VIII Всероссийская научно-техническая конференция ПРОБЛЕМЫ РАЗРАБОТКИ ПЕРСПЕКТИВНЫХ МИКРО- и НАНОЭЛЕКТРОННЫХ СИСТЕМ – 2018

СБОРНИК ТРУДОВ. Выпуск II



УДК 621.3.049.77:658.512

Проблемы разработки перспективных микро- и наноэлектронных систем – 2018. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2018. Выпуск II. 198 с.

Выпуск II настоящего периодического издания составлен по материалам секций 4 и 7 («Верификация и тестирование», «Проектирование цифровых функциональных блоков СБИС и подсистем СБИС») VIII Всероссийской научно-технической конференции «Проблемы разработки перспективных микро- и наноэлектронных систем – 2018» (Москва, Зеленоград, 01.10-05.10.2018 г.). Представленные работы выполнены научными сотрудниками и аспирантами РАН, специалистами российских научно-производственных организаций и предприятий, преподавателями, научными сотрудниками, аспирантами и студентами высших учебных заведений, а также сотрудниками ряда зарубежных компаний, лидирующих в области проектирования микро- и наноэлектронных изделий. Издание предназначено для научных работников, специалистов, аспирантов и студентов, занимающихся проблемами разработки, анализа, тестирования сложных микро- и наноэлектронных схем и систем и соответствующих программных средств.

Материалы издания отражают современное состояние российской микро- и наноэлектроники, методов и средств разработки микро- и наноэлектронных схем и систем и являются важным источником информации по перспективным направлениям исследований и инвестиций в области микро- и наноэлектроники.

Все статьи прошли рецензирование и одобрены Редакционной коллегией сборника, Организационным и Программным комитетами конференции.

Конференция проводится при поддержке Российского фонда фундаментальных исследований (проект № 18-07-20078\18).

Ответственный за выпуск к.т.н., с.н.с. Ходош Л.С.

Издается с 2005 года. Включено в систему Российского индекса научного цитирования. Аннотации на русском и английском языках находятся в свободном доступе на сайтах конференции (http://www.mesconference.ru) и научной электронной библиотеки (http://elibrary.ru). Электронные версии полнотекстовых статей с 2010 г. по 2018 г. размещены на сайте конференции в разделе «APXИВ». Там же расположен полный каталог статей с 2005 по 2018 г.

Данное периодическое издание включено в Перечень ВАК российских рецензируемых научных журналов и периодических изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученых степеней доктора и кандидата наук.

ПОДПИСКА:

- по каталогу «Издания органов научно-технической информации» АО «Агентство Роспечать» (индекс 59883);
- в редакции Сборника (тел./факс: (499)729-9208).

ISBN 978-5-94627-157-8 ISBN 978-5-94627-159-2 (Выпуск II) ISSN 2078-7707

> © Федеральное государственное бюджетное учреждение науки Институт проблем проектирования в микроэлектронике Российской академии наук

Организационный комитет

| Стемпковский А.Л. | - Председатель, академик РАН, д.т.н., проф., ИППМ РАН |
|-------------------|---|
| Гаврилов С.В. | - заместитель Председателя, д.т.н., проф., ИППМ РАН |
| Ходош Л.С. | - <i>Ученый секретарь</i> , к.т.н., с.н.с., ИППМ РАН |
| Члены Оргкомитета | |
| Беспалов В.А. | - д.т.н., проф., НИУ «МИЭТ» |
| Бобков С.Г. | - д.т.н., ИППМ РАН |
| Борискин В.С. | - ИППМ РАН |
| Быков В.А. | - д.т.н., проф., ООО «НТ-МДТ СИ» |
| Верба В.С. | - члкорр. РАН, д.т.н., проф., АО «Концерн радиостроения Вега» |
| Зайцев В.В. | - Казенное предприятие города Москвы «Корпорация развития Зеленограда» |
| Ким А.К. | - к.т.н., ПАО «ИНЭУМ им. И.С. Брука» |
| Ковалев А.А. | - д.т.н., АО «Зеленоградский нанотехнологический центр» |
| Курейчик В.М. | - д.т.н., проф., ТТИ ЮФУ, г. Таганрог |
| Мальцев П.П. | - д.т.н., проф., ИСВЧПЭ РАН |
| Медведев А.М. | - к.э.н., Министерство науки и высшего образования РФ |
| Новожилов А.Е. | - Префектура Зеленоградского АО г. Москвы |
| Павлюк М.И. | - АО «ПКК Миландр» |
| Панченко В.Я. | - академик РАН, д.фм.н., проф., РФФИ |
| Петричкович Я.Я. | - д.т.н., проф., АО НПЦ «ЭЛВИС» |
| Сауров А.Н. | - академик РАН, д.т.н., проф., ИНМЭ РАН |
| Северцев В.Н. | - д.т.н., ИППМ РАН |
| Соколов И.А. | - академик РАН, д.т.н., проф., ФГУ «ФИЦ ИУ» РАН |
| Суетин Н.В. | - д.фм.н., Фонд «Сколково» |
| Телец В.А | - д.т.н., проф., ИЭПЭ НИЯУ «МИФИ» |
| Филачев А.М. | - члкорр. РАН, д.т.н., проф., АО «НПО «Орион» |
| Чаплыгин Ю.А. | - академик РАН, д.т.н., проф., НИУ «МИЭТ» |
| Шахнов В.А. | - члкорр. РАН, д.т.н., проф., МГТУ им. Н.Э.Баумана |
| Щелоков А.Н. | - к.фм.н., доц., ИППМ РАН |

Программный комитет

| Иванников А.Д. | Председатель, д.т.н., проф., ИППМ РАН |
|-------------------|--|
| Члены программног | о комитета: |
| Бибило П.Н. | - д.т.н., проф., ОИПИ НАН Беларуси |
| Гаврилов С.А. | - д.т.н., проф., НИУ «МИЭТ» |
| Джиган В.И. | - д.т.н., доц., ООО «Техкомпания Хуавэй» |
| Зольников В.К. | - д.т.н., проф., «ВГЛУ им. Г.Ф. Морозова», г. Воронеж |
| Константинов И.С. | д.т.н., проф., Белгородский государственный национальный исследовательский университет |
| Коротков А.С. | д.т.н., проф., Санкт-Петербургский политехнический университет Петра Великого |
| Кулагин В.П. | - д.т.н., проф., МИРЭА - российский технологический университет |
| Курейчик В.В. | - д.т.н., проф., ТТИ ЮФУ, г. Таганрог |
| Ложкин С.А. | - д.фм.н., проф., МГУ им. М.В. Ломоносова |
| Марченко А.М. | - д.т.н., доц., Mentor Graphics |
| Меликян В.Ш. | - члкорр. НАН Армении, д.т.н., проф., ЗАО «Синопсис Армения», Армения, Ереван |
| Переверзев А.Л. | - д.т.н., доц., Институт микроприборов и систем управления МИЭТ |
| Петросянц К.О. | - д.т.н., проф., МИЭМ НИУ ВШЭ |
| Пугачёв А.А. | - к.т.н., АО «НПП «Пульсар» |
| Русаков С.Г. | - члкорр. РАН, д.т.н., проф., ИППМ РАН |
| Рыжов А.П. | - д.т.н., проф., МГУ им. М.В. Ломоносова |
| Стенин В.Я. | - д.т.н., проф., НИЯУ «МИФИ» |
| Стешенко В.Б. | - к.т.н., доц., АО «Российские космические системы» |
| Чумаков А.И. | - д.т.н., проф., НИЯУ «МИФИ» |
| Шагурин И.И. | - д.т.н., проф., НИЯУ «МИФИ» |

Перечень докладов сессии «Презентации научно-технических достижений российских и зарубежных компаний, а также организаций, способствующих развитию микроэлектроники и информационных технологий в России»

- 1. Искусственный интеллект. Когнитивные процессоры. Новые вызовы. Я.Я. Петричкович, д.т.н., проф., Генеральный директор (АО НПЦ «ЭЛВИС»)
- МЭМС-ФАУНДРИ. Разработка и производство интеллектуальных датчиков и сенсоров. Рынки гражданских применений.
 А.А. Ковалев, д.т.н., Генеральный директор (АО «ЗНТЦ»)
- Текущее состояние и дальнейшее развитие маршрута проектирования СБИС компании Mentor Graphics.
 А.В. Рабоволюк, директор направления «Верификация систем на кристалле» (АО «Мегратек»)
- 4. Решения в области прототипирования и эмуляции от компании Synopsys. С.А. Белоусов, ведущий инженер по применению САПР (Synopsys, Inc.)
- 5. Аспекты радиационной стойкости синтезаторов частоты на основе ФАПЧ. Л.Е. Переверзев, Технический директор (ООО «Альфачип»)
- Распределенная гетерогенная (PLC/RF) система связи для коммуникации машинамашина (M2M), основанная на стеке протоколов IPv6 и построенная на отечественном сигнальном процессоре.
 Ю.О. Мякочин, директор ЦП РЭА (АО «ПКК Миландр»)
- 7. Интегрально-оптические мультиплексоры для спектрального уплотнения потоков данных в волоконно-оптических линиях связи и в коммуникационных устройствах. И.К. Корепанов, руководитель проекта (АО «ЗНТЦ»)

Перечень аналитических докладов

Прокопенко Н.Н., д.т.н., проф.

Аналоговые интерфейсы на основе дифференциальных и мультидифференциальных операционных усилителей: проблемы проектирования и пути их решения.

Петросянц К.О., д.т.н., проф.

Состояние работ в области TCAD и SPICE моделирования элементов БиКМОП БИС с учетом влияния радиации и температуры.

Бычков И.Н., д.т.н.

Разработка и применение прототипов и стендов для верификации процессоров.

Лесников В.А., к.т.н., доц. Синтез цифровых фильтров с бесконечной импульсной характеристикой: проблемы и их решения.



Фонд инфраструктурных и образовательных программ – один из крупнейших институтов развития инновационной инфраструктуры в России. Фонд создан на основании закона «О реорганизации Российской корпорации нанотехнологий».

Деятельность Фонда нацелена на поддержку и развитие всех российских предприятий наноиндустрии по таким направлениям, как развитие технологической инфраструктуры, кадрового потенциала,

стимулирование спроса, стандартизация и сертификация новой продукции, совершенствование законодательства, популяризация высоких технологий.

Фонд поддерживает и развивает сеть нанотехнологических центров. Совместно с партнерами Фонд создает технологические инжиниринговые компании, которые по заказу сторонних компаний разрабатывают и внедряют оригинальные технологии для различных отраслей промышленности. Для расширения сферы применения инноваций Фонд совершенствует систему технических регламентов и стандартов.

Образовательные программы, создаваемые при поддержке Фонда, нацелены на подготовку высококвалифицированных кадров для наноиндустрии. На базе вузов при поддержке ФИОП открыты 155 образовательных программ, по которым прошли профессиональную переподготовку более 54 тысяч выпускников высшей школы и сотрудников промышленных предприятий. Межвузовская программа подготовки инженеров в сфере высоких технологий, запущенная совместно с мэрией Москвы, объединила ведущие высшие учебные заведения: НИЯУ «МИФИ», НИТУ «МИСиС», РАНХиГС. В программе «Школьная лига РОСНАНО», внедряющей эффективные технологии естественно-научного образования, участвуют более 900 школ в 73 регионах России.

Тел.: +7 495 988 5388; E.: info@rusnano.com; http://www.rusnano.com/infrastructure

Главный редактор

А.Л. Стемпковский, академик РАН, д.т.н., проф.

Редакционная коллегия: Бибило П.Н., д.т.н., проф. Бобков С.Г., д.т.н. Гаврилов С.В., д.т.н., проф. Джиган В.И., д.т.н, доц. Зольников В.К., д.т.н., проф. Иванников А.Д. (заместитель главного редактора), д.т.н., проф. Константинов И.С., д.т.н., проф. Коротков А.С., д.т.н., проф. Кулагин В.П., д.т.н., проф. Курейчик В.В., д.т.н., проф. Мальцев П.П., д.т.н., проф. Марченко А.М., д.т.н., доц. Меликян В.Ш., д.т.н., проф., чл.-корр. НАН Армении Переверзев А.Л., д.т.н., доц Петросянц К.О., д.т.н., проф. Пугачёв А.А., к.т.н. Русаков С.Г., чл.-корр. РАН, д.т.н., проф. Рыжов А.П., д.т.н., проф. Стенин В.Я., д.т.н., проф. Стешенко В.Б., к.т.н., доц. Чумаков А.И., д.т.н., проф. Шагурин И.И., д.т.н., проф.

Адрес редакции: 124365, Москва, Зеленоград, ул. Советская, д. 3, ИППМ РАН. Тел./Факс: 8-499-729-9208 E-mail: ippm@ippm.ru http://www.mes-conference.ru

СОДЕРЖАНИЕ

| Организационный комитет | . III |
|--|-------|
| Программный комитет | IV |
| Перечень докладов сессии «Презентации научно-технических | |
| достижений российских и зарубежных компаний, а также | |
| организаций, способствующих развитию микроэлектроники и | |
| информационных технологий в России» | V |
| Перечень аналитических докладов | . VI |
| О Фонде инфраструктурных и образовательных программ | . VII |
| Предисловие к II выпуску издания | XI |

Верификация и тестирование

| А.С. Камкин, А.С. Проценко, С.А. Смолов, А.Д. Татарников Генератор тестовых программ для архитектуры RISC-V | ~ |
|---|----|
| на основе инструмента MicroTESK | .2 |
| <i>А.А. Сохацкий</i> Практические аспекты формальной верификации проектов блоков сетевых СБИС1 | 16 |
| А.П. Евдокимов, В.Г. Рябцев, А.В. Меликов | |
| Принципы проектирования устройств тестового диагностирования быстродействующих микросхем и модулей полупроводниковой памяти2 | 23 |
| <i>А.В. Смирнов, П.А. Чибисов</i> Генератор тестов для проверки когерентности кэш-памятей многоядерных микропроцессоров (ristretto)3 | 31 |
| <i>А.С. Шалумов, Д.Н. Травкин, М.В. Тихомиров</i> Виртуальные испытания микро- и наноэлектронных систем на внешние воздействия | 39 |
| <i>А.Д. Иванников</i> Разработка и исследование моделей блоков цифровых систем на основе их представления в виде семейства стационарных динамических систем4 | 46 |
| <i>П.А. Чибисов, Н.А. Гревцев</i> Подход к стохастическому тестированию RTL-моделей многоядерных микропроцессоров | 52 |
| С.Г. Мосин | |
| Метод снижения размерности обучающих наборов при построении нейроморфного справочника неисправностей для аналоговых интегральных схем5 | 59 |
| М.С. Ладнушкин | |
| Метод дублирования триггеров в средствах тестирования с компрессией | 54 |

| <i>А.В. Кобыляцкий, Д.К. Сергеев</i> Методы верификации на кристалле задержек распространения стандартных цифровых |
|--|
| элементов |
| <i>А.В. Андрианов</i> Методы обеспечения переносимости тестовых сценариев между различными верификационными окружениями |
| <i>О.И. Эсула, М.Е. Барских</i> Верификация алгоритма арбитража потоков запросов к памяти |
| <i>А.С. Щербаков</i> Быстрый алгоритм нахождения доступных вершин графа управления при ограничениях траекторий |
| <i>Ю.А. Татарников</i> Использование формального метода для улучшения покрытия проекта оцениваемого с помощью метрики «code coverage» |
| Проектирование цифровых функциональных блоков СБИС и подсистем СБИС |
| <i>А.И. Власов, П.В. Григорьев, В.А. Шахнов</i> Разработка конструкции гибридных сенсорных мультисборок с элементами радиочастотной идентификации |
| <i>А.В. Антонюк, П.В. Степанов</i> Анализ потребляемой мощности схем суммирования сигналов сопоставления КМОП 65-нм регистров ассоциативной памяти |
| А.В. Ларионов, О.Н. Буякова, О.В. Сысоева, С.Э. Осина, С.О. Задябин, П.А. Алексан, И.В. Тарасов, Ю.Б. Рогаткин, В. Мастеров Четырехканальный мультистандартный адаптивный последовательный приемопередатчик для диапазона 1.25-10.3Гб/с по технологии КМОП 65нм |
| <i>Л.А. Щигорев</i> Развитие структуры и алгоритма работы устройства встроенного саморемонта статической оперативной памяти |
| А.А. Старых, Е.Б. Лукьяненко Самосинхронный D-триггер с «защелкой» |
| <i>И.В. Егоров</i> Способ организации автомата Мура с повышенной устойчивостью к мягким отказам136 |
| <i>В. Жмылев</i> Детектор свободных участков радиочастотного спектра144 |
| <i>А.Н. Якунин, Аунг Мьо Сан</i> Повышение скорости работы многоразрядного двоичного умножителя |
| <i>Е.В. Ливенцев, А.Л. Переверзев, Е.В. Примаков, Д.В. Рыжкова, А.М. Силантьев</i> Цифровой измеритель частоты с повышенной точностью и быстродействием для доплеровского измерителя скорости |
| <i>И.Ю. Сысоев, Н.Н. Хайло, Д.И. Воронков, А.В. Руткевич, А.А. Вейков</i> Опыт разработки радиационно-стойкого контроллера накопителя для бортовой космической аппаратуры |
| И.А. Соколов, Ю.В. Рождественский, Ю.Г. Дьяченко, Ю.А. Степченков, Н.В. Морозов, Д.Ю. Степченков, Д.Ю. Дьяченко Нечувствительный к залержкам блок умножения-сложения-вычитания с плавающей точкой 170 |
| А.А. Шевченко, Р.О. Масленников, М. Махлышев, Р.С. Кобяков Сложнофункциональный блок сетевого коммутатора Ethernet для радиорелейной |
| системы связи |

Именной указатель авторов статей196

Предисловие ко II выпуску издания

В выпуск II настоящего издания вошли доклады секции 4 «Верификация и тестирование» и секции 7 «Проектирование цифровых функциональных блоков СБИС и подсистем СБИС».

Верификация и тестирование. Всего включено 15 докладов 20 авторов, представляющих ФГУ «ФНЦ НИИСИ РАН», ИППМ РАН, Московский физико-технический институт, НИЯИ МИФИ, Сиско Системс Инк., ОАО НПЦ «ЭЛВИС», ЗАО НТЦ «Модуль», Институт системного программирования РАН, Казанский федеральный университет, Университет Мельбурна, Волгоградский государственный аграрный университет, НИИ «АСОНИКА». Также на данной секции представлен аналитический доклад д.т.н. И.Н. Бычкова И.Н. «Разработка и применение прототипов и стендов для верификации процессоров».

Проектирование цифровых функциональных блоков СБИС и подсистем СБИС. Включено 12 докладов 38 авторов, ИПИ РАН, НИИСИ РАН, Национальный исследовательский университет «МИЭТ», ЗАО НТЦ «Модуль», МГТУ им. Н.Э.Баумана, С-Пб. ПУ Петра Великого, ООО «НПП «Цифровые решения», ООО «Радио Гигабит», ННГУ им. Н.И.Лобачевского.



Генератор тестовых программ для архитектуры RISC-V на основе инструмента MicroTESK

А.С. Камкин^{1, 2, 3, 4}, А.С. Проценко¹, С.А. Смолов¹, А.Д. Татарников^{1, 4}

¹Институт системного программирования им. В.П. Иванникова РАН, г. Москва

²Московский государственный университет имени М.В. Ломоносова, г. Москва

³Московский физико-технический институт, г. Москва

⁴Национальный исследовательский университет «Высшая школа экономики», г. Москва

{kamkin, protsenko, smolov, andrewt}@ispras.ru

Аннотация — В работе рассматривается генератор тестовых программ, предназначенный для верификации микропроцессоров с архитектурой RISC-V. Генератор разработан на основе инструмента MicroTESK и состоит из формальных спецификаций архитектуры RISC-V и архитектурно независимого ядра. Спецификации задают синтаксис и семантику команд. Ядро реализует техники построения последовательностей команд и генерации данных. Генерация осуществляется на основе шаблонов, описывающих структурные и поведенческие свойства программ. Инструмент позволяет расширять систему команд и поддерживает случайные, комбинаторные и основанные на ограничениях техники генерации.

Ключевые слова — микропроцессор; система команд; формальная спецификация; верификация; генерация тестовых программ; RISC-V; nML; MicroTESK.

I. Введение

RISC-V – это открытая архитектура, основанная на концепции RISC [1]. Архитектура разработана в Калифорнийском университете в Беркли в 2010 г. В настоящее время основным институтом ее развития и стандартизации является RISC-V Foundation [2]. Ключевыми особенностями архитектуры RISC-V являются открытость и расширяемость. Над созданием микропроцессоров с архитектурой RISC-V работают такие компании, как NVIDIA [3] и Samsung [4]; совместные исследования в этой области ведутся в Швейцарской высшей технической школе Цюриха и в Болонском университете [5].

При проектировании микропроцессоров неизбежны ошибки, поэтому верификация является приоритетной задачей. Одним из основных подходов к верификации является исполнение *тестовых программ* на RTLмодели микропроцессора и сравнение результатов исполнения с данными, полученными на эталонном симуляторе [6]. Тестовые программы создаются с помощью *генераторов тестовых программ*. Важным требованием к генераторам для микропроцессоров RISC-V является возможность адаптации к различным версиям архитектуры, включающим разные наборы расширений. На наш взгляд, наиболее подходящее решение основано на использовании формальных спецификаций [7]. При таком подходе описание команд отделено от логики генерации тестов и может быть модифицировано без изменения ядра генератора.

В работе рассматривается генератор MicroTESK for RISC-V [8], разработанный с помощью инструмента MicroTESK (Microprocessor TEsting and Specification Kit) [9-10]. Генератор состоит из двух частей: архитектурно независимого ядра (библиотечных компонентов) и формальных спецификаций команд RISC-V на языке nML [11]. Такое устройство генератора снижает затраты на разработку и упрощает адаптацию к изменениям в архитектуре.

Оставшаяся часть статьи организована следующим образом. В разделе II рассматриваются существующие решения в области генерации тестовых программ для микропроцессоров. Раздел III посвящен генератору MicroTESK for RISC-V; здесь описываются базовые принципы используемого подхода и особенности его применения к архитектуре RISC-V. В разделе IV приводятся данные о генераторе: поддерживаемые команды, объем спецификаций, трудоемкость разработки. Раздел V резюмирует работу и описывает направления дальнейших исследований.

II. Обзор существующих решений

Сотрудниками Калифорнийского университета в Беркли, участвующими в разработке RISC-V, был создан набор тестовых программ для проверки правильности реализации системы команд [12]. Эти программы тестируют каждую команду в отдельности и не покрывают ситуации, связанные с особенностями микроархитектуры. Для проверки разных вариантов совместного исполнения команд на конвейере был разработан генератор RISC-V Torture Test Generator (далее – Torture) [13]. Он строит тестовые программы случайным образом, комбинируя описанные вручную последовательности команд небольшой длины и выбирая используемые ими регистры. Существуют универсальные генераторы тестовых программ, применимые к разным архитектурам. Наиболее известным из них является Genesys-Pro [7] от IBM Research. Генератор использует следующие входные данные: *модель микропроцессора* и *шаблоны*, описывающие последовательности команд и задающие для них распределения вероятностей и ограничения. При генерации инструмент предсказывает состояние регистров и памяти путем исполнения построенного кода на внешнем симуляторе. Что используется для создания *самопроверяющих тестовых программ*.

Другой генератор, RAVEN (Random Architecture Verification Machine) [14], разработан в компании Obsidian Software, поглощенной ARM. Инструмент основан на том же принципе, что и Genesys-Pro – логика генерации отделена от модели, описывающей архитектуру. RAVEN генерирует программы на основе шаблонов и может создавать случайные и нацеленные тесты. Как и Genesys-Pro, RAVEN использует внешний симулятор для определения корректного состояния микропроцессора.

Рассмотренные генераторы тестовых программ имеют следующие недостатки. Инструмент Torture ориентирован на случайную генерацию и не умеет строить тесты по пользовательским сценариям; кроме того, для расширения набора поддерживаемых команд нужно модифицировать код генератора. Инструменты RAVEN и Genesys-Pro лишены указанных недостатков, но не поддерживают RISC-V. Следует отметить, что создание и поддержка модели архитектуры является непростой задачей; помимо этого, при расширении системы команд требуется менять внешний симулятор.

Подход, реализованный в инструменте MicroTESK, нацелен на упрощение разработки и сопровождения генераторов тестовых программ за счет извлечения всей необходимой информации из единого источника – формальных спецификаций системы команд.

III. ΓΕΗΕΡΑΤΟΡ ΤΕСΤΟΒЫΧ ΠΡΟΓΡΑΜΜ MICROTESK FOR RISC-V

А. Инструмент MicroTESK

Генератор тестов MicroTESK for RISC-V создан с помощью инструмента MicroTESK [9-10]. Инструмент включает две основные части: *среду моделирования* и *среду генерации*. Первая отвечает за построение модели архитектуры микропроцессора на основе формальных спецификаций; вторая – за построение тестовых программ на основе модели архитектуры и тестовых шаблонов. Архитектура MicroTESK показана на рис. 1.

Среда моделирования использует в качестве входных данных формальные спецификации системы команд микропроцессора на языке nML [11]. Она реализует анализаторы и конструкторы, применяемые для построения модели микропроцессора. Построенная модель состоит из следующих частей:

1) *метаданные* — описывают поддерживаемые регистры и команды;

2) *симулятор* — исполняет команды и предоставляет информацию о состоянии регистров и памяти;

 модель тестового покрытия — описывает возможные пути исполнения команд.



Рис. 1. Архитектура инструмента MicroTESK

Среда генерации получает на вход *тестовые шаблоны* на языке, основанном на Ruby [15]. Язык позволяет описывать сценарии тестирования на абстрактном уровне, где состав команд, их порядок и операнды не фиксируются, а выбираются динамически в зависимости от параметров генерации. В шаблонах можно обращаться к регистрам и вызывать команды, используя синтаксис близкий к языку ассемблера. Все конструкции языка, необходимые для работы с той или иной архитектурой, создаются динамически на основе метаданных.

Тестовые шаблоны разрабатываются инженерамиверификаторами, исходя из своих целей. Создание некоторых типов тестов автоматизируется средствами MicroTESK:

1) тесты на отдельные команды, использующие случайные значения операндов;

2) тесты на отдельные команды, использующие граничные значения операндов;

3) тесты на отдельные команды, покрывающие все пути исполнения, описанные в спецификациях;

4) тесты на короткие цепочки команд, полученные перебором команд разного типа.

Процесс генерации тестовых программ по шаблону состоит из следующих стадий:

1) анализ тестового шаблона и построение его внутреннего представления;

2) построение абстрактных последовательностей команд (без указания конкретных тестовых данных); для каждой из них применяются следующие действия:

- а) конкретизация команд:
 - выбор регистров;
 - о генерация данных;
 - о построение инициализирующего кода;

- b) исполнение команд на внутреннем симуляторе MicroTESK;
- с) при генерации самопроверяющих программ: вставка в последовательность проверок на основе данных симулятора;
- 3) печать тестовой программы на языке ассемблера.

В. Спецификации системы команд

Архитектура RISC-V описана в руководстве «*The RISC-V Instruction Set Manual*» [16-17]. Система команд состоит из базового набора команд и расширений. Предусмотрены 32-, 64- и 128-битные реализации RISC-V. Для 32- и 64-битных версий определены следующие подмножества системы команд:

1) **RV32I** (базовый набор) — содержит команды целочисленной арифметики, ветвления, доступа к памяти и системных вызовов, общие для 32- и 64битной версий;

2) **RV64I** (дополнение к **RV32I**) — содержит дополнительные команды целочисленной арифметики и доступа к памяти для 64-битной версии;

3) **RV32M** (стандартное расширение) — содержит команды целочисленного умножения и деления, общие для 32- и 64-битной версий;

4) **RV64M** (дополнение к **RV32M**) — содержит дополнительные команды целочисленного умножения и деления для 64-битной версии;

5) **RV32A** (стандартное расширение) — содержит команды атомарных операций над данными из памяти для 32- и 64-битной версий;

6) **RV64A** (дополнение к **RV32A**) — содержит дополнительные команды атомарных операций над данными из памяти для 64-битной версии;

7) **RV32F** (стандартное расширение) — содержит команды арифметики с плавающей точкой одинарной точности и команды конвертации в целочисленный формат, общие для 32- и 64-битной версий;

8) **RV64F** (дополнение к **RV32F**) — содержит дополнительные команды конвертации для 64-битной версии;

9) **RV32D** (стандартное расширение) — содержит команды арифметики с плавающей точкой двойной точности и команды конвертации в целочисленный формат, общие для 32- и 64-битной версий;

10) **RV64D** (дополнение к **RV32D**) — содержит дополнительные команды конвертации для 64-битной версии;

11) **RV32C** (стандартное расширение) — содержит сжатые (16-битные) команды, общие для 32- и 64-битной версий;

12) **RV64C** (дополнение к **RV32C**) — содержит дополнительные сжатые (16-битные) команды для 64-битной версии;

13) привилегированные системные команды.

Состояние микропроцессоров RISC-V описывается следующими регистрами:

1) счетчик команд РС размером 32 или 64 бита;

32 регистра общего назначения x0-x31 размером
 или 64 бита (RV32I и RV64I);

3) 32 регистра для хранения чисел с плавающей точкой одинарной точности f0-f31 (RV32F и RV64F);

4) 32 регистра для хранения чисел с плавающей точкой двойной точности f0-f31 (RV32D и RV64D);

5) управляющий регистр для операций с плавающей точкой FCSR (Floating-Point Control and Status Register) размером 32 бита (RV32F, RV64F, RV32D и RV64D);

6) управляющие системные регистры CSR (Control and Status Registers) размером 32 или 64 бита (RV32I и RV64I) в количестве до 4096.

Микропроцессоры RISC-V поддерживают три уровня привилегий, переключение между которыми осуществляется системными командами:

1) М (Machine) — высший уровень привилегий;

2) S (Supervisor) — уровень операционной системы;

3) U (User) – уровень пользовательских приложений.

На языке nML были описаны все перечисленные подмножества системы команд. Версия архитектуры (32 или 64 бита) задается при помощи директивы.

В приведенном ниже примере описываются регистры общего назначения, режим адресации для доступа к ним и команды целочисленного сложения.

// Разрядность слова (зависит от конфигурации) let XLEN = #ifdef RV64I 64 #else 32 #endif type XWORD = card(XLEN)

// Регистры общего назначения reg XREG [32, XWORD]

// Режим адресации для регистров общего назначения **mode** X (i: **card**(5)) = XREG [i]

syntax = **format**("x%d", i) // Ассемблерный формат image = **format**("%5s", i) // Бинарный формат

// Команда сложения регистра и 12-битного значения **ор** addi(rd: X, rs1: X, imm: **card**(12))

syntax = format("addi %s, %s, 0x%x",

rd.syntax, rs1.syntax, imm)

action = { // Осуществляемое действие rd = rs1 + sign_extend(XWORD, imm); }

В описании команд может быть несколько путей исполнения. В качестве примера ниже приведена спецификация команды целочисленного деления DIV.

Для всех путей исполнения MicroTESK строит описывающие их ограничения (формулы). С помощью оператора *mark* этим ограничениям можно назначить имена (см. пример ниже), позволяющие ссылаться на ограничения из тестовых шаблонов.

```
op div(rd: X, rs1: X, rs2: X)
 syntax = format("div %s, %s, %s",
                  rd.syntax, rs1.syntax, rs2.syntax)
 image = format("0000001\%s\%s100\%s0110011".
                  rs2.image, rs1.image, rd.image)
 action = {
  if rs2 == 0 then
   mark("div_by_zero");
   rd = -1:
  elif rs1 == 1 \ll (XLEN - 1) \&\& rs2 == -1 then
   mark("min_xint_negation");
   rd = rs1;
  else
   mark("normal");
   rd = cast(XINT, rs1) / cast(XINT, rs2);
  endif;
```

С. Генерация тестовых программ

После того как разработаны спецификации и по ним автоматически построена модель архитектуры, можно использовать среду генерации MicroTESK для построения тестовых программ. Для этого на вход инструменту подаются тестовые шаблоны на языке Ruby, описывающие сценарии тестирования с помощью специальных конструкций.

При построении тестовых программ задействуются разные техники: рандомизация, перебор, разрешение ограничений. Например, приведенный ниже тестовый шаблон порождает 10 тестовых воздействий, каждое из которых состоит из команды ADD со случайными номерами регистров.

Класс на Ruby, определяющий тестовый шаблон **class** RandomTemplate < RISCVBaseTemplate

Главный метод тестового шаблона **def** run

Pacnpedeлeниe вероятностей для входных данных int dist = **dist**(

range(:*value* => 0, :*bias* => 25), **range**(:*value* => 1..2, :*bias* => 25), **range**(:*value* => 0xffffFFFE..0xffffFFFF, :*bias* => 50))

Описание структуры тестового воздействия sequence {

add x(_), x(_), x(_) do # Задание способа генерации входных данных testdata('random_biased', :*dist* => int_dist) end }.run 10 # Число тестовых воздействий

```
end
```

```
end
```

Ниже приведен фрагмент шаблона, описывающий последовательности команд ADD и SUB, которые принимают в качестве входных данных декартово произведение значений из диапазона [1, 3]. Всего получается 81 последовательность (3 ^ 4, где 3 – число значений, а 4 – число входных регистров).

```
sequence(:data_combinator => 'product') {
   add x8, x9, x10 do
    testdata('range', :min => 1, :max => 3)
   end
   sub x11, x12, x13 do
   testdata('range', :min => 1, :max => 3)
   end
```

}.run

Следующий пример демонстрирует совместное применение комбинаторной генерации и генерации на основе ограничений. Входные данные для команды DIV строятся таким образом, чтобы покрыть все пути исполнения, заданные в спецификации. При этом каждый вариант дополняется как командой SLT, так и командой MUL, что дает 6 последовательностей.

Комбинирует результаты вложенных блоков block(:combinator => 'product', :compositor => 'random'){

iterate { # Итерирует по вложенным командам div x8, x9, x10 do situation('div_by_zero') end div x8, x9, x10 do situation('min_xint_negation') end div x8, x9, x10 do situation('normal') end }

iterate { *# Итерирует по вложенным командам* slt x(_), zero, x8 mul x(_), x8, x10

```
}
\ =====
```

}.run

Для каждого тестового воздействия строится инициализирующий код, помещающий тестовые данные в регистры и память. Для этого используются так называемые *препараторы*. Можно определить несколько препараторов, которые будут применяться в зависимости от формата тестовых данных. Ниже приведены препараторы, записывающие данные в регистры общего назначения.

Препаратор по умолчанию для регистров X preparator(:target => 'X') {

Записывает значение value в perucmp target li target, value

```
}
```

Препаратор для случая, когда value равно 0 preparator(:target => 'X', :mask => '0000_0000') { or target, zero, zero

}

IV. ХАРАКТЕРИСТИКИ РАЗРАБОТАННОГО ГЕНЕРАТОРА ТЕСТОВЫХ ПРОГРАММ

Инструмент MicroTESK for RISC-V позволяет генерировать следующие виды тестов:

- 1) случайные тесты:
 - а) рандомизация последовательностей команд;
 - b) рандомизация тестовых данных:
 - і. задание распределений вероятностей;
 - ii. случайный выбор значений из диапазонов;

2) комбинаторные тесты:

4)

- а) комбинирование последовательностей команд;
- b) перебор тестовых данных:
 - і. перебор граничных значений;
 - іі. перебор значений из диапазонов;

3) комбинаторные тесты на команды ветвления:

- а) перебор графов потока управления;
- b) перебор трасс исполнения;
- тесты на покрытие путей исполнения команд:
- а) перебор путей исполнения;
- b) разрешение ограничений;
- 5) тесты со встроенными проверками;

6) комбинированные тесты, сочетающие свойства перечисленных выше типов тестов.

Пакет MicroTESK for RISC-V включает тестовые шаблоны, демонстрирующие возможности генератора. Также он содержит шаблоны, описывающие тесты из набора Университета Калифорнии в Беркли [12].

Таблица 1

| Специфицированные | команды | RISC-V |
|-------------------|---------|--------|
|-------------------|---------|--------|

| Подмножество | Описание | Число |
|---------------|--|-------|
| RV32I | целочисленная арифметика, доступ к памяти, ветвления | 37 |
| RV32I | доступ к управляющим регистрам | 6 |
| RV32I | системные команды | 9 |
| RV64I | целочисленная арифметика, доступ к памяти | 12 |
| RV32M | целочисленное умножение и деление | 8 |
| RV64M | целочисленное умножение и деление | 5 |
| RV32A | атомарные операции | 11 |
| RV64A | атомарные операции | 11 |
| RV32F | плавающая арифметика | 26 |
| RV64F | плавающая арифметика | 4 |
| RV32D | плавающая арифметика | 26 |
| RV64D | плавающая арифметика | 6 |
| RV32C | сжатые команды | 35 |
| RV64C | сжатые команды | 7 |
| Псевдокоманды | псевдокоманды | 59 |
| Всего | | 262 |

В работе по созданию формальных спецификаций системы команд RISC-V на языке nML участвовало 2 человека; общие трудозатраты составили около 4 человеко-месяцев. Всего описано 262 команды (включая 59 псевдокоманд, используемых ассемблером). Объем спецификаций составил около 3500 строк кода. В таблице 1 приведены данные по специфицированным командам.

При разработке спецификаций возможны ошибки, которые выражаются в некорректном ассемблерном формате команд или некорректных результатах их исполнения на встроенном симуляторе. В результате тестовые программы либо не компилируются, либо приводят к ошибочным результатам. Чтобы избежать подобных проблем для всех тестовых шаблонов, входящих в MicroTESK for RISC-V, при сборке дистрибутива осуществляются следующие действия.

1) По шаблонам генерируются тестовые программы. При генерации программа исполняется на встроенном симуляторе, создающем *трассы исполнения* (трассы фиксируют такие события, как исполнение команды, запись в регистр и обращение к памяти).

2) Построенные инструментом тестовые программы компилируются и исполняются на таких симуляторах, как QEMU for RISC-V [18] и Spike [19], создающих трассы того же формата.

3) Трассы исполнения одной и той же программы сравниваются с помощью средства Trace Matcher [20], которое генерирует отчет о найденных различиях.

V. ЗАКЛЮЧЕНИЕ

В работе описан генератор тестовых программ для микропроцессоров с архитектурой RISC-V, созданный в ИСП РАН с помощью инструмента MicroTESK. Инструмент позволяет генерировать тесты разных типов, в частности аналогичные тем, что входят в тестовый набор Университета Калифорнии в Беркли, и тем, что создаются инструментом Torture. Важными преимуществами MicroTESK for RISC-V являются открытость и расширяемость, что позволяет пополнять систему команд без модификации ядра генератора.

