровой обработки сигналов. Существует два режима работы блока умножения с накоплением – обычный режим и режим насыщения. В режиме насыщения, если значащая часть результата занимает больше 32 разрядов, то результат выставляется в максимально (минимально) возможный. В примере, представленном в таблице 2, соответствующий блок был сконфигурирован без учета режима насыщения, на что отреагировало средство автоматизированного тестирования ВБ РОУ.

В табл. 3 представлен пример некорректной работы команды округления. Что примечательно – ошибка проявлялась не всегда. Такого типа ошибки относятся к классу плавающих ошибок. Как позже выяснилось при разработке VHDL-модели программист использовал метод округления, отличный от требуемого в спецификации.

Вместо использования несмещенной схемы округления была применена смещенная схема. Стандартный метод смещенного округления состоит в прибавлении 1 к последнему разряду округляемого числа. Этот метод вызывает ряд положительных смещений, начиная с середины младшей части (0x8000), округляя результат вверх. При применении несмещенной схемы округления это смещение устраняется. Младшая часть округляемого числа, включая среднее значение (0x8000), не влияет на результат округления. При этом четные значения округляются вниз, а нечетные вверх. Таким образом получается однородная последовательность смещений, что и было зафиксировано в результате тестирования.

После исправления данной ошибки фактические результаты совпали с ожидаемыми.

V. Заключение

Создание средства автоматизированного тестирования на базе существующей системы СКАТ позволило с минимальными временными затратами осуществить задачу тестирования ВБ РОУ. Наличие единого XML-описания, используемого для целого ряда отдельных задач, таких как: создание капсулы, анализ результатов моделирования в ModelSim, создание HTML-документации пользователя, автоматизация тестирования, – позволяет избежать всевозможного дублирования, что, в свою очередь, сокращает объемы работ и уменьшает риск возникновения ошибок. Это еще раз подтверждает эффективность и универсальность решений, заложенных в СКАТ на этапе проектирования.

Таблица 2

Фрагмент табличного отчета с результатами работы средства автоматизированного тестирования ВБ РОУ. Ошибка в настройке блока умножения с накоплением

Мнемоника	Результат в СКАТ	Результат в ModelSim	Статус результата
`MACac 0x4F70, 0x0040;` `regC = 0x00E1700000;`	`regC = 0x00E183DC00`	`regC = 0x00E183DC00`	Верно
`MACac 0x4F70, 0x90CE;` `regC = 0x00E1700000;`	`regC = 0x00FFFFFFFF`	`regC = 0x010E5EEC20`	Неверно

Таблица 3

Фрагмент табличного отчета с результатами работы средства автоматизированного тестирования ВБ РОУ. Ошибка в команде округления результата вычисления

Мнемоника	Результат в СКАТ	Результат в ModelSim	Статус результата
`RNDa 0x00E17000DE`	`regA = 0x00E1700000`	`regA = 0x00E1700000`	Верно
`RNDa 0x00FE138000`	`regA = 0x00FE130000`	`regA = 0x00FE240000`	Неверно

В ходе тестирования ВБ РОУ были выявлены некоторые незначительные ошибки в VHDL-модели, которые в дальнейшем были устранены в процессе отладки. Как уже отмечалось в [3], демонстрационной задачей, призванной показать работоспособность законченного РОС, будет задача распознавания слов диктора. Капсулы для реализации этой задачи в данный момент находятся на завершающем этапе разработки.

Выполнение ряда капсул, уже завершенных на текущий момент на модели РОУ, показало, что фактическая работа вычислителя соответствует ожидаемой, указанной в спецификации, что позволяет утверждать об эффективности проведения тестирования данного функционального узла. После проведения моделирования VHDL-модель РОУ была синтезирована в ПЛИС Stratix III и проверена на комплекте разработчика Stratix III Development Kit, произведенном фирмой Altera [14]. Результаты выполнения ряда капсул (завершенных на данный момент) из программы распознавания слов диктора в аппаратуре оказались верными и совпали с результатами программного моделирования.