В ближайшем будущем мы планируем описать подсистему памяти микропроцессоров RISC-V, а также команды из появляющихся стандартных расширений (векторные команды и другие). Также мы планируем реализовать следующие вспомогательные средства инструмента MicroTESK:

1) генераторы отчетов о тестовом покрытии уровня команд, достигаемом сгенерированными программами;

2) инструмент автоматизированного создания onlineгенераторов тестов – генераторов, работающих на ПЛИС-прототипах или опытных образцах СБИС;

3) инструмент статического анализа и дедуктивной верификации бинарного кода.

ЛИТЕРАТУРА

- [1] URL: https://en.wikipedia.org/wiki/RISC-V (дата обращения: 28.03.2018)
- [2] URL: https://riscv.org (дата обращения: 28.03.2018) [3] URL:
- https://www.phoronix.com/scan.php?page=news_item&px= NVIDIA-RISC-V-Next-Gen-Falcon (дата обращения: 28.03.2018)
- [4] URL: https://www.electronicsweekly.com/blogs/mannerisms/dile mmas/samsung-defection-arm-risc-v-2016-11 (дата обращения: 28.03.2018)
- [5] URL: https://www.pulp-platform.org (дата обращения: 28.03.2018)
- [6] Камкин А.С. Генерация тестовых программ для микропроцессоров. Труды ИСП РАН, Т. 14, Ч. 2, 2008. С. 23-64.
- [7] Adir A., Almog E., Fournier L., Marcus E., Rimon M., Vinov M., and Ziv A. Genesys-Pro: Innovations in Test Program Generation for Functional Processor Verification // IEEE Design & Test of Computers, Volume 21, Issue 2, 2004, P. 84–93.
- [8] URL: https://forge.ispras.ru/projects/microtesk-riscv (дата обращения: 28.03.2018)
- [9] URL: https://forge.ispras.ru/projects/microtesk (дата обращения: 28.03.2018)
- [10] Chupilko M., Kamkin A., Kotsynyak A., Tatarnikov A. MicroTESK: Specification-Based Tool for Constructing

Test Program Generators. Hardware and Software: Verification and Testing. Proceedings of 13th International Haifa Verification Conference, 2017. P. 217-220.

- [11] Freericks M. The nML Machine Description Formalism. Techical Report, TU Berlin, FB20, Bericht 1991/15.
- [12] URL: https://github.com/riscv/riscv-tests (дата обращения: 28.03.2018)
- [13] URL: https://github.com/ucb-bar/riscv-torture (дата обращения: 28.03.2018)
- [14] URL: http://www.slideshare.net/DVClub/introducingobsidian-software-and-ravengcs-for-powerpc (дата обращения: 28.03.2018)
- [15] Tatarnikov A.D. Language for Describing Templates for Test Program Generation for Microprocessors. Trudy ISP RAN (Proceedings of ISP RAS), Volume 28, Part 4, 2016. P. 81-102.
- [16] Waterman A., Asanovic K. The RISC-V Instruction Set Manual. Volume I: User-Level ISA. Version 2.2. University of California, Berkeley. May 7, 2017. 145 P.
- [17] Waterman A., Asanovic K. The RISC-V Instruction Set Manual. Volume II: Privileged Architecture. Version 1.10. University of California, Berkeley. May 7, 2017. 91 P.
- [18] URL: https://forge.ispras.ru/projects/qemu-riscv (дата обращения: 28.03.2018)
- [19] URL: https://github.com/riscv/riscv-isa-sim (дата обращения: 28.03.2018)
- [20] URL: https://forge.ispras.ru/projects/traceutils (дата обращения: 28.03.2018)

MicroTESK-Based Test Program Generator for the RISC-V Architecture

A.S. Kamkin^{1, 2, 3, 4}, A.S. Protsenko¹, S.A. Smolov¹, A.D. Tatarnikov^{1, 4}

¹Ivannikov Institute for System Programming of the Russian Academy of Sciences, Moscow

²Lomonosov Moscow State University, Moscow

³Moscow Institute of Physics and Technology, Moscow

⁴National Research University Higher School of Economics, Moscow

{kamkin, protsenko, smolov, andrewt}@ispras.ru

Abstract — In this paper, a specification-based test program functional verification generator for of RISC-V microprocessors is presented. The tool is based on the MicroTESK framework and consists of two main parts: (1) the formal specifications of the RISC-V ISA and (2) the ISA-independent generation core. Test programs are generated on the basis of the ISA specifications and test templates that are high-level descriptions of test scenarios. The RISC-V ISA specifications are written in nML language. They describe the syntax and semantics of the instructions. The information extracted from the specifications is used in multiple ways: to get instruction signatures to be used in test templates; to build the test coverage model that holds constraints describing execution paths of the instructions; to construct the instruction set simulator that serves as a reference model. Test templates are created using a Rubybased domain-specific language and describe how instruction sequences are composed and what constraints and generation methods are applied. The generation core provides random, combinatorial, and constraint-based test generation methods. The built-in instruction set simulator allows executing instructions in the process of test program generation. This allows predicting the microprocessor state to ensure the validity of the tests, to create self-checks, and to solve constraints that use state information. MicroTESK for RISC-V supports the following instruction subsets: RV32I, RV64I, RV32M, RV64M, RV32A, RV64A, RV32F, RV64F, RV32D, RV64D, RV32C, and RV64C. In total, the specifications cover 262 instructions. The effort required to develop the specifications constituted about 4 man-months. The specifications can be easily modified to support more instructions. MicroTESK for RISC-V includes a set of test templates that provide basic ISA coverage and demonstrate the generator facilities.

Keywords — microprocessors; formal specifications; instruction set architecture; functional verification; test program generation; RISC-V; nML; MicroTESK.

REFERENCES

- [1] URL: https://en.wikipedia.org/wiki/RISC-V (access date: 28.03.2018)
- [2] URL: https://riscv.org (access date: 28.03.2018)
- [3] URL:
- https://www.phoronix.com/scan.php?page=news_item&px= NVIDIA-RISC-V-Next-Gen-Falcon (access date: 28.03.2018)

 [4] URL: https://www.electronicsweekly.com/blogs/mannerisms/dile mmas/samsung-defection-arm-risc-v-2016-11 (access date: 28.03.2018)

- [5] URL: https://www.pulp-platform.org (access date: 28.03.2018)
- [6] Kamkin. A. Generatsiya testovykh programm dlya mikroprotsessorov (Test Program Generation for Microprocessors). Trudy ISP RAN (Proceedings of ISP RAS), Volume 14, Part 2, 2008, P. 23-64.
- [7] Adir A., Almog E., Fournier L., Marcus E., Rimon M., Vinov M., and Ziv A. Genesys-Pro: Innovations in Test Program Generation for Functional Processor Verification // IEEE Design & Test of Computers, Volume 21, Issue 2, 2004, P. 84–93.
- [8] URL: https://forge.ispras.ru/projects/microtesk-riscv (access date: 28.03.2018)

- [9] URL: https://forge.ispras.ru/projects/microtesk (access date: 28.03.2018)
- [10] Chupilko M., Kamkin A., Kotsynyak A., Tatarnikov A. MicroTESK: Specification-Based Tool for Constructing Test Program Generators. Hardware and Software: Verification and Testing. Proceedings of 13th International Haifa Verification Conference, 2017. P. 217-220.
- [11] Freericks M. The nML Machine Description Formalism. Techical Report, TU Berlin, FB20, Bericht 1991/15.
- [12] URL: https://github.com/riscv/riscv-tests (access date: 28.03.2018)
- [13] URL: https://github.com/ucb-bar/riscv-torture (access date: 28.03.2018)
- [14] URL: http://www.slideshare.net/DVClub/introducingobsidian-software-and-ravengcs-for-powerpc (access date: 28.03.2018)
- [15] Tatarnikov A.D. Language for Describing Templates for Test Program Generation for Microprocessors. Proceedings of ISP RAS, Volume 28, Part 4, 2016. P. 81-102.
- [16] Waterman A., Asanovic K. The RISC-V Instruction Set Manual. Volume I: User-Level ISA. Version 2.2. University of California, Berkeley. May 7, 2017. 145 P.
- [17] Waterman A., Asanovic K. The RISC-V Instruction Set Manual. Volume II: Privileged Architecture. Version 1.10. University of California, Berkeley. May 7, 2017. 91 P.
- [18] URL: https://forge.ispras.ru/projects/qemu-riscv (access date: 28.03.2018)
- [19] URL: https://github.com/riscv/riscv-isa-sim (access date: 28.03.2018)
- [20] URL: https://forge.ispras.ru/projects/traceutils (access date: 28.03.2018)

Система комбинируемых специализированных генераторов тестов для нового поколения VLIW DSP процессоров с архитектурой Elcore50

А.В. Гаращенко^{1,2}, А.В. Николаев¹, Ф.М. Путря¹, С.С. Сардарян¹

¹ОАО НПЦ «ЭЛВИС», г. Москва, г. Зеленоград,

²Национальный исследовательский университет «МИЭТ», г. Москва, г. Зеленоград,

ant.gar1@mail.ru

Аннотация — Даже если не вдаваться в подробности реализации микроархитектуры, пространство состояний современного ядра астрономически огромно. Оно определяется сочетанием набора команд во VLIW инструкции, динамическим сочетанием команд и зависимостей между ними в конвейере, динамикой исполнения обращений к памяти (которых во VLIW процессоре может исполняться несколько в рамках одной команды), внешними прерываниями, состоянием подсистемы отладки. В данной работе рассмотрено решение основе системы сочетаемых на позволяющих специализированных генераторов, организовывать иерархические вызовы разных генераторов в процессе создания теста, и, таким образом, добиться расширения покрываемых подмножеств глобального пространства состояний процессора. не снижая при этом вероятность формирования краевых ситуаций для отдельных подсистем и подмножеств свойств процессора, на которые нацелены каждый из специализированных генераторов в отдельности.

Ключевые слова — верификация процессоров, широкое командное слово, подсистема памяти, генерация тестов, покрытие.

I. Введение

В связи с архитектурной сложностью современных применяемых многоядерных процессоров, при разработке систем на кристалле (СнК), более шестидесяти процентов ресурсов проектирования тратится на их верификацию. Это обусловлено высокой комбинаторной сложностью проверки корректности работы как отдельных ядер, так и системы в целом. Помимо этого, существует тенденция усложнения СнК за счет повышения гетерогенности, связанной с необходимостью увеличения скорости выполнения отдельных классов задач. Такие вычислительные системы состоят из ядер общего назначения и специализированных вычислительных ядер. Т.е. стоит задача проверки реализаций разных архитектур вычислительных ядер, входящих в состав проектируемой СнК, которые могут отличаться не только системой команд, но и способом компоновки команд во VLIW-пакет (VLIW - Very Long Instruction

Word), его шириной, организацией регистровых файлов, кэшей и многим другим. Вычислительная сложность современных алгоритмов формальной верификации ограничивает область их применения небольшими блоками (типа ALU или отдельных элементов подсистемы памяти) или отдельными свойствами процессора, которые легко локализовать, что резко ограничивает использование формальных методов для тестов уровня полного ядра процессора [1-4]. Таким образом, динамическая верификация пока ещё остается одним из основных методов проверки вычислительных ядер.

Если касаться особенностей верификации именно специализированных микропроцессоров обработки цифровых сигналов (далее DSP), то среди них стоит выделить широкое комбинаторное пространство возможных ситуаций. Зачастую такие процессоры имеют гарвардскую VLIW архитектуру со скалярным и векторным исполнительными каналами, аппаратными циклами и многоканальной памятью. Их верификация требует огромного объема сложных тестов, что становится основной проблемой функциональной верификации. Однако ограничивающим фактором является время, требуемое для разработки полного набора тестов. Ввиду чего не может идти речи о написании всех тестов вручную. Это определяет высокую актуальность создания новых средств для проверки правильности работы подобных структур.

Для увеличения скорости моделирования, а, значит, и проверки правильности работы, каждое процессорное ядро и даже некоторые его модули верифицируются в автономных тестовых окружениях (при условии, что проект процессора написан с учетом требований со стороны задачи декомпозиции верификации [5, 6]), так как скорость моделирования отдельных частей процессора на порядок больше. Однако задача создания всеобъемлющего набора тестовых последовательностей для процессорного ядра остается критической. Далеко не все подсистемы можно локализовать в виде отдельного блока, и даже там, где это возможно, могут иметь место ошибки в протоколе межблочного взаимодействия. Даже если не вдаваться в подробности реализации микроархитектуры, пространство современного состояний ядра астрономически огромно. Оно определяется сочетанием набора команд во VLIW-инструкции, динамическим сочетанием команд и зависимостей между ними в конвейере, динамикой исполнения обращений к памяти, которых во VLIW-процессоре может исполняться несколько в рамках одной команды, внешними прерываниями, состоянием подсистемы отладки. Генераторы тестов уже давно служат в качестве основного инструмента для покрытия вычислительных ядер тестами [7-9]. Однако объем пространства состояний такой, что один генератор общего назначения будет чрезмерно сложным и с редким выходом краевых ситуаций, что приведет к неприемлемо долгому процессу генерации и запуску тестов до момента достижения требуемого покрытия. Специализированные генераторы тестов на отдельные подсистемы или подмножества свойств процессора, например, отдельные генераторы тестов на набор команд программного управления, на подсистему памяти, на подсистему прерываний позволяют создать большое число краевых ситуаций для целевой подсистемы и, соответственно, добиться их большего покрытия. Однако, с точки зрения системы, специализированные генераторы покрывают лишь отдельные локализованные подмножества глобального пространства состояний, оставляя большие пробелы между подмножествами в этом пространстве. промежуточного состояния, Примером не покрываемого специализированными генераторами, может быть прерывание в момент исполнения заданной программного комбинации команд управления параллельно с исполнением команд обращения в кэш память, конфликтующих друг с другом при обращении к кэш строке. Полностью случайный поток команд такое состояние создаст через годы моделирования, а направленные специализированные тесты не создадут вовсе, поскольку используют ограничивающие их поведение шаблоны или модели.

В данной работе предложено решение на основе системы сочетаемых специализированных генераторов, позволяющих организовывать иерархические вызовы разных генераторов в процессе создания теста, и, таким образом. добиться расширения покрываемых подмножеств глобального пространства состояний процессора. Например, сочетание генератора на программное управление (генерация исключений) с генератором на подсистему памяти (исключения в условиях длительной блокировки конвейера). Такие тесты выполняют проверку системы в целом, так как одновременная работа многих частей процессорного ядра создаёт критические ситуации, вероятность появления которых при их направленной верификации очень мала. Для их генерации необходимо учесть существующие методики и подходы тестирования различных блоков процессора. Сгенерированные тестовые последовательности позволяют осуществлять взаимодействия проверку различных устройств процессорного ядра в рамках одного теста.

II. МОДЕЛЬ ГЕНЕРАТОРА ПОТОКА СЛУЧАЙНЫХ ИНСТРУКЦИЙ

Архитектура модели, позволяющей решить задачу генерации случайных последовательностей, приведена на рис. 1. Тесты для системы команд DSP представляют собой последовательности ассемблерных инструкций. тестирование позволяет проверить Такое эти инструкции на предмет правильности выполнения с поведенческой точки зрения путем тестирования случайными входными данными. Для кажлой инструкции случайными данными являются: исходное состояние используемых регистров (как входных, так и выходных), номера этих регистров, состояние и адреса ячеек памяти в форматах, требующих пересылок. Сгенерированная программа предназначается для выполнения на RTL-модели (register transfer level) и симуляторе. Для применения такого генератора при верификации другого микропроцессора необходима минимальная модификация программы, а именно – изменить конфигурационный файл, содержащий мнемоники ассемблерных команд и их форматы.

Рис. 1. Архитектура генератора случайных инструкций



Генерация тестовых данных включает в себя произвольный выбор тестируемой инструкции из списка, получаемого из конфигурационного файла; выбор формата из возможных для данной инструкции; выбор регистров для инструкции в зависимости от формата пересылок; инициализацию регистра состояния. Для заданной инструкции и параметров вычисляется эталонное значение. При инициализации значений 64-x И 128-и разрядных регистров используются промежуточные пересылки через два 32х разрядных регистра, поэтому они зарезервированы и используются в качестве источников не ипи приемников вычислительных команд и пересылок.

Возможны два сценария использования данной модели генератора.

По первому сценарию тест состоит из нескольких подтестов, записываемых в PRAM-память процессора. Каждый подтест заключается в инициализации начальных данных (регистров и памяти), выполнении одной тестовой команды, проверке результата и записи результата выполнения в память. По умолчанию программа генерируют тесты по первому сценарию работы.

По второму сценарию тест заключается в регистрового инициализации файла. адресных регистров и соответствующих им ячеек памяти начальными произвольными значениями; выполнении заданного количества произвольных инструкций. Количество генерируемых инструкций ограничено 4000. Поддерживается возможность задания доли базовых инструкций и, соответственно, расширенных. Результатом теста является состояние регистрового файла. Корректность выполнения теста проверяется сравнением архитектурного состояния RTL эталонной модели процессора.

Особенностью данного генератора является способность создавать VLIW-пакеты, содержащие векторные и скалярные инструкции. Поддерживается генерация пакетов как заданной длины, так и случайной (ограниченной максимальной длиной). Инструкции попадают в один пакет с учетом ограничения количества команд данного типа, которые можно исполнить в рамках одного VLIW-пакета.

III. МОДЕЛЬ ГЕНЕРАТОРА ТЕСТОВ НА ПРОГРАММНОЕ УПРАВЛЕНИЕ

При верификации устройства управления DSPпроцессора следует учитывать особенности его архитектуры. Основным объектом тестирования является конвейер микропроцессора. В основном проверяются ситуации, приводящие его к различным блокировкам. В рамках рассматриваемой модели генератора существует необходимость покрытия пространства блокировок, вызванных программными переходами, подпрограммами, аппаратными циклами и исключениями; а также блокировок, вызванных зависимостью по данным между следующими друг за другом инструкциями.



Рис. 2. Структурная схема генератора на программное управление

Структура, позволяющая решить данную задачу, приведена на рис. 2. Модуль управления генераторами задаёт общие правила генерации теста, описанные в конфигурационном файле. Они поступают на вход генераторов инструкций и данных, которые осуществляют формирование последовательностей, состоящих из исполняемого кода, связей между ними; а также создание зависимых и независимых операндов для инструкций соответственно. Для расчёта значений регистров и памяти полученный код запускается на функциональной модели, помимо этого обеспечивая возможность получения текущего состояния тестируемой системы в целом. Диспетчер запросов отвечает за обработку запросов функциональной модели к памяти за инструкциями и данными.

С помощью такого генератора возможно генерировать тесты вида, который приведен на рис. 3, где доступны программные условные и безусловные переходы по относительным (j-команды) и абсолютным (b-команды) адресам со слотами задержки, а также с сохранением адреса возврата; вызов подпрограмм, в том числе, и из подпрограммы (кроме рекурсии); вложенные аппаратные циклы [10].



Рис. 3. Структурная схема теста

Основными составляющими секции с переходами являются последовательности двух типов: seq_n, seq_subroutine_n. Seq – часть кода, состоящая из случайных инструкций, связанных условными или безусловными переходами. Seq_subroutine – подпрограммы со своим стеком. Переходы генерируются как с положительным, так и с отрицательным смещением.

Вызов подпрограмм реализован с помощью передачи адреса возврата, а также аргументов через регистры в функцию с дальнейшей записью в память (общий стек). Указатель на верхушку стека размещается в памяти по известному адресу, при этом этот адрес генерируется случайным образом с учетом ограничений карты памяти микропроцессора. Когда происходит запись в память, указатель инкрементируется, при извлечении декрементируется. После выполнения тела функции адрес читается из стека в регистр с последующим переходом по нему. Возможен вызов подпрограмм из других подпрограмм, при этом стоят ограничения от зацикливаний, учитывая возможную косвенную или прямую рекурсию. Количество вызовов подпрограмм ограничено размером стека.

Реализована генерация программных и аппаратных циклов. Программные циклы сделаны с помощью условных и безусловных переходов. Глубина циклов, как и количество вызовов подпрограмм, задаётся в конфигурационном файле. Однако следует учесть, что глубина аппаратных циклов ограничена аппаратурой, поэтому при генерации циклов в подпрограммах учитывается их вложенность до этого.

Отличительной особенностью рассматриваемого генератора является возможность создания обработчиков исключений, адреса которых помещаются в специальный регистр. Помимо этого генерируются исключения трех типов:

1) UI (Unknown Instruction) – неизвестная инструкция;

- 2) BA (Bad Addres) неверный адрес;
- 3) SYSCALL системные вызовы.

Первый тип исключений создаётся с помощью различных прыжков на адреса с несуществующими инструкциями, а также прямой записью в регистр исключений.

Второй тип генерируется путем обращения в память PRAM по запрещенным DSP-процессором адресам или Scatter-Gather- обращением в кэш с одним адресом, не принадлежащим к диапазону PRAM памяти.

```
Пример:
```

| 1 1 | |
|-----------------------|-----------------------|
| seq_subroutine _1: | seq _0: |
| stl r31, (r0) | subl r22, r12, r15 |
| addl 4, r0, r0 | addl r22, r17, r19 |
| do 4 llsub_11_end | bs seq_subroutine _1 |
| xor r12, r1, r3 | bs seq_subroutine _1 |
| minl r7.l, r4.l, r6.l | stl r0, (r=0x1200000) |
| do 6, llsub_12_end | j test_stop |
| bs seq_subroutine _2 | |
| llsub_12_end: | |
| subl 4, r0, r0 | |
| llsub_11_end: | |
| ldl (r0), r31 | |
| b r31 | |
| | |

В примере выше приведена часть сгенерированного теста, состоящая из одной последовательности типа seq, в которой присутствует два вызова подпрограммы seq_subroutine_1. В подпрограмме seq_subroutine_0 сначала сохраняется адрес возврата, затем указатель стека смещается на 4 единицы вверх, после чего во вложенном цикле идет вызов подпрограммы seq subroutine 2. Команды разделяются сгенерированными случайными инструкциями. В конце seq_subroutine 0 считывается адрес возврата в регистр 31, указатель на верхушку стека смещается на 4 единицы вниз и происходит переход по адресу в регистре 31.

Несмотря на наличие функциональной модели у генератора. для проверки тестируемого ядра микропроцессора на предмет несоответствия поведения тестируемого процессора и эталонной модели в основном используется режим сравнения результатов RTL И симулятора (TLM работы молепи) Преимущество такого подхода состоит в том, что проверка правильности поведения компонента выполняется автоматически.

IV. МОДЕЛЬ ГЕНЕРАТОРА ТЕСТОВ ИЕРАРХИИ КЭШЕЙ

Конструирование тестов данным генератором осуществляется путем построения графовой модели иерархии кэш-памяти, вершинами которой являются состояния кэша (рис. 4), а ребрами – переходы между состояниями (инструкции, написанные на метаязыке).



Рис. 4. Графовая модель иерархии кэш памяти

Метаязык дает возможность работать с памятью микропроцессора на более абстрактном уровне, отвязывая тесты от конкретного ассемблера, что позволяет переносить тесты для верификации процессоров с отличной архитектурой. Он содержит функции чтения (read) и записи (write). Предусмотрена векторно-блочная работа с памятью. Для конвертации метаязыка в ассемблерные команды разработан специальный генератор, который поддерживает не более 16 векторно-блочных обращений к памяти.

Между состояниями (вершинами графа) кэша генератором предусмотрены следующие переходы:

 hit(int mode) – попадание в выбранный кэш по чтению или записи. Параметр mode определяет какой адрес будет использован, если он равен 0, используется случайный закэшированный адрес, в случае если mode равен 1, используется адрес последней исполненной команды.

2) miss() – промах по чтению или записи.

 parallel_hit_miss(int mode=0) - параллельные операции попадания и промахов по чтению и записи. Параметр mode работает как в случае с hit.

4) replace() - вытеснение строки, использованной в последней исполненной команде.

5) next_line() – чтение или запись в двух соседних строках от строки, использованной в последней исполненной команде, при этом в саму строку обращения нет.

6) next_way() – чтение или запись в случайный ассоциативный путь для строки, использованной в последней исполненной команде, при этом в саму строку обращения нет.

При генерации всех переходов адрес, тип обращения и размер транзакции задаются случайным

образом. Пример графа для строки одного кэша и двух



ассоциативных путей показан на рисунке 5.

Рис. 5. Графовая модель для одной кэш строки

По описанию, приведенному в конфигурационном файле (рис. 6), строится графовая модель. Каждый переход в графе - это отдельный тест. Сначала генерируется преамбула, состоящая из последовательностей команд метаязыка, приводящих систему кэш-памяти к нужному состоянию; далее идет блок фиксации дампа памяти; затем инструкция, отвечающая за переход; блок проверки дампа памяти с

| L0 { | L1 { | L2 { |
|--------------------|------------------|-------------------|
| <u>ways</u> $= 1;$ | <u>ways</u> = 8; | <u>ways</u> = 16; |
| lines=128; | lines=256; | lines=256; |
| lineSize=32; | lineSize=64; | lineSize=64; |
| group=0; | group=A[6]; | group=A[6]; |
| lineAdr=0; | lineAdr=A[11:7]; | lineAdr=A[11:7]; |
| tag=A[31:5]; | tag=A[31:12]; | tag=A[31:12]; |
| byteAtdr=A[4:0]; | byteAtdr=A[5:0]; | byteAtdr=A[5:0]; |
| } | } | } |

эталонной моделью.

Рис. 6. Описание иерархии кэш памяти

Возможные критические ситуации для кэша (без учета когерентности), создаваемые рассматриваемым генератором: цепочки запись-чтение, запись-записьчтение-чтение; обращения по адресам после закэшированных вытеснения по ним данных; вытеснение строчки с дальнейшим чтением из нее; запись или чтение данных, переходящих через границу одной строки в одном ассоциативном пути для вытеснения в нем сразу двух строк; запись или чтение переходящих через данных, границу одного ассоциативного пути, чтобы вытеснять сразу две строки в двух разных путях. Генерация команды инвалидации позволяет создать дополнительную нагрузку на систему кэш-памяти. В дальнейшем предполагается наложить все это на механизм когерентности.

Для генерации тестов для связки L1 и L2 кушей требуется дополнительно указать в конфигурационном файле следующие моменты: является ли L1 инклюзивным; как происходит вытеснение из L1 и вытеснение сразу из L1 и L2; как происходит промах в L0 с вытеснением в L1.

Отличительной особенностью генератора является поддержка генерации тестовых сценариев для многоканального режима обращения в кэш-память.

Вариант по умолчанию — преамбула, состоящая из последовательностей команд метаязыка на одном канале с одновременным переходом в разные состояния по всем допустимым каналам и проверка данных сравнением с эталонной моделью. Также в генерацию тестов добавлены случайные инструкции обращения к внешней и внутренней памяти.

V. Генерация внешних прерываний

Внешнее прерывание, приходящее в общем случае асинхронно к потоку исполнения команд, должно привести к выходу на обработчик, его исполнению, и корректному возврату к процессу исполнения основного потока управления при любом исходном состоянии конвейера. Внешнюю генерацию прерываний для RTL-модели организовать легко. Однако, есть две проблемы.

Первая проблема — нужно комбинаторно перебрать состояния ядра в момент прихода прерывания. Для этих целей в качестве программы жертвы используется программа, генерируемая с помощью всех описанных выше генераторов и их комбинаций.

Вторая проблема — проверка корректности входа в обработчик и возврата из него. Классическим способом проверки является сравнение трасс и состояний в контрольных точках RTL и TLM-моделей. Однако реализовать подачу прерываний в одинаковые моменты времени с точки зрения конвейера моделей разных vровней абстракции крайне сложно. Можно было бы добиться потактовой точности ТLМ-модели И использовать информацию о внутреннем состоянии конвейера для генерации события прерывания, однако, данный подход требует слишком больших трудозатрат. данном этапе было решено использовать Ha упрощенный механизм сравнения. Факт выхода в обработчик и корректность его работы проверяется отдельным монитором (при этом совпадение трасс не требуется). В свою очередь трассы сравниваются для всех инструкций, кроме тех, которые исполняются в обработчике прерываний. Сходимость основной программы обеспечивается процедурой восстановления контекста при выходе из обработчика.

VI. ИНТЕГРАЦИЯ ГЕНЕРАТОРОВ

Для создания более сложных, комплексных тестов необходима интеграция генераторов друг в друга, так как взаимодействие разных устройств микропроцессора, в том числе и DSP-процессора, порождает огромное множество возможных ситуаций.

С помощью генератора случайных инструкций создаются VLIW-пакеты, которые случайным образом помещаются в тело подпрограмм, циклов, обработчиков исключений, а также вставляются между программными переходами и командами обращения в память.

Комбинирование генератора на программное управление с генератором на подсистему памяти позволяет создавать критические ситуации для микропроцессора. Например, исключения в условиях длительной блокировки конвейера, такие как блокировки, связанные с обращениями к памяти, которые, в свою очередь, подразделяются на блокировки арбитража, вызванные конфликтами при обращении к блокам памяти, и блокировки, вызванные задержкой ответа со стороны памяти (это имеет место при промахах кэш, а также некэшируемых обращениях к внешней памяти).

Для обеспечения комбинируемости генераторов в генерируемом коде, выполняющем проверку определенной подсистемы, предусмотрены специальные секции, в которые допускается интеграция кода генераторов потока арифметических команд или тестов доступа к памяти ортогональных по используемым ресурсам к тестовой программе верхнего уровня.

VII. ЗАКЛЮЧЕНИЕ

В работе проведен анализ проблем, возникающих при автоматизации генерации псевдослучайных тестов для вычислительных ядер. Описаны пути их решения с формированием рекомендаций по оптимизации соответствующих архитектурных реализаций на примере разработки специализированных генераторов, таких как: генератор потока случайных инструкций, тестов на программное управление, тестов иерархии кэшей и тестов внешних прерываний. Созданные были применены генераторы при проверке корректности работы VLIW DSP-процессора архитектурой Elcore50. Также предложен подход к их объединению, который позволил протестировать взаимодействие разных блоков процессора и выявить несколько критичных ошибок в работе DSP-ядра.

По истечении года при помощи описанных моделей была создана система генерации тестов для разрабатываемого процессорного ядра с VLIWархитектурой, решающих комплексную задачу проверки всех подсистем ядра и их взаимодействия путем комбинирования уже существовавших и вновь разрабатываемых специализированных генераторов на отдельные подсистемы ядра.

ЛИТЕРАТУРА

- Vigyan Singhal, Starting Formal Right from Formal Test Plannin, Oski Technology, Verification Academy at DAC-2015. S. 1–10.
- [2] David M. Russinoff. Formal Verification of Floating-Point RTL at AMD Using the ACL2 Theorem Prover. In Computer-Aided Reasoning: ACL2 Case Studies, chapter 13. Kluwer Academic Publishers. 2000. S. 1–8.
- [3] Камкин А.С., Петроченков М.В. Система поддержки верификации реализаций протоколов когерентности с использованием формальных методов // Вопросы радиоэлектроники. 2014. Т. 5. № 2. С. 5–17.
- [4] Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем СПб.: БХВ-Петербург, 2010. 560 с.
- [5] Bening, Lionel, Foster, Harry D. Principles of Verifiable RTL Design: A functional coding style supporting verification processes in Verilog Hardcover, Springer – May 31, 2001 g., S. 12–17.
- [6] Greene B. and McDaniel M. The Cortex-A15 Verification Story // DVClub, Austin, december 7, 2011 g., S. 1–7.
- [7] Камкин А.С., Коцыняк А.М., Смолов С.А., Татарников А.Д., Чупилко М.М., Сортов А.А. Средства функциональной верификации микропроцессоров / Сб. трудов Института системного программирования РАН Т. 26. 2014 С. 149–206.
- [8] Мешков А.Н., Рыжов М.П., Шмелев В.А. Развитие средств верификации микропроцессора «Эльбрус-2S» // Вопросы радиоэлектроники. 2014. Т. 4. № 3. С. 5–17.
- [9] Путря Ф.М. Применение генераторов случайных программ и случайных фоновых воздействий при функциональной верификации многоядерных систем на кристалле: материалы седьмой международной конференции «Автоматизация проектирования дискретных систем», Минск, 16-17 ноября 2010 г., С. 234–241.
- [10] Гаращенко А.В, Гагарина Л.Г., Федотова Е.Л., Высочкин А.В., Зайцев В.В. Разработка генератора верификационных тестов для многоядерных структур // Информатизация и связь. 2017. №4. С. 20–25.

System of Combined Specialized Test Generators for the New Generation of VLIW DSP Processors with Elcore50 Architecture

A.V. Garashchenko^{1,2}, A.V. Nikolaev¹, F.M. Putrya¹, S.S. Sardaryan²

¹Open Joint-Stock Company Research & Development Center «ELVEES», Zelenograd, Moscow

²National Research University of Electronic Technology (MIET), Zelenograd, Moscow,

ant.gar1@mail.ru

Abstract — In connection with the architectural complexity of modern multi-core structures, more than 60% of the design resources are spent on verification during the development of the processor. Automatic generation of tests is often used to increase test coverage and reduce overall test time. Therefore, the creation of verification test generators to verify the correct operation of microprocessors is becoming increasingly important.

This paper describes the technique of development of the several tests generators used for microprocessor verification. The first one is designed for generating VLIW of packets. The second one is for the verification of the control flow. With the help of it creates sequences of assembler instructions are created to check the pipeline. Software and hardware cycles, subprogram calls, conditional and unconditional conversions are possible. The third generator is aimed at checking the cache memory of the processor. It is based on the graph model of the memory subsystem built on its description.

In the suggested approach, the source code of the tests is constructed by using combinatorial techniques, that is all possible combinations of instructions, situations, and dependencies are sorted taking into account the constraints that direct the tests to check certain situations and also exclude the possibility of generating infinite cycles. The generated test sequences allow for various tests.

To create the more complex tests possible integration of generators into each other is considered since the interaction of different devices of the processor generates a large number of critical situations. The proposed approach makes it possible to improve the efficiency of microprocessor testing.

Keywords — verification of processors, wide command word, memory subsystem, test generation, coverage.

REFERENCES

- Vigyan Singhal, Starting Formal Right from Formal Test Plannin, Oski Technology, Verification Academy at DAC-2015. S. 1–10.
- [2] David M. Russinoff. Formal Verification of Floating-Point RTL at AMD Using the ACL2 Theorem Prover. In

Computer-Aided Reasoning: ACL2 Case Studies, chapter 13. Kluwer Academic Publishers. 2000. S. 1–8.

- [3] Kamkin A.S., Petrochenkov M.V. Sistema podderzhki verifikacii realizacij protokolov kogerentnosti s ispol'zovaniem formal'nyh metodov (System for supporting the verification of coherence protocol implementations using formal methods) // Voprosy radioehlektroniki. 2014. T. 5. № 2. S. 5–17.
- [4] Karpov YU.G. Model Checking. Verifikaciya parallel'nyh i raspredelennyh programmnyh sistem (Verification of parallel and distributed software systems) SPb.: BHV-Peterburg, 2010. 560 s.
- [5] Bening, Lionel, Foster, Harry D. Principles of Verifiable RTL Design: A functional coding style supporting verification processes in Verilog Hardcover, Springer – May 31, 2001 g., S. 12–17.
- [6] Greene B. and McDaniel M. The Cortex-A15 Verification Story // DVClub, Austin, december 7, 2011 g., S. 1–7.
- [7] Kamkin A.S., Kocynyak A.M., Smolov S.A., Tatarnikov A.D., CHupilko M.M., Sortov A.A. Sredstva funkcionalnoj verifikacii mikroprocessorov (Tools for Functional Verification of Microprocessors) / Sb. trudov Instituta sistemnogo programmirovaniya RAN T. 26. 2014 S. 149– 206.
- [8] Meshkov A.N., Ryzhov M.P., SHmelev V.A. Razvitie sredstv verifikacii mikroprocessora «Elbrus-2S» (The development of the verification tools of the "Elbrus-2s" microprocessor) // Voprosy radioehlektroniki. 2014. T. 4. № 3. S. 5–17.
- [9] Putrya F.M. Primenenie generatorov sluchajnyh programm i sluchajnyh fonovyh vozdejstvij pri funkcional'noj verifikacii mnogoyadernyh sistem na kristalle (Application of generators of random programs and random background influences in the functional verification of multi-core systems on a chip): materialy sed'moj mezhdunarodnoj konferencii "Avtomatizaciya proektirovaniya diskretnyh sistem". 16–17 noyabrya 2010 g., Minsk, S. 234–241.
- [10] Garashchenko A.V, Gagarina L.G., Fedotova E.L., Vysochkin A.V., Zajcev V.V. Razrabotka generatora verifikacionnyh testov dlya mnogoyadernyh struktur (Development of a verification test generator for multi-core structures) // Informatizaciya i svyaz'. 2017. №4. S. 20–25.

Practical Aspects of Formal Verification of Networking Chips

A.A. Sokhatski

Cisco Systems Inc., asokhats@cisco.com

Abstract — Formal Verification (FV) is becoming now important part of Design Verification (DV) especially for areas which has higher requirements for quality of chips such as networking chips which should work 24x7 without failures while re-spin cost for huge chips is high. It is difficult to cover all possible scenarios by simulation having a lot of corner cases for packet alignments, sizes, combinations of values of configuration ports and registers. Formal Verification should be able to help to improve quality and reduce Time to Market but it requires:

- Selection of right scope, candidate and method for Formal Verification;
- Addressing Formal challenges, main of such is fighting with complexity and exponential grow of proof time with each proof cycle;
- Consistent Methodology to ensure verification coverage and to reduce effort.

The paper goes through those aspects basing on experience at Cisco Systems Inc. with help from OSKI Technology [1], Formal Verification service provider and Formal sign-off company[2]. The paper covers:

- 1) Brief review of Formal Applications while concentrating on End-to-End Formal sign-off for Design Modules along with criteria for selection of good candidates for Formal;
- 2) Structure of simple Formal Environment;
- Methods helping to fight with complexity which author found especially useful for data transport modules of networking chips, such as floating pulse method, Wolper method, use of abstraction models;
- 4) Formal methodology aspects including:
- Document and test plan flow;
- Run Flow, Regression & Scripting;
- Coverage Flow;
- Reuse for Formal & Simulation.

Keywords — Design Verification, Formal Verification, SystemVerilog, SVA.

I. INTRODUCTION

This paper is primarily based on the experience in Design Verification of networking chips at Cisco Systems

Inc. There is good history of using Formal Verification at Cisco. However in our department we found that we need more consistent shared flow and infrastructure support. We started from training from OSKI Technology, recognized experts delivered Formal Verification services, and then worked on establishing Formal Methodology.

Formal Verification now is a good complimentary to simulation based Design Verification [3]. It has advantages such as:

- High quality: potentially exhaustive proof for implemented checkers for any legal input sequence and configuration combinations; is able to catch corner case bugs;
- Typically less time to setup and verify; could be started by designer at early stages.

On the other hand it has the following restrictions and challenges:

- Restrictions on design complexity and input sequence length;
- Might require application of special techniques to fight with complexity which needs expertise;
- Time to closure is hard to predict, sometimes it's even difficult to tell if design if suitable for FV.

The key point is to select right FV application, level and design entity.

II. FORMAL APPLICATIONS & DESIGN SELECTION

A. Formal applications types

Formal applications [4], [5] could be classified by objective, automation and effort from user, in particular:

1) Automated Formal Linter which doesn't require writing user code but allows to find such issues as array boundary violation, multiple active drivers for the signal; some tools consider check for dead code, etc. from the next section also as part of Automated Formal check;

2) Formal Apps for particular verification aspects including:

- Formal Coverage Analysis (FCA) [6]: allows to find dead code, unreachable FSM state, etc.; we made it as part of DV Flow; pretty useful for simulation coverage closure;
- Connectivity Check: allows to describe in simple way source, destination points and conditions for

connectivity checks; easy to apply, could be used at different levels including chip level;

- X-Propagation: allows to find RTL issues even before simulation started, especially critical for X-optimism when RTL simulation shows concrete value instead of 'X';

3) Assertion Based Design Exploration / Bug Hunting: assumes writing some number of checker assertions (along with assumptions and coverage assertions) and trying to prove them; it could be used from early stage of the project by designer to late stage by DV engineer, some applications are listed below:

- Designer or DV engineer describes assertions for DUT interfaces; for input interfaces assertions should be turned later to assumptions by Formal Tool commands; for standard interfaces Assertion VIP could be used if available; Formal Tool should be able to check compliance with interface protocols [7]; internal assertions for modules could be added by designer and verified as well;
- Designer could add to that coverage assertions exercising some simple or corner-case scenarios and get waveforms without running simulation;
- At later stage of the project along with simulation DV engineer could write specific assertions to hit and verify corner case scenarios and close coverage; advanced Bug Hunting techniques could use as initial state for Formal proof, state, where design comes after some simulation cycles;
- During top-level simulation or even while exercising manufactured chip there could be found issues which require reproducing them; Formal could help here and also ensure that fix is complete;

4) Equivalence Check: assumes comparison between reference design / model and design to be verified; historically it was one of the first widely used Formal applications to check equivalence between RTL and synthesized code; now dedicated tools are introduced which do comparison between two RTL designs or even with C-model; this application requires RTL / accurate reference model;

5) End-to-end sign-off check of design modules basing on SVA assertions which assumes exhaustive verification at some scope.

While all Formal applications are pretty beneficial, the paper is focusing on the End-to-end sign-off Formal Verification which could replace simulation for certain design modules, release DV engineering resources and exhaustively check design at that level. Formal could be applied at different levels including architectural. The paper primarily talks about Formal Verification of RTL design code.

B. Design Block selection for Formal Verification

Due to Formal Verification complexity & potential exponential grow of Proof time for each next clock cycle because of exhaustive nature of Formal proof, special care needs to be taken when selecting modules for Formal proof. The following factors need to be taken into account when identifying modules for formal proof:

- Type of the design, best fit is control or data transport design; data transform design having multipliers, wide adders inside is difficult for Formal model check, for this type of design equivalence check might work;
- Complexity of design: size of design should be decent, pipeline depth and input sequence for verification should not be too long; it is difficult to give certain numbers as it depends on functional complexity, symmetry of design, used Formal Tool, etc., just some example of design proven by FV has several thousands of flip-flops, several hundred inputs, summary of pipeline depth and input sequence to prove – around 20 cycles;
- How much benefits we are getting from Formal comparing to simulation: number of corner cases, configuration combinations, is it reuse block which should work in different modes and should be exhaustively verified;
- Status and history of verification: if previous version was verified via simulation, how different is the new one; what is simulation coverage; were any bugs found after simulation verification done.

After identifying candidates, evaluation and planning needs to be done including:

- Measuring design complexity: number of flip-flops, delay;
- Better understanding design, parameters;
- Matching skill set of available FV resources.
 - III. STRUCTURE OF SIMPLE FORMAL ENVIRONMENT

Formal Environment for end-to-end check targeting sign-off typically includes:

- Interface components implemented protocol checks;
- End-to-end checkers.

Structure presented in the figure below.



Figure 1. Simple Formal Environment Structure

Interface component features and applications:

- Located in separate modules;
- Contains SVA checker assertions, assistant code and coverage assertions, no assumptions;

- Assertion naming convention should reflect direction, for example source-destination: *src__dst_<name>;* it is critical when signals of both directions are integrated inside the same interface;
- Assertions changed to constraints (assumptions) from inside Formal environment script for input signals / Input Interface Components;
- Interface components instantiated inside simulation environment as well as in Formal which is especially important for Input Interface Components to check for over-constraints (failure of assertion in simulation is sign of over-constraint);
- Interface Components could contain intermediate results which could be used inside End-to-end Checkers

End-to-end checker features:

- Typically more complex than Interface Components and require special methods to fight with complexity and exponential time grow; the methods are described in the next section;
- Could use intermediate results from Interface Components.

As an example of data transport module from networking chip we'll take Delimiter Removal module which converts one packet protocol with Start-Of-Packet (SOP) – End-Of-Packet (EOP) delimiters located inside data flow to another protocol which has those signals located in separate signals. Next figure illustrates the simplified function. In real life the protocols are more complicated: there are several bytes passed each cycle; packet with low priority could be interrupted by packet with high priority.



Figure 2. Simplified function of Delimiter Removal Module

Here Input Protocol is more complex and requires functions to extract SOP-EOP flags from data stream using assistant Verilog code. The results could be used by End-toend checkers.

IV. METHODS TO REDUCE PROOF TIME

Formal Verification exhaustively verifies design for any legal combination of inputs and design states. It starts from initial state (typically after reset) and checks behavior for any combination of input signals. Then goes from the set of reached states and checks any combinations of inputs again, etc. That is why each next step (clock cycle) proof could take same amount of time as for all previous steps together leading to exponential grow of proof time depending on proof depth (number of clock cycles). It makes sometime non-realistic to get full proof or reach required proof depth in reasonable time. There are several approaches how to fight with that by reducing proof complexity. Some of them are described below.

A. Use of symbolic variables

Basic idea: we are tracking and checking only one arbitrary instance of design or sequence item selected by symbolic variable. It could significantly reduce proof time. Note that we are not extra restricting input space but allowing any legal input sequences.

For example, if we have several input ports and several output ports we could track only transactions which go from some arbitrary selected input to some arbitrary selected output port. Selected variable values should be unchanged through the proof.

Symbolic variables also could be used to select one arbitrary bit inside byte, for example:

B. Selection of sequence item with Floating Pulse method

Here we are tracking and checking only one sequence item, it could be byte, it could be packet or another transaction. Floating pulse is binary signal which is constrained to be active during any but only one cycle. It selects the sequence item.

Item which is going to be tracked is typically marked (colored) at the input – so we should be able to detect it at the output. One bit could be used for coloring data byte. That bit has inactive (typically 0) value for all bytes except colored byte when the bit has active (typically 1) value. It is done via constraints (SVA assumptions).

Here is illustration of application of floating pulse method for Delimiter Removal Module.



Figure 3. Floating Pulse method application

Example of code for generation of floating pulse:

wire floating_pulse; reg floating_pulse_reg; always @ (posedge clk) begin if (reset)

```
floating_pulse_reg <= 1'b0;
else if (floating_pulse)
    floating_pulse_reg <= 1'b1;
floating_pulse_model: assert property (
    floating_pulse_reg |-> !floating_pulse
);
```

Example of code for coloring bit 0 inside input byte:

```
color_bit: assert property(
  @(posedge clk) disable iff (reset)
  data_bus[0] == floating_pulse
);
```

In the discussed Formal environment Floating Pulse method used in the following checkers:

 Check that any valid input byte after some number of cycles is detected at the output (Forward Progress Checker): additional counter is used which counts valid cycles after floating pulse before colored byte is detected at the output; counter value is compared with pipeline depth via assertion;

2) Check that packet boundary is preserved at the output: at the input we arbitrary color first byte of a packet and then check that colored byte is still the first at the output

Another method described below is used to check byte contents and bytes order.

C. Using Wolper method to check data contents and order

Wolper method colors two consecutive items at the input and checks that only two consecutive items are colored at the output. For completeness both 0 and 1 should be used for coloring. This method allows to prove that data contents and order is preserved, no drops, no injections.

It is used in the discussed environment for that purpose.

Complication here though for Delimiter Removal Module is that it should maintain order of bytes within certain packet priority but packets of low priority could be interrupted by higher priority packet. That is why two consecutive bytes of low priority packets could be apart. It requires additional proof depth and extra care. So it makes difficult to reach required proof in reasonable time. Case splitting approach described below is used to resolve it.

D. Splitting Checkers

Idea behind that technique is to reduce complexity – Cone of Influence (COI) of each checker. It could be done different ways:

- By slitting property when complex expression is used on the right side of implication;
- By defining separate checkers for different functional aspects, for example, different operations;
- By defining separate checkers for different modes;
- Using Assume-Guarantee approach with selecting internal points and defining checkers to and from internal points

In the discussed Formal environment we had to fix values of some symbolic variables used along with Wolper method and run Formal proof for various values separately. For example position of bit inside byte was fixed to certain value, position of byte inside input bus was fixed. Note that in some cases after design analysis and discussion with designer, conclusion could be made that it is enough to make proof only for some corner case values of symbolic variables, for example first and last bit position.

In the following couple of sections we are discussing aspects of design abstraction models restricting them to few typical examples.

E. Reset Abstraction Model

It is difficult or sometimes not possible to reach large proof depth in reasonable time. On the other hand we need to ensure that design is working properly from any state even if it requires long sequence to reach that state.

For example, design has wide counter and action happened when counter reaches some big value. If we start at reset state then big number of cycles is required to exercise the action.

The idea behind Reset Abstraction is to make action state closer. To make it happened instead of getting initial value at reset we are allowing any value. So counter could get high value right at reset and action could be verified. Special care might need to be taken to sync other signals with arbitrary reset value.

Here is example from the discussed Formal environment. Here we have packet length counter which is reset for every new packet and if packet count reaches certain configurable value, packet should be marked with error and truncated. Reset abstraction leaves value at reset not driven but keep all other functionality not touched. This is illustrated in the next figure.



Figure 4. Example of Counter Reset Abstraction

F. Memory & FIFO Abstraction Models

Other typical examples of abstraction models which are used to reduce proof complexity are Memory & FIFO Abstraction models.

Example of Memory Abstraction Model:

Maintains data only for one symbolic address : writes and reads data when actual address matches the symbolic one; - Returns random data for all other addresses.

Examples of FIFO Abstraction Model:

- Use symbolic "any" depth of FIFO: it will allow to reach full condition earlier;
- If depth cannot be compromised and made symbolic, then use reset abstraction for read and write pointers and sync other signals with them;
- Keep track of only one data entry selected by floating pulse or by colored bit inside the data

V. FORMAL METHODOLOGY ASPECTS

A. Documents & Test Plan Flow

Formal Verification is less standardized than simulation. That is why having good set of documentation is especially important for knowledge transfer.

We developed some number of documents for the Formal flow support including:

- Formal Verification Process Description which describes phases and milestones from Formal Planning to Formal Complete, focus and exit criteria for each phase including reviews and approvals of checklists;
- Formal Milestone Checklist for different milestones to ensure that nothing is missed; examples of checklist items: "Reviewed Required Proof Depth with designer", "All tests in Formal regression passed";
- Formal Environment Template, including description of Interface Components; End-to-end Checkers, Test bench Configurations, Interesting scenarios, etc.
- Test Plan template.

Test Plan is supported by in-house tool and shared with simulation. In the Test Plan some intends could be covered by simulation and some intends could be covered by Formal verification methods. Test plan metrics contain references to actual code: for Formal proof metrics we require reference not just to check assertion but also to coverage assertion.

B. Run Flow, Regressions & Scripting

One could use just Formal Tool to run Formal tests especially at initial stages with GUI. However at some point typically you need to run Formal tool for different constraints, parameters, etc. Eventually you will have regression list which will be run in batch mode.

In-house script simplifies passing tests to Formal tool, setting signals and parameters, selecting assertions. The same script is used for running simulation as well.

From our experience it is difficult to get full proof for end-to-end checkers but bounded proof for certain number of cycles could be enough. That is why it is important to analyze design and calculate Required Proof Depth (RPD) taking into account:

- Pipeline depth / Latency of the design;
- Results of Microarchitecture Analysis with and without designer;

Length of input sequences which require proof including sequences for interesting corner case scenarios.

If Formal proof reaches RPD when proving checker or coverage property, we could consider that property is proven; if all properties are proven then the test passes. Support for defining and checking RPD is part of the flow.

We have regression support shared with simulation which allows to run set of tests with various combinations of configuration values when needed for case splitting.

C. Coverage Flow

Formal covers all possible combinations of input stimulus and states inside Cone Of Influence (COI) of particular checker assertion but we need to ensure that all design constructs are covered, there is no over-constraints, there is enough checkers, so they are able to catch DUT bugs.

We used the following techniques / rules to get confidence that DUT is formally verified:

1) Interesting scenarios

Coverage properties should be implemented to ensure that "interesting" corner case scenarios could be reached within RPD. It is recommended to include check for output results and ensure that design comes to idle state. Here is some example below:

```
intscen_interleave_cover: cover property (
// input sequence
(Srdy && S_intf.sop && (S_intf.pri==PRI_MC))##1
(Srdy && S_intf.sop && (S_intf.pri==PRI_UC))##1
. . .
##[1:PIPE_DELAY]
// output results & state
((uc_cnt==2) && (mc_cnt==2)&& D_intf.idle)
);
```

2) Coverage for each group of checkers

Each group of checker assertions should be accompanied by coverage assertions to ensure hitting corner case scenarios

3) Controllability Input Stimulus coverage.

Main purpose here is to ensure that there is no overconstraints which do not allow to cover certain parts of design. The following flow support was implemented:

- Collect Formal coverage including line, FSM, toggle coverage:
 - For Formal environment with disabled constraints;
 - For regular Formal environment with constraints;
- Compare the coverage results to ensure there is no overconstraints.
 - 4) Observability Checkers coverage.

This step is to ensure that all parts of design are actually checked by at least one checker. The step is more dependent than the other steps on Formal Tool which could help to collect such information.

Some tools could collect Proof-Core coverage for a checker which covers constructs really affecting particular checker results. The Proof-Core coverage per checker is merged for DUT.

Another approach which could be used along or without Proof-Core coverage and we applied is mutation coverage. It requires more time but gives more confidence. It could be implemented outside of Formal Tool. We use in-house mutation run script with the following base algorithm:

- Do the following for every flip-flop in the design (for data buses it could be restricted to only LSB and MSB bits):
 - Inject bug into design; We used constant 0 for the flip-flop;
 - o Do Formal proof;
 - Ensure that some checker fails.

It could be easily implemented, fast enough, gives extra confidence for Formal and much more than regular code coverage for simulation.

D. Reuse

Reuse is important part of any flow. We have some reuse components in place and some to be implemented including:

- Components instantiated inside Formal Environment:
 - o FIFO,
 - o Pipeline / Delay component,
 - Components supporting Formal methods: Floating Pulse, Wolper;
- Deign abstraction models:
 - o Memory,
 - o FIFO.
 - VI. CONCLUSIONS

Formal Verification could be successfully used along with simulation in various applications.

We got the most benefits using end-to-end Formal Verification for design modules of decent size with big number of combinations of packet flow parameters and corner cases. It allowed to find corner case bugs & increase verification quality.

In order to reach results we had to establish consistent Formal Verification flow, including coverage flow and apply techniques which help to fight with FV complexity.

ACKNOWLEDGEMENTS

Author thanks OSKI Technology team who delivered training on Formal Verification and helped to establish FV flow and his colleagues from Cisco Systems who worked on development of the flow.

REFERENCES

- [1] Oski Technology. Formal Verification Methodology to enable Formal Sign-off. Available at http://www.oskitechnology.com (accessed 03.05.2018)
- [2] Singhal V. The evolution of Formal Verification Sign-off. Available at https://www.cadence.com/content/dam/cadencewww/global/en_US/documents/company/Events/jug/secure d/jug-2017/wed-9-30am-fv-signoff-evolution-oski.pdf (accessed 03.05.2018)
- [3] Sokhatski A.A. Practical Aspects of Design Verification of Complex Chips // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2016. Part2. P. 16-23.
- [4] Seligman E., Schubert T., Kumar A.K. Formal Verification: An Essential Toolkit for Modern VLSI Design. Waltham, MA, USA: Elsevier, 2015, 352P.
- [5] Murphy B., Pandey M., Safarpour S., Finding Your Way Through Formal Verification. Danville, CA, USA: SemiWiki LLC, 2018, 133P
- [6] Tatarnikov Y., Labib K. Using Synopsys VC Formal Coverage Analyser (FCA) for Code Coverage Improvement. Available at https://www.synopsys.com/community/snug/snugsilicon-valley/location-proceedings-2016.html (accessed 03.05.2018)
- [7] Tatarnikov Y., Labib K. Next step of Formal Verification utilization Available at https://www.synopsys.com/community/snug/snug-siliconvalley/location-proceedings-2018.html (accessed 03.05.2018)

Практические аспекты формальной верификации проектов блоков сетевых СБИС

А.А. Сохацкий

Сиско Системс Инк., asokhats@cisco.com

Аннотация — Формальная верификация в наши дни важной составляющей верификации стяновится проектов цифровых блоков в особенности в областях, предъявляющих повышенные требования к качеству проверки. Так сетевые СБИС должны фунционировать безошибочно без перерывов в течении длительного времени, при том, что перепроектирование и повторное изготовление этих коплексных микросхем требует существенных затрат. Довольно сложно проверить функционирование на всех возможных входных последовательностях путем моделирования при наличии множества граничных условий связанных с различной длиной и выравниванием пакетов, комбинациями значений входных настроечных портов и регистров. Формальные методы должны помочь достичь полноту проверки и сократить время разработки, но это требует:

- правильной стратегии выбора блоков и модулей проекта для формальной верификации, а также метода формальной верификации;
- применения решений для ответа на проблему экспотенциального роста времени формального доказательства в зависимости от глубины доказательства;
- последовательной методологии для обеспечения верификационного покрытия и сокращения затрат.

В статье рассматриваются вопросы формальной верификации на основе опыта работы автора в компании Сиско Системс Инк. и консультаций поставщика услуг формальной верификации компании OSKI Technology. Излагаются следующие аспекты:

- Краткий обзор применений формальной верификации. При этом статья фокусируется на задаче полной (sign-off) сквозной проверки (end-toend check) отдельных модулей проекта. Рассатриваются вопросы выбора модулей для формальной веривикации;
- 2) Структура простого окружения для формальной верификации;
- 3) Методы, позволяющие увеличить глубину доказательства, в частности, использование

символических переменных, символического выбора элемента последовательности с применением плавающего импульса (floating pulse), метод Волпера (Wolper), использование абстрактных моделей;

4) Аспекты методологии формальной верификации, включая технологии планирования, документирования, исполнения и регрессионных последовательностей, покрытия проекта, совместного использования для формальной верификации и моделирования

Ключевые слова — СБИС, формальная верификация, моделирование, RTL, SystemVerilog, SVA.

ЛИТЕРАТУРА

- [1] Oski Technology. Formal Verification Methodology to enable Formal Sign-off. Available at http://www.oskitechnology.com (accessed 03.05.2018)
- [2] Singhal V. The evolution of Formal Verification Sign-off. Available at https://www.cadence.com/content/dam/cadencewww/global/en_US/documents/company/Events/jug/secure d/jug-2017/wed-9-30am-fv-signoff-evolution-oski.pdf (accessed 03.05.2018)
- [3] Сохацкий А.А. Практические аспекты верификации проектов СБИС // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №2. С. 16-23.
- [4] Seligman E., Schubert T., Kumar A.K. Formal Verification: An Essential Toolkit for Modern VLSI Design. Waltham, MA, USA: Elsevier, 2015, 352P.
- [5] Murphy B., Pandey M., Safarpour S., Finding Your Way Through Formal Verification. Danville, CA, USA: SemiWiki LLC, 2018, 133P
- [6] Tatarnikov Y., Labib K. Using Synopsys VC Formal Coverage Analyser (FCA) for Code Coverage Improvement. Available at https://www.synopsys.com/community/snug/snugsilicon-valley/location-proceedings-2016.html (accessed 03.05.2018)
- [7] Tatarnikov Y., Labib K. Next step of Formal Verification utilization Available at https://www.synopsys.com/community/snug/snug-siliconvalley/location-proceedings-2018.html (accessed 03.05.2018).

Принципы проектирования устройств тестового диагностирования быстродействующих микросхем и модулей полупроводниковой памяти

А.П. Евдокимов, В.Г. Рябцев, А.В. Меликов

Волгоградский государственный аграрный университет, akim.onoke@mail.ru

Аннотация — Рассматривается проблема повышения быстродействия устройств тестового диагностирования микросхем и модулей памяти. Предложена структура мультипроцессорного устройства тестового диагностирования, обладающего высоким быстродействием и обеспечивающего одновременное формирование тестовых воздействий для нескольких смежных тактов за один период сигнала синхронизации.

Ключевые слова — микросхема памяти, тестовое диагностирование, устройство тестового диагностирования.

I. Введение

Вместе с микропроцессорами, быстродействие которых ежегодно возрастает, улучшаются также параметры полупроводниковых запоминающих устройств, что повышает общую эффективность вычислительной системы. Современные микросхемы памяти содержат программируемый контроллер, логический интерфейс приема-передачи данных, это позволяет отнести их к цифровым системам, содержащим встроенную память.

Для повышения технических характеристик микросхем памяти многие фирмы применяют новые технологии [1]. Например, фирма Samsung Semiconductor, Inc с 1998 года ведет разработку технологии полупроводниковых запоминающих устройств DDR SDRAM (Double Data Rate SDRAM), которая является логическим продолжением синхронной памяти (SDRAM). Микросхемы памяти DDR SDRAM были одобрены в качестве стандарта Объединенным техническим советом по электронным приборам (JEDEC) и в настоящее изготавливаются многими фирмами [2]. Технология DDR позволяет записывать и считывать данные с частотой в два раза большей, чем частота системной шины, за счет передачи данных по обоим фронтам синхросигнала.

Частота передачи данных в микросхемах DDR1 составляет 400 МГц, в микросхемах DDR2 увеличена до 800 МГц, а в микросхемах DDR3 достигает 1800 МГц. В микросхемах памяти типа DDR4 частота достигает 3,2 ГГц. В 2018 году планируется выпуск микросхем типа DDR5, которые будут работать на частоте 3,2 ГГц, в недалеком будущем частота может увеличиться до 6,4 ГГц.

Эффективное диагностирование тестовое быстродействующих микросхем и модулей памяти можно выполнять только на реальной рабочей частоте. Однако максимальная частота устройств тестового диагностирования больших интегральных схем оперативных запоминающих устройств (БИС ОЗУ) ограничена быстродействием блока микропрограммного управления, который формирует микрооперации для алгоритмической генерации тестовых воздействий и эталонных реакций [3, 4].

Вместе с выпуском микросхем и модулей памяти нового типа проектируются тестеры для выполнения тестового диагностирования на рабочей частоте. Так компания CST Inc., частная занимаюшаяся проектированием тестеров для полупроводниковой памяти, анонсировала свой новый тестер Eureka DDR4 2400. Данный тестер способен тестировать модули DDR4 UDIMM / RDIMM / LRDIMM и SODIMM с широким диапазоном частот от 1600 до 2400 МГц. Tecrep Eureka DDR4 2400 также может быть сопряжен с загрузчиком RoboFlex CST для автоматизации массового тестирования [5]. Однако структура и архитектура данного тестера и других аналогичных устройств не раскрыта в литературе подробно, что не позволяет производить их детальный анализ.

Отличительной особенностью тестеров для диагностирования микросхем и модулей памяти является алгоритмический способ формирования тестов под управлением блока микропрограммного управления (БМУ). Длительность минимального цикла БМУ определяется промежутком времени от момента выделения условий переходов до момента получения управляющих сигналов и вычисляется по формуле:

$$T_{u} = T_{y} + T_{a} + T_{\mathcal{M}} + T_{p} + T_{\partial};$$

где T_y – время выделения условий переходов, формируемых операционными секциями;

T_a – время формирования адреса следующей микрокоманды;

 T_{M} – время выборки памяти микрокоманд;

*Т*_{*p*} – время переключения регистра микрокоманды;

*T*_л – время дешифрации полей микрокоманды.

Целью настоящей работы является разработка принципов построения структуры и архитектуры

устройств тестового диагностирования быстродействующих микросхем и модулей полупроводниковой памяти.

II. МЕТОД РАСПАРАЛЛЕЛИВАНИЯ ОПЕРАЦИИ ТЕСТОВОГО ДИАГНОСТИРОВАНИЯ ПОЛУПРОВОДНИКОВОЙ ПАМЯТИ

Значительного повышения быстродействия тестового диагностирования БИС ОЗУ можно достичь за счет распараллеливания процесса формирования векторов воздействий, последовательной передачи их при помощи коммутаторов на входы диагностируемой микросхемы памяти и параллельной обработки считанных реакций [6-8]. Для реализации данного метода повышения быстродействия предлагается в структуру устройства диагностирования микросхем и модулей памяти ввести группы блоков формирования адреса, блоков формирования данных и блоков сравнения. Учитывая, что число ячеек микросхем памяти кратно 2^n , где n – любое целое положительное, поэтому количество блоков $p = 2^r$, r = 0, 1, 2, 3, 4 и т.д., что позволяет равномерно распределить ячейки между формирователями программ тестов.

Предлагаемый метод позволяет снизить требования по быстродействию к основным блокам устройства и увеличить частоту формирования тестовых воздействий и обработки эталонных реакций. При этом частота работы БМУ вычисляется по формуле:

$$f_{mpu}=f_d/p$$
,

где *f_{mpu}* – частота работы блока микропрограммного управления;

 f_d – частота диагностирования микросхем памяти;

 р – коэффициент распараллеливания операций формирования векторов воздействий и обработки считанных реакций.

диагностирования III. АРХИТЕКТУРА МУЛЬТИПРОЦЕССОРНОГО УСТРОЙСТВА и модулей ТЕСТОВОГО ДИАГНОСТИРОВАНИЯ ПОЛУПРОВОДНИКОВОЙ ПАМЯТИ

Структура мультипроцессорного устройства тестового диагностирования (УТД) состоит из взаимодействующих управляющего и нескольких операционных процессоров:

$$M = \{ Y\Pi, O\Pi_0, O\Pi_1, ..., O\Pi_{p-1} \},\$$

где УП – управляющий процессор;

 $O\Pi_0, O\Pi_1, \ldots, O\Pi_{p-1}$ – операционные процессоры;

р – число параллельных операционных процессоров.

Каждый операционный процессор формирует код адреса диагностируемой запоминающей ячейки А_i и коды данных Т_i, которые используются для задания входных данных объекта диагностирования при выполнении операции записи, а также как эталонные данные при выполнении операций сравнения ответных реакций. Структурная схема мультипроцессорного УТД, состоящего из управляющего процессора (УП) и операционных процессоров нескольких $O\Pi_i = \{FT_i, FA_i\},\$ из формирователей собранных данных FT_i и формирователей адреса FA_i, приведена на рис. 1. Управляющий процессор формирует коды микроопераций, задающих режимы работы, для процессоров операционных обеспечивает И формирование вектора рабочих микроопераций:

$$C = \{W, R, XOR, M\},\$$

где W – код микрооперации записи данных в диагностируемую ячейку; R – код операции считывания данных из диагностируемой ячейки; XOR – код операции сравнивания считанных и эталонных данных; М – код маски, обычно используется для управления электронными узлами согласования устройства и объекта диагностирования.



Рис. 1. Структурная схема устройства тестового диагностирования

Коды рабочих микроопераций образуют множества, число элементов в каждом из которых равно количеству параллельно работающих операционных процессоров:

$$W = \{W_0, W_1, ..., W_{p-1}\};\$$

$$R = \{R_0, R_1, ..., R_{p-1}\};\$$

$$XOR = \{XOR_0, XOR_1, ..., XOR_{p-1}\};\$$

$$M = \{M_0, M_1, ..., M_{p-1}\}.$$

Управляющий процессор обеспечивает передачу кодов микроопераций формирователям операционных процессоров и генерирует коды рабочих операций для нескольких смежных тактов диагностирования. Преобразование параллельных кодов данных, операций и адреса в последовательные коды обеспечивают коммутаторы K1, K2 и K3 соответственно. Хранение кодов тестовых воздействий осуществляется регистрами RT, RC и RA.

Вектор тестовых воздействий $V = \{V_0, V_1, ..., V_{i}, ..., V_{p-1}\}$ определяется при помощи следующих условий:

$$\mathbf{V} = \begin{cases} \{\mathbf{A}_{0}, \mathbf{C}_{0}, \mathbf{T}_{0}\} \Leftarrow \mathbf{E}\mathbf{A} = 0; \\ \{\mathbf{A}_{1}, \mathbf{C}_{1}, \mathbf{T}_{1}\} \Leftarrow \mathbf{E}\mathbf{A} = 1; \\ \\ \\ \mathbf{A}_{p-1}, \mathbf{C}_{p-1}, \mathbf{T}_{p-1}\} \Leftarrow \mathbf{E}\mathbf{A} = p-1, \end{cases}$$

где EA – коды выбора направления передачи векторов воздействий через коммутаторы К₁, К₂, и К₃.

Множества тестовых воздействий, формируемые ОП параллельно, могут быть пересекающимися, что соответствует обращению к общим запоминающим элементам объекта диагностирования. В предлагаемой архитектуре УТД коды выбора направления передачи данных формируется в режиме двоичного счетчика.

Для повышения производительности обработки ответных реакций в операционных процессорах используется *p* схем сравнения считанных и эталонных данных, выходы которых подключены к входам элемента ИЛИ (OR). Разряды, которым соответствует код единица регистра разрешения сравнения *B*, участвуют в формировании вектора ошибок Error. Таким образом, вектор ошибок, формируемый блоком параллельной обработки считанных реакций, структура которого приведена на рис. 2, определяется по формуле:

$$\operatorname{Error} = \bigvee_{k=0}^{r-1} (\bigvee_{i=0}^{p-1} (D_i \oplus T_i) \wedge \operatorname{xor}_i)_k \wedge b_k,$$

где r – число разрядов данных;

 $\boldsymbol{b}_k \in \mathbf{B}-i$ -ый разряд вектора B разрешения сравнения данных.

Вектор тестовых воздействий, формируемых УП и группой ОП, образует кортеж $V = \langle C, A, T \rangle$. Предшествование вектора тестового воздействия V_1 воздействию V_2 на объект диагностирования (od) обозначим: $V_1 >_{od} V_2$. Тогда тестовый набор векторов воздействий можно представить в виде цепочки последовательных воздействий: $V_0 >_{od} V_1 > ... >_{od} V_{p-1}$.

Каждый вектор V_i , при i = 0, p - 1 формируется соответствующим операционным процессором совместно с управляющим процессором, который вырабатывает коды рабочих микроопераций C_i для каждого вектора V_i приведенной выше цепочки.

Цепочка воздействий называется полной, если ее элементы не содержат пустые рабочие микрооперации. Цепочка предшествующих воздействий будет неполной, если хотя бы один из кодов рабочих микроопераций является пустым оператором. Наличие пустых операторов в тестовом наборе приводит к возникновению пропусков тактов обращения к диагностированному изделию, что снижает и увеличивает эффективность диагностирования продолжительность выполнения тестов. Для исключения пропусков тактов обращения необходимо, чтобы количество рабочих микроопераций в цепочках воздействий было кратно числу применяемых операционных процессоров [9].

Обычно учитываются только операции непосредственного обращения к объекту диагностирования (r_i, w_i), тогда для обеспечения выполнения указанного выше требования необходимо обеспечить выполняемые условия:

$$\sum_{r=1}^{s} \sum_{l=1}^{m} \left(\sum_{i=0}^{p-l} (r_i + w_i) \right)_{rl} / p = z,$$

где r = 1, s – количество циклов повторения команд, формируемых цепочки воздействий;

l = 1, m – число следующих друг за другом цепочек в программе теста;

z – целое положительное число.

Для выполнения данного условия в ряде случаев требуется осуществить реконфигурацию программы теста с целью исключения пропусков тактов обращения к объекту диагностирования [10]. Параллельное формирование кодов адресов диагностируемых ячеек позволяет создать гибкую систему признаков условных переходов. Структура блока формирования признаков, предназначенных для условных выполнения команд переходов И учитывающих коды конечных адресов NX и NY, приведена на рис. 3.


Рис. 2. Структура блока параллельной обработки считанных реакций





Выбор условных переходов обеспечивают коммутаторы К1 и К2, на адресные входы которых подается двоичный код Е, соответствующий номеру адресного формирователя, сигналы схем сравнения которого используется в данном такте диагностирования.

Для реализации команд условных переходов, обеспечивающих обнаружение соответствия кодов текущих адресов X и Y кодам конечных адресов NX, NY, или начальных адресов GX или GY, признаки переходов вычисляются при помощи выражений:
$$\begin{split} F_{\text{nx}} = \begin{cases} (X_0 \neq NX) \Leftarrow E = 0; \\ (X_1 \neq NX) \Leftarrow E = 1; \\ & \dots \\ (X_{p-1} \neq NX) \Leftarrow E = p-1; \end{cases} \\ F_{\text{ny}} = \begin{cases} (Y_0 \neq NY) \Leftarrow E = 0; \\ (Y_1 \neq NY) \Leftarrow E = 1; \\ & \dots \\ (Y_{p-1} \neq NY) \Leftarrow E = p-1; \end{cases} \end{split}$$

$$F_{gx} = \begin{cases} (X_0 \neq GX) \Leftarrow E = 0; \\ (X_1 \neq GX) \Leftarrow E = 1; \\ & \dots \\ (X_{p-1} \neq GX) \Leftarrow E = p-1; \end{cases}$$

$$F_{gy} = \begin{cases} (Y_0 \neq GY) \Leftarrow E = 0; \\ (Y_1 \neq GY) \Leftarrow E = 1; \\ & \dots \\ (Y_{p-1} \neq GY) \Leftarrow E = p-1. \end{cases}$$

Аналогичные выражения можно получить для вычисления значения флагов, учитывающих коды NX и NY или комбинации кодов (X \neq NX) \neq (Y \neq NY), (X \neq GX) \neq (Y \neq GY), которые является дизьюнкцией двух рассмотренных выше условий. Формирователи кода адреса генерирует вектор адреса A = {X, Y}, который обеспечивает выборку строк X и столбцов Y ячеек памяти. Информационные данные T, которые включаются в вектор тестовых воздействий, используются также в качестве кодов эталонных реакций.

IV. ПРОГРАММНАЯ МОДЕЛЬ МИКРОПРОГРАММНЫХ СЧЕТЧИКОВ

При выполнении операции считывания данные из выбранной ячейки памяти заносятся в регистры реакций Dout₀, Dout₁, ..., Dout_{p-1}. Разрешение записи данных обеспечивают синхронизирующие сигналы, которые разнесены по времени друг от друга на период такта обращения к диагностируемому изделию. Считанные и эталонные данные сравниваются схемами сравнения S₀, S₁, ..., S_{p-1}, на выходах которых формируется результат диагностирования, который может маскироваться кодом специального регистра В.

Формирование тестовых воздействий обеспечивается алгоритмическим способом по командам УП, программная модель счетчиков которого приведена на рис. 4.



Рис. 4. Программная модель счетчиков управляющего процессора

Количество и число разрядов счетчиков определяется максимальным количеством циклов повторения микроопераций в наиболее распространенных тестах. Максимальное значение переменных, которые заносятся в счетчики J, Q, H, определяется по формулам:

$$J = \log_2 n - 3, Q = \sqrt{n} - 1, H = n/2 - 1, E = (\sqrt{n} - 4) / 2 - 1,$$

где *n* – емкость памяти диагностируемого изделия.

Количество разрядов счетчика адреса команд Р определяется емкостью микропрограммной памяти, необходимой для хранения программ тестов.

V. РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ ОСНОВНЫХ КОМПОНЕНТОВ УТД

В среде Active-HDL разработан проект УТД, структура которого приведена на рис. 5 в составе: микроконтроллер, счетчик адреса микрокоманд, блок формирования адреса, блок формирования данных, компаратор данных и блок формирования флагов условных переходов. На входы микроконтроллера, переходов поступают коды флагов Flag(2:0). микрокоманды Сор(3:0), сигнал сброса RST и тактовые CLK. Микроконтроллер осуществляет импульсы дешифрацию кодов микрокоманды и формирует определяющие режимы работы микрооперации, остальных блоков УТД. Выполненная верификация проекта подтвердила его работоспособность.

Разработку программ диагностирования изделий полупроводниковой памяти для устройства, имеющего мультипроцессорную структуру, предлагается осуществлять на машиноориентированном языке программирования Prover, синтаксис которого приведен ниже.

СИНТАКСИС МАШИНООРИЕНТИРОВАННОГО ЯЗЫКА PROVER

Следует выделить основные микрооперации, которые при тестировании полупроводниковой памяти для повышения быстродействия надо выполнять параллельно:

а) формирование кода адреса;

б) формирование кода данных;

в) рабочие микрооперации записи (W), считывания (R) и сравнения считанных и эталонных данных (A).

Для разработки программ тестового диагностирования запоминающих устройств создан язык программирования Prover, обеспечивающий задание необходимых команд и микроопераций, которые записываются в соответствии с синтаксисом, приведенном в табл. 1.

Таблица 1

Формат микроопераций языка Prover

| Позиции экрана при записи команд | | | | | | | | | |
|----------------------------------|-------------------|------|----------------|---|--------|------|-------|----|--|
| 1 | 1 2 3 4 5 6 29 30 | | | | | | | | |
| Me | етка, н | юмер | Микрооперации, | | | | | | |
| такта | | | | р | азделе | нные | запят | ой | |



Рис. 5. Структура проекта УТД

В первых трех позициях допускается использовать метки команд, по которым осуществляются передачи управления или номер такта, в котором осуществляется выполнение микроопераций. В четвертой позиции всегда должен размещаться разделительный пробел. В позициях с 5 по 30 помещаются микрооперации, которые разделяются друг от друга запятыми. Символ пробела в этих позициях обозначает окончание команды.