Создание первого демонстрационного макета РОС аппаратной реализации запланировано на последний квартал 2012 года. Полученные в ходе тестирования результаты позволяют с оптимизмом утверждать, что данная задача к обозначенным срокам будет решена.

Благодарности

Выражаем благодарность заведующему отделом 22 «Архитектур перспективных компьютерных систем» кандидату технических наук Степченкову Юрию Афанасьевичу, а также научным сотрудникам отдела: Петрухину Владимиру Сергеевичу, Шнейдеру Александру Юльевичу и Хилько Дмитрию Владимировичу за рецензирование данной публикации.

Работа выполнена при частичной финансовой поддержке по Программе фундаментальных исследований ОНИТ РАН на 2011 г. Проект 1.5.

Литература

[1] Bergeron, Janick. Writing testbenches: functional verification of HDL models. Kluwer Academic Publishers, 2000.

[2] Губенко Я.С., Камкин А.С., Чупилко М.М. Сравнительный анализ современных технологий разработки тестов для моделей аппаратного обеспечения – Институт системного программирования РАН,2009. URL: http://www.citforum.ru/SE/testing/hardware_models/ (дата обращения: 7.06.2011)

[3] В.Н.Волчек, Степченков Ю.А., Петрухин В.С., А.А. Прокофьев, Р.А. Зеленов. Цифровой сигнальный процессор с нетрадиционной рекуррентной потоковой архитектурой. // Проблемы разработки перспективных микро- и наноэлектронных систем – 2010. Сборник трудов. – М.: ИППМ РАН,2010.-694 с.

[4] Ю.А. Степченков, В.Н. Волчек, В.С. Петрухин, А.А. Прокофьев, Р.А. Зеленов. Механихмы обеспечения поддержки алгоритмов цифровой обработки речевых сигналов в рекуррентном обработчике сигналов. // Системы и средства информатики. Выпуск 20. Номер 1. – М.: ИПИ РАН, 2010. - С. 31–47.

[5] Ali R. Hurson, Krishna M. Kavi: Dataflow Computers: Their History and Future. Wiley Encyclopedia of Computer Science and Engineering 2008. URL: http://csrl.unt.edu/~kavi/Research/encyclopedia-dataflow.pdf (дата обр.: 08.06.2012)

[6] В. Ематин, Б. Позин. Автоматизированное тестирование при разработке ПО. URL: http://citforum.ru/programming/digest/testirovanie/ (дата обр.: 14.01.2012).

[7] ГОСТ 19.301-2000 ЕСПД. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества.

[8] В. Панкратов, Р. Борук, Д. Лысенко. Автоматизированное тестирование ПО. URL: www.it4business.ru/docs/qa/QAExpert_Test_Automation.ppt (дата обр.: 14.05.2012).

[9] Петр Можаев. Средства автоматизированного тестирования. URL:http://www.osp.ru/os/2009/03/8161608/ (дата обр.: 14.01.2012).

[10] Зеленов Р.А., Степченков Ю.А., Волчек В.Н., Петрухин В.С., Прокофьев А.А., Хилько Д.В. Система капсульного программирования и отладки (СКАТ) // Свидетельство об официальной регистрации программы для ЭВМ № 2010610023 от 20.01.2010.

[11] Р.А. Зеленов, Ю.А. Степченков, В.Н. Волчек, Д.В. Хилько, А.Ю. Шнейдер, А.А. Прокофьев. Система капсульного программирования и отладки. // Системы и средства информатики. Выпуск 20. Номер 1. – М.: ИПИ РАН, 2010. - С. 24–30.

[12] Mentor Graphics. ModelSim SE User's Manual. URL:http://www.actel.com/documents/modelsim_tutorial_ug.pdf (дата обр.: 30.01.2012)

[13] А.С. Камкин. Генерация тестовых программ для микропроцессоров. // Труды Института системного программирования РАН, том 14. Часть 2. – М.: ИСП РАН, 2008. - С. 23–64.

[14] Altera, Stratix III Device Handbook. URL: http:// www. altera.com/literature/lit-stx3.jsp (дата обращения: 29.01.2012).