Для формирования кодов адреса применена двумерная система координат (X, Y), что удобно для выбора строк и столбцов запоминающих ячеек. Для синтеза программ применен набор микроопераций формирования кода адреса, приведенный в табл. 2.

Таблица 2

| Микро- операции | Выполняемые действия | | | | |
|--------------------------------------|-----------------------------------|--|--|--|--|
| X:=GX | Запись начального адреса GX в X | | | | |
| X:=X | Сохранение содержимого Х | | | | |
| X:=X+1 | Увеличение Х на единицу | | | | |
| X:=X+2 | Увеличение Х на 2 | | | | |
| X:=X+3 | Увеличение Х на 3 | | | | |
| X:=X+4 | Увеличение Х на 4 | | | | |
| X:=NX | Запись конечного адреса NX в X | | | | |
| Y:=NY | Запись конечного адреса NY в Y | | | | |
| Y:=GY | Запись начального адреса GY в Y | | | | |
| Y:=Y | Сохранение содержимого Ү | | | | |
| Y:=Y+1 | Увеличение У на 1 | | | | |
| Y := Y + 2 | Увеличение У на 2 | | | | |
| Y:=Y+3 | Увеличение У на 3 | | | | |
| Y:=Y+4 | Увеличение У на 4 | | | | |
| V·−V+1* | Увеличение Ү на 1, если есть | | | | |
| 11+1' | перенос из координаты Х | | | | |
| V ·− V ₋ 1* | Уменьшение Ү на 1, если есть заем | | | | |
| 11-1 | из координаты Х | | | | |

Формат микроопераций изменения кода адреса

Для формирования кодов данных реализованы следующие микрооперации:

- сохранение данных (T:=T);
- занесение исходных данных (T:=D);
- инверсия данных (T:=NOTT).

параллельно Для кодирования выполняемых микроопераций в команде выделены отдельные операционные поля. Число полей определяется количеством совместно выполняемых микроопераций. Для циклического повторения фрагментов тестов используются следующие признаки ветвления X≠NX, Y≠NY, X≠GX, Y≠GY программы: и комбинации данных условий: Х≠NX и Y≠NY, Х≠GX и Y≠GY.

Пример 1.

- MET JXYN, MET, 1 0 W.X·=X+1.Y·=Y+1*
- 0 W,X:=X+1,Y:=Y+1* 1 R,X:=X+2>X,Y:=Y+1*
- 3 A, X:=X+2, Y:=Y+1*

В данном примере микрооперации в тактах 0,1 и 3 будут выполняться до тех пор, пока выполняется условие X≠XN и Y≠YN. При невыполнении указанного выше условия микрооперации выполняются последний раз, и осуществляется переход к следующей команде.

Команды занесения данных устанавливают исходные состояния программных регистров, начальные и конечные адреса диагностируемой памяти коды регистров данных. Установка исходных значений регистров GX, GY определяет начальный адрес диагностируемого объема памяти. Запись данных в регистры NX, NY обеспечивает указание конечного адреса диагностируемого изделия.

Команды условных переходов выполняются совместно с микрооперациями сопутствующих им тактов. Сопутствующие такты определяются следующим образом: первый такт, следующий за командой условного перехода всегда сопутствующий; последующий такт считается сопутствующим, если его номер больше предыдущего (всего 4 такта, т.е. максимальное количество сопутствующих тактов равно четырем).

Переходы в этих командах осуществляются при выполнении соответствующих условий. Для поиска условий в команде может быть указан такт, по которому проводится эта проверка. При отсутствии номера такта интерпретатор выдает сообщение об ошибке. Таким образом, в программе необходимо предусмотреть переменную, указывающую номер такта проверки. При невыполнении условий перехода ни в одном такте интерпретатор переходит к рассмотрению команды или микроопераций, следующих за данной командой условного перехода. Если же условия выполнены, программа производит следующие действия: запоминает метку, на которую осуществить переход, необходимо выполняет микрооперации в сопутствующих команде тактах и выполняет переход на требуемую метку.

При работе интерпретатора в памяти компьютера выделяется участок объемом (NX+1)*(NY+1). В данный участок памяти может производиться запись различных кодов данных.

отлаживаемого теста данные По алгоритму считываются сравниваются с эталонными и значениями. При выполнении команды сравнения данных, считанных из выбранной ячейки, и эталонного значения счётчик К переключается в 1, если эти значения различны, т.е. при возникновении ошибки. Регистр ошибок фиксирует одиночные ошибки в разрядах данных, что позволяет по его значению в конце выполнения теста определить, было ли произведено хотя бы одно ошибочное считывание данных при диагностировании изделия.

Команды останова отображают конечный результат выполнения программы теста. Индикация при останове определяется по значению счетчика К – указателя ошибки: ENDP – останов с индикацией «ГОДЕН»; ENDE – останов с индикацией «БРАК».

Выводы

В предлагаемой структуре УТД частота формирования тестов определяется только временем переключения коммутаторов и выходных регистров и задержками времени распространения сигналов по линиям связи. Предложный принцип построения УТД полупроводниковой памяти может быть применен для проектирования комплексов тестового диагностирования цифровых систем, содержащих встроенную память. Формат команд языка Prover позволяет разрабатывать программы тестов для запоминающих устройств с различной структурой, что обеспечивает тестирование изделий полупроводниковой памяти с широким диапазоном основных параметров.

Поддержка

Работа выполнена при финансовой поддержке гранта РФФИ (проект 16-08-00393).

ЛИТЕРАТУРА

- [1] Lee, Seung-Joo, Dynamic Capabilities at Samsung Electronics: Analysis of its Growth Strategy in Semiconductors (August 22, 2011). Available at SSRN: https://ssrn.com/abstract=1914116
- [2] Main Memory: DDR4 & DDR5 SDRAM. [Electronic resource]. Access mode: https://www.jedec.org/category/ technology-focus-area/main-memory-ddr3-ddr4-sdram
- [3] Yarmolik S. V., Yarmolik V. N. Nondestructive RAM testing based on multiple signature comparison // Automatic Control and Computer Sciences, August 2009, Volume 43, Issue 4. – Pp. 173-178.
- [4] Muddapu Parvathi, N. Vasantha, Satya Parasad Modified March C - Algorithm for Embedded Memory Testing// International Journal of Electrical and Computer Engineering (IJECE) Vol. 2, № 5, October 2012. – Pp. 571-576.
- [5] CST Start to Deliver Eureka 2400 DDR4 Memory Tester/ [Electronic resource]. Access mode: http://www.simmtester.com/page/news/showcstnews.asp? num=120
- [6] Almadi M.K., Moamar D.N., Ryabtsev V.G. Methodology of Algorithms Synthesis of Memory Test Diagnosing // Proceedings of IEEE East-West Design & Test Symposium 2010 (EWDTS'10). St. Petersburg, 17-20 September 2010. – Pp. 366-370.
- [7] Almadi M.K., Ryabtsev V.G. New Infrastructure for Memory Tests Design // Proceedings of the International Workshop Critical Infrastructure Safety and Security (CrISS-DESSERT 2011). Kirovograd, May 11-13, 2011. – Pp. 434-440.
- [8] Ryabtsev V., Evseev K., Almadi M. The Concept of Memory Device Diagnosis Algorithm Design// Journal of Multidisciplinary Engineering Science and Technology (JMEST), Vol. 3, Issue 10, October. 2016. – Pp. 5771-5774.
- [9] Ryabtsev V.G., Kudlaenko V.M, Movchan Y.U. Method of estimation diagnostic properties of the Test Family March. // Proceeding of East-West & Test International Workshop (EWDTW'04).Yalta-Alushta, Criema, Ukraine. September 23-26, 2004. – Pp. 220-224.
- [10] Рябцев В.Г., Шубович А.А., Евсеев К.В. Автоматизированное проектирование дискретных тестов диагностирования запоминающих устройств / Современные наукоемкие технологии, № 6, 2016. – С. 288-294.

Principles of Designing Devices for Test Diagnosing of High-speed Microchips and Semiconductor Memory

A.P. Evdokimov, V.G. Ryabtsev, A.V. Melikov

Volgograd Agrarian State University, akim.onoke@mail.ru

Abstract — The article describes applying parallelization of the formation process of forcing vectors to improve the performance of test devices for diagnosing microcircuits and memory modules. Their sequential transfer to the inputs diagnosed device is by switches. In this article, it is parallel processes of reading responses.

The proposed method makes it possible to increase the frequency of the test actions formation and the processing of reading responses p-times, where p – is the coefficient of operation parallelization.

The structure of the multiprocessor diagnostic test device consists of the interacting control and p operation processors. Each operational processor generates the address code of the diagnosed memory cell Ai and the data codes Ti, which are used to specify the input data of the diagnostic object when the write operation is performing. The operational processors contain data generation units FTi and address generation units FAi that generate test impacts for several next times of diagnostic.

Transfer flags are formed because of comparing the codes of the current addresses in X and Y coordinates with NX, NY end address codes or GX, GY begin address codes. It does for implementing conditional transfer commands that provide access to all cells of the tested memory.

When the read operation is performed, data from the selected memory cell is recorded in the registers of reaction Dout0, Dout1,..., Doutp-1. Synchronizing signals provides resolution of data recording. They have spaced apart from each other for the period of the access time to the diagnosed product. The read and reference data are compared by comparison schemes S0, S1, ..., Sp-1, at the outputs of which the diagnostic result is generated. It can be masked by the code of the particular register B.

The project of a test diagnosis device has been developed in the Active-HDL. It consists of a microcontroller, an address counter of micro-operations, an address generation unit, a data generation unit, a data comparator, and a conditional transfer flags generation unit. The completed verification of the project confirmed its operability.

In the proposed structure of test diagnosis device, the test generation frequency is determined only by the switching time and the output registers and depends on the propagation time of the signals through the communication line. The proposed principle of constructing a test diagnosis device can be applied for designing of complexes for test diagnostic of digital systems containing built-in memory. Keywords — memory chip, test diagnostics, diagnostic test device

PROJECT SUPPORT

The research is powered by RFBR grant financial support (project 16-08-00393).

REFERENCES

- [1] Lee, Seung-Joo, Dynamic Capabilities at Samsung Electronics: Analysis of its Growth Strategy in Semiconductors (August 22, 2011). Available at SSRN: https://ssrn.com/abstract=1914116
- [2] Main Memory: DDR4 & DDR5 SDRAM. [Electronic resource]. Access mode: https://www.jedec.org/category/ technology-focus-area/main-memory-ddr3-ddr4-sdram
- [3] Yarmolik S. V., Yarmolik V. N. Nondestructive RAM testing based on multiple signature comparison // Automatic Control and Computer Sciences, August 2009, Volume 43, Issue 4. – Pp. 173-178.
- [4] Muddapu Parvathi, N. Vasantha, Satya Parasad Modified March C - Algorithm for Embedded Memory Testing // International Journal of Electrical and Computer Engineering (IJECE) Vol. 2, № 5, October 2012. – Pp. 571-576.
- [5] CST Start to Deliver Eureka 2400 DDR4 Memory Tester/ [Electronic resource]. Access mode: http://www.simmtester.com/page/news/showcstnews.asp? num=120 (December 28, 2017).
- [6] Almadi M.K., Moamar D.N., Ryabtsev V.G. Methodology of Algorithms Synthesis of Memory Test Diagnosing // Proceedings of IEEE East-West Design & Test Symposium 2010 (EWDTS'10). St. Petersburg, 17-20 September 2010. – Pp. 366-370.
- [7] Almadi M.K., Ryabtsev V.G. New Infrastructure for Memory Tests Design // Proceedings of the International Workshop Critical Infrastructure Safety and Security (CrISS-DESSERT 2011). Kirovograd, May 11-13, 2011. – Pp. 434-440.
- [8] Ryabtsev V., Evseev K., Almadi M. The Concept of Memory Device Diagnosis Algorithm Design // Journal of Multidisciplinary Engineering Science and Technology (JMEST), Vol. 3, Issue 10, October. 2016. – Pp. 5771-5774.
- [9] Ryabtsev V.G., Kudlaenko V.M, Movchan Y.U. Method of estimation diagnostic properties of the Test Family March. // Proceeding of East-West & Test International Workshop (EWDTW'04).Yalta-Alushta, Criema, Ukraine. September 23-26, 2004. – Pp. 220-224.
- [10] Rjabcev V.G., Shubovich A.A., Evseev K.V. Avtomatizirovannoe proektirovanie diskretnyh testov diagnostirovanija zapominajushhih ustrojstv / Sovremennye naukoemkie tehnologii, № 6, 2016. – C. 288-294.

Генератор тестов для проверки когерентности кэш-памятей многоядерных микропроцессоров (ristretto)

А.В. Смирнов, П.А. Чибисов

ФГУ ФНЦ НИИСИ РАН, г. Москва, chibisov@cs.niisi.ras.ru

Аннотация — Современные системы на кристалле содержат множество вычислительных ядер, каждое из которых имеет кэш-память нескольких уровней, а также системный контроллер с симметричным доступом к общей памяти ОЗУ. Функциональная верификация моделей микропроцессора, содержащих два и более ядер, представляет собой трудоемкий этап проектирования таких систем. Тестирование подсистемы памяти и контроллеров когерентности устройств, обеспечивающих согласованный обмен данными между ялрами. часто проволится помошью с автоматизированных генераторов тестов. В данной статье предложен метод автоматизированной генерации тестов, направленных на проверку когерентности кэшпамятей И подсистему памяти многоядерного микропроцессора, а также представлено описание созданного на основе этого метода генератора тестов.

Ключевые слова — многоядерный микропроцессор, псевдослучайные тесты, функциональная верификация, RTL-модель, ПЛИС-прототип, когерентность кэшпамяти, false sharing, самопроверка, скрытые ошибки.

I. Введение

В настоящее время широко распространены динамические методы верификации (так называемое имитационное тестирование) RTL-молелей микропроцессоров. Важную роль в традиционном маршруте верификации при тестировании моделей микропроцессора играет стохастическое тестирование. Существуют специальные генераторы комбинаторных псевдослучайных тестов, направленных на какой-либо микроархитектуры, аспект решающие залачу удовлетворения заданных пользователем ограничений. Однако такие генераторы являются трудоемкими в создании и настройке, а ошибки, которые находятся в результате созданных ими тестов - в большинстве случаев крайне специфичны из-за нацеленности таких генераторов тестов на очень узкий класс ситуаций.

Наращивание числа вычислительных ядер на одном кристалле приводит к повышению комбинаторной сложности подсистемы памяти. Требуется тем или иным способом доказывать корректность работы как отдельно вычислительного ядра разрабатываемого микропроцессора, его системного контроллера, блоков, обеспечивающих согласованное межъядерное взаимодействие, так и всех блоков в совокупности.

Предлагаемый в статье подход автоматизированной генерации тестов, направленных на проверку

когерентности кэш-памятей и подсистему памяти многоядерного микропроцессора, обеспечивает системное тестирование подсистемы памяти, и, в том числе, механизмов, отвечающих за когерентность кэшпамятей. Также описывается созданный на основе этого метода генератор тестов, который получил внутреннее название *ristretto* (из-за схожести данного слова с аббревиатурой, образованной от слов random instruction sequence, и словом threads). В качестве идей и прототипов при проектировании этого генератора тестов были рассмотрены методы и созданные на их основе генераторы тестов [1], [2]. В частности, был изучен опыт корпораций ARM [3] и IBM [4] в исследуемой области.

II. Описание генератора тестов

Генератор тестов *ristretto* ориентирован на создание определенных тестовых ситуаций, направленных на подсистему памяти (MMU, TLB, кэш-памяти всех уровней, механизмы поддержки когерентности данных, буферы предварительной подкачки данных), И вырабатывает тестовый случайный код для многоядерных микропроцессоров архитектуры КОМДИВ64 (подобна архитектуре MIPS64). Исходные коды генератора написаны на языке PERL. На вход генератор получает конфигурационный файл с настройками, на выходе генератора создается тест на языке ассемблер.

Для того чтобы обеспечить требуемые воздействия на подсистему памяти на системном уровне в рамках выбранного подхода, создаются тестовые ситуации, состоящие из комбинаций инструкций обращения к памяти (кэшируемые инструкции чтения из памяти и инструкции записи в память). При этом на каждом ядре запускается свой поток инструкций (термин поток здесь схож с термином «поток выполнения» в операционных системах). Потоки выполняются одновременно, каждое ядро обрабатывает отдельный поток, совместно или раздельно используя задаваемые конфигурационным файлом области памяти. Области памяти могут быть общими (совместными) или приватными (доступными ядру), также для только одному повышения реалистичности тестов можно включить R конфигурацию общую область, доступную только на чтение и общую область, доступную только на запись. Упрощенный пример конфигурации областей памяти для двухъядерного микропроцессора показан на рис. 1.

Направление стрелок на рисунке совпадает с направлением потоков данных.

Любая область, в которую разрешено писать более чем одному ядру, может содержать недетерминированные значения так как порядок операций записи от различных ядер невозможно проследить. Данные в области памяти «только на запись» не подлежат проверке. Для того, чтобы разрешить проблему проверки данных для других общих областей предлагается техника, описанная ниже.



Рис. 1. Пример конфигурации областей памяти на примере двухъядерного процессора

На каждом ядре микропроцессора выполняется тестовый код (или с программной точки зрения – поток исполнения, thread), созданный генератором из заданного набора инструкций. Большая часть инструкций из этого набора является кэшируемыми обращениями память. Для каждого В ядра микропроцессора выделяется несколько областей памяти, так называемая «глобальная» карта памяти совокупность областей памяти размером приблизительно от 3 до 5 кэш-линий каждая, их возможная конфигурация описана выше. Во время инициализации каждая область памяти заполняется случайными значениями «своим» (тем, которому она назначена) ядром. Максимальное и минимальное число областей также указывается в конфигурационном файле.

Каждое ядро микропроцессора во время процедуры инициализации заполняет эти области случайными значениями.

Тест состоит из заданного числа временны́х зон (интервалов) – независимых секций теста. Временная зона – это одна подпрограмма, содержащая одну итерацию теста (будем также называть ее «подтестом»). Каждая такая подпрограмма создается генератором тестов согласованно для каждого ядра.

Между временными зонами ядра микропроцессора (потоки выполнения – с точки зрения программиста) синхронизируются для того, чтобы каждая итерация теста начиналась одновременно.

Для каждой временной зоны (итерации теста) во время генерации теста выбирается набор областей памяти из «глобальной» карты памяти – в эти области генератор «направляет» инструкции обращения в память. Назовём их «локальными» картами памяти. Пример распределения памяти приведён на рис. 2.



Рис. 2. Пример разбиения общих областей по признаку принадлежности ядрам

Отдельные области (выделенные серым и белым цветами) можно называть «элементами памяти».

Идея таких карт памяти заключается в конфигурации областей для ложного разделения данных (false sharing) В памяти между вычислительными ядрами - явление, возникающее, когда запущенные на двух разных ядрах процессора потоки обращаются к элементам данных, которые находятся в одной строке кэш-памяти. Каждое такое обращение требует обновления кэш-памятей обоих Если обновляемый элемент, к которому ядер. происходит обращение, используется только одним ядром процессора, то всем остальным ядрам всё равно приходится обновлять свои кэш-линии, несмотря на то, что им не нужно знать об этом изменении данных (отсюда термин «ложное разделение данных»).

Как уже было сказано, каждая область памяти разбивается на элементы памяти. Размер элемента памяти выбирается случайно так, чтобы каждая область памяти имела не менее двух элементов. Каждому элементу памяти ставится в соответствие набор инструкций чтения или записи максимально возможного размера. Также для выбранных инструкций генерируются «аналоги» меньшего размера (1 store double = 2 store word = 4 store half = 8 store byte; 1 load double = 2 load word = 4 load half = 8 load byte). Таким образом, генератор полностью заполняет каждую

область памяти различными операциями обращения в память.

При генерации подтеста (временной зоны) случайно выбираются две области памяти (области памяти могут совпадать). В процессе генерации инструкции могут быть расположены различным образом: инкремент смещения относительно начального адреса, декремент смещения, случайно перемешанные смещения. При выборе генератором одинаковых начальных адресов для двух (нескольких) ядер микропроцессора получим ситуации «false sharing», так как эти адреса будут с некоторой вероятностью находиться внутри одной кэшкаждой области генератор линии. Для залает виртуальные адреса И соответствующие им назначаемые отображения виртуальных адресов в физические, инициализирует регистры, таблицы значений в памяти, а также выдает значения для самопроверки.

Пример архитектуры теста с несколькими временными зонами в двухъядерной конфигурации с детальным отображением областей памяти показан на рис. 3. Во «Временной зоне 0» каждое ядро независимо от других выполняет инициализацию «своих» областей памяти случайными значениями, во «Временной зоне 1» оба ядра взаимодействуют и производят обращения на запись и чтение к одной области памяти (однако в разные байты – согласно описанной выше случайно созданной для этого подтеста «локальной» карте памяти).



Рис. 3. Архитектура теста в двухъядерной конфигурации с детальным отображением областей памяти

Благодаря такому разделению данных внутри любой отдельно взятой кэш-линии, каждая общая область будет содержать детерминированные значения и, таким образом, для всех данных из этой области правильность их значения может быть подтверждена в конце зоны. Во «Временной зоне 2» представлен совместной одного пример работы потока с работой одновременно стресс-функции. Bo «Временной зоне 3» каждое ядро работает независимо со своей областью, однако при этом естественным образом учитывается вся предыстория и достигнутое к моменту начала зоны состояние кэш-памятей; данный фрагмент кода теста будет выполнен дважды. В конце каждой зоны и в конце каждого теста осуществляется самопроверка различной степени детализации (настраивается пользователем).

В качестве входных данных генератор получает текстовый файл со следующими параметрами:

- число ядер процессора в тестируемой системе;
- максимальное и минимальное количество областей памяти для каждого ядра;
- размер кэш-памятей всех уровней, а также их ассоциативность;
- размер одной строки кэш-памяти;

- вероятность кратности начального адреса генерируемой области размеру кэш-памятей L1, L2 или размеру одной их секции;
- количество подтестов (временных зон);
- максимальный и минимальный размер случайно генерируемой области;
- вероятности случайного и упорядоченного распределений смещений в подтесте;
- вероятность повторения подтеста после завершения его выполнения;
- вероятность появления ситуаций «false sharing»;
- вероятность сдвига случайных начальных адресов на половину размера области влево;
- максимальное и минимальное количество инструкций в подтесте;
- частота появления стресс-функций в тестах;
- количество и детализация проверок.

Тест генерируется на основе входных данных, получаемых из файла с настройками и шаблона теста. Назначение шаблона в данном генераторе отличается от назначения шаблона в традиционном генераторе псевдослучайных тестов. Здесь шаблон не является в полной мере заданием на генерацию тестового кода и не включает в себя программу построения теста на специальном языке (псевдокоде). Он лишь задает веса или отдельных отдельных конструкций групп инструкций, отношения полностью случайных фрагментов кода к детерминированным макросам (пользовательским функциям или стресс-функциям, которые будут описаны ниже), задает частоту проверок и другие параметры: режимы, распределение памяти, и другие, необходимые для построения тестового кода Также в шаблоне задаются образцы данные. определенных фрагментов стресс-функций и их допустимые степени рандомизации.

III. Описание тестовых ситуаций

Генератор тестов *ristretto* вырабатывает случайные тестовые последовательности инструкций обращения к памяти для многоядерных микропроцессоров, их RTL-моделей и ПЛИС-прототипов.

Для каждого ядра микропроцессора создается свой синхронизация поток инструкций, потоков обеспечивается макросами синхронизации. Каждый подтест (или иногда и весь тест) может быть выполнен повторно с заданной вероятностью. Это помогает повысить эффективность тестирования механизмов межъядерных взаимодействий (временные взаимоотношения между ядрами и кэш-памятью ядер) за счёт большего разнообразия тестовых ситуаций так как при первом проходе теста большинство обращений в кэш-память имеют большую долю промахов, тогда как при последующих проходах теста – попаданий.

Так как основным назначением генератора является проверка взаимодействия компонентов подсистемы памяти, к генерируемому коду предъявляются следующие требования.

Для того чтобы достичь требуемого уровня функционального покрытия, генератор предпринимает попытку задействовать все возможные сценарии, по которым может происходить взаимодействие между потоками с точки зрения зависимостей по данным для таких разделяемых ресурсов как строки кэш-памяти и страницы физического адресного сегмента памяти.

Во время выполнения одного подтеста благодаря случайному выбору начальных адресов возможны следующие варианты зависимостей по разделяемой памяти:

 раздельное использование памяти. Несвязанные области памяти используются в подтесте, при этом все псевдослучайно создаваемые обращения к памяти рассматриваются как нагрузка на всю систему в целом;
 ложное разделение данных (false sharing). Случайно выбираемые ядра микропроцессора (число ядер больше или равно двум) взаимодействует через обращения к нескольким кэш-линиям. При этом обращения производятся в различные, несовпадающие байты для того, чтобы сохранить детерминированность значений;

3) истинное разделение данных (true sharing). Ядра процессора обращаются по одним и тем же адресам памяти, при этом порядок изменения данных контролируется посредством механизмов синхронизации, то есть записи в память происходят не одновременно, а в различных временных зонах, между которыми ядра синхронизируются;

4) недетерминированное истинное разделение данных. Порядок операций записи в память от различных ядер в этом случае невозможно проследить (это не является необходимым), так как данные в области памяти «только на запись» не подлежат проверке.

С точки зрения одного ядра генератором автоматически создаются тестовые ситуации, в которых можно выделить зависимости инструкций процессора по обращениям в память (зависимость по адресам) или по конвейеру ядра (зависимость по регистрам): RAR (read after read), RAW (read after write), WAR (write after read), WAW (write after write). Также создается нагрузка на блок преобразования адресов (TLB), на различные устройства буферизации данных на чтение из памяти и запись в память, такие как буфер опережающей предвыборки данных из памяти и буфер перестановки и слияния потоков данных, а также на все уровни иерархии кэш-памяти и механизмы возникновения исключительных ситуаций.

Для того чтобы каждая итерация теста начиналась одновременно требуется синхронизация потоков. Процедура синхронизации может быть организована любым доступным способом, программным или аппаратным. Среди методов синхронизации потоков, применяемых в многоядерных системах, известны такие методы как межпроцессорные прерывания и передача сообщений через специальные регистры межпроцессорного обмена (mailboxes). Также может быть предусмотрен механизм передачи управления между ядрами, реализуемый через атомарные операции типа «чтение-модификация-запись». Генератор тестов ristretto поддерживает несколько механизмов синхронизации, которые с программной точки зрения представляют собой набор библиотечных примитивов. В частности, в библиотеку входит функция для организации критических секций в коде теста, необходимых для разделения во времени последовательностей обращения разных потоков к общим ресурсам, например, области памяти с истинным разделением данных.

IV. Стресс-функции

В работе [4] было введено понятие стресс-функции (thread irritator) для тестирования одновременной процессорах IBM. многопоточности в Были три методики усовершенствования предложены генерации тестовых программ: стрессовая функция (поток), слияние потоков и репликация потоков. Этот подход был адаптирован и успешно применен в рассматриваемом в данной статье генераторе тестов. Он позволяет достичь некоторых редких тестовых ситуаций, которые могут привести к нахождению ошибок на предсказывающих интеллектуальных буферах данных и механизмах перестановки и склейки обратных записей в память. Так как случайно задаваемых инструкций недостаточно, необходимо времени вместо случайного потока время от инструкций обращения к памяти вставлять в тест специальные стресс-функции.

Идея подхода, связанного с введением понятия стресс-функцией, заключается в следующем. Предположим, что существует некоторый поток (первичный, инструкций обращения к памяти основной), запущенный на первом ядре микропроцессора. На одном или более из числа оставшихся ядер вместо других таких же потоков заданное число стрессовых функций. запускается Стресс-функции представляют собой повторяющиеся однотипные обращения в память, целью которых является нагрузка контроллера памяти. Стресс-функция может являться коротким циклом или последовательностью одинаковых инструкций обращения памяти. Стрессовая функция к взаимодействует с основным потоком для того, чтобы повысить степень взаимодействий между ядрами на микроархитектурном уровне. Результатом применения подхода является повышение вероятности нахождения ошибок в RTL-модели или ПЛИС-прототипе проектируемого микропроцессора, таких как зависание, динамическая и другие виды взаимоблокировок, логические ошибки в блоках подсистемы памяти, а также ошибки, связанные с неправильным или несвоевременным обновлением состояния кэш-памяти.

Также в [4] предложен метод слияния потоков, который заключается в том, что генератор тестовых программ создает фрагменты однопоточных тестов для каждого ядра и затем соединяет их в многопоточный тест. Процесс построения тестового фрагмента должен гарантировать, что однопоточный тест не изменяет общие области памяти при его распараллеливании на несколько потоков. Затем многопоточные тесты многократно создаются путем произвольной выборки однопоточных фрагментов из уже готового набора и их последующего слияния. В этом подходе в результате слияния потоков инструкции обращения к памяти из каждого однопоточного фрагмента теста могут быть выполнены множество раз, каждый раз в сочетании с различными тестовыми воздействиями из других потоков. Случайные комбинации однопоточных тестов воспроизводят уникальные ситуации, значительно изменяя способ тестирования модели, а также вероятность увеличивают нахожления труднообнаруживаемых микроархитектурных ошибок.

В методе репликации потоков генератор тестов создает однопоточные тесты, а затем воспроизводит множество его копий на несколько потоков, моделируя многопоточный сценарий. Пользователь настраивает генератор таким образом, чтобы тест можно было запустить в нескольких потоках. Это достигается за счет того, что никакие две инструкции обращения к памяти не могут иметь доступ к одной и той же ячейке памяти. В таком случае при репликации потоков можно получить детерминированный результат во время выполнения, совпадающий с предсказанным значением, полученным во время генерации теста.

В результате репликации потоков могут быть созданы интересные многопоточные сценарии благодаря тому, что в двух или более потоках выполняются одинаковые последовательности инструкций и задействуются при этом одинаковые ресурсы.

В генераторе тестов **ristretto** предполагается автоматизированное создание стресс-функций, однако требуется дополнительное исследование эффективности и целесообразности применения методов слияния и репликации потоков.

V. МЕТОДЫ ПРОВЕРКИ

Существует два варианта реализации описываемого генератора тестов. Первый вариант разрабатывался для автоматизированного построения тестов, нацеленных на проверку подсистемы памяти и механизмов обеспечения когерентности данных в кэш-памятях при отладке RTL-модели проектируемого многоядерного процессора. Известный способ проверки корректности работы RTL-моделей – это совместное моделирование (симуляция) работы теста на подлежащей тестированию модели (RTL-модель) и на эталонном эмуляторе (instruction set simulator, ISS). разрабатываемом на ином уровне абстракции. После прохождения теста на обеих моделях результаты сравниваются. При нахождении несоответствия

определяется блок, некорректное поведение которого привело к ошибке. Преимуществом этого подхода является то, что он позволяет выявить ошибку непосредственно в том месте, где она возникла с точностью до инструкции.

Второй вариант реализации генератора тестов создает тесты со встроенными самопроверками для последующего запуска ПЛИС-прототипе на разрабатываемого микропроцессора (post-silicon validation stage). В этом варианте нет необходимости в применении внешнего (по отношению к среде генерации) эталонного эмулятора. При этом время выполнения одного теста сокращается на несколько порядков в сравнении со временем моделирования на RTL [5]. Однако для получения внутреннего состояния микропроцессора (это верно и для ПЛИС-прототипа) требуется специальная инфраструктура, кроме того доступ к внутренним сигналам ограничен, из-за чего диагностика первопричины ошибки чрезвычайно затруднена [6]. Это происходит из-за того, что эффект от произошедшей в аппаратуре ошибки зачастую проявляется только через миллионы тактов после ее возникновения, и инженерам требуется значительное время на воспроизведение ошибки. Поэтому основным вопросом при применении такого варианта реализации генератора является выбор критериев корректности прохождения теста, то есть вопрос выбора метода, по которому будет осуществляться проверка результата работы теста [7], [8].

Существует методика, которая называется «многопроходной проверкой непротиворечивости» или иначе «многопроходной проверкой совпадения результатов со значениями, принимаемыми за эталонные» (multi pass consistency check) [7]. Согласно этой методике каждый тест выполняется несколько раз. Результаты первого прохода теста считаются образцовыми. После каждого последующего прохода проверяется, что определенные ресурсы процессора, включая регистры общего назначения и определенные области памяти, совпадают с образцовыми значениями. Непротиворечивость гарантируется соблюдением ограничений при построении тестов:

1) в поток псевдослучайных инструкций не включаются инструкции, результат выполнения которых не предсказуем полностью;

 любые конфликты вида «запись-запись» в известных областях памяти запрещаются, так как невозможно предсказать порядок выполнения конфликтующих записей;

3) любые данные, записываемые в разрешенный локальный ресурс, должны быть иметь одни и те же значения в каждом проходе.

Несмотря на эти ограничения, целесообразно генерировать не подлежащие проверке конфликтные события, так как они повышают степень нагрузки на подсистему памяти и помогают обнаружить ошибки, которые не находятся другими способами [5]. Также в работе [7] предлагается вводить некоторые ограниченные изменения в код во время повторных проходов теста. Рекомендуется также на втором проходе вносить в исполняемый код несколько не влияющих на результат выполнения теста инструкций, либо менять приоритеты потоков, для того, чтобы на повторных проходах теста его выполнение осуществилось с измененными зависимостями по ресурсам и с иным состоянием конвейера.

Также была рассмотрена методика создания самопроверяющихся тестов Reversi [8]. Генератор Reversi создает псевдослучайные программы таким образом, чтобы их корректное конечное состояние было время генерации, необходимость известно BO архитектурного моделирования при этом отсутствует. Ключевое наблюдение, которое легло в основу разработки Reversi, состоит в том, что у многих инструкций в наборе инструкций (ISA) процессора есть противоположные «аналоги», то есть такие инструкции, у функциональности которых есть соответствующая обратная инструкция, например, сложение/вычитание целых чисел, загрузка/сохранение в/из памяти. Соответственно, ошибки могут быть обнаружены при расхождении начального и конечного состояния микропроцессора после прохождения прямого и обратного тестового фрагментов. В другой работе [9] предлагается просто повторять каждую инструкцию два раза и контролировать корректность выполнения теста, сравнивая результаты их выполнения.

Ещё один способ обеспечения самопроверки в тестах – это создание эквивалентных тестов с результатов последующей сверкой выполнения каждого из тестов (методика «ISA diversity») [10]. Под заменой инструкции на эквивалентный аналог понимается подбор последовательности инструкций, результат выполнения которой будет полностью совпадать с результатом выполнения оригинальной инструкции. Самая трудоемкая часть в данном подходе тестирования заключается в проведении анализа набора инструкций и составление таблицы преобразований инструкций, которые могут быть заменены на полностью эквивалентную последовательность. Все инструкции можно разделить на три категории:

1) полностью эквивалентные: инструкции имеют один или несколько способов замены на эквивалентную последовательность инструкций. Эта категория включает большинство арифметических и логических инструкций, операции работы с памятью и большинство операций ветвления.

2) Частично эквивалентные: эта группа включает в себя инструкции, при замене которых теряется точность вычислений (операции с плавающей запятой).

 Эквивалентная инструкция отсутствует: группа инструкций, которые нельзя заменить на аналогичные.
 Эта категория включает привилегированные инструкции доступа к системным ресурсам и другие.

Согласно исследованию [10], 74 – 79% всех команд для различных современных архитектур можно представить в виде последовательности эквивалентных инструкций.

VI. ОЦЕНКА ВЕРОЯТНОСТИ СКРЫТЫХ ОШИБОК

Описанные выше подходы к построению самопроверяющихся тестов не позволяют точно указать реальное место первого появления ошибки. Кроме того, рассмотренные подходы ограничены невозможностью наблюдения внутренних сигналов, что влияет на их способность обнаруживать и диагностировать ошибки: ошибка, проявляющаяся при запуске случайных тестов. может быть скрыта и не обнаружена при завершении теста. Ошибка перестаёт быть видна после выполнения некоторого количества инструкций, которые ее замаскируют (пример маски́рования ошибки показан на рисунке 4).

LW R2,0x45(R3) >>> R2=0x12E5 /* ЗАГРУЗКА ОШИБОЧНОГО ЗНАЧЕНИЯ! */ LB R2,0x3F(R5) >>> R2=0xB7 /* МАСКИРОВАНИЕ ОШИБКИ */ СРАВНЕНИЕ РЕГИСТРОВ, В ТОМ ЧИСЛЕ R2: (R2 = = 0xB7) ? ДА – ОШИБКИ НЕТ.

Рис. 4. Пример маскирования ошибки при самопроверке

Оценка вероятности пропуска ошибок в тестах позволяет выбирать эффективные тесты для регрессии, а также экспериментально подобрать сбалансированное количество проверок на один тест таким образом, чтобы не упростить тестовые ситуации и сохранить вероятность нахождения трудно обнаруживаемых и редко проявляющихся ошибок.

В работе [11] предлагается программный инструмент, который позволяет проводить анализ числа потенциально пропущенных ошибок. С его помощью можно отбирать тесты для базы регрессии, код которых минимально маски́рует ошибки. Также предлагается подход, связанный с внесением изменений в тест таким образом, чтобы ошибка как можно дольше могла оставаться в незамаскированном виде и, следовательно, могла быть обнаружена при выполнении процедуры самопроверки.

Рассмотренный подход к оцениванию вероятности маскирования ошибок остается актуальным в случаях, когда самопроверка осуществляется за счет сравнения получаемых в тесте результатов с подгружаемыми данными от внешнего образцового эмулятора либо от встроенного интерпретатора инструкций. Исходный код генератора со встроенным интерпретатором должен быть разработан на языке С++ для обеспечения возможности запуска программы на целевой платформе (ПЛИС-прототипе).

VII. ЗАКЛЮЧЕНИЕ

Тестирование подсистемы памяти и входящих в нее блоков обеспечения согласованного обмена данными между ядрами часто проводится с помощью автоматизированных псевдослучайных генераторов тестов. В статье предложен метод автоматизированного построения тестов, направленных на проверку когерентности кэш-памятей и подсистемы памяти многоядерного микропроцессора, а также представлено описание созданного на основе этого метода генератора тестов.

Для повышения функционального покрытия генератор задействует все возможные сценарии, по которым может происходить взаимодействие между ядрами с точки зрения зависимостей по данным.

Также в статье обсуждаются подходы к выбору критериев корректности прохождения теста в случае работы на ПЛИС-прототипе, так как требуется определить способы самопроверки результата прохождения теста. Кроме того, поднимается вопрос, связанный с количественной оценкой вероятности маскирования ошибок.

ЛИТЕРАТУРА

- Hudson J., Kurucheti G. A Configurable Random Instruction Sequence (RIS) Tool for Memory Coherence in Multiprocessor Systems. Workshop on Microprocessor Test and Verification, 2014, pp. 98-101. DOI: 10.1109/MTV.2014.26
- [2] Venkatesan D., Nagarajan P. A Case Study of Multiprocessor Bugs Found Using RIS Generators and Memory Usage Techniques. Workshop on Microprocessor Test and Verification, 2014, pp. 4-9. DOI: 10.1109/MTV.2014.28
- [3] S. Thiruvathodi and D. Yeggina, "A Random Instruction Sequence Generator for ARM Based Systems," 2014 15th International Microprocessor Test and Verification Workshop (MTV), Austin, TX, USA, 2014, pp. 73-77. doi:10.1109/MTV.2014.20
- [4] Ludden J.M., Rimon M., Hickerson B.G., Adir A. (2011) Advances in Simultaneous Multithreading Testcase Generation Methods. In: Barner S., Harris I., Kroening D., Raz O. (eds) Hardware and Software: Verification and Testing. HVC 2010. Lecture Notes in Computer Science, vol 6504. Springer, Berlin, Heidelberg
- [5] Wisam, K., et al., "Improving Post-Silicon Validation Efficiency by Using Pre-Generated Data," Proc. Intl. Haifa Verification Conf., pp. 166-181, 2013
- [6] Satish Kumar Sadasivam, Sangram Alapati, Varun Mallikarjunan: Test Generation Approach for Post-Silicon Validation of High End Microprocessor. DSD 2012: 830-836
- [7] Allon Adir, Amir Nahir, Avi Ziv: Concurrent Generation of Concurrent Programs for Post-Silicon Validation. IEEE Trans. on CAD of Integrated Circuits and Systems 31(8): 1297-1302 (2012)
- [8] Ilya Wagner, Valeria Bertacco Post-Silicon and Runtime Verification for Modern Processors, Springer, 2011, 224 p
- [9] Lin, David ; Singh, Eshan ; Barrett, Clark ; Mitra, Subhasish. / A structured approach to post-silicon validation and debug using symbolic quick error detection. International Test Conference 2015, ITC 2015 - Proceedings. Vol. 2015-November Institute of Electrical and Electronics Engineers Inc., 2015
- [10] Nikos Foutris, Dimitris Gizopoulos, Mihalis Psarakis, Xavier Vera, and Antonio Gonzalez. 2011. Accelerating

microprocessor silicon validation by exposing ISA diversity. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44). ACM, New York, NY, USA, 386-397 Probabilistic bug-masking analysis for post-silicon tests in microprocessor verification. DAC 2016: 24:1-24:6

[11] Doowon Lee, Tom Kolan, Arkadiy Morgenshtein, Vitali Sokhin, Ronny Morad, Avi Ziv, Valeria Bertacco:

Random Test Generator for Multicore Microprocessor Cache Coherence Verification (Ristretto)

A.V. Smirnov, P.A. Chibisov

Scientific Research Institute of System Analysis (SRISA RAS), chibisov@cs.niisi.ras.ru

Abstract — modern system-on-chip designs contain multiple computational cores with several levels of caches, as well as a sophisticated memory subsystem. Functional verification of multi-core microprocessor models is known to be a big challenge. There are different approaches for memory subsystem and cache coherence controller verification, but an automated functional test generation strategy is the most commonly used in the industry.

In this paper, the technique of automated multi-core test generation is proposed. It can be applied for cache coherence and memory subsystem check in a top-level multi-core RTLmodel simulation. Moreover, the presented test generator can be very effective in generating test scenarios for FPGAprototypes of SoC being designed. In this paper we also give a detailed description of the random test generator itself and capabilities of generated test cases.

The proposed test generator got its name "ristretto" due to the similarity of the word "ristretto" with the abbreviation formed from the words "random instruction sequence" (RIS), and the word "threads" (and because ristretto is so concentrated and intense).

Some self-checking validation approaches are suggested to obtain correct responses in FPGA-based verification (postsilicon validation). In the paper we also discuss bug-masking problem in post-silicon random instruction tests that arises due to limited observability.

Keywords — multicore microprocessor, pseudorandom tests generation, functional verification, RTL-model, cache coherence, false sharing, memory subsystem, post-silicon validation, self-checking, bug masking.

REFERENCES

- Hudson J., Kurucheti G. A Configurable Random Instruction Sequence (RIS) Tool for Memory Coherence in Multiprocessor Systems. Workshop on Microprocessor Test and Verification, 2014, pp. 98-101. DOI: 10.1109/MTV.2014.26
- [2] Venkatesan D., Nagarajan P. A Case Study of Multiprocessor Bugs Found Using RIS Generators and Memory Usage

Techniques. Workshop on Microprocessor Test and Verification, 2014, pp. 4-9. DOI: 10.1109/MTV.2014.28

- [3] S. Thiruvathodi and D. Yeggina, "A Random Instruction Sequence Generator for ARM Based Systems," 2014 15th International Microprocessor Test and Verification Workshop (MTV), Austin, TX, USA, 2014, pp. 73-77. doi:10.1109/MTV.2014.20
- [4] Ludden J.M., Rimon M., Hickerson B.G., Adir A. (2011) Advances in Simultaneous Multithreading Testcase Generation Methods. In: Barner S., Harris I., Kroening D., Raz O. (eds) Hardware and Software: Verification and Testing. HVC 2010. Lecture Notes in Computer Science, vol 6504. Springer, Berlin, Heidelberg
- [5] Wisam, K., et al., "Improving Post-Silicon Validation Efficiency by Using Pre-Generated Data," Proc. Intl. Haifa Verification Conf., pp. 166-181, 2013
- [6] Satish Kumar Sadasivam, Sangram Alapati, Varun Mallikarjunan: Test Generation Approach for Post-Silicon Validation of High End Microprocessor. DSD 2012: 830-836
- [7] Allon Adir, Amir Nahir, Avi Živ: Concurrent Generation of Concurrent Programs for Post-Silicon Validation. IEEE Trans. on CAD of Integrated Circuits and Systems 31(8): 1297-1302 (2012)
- [8] Ilya Wagner, Valeria Bertacco Post-Silicon and Runtime Verification for Modern Processors, Springer, 2011, 224 p
- [9] Lin, David ; Singh, Eshan ; Barrett, Clark ; Mitra, Subhasish. / A structured approach to post-silicon validation and debug using symbolic quick error detection. International Test Conference 2015, ITC 2015 - Proceedings. Vol. 2015-November Institute of Electrical and Electronics Engineers Inc., 2015
- [10] Nikos Foutris, Dimitris Gizopoulos, Mihalis Psarakis, Xavier Vera, and Antonio Gonzalez. 2011. Accelerating microprocessor silicon validation by exposing ISA diversity. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44). ACM, New York, NY, USA, 386-397
- [11] Doowon Lee, Tom Kolan, Arkadiy Morgenshtein, Vitali Sokhin, Ronny Morad, Avi Ziv, Valeria Bertacco: Probabilistic bug-masking analysis for post-silicon tests in microprocessor verification. DAC 2016: 24:1-24:6

Виртуальные испытания микро- и наноэлектронных систем на внешние воздействия

А.С. Шалумов, Д.Н. Травкин, М.В. Тихомиров

Научно-исследовательский институт «АСОНИКА», г. Ковров, als@asonika-online.ru

Аннотация — В статье рассмотрено назначение виртуальных испытаний микро- и наноэлектронных систем и их оптимальное сочетание с натурными испытаниями. Представлены возможности Автоматизированной системы АСОНИКА, на базе которой проводятся виртуальные испытания на механические (вибрации, удары, линейные ускорения, акустические шумы), тепловые, электромагнитные воздействия.

Ключевые слова — виртуальные испытания, моделирование, ускорения, напряжения, температуры, усталостные разрушения.

Работу микро- и наноэлектронных систем (МЭС) значительно ухудшает воздействие вибраций, ударов, тепла, электромагнитных полей, радиации и т.д. Поэтому важным этапом создания МЭС являются её испытания на все эти воздействия.

В России существуют испытательные центры, позволяющие проводить подобные натурные испытания МЭС. Оптимальное сочетание натурных испытаний с виртуальными позволит повысить эффективность проектирования МЭС:

 обеспечить успешность прохождения натурных испытаний опытных образцов МЭС;

- сократить количество итераций по доработке МЭС по результатам натурных испытаний;

- обеспечить значительную экономию денежных средств и сокращение сроков создания МЭС при одновременном повышении качества и надежности за счет сокращения количества испытаний.

Назначение виртуальных испытаний:

- определить тепловые, механические и другие характеристики МЭС при внешних воздействующих факторах (ВВФ) на ранних этапах проектирования МЭС, когда еще не создан опытный образец МЭС, и обеспечить стойкость МЭС к ВВФ;

- добившись адекватности виртуальных и натурных испытаний путём идентификации параметров моделей МЭС, проверить работоспособность МЭС в критических режимах в условиях ВВФ.

Назначение натурных испытаний:

- провести анализ стойкости опытных образцов МЭС к ВВФ;

- получить для МЭС допустимые значения ускорений, температур и других характеристик;

- провести идентификацию параметров моделей МЭС, используемых при виртуальных испытаниях.

В 2018 году впервые в России создан Центр виртуальных испытаний МЭС «АСОНИКА» (г. Владимир), который базируется на российской Автоматизированной системе обеспечения надежности и качества аппаратуры (АСОНИКА), разработанной научным коллективом ООО «НИИ «АСОНИКА» [1] -[3]. Система АСОНИКА уже более 30-и лет применяется на многих российских предприятиях, прежде всего оборонной, космической и авиационной отраслей. Система аттестована Министерством обороны РФ, выпущены Руководящие документы военные. В рамках системы АСОНИКА создана связанная с моделирующими подсистемами база электрорадиоизделий и материалов по данных геометрическим, физико-механическим, усталостным, теплофизическим, электрическим, электромагнитным и надежностным параметрам.

Вышла в свет новая книга по системе АСОНИКА: «Опыт применения автоматизированной системы АСОНИКА в промышленности Российской Федерации» [4]. B монографии рассмотрено множество примеров, полученных в результате многолетнего (27 лет) применения системы АСОНИКА. Книгу можно скачать с сайта www.asonika-online.ru в разделе «Книги» по ссылке: http://asonika-online.ru/books/.

Технология на основе системы АСОНИКА единственная в России, позволяющая осуществить сквозное проектирование высоконадежных МЭС космических, авиационных и других подвижных объектов с учетом внешних тепловых, механических, электромагнитных воздействий от технического задания и до изготовления опытного образца. Созданная электронная модель впервые позволит реализовать CALS-технологии в электронике на всех 11 стадиях жизненного цикла от маркетинговых исследований и до утилизации. Автоматизированная АСОНИКА не имеет аналогов система или сопоставимых прототипов в области моделирования высоконадежной электроники как в России, так и за рубежом.

МЭС-2018. Россия, Москва, октябрь 2018. © ИППМ РАН

Преимущества системы АСОНИКА:

1. Моделирование тепловых и механических, в том числе усталостных, процессов в интегральных микросхемах, в том числе на наноуровне.



Рис. 1. Конечно-элементная сетка для интегральных микросхем





Рис. 2. Температурные зависимости для интегральных микросхем





Рис. 3. Механические напряжения в интегральных микросхемах



Рис. 4. Усталостные разрушения в интегральных микросхемах







Рис. 5. Моделирование интегральных микросхем на наноуровне (25 нм)

2. В системе АСОНИКА созданы простые и интуитивно понятные интеллектуальные графические интерфейсы, состыкованные с базой данных материалов и электронных компонентов, содержащей геометрические, физико-механические, теплофизические и другие параметры, а также

допустимые значения характеристик, необходимые для принятия решения. При этом печатные узлы автоматически конвертируются из известных САПР: Mentor Graphics, Altium Designere, OrCAD и других - в формате IDF. Исключаются ошибки человеческого фактора при задании исходных данных. В отличие от систем ANSYS, NASTRAN, COSMOS, COMSOL и др., которые ничего этого не имеют, система ACOHИКА специализирована в области электроники и является инструментом разработчика электроники.



Рис. 6. Конвертирование печатных узлов из известных САПР: Mentor Graphics, Altium Designere, OrCAD и в других - в формате IDF (АСОНИКА-ТМ)



Рис. 7. База данных материалов и электронных компонентов (АСОНИКА-БД)



Рис. 8. Автоматическое конвертирование 3D-моделей произвольных конструкций электроники из известных САПР: ProEngineer, SolidWorks, Inventor и других - в форматах IGES и SAT, в том числе автоматические разбиение и построение сетки, а также склеивание моделей в местах стыковки деталей с разными шагами сетки (АСОНИКА-М-3D)



Рис. 9. Графический интерфейс для ввода типовых констукций шкафов и блоков электроники, в том числе на виброизоляторах, включающий автоматические разбиение и построение сетки, а также склеивание моделей в местах стыковки деталей с разными шагами сетки (АСОНИКА-М-ШКАФ, АСОНИКА-М, АСОНИКА-В)

3. Учтены особенности свойств материалов, применяемых в электронике, например, их нелинейные свойства, не свойственные другим изделиям, например, машиностроительным, для моделирования которых используются широко известные системы ANSYS, NASTRAN, COSMOS, COMSOL и др., не специфику электроники. учитывающие Данная специфика выражается, например, в нелинейности демпфирующих свойств современных материалов. В АСОНИКА системе заложена зависимость демпфирования от механических напряжений, чего нет в известных системах. В связи с этим в известных системах невозможно точно определить ускорения электронных компонентов, особенно на резонансах. А для электронных компонентов это особенно важно, так как для них задаются допустимые ускорения при всех механических воздействиях, которые ни в коем случае нельзя превышать.

| CAN'T INDERING FOR THE PARTY AND A | чализ Прилскения Помощь | |
|--|--|---------------------|
| WHERE ER . | - 8 - 1 - 2 | |
| | 2- h, h, 1- h X | |
| anticent Construction of pages | Xepartequences (Nanonecopyce) (Washeemon (P)) | Nacros Fal |
| Description | B December 4 19 4 | |
| R dil Case | I CHEADE SAMPLE (20 DELETICODES) 10 DELETICODES IN PE | and [2] sociated] |
| the section of the se | Historier rogaletta | Distance reportings |
| t de Centres conces | Digerargia crea | |
| (6-R) 3 renzologiani caperier (27) | Tonare crox. (-e- | 250 |
| (i) Taxana reserve unter | Manaphan cited | CP-1-25 |
| B D Farman | Каллество вырязов | 0 |
| - TIDA CIDAN | | |
| the Registration of the second | Dig averga i vanga ave | |
| 20 C Tennine ryonness groups | Demons, John 3 | 19600 |
| a-E forme3PM | Kanopauser technologiacore (Rr.675-4 | 0.20 |
| 8 D Kpenmeen | Manager receive each. (Darbor K) | 423.00 |
| В Контральные тован | Condensations' universa (cons est) | 100 |
| Checkpoint Checkpoint | Marcan appropriate app// [TIal | 16.98 |
| з 🖸 Воздействик | Name and and The | 14.48 |
| -// Гармонногая вибрация | Marcan adjustment and a first (TDa) | 1.0 |
| | Emotorement Description to and Line and | 10.07 27 |
| - CL. Opercend gop | Constitutions (Constitution and Village and | 6.22 |
| | Contraction of the second second second | V64 |
| | To an and a second second second second second | 15A |
| | Carry and an an and a second s | 2.05 W |
| | NOTIFICATION OF THE PROPERTY AND A DESCRIPTION OF THE PROPERTY AND A DESCR | 2000.00 |
| | Proventiers on the contraction the probability of | 7 MB 710 |
| | Appopulation and AMTO Approximation graph, (Atta) | 7482-10 |
| | Nonewersen, og regiss applagerere og vig upseptiger i up ci | 014 |
| | Kandekapeer aan regges ypy acture oos Y or tereteperges, (TTa/C) | 014 |
| | Казейныниг эж. нарукнурузститка улон 45 от тентературы, [11а/С] | 014 |
| | Kabilekuleni sab XNET or reviepatypu (periodpape) (1/10) | 4.1 E (6 |
| | Kangduapeer aan XMT or reverspangav (avergogia) (37%) | 4.102_C8 |
| | Kabél ball or teletepyrgau kabél ball för Harpinotek ("Aktodo), (1/f1a"C | 3 40843 |
| | Koole can or teverepargui i cole can FMT or varpexeevengure place, (14/1/17 | (g 4.08.1) |
| | Marcanahanan porgonanan namepangsa warpasa, [12] | 6.12 |
| | Макалальная датустиная тантяратура онгаждания. [10] | 0.00 |
| | Manzanance.porgchance vargesame reversel, (MTa) | 00 |

Theatmad year/Conf.com

Рис. 10. Параметры материалов для учета зависимости демпфирования от механических напряжений (АСОНИКА-ТМ)



Рис. 11. Ускорения электронных компонентов, в том числе на резонансах, рассчитанные с учетом нелинейности демпфирующих свойств современных материалов (АСОНИКА-ТМ)

4. В системе ACOHИКА созданы простые и интуитивно понятные постпроцессоры, позволяющие в отличие от системы ANSYS и др. сразу определять все необходимые выходные тепловые, механические и другие характеристики, необходимые для принятия решения разработчиком электроники при проектировании.

| юект Правка Вид Настройка | Анализ Приложен | ния Помощь | | | | | |
|--|---------------------|------------------------|---|---------------------|----------------|--------------|--|
| | - 8 - 1-0 | No * - 0 = | MAACTA | Lu / 5 (d) Lt Lu | | | |
| | | X | | | | | |
| manistrana [Tapetreventor.or befpitter | 🔅 Xaga | arequires Strapeneted | enyce DPM is year trice (T (F) | Marrana, J'al 💿 💠 🗢 | | | |
| Devetweix.gen | R Dargestamo | and 20 for an encourse | In the enversage in the | meet by aucator | | | |
| R- Cate | and the second | | To be a pactor of a | [constant] | | | |
| E - Creat | M 1 2 13 | Part Captoposta To | nan na kana na kana na kana kana kana k | 26 36 | | | |
| C Korrgi orea | Nº Déconomiese 3 | PH Crapove Vacrona. | Fal Disagene | feefa | Kanes response | Depergence a | |
| () -th flapsar cupora | 1 56 | | | | | | |
| (10)_measurement (21) | 2 12 | 1 1791.90 | 705.36 | 20.10 | 39.32 | 798.36 | |
| S. C. Terrine riseres groups | 3 (28 | 1 1791.90 | 745.52 | 20.14 | 37.38 | 725.97 | |
| ID 4D Fpyme 3PH | 4 CK5 | 1 1211.90 | 525.04 | 20.18 | 26.30 | 505.04 | |
| S — Balbon calbona | 5 (24 | 2 1791.90 | 433.57 | 20.10 | 2450 | 473.97 | |
| (i) (i) 2 mergegeater egener, (125) | 6 (01 | 2 1781.90 | 443.53 | 20.88 | 22.40 | 423.53 | |
| (i) O Tenning changes in the second secon | 7 (52 | 1 1651.90 | 415.00 | 20.16 | 20.90 | 295.03 | |
| In the state of th | 0 (0 | 2 1651.90 | #12.28 | 20.10 | 2014 | 362.23 | |
| E-D therees | 2 (2 | 2 1651.90 | 283.10 | 20.88 | 1945 | 373.93 | |
| 2. C voaterere oan | 10 016 | 2 409.90 | 394.05 | 20.14 | 19.96 | 574.25 | |
| II V DOLDSKIBA | 11 647 | 2 511.00 | 200.43 | 20.16 | 1952 | 323.43 | |
| The Construction and and and | 12 /20 | 2 515.93 | 793.43 | 20.88 | 1997 | 101.41 | |
| - Copenses and when | 43 65 | 2 548.00 | 200.42 | 20.04 | 1073 | 222.42 | |
| - C Operation (CD) | 14 578 | 2 1001.00 | 20.43 | 20.88 | 10.30 | 372.43 | |
| | 14 124 | 2 100 00 | 200.00 | 10.14 | 10.00 | 000.00 | |
| | 10 1/4 | 2 405.00 | 204.00 | 20.00 | 1920 | 201.02 | |
| | 14 0.04 | 4 1997.00 | 201.02 | 2018 | 19.00 | 201.01 | |
| | 17 1.00 | 2 625.00 | A11 A7 | 2018 | 1990 | AL 10 | |
| | 11 1.0 | 1 1791.90 | 113.78 | 2018 | 10.55 | 201.01 | |
| | 19 07 | 2 1791.90 | 783.59 | 40.14 | 1870 | 708.99 | |
| | 20 042 | 2 1911.90 | 371.76 | 20.10 | 10.99 | 21.5 | |
| | 21 (58 | 2 191.90 | 253.68 | 2010 | 1788 | 333.96 | |
| | 22 (19 | 2 1791.90 | 251.58 | 20.10 | 17.98 | 201.99 | |
| | 23 C16 | 2 1681.90 | 261.07 | 20.10 | 1755 | 228.87 | |
| | 24 054 | 1 1651.90 | 361.02 | 20.10 | 1795 | 338.67 | |
| | 25 (20) | 1 1651.90 | 251.07 | 50.10 | 1755 | 201.07 | |
| | 26 (27 | 2 1681.90 | 344.47 | 20.10 | 17.22 | 324.47 | |
| | 27 (197 | 2 489.90 | 343.36 | 20.00 | 1217 | 323.06 | |
| | 29 064 | 2 409.90 | 340.36 | 20.88 | 1717 | 323.36 | |
| | 29 (23 | 2 485.90 | 343.36 | 20.88 | 1217 | 323.36 | |
| | 30 017 | 2 1651.90 | 342.11 | 20.14 | 1711 | 322.11 | |
| | 31 CK8 | 2 403.90 | 241.54 | 20.88 | 17.00 | 327.54 | |
| | 32 814 | 2 1791.90 | 603.03 | 40.18 | 1673 | 623.03 | |
| | 33 877 | 2 1791.90 | 663.02 | 40.18 | 16.73 | 629.03 | |
| | 34 081 | 2 403.30 | 239.32 | 20.86 | 16.42 | 208.32 | |
| | 16 (22 | 2 403.90 | 329 32 | 20.16 | 16.42 | 208.32 | |
| | 36 12 | 2 111.90 | 325.67 | 20.88 | 16.20 | 305.67 | |
| | 27 812 | 2 1791.90 | 643.63 | 40.18 | 16.22 | 603.63 | |
| | The state of states | Canal Canall | | | | | |
| | Treves Hall youry | Concernants | | | | | |



Рис. 12. Ускорения электронных компонентов в сравнении с допустимыми значениями, автоматически взятыми из базы данных (АСОНИКА-ТМ)





Рис. 13. Температуры электронных компонентов в сравнении с допустимыми значениями, автоматически взятыми из базы данных (АСОНИКА-ТМ)



Рис. 14. Ускорения и механические напряжения в блоках (АСОНИКА-М-3D)



Рис. 15. Ускорения и механические напряжения в шкафах (АСОНИКА-М-ШКАФ)

5. В системе ACOHUKA в отличие от системы ANSYS и др. есть возможность идентификации физико-механических, теплофизических и других параметров, что крайне необходимо для обеспечения точности моделирования, так как многие параметры отсутствуют в справочниках.



Рис. 16. Идентификация физико-механических параметров в подсистеме АСОНИКА-ИД

6. В системе ACOHUKA в отличие от системы ANSYS и др. есть возможность параметрической и структурной оптимизации конструкций на виброизоляторах в подсистеме ACOHUKA-B, что крайне необходимо для обеспечения стойкости аппаратуры к механическим воздействиям.



Рис. 17. Параметрическая и структурная оптимизация конструкций на виброизоляторах в подсистеме АСОНИКА-В



Рис. 18. Создание карт рабочих режимов электронных компонентов в подсистеме АСОНИКА-В

7. В системе АСОНИКА в отличие от системы ANSYS и др. есть возможность создания карт рабочих режимов электронных компонентов с учетом тепловых и механических характеристик, полученных в результате моделирования и автоматически передаваемых в подсистему АСОНИКА-Р.

8. В системе ACOHUKA в отличие от системы ANSYS и др. есть возможность анализа показателей надёжности электронных компонентов и аппаратуры в целом с учетом тепловых и механических характеристик, полученных в результате моделирования и автоматически передаваемых в подсистему ACOHUKA-Б.

| 101 | | | | |
|-------------------------|---|--|---------------------------------------|--|
| 101 | | Tonico pedikticuji we nepikretou | | |
| | F | Orecones (seared eccal | 2-m-mark | |
| 01 | | Harranceaters | 1000-02 | |
| 100 | | O for some warman | DI | |
| 102 | | Hauteneness status 251 | Penantania | |
| 105 | | Harmonian marka 201 | Konstructure autoreaute of the second | |
| | | Consult and other than the feature and and only | 0.999853298242953 | |
| 107 | | Departments of tasks | 0.030746209232140993 | |
| 0.00 | | Descent upperforme an original instruction of | BN0912142 857141 | |
| 0.05 | | Constante a presse dia batta da poli pradicita, 14 | BR0902142 857141 | |
| 010 | | Octaving a particular bit | 852776142857141 | |
| 1011 | | Descent state and | 0.000002000000 | |
| 012 | - | Dependence in a second material parameter | 1.47-0 | |
| D13 | - | An and a set of the se | 0.1 | |
| 014 | - | A stangertypen i a metopermitteten metopeten (optimet) | | |
| 015 | 5 | Proventier and a start per second start and a 20 | | |
| 016 | - | Comparison of the second | 60 61 | |
| O17 | - | And Andrews and Street Andrews | R1 | |
| 018 | - | Network Presses, IV | 13264 | |
| O19 | - | Ford-district a seence of residential ord-distribution to an | 1 | |
| 020 | - | Tobe disamini regenia | 0.854 | |
| < O21 | - | Food disadely including | 0.033 | |
| 022 | - | Foogeruppert regetin | 371 | |
| 023 | - | Foogerupht Hogeth | 7.221 | |
| 024 | - | Kooperuser regete | 0.01 | |
| 015 | - | Folderware Production | 1.325 | |
| 026 | - | Козфеналінт рекнича в завикончасти от злектрическої натрузки и тентератури акрукахецей среди | 0.760635653479090 | |
| 022 | - | Постоянные ноза фикциенты изденти | 0.033 | |
| 026 | - | Постоянные ноза фикциенты изденти | 0.055 | |
| 019 | | Постоянные нова фикциенты изделян | 323 | |
| 010 | | Постоянная нояффикциянты назвити | 7.223 | |
| 011 | | Docromous x030 (usual to 1900) | 2.095 | |
| 0.022 | | Постоянные ноздірнациента надели | 1 | |
| 012 | | Pocromise Hoad distant a Highty | 1.335 | |
| 014 | | 3HP-KHARI KOSAGHUHANTR ROHDRHH | 1 | |
| 0.004 | | Consequences of the conseq | 05 | |
| 010 | | Constant which is independent of otherwises patients and constant to post-off-there are the patient of the pati | | |
| 0.000 | | Colonal which wron at here | 1 | |
| C D 3/ | _ | La . | | |

Рис. 19. Расчёт надёжности в подсистеме АСОНИКА-В

9. В системе АСОНИКА в отличие от системы ANSYS и др. создается электронная модель изделия в подсистеме АСОНИКА-УМ, необходимая для реализации CALS-технологий в электронике.



Рис. 20. Создание электронной модели изделия в подсистеме АСОНИКА-УМ

ЛИТЕРАТУРА

- [1] Автоматизированная система АСОНИКА для проектирования высоконадежных радиоэлектронных средств на принципах CALS-технологий. Том 1/ Под ред. Кофанова Ю.Н., Малютина Н.В., Шалумова А.С. М.: Энергоатомиздат, 2007. 368 с.
- [2] Автоматизированная система АСОНИКА для моделирования физических процессов в радиоэлектронных средствах с учетом внешних

воздействий / Под ред. А.С. Шалумова. М.: Радиотехника, 2013. 424 с.

- [3] Шалумов М.А., Шалумов А.С. Виртуальная среда проектирования РЭС на основе комплексного моделирования физических процессов. Владимир: Владимирский филиал РАНХиГС, 2016. 87 с.
- [4] Шалумов А.С., Шалумов М.А. Опыт применения автоматизированной системы АСОНИКА в промышленности Российской Федерации: монография. Владимир : Владимирский филиал РАНХиГС, 2017. 422 с.

Virtual Tests of Micro- and Nanoelectronic Systems on External Influences

A.S. Shalumov, D.N. Travkin, M.V. Tikhomirov

Scientific-research institute «ASONIKA», Kovrov, als@asonika-online.ru

Abstract — The article considers the purpose of virtual tests of micro- and nanoelectronic systems and their optimal combination with full-scale tests. The capabilities of the automated system ASONIKA are presented, on the basis of which virtual tests for mechanical (vibration, shock, linear accelerations, acoustic noises), thermal, electromagnetic effects are carried out. The technology based on the ASONIKA system is the only one in Russia that allows the end-to-end design of highly reliable MES of space, aviation and other mobile objects taking into account external thermal, mechanical, electromagnetic influences from the technical task and to the production of a prototype. The created electronic model for the first time will allow implement CALS-technology in electronics at all 11 stages of the life cycle from marketing research and to utilization. Automated system ASONIKA has no analogues or comparable prototypes in the field of modeling of highly reliable electronics both in Russia and abroad and has incomparable advantages in comparison with the wellknown foreign systems ANSYS, NASTRAN, COSMOS, COMSOL, etc., which are discussed in detail in this article.

Keywords — virtual testing, simulation, acceleration, stress, temperature, fatigue failure.

REFERENCES

- Automated system ASONIKA for the design of highly reliable radioelectronic facilities on the principles of CALStechnologies. Volume 1 / Ed. Kofanov Yu.N., Malyutin N.V., Shalumov A.S. Moscow: Energoatomizdat, 2007. 368 p.
- [2] Automated system ASONIKA for modeling of physical processes in radio-electronic means taking into account external influences / Ed. A.S. Shalumov. Moscow: Radio Engineering, 2013. 424 p.
- [3] Shalumov M.A., Shalumov A.S. Virtual environment for design of RES on the basis of complex modeling of physical processes. Vladimir: Vladimir Branch of the Russian Academy of Sciences, 2016. 87 p.
- [4] Shalumov A.S., Shalumov M.A. Experience of application of the automated system ASONIKA in the industry of the Russian Federation: monograph. Vladimir: Vladimir Branch of the Russian Academy of Science and Technology, 2017. 422 p.

Разработка и исследование моделей блоков цифровых систем на основе их представления в виде семейства стационарных динамических систем

А.Д. Иванников

Институт проблем проектирования в микроэлектронике РАН (ИППМ РАН),

adi@ippm.ru

Аннотация — При отладке проектов цифровых систем методом моделирования важной задачей является выбор набора отладочных тестов, то есть входных воздействий, которые подаются на компьютерную модель проектируемой системы с целью проверки правильности ее функционирования. Формирование полного в какомто смысле набора отладочных тестов возможно тем или иным способом, если известно множество допустимых входных воздействий на проектируемую систему. Формирование описания такого множества возможно, если известно описание множества допустимых входных воздействий на блоки проектируемой цифровой системы. В статье осуществляется исследование моделей блоков цифровых систем прежде всего с точки зрения описания множества допустимых входных воздействий. В качестве модели цифровых блоков используется множество стационарных динамических систем с непрерывным временем и дискретными значениями логических сигналов. Поскольку в ряде случаев обмен сигналами цифровых блоков с другими блоками системы и внешним миром инициируется самими блоками, то в качестве рассматриваются отладочных тестов входные взаимодействия, включающие изменения как входных сигналов блока, так и выходных сигналов управления графовое обменом. Предлагается представление допустимых входных взаимодействий цифровых блоков и системы в целом для каждой выполняемой функции.

Ключевые слова — логико-временной анализ цифровых систем, отладка методом моделирования, структура множества входных взаимодействий, входные взаимодействия цифровых блоков, графовое представление множества входных взаимодействий

I. Введение

При проектировании цифровых систем для отладки проектов широко используется метод моделирования. На компьютерную модель цифровой системы подаются некоторые входные воздействия, а реакция модели проектируемой системы проверяется на соответствие техническому заданию [1-4].

При этом важной задачей является выбор конечного числа конечных по времени тестовых входных взаимодействий (тестовых примеров). С ростом сложности проектируемых цифровых систем и, соответственно, ростом сложности и длительности тестирования их проектов все более актуальной становится задача выбора минимального полного в определенном смысле набора тестов, правильное выполнение которого позволяет убедиться в отсутствии ошибок проектирования [5-7].

Для того, чтобы составить набор входных тестовых примеров необходимо иметь описание множества допустимых входных воздействий как на разрабатываемую цифровую систему, так и на ее блоки. Целью настоящего исследования является разработка моделей блоков цифровых систем прежде всего с точки зрения описания множества допустимых входных воздействий.

II. Описание используемой модели

При отладке проектов цифровых систем методом моделирования необходимо выбрать уровень или уровни моделей цифровой системы и ее блоков. Обычно осуществляется декомпозиция задачи отладки проекта [8] прежде всего по типу выявляемых ошибок. Так, для верификации временных диаграмм обмена информации между блоками используются модели цифровых элементов с многозначным представлением [9]. электрических сигналов Для проверки правильности логического функционирования используются модели с булевым представлением сигналов на входах и выходах [6, 10]. Используются также различные высокоуровневые модели [11-14].

При проектировании сложных цифровых систем разработчик должен обеспечить, прежде всего, требуемое внешнее поведение цифровой системы, то есть требуемое взаимодействие системы с внешней средой. При этом существенным является как последовательность выходных сигналов цифровой системы, так и моменты времени появления и изменений этих сигналов, причем временные ограничения обычно задаются интервалами значений. В связи с этим математической моделью внешнего поведения проектируемой цифровой системы или ее блока может служить семейство стационарных динамических систем [1, 15].

Взаимодействие цифровой системы с объектом управления и внешним миром вообще осуществляется

через внешние линии и шины – наборы линий, по которым передается однородная информация, например, адреса или данные. Причем в цифровых управления широко используются системах двунаправленные шины и линии, имеющие также состояние с высоким выходным сопротивлением Будем (отключенное состояние). рассматривать логическую модель сигналов на шинах и линиях цифровых систем, то есть считать, что значения сигналов представляются как 0 или 1 на линиях и как число из диапазона 0 - 2ⁿ-1 на шинах системы. Цифровые сигналы внешних шин и линий назовем терминальными переменными – множество Р. Переменная *p*∈**P** всегда имеет одно из значений конечного множества **Z**_p , элементы которого определяют как целочисленное значение сигнала, так и направленность работы шины или линии.

Событием по переменной *p* называется изменение переменной *p* со значения $z_1 \in \mathbf{Z}_p$ на значение $z_2 \in \mathbf{Z}_p$ в момент времени *t*. Обозначим такое событие χ_{p,z_1,z_2}^t . Взаимодействие цифровой системы с внешней средой, включая управляемый объект, есть последовательность переключений сигналов на терминальных шинах и линиях, то есть последовательность событий. Для каждой проектируемой системы имеется множество Ψ допустимых взаимодействий с внешней средой, каждое из которых есть отображение ψ : $[0,t) \rightarrow \mathbf{Q}, t \in \mathbf{T}, \mathbf{Q} = \prod_{p \in \mathbf{P}} \mathbf{Z}_p$.

В цифровых системах для каждого конечного временного интервала количество событий по терминальным переменным, то есть количество изменений их значений, конечно. В связи с этим любое взаимодействие ψ может быть представлено в виде вектора ($z_{p_1,...,}^{H}z_{p_k}^{H}$) начальных значений переменных $p_1,..., p_k$ (k – мощность множества **P**) в момент времени t = 0 и последовательности событий по переменным множества **P** с конечным числом событий за любой конечный интервал времени:

$$\psi = (z_{p_1,\dots,}^{\mathsf{H}} z_{p_k}^{\mathsf{H}}), \, \chi_{p_{i_1},z_{j_1},z_{j_2}}^{t_1}, \, \chi_{p_{i_2},z_{j_3},z_{j_4}}^{t_2}, \, \chi_{p_{i_3},z_{j_5},z_{j_6}}^{t_3} \dots, \quad (1)$$

где $t_1 \le t_2 \le t_3 \le \dots$ – упорядоченная последовательность времен событий;

 $p_{i_1,p_{i_2,p_{i_3,\dots}}$ – переменные, принадлежащие множеству **Р**;

 Z_{j_1} , Z_{j_3} , Z_{j_5} ,... – значения переменных непосредственно перед событием;

 Z_{j_2} , Z_{j_4} , Z_{j_6} ,... – значения переменных непосредственно после события.

Если в последовательности (1) выделить только события, являющиеся изменениями входных сигналов, то такую последовательность можно назвать входным воздействием. Однако часто моменты подачи входных сигналов на цифровую систему определяются готовностью системы принять эти сигналы, на что указывают определенные выходные сигналы системы. Выполнение какой-либо операции, например, считывания данных цифровой системой, может инициироваться не сигналами внешней среды, а самой системой. В связи с этим использование в качестве аргументов функционирования цифровой системы входных воздействий не всегда удобно.

Выделим из последовательности событий (1) взаимодействия ψ – последовательность входных событий и выходных событий управления обменом, которые по заданному протоколу обмена обуславливают моменты времени входных событий. Назовем эту последовательность входным взаимодействием:

$$\mu = (z_{p_1,\dots,z_{p_{n+q}}}^{\mathsf{H}}), \chi_{p_{i_1},z_{j_1},z_{j_2}}^{t_1}, \chi_{p_{i_2},z_{j_3},z_{j_4}}^{t_2}, \chi_{p_{i_3},z_{j_5},z_{j_6}}^{t_3} \dots,$$

$$(2)$$

где $\chi_{p_{i_1},z_{j_1},z_{j_2}}^{t_1}$, $\chi_{p_{i_2},z_{j_3},z_{j_4}}^{t_2}$, $\chi_{p_{i_3},z_{j_5},z_{j_6}}^{t_3}$... – входные события и выходные события управления обменом;

 $t_1 \le t_2 \le t_3 \le \ldots$ – упорядоченная последовательность времен событий входного взаимодействия.

В рассматриваемой модели в качестве аргументов функционирования цифровых систем используются входные взаимодействия, что дает возможность рассматривать режимы работы, инициируемые как внешними входными сигналами, так и самими цифровыми системами [1, 13]. Все вышесказанное относится не только к цифровым системам в целом, но и к цифровым блокам, из которых цифровые системы состоят.

III. УЧЕТ ВРЕМЕННЫХ ОГРАНИЧЕНИЙ В МОДЕЛИ ДОПУСТИМЫХ ВЗАИМОДЕЙСТВИЙ

Каждый блок цифровой системы в процессе функционирования выполняет ту или иную последовательность функций (операций) из конечного алфавита функций **К**. Выполнение каждой функции вызывается одним из входных взаимодействий определенного класса, причем каждое входное взаимодействие этого класса содержит конечное число событий.

Обозначим через f конечную последовательность функций, а через \mathbf{F} в общем случае счетное множество конечных последовательностей f. Каждая последовательность функций f, начинающаяся с момента времени t = 0 (например, включения питания), крайней мере одним задается по входным взаимодействием $\mu^f \in \mathbf{M}$. Этот факт следует из того, что Μ содержит все допустимые входные взаимодействия любой допустимой для последовательности функций цифрового блока.

В большинстве случаев одни и те же функции могут выполняться с различными наборами данных, что обуславливает задание различными μ одной и той же последовательности функций *f*. В связи с тем, что для различных экземпляров блока цифровой системы задержки выходных событий управления обменом относительно входных событий различаются в определенных пределах, а также в связи с допустимостью варьирования моментов времени входных событий относительно друг друга и относительно выходных событий управления обменом, множество **M** содержит континуальное подмножество $\mathbf{M}^{f} \subset \mathbf{M}$ входных взаимодействиий, каждое из которых вызывает выполнение цифровым блоком конечной последовательности функций *f*. Множество входных взаимодействий может быть представлено в виде:

$$\mathbf{M} = \bigcup_{f \in \mathbf{F}} \mathbf{M}^{f}; \ \mathbf{M}^{f'} \cap \mathbf{M}^{f''} = \emptyset$$
 при $f' \neq f''$. (3)

Входное взаимодействие $\mu \in \mathbf{M}^f$ содержит конечное множество событий

$$\{\chi_{p_{i_1}, z_{j_1}, z_{j_2}}^{t_1}, \dots, \chi_{p_{i_n}, z_{j_{2n-1}}, z_{j_{2n}}}^{t_n}\},\tag{4}$$

где *n* – количество событий в *µ*.

Множество \mathbf{M}^{f} содержит также входные взаимодействия, времена событий в которых различаются в определенных пределах. Ограничения на эти различия могут быть заданы в виде:

$$t_{min}^{l,m} \le t_m - t_l \le t_{max}^{l,m}$$
,
 $(l,m) \in \mathbf{C}, \mathbf{C} \subset \{1,2,...,n\} \times \{1,2,...,n\},$

где $t_{min}^{l,m}$, $t_{max}^{l,m}$ – минимально и максимально допустимые промежутки времени между *l*-м и *m*-м событиями;

С – конечное множество пар событий из (2), для которых заданы временные ограничения.

Выделим в C все пары (l,m), для которых t_m есть время выходного события управления обменом, в множество C_{вых}, а все пары (l,m), для которых t_m есть время входного события, в множество C_{вх}. Тогда ограничения на моменты времени выходных событий обмена есть

$$t_{min}^{l,m} \leq t_m - t_l \leq t_{max}^{l,m}$$
, $(l,m) \in \mathbf{C}_{\text{bbin}}$,

а ограничения на моменты времени входных событий

$$t_{min}^{l,m} \le t_m - t_l \le t_{max}^{l,m} , (l,m) \in \mathbf{C}_{\text{BX}}.$$
 (5)

Рассмотрим пространство $G = \prod_{CBMX} \{t_m - t_l | t_m - t_l \ge 0\}$. Каждая точка $g \in G$ определяет конкретные значения задержек выходных событий управления обменом. В пространстве G выделим область $G_f \in G$, для всех точек которой выполняются ограничения (5). Область G_f определяет допустимые задержки выходных событий управления обменом. При этом $G_f \ne \emptyset$.

Для любой точки $g \in \mathbf{G}_f$ в связи с допустимостью таких задержек выходных событий управления обменом существует непустое множество входных взаимодействий \mathbf{M}_g^f , обеспечивающих выполнение цифровым блоком последовательности функций *f*.

$$\mathbf{M}^f = \bigcup_{g \in \mathbf{G}_f} \mathbf{M}^f_g, \mathbf{M}^f_g \neq \emptyset,$$

где \mathbf{M}_{g}^{f} — множество входных взаимодействий, обеспечивающих выполнение конечной последовательности функций f при фиксированных задержках выходных событий управления обменом, определяемых $g \in \mathbf{G}_{f}$.

Таким образом, множество допустимых входных взаимодействий представимо в виде:

$$\begin{split} \mathbf{M} &= \bigcup_{f \in \mathbf{F}} \mathbf{M}^{f}; \, \mathbf{M}^{f'} \cap \mathbf{M}^{f''} = \varnothing \text{ при } f' \neq f'''; \\ \mathbf{M}^{f} &= \bigcup_{g \in \mathbf{G}_{f}} \, \mathbf{M}_{g}^{f}; \, \mathbf{G}_{f} \neq \varnothing; \, \mathbf{M}_{g}^{f} \neq \varnothing \text{ при } f \in \mathbf{F}. \end{split}$$
 (6)

Таким образом, мы определяем структуру множества допустимых входных взаимодействий, то есть структуру возможных аргументов функционирования цифровых блоков.

IV. ЗАДАНИЕ МНОЖЕСТВА ВХОДНЫХ ВЗАИМОДЕЙСТВИЙ ДЛЯ КАЖДОЙ ФУНКЦИИ

Рассмотрим способ задания множества М^{*k*} входных взаимодействий, обуславливающих выполнение цифровым блоком функции *k*.

Учитывая (2) и (4), каждое $\mu, \mu \in M^k$ может быть задано в виде:

$$\begin{aligned} &((z_{p_1}^{\scriptscriptstyle H}, \ldots, z_{p_m}^{\scriptscriptstyle P}), \{(t_1, p_{i_1}, z_1', z_1''), (t_2, p_{i_2}, z_2', z_2''), \ldots, \\ &(t_n, p_{i_n}, z_n', z_n'')\}), \end{aligned}$$

где $z_{p_1}^{H}$, ..., $z_{p_m}^{H}$ – начальные значения переменных;

 t_i, p_i, z'_i, z''_i – четверка, описывающая i-ое событие.

Исходя из этого, попробуем задать все множество М^{*k*} как:

$$\begin{split} & (\tilde{\mathbf{Z}}_{\mathbf{p}_{1}}^{\text{H}}, \dots, \tilde{\mathbf{Z}}_{\mathbf{p}_{m}}^{\text{H}}), \{(\theta_{1}, p_{i_{1}}, \tilde{\mathbf{Z}}_{1}', \tilde{\mathbf{Z}}_{1}''), (\theta_{2}, p_{i_{2}}, \tilde{\mathbf{Z}}_{2}', \tilde{\mathbf{Z}}_{2}''), \dots, \\ & (\theta_{n}, p_{i_{n}}, \tilde{\mathbf{Z}}_{n}', \tilde{\mathbf{Z}}_{n}'')\}, \end{split}$$

$$\begin{split} & (\theta_{1}, p_{i_{n}}, \tilde{\mathbf{Z}}_{n}', \tilde{\mathbf{Z}}_{n}'')\}, \end{split}$$

$$\end{split}$$

$$\end{split}$$

$$\end{split}$$

$$\end{split}$$

$$\end{split}$$

$$\end{split}$$

$$\end{split}$$

где $\theta_1, \theta_2, \dots, \theta_n$ – времена событий;

 $\tilde{\mathbf{Z}}_{\mathbf{p}_{i}}^{\text{H}}$, *i*=1,2,...,*m* – подмножество возможных начальных значений переменных множества $\mathbf{P}' \cup \mathbf{P}^{0}$;

m – мощность множества $\mathbf{P}' \cup \mathbf{P}^0$;

 $p_{i_1}, p_{i_2}, \dots, p_{i_n}$ – принадлежат множеству $\mathbf{P}' \cup \mathbf{P}^0$;

 $\tilde{\mathbf{Z}}'_1$, $\tilde{\mathbf{Z}}'_2$, ..., $\tilde{\mathbf{Z}}'_n$ — подмножества возможных значений переменных $p_{i_1}, p_{i_2}, ..., p_{i_n}$ перед событием;

 $\tilde{Z}_{1}^{''}$, $\tilde{Z}_{2}^{''}$, ..., $\tilde{Z}_{n}^{''}$ – подмножества возможных значений переменных $p_{i_{1}}, p_{i_{2}}, ..., p_{i_{n}}$ после события;

С – множество пар событий, для промежутков между которыми заданы временные ограничения.

Входное взаимодействие μ , заданное в виде (7), принадлежит M^k , определенному в виде (8), если:

a)
$$z_{p_1}^{\scriptscriptstyle H} \in \tilde{\mathbf{Z}}_{\mathbf{p}_1}^{\scriptscriptstyle H}$$
, ..., $z_{p_m}^{\scriptscriptstyle H} \in \tilde{\mathbf{Z}}_{\mathbf{p}_m}^{\scriptscriptstyle H}$

б) существует изоморфизм между множеством событий (7) и множеством четверок в (8), такой, что у соответствующих событий совпадают имена переменных: $z'_1 \in \tilde{\mathbf{Z}}'_1, ..., z'_n \in \tilde{\mathbf{Z}}'_n, z''_1 \in \tilde{\mathbf{Z}}''_1, ..., z''_n \in \tilde{\mathbf{Z}}''_n$, а времена событий $t_1, ..., t_n$ в (9) удовлетворяют ограничениям из (8).

Рассматриваемое представление M^k и введение подмножеств $\tilde{\mathbf{Z}}_{\mathbf{P}_l}^{\mathsf{H}}, \tilde{\mathbf{Z}}', \tilde{\mathbf{Z}}''$ позволяют представить множество входных взаимодействий с различными наборами данных.

Пусть задано множество терминальных переменных **P**, каждая переменная с множеством значений \mathbf{Z}_p . Для каждой переменной *p* рассмотрим некоторый алфавит значений **3** $_p$ = $\mathbf{Z}_p \cup \{\mathbf{3}_{ij} | (z_i \in \mathbf{Z}_p) \otimes (z_j \in \mathbf{Z}_p) \otimes (z_i \neq z_j) \}$, где $\mathbf{3}_{ij}$ – переход от $z_i \ltimes z_j$. В алфавите $\mathbf{3}_p$ выделим непустые подмножества $\mathbf{3}_l^p \subset \mathbf{3}_p$, такие, что $\mathbf{3}_p = \bigcup_l \mathbf{3}_l^p$. Например, если сигнал на шине может принимать значения 0-255, \$ (\$ – состояние с высоким выходным сопротивлением), то возможно использование следующих подмножеств:

Здесь $\mathbf{3}_2$ – стабильное значение данных, $\mathbf{3}_4$ – нестабильное значение.





Подмножества $\mathbf{3}_{l}^{p}$, l=1,2,... образуют конечный алфавит $\mathbf{\hat{3}}_{p}$. Каждое множество M^{k} может быть представлено как конечное множество событий в алфавитах $\mathbf{\hat{3}}_{p}$. Так как на множестве событий множества M^{k} определен частичный порядок событий по времени, то естественным представлением M^{k} является ориентированный граф $\mathcal{G}^{k}(\mathbf{V}^{k}, \mathbf{E}^{k})$, где каждая вершина из \mathbf{V}^{k} соответствует переходу одной из переменных из одного значения $\mathbf{3}_{l}^{p}$ в другое. Каждую вершину $v, v \in \mathbf{V}^{k}$ пометим обозначением переменной p и множествами $\mathbf{3}_{l}^{p}, \mathbf{3}_{l'}^{p}$, если вершине vсоответствует переход переменной p из значения $\mathbf{3}_{l}^{p}$ в значение $\mathbf{3}_{l'}^{p}$.

На множестве вершин \mathbf{V}_p^k , соответствующих изменениям значений одной и той же переменной p, задано отношение частичного порядка во времени, что определяет множество ребер \mathbf{E}^k . Каждое ребро пометим двумя числами t_{\min} , t_{\max} , причем $0 \le t_{\min} \le t_{\max} \le \infty$.

В качестве примера рассмотрим некоторый цифровой блок параллельного интерфейса вводавывода, представленный на рис. 1а, временные диаграммы работы которого приведены на рис. 1б. При функционировании блока используются сигналы: ШД[0-7] – шина данных; А[0-1], А[2-15] – шины адреса соответствующей разрядности; СБР, ЧТ, ЗП – сигналы сброса, чтения и записи; ВБ, КГТ, ППРД – сигналы выбора блока, канал готов, подтверждения передачи; ШКН1, ШКН2, ШКН3 – шины каналов 1, 2 и 3 с указанием разрядности.



Рис. 2. Задание множества входных взаимодействий для функции записи информации в блок параллельного ввода-вывода

На рис. 2 представлено графовое задание множества М^k входных взаимодействий, соответствующих записи информации в рассматриваемый цифровой блок. Для этого случая:

P={A, ШД, $\overline{B}\overline{B}$, $\overline{3}\overline{\Pi}$, $\overline{K}\Gamma\overline{T}$, $\overline{\Pi}\overline{\Pi}\overline{P}\overline{A}$ };

$$\begin{split} \mathbf{Z}_{A} = \{0, 1, 2, 3, \$\}; \ \mathbf{Z}_{III,A} = \{0, 1, ..., 255, \$\}; \ \mathbf{Z}_{\overline{B}\overline{B}} = \{0, 1\}; \\ \mathbf{Z}_{\overline{3}\Pi} = \{0, 1\}; \ \mathbf{Z}_{\overline{K}\overline{\Gamma}\overline{T}} = \{0, 1\}; \ \mathbf{Z}_{\overline{\Pi}\overline{\Pi}\overline{P}\overline{A}} = \{0, 1\}; \\ \widehat{\mathbf{3}}_{A} = \{C, HC, BH, X\}, C = \{0, 1, 2, 3\}; HC = C \cup \\ \{\chi_{i,k} | z_i \neq z_k, z_i \in C, z_k \in C\}; \\ BH = \{\$\}, X = HC \cup BH \cup \{\chi_{i,\$} | z_i \in C\} \cup \{\chi_{\$,i} | z_i \in C\}; \\ \widehat{\mathbf{3}}_{III,\overline{A}} = \{C, HC, BH, X\}, C = \{0, 1, ..., 255\}; \\ HC = C \cup \{\chi_{i,k} | z_i \neq z_k, z_i \in C, z_k \in C\}; \\ BH = \{\$\}, X = HC \cup BH \cup \{\chi_{i,\$} | z_i \in C\} \cup \{\chi_{\$,i} | z_i \in C\}; \\ \widehat{\mathbf{3}}_{\overline{B}\overline{B}} = \{0, 1, X\}, X = \{0, 1, \mathfrak{z}_{0,1}, \mathfrak{z}_{1,0}\}; \\ \widehat{\mathbf{3}}_{\overline{3}\overline{\Pi}} = \{0, 1, X\}, X = \{0, 1, \mathfrak{z}_{0,1}, \mathfrak{z}_{1,0}\}; \end{split}$$

 $\widehat{\mathbf{3}}_{\overline{\mathrm{KTT}}} = \{0, 1, X\}, X = \{0, 1, \mathfrak{z}_{0,1}, \mathfrak{z}_{1,0}\};$

 $\widehat{\mathbf{3}}_{\overline{\Pi}\overline{\Pi}\overline{P}\overline{A}} = \{0, 1, X\}, X = \{0, 1, \mathfrak{z}_{0,1}, \mathfrak{z}_{1,0}\}.$

На рис. 2 вершины графа помечены обозначением переменной и новым ее значением. Если заданы начальные значения переменных в алфавите $\hat{\mathbf{3}}_p$, этого достаточно. При наличии графа $\mathcal{G}^k(\mathbf{V}^k, \mathbf{E}^k)$ указанного вида для каждого μ , заданного в виде (7), можно определить, принадлежит ли μ множеству М^k или нет.

Граф $G^{k}(\mathbf{V}^{k}, \mathbf{E}^{k})$ более наглядно задает множество M^{k} , чем временная диаграмма (рис. 1б), которая обычно используется для представления режимов работы цифровых блоков.

V. Заключение

Граф $\mathcal{G}^{k}(\mathbf{V}^{k}, \mathbf{E}^{k})$ определяет множество входных взаимодействий для выполнения функции k цифровым блоком. Используя тот же алгоритм, можно построить аналогичный граф для каждой функции цифровой системы в целом.

Этот граф вместе с выражениями (3) и (6) определяет структуру множества допустимых входных взаимодействий как для каждого цифрового блока в отдельности, так и для цифровой системы в целом. Структура множества допустимых взаимодействий служит исходными данными для выбора набора тестов для отладки проектов цифровых систем методом моделирования.

Поддержка

Работа выполнена при поддержке гранта РФФИ № 17-07-00683.

ЛИТЕРАТУРА

- [1] Иванников А.Д., Стемпковский А.Л. Формализация задачи отладки проектов цифровых систем // Информационные технологии. 2014. № 9. С. 3-10.
- [2] Lin, Yi-Li; Su, Alvin W.Y. Functional Verification for SoC Software/Hardware Co-Design: From Virtual Platform to Physical Platform // 2011 IEEE International SOC Conference (SOCC), pp. 201-206.
- [3] Matsuda, A.; Ishihara, T. Developing an Integrated Verification and Debug Methodology // Design, Automation

& Test in Europe Conference & Exhibition (DATE), 2011, pp. 1-2.

- [4] Shi, Jin; Liu, Weichao; Jiang, Ming; el al. Software Hardware Co-Simulation and Co-Verification in Safety Critical System Design // 2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT), pp. 71-74.
- [5] Иванников А.Д. Формирование отладочного набора тестов для проверки функций цифровых систем управления объектами // Мехатроника, автоматизация, управление. 2017. Т. 18. № 12. С. 795-801.
- [6] Кащеев Н.И., Пономарев Д.М., Подъяблонский Ф.М. Построение тестов цифровых схем с использованием обобщенной модели неисправностей и непрерывного подхода к моделированию // Вестник Нижегородского университета им. Н.И.Лобачевского. 2011. №3 (2). С. 72-77.
- [7] Cruz, A.M., Fernandez, R.B., Lozano, H.M., Ramirez Salinas, M.A., Vila Vargas, L.A. Automated Functional Test Generation for Digital Systems Through a Compact Binary Differential Evolution Algorithm // Journal of Electronic Testing-Theory and Applications. 2015. V. 31. № 4. P. 361-380.
- [8] Иванников А.Д. Анализ методов декомпозиции задачи отладки проектов цифровых систем // Информационные технологии. 2016. Т7 22. № 10. С. 758-763.
- [9] Стемпковский А.Л., Гаврилов С.В., Глебов А.Л. Методы логического и логико-временного анализа цифровых КМОП СБИС. М.: Наука, 2007. 220 с.
- [10] Jasnetski, A., Oyeniran, S. A., Tsertoy, A. High-Level Modeling and Testing of Multiple Control Faults in Digital Systems // IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS). 2016. Paper # 7482445.
- [11] Березкин А.В., Федотов A.A., Филиппов А.С. Тестирование цифровых заданных систем, высокоуровневыми спецификациями // Научно-Санкт-Петербургского технические ведомости политехнического государственного университета. Информатика. Телекоммуникации. Управление. 2011. Т. 6-1. № 138. C. 62-70.
- [12] Jain, S., Govani, P., Poddar, K.B., Lal, A.K., Parmar, R.M. Functional verification of DSP based on-boad VLSI design // International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA). 2016. P. 1-4.
- [13] Гаврилов С.В., Иванова Г.А., Стемпковский А.Л. Теоретико-графовая модель сложно-функциональных блоков для КМОП технологий с трехмерной структурой транзистора // Известия ЮФУ. Технические науки. 2014. № 7 (156). С. 58-68.
- [14] Jasnetski A., Oyeniran S.A., Tsertoy A. High Level Modeling and Testing Of Multiple Control Faults in Digital Systems // IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS). 2016. Paper # 7482445.
- [15] Иванников А.Д., Стемпковский А.Л. Математическая модель отладки проектов сложных цифровых систем и микросистем на основе представления последних в виде семейства стационарных динамических систем // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2014. Часть II. С. 123-128.

Research and Development of Digital System Block Models Based on their Description as a Stationary Dynamical System Family

A.D. Ivannikov

Institute for Design Problems in Microelectronics of Russian Academy of Sciences (IPPM RAS), adi@ippm.ru

Abstract — While digital system design debugging by computer simulation the important task is to generate debugging test set, e.g. set of input signals which are applied to a designing system computer model for checking the correctness of its functioning. The generation of complete in some sense debugging test set is possible by some way if the permissible input action set for the system is known. Description forming of such a set is possible if permissible input interaction set for digital system blocks are known. Digital system block model investigation is carried out, first of all, from the point of a set of permissible input interactions. The family of stationary dynamic systems with continuous time and logical signal discrete values are used as models for digital system blocks. In some cases signal exchange between blocks and with outer world is initiated by a block itself. That is why input interactions including input signals and output exchange driving signals are considered as debugging tests. For the description of permissible input interactions of digital system blocks and the system as a whole graph representation is proposed for each fulfilled function.

Keywords — digital system logical and timing analysis, debugging by simulation, input interaction set structure, digital block input interactions, graph representation for input interaction set.

REFERENCES

- Ivannikov A.D., Stempkovsky A.L. Formalizaciya zadachi otladki proektov cifrovih system (Formal Model of Digital System Design Debugging Task). Informacionnie Technologii, 2014, no. 9, pp. 3-10 (in Russian).
- [2] Lin, Yi-Li; Su, Alvin W.Y. Functional Verification for SoC Software/Hardware Co-Design: From Virtual Platform to Physical Platform. 2011 IEEE International SOC Conference (SOCC), pp. 201-206.
- [3] Matsuda, A.; Ishihara, T. Developing an Integrated Verification and Debug Methodology. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011, pp. 1-2.
- [4] Shi, Jin; Liu, Weichao; Jiang, Ming; el al. Software Hardware Co-Simulation and Co-Verification in Safety Critical System Design. 2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT), pp. 71-74.
- [5] Ivannikov A.D. Formirovanie otladochnogo nabora testov dlya proverki funkciy cifrovih system upravleniya obektami (Debugging Input Set Generation for Testing of Control Digital Systems Functions). Mekhatronika, Avtomatizatciya, Upravlenie, 2017, vol. 18, no.12, pp. 795-801.
- [6] Kasheev N.I., Ponomarev D.M., Podyablonsky F.M. Postroenie testov cifrovih chem. C ispolzovaniem obobshennoi modeli neispravnostei i neprerivnogo podhoda k modelirovaniu (Digital Circuits Test Generation Based on Generalized Malfunction Model and Continuous Simulation Approach). Vestnik Nijegorodskogo Universiteta, 2011, no. 3(2), pp. 72-77 (in Russian).

- [7] Cruz, A.M., Fernandez, R.B., Lozano, H.M., Ramirez Salinas, M.A., Vila Vargas, L.A. Automated Functional Test Generation for Digital Systems Through a Compact Binary Differential Evolution Algorithm // Journal of Electronic Testing-Theory and Applications. 2015. V. 31. № 4. P. 361-380.
- [8] Ivannikov A.D. Analiz metodov dekompozicii zadachi otladki proektov cifrovih system (Decomposition Methods Analisys for Digital System Design Debugging). Informacionnie Technologii, 2016, vol. 22, no. 10, pp. 758-763.
- [9] Stempkovsky F.L., Gavrilov S.V., Glebov A.L. Metodi logicheskogo i logiko-vremennogo analiza cifrovih CMOS VLSI (Logical and Logical-Timing Analisys for Digital CMOS VLSI). Moscow, "Nauka", 2007, 220 p. (in Russian).
- [10] Jasnetski, A., Oyeniran, S. A., Tsertoy, A. High-Level Modeling and Testing of Multiple Control Faults in Digital Systems // IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS). 2016. Paper # 7482445.
- [11] Berezkin A.V., Fedotov A.A., Filippov A.S. Testirovanie cifrovih system, zadannih visokourovnevimi specifikaciyami (Testing of Digital Systems, Defined by High Level Specifications). Nauchno-Tehnicheskie Vedomosti Sankt-Peterburgskogo Gosudarstvennogo Politechnicheskogo Universiteta. Informatika. Telecommunicacii. Upravlenie, 2011, vol. 6-1, no. 138, pp.62-70.
- [12] Jain, S., Govani, P., Poddar, K.B., Lal, A.K., Parmar, R.M. Functional verification of DSP based on-boad VLSI design. International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA). 2016. P. 1-4.
- [13] Gavriliv S.V., Ivanova G.A., Stempkovsky A.L. Teoretikografovaya model slojno-funkcionalnih blokov dlya CMOS tehnologiy s trehmernoy strukturoy transistor (Theoretical Graph Model of Complex Functional Blocks for CMOS Technology with Three Dimension Transistor Structure). Izvestiya UFU. Tehnicheskie Nauki, 2014, no. 7 (156), pp.58-68 (in Russian).
- [14] Jasnetski A., Oyeniran S.A., Tsertoy A. High Level Modeling and Testing Of Multiple Control Faults in Digital Systems // IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS). 2016. Paper # 7482445.
- [15] Ivannikov A.D., Stempkovsky A.L. Matematicheskaya model otladki proektov slojnih cifrovih system i mikrosystem na osnove predstavleniya poslednih v vide semeistva stacionarnih dinamicheskih system (Design Debugging Mathematical Model for Complex Digital and Microsystems on the Basis of Their Representation as a Family of Stationary Dynamic Systems). Problemi Rasrabotki Perspektivnih Mikro- i Nanjelectronnih System (MES), 2014, no. II, pp. 123-128.

Подход к стохастическому тестированию RTL-моделей многоядерных микропроцессоров

Н.А. Гревцев^{1,2}, П.А. Чибисов¹

¹ФГУ ФНЦ НИИСИ РАН, г. Москва, chibisov@cs.niisi.ras.ru

²МФТИ ГУ, г. Долгопрудный

Аннотация — В статье предложен маршрут раннего тестирования моделей многоядерных микропроцессоров без создания отдельных генераторов тестов, направленных исключительно на многоядерное тестирование. В данной работе рассматривается способ адаптации имеющихся средств одноядерного тестирования полнопенный стохастического пол инструмент многоядерного тестирования. Построение многопоточных тестовых программ осуществляется генератором. основное предназначение которого заключается в генерации одноядерных тестовых программ. При этом сам генератор тестов не требует внесения значительных исправлений и доработок. Предложенный метод был успешно применен для тестирования RTL-модели разрабатываемого в ФГУ ФНЦ НИИСИ РАН двухъядерного микропроцессора с SMP.

Ключевые слова — функциональная верификация, многоядерные микропроцессоры с общей памятью, когерентность кэш-памяти, стохастическое тестирование, генерация псевдослучайных тестов, MOESI-протокол.

I. Введение

верификации Стратегия RTL-модели многоядерного микропроцессора, рассматриваемая в данной работе, состоит в том, чтобы осуществить как можно более полное тестирование проекта на ранних этапах цикла проектирования с использованием уже готовых инструментов тестирования одноядерного микропроцессора. Цель рассматриваемого системного подхода — верификация межъядерных взаимодействий между отдельными блоками подсистемы памяти, и самих блоков в масштабе всей системы непосредственно в процессе разработки модели. Следует подчеркнуть, что генератор одноядерных тестовых программ при этом не требует внесения значительных исправлений и доработок.

Многоядерная архитектура микропроцессора с симметричным доступом к памяти (SMP) позволяет использовать ресурсы дополнительных процессорных ядер для распараллеливания ресурсоемких вычислений или одновременного выполнения нескольких задач. Общий системный контроллер в такой системе обеспечивает доступ к ОЗУ для каждого микропроцессорного ядра, а также подключение периферийных устройств. Пример структурной схемы многоядерной системы на кристалле с архитектурой с симметричным доступом к памяти SMP показан на рис. 1.



Рис. 1. Многоядерная архитектура с симметричным доступом к памяти SMP

Проблема целостности данных (иначе говоря, когерентности данных) может возникнуть в случае одновременного обращения двух или более вычислительных ядер к одним и тем же данным, если одно или более из этих обращений осуществляется на запись. Таким образом, наиболее актуальные данные могут оказаться в кэш-памяти одного из ядер и будут недоступны остальным устройствам в системе. Эта сохранения целостности проблема данных B многоядерных микропроцессорах решается с помощью протокола когерентности, согласно которому обеспечивается синхронизация данных между процессорными ядрами периферийными И устройствами.

Статья является продолжением исследований [1], [2], посвященных стохастическому тестированию моделей микропроцессоров. В последующих разделах статьи приводится описание предлагаемого нового подхода к раннему тестированию RTL-моделей многоядерных микропроцессоров, ключевыми моментами которого можно назвать акцент на раннем этапе проектирования, а также применимость однопоточных средств тестирования.

II. ОСОБЕННОСТИ ТЕСТИРОВАНИЯ ПОДСИСТЕМЫ ПАМЯТИ

При проектировании новых микропроцессоров может потребоваться исследование влияний новых микроархитектурных решений. Для этого создается ряд экспериментальных моделей, на которых оценивается эффективность решений на множестве ключевых критериев, таких как: сложность реализации в HDLколе. временные затраты на реализацию, производительность, размер занимаемой площади на кристалле, энергопотребление. Разные подходы к разработке требуют итерационного создания моделей экспериментальных для оценки работоспособности И производительности принимаемых решений.

При каждом новом изменении работоспособность модели должна быть протестирована за ограниченное время. Например, в процессе разработки потребовалось оценить применимость одного из двух вариантов (инклюзивного и эксклюзивного) взаимодействия между кэш-памятью первого и второго уровня. Инклюзивная организация предполагает дублирование информации, находящейся в L1-кэше и L2-кэше, в то

время как эксклюзивная кэш-память предполагает уникальность информации, находящейся в L1-кэше и L2-кэше. Для оценки оптимальности подобных принимаемых архитектурных решений необходима тестовая система, в которую входят тесты аттестации архитектуры и тесты производительности.

R статье предлагается маршрут раннего тестирования моделей многоядерных микропроцессоров И получение полноценного инструмента многоядерного тестирования за счет алаптании имеющихся средств одноядерного тестирования. Олним из методов тестирования моделей на системном уровне является метод направленного стохастического тестирования. При этом в условиях крайне ограниченных временных рамок и человеческих ресурсов требовалось осуществить тестирование разрабатываемой RTL-модели многоядерного микропроцессора без создания новых инструментов тестирования, направленных исключительно на верификацию протокола когерентности.



Рис. 2. Протокол когерентности кэш памяти MOESI, спроецированный на два ядра

Подсистема памяти современных многоядерных микропроцессоров представляет собой совокупность множества компонентов на одном кристалле: MMU, TLB, кэш-памяти всех уровней и их контроллеры, механизмы поддержки когерентности данных, системный контроллер, буферы предварительной подкачки данных, буферы упорядочивания записи данных. Наличие множества вычислительных ядер повышает комбинаторную сложность тестирования подсистемы памяти.

Протокол когерентности описывает состояние одной строки кэш-памяти относительно аналогичной строки в другом ядре и не затрагивает операции над другими строками кэш-памяти. Под состоянием строки кэш-памяти понимается состояние соответствующего контроллера кэш-памяти. Все состояния можно поделить на основные и промежуточные. Основные состояния определяются подмножеством состояний Modified, Owned, Exclusive, Shared, Invalid (MOESI). Переходы из одного основного состояния в другое в современных протоколах когерентности кэш-памяти происходят не мгновенно, а посредством переходных состояний [3].

Высокая сложность задачи проверки (верификации) корректности протоколов когерентности обуславливается тем, что простой перебор состояний быстро приводит к комбинаторному взрыву. В работе [4] показано, что протокол когерентности можно верифицировать отдельно по его спецификации, применяя метод верификации моделей (model checking). Однако для исследуемых в рамках настоящей статьи моделей стояла задача проверить не только проект и реализацию выбранного протокола когерентности, но и реализацию всей подсистемы памяти в совокупности с прочими микроархитектурными усовершенствованиями, упомянутыми выше.

На рис. 2 показаны все возможные переходы от операций загрузки-сохранения данных из памяти, осуществляемых вычислительными ядрами микропроцессора, на примере двух ядер. Большинство таких операций, приводящих к смене состояний конечного автомата MOESI, отображено на рис. 2. Однако в реальной системе они задействуют множество прочих блоков подсистемы памяти.

Одним из методов тестирования моделей на системном уровне является метод направленного стохастического тестирования, показавший высокую эффективность [2].

III. ПРИМЕНИМОСТЬ ИМЕЮЩИХСЯ ИНСТРУМЕНТОВ

До разработки узконаправленного генератора случайных тестов для тестирования подсистемы памяти многоядерных систем предлагается доработать имеющиеся средства тестирования и создавать отдельные тесты для каждого ядра, а затем запускать их параллельно. При этом ответственность за корректное распределение памяти между ядрами лежит на разработчике шаблона.

Для создания тестового кода для каждого ядра необходим собственный шаблон и один запуск одноядерного генератора псевдослучайных тестов. При построении шаблонов учитывается распределение памяти между ядрами, инструкции обращения к памяти выбираются случайно в пределах заданных ограничений. Пример маршрута стохастического тестирования для верификации двухъядерной системы показан на рис. 3.



Рис. 3. Схема применения одноядерного генератора тестов для верификации многоядерных систем

Достоинства метода:

- возможность сразу начать тестирование, не тратя ресурсы на разработку отдельного генератора,
- масштабируемость: легко расширить подход на любое число ядер (2-16),
- возможность точного задания тестовых ситуаций с разными степенями свободы,
- отсутствие необходимости видоизменять процесс генерации при изменении различных архитектурных решений.

Недостатки:

- необходимость дополнительного контроля над распределением памяти между ядрами со стороны разработчика шаблонов,
- необходимость контроля синхронизации потоков,
- область тестирования ограничена подсистемой памяти.

IV. ПРОЦЕСС ГЕНЕРАЦИИ ТЕСТОВ ДЛЯ МНОГОЯДЕРНЫХ МИКРОПРОЦЕССОРОВ

В данной главе описаны основные этапы тестирования многоядерных подсистем памяти, соответствующие разным уровням завершенности функциональной разработки RTL-модели проектируемого микропроцессора.

А. Рукописные тесты, направленные на протокол

Процесс тестирования начинается с создания простых рукописных тестов на языке ассемблера, направленных на проверку протокола когерентности MOESI. В таких тестах отдельно проверяются все переходы между состояниями MOESI (рис. 2), формализуются и проверяются алгоритмы синхронизации ядер, которые будут использованы при дальнейшем тестировании. На примере таких тестов отлаживается механизм проверки корректности выполнения теста. Для этого используется система сравнения логов эталонного эмулятора и RTL-модели.

В. Рукописные тесты с самопроверкой

Исходная идея заключается в том, что ситуации, связанные с ложным разделением данных между потоками (false sharing) при запуске многопоточных приложений, приводят к потере производительности вычислений. Модификация даже одного байта приводит к обновлению целой строки кэш-памяти, и в этом случае могут возникать ситуации, когда потоки, параллельно выполняющиеся на разных ялрах. поочередно обновляют разные переменные, попадающие в одну и ту же строку кэш-памяти. В таких случаях обновление строки на одном ядре микропроцессора будет приводить к вытеснению соответствующей строки из кэш-памяти другого ядра [5].

В работах [6], [7] также исследуются способы обнаружения ложного совместного использования данных, анализируется влияние на И производительность конфликтов, связанных с использованием одной кэш-линии разными потоками. При проектировании параллельных программ рекомендуется избегать описанных выше конфликтов. Однако для того, чтобы протестировать межъядерные взаимодействия, имеет смысл намеренно (искусственно) создавать задачи, в которых память между ядрами распределена таким образом, что оба потока начинают бороться (итерационно обращаться) за одну кэш-линию. За эталонный результат таких тестов принимается результат той же задачи, реализованный с другой схемой распределения данных между потоками (не требующей пересылок данных между кэш-линиями разных ядер).

Самым простым примером рукописного теста со встроенной самопроверкой, основанного на ложном распределении данных, является классическая задача программирования — сложение двух массивов (рис. 4). Для организации самопроверки массивы складываются двумя различными способами.

Способ 1. Правильное с точки зрения распределения памяти между потоками разделение по кэш-линиям. Каждое ядро осуществляет обработку данных в пределах выделенных ему кэш-линий, например, нулевое ядро обрабатывает четные кэш-линии, первое — нечетные.

Способ 2. Применяется ложное разделение данных. Каждое ядро имеет доступ к различным местам совпадающих кэш-линий, что приводит к постоянным пересылкам данных между ядрами.

По окончании теста массивы, полученные разными способами, подлежат сравнению. Расхождение элементов массивов свидетельствует о потере данных при пересылке между кэш-памятью ядер или о получении доступа к устаревшей копии данных.

С. Псевдослучайная генерация тестов

Для повышения вероятности нахождения редких и труднообнаружимых ошибок, возникающих в результате взаимодействия нескольких инструкций в конвейере, а также в результате множества одновременных запросов в кэш-память, применяется метод стохастического тестирования, заключающийся в генерации псевдослучайных тестов по заданному шаблону инструкций [2].

Ниже будут рассмотрены три ступени псевдослучайного тестирования, разработанные для тестирования RTL-модели на разных стадиях функциональной разработки.



Рис. 4. Сложение массивов двумя способами как пример теста с самопроверкой

D. Изолированные друг от друга линии кэш-памяти

На начальной стадии тестирования RTL-модели многоядерного микропроцессора вычислительные ядра могут быть протестированы совместно, но с минимальным межъядерным обменом. В таком случае шаблоны для генератора псевдослучайных тестов устроены так, что каждое из ядер использует области данных, отображение которых на кэш-память обоих уровней будет являться персональным для каждого ядра. Ниже приведен пример описания физической памяти, задаваемой в настройках генератора тестов.

| Name | Lower | Upper |
|--------------|------------|------------|
| data1_core0, | 0x0020000, | 0x00207FF, |
| data2_core0, | 0x0030000, | 0x00307FF, |
| data1_core1, | 0x0020800, | 0x0020FFF, |
| data2_core1, | 0x0030800, | 0x0030FFF, |

В приведенном примере видно, что области данных для нулевого и первого ядра занимают различные части кэш-памяти первого и второго уровней и не пересекаются между собой.

Данное приближение было сделано для того, чтобы инструкции загрузки и сохранения от разных ядер в пределах одной итерации (между синхронизациями ядер) теста, во-первых, не могли писать данные в одно место, во-вторых, не обращались к совпадающей линии кэш-памяти. Иначе нельзя гарантировать, что обращения к одной области памяти (или одной линии кэш-памяти) от разных ядер на RTL-модели будут происходить в той же последовательности, что и на эталонном эмуляторе. В первом случае возникает неопределенность (race condition), какое из ядер успело первым изменить данные по совпадающему адресу, а во втором случае, даже при разделении памяти между ядрами, данные из разных областей попадают в разные секции одной линии кэш-памяти. Это приводит к невозможности добиться полного совпадения логфайлов RTL-модели и эмулятора.

Е. Перекрестный вторичный запуск

В случае изолированных строк кэш-памяти в конце строки переходят случайное теста все в модифицированное состояние. Это состояние может быть использовано в качестве отправной точки для построения более содержательного теста. Во второй половине задачи тестовый код, выполненный на нулевом ядре, передается на выполнение первому и наоборот. Тем самым гарантируется, что каждая запрошенная строка кэш-памяти этого ядра находится в модифицированном состоянии в кэш-памяти другого ядра. Этот прием дает возможность повысить эффективность тестирования механизмов межъядерных взаимодействий (временные взаимоотношения между ядрами и кэш-памятью ядер) за счёт большего разнообразия тестовых ситуаций. Структура теста, основанного на перекрестном вторичном запуске, с примерами переходов состояний MOESI показана на рис. 5.



Рис. 5. Структура теста, основанного на перекрестном вторичном запуске

Первая половина теста на этой стадии по уровню покрытия конечного автомата протокола MOESI дает такой же результат, как и тесты из предыдущей стадии. Однако при перекрестном запуске (вторая половина теста) пространство достижимых состояний расширяется за счёт переходов MOESI от запросов другого ядра.

F. Структуры данных с чередованием

Из-за приближений, сделанных в предыдущих пунктах, упускается из рассмотрения целый класс потенциальных ошибок, связанных с одновременным обращением обоих ядер к одной области памяти.

Одно из возможных решений проблемы — структуры данных с чередованием (*interleaved memory structure*).

Для того чтобы обеспечить полноту тестирования протокола когерентности кэш-памяти, необходимо реализовать в тестовой системе возможность одновременного (в пределах зоны синхронизации) доступа к одной ячейке кэш-памяти обоими ядрами. Для этого на выбранную область памяти накладывается периодическая маска с размером ячейки меньше размера строки кэш-памяти. При этом нулевое ядро получает доступ ко всем нечетным элементам, а первое — к четным.

Важным следствием такой организации данных в памяти является тот факт, что оба ядра получают доступ к одной строке кэш-памяти одновременно и на запись, и на чтение, не нарушая целостности данных (обращения происходят в разные байты).

Благодаря такому разделению данных внутри любой отдельно взятой кэш-линии, каждая общая область, в которую разрешено писать более чем одному ядру, будет содержать детерминированные значения, и, таким образом, в любой момент времени все данные этой области являются предсказуемыми. Чтобы протестировать оставшуюся непокрытой группу ситуаций, связанных с одновременным (в пределах нескольких тактов) доступом обоих ядер к одному и тому же адресу в памяти, создаются отдельные области «только на чтение» и «только на запись». Операции загрузки и сохранения в пределах этих областей происходят бесконтрольно из-за ситуаций, связанных с несинхронностью потоков обращений к памяти, работающих с общими данными (*race condition*). Такие обращения не подлежат проверке, так как направлены исключительно на поиск зависаний конвейера (*deadlock*).



Рис. 6. Структурная схема распределения памяти между ядрами с использованием чередующейся маски

В приведенном примере (рис. 6) оба ядра имеют общие области памяти, но при этом одно ядро будет иметь доступ только к четным двойным словам, а другое — к нечетным. Таким образом, тесты позволяют охватить гораздо большее число сложно достижимых ситуаций, потенциально приводящих к ошибкам в подсистеме памяти. Пример распределения адресов в тестовых шаблонах может выглядеть следующим образом:

валидные адреса для ядра 0: XXXX0-XXXX7,

валидные адреса для ядра 1: XXXX8-XXXXF.

При использовании предложенной модели распределения памяти необходимо исключить из сравнения с эталонным эмулятором лог-файлы кэшпамятей из-за невозможности гарантировать одинаковую очередность обращений. При этом заведомо корректное состояние памяти и регистров общего назначения позволяет с высокой вероятностью находить ошибки RTL-модели.

В усовершенствованной системе тестирования одновременно применяются все вышеперечисленные методы, а также в ходе выполнения теста несколько раз может меняться периодичность разбиения interleaved-областей.

V. ОЦЕНКА КАЧЕСТВА СОЗДАВАЕМЫХ ТЕСТОВ

Для оценки качества создаваемых псевдослучайных тестов была задана обобщенная метрика основанная функционального покрытия, на пространстве состояний, построенном на комбинации тестовых ситуаций, которые задаются такими событиями, как: тип операции, попадание или промах в любого кэш-память уровня, замешение модифицированных строк в кэш-памятях, вилы переходов конечного автомата MOESI. На основании эвристического анализа вводится понятие условного тестового покрытия и определяется его максимальное значение, которое соответствует 100% тестового покрытия.

Введение метрики функционального покрытия позволяет количественно измерить степень завершенности работ по тестированию, а также оценить качество создаваемых генератором тестов.



Рис. 7. Графическое отображение функционального покрытия для трех видов тестов

На рис. 7 показан рост функционального покрытия в ходе выполнения теста для трех описанных выше техник построения тестов. Резкий скачок в середине тестов (50% по оси абсцисс) обусловлен тем, что в этой точке происходит перекрестный вторичный перезапуск.

VI. ЗАКЛЮЧЕНИЕ

Предложенный в статье подход первоначально рассматривался как этап тестирования RTL-модели проектируемого микропроцессора на ранних стадиях, однако возможности такого подхода также представляют интерес и для тестирования модели многоядерного микропроцессора и на поздних этапах завершенности ее функциональной разработки.

Процесс тестирования начинается с создания простых тестов, направленных на проверку протокола когерентности MOESI. Для повышения вероятности нахождения редких и трудно обнаруживаемых ошибок в подсистеме памяти многоядерного микропроцессора предложен метод стохастического тестирования, основанный на использовании развитых одноядерных инструментов псевдослучайной генерации тестов, адаптированных под модели микропроцессоров с произвольным числом вычислительных ядер.

ЛИТЕРАТУРА

- [1] Хисамбеев И.Ш., Чибисов П.А. Об одном методе построения метрик функционального покрытия в тестировании микропроцессоров // Проблемы разработки перспективных микро- и наноэлектронных систем - 2014. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2014. Часть2. С. 63-68.
- [2] Гревцев Н.А., Хисамбеев И.Ш., Чибисов П.А. Исследование способов повышения эффективности стохастического тестирования моделей микропроцессоров // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №2. С. 8-15.
- [3] Sorin, D. A Primer on Memory Consistency and Cache Coherence / Morgan & Claypool, 2012. - 210 p.
- [4] Камкин А.С., Буренков В.С. Метод масштабируемой верификации PROMELA-моделей протоколов когерентности кэш-памяти // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №2. С. 54-60.
- [5] Велесевич Е. А. Обнаружение и оценка количества промахов когерентности на основе вероятностной модели, Труды ИСП РАН, 27:4 (2015), 39-48
- [6] T. Liu et al., "PREDATOR: Predictive false sharing detection", PPoPP 2014.
- [7] Tongping Liu, Xu Liu, Cheetah: detecting false sharing efficiently and effectively, Proceedings of the 2016 International Symposium on Code Generation and Optimization, March 12-18, 2016.

A Practical Approach to Verification of Multicore Microprocessor Models

N.A. Grevcev^{1,2}, P.A. Chibisov¹

¹Scientific Research Institute of System Analysis (SRISA RAS), chibisov@cs.niisi.ras.ru

$^{2}MIPT$

Abstract — in the paper, the early stage of verification technology for multicore processor models testing is proposed. We demonstrate the applicability of the single core verification method extension where a creating new multicore test generator is not required. The solution scheme deals with the adaptation method of some available single core stochastic testing approaches to a fully functional multicore testing tool.

The proposed technique has been successfully applied to test RTL-model of dual-core microprocessor with SMP developed in SRISA. The discussed approach was initially considered to be a first stage of RTL-model testing, but the possibilities of the approach are also of interest for testing the model at the later stages of its design and functional maturity.

The testing process begins by creating simple random tests that check the MOESI coherence protocol. New advanced random testing method based on the usage of proposed interleaved memory structures is developed to increase the probability of finding rare and hard to detect bugs in the memory subsystem.

The great advantage of the proposed memory allocation structure is that both cores gain access to the same cache-line simultaneously for read and write. Despite this accessibility, the methodology avoids losses of data consistency due to cores access to different bytes. Furthermore, each common area that is allowed to write to more than one core will contain deterministic values at every time.

The proposed approach aims to detect failures in cache coherence protocol, memory subsystem and memory buffers. Also, this testing system helps to find machine state errors that lead to dead lock.

Keywords — functional verification, multicore, stochastic testing, pseudorandom tests generation, memory subsystem, SMP, MOESI, cache coherence, pre-silicon verification.

References

- [1] KHisambeev I.SH., CHibisov P.A. Ob odnom metode postroeniya metrik funktsional'nogo pokrytiya v testirovanii mikroprotsessorov (On a method for constructing functional coverage metrics in microprocessor testing) // Problemy razrabotki perspektivnykh mikro- i nanoehlektronnykh sistem. Sbornik trudov / M.: IPPM RAN, 2014. S. 63-68.
- [2] Grevtsev N.A., KHisambeev I.SH., CHibisov P.A. Issledovanie sposobov povysheniya ehffektivnosti stokhasticheskogo testirovaniya modelej mikroprotsessorov (Methods to improve efficiency of microprocessor model stochastic tests) // Problemy razrabotki perspektivnykh mikro- i nanoehlektronnykh sistem (MES'2016).
- [3] Sorin, D. A Primer on Memory Consistency and Cache Coherence / Morgan & Claypool, 2012. - 210 p.
- [4] Kamkin A.S., Burenkov V.S. Metod masshtabiruemoj verifikacii PROMELA-modelej protokolov kogerentnosti ke`sh-pamyati (A method for scalable verification of PROMELA models of cache coherence protocols) // Problemy` razrabotki perspektivny`x mikro- i nanoe`lektronny`x sistem (MES). 2016. №2. S. 54-60
- [5] Velesevich E. A. Obnaruzhenie i ocenka kolichestva promaxov kogerentnosti na osnove veroyatnostnoj modeli (Number of coherence misses detection based on the probabilistic model), Trudy` ISP RAN, 27:4 (2015), 39-48
- [6] T. Liu et al., "PREDATOR: Predictive false sharing detection", PPoPP 2014.
- [7] Tongping Liu , Xu Liu, Cheetah: detecting false sharing efficiently and effectively, Proceedings of the 2016 International Symposium on Code Generation and Optimization, March 12-18, 2016.

Метод снижения размерности обучающих наборов при построении нейроморфного справочника неисправностей для аналоговых интегральных схем

С. Г. Мосин

Казанский федеральный университет, smosin@ieee.org

Аннотация — Методы машинного обучения активно используются для построения нейроморфных справочников неисправностей (НСН), которые обеспечивают диагностику неисправностей аналоговых и смешанных интегральных схем в ассоциативном режиме. Многие проблемы обучения нейронной сети, связанные с большим объемом исходных данных, могут быть решены путем уменьшения размеров обучающих наборов и использования в них только существенных характеристик. В статье предложен метод, основанный на вычислении энтропии, для выбора существенных характеристик обучающего набора, разработан соответствующий алгоритм. Представлены результаты экспериментальных исследований для аналогового фильтра, которые демонстрируют высокую эффективность предлагаемого метода: снижение времени обучения нейронной сети в 192 раза, покрытие полученным НСН до 95.0% катастрофических и до 84.81% параметрических неисправностей при диагностике.

Ключевые слова — диагностика неисправностей, нейроморфный справочник неисправностей, аналоговые ИС, энтропия, машинное обучение, автоматизация проектирования.

I. Введение

Диагностика неисправностей аналоговых И смешанных интегральных схем (ИС) остается важным этапом производственного цикла, обеспечивающим качество надежность высокое И изделий микроэлектроники [1-2]. Рост функциональной и структурной сложности современных аналоговых и смешанных ИС определяет для них увеличение сложности диагностики неисправностей. Методы, основанные на моделировании неисправностей, в отличие от методов, основанных на контроле соответствия спецификации, обеспечивают решение не только задачи тестирования, но и диагностики неисправностей [3]. Моделирование поведения исправной схемы и схемы с возможными неисправностями с использованием соответствующих моделей требует серьезных вычислительных И временных затрат [4]. В результате моделирования накапливаются большие объемы информации о поведении схемы.

Интенсивный рост производительности современных вычислительных систем определяет бурное развитие приложений, основанных на методах машинного обучения (МО) и интеллектуального анализа данных, в том числе для задачи диагностики неисправностей в аналоговых и смешанных ИС [5]. Использование MO позволило перейти от традиционных справочников неисправностей (СН) в виде параметрических таблиц к нейросетевым, работающим в ассоциативном режиме. В этом случае снижается время обнаружения существенно И диагностики потенциальной неисправности, но появляются традиционные проблемы, связанные с обучением нейронной сети (НС) [6-9]:

- выбор архитектуры нейронной сети (вид HC, количество слоев и нейронов, используемая передаточная функция и др.);

- выбор обучающих наборов;

- снижение размерности обучающих наборов и др.

В данной работе предложен основанный на вычислении энтропии метод снижения размерности обучающего набора, а, следовательно, и снижения нейронов во входном слое HC.

II. АЛГОРИТМ ВЫБОРА СУЩЕСТВЕННЫХ ХАРАКТЕРИСТИК НА ОСНОВЕ ВЫЧИСЛЕНИЯ ЭНТРОПИИ

Исходные данные для построения нейросетевого справочника неисправностей накапливаются в ходе моделирования во временной области исправного и неисправного поведения схемы для заданного набора неисправностей. Для учета влияния допусков на поведение схемы каждое состояние моделируют многократно с применением метода Монте-Карло. Временные отклики схемы в тестовых узлах нецелесообразно напрямую использовать для обучения HC. Как правило, предварительно выполняют спектральное разложение непрерывного сигнала во временной области в дискретный сигнал в частотной области, используя Фурье или вейвлет преобразования. В итоге для каждого і-го отклика формируют вектор коэффициентов $\mathbf{x}_i = \begin{bmatrix} x_{ij} \end{bmatrix}$, $i, j \in \mathbf{Z}_+$, $x_{ij} \in \mathbf{R}$, i – количество откликов схемы, *j* – количество

коэффициентов для каждого отклика. Поведение исправной и неисправной схемы можно представить матрицей $\mathbf{X} = [\mathbf{x}_i]$, $\mathbf{X} \in \mathfrak{R}_{i \times j}$, которая состоит из подматриц $\mathbf{X}^k \in \mathfrak{R}_{m_k \times j}$ для каждого рассматриваемого k-го состояния схемы, $m_k \in \mathbf{Z}_+$ определяет число итераций методом Монте-Карло для k-го состояния.

В общем случае поведение схемы для различных состояний может совпадать, т.е.

$$\exists (i,j), \mathbf{X}^{i} \cap \mathbf{X}^{j} \neq \emptyset, i \neq j, i, j \in \mathbf{Z}_{+},$$
(1)

тогда диагностика будет возможна с точностью до двойственной группы (AG).

Для однозначной диагностики важно выполнение следующего условия

$$\mathbf{X}_{i}^{p} \neq \mathbf{X}_{j}^{r}, \ \forall i, j, p, r, \ i \in \left[1..m_{p}\right], \ j \in \left[1..m_{r}\right], \ p \neq r \ . \ (2)$$

Не все коэффициенты в векторе \mathbf{x}_i значимы для решения задачи диагностики. Для снижения неоднозначности, а также сложности обучения нейронной сети с использованием \mathbf{x}_i важно выбрать только существенные коэффициенты, влияющие на различимость различных состояний схемы. Для выбора соответствующих коэффициентов предложено использовать энтропию.

Для расчета энтропии необходимо преобразовать матрицу **X** согласно следующим правилам:

1. Для каждого столбца *j* близкие значения коэффициентов включить в двойственную группу \mathbf{AG}_{j}^{k} , $k = 1..n_{j}$, n_{j} – количество двойственных групп в столбце *j*.

2. Заменить в каждом столбце значения коэффициентов номерами двойственных групп (k), к которым они принадлежат.

Таким образом, двойственное группирование $AG: \mathbf{X} \rightarrow \mathbf{S}$ приводит к построению матрицы $\mathbf{S} = \begin{bmatrix} s_{ii} \end{bmatrix}$, $s_{ii} \in \mathbf{Z}_+$ с номерами двойственных групп.

Пусть N_{pr} , $p \in [1..k]$ – количество элементов в двойственной группе \mathbf{AG}_r^p . Вероятность появления любого состояния из двойственной группы \mathbf{AG}_r^p равна отношению N_{pr} / m_i , m_i – количество строк в матрице **S**. Тогда энтропия для любого *j*-го столбца матрицы **S** вычисляется согласно выражению (3).

Количество строк в матрице **S** есть фиксированная величина, поэтому объем информации, важной для диагностики неисправностей с применением *j*-го коэффициента, становится максимальным при минимизации коэффициента энтропии (4)-(5).

$$E_{j} = -\left[\frac{N_{1j}}{m_{i}}\log\left(\frac{N_{1j}}{m_{i}}\right) + \frac{N_{2j}}{m_{i}}\log\left(\frac{N_{2j}}{m_{i}}\right) + \dots$$

$$\dots + \frac{N_{kj}}{m_{i}}\log\left(\frac{N_{kj}}{m_{i}}\right)\right] = \log(m_{i}) - \frac{1}{m_{i}}\sum_{i=1}^{k}N_{ij}\log(N_{ij}) .$$
(3)

$$\arg\max_{j} \left(E_{j} \right) = \arg\min_{j} \left(ER_{j} \right), \tag{4}$$

$$ER_j = \sum_{i=1}^k N_{ij} \log(N_{ij}).$$
⁽⁵⁾

III. ОПИСАНИЕ АЛГОРИТМА

Алгоритм выбора существенных коэффициентов включает следующие шаги:

1. Сформировать матрицу **S** по известным значениям коэффициентов выходных откликов схемы из **X**.

2. Вычислить число элементов в каждой двойственной группе для каждого *j*-го столбца матрицы **S**.

3. Рассчитать коэффициент энтропии ER(j).

4. Добавить *j*-й коэффициент с минимальным значением *ER*(*j*) во множество выбранных ранее коэффициентов и исключить соответствующий столбец из дальнейших расчетов.

5. Переформировать матрицу **S** в соответствии с порядком двойственных групп выбранного коэффициента, а также удалить из нее строки, мощность двойственной группы для которых равна 1 для данного коэффициента. В случае k двойственных групп для выбранного j-го столбца матрицы **S** формируют k подматриц **S**^{*i*}, i = 1.k.

6. Рассчитать коэффициент ER(j) для оставшихся коэффициентов с учетом присутствия двойственных групп в каждой из подматриц **S**^{*i*}

$$ER(j) = \sum_{i=1}^{k} ER^{i}(j), \qquad (6)$$

где $ER^{i}(j)$ - коэффициент энтропии, рассчитанный для подматрицы S^{i} .

7. Если коэффициент ER(j) оказывается равным нулю, для всех *j* или ER(j) принимает то же значение, что и ранее, для всех *j*, то процесс прекращается. В противном случае необходимо перейти к шагу 4.

IV. Экспериментальная часть

Исследование предлагаемого метода выполнено для контрольной схемы аналогового фильтра Саллена-Ки (рис. 1). В качестве тестового сигнала используется синусоидальный сигнал с амплитудой 1 В и частотой 72 Гц. Допуски на параметры внутренних компонентов схемы определены равными 10 % для резисторов и конденсаторов. Список рассмотренных одиночных неисправностей включает 14 катастрофических и 14 параметрических неисправностей. Катастрофические неисправности представлены обрывом цепи (ОЦ) и коротким замыканием (КЗ) для каждого компонента схемы. Параметрические неисправности для каждого компонента задаются отклонениями ± 50% от его номинального значения.



R1 = R3 = RB = 10 кОм, R2 = 20 кОм, RA = 5 кОм, C1 = C2 = 220 нФ

Рис. 1. Схема аналогового фильтра Саллена-Ки

Моделирование схемы фильтра для 29 состояний (одно исправное и 28 неисправных) выполнено во временной области с использованием метода Монте-Карло. Для моделирования схем использован пакет PSpice CAПР CADENCE. Количество итераций для исправного состояния составляет 20 000, а для каждого неисправного состояния – 5 000. Выходные отклики измеряются в течение одного периода относительно тестового узла 4 через 80 мс с момента запуска теста (по завершении переходных процессов).

Вейвлет Добеши четвертого порядка (db4) используется для преобразования выходных откликов из временной области в частотную. В результате для каждого сигнала формируется вектор коэффициентов длиной 148 элементов. В обучающее множество включены 600 векторов из 20 000 для исправного состояния и 200 векторов из 5 000 для каждого неисправного состояния. В результате обучающий набор представлен матрицей коэффициентов вейвлетпреобразования размерностью 6 200 на 148 элементов. Отклики, не включенные в обучающий набор, образуют тестовый набор.

Трехслойная нейронная сеть использована для построения нейроморфного справочника неисправностей. Количество нейронов во входном слое равно количеству коэффициентов в строке матрицы обучающих наборов. Выходной слой состоит из 29 нейронов для представления рассматриваемых состояний схемы относительно соответствующего отдельного выходного нейрона. Количество нейронов во входном слое для исходной матрицы равно 148, после применения метода выбраны только 19 существенных коэффициентов, что позволило сократить размерность обучающего набора в 7.8 раз.

Пакет инструментов Neural Network программы математических расчетов МАТLAВ использован для построения и обучения нейронной сети на вычислительной системе с процессором Intel® Core ™ i7-4770 CPU @ 3.40 ГГц и оперативной памятью 8 ГБ.

Таблица 1 Статистика покрытия отдельных неисправностей

| | | | - | | | _ | | | | |
|---|----------------------|-------------------------------|---------|------------------------------------|-------|-------------------------|------|-----------------------------------|-------|-------|
| Номер НСН | | | | 1 | | 2 | | | | |
| Архитектура НСН¹ | | | 148 : 2 | 20 : 29 | | 19 : 20 : 29 | | | | |
| Время одной эпохи обу- чения, сек. Состояние схемы Исправное | | μ² | | σ ³ | | μ | | σ | | |
| | | 2328.95 FC _{test} | | 30.04 FC _{diag} | | 12.11 FC test | | 1.53 FC _{diag} | | |
| | | | | | | | | | | μ |
| | | 92.09 | 0.85 | - | - | 90.46 | 1.08 | - | - | |
| | | | R1_K3 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | 99.99 |
| | R₁_ОЦ | 100.0 | 0.0 | 99.77 | 0.30 | 95.35 | 1.84 | 77.41 | 10.05 | |
| | R ₂ _K3 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | |
| NI | R₂_ОЦ | 100.0 | 0.0 | 99.60 | 0.89 | 100.0 | 0.0 | 99.91 | 0.05 | |
| BHOC | R ₃ _K3 | 99.95 | 0.07 | 95.83 | 0.65 | 99.97 | 0.04 | 94.38 | 1.10 | |
| астрофические неиспра | R₃_ОЦ | 100.0 | 0.0 | 99.93 | 0.15 | 100.0 | 0.0 | 99.70 | 0.24 | |
| | R _A _K3 | 100.0 | 0.0 | 99.36 | 0.12 | 99.99 | 0.03 | 99.63 | 0.29 | |
| | R₄_ОЦ | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | |
| | R _B _K3 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | |
| | R _₿ _ОЦ | 100.0 | 0.0 | 99.50 | 0.07 | 100.0 | 0.0 | 95.78 | 0.29 | |
| Кат | C1_K3 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | |
| | С1_ОЦ | 99.84 | 0.20 | 95.70 | 1.86 | 99.91 | 0.12 | 87.19 | 5.26 | |
| | C2_K3 | 100.0 | 0.0 | 58.92 | 0.61 | 100.0 | 0.0 | 75.98 | 13.87 | |
| | С2_ОЦ | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | |
| | R1_+50% | 80.21 | 0.84 | 57.23 | 14.51 | 81.69 | 1.43 | 61.53 | 16.22 | |
| | R150% | 96.09 | 0.31 | 91.96 | 0.54 | 95.93 | 0.75 | 91.14 | 1.74 | |
| | R ₂ +50% | 98.10 | 0.20 | 86.58 | 4.05 | 97.70 | 0.56 | 86.02 | 4.72 | |
| СТИ | R ₂ _50% | 100.0 | 0.0 | 86.58 | 10.18 | 100.0 | 0.0 | 94.81 | 6.85 | |
| авно | R ₃ _+50% | 92.88 | 0.36 | 85.80 | 0.83 | 93.90 | 0.61 | 87.25 | 3.83 | |
| испр | R ₃ 50% | 99.93 | 0.13 | 71.12 | 6.01 | 99.96 | 0.05 | 67.89 | 13.82 | |
| Ie Hei | R _A _+50% | 96.93 | 0.11 | 74.17 | 12.89 | 97.61 | 0.50 | 87.19 | 5.29 | |
| Teck | R _A 50% | 99.82 | 0.21 | 87.68 | 1.38 | 99.98 | 0.03 | 80.75 | 3.89 | |
| ыцте | R _B _+50% | 95.44 | 3.06 | 66.82 | 22.98 | 98.33 | 0.84 | 80.22 | 4.03 | |
| раме | R_{B} –50% | 100.0 | 0.0 | 96.73 | 1.32 | 100.0 | 0.0 | 94.51 | 1.55 | |
| Ē | C1_+50% | 99.79 | 0.05 | 95.04 | 0.24 | 99.71 | 0.26 | 88.90 | 3.08 | |
| | C150% | 99.98 | 0.02 | 96.87 | 1.07 | 99.99 | 0.03 | 93.02 | 2.26 | |
| | C ₂ _+50% | 94.29 | 0.63 | 73.42 | 7.57 | 95.76 | 1.01 | 80.06 | 6.50 | |
| | C ₂ _50% | 100.0 | 0.0 | 95.95 | 4.91 | 100.0 | 0.0 | 94.00 | 5.99 | |

Количество нейронов в первом: втором: третьем слое НС;

² µ – математическое ожидание;

σ – среднеквадратическое отклонение.
Проведено многократное обучение НСН с использованием полного и сокращенного обучающего Статистические результаты набора. покрытия отдельных неисправностей для тестового набора, полученные на двух архитектурах НСН, показаны в Таблице 1. значений Анализ покрываемости отдельных неисправностей для режимов тестирования (FC_{test}) и диагностики (FC_{diag}) демонстрирует адекватность перехода от архитектуры 1 к архитектуре 2.

Обнаружение исправного состояния фильтра на уровне порядка 90.5 % связано с подобием поведения исправно функционирующей схемы и схемы с несколькими типами параметрических неисправностей. Увеличение количества откликов в обучающем наборе для исправной схемы улучшит качество диагностики для этого состояния, но в то же время приведет к увеличению ошибки второго рода, когда для соответствующих параметрических неисправностей схема будет признаваться исправной.

Общие статистические значения покрываемости тестирования неисправностей для режимов диагностики, обеспеченные НСН с различными рассмотренными архитектурами, представлены в таблице 2. Применение предложенного метола обеспечило уменьшение размера обучающего набора за счет выбора существенных коэффициентов для диагностики рассматриваемых в схеме фильтра неисправностей в 7.8 раз и сокращение времени обучения в 192 раза, демонстрируя при этом высокий уровень покрытия неисправностей. Результирующий НСН обеспечивает до 100 % покрытия отдельных неисправностей и до 99.7 % полного покрытия неисправностей в схеме фильтра.

Таблица 2

| Номер | α | | FCall | | FC _{cat} | | FC _{par} | |
|--------------|------|------|-------|------|--------------------------|------|--------------------------|------|
| нсн | μ | σ | μ | ь | μ | υ | μ | ь |
| Тестирование | | | | | | | | |
| 1 | 7,91 | 0,85 | 98,33 | 0,16 | 99,99 | 0,02 | 96,68 | 0,32 |
| 2 | 9,54 | 1,08 | 98,42 | 0,17 | 99,66 | 0,13 | 97,18 | 0,27 |
| Диагностика | | | | | | | | |
| 1 | - | - | 89,40 | 0,54 | 95,51 | 1,93 | 83,28 | 2,19 |
| 2 | - | - | 89,90 | 1,01 | 95,00 | 1,32 | 84,81 | 2,07 |

Статистика общего покрытия неисправностей

 α – величина ошибки первого рода для исправного состояния; FC_{all} – покрываемость всех типов рассмотренных неисправностей; FC_{cat} – покрываемость катастрофических неисправностей; FC_{par} – покрываемость параметрических неисправностей.

V. Заключение

Современные вычислительные системы позволяют провести исчерпывающее моделирование поведения аналоговых и смешанных интегральных схем с учетом как исправного состояния, так и потенциальных неисправных состояний с учетом допусков компонентов и использования метода статистических испытаний. Всестороннее моделирование поведения схемы приводит к накоплению больших объемов ланных. которые могут быть обработаны с использованием методов машинного обучения и интеллектуального анализа данных. Методы машинного обучения активно используются для нейроморфных справочников построения неисправностей, которые обеспечивают диагностику неисправностей аналоговых и смешанных ИС в ассоциативном режиме. Многие проблемы обучения нейронной сети, связанные с большим объемом исходных данных, могут быть решены путем уменьшения размера обучающих наборов и использования только существенных характеристик.

Предлагаемый метод на основе энтропии позволяет уменьшить количество коэффициентов во входном обучающем наборе. Экспериментальные результаты подтвердили эффективность предлагаемого метода: сокращение обучающего набора в 7.8 раза привело к сокращению времени обучения более чем в 192 раза. НСН, полученный в результате, обеспечивает решение тестирования, залач. как так и диагностики неисправностей в ассоциативном режиме, что позволяет существенно сократить время реализации данных процедур для партии ИС. Экспериментальные исследования демонстрируют высокий уровень покрытия неисправностей: при тестировании для катастрофических неисправностей до 99.66% и для параметрических неисправностей до 97.18%; а при диагностике до 95.0% и 84.81% для катастрофических и параметрических неисправностей соответственно. Предлагаемый метод может быть интегрирован в маршрут тестопригодного проектирования аналоговых и смешанных ИС.

Работа выполнена за счет средств субсидии, выделенной в рамках государственной поддержки Казанского (Приволжского) федерального университета в целях повышения его конкурентоспособности среди ведущих мировых научно-образовательных центров.

ЛИТЕРАТУРА

- [1] Емашов А.В. Задачи построения алгоритмов поиска неисправностей аналоговых промышленных объектов // Системы. Методы. Технологии, 2017, № 1(33), с. 85-90.
- [2] Воловикова Е.В., Увайсов С.У. Диагностика аналоговых схем с учетом тепловых режимов электро- и радиоэлементов // Качество. Инновации. Образование, 2009, № 3(46), с. 23-29.
- [3] Bilski P., Wojciechowski J. Automatic parametric fault detection in complex analog systems based on a method of minimum node selection // International Journal of Applied Mathematics and Computer Science, 2016, vol. 26, No. 3, pp. 655–668
- [4] Xu Q., Cui S., Xu C., Han L. The Research on Fault Model in Nonlinear Analog and Mixed-Signal Circuits // Procedia Engineering, 2012, vol. 29, pp. 4152-4156.
- [5] Bilski P., Wojciechowski J. Artificial intelligence methods in diagnostics of analog systems // International Journal of Applied Mathematics and Computer Science, 2014, vol. 24, No. 2, pp. 271–282.

- [6] Мосин С.Г. Тестирование аналоговых схем с использованием нейросетевого сигнатурного анализатора // Вестник информационных и компьютерных технологий, 2012, № 10, с. 3-8.
- [7] Rutkowski J., Grzechca D. Use of Artificial Intelligence Techniques to Fault Diagnosis in Analog Systems // Proc. 2nd European Computing Conference (ECC'08), 2008, pp. 267-274.
- [8] Mosin S. Quality improvement of analog Circuits Fault Diagnosis based on ANN using clusterization as preprocessing // Proc. of IEEE East-West Design and Test Symposium (EWDTS'2015), 2015. pp. 96-99.
- [9] Mosin S. A technique of analog circuits testing and diagnosis based on neuromorphic classifier // Advances in Intelligent Systems and Computing, 2016, vol. 425, pp. 381-393.

Method for Reducing the Dimension of Training Sets at Constructing Neuromorphic Fault Dictionary for Analog Integrated Circuits

S.G. Mosin

Kazan Federal University, smosin@ieee.org

Abstract - Machine learning methods are actively used for the construction of neuromorphic fault dictionaries (NFD), which provide diagnostics of faults in analog and mixed-signal integrated circuits in the associative mode. Many problems of a neural network training associated with a large amount of raw data can be solved by reducing the dimension of training sets and using only significant characteristics for training purpose.

Entropy-based method is proposed for selecting the essential characteristics of a training set. The algorithm implementing the proposed method is considered and used for development of corresponding software.

The study of the proposed method is performed for the benchmark circuit of the Sallen-Key analog filter. The results of experimental studies demonstrate the high efficiency of the proposed method.

The application of the proposed method has provided reducing the dimension of training set by 7.8 times by selecting the sufficient coefficients and reducing the training time by 192 times, while demonstrating high level of fault coverage. The resulting NFD provides coverage up to 100% of individual faults and up to 99.7% of the overall fault coverage for the filter.

The proposed method can be integrated into the design-fortestability flow for analog and mixed-signal ICs.

Keywords — fault diagnostics, neuromorphic fault dictionary, analog integrated circuits, entropy, machine learning, design automation.

ACKNOWLEDGEMENT

The work is supported by the Russian Government Program for Competitive Growth of Kazan Federal University.

REFERENCES

- [1] Emashov A.V. The problems of constructing algorithms for troubleshooting analog industrial objects // Sistemy. Metody. Tekhnologii, 2017, № 1(33), pp. 85-90. (In Russian)
- [2] Volovikova E.V., Uvaysov S.U. Diagnostics of analog circuits taking into account thermal conditions of electroand radio elements // Kachestvo. Innovatsii. Obrazovanie, 2009, № 3(46), pp. 23-29. (In Russian)
- [3] Bilski P., Wojciechowski J. Automatic parametric fault detection in complex analog systems based on a method of minimum node selection // International Journal of Applied Mathematics and Computer Science, 2016, vol. 26, No. 3, pp. 655–668
- [4] Xu Q., Cui S., Xu C., Han L. The Research on Fault Model in Nonlinear Analog and Mixed-Signal Circuits // Procedia Engineering, 2012, vol. 29, pp. 4152-4156.
- [5] Bilski P., Wojciechowski J. Artificial intelligence methods in diagnostics of analog systems // International Journal of Applied Mathematics and Computer Science, 2014, vol. 24, No. 2, pp. 271–282.
- [6] Mosin S.G. A testing of analog circuits using neural network signature analyzer // Vestnik komp'iuternykh I informatsionnykh tekhnologii. 2012, vol. 10. pp. 3–8. (In Russian)
- [7] Rutkowski J., Grzechca D. Use of Artificial Intelligence Techniques to Fault Diagnosis in Analog Systems // Proc. 2nd European Computing Conference (ECC'08), 2008, pp. 267-274.
- [8] Mosin S. Quality improvement of analog Circuits Fault Diagnosis based on ANN using clusterization as preprocessing // Proc. of IEEE East-West Design and Test Symposium (EWDTS'2015), 2015. pp. 96-99.
- [9] Mosin S. A technique of analog circuits testing and diagnosis based on neuromorphic classifier // Advances in Intelligent Systems and Computing, 2016, vol. 425, pp. 381-393.

Метод дублирования триггеров в средствах тестирования с компрессией

М.С. Ладнушкин

НИИ системных исследований РАН, Mocквa, maximsl@gmail.com

Аннотация — Предложен метод сокращения времени тестирования неисправностей цифровой СБИС за счёт дублирования отдельных функциональных триггеров. обусловлено Сокращение времени тестирования тестируемости сигналов, а также **увеличением** снижением взаимных конфликтов неисправностей в логических путях СБИС. Предложен алгоритм отбора триггеров для дублирования на основе поиска логических путей с наибольшим числом источников сигналов, который был использован при проектировании встроенных средств тестирования ряда заказных блоков и систем-на-кристалле. Результаты показали снижение времени тестирования в среднем на 14,4% при аппаратурных затратах, не превышающих 1,2% общей площади СБИС.

Ключевые слова — тестирование, отбраковка микросхем, дублирование триггеров, компрессия тестовых сигналов, моделирование.

I. Введение

Внедрение дополнительных тестовых структур контроля и наблюдения внутренних узлов в СБИС позволяет увеличить тестовое покрытие и сократить время тестирования. Создание тестового режима работы, в котором триггерная подсистема СБИС используется в качестве сдвигового регистра (скансхемы) [1], позволяет полностью контролировать и наблюдать состояние всех триггеров СБИС и использовать их для тестирования комбинационной части СБИС.

Объемы тестовых данных, необходимых для тестирования микросхем, растут, и встраиваемые средства сжатия [2], [3] не всегда позволяют сократить время тестирования до приемлемых величин ввиду многих причин, включая рост длин логических путей, увеличение количества взаимных конфликтов неисправностей, а также необходимость снижения уровня абстракции при создании тестовых последовательностей для СБИС с уровня логических элементов до уровня транзисторов [4], [5].

С ростом длин логических комбинационных путей ростом количества сходящихся разветвлений И снижается наблюдаемость и контролируемость узлов этих путей в режиме тестирования [6], [7], [8]. Для решения данной проблемы в определённые узлы СБИС встраиваются дополнительные тестовые схемы тестовые точки контроля и обзора – дополнительные триггеры с управляющей логикой, позволяющие увеличить наблюдаемость И контролируемость отдельных узлов комбинационной подсистемы СБИС [9].

Установка тестовых точек приводит к сокращению времени тестирования до 35% [10], [11]. Современный метод установки тестовых точек, снижающий количество взаимных конфликтов неисправностей типа «залипание», позволяет сократить время тестирования в среднем в 2,2 раза для скан-схем с компрессией за счёт увеличения количества неисправностей, тестируемых каждым тестовым вектором [12]. Однако установка тестовых точек уменьшает ресурс трассировки, увеличивает задержки распространения сигналов и увеличивает площадь тестовой логики, которая используется только в режиме тестирования и в рабочем режиме не функционирует. Современные методы создания тестовых используют существующие точек функциональные триггеры СБИС вместо дополнительных, что позволяет сократить аппаратурные затраты на тестовую логику, однако увеличения задержек критических путей после установки тестовых точек избежать не удается [13], [14]. Увеличение контролируемости и наблюдаемости критических с точки зрения быстродействия путей представляет серьёзную проблему при проектировании средств тестирования.

Метод создания копий отдельных элементов (дублирования) известен как способ сокращения длин проводников при проектировании топологии СБИС, что позволяет сократить задержки распространения сигналов [15]. Триггер и его копия в функциональном режиме находятся в одинаковом логическом состоянии в любой момент времени. Считается, что и в режиме скан-тестирования должно сохраняться равенство состояний этих триггеров, так как загрузка различных значений в дублированные скан-триггеры может привести к выходу микросхемы из строя в случае, если схеме присутствуют логические элементы, в допускающие сквозные токи [16]. Однако, если таких элементов на кристалле немного или нет вовсе, то использование копий триггеров как независимых переменных при генерации тестовых последовательностей может увеличить тестируемость узлов комбинационной части СБИС.

II. УВЕЛИЧЕНИЕ ТЕСТИРУЕМОСТИ УЗЛОВ КОМБИНАЦИОННОЙ СХЕМЫ

В режиме скан-тестирования тестируемость логического узла (или вероятность обнаружения неисправности) комбинационной схемы может быть рассчитана как минимальное количество тестовых последовательностей, необходимых для обнаружения неисправности, делённое на максимальное количество уникальных входных воздействий [6]. Рассмотрим комбинационную схему с п входами $\vec{x} = (x_1, x_2, ..., x_n)$ и т выходами $\vec{F} = (F_1, F_2, ..., F_m)$. Пусть $g(\vec{x})$ – внутренний сигнал схемы, тогда тестируемость неисправностей типа «залипание-в-0» и «залипание-в-1» в сигнале д могут быть соответственно выражены [6],[17]

$$t(g/0) = S\left(g(\vec{x})\sum_{j=1}^{m} \frac{\partial F_j}{\partial g}\right),\tag{1}$$

$$t(g/1) = S\left(\overline{g(x)}\sum_{j=1}^{m} \frac{\partial F_{j}}{\partial g}\right), \qquad (2)$$

где t(g/0) — тестируемость неисправности типа «залипание-в-0», а t(g/1) - типа «залипание-в-1». S(F) — синдром функции F:

$$S(F) = \frac{K}{2^n} \tag{2}$$

где К – количество минтермов функции F, а n – количество её входов. Произведение функций в скобках – это пересечение множеств входных воздействий, которые задают необходимое значение в сигнале $\vec{g(x)}$ и обеспечивают наблюдение этого значения на выходах \vec{F} . Если схема содержит сходящиеся разветвления, то это пересечение множеств может оказаться довольно небольшим, что в результате приводит к снижению величин t(g/i),i $\in \{0;1\}$.

Входные сигналы, имеющие разветвление на входах, могут быть разделены на отдельные независимые сигналы путём дублирования триггеровисточников, что позволяет сократить число сходящихся разветвлений в схеме, а значит и увеличить тестируемость этой схемы.

Рассмотрим изменение тестируемости узлов комбинационной схемы при дублировании триггеров в сравнении с использованием стандартных тестовых точек контроля и обзора.

На рис. 1а изображен пример фрагмента скан-цепи (триггеры 1-2) и связанная с ней комбинационная схема. Комбинационная схема имеет два независимых входа, которые обозначены переменными x₁ и x₂, и один выход – f. Данная комбинационная схема содержит сходящееся разветвление через узел x₂. Представим, что в сигнале g есть константная неисправность типа «залипание-в-0» (g/0). Тестируемость сигнала $\underline{g=x_2}$ согласно формуле (1) будет равна $t(g/0) = S(x_2 x_1 x_2) = 0$.



Рис. 1. Пример увеличения тестируемости в комбинационной схеме: а) исходная схема; б) модифицированная схема с дублированием триггера

Тестируемость неисправности g/0 равна 0, и это значит, что неисправность «залипание-в-0» в данном узле средствами тестирования установить невозможно. Для увеличения тестируемости t(g/0) этого узла необходимо внести аппаратные изменения в данную схему.

Рассмотрим модифицированную схему с дублированием триггера 2 (см. рис. 1б). В данной схеме разветвление на выходе триггера 2 было разбито на два независимых логических сигнала, управляемых независимо триггерами 2 и 2. Независимость управления достигается за счёт загрузки различных значений в дублированные тригтеры в процессе тестирования. Чтобы комбинационная схема была функционально эквивалентна исходной, входы D триггеров 2 и 2 объединяются. Оценка тестируемости неисправности g/0 для модифицированной схемы с дублированием триггера показала увеличение значения t(g/0), которое составила

$$t(g/0) = S(x_2 x_1 x_2') = 1/8.$$

III. Сокращение взаимных конфликтов неисправностей

Рассмотрим теперь дублирование триггеров как средство сокращения конфликтов неисправностей. Чтобы сократить количество тестовых векторов, необходимое для тестирования неисправностей СБИС, каждый тестовый вектор должен иметь возможность обнаруживать как можно большее число дефектов, чему препятствуют взаимные конфликты неисправностей. Пусть s₀ – источник разветвления сигналов с ветвями s₁,...,s_n, тогда величины

конфликтов выставления 0 и 1 в любой ветви s_k , $k \in [1;n]$ могут быть рассчитаны следующим образом [18]:

$$c_{s_k} = \min\{b_{s_k}; F_{s_k}\},$$
 (4)

$$C_{s_k} = \min\{B_{s_k}; f_{s_k}\},$$
 (5)

где:

- с_{sk} и C_{sk} количество конфликтов выставления логического «0» и «1» в узле sk, соответственно,
- *b_{sk}* и *B_{sk}* необходимое количество логических состояний ветви s_k в «0» и «1», соответственно, для обеспечения наблюдаемости неисправностей на всех остальных ветвях разветвления с источником s₀,
- f_{s_k} и F_{s_k} количество логических состояний ветви s_k в «0» и «1», соответственно, необходимое для наблюдения всех неисправностей, являющихся источником для сигнала s₀, причем

$$F_{s_k} = F_{s_0} + \sum_{i=1}^n B_{s_i}, i \neq k,$$
(6)

$$f_{s_k} = f_{s_0} + \sum_{i=1}^n b_{s_i}, i \neq k,$$
 (7)

Для разветвления сигнала s на выходе тригтера значение $F_{s_0} = f_{s_0} = 0$, то есть $F_{s_k} = \sum_{i=1}^n B_{s_i}, i \neq k$, , $f_{s_k} = \sum_{i=1}^n b_{s_i}, i \neq k$, . Тогда, если число ветвей разветвления сократить до n = 1, то $F_{s_k} = f_{s_k} = 0$. Отсюда $C_{s_k} = c_{s_k} = 0$ в данном разветвлении, то есть конфликтов неисправностей в нём не будет. Более того, для любого сигнала x в путях от s₀ до конечных приёмников сигнала (тригтеров или портов вывода), значения f_x и F_x всех разветвлений будут сокращены на $\sum_{i=1}^n b_{s_i}$ и $\sum_{i=1}^n B_{s_i}$, соответственно. Отсюда по формулам (4) и (5) получаем, что количество конфликтов с_x и C_x любого сигнала x может быть сокращено на $\sum_{i=1}^n B_{s_i}$ и $\sum_{i=1}^n b_{s_i}$, соответственно.

Дублирование триггера-источника разветвления позволяет сократить число ветвей разветвления до 1, что приведет к снижению числа взаимных конфликтов во всех логически путях от этих триггеров.

На рис. 2а показан пример комбинационной схемы с взаимным конфликтом неисправностей в сигнале s. Отдельные части комбинационной схемы отмечены треугольниками, в которых количество неисправностей составляет M_i. По формулам (4)-(7) рассчитаем величины конфликтов в узлах s₁ и s₂, получим:

$$c_{s_1} = \min\{b_{s_1}; F_{s_1}\} = \min\{0; \min\{M_2; M_1\}\} = 0,$$

$$C_{s_2} = \min\{B_{s_2}; f_{s_2}\} = \min\{M_3; M_4\},$$

 $c_{s_2} = \min\{b_{s_2}; F_{s_2}\} = \min\{M_4; \min\{M_1; M_2\} + M_3\},\$

$$C_{s_2} = \min\{B_{s_2}; f_{s_2}\} = \min\{0; 0\} = 0.$$



Рис. 2. Пример сокращения конфликтов неисправностей в комбинационной схеме: а) исходная схема; б) модифицированная схема с дублированием триггеров

Согласно оценке возникает конфликт выставления «0» в узле s_2 и конфликт выставления «1» в узле s_1 . Для разрешения данного конфликта авторами [18] предлагается установить тестовую точку контроля ИЛИ-типа на сигнал s_2 , что приводит к снижению до 0 в данной схеме величин c_{s_1} и C_{s_1} .

Рассмотрим эквивалентную модифицированную схему с дублированием триггеров 1 и 2 (см. рис. 2б). Получается следующий набор выражений, характеризующий сигналы s₁ и s₂ в схеме:

$$c_{s_1} = \min\{0;0\} = 0,$$

$$C_{s_1} = \min\{M_3; M_4\},$$

$$c_{s_2} = \min\{M_4; M_3\},$$

$$C_{s_3} = \min\{0;0\} = 0.$$

Сравнивая результаты оценок конфликтов неисправностей модифицированной и исходной схемы можно заключить, что величина конфликта

выставления «0» в узле s_2 теперь не зависит от переменных M_1 и M_2 , что приводит к снижению c_{s_2} на величину до min{ M_2 ; M_1 }.

Так как предложенный метод дублирования триггера оказывает влияние на любой логический путь от этого триггера до конечных приёмников сигналов за счёт снижения величин взаимных конфликтов разветвления s на выходе этого триггера, то можно сделать вывод, что наибольший эффект от данного метода будет достигнут при дублировании триггеров с наибольшим количеством комбинационных элементов в выходных путях этого триггера.

IV. АЛГОРИТМ ПОИСКА ТРИГГЕРОВ-ИСТОЧНИКОВ ДЛЯ ДУБЛИРОВАНИЯ

Известны методы поиска всех сходящихся разветвлений по логическому описанию схемы [7], [8]. Вариант дублирования всех триггеров, имеющих на выходе разветвление, приводит к чрезмерному росту площади. Таким образом целесообразно выделить для дублирования только те триггеры, которые дают максимальный эффект увеличения тестируемости и сокращения количества конфликтов неисправностей. В данной работе анализируется алгоритм, который основан на поиске логических деревьев с наибольшим числом источников.

Таблица 1

| Характеристика | BM8 | СМПО | Обработка- 2 | Схема-6 | |
|--|--------------|----------------|-----------------|---------|--|
| Технология | Объемн КМ | ая 65 нм ЮП | 250 нм КНИ | | |
| Количество триггеров, 513,8 тысячи | | 344,8 | 103,0 | 76,1 | |
| Количество комбинационных элементов, тысячи | 2084,7 | 878,0 | 528,3 | 191,2 | |
| Количество портов 639 ввода/вывода | | 311 | 353 | 140 | |
| Количество ОЗУ и заказных блоков | 1055 | 371 | 110 | 24 | |

Параметры исследуемых СБИС

Длинные сходящиеся пути являются, как правило, логическими функциями большого числа переменных. Проведён анализ логических путей ряда СБИС с проектными нормами от 65 до 250 нм путём подсчёта количества конечных источников сигнала для каждого триггера схемы (см. табл. 1). Анализ показал, что значительная часть (более 70%) триггеров имеет один конечный источник сигнала. Распределение триггеров по количеству источников сигналов с шагом в 100 единиц дано на рис. 3. В СБИС ВМ8 встречались единицы триггеров с количеством различных источников сигналов вплоть до 7400. Логические функции с таким количеством источников представляют сложность при тестировании, поэтому их дублировать имеет смысл в первую очередь.



☑Схема-6 ВОбработка-2 □СМПО □ВМ9

Рис. 3. Распределение триггеров по количеству источников сигналов

Алгоритм поиска триггеров-источников критических путей с порогом dmax по количеству элементов содержит следующую последовательность шагов:

1) задание количества тригтеров для дублирования $d_{\text{max}};$

2) анализ схемы и получение множества сигналов L с низкой тестируемостью;

3) получение множества конечных источников $S_{\rm i}$ для каждого сигнала $L_{\rm i};$

 исключение из каждого множества S_i портов ввода-вывода и триггеров без разветвления на выходе;

5) сортировка множеств $S_{\rm i}$ по убыванию количества элементов;

6) последовательное добавление элементов из каждого множества S_i в множество D до тех пор, пока их количество не превысит d_{max} .

Результатом алгоритма является множество триггеров D, которые подлежат дублированию.

V. МАРШРУТ ПРОЕКТИРОВАНИЯ СБИС с Дублированием триггеров

Дублирование триггеров осуществляется после того, как получена модель СБИС на уровне логических элементов (нетлист). Сначала осуществляется поиск триггеров-источников критических путей с порогом d_{max}. Триггер, который имеет в выходном логическом пути элемент. допускающий сквозные токи. исключается из списка. Полученный список триггеров D дублируется, после чего осуществляется логическая оптимизация схемы поскольку дублирование триггеров может привести к изменению задержек распространения сигналов. Ha основе оптимизированной схемы создаётся скан-схема с компрессией.



Рис. 4. Маршрут проектирования

Генерация и моделирование тестовых векторов для полученной схемы позволяют оценить время тестирования и коэффициент тестового покрытия. Если необходимые параметры получены, либо достигнут предел заполнения элементов на кристалле, то далее проектируется топология устройства. Если же необходимый коэффициент покрытия не достигнут, необходимость сократить либо есть время тестирования И при этом пространство лля дополнительных триггеров есть, можно изменить параметры скан-схемы [19], либо дополнительно дублировать триггера в критических путях. Маршрут проектирования согласно изложенной методике приведен на рис. 4.

VI. РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ СКАН-СХЕМ СБИС

Предложенная методика была использована при проектировании средств тестирования семи IP-блоков: 64-разрядного микропроцессора ядра (cpu), контроллера 2d графики (2d), контроллера Ethernet 10/100/1000 Мбит/с (eth), контроллера PCI-Е 2.0 8х (рсіе 8x), контролера последовательного RapidIO 4X с частотой передачи 3,125 Гбит/с (rio), контролера SATA 3 Гбит/с (sata) и контроллера USB 2.0 (usb). Также согласно методике были спроектированы средства тестирования трёх систем-на-кристалле (SoC): 64разрядного микропроцессора архитектурой с КОМДИВ последовательными И встроенными каналами RapidIO (proc), шестиканального коммутатора высокоскоростных последовательных каналов RapidIO 10 Гбит/с (smpo) и шестиканального коммутатора PCI Express 2.0 (basis). Вышеперечисленные СБИС отличаются количеством триггеров, объемом ОЗУ, количеством встроенных заказных блоков, количеством доменов синхросигналов, рабочими частотами. площадью комбинационной и триггерной логики. Все СБИС стандартных ячейках синтезировались на ИЗ библиотеки элементов TSMC с проектными нормами 65 нм, а также со встроенными интерфейсными приёмопередатчиками, заказными блоками и блоками встроенной памяти ОЗУ. Параметры всех схем приведены в табл. 2.

Все вышеперечисленные СБИС были исследованы на предмет трудно тестируемых узлов, в результате были получены множества узлов L с низкой тестируемостью для каждой СБИС.

Таблица 2

| | KOURDECTRO TRUFFEROR | | Площадь, | Колинество портов | | |
|---------|----------------------|-----------|--------------------------|--------------------------|-------|--------------|
| Проект | тысячи | Триггеров | Комбинационной логики | ОЗУ и заказных блоков | Общая | ввода/вывода |
| cpu | 96,2 | 1,04 | 2,98 | 6,58 | 10,59 | 840 |
| eth | 31,2 | 0,33 | 0,23 | 2,03 | 2,59 | 602 |
| pcie_8x | 98,3 | 1,05 | 0,90 | 1,23 | 3,18 | 3241 |
| rio | 65,0 | 0,71 | 0,91 | 0,35 | 1,96 | 1638 |
| sata | 14,0 | 0,15 | 0,21 | 0,11 | 0,47 | 2465 |
| usb | 25,8 | 0,28 | 0,32 | 3,51 | 4,11 | 617 |
| 2d | 6,0 | 0,06 | 0,07 | 0,65 | 0,79 | 1857 |
| proc | 513,8 | 6,21 | 8,55 | 43,18 | 57,95 | 639 |
| smpo | 344,8 | 3,79 | 4,06 | 51,78 | 59,62 | 311 |
| basis | 187,6 | 2,20 | 2,77 | 22,31 | 27,28 | 226 |

Параметры модифицируемых СБИС

Порог d_{max} по количеству дополнительно введённых элементов для нужд средств тестирования был задан равным 2% от числа всех триггеров СБИС, что является приемлемым значением роста тестовой логики [20]. После чего, согласно вышеописанному алгоритму, с заданным порогом d_{max} были получены множества триггеров D для каждой схемы, все элементы которых были дублированы.

Затем в каждой из полученных СБИС были созданы средства скан-тестирования (скан-схемы) с [21]. маршрут компрессией Описанный проектирования системы тестирования СБИС реализован в САПР Synopsys DFT Compiler. Параметры полученных схем, такие как количество внутренних скан-цепей и их длина, разрядность внешней шины тестовых данных, а также количество введённых триггеров в процессе дублирования триггеров приведены в табл. 3. Рост площади логики СБИС, вызванный введением дополнительной логики, рассчитан после оптимизаций схем по площади и быстродействию. В среднем рост площади составил 0.38%.

Проведено моделирование полученных скан-схем. Тестовые последовательности были созданы с помощью инструмента ATPG (Automatic Test Pattern Generation) – Synopsys Tetramax.

В итоге определены параметры скан-схем: тестовое покрытие и количество тестовых векторов. Исходя из частоты тестового синхросигнала 10 МГц, длины сканцепей и количества векторов было рассчитана длительность тестирования каждой СБИС (см. табл. 4). Для тестирования скан-схем с дублированием функциональных триггеров потребовалось 4,8-39,1% меньше тестовых последовательностей лля достижения заданного тестового покрытия, чем исходным скан-схемам без дублирования, в среднем по всем исследуемым проектам - на 16,1% меньше. С введением дополнительных триггеров выросли длины скан-цепей в схемах, что привело к увеличению длительности прохождения каждого тестового вектора. Тем не менее за счёт сокращения количества векторов время тестирования сократилось на величины от 3,3% (ІР-блок гіо) до 37,1% (СБИС ргос), в среднем на 14.4%.

Таблица 3

| Проект | Количество скан- | Разрядность шины данных скан-схем, | Длины скан-це тригт | пей скан-схем, геров | Количество введённых | Рост общей площади логики, % |
|---------|------------------|---------------------------------------|------------------------|-------------------------|-------------------------|------------------------------------|
| | Henen | входная/выходная | без дубл. | с дубл. | триггеров | |
| cpu | 164 | 6/6 | 586 | 598 | 1828 | 0,16 |
| eth | 51 | 5/5 | 612 | 622 | 505 | 0,21 |
| pcie_8x | 164 | 6/6 | 601 | 609 | 1411 | 1,18 |
| rio | 111 | 6/6 | 587 | 598 | 1276 | 0,97 |
| sata | 23 | 4/4 | 610 | 622 | 254 | 0,09 |
| usb | 43 | 5/5 | 602 | 618 | 684 | 0,14 |
| 2d | 10 | 3/3 | 595 | 615 | 177 | 0,47 |
| proc | 2000 | 10/10 | 346 | 351 | 7207 | 0,36 |
| smpo | 1000 | 10/10 | 252 | 260 | 5018 | 0,09 |
| basis | 1000 | 9/9 | 189 | 193 | 3725 | 0,14 |

Параметры скан-схем СБИС

Таблица 4

Результаты моделирования скан-схем СБИС

| Проект | Тестовое | Количество тестовых векторов | | Время тестирования скан-схем, с | | Сокращение времени тестирования, % |
|---------|-------------|---------------------------------|---------|---------------------------------|---------|---------------------------------------|
| 1 | покрытие, % | без дубл. | с дубл. | без дубл. | с дубл. | |
| cpu | 93,6 | 4840 | 4418 | 0,284 | 0,264 | 7,0 |
| eth | 94,1 | 1672 | 1565 | 0,102 | 0,097 | 4,9 |
| pcie_8x | 95,3 | 5365 | 5077 | 0,322 | 0,309 | 4,0 |
| rio | 96,6 | 2270 | 2138 | 0,133 | 0,128 | 3,8 |
| sata | 86,2 | 816 | 593 | 0,050 | 0,037 | 26,0 |
| usb | 91,4 | 504 | 480 | 0,030 | 0,029 | 3,3 |
| 2d | 79,6 | 421 | 261 | 0,025 | 0,016 | 36,0 |
| proc | 91,6 | 55236 | 33656 | 1,391 | 0,875 | 37,1 |
| smpo | 90,0 | 19374 | 14993 | 0,670 | 0,526 | 21,5 |
| basis | 91,5 | 10401 | 10147 | 0,196 | 0,195 | 0,5 |

VII. ЗАКЛЮЧЕНИЕ

Предложен метод увеличения тестируемости и взаимных снижения конфликтов неисправностей цифровой СБИС. который заключается в дублировании триггеров в составе трудно тестируемых путей. Показано, что дублирование триггеров в путях со сходящимися разветвлениями позволяет увеличить тестируемость узлов этих путей, а дублирование триггеров-источников логических путей позволяет снизить число взаимных конфликтов неисправностей во всех узлах этих путей. Предложенная методика дублирования отдельных триггеров была реализована в 3 проектах СБИС и 7 проектах IP-блоков при проектировании средств скан-тестирования с компрессией. Результаты показали снижение времени тестирования в среднем на 14,4% при аппаратурных затратах, не превышающих 1,2% общей площади СБИС.

ЛИТЕРАТУРА

- Abramovici M., Breuer M. A. and Friedman A.D. Digital Systems Testing and Testable Design, Computer Science Press, 1990. 364-366 p.
- [2] Touba N. A. Survey of test vector compression techniques // IEEE Design & Test of Computers. – July-August 2006. – Vol. 23. №4. - P. 294–303.
- [3] Kapur R., Historical perspective on scan compression // IEEE Design & Test of Computers. – March-April 2008. – Vol. 25. №2. – P. 114-120.
- [4] Hapke F., Redemund W., Glowatz A., Rajski J., Reese M., Hustava M., Keim M., Schloeffel J., Fast A. Cell-Aware Test // IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.- Sep. 2014. - Vol. 33. - №9. – P. 1396-1409.
- [5] Acero C., Feltham D., Hapke F., Moghaddam E., Mukherjee N., Neerkundar V., Patyra M., Rajski J., Tyszer J., Zawada J. Embedded deterministic test points for compact cellaware tests // Test Conf ITC 2015 IEEE Int. - Oct. 2015. -P. 1-8.
- [6] Savir J. Good Controllability and Observability Do Not Guarantee Good Testability // IEEE Transactions on Computers. - 1983. - V. 32. №12. - P.1198-1200.
- [7] Robert M.W., Lala P. K. Algorithm to Detect Reconvergent Fanout in Logic Circuits // IEEE Proceedings. - 1987. -Vol.134. №2. - P.105-111.
- [8] Xu S., Edirisuriya E. A new way of detecting reconvergent fanout branch pairs in logic circuits // Asian Test Symposium (ATS'04). - 2004. - P. 354-357.

- [9] Pomeranz I., Reddy S.M. Test-point insertion to enhance test compaction for scan designs // Proc. ICDSN. – 2000. - P. 375-381.
- [10] Geuzebroek M.J., Linden J.T., Goor A.J. Test point insertion that facilitates ATPG in reducing test time and data volume // Proc. ITC. – 2002. - P. 138-147.
- [11] Kumar A., Rajski J., Reddy S.M., Rinderknecht T. On the Generation of Compact Deterministic Test Set for BIST Ready Designs // Asian Test Symposium. – 2013. - P. 201-206.
- [12] Acero C., Feltham D., Liu Y., Moghaddam E., Mukherjee N., Patyra M., Rajski J., Reddy S.M., Tyszer J., Zawada J. Embedded deterministic test points // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 2017. - P. 1-13.
- [13] Ren H., Kusko M., Kravets V., Yaari R. Low cost test point insertion without using extra registers for high performance design // Proc. ITC. - 2009. – P. 1-8.
- [14] Yang J., Touba N. A., Nadeau-Dostie B. Test Point Insertion with Control Points Driven by Existing Functional Flip-Flops // IEEE Transactions on Computers. 2012. V. 61. P. 1473-1483.
- [15] Srivastava A., Kastner R., Sarrafzadeh M. Timing driven gate duplication: Complexity issues and algorithms // Proc. ICCAD. - 2000. - P. 447-450.
- [16] Goessel M., Singh A., Sogomonyan E. Scan-path with directly duplicated and inverted duplicated registers // Proceedings 20th IEEE VLSI Test Symposium. 2002. P. 47-52.
- [17] Savir J. Syndrome-testable design of combinational circuits // IEEE Transactions on Computers, 1980. V.29. - P. 442-451.
- [18] Liu Y., Moghaddam E., Mukherjee N., Reddy S. M., Rajski J., Tyszer J. Minimal area test points for deterministic patterns // Proc. ITC. Nov. 2016. Р. 1-7
 [19] Ладнушкин М.С. Снижение аппаратурных затрат и
- [19] Ладнушкин М.С. Снижение аппаратурных затрат и увеличение коэффициента компрессии средств тестирования константных неисправностей КМОП цифровых СБИС // VII Всероссийская научнотехническая конференция «Проблемы разработки перспективных микро- и наноэлектронных систем – 2016». Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2016. Ч.2, С. 68-75.
- [20] Acero C., Feltham D., Patyra M. et al. On new test points for compact cell-aware tests // *IEEE Des. Test.* - Dec. 2016.
 - Vol. 33 №6. - P. 7-14.
- [21] Wohl P., Waicukauski J.A., Ramnath S. Fully X-Tolerant Combinational Scan Compression // International Test Conference. - 2007. - P. 1-10.

Register Duplication for Scan Compression Designs

M.S. Ladnushkin

Department of Russian Academy of Sciences Institute of System Researches of CAD Systems, maximsl@gmail.com

Abstract — Volumes of test data are growing and test compression schemes are not enough for today's challenges such as growth of length of logic paths and higher amount of blocking nonequivalent faults during test. These factors cause not only inflation of test sets and new requirements of storage but also increase test application time. Test point are widely used to improve testability and decrease number of blocking faults by inserting controlling and observing additional logic.

Method of duplicating cells is known as a timing improving approach which is used during topology planning. In this paper it was shown that duplicating of functional register can improve testability if there are reconvergent fanouts on trigger's output. Also it was shown that duplicating of register can reduce number of blocking faults on each combinational cell in all output paths of that register.

Complexity analysis of logic paths of several VLSI's was presented. It was observed that more than 70% of all paths have one source. But there were a few paths with 7400 startpoints in one of the designs. Such paths are the most hard to test paths so duplicating of startpoint triggers of these paths would have maximum effect on testability.

An algorithm of selecting registers for duplication with long output paths was proposed. Experiments on 10 industrial projects show average test time reduction 14,4% while area overhead was less than 1,2%.

Keywords — scan testing, register duplicating, test compression, modeling.

REFERENCES

- Abramovici M., Breuer M. A. and Friedman A.D. Digital Systems Testing and Testable Design, Computer Science Press, 1990. 364-366 p.
- [2] Touba N. A. Survey of test vector compression techniques // IEEE Design & Test of Computers. – July-August 2006. – Vol. 23. №4. - P. 294–303.
- [3] Kapur R., Historical perspective on scan compression // IEEE Design & Test of Computers. – March-April 2008. – Vol. 25. №2. – P. 114-120.
- [4] Hapke F., Redemund W., Glowatz A., Rajski J., Reese M., Hustava M., Keim M., Schloeffel J., Fast A. Cell-Aware Test // IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.- Sep. 2014. - Vol. 33. - №9. – P. 1396-1409.
- [5] Acero C., Feltham D., Hapke F., Moghaddam E., Mukherjee N., Neerkundar V., Patyra M., Rajski J., Tyszer J., Zawada J. Embedded deterministic test points for compact cellaware tests // Test Conf ITC 2015 IEEE Int. - Oct. 2015. -P. 1-8.
- [6] Savir J. Good Controllability and Observability Do Not Guarantee Good Testability // IEEE Transactions on Computers. - 1983. - V. 32. №12. - P.1198-1200.

- [7] Robert M.W., Lala P. K. Algorithm to Detect Reconvergent Fanout in Logic Circuits // IEEE Proceedings. - 1987. -Vol.134. №2. - P.105-111.
- [8] Xu S., Edirisuriya E. A new way of detecting reconvergent fanout branch pairs in logic circuits // Asian Test Symposium (ATS'04). - 2004. - P. 354-357.
- [9] Pomeranz I., Reddy S.M. Test-point insertion to enhance test compaction for scan designs // Proc. ICDSN. – 2000. - P. 375-381.
- [10] Geuzebroek M.J., Linden J.T., Goor A.J. Test point insertion that facilitates ATPG in reducing test time and data volume // Proc. ITC. – 2002. - P. 138-147.
- [11] Kumar A., Rajski J., Reddy S.M., Rinderknecht T. On the Generation of Compact Deterministic Test Set for BIST Ready Designs // Asian Test Symposium. – 2013. - P. 201-206.
- [12] Acero C., Feltham D., Liu Y., Moghaddam E., Mukherjee N., Patyra M., Rajski J., Reddy S.M., Tyszer J., Zawada J. Embedded deterministic test points // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 2017. - P. 1-13.
- [13] Ren H., Kusko M., Kravets V., Yaari R. Low cost test point insertion without using extra registers for high performance design // Proc. ITC. - 2009. – P. 1-8.
- [14] Yang J., Touba N. A., Nadeau-Dostie B. Test Point Insertion with Control Points Driven by Existing Functional Flip-Flops // IEEE Transactions on Computers. 2012. V. 61. P. 1473-1483.
- [15] Srivastava A., Kastner R., Sarrafzadeh M. Timing driven gate duplication: Complexity issues and algorithms // Proc. ICCAD. - 2000. - P. 447-450.
- [16] Goessel M., Singh A., Sogomonyan E. Scan-path with directly duplicated and inverted duplicated registers // Proceedings 20th IEEE VLSI Test Symposium. 2002. P. 47-52.
- [17] Savir J. Syndrome-testable design of combinational circuits // IEEE Transactions on Computers, 1980. V.29. - P. 442-451.
- [18] Liu Y., Moghaddam E., Mukherjee N., Reddy S. M., Rajski J., Tyszer J. Minimal area test points for deterministic patterns // Proc. ITC. - Nov. 2016. – P. 1-7
- [19] Ladnushkin M.S. Snizhenie apparaturnyh zatrat i uvelichenie kojefficienta kompressii sredstv testirovanija konstantnyh neispravnostej KMOP cifrovyh SBIS (Reducing area and increasing compression ratio of scan compression system for digital VLSI using stuck-at fault model) // VII Vserossijskaja nauchno-tehnicheskaja konferencija «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem – 2016». Sbornik trudov / pod obshh. red. akademika RAN A.L. Stempkovskogo. M.: IPPM RAN, 2016. Ch.2, S. 68-75.
- [20] Acero C., Feltham D., Patyra M. et al. On new test points for compact cell-aware tests // *IEEE Des. Test.* - Dec. 2016.
 - Vol. 33 №6. - P. 7-14.
- [21] Wohl P., Waicukauski J.A., Ramnath S. Fully X-Tolerant Combinational Scan Compression // International Test Conference. - 2007. - P. 1-10.

Методы верификации на кристалле задержек распространения стандартных цифровых элементов

А.В. Кобыляцкий, Д.К. Сергеев

НИЯУ МИФИ, г. Москва, AVKobylyatskiy@mephi.ru, DKSergeev@mephi.ru

Аннотация — В статье представлен сравнительный анализ методов измерения временных интервалов на кристалле, которые могут быть применены для верификации задержек распространения стандартных цифровых элементов. Сравнение производится по сложности реализации, точности определения задержек, занимаемой площади и т.д. Один из рассмотренных методов использован авторами на тестовом кристалле. На основе проведенного моделирования дана оценка точности определения задержек при измерении с использованием разработанной схемы.

Ключевые слова — СБИС, система-на-кристалле, стандартный цифровой элемент, временные параметры, задержка распространения, верификация на кристалле, тестовая структура, характеризация, кольцевой генератор.

I. Введение

Уменьшение проектных норм позволило добиться высокой степени интеграции элементов интегральных схем, а также значительно увеличить их быстродействие. Однако проведение статического временного анализа и предсказание выхода годных таких схем является сложной задачей, поскольку при нанометровые проектные переходе на нормы увеличение наблюдается существенное вклала перекрестных помех, вариаций технологии и внешних факторов в разброс временных параметров схем. По этой причине проблема верификации разрабатываемых устройств на кристалле особенно актуальна в наши дни.

Проектирование современных СБИС осуществляется на основе стандартных цифровых элементов (СЦЭ) и сложно-функциональных (СФ) Библиотеки макроблоков. СЦЭ являются законченными функциональными изделиями. работоспособность которых, как и любого устройства или изделия, должна быть подтверждена в кремнии (Silicon-proven). Это необходимо, в первую очередь, для того, чтобы разработчики СБИС, использующие данные библиотеки, были уверены в том, что реальные характеристики лежат в элементов пределах заявленных значений. Недостаточно точная характеризация СЦЭ может привести к уменьшению выхода годных и существенному увеличению временных и экономических издержек.

Функциональная верификация элементов библиотек представляется тривиальной задачей, в то время как

верификация их временных параметров является более сложной задачей, требующей наличия дополнительных измерительных схем или специального оборудования. Одним из важнейших временных параметров цифровых элементов является задержка распространения. Для проведения верификации задержки отдельного СЦЭ, необходимо измерить временной интервал, лежащий в Большинство пикосекундном диапазоне. представленных на настоящий момент решений, предназначенных для измерения временных интервалов, имеют разрешение, недостаточное для оценки столь малой задержки. Поэтому схема для верификации задержек СЦЭ должна иметь точность, не хуже единиц пикосекунд.

Проблема верификации временных параметров СЦЭ широко освещена в зарубежной литературе, в то время как отечественные источники практически не акцентируют на ней внимание. В настоящее время существует большое количество методик измерения задержек на кристалле, отличающихся точностью, сложностью реализации и т. п. Однако малое количество из них упоминается в контексте верификации СЦЭ. Таким образом, информация о существующих методах верификации задержек СЦЭ не систематизирована, что значительно усложняет процесс поиска оптимального решения для интеграции на собственные тестовые структуры.

В данной работе проводится сравнительный анализ применимых для верификации задержек распространения СЦЭ методов измерения временных интервалов на кристалле. Один из рассматриваемых методов использован авторами на тестовом кристалле. По результатам проведенного моделирования дана оценка точности определения задержек при измерении с использованием разработанной схемы.

II. МЕТОДЫ ИЗМЕРЕНИЯ ВРЕМЕННЫХ ИНТЕРВАЛОВ НА КРИСТАЛЛЕ

В данной главе рассмотрены описанные в литературе методы измерения временных интервалов, позволяющие с достаточной точностью определить задержки распространения СЦЭ.

А. Использование специального оборудования

Современные технологии позволяют бесконтактно проводить внутрисхемные измерения временных характеристик. Наиболее известным из таких методов является зондирование электронным лучом (Е-Веат

Probing), однако с появлением новых технологий данный метод стал менее популярным [1]. На настоящий момент в зарубежной микроэлектронной индустрии наиболее часто используются лазерное зондирование (ЛЗ, Laser Voltage Probing) и временные измерения при фотонной эмиссии (ВИФЭ, Time Resolved Photon Emission).

Тестирование схем по методам ЛЗ и ВИФЭ осуществляется со стороны подложки. Для этого кристалл декорпусируется, после чего подложка утончается до толщины 50-100 мкм [2]. Далее кристалл устанавливается в стенд для тестирования, а на его входы подаются периодические электрические сигналы. Детектор регистрирует отраженное лазерное (в случае ЛЗ), либо индуцированное (в случае ВИФЭ) фотонное излучение, а специализированный анализатор восстанавливает временные диаграммы интересующих электрических узлов. По полученным диаграммам определяется задержка соответствующей схемы.

высокую Несмотря на цену отсутствие И отечественных разработок, ЛЗ и ВИФЭ являются инструментами мощнейшими арсенале в инженера-тестировщика. Наличие таких установок существенно ускоряет и упрощает процесс поиска некорректно работающих узлов СБИС. Временное разрешение такого оборудования достаточно высоко для оценки задержек СЦЭ.

В. Линия Вернье

Линия Вернье (Vernier Delay Line) состоит из триггеров и двух линий задержки (рис. 1). Для измерения задержки интересующего элемента, на входы START и STOP подаются соответственно входные и выходные напряжения этого элемента. Параметры буферов выбираются таким образом, чтобы задержка между сигналами после каждого каскада сокращалась на величину δ . В зависимости от задержки между сигналами START и STOP срабатывает *i* триггеров. Тогда измеренная задержка T_3 определяется по формуле:

$$T_{3} = (i-1) \times \delta .$$

Элементы задержки (буферы) такой линии крайне чувствительны к шумам, технологическим и температурным вариациям. Кроме того, временное разрешение ограничено временами предустановки и удержания триггеров. Для повышения точности данного метода используется автоподстройка по задержке (Delay-locked loop) [3], а также дополнительная обвязка и схема калибровки [4].

Основным недостатком данного метода является большая занимаемая площадь, поскольку при уменьшении значения δ требуется большее количество триггеров и калибровочных блоков. Так, например, в [4] для размещения измерительной схемы с временным разрешением 5 пс, используется весь кристалл. Кроме того, большой объем ручного проектирования делает данный метод довольно ресурсозатратным.



Рис. 1. Принципиальная схема линии Вернье

С. «Встроенный осциллограф»

Рассматриваемый метод имеет сходства в подходе к обработке сигналов с цифровым осциллографом, принцип работы которого основан на оцифровке исследуемого сигнала. Цифровой осциллограф производит составление выборки (дискретизацию) мгновенных значений исследуемого сигнала, а затем записывает полученные данные в запоминающее устройство для дальнейшей обработки и вывода на экран. Метод «встроенного осциллографа» основан на аналогичных операциях в пределах кристалла.

На тестируемый элемент подается периодическое входное воздействие. Затем составляются выборки мгновенных значений входного и выходного сигналов («осциллограммы»). Эти данные поступают на контроллер, который вычисляет задержку между этими сигналами. Дискретизация сигнала может проводиться, как в псевдослучайные [5]-[7], так и в детерминированные [8]-[9] моменты времени.

«встроенного осциллографа» является Метол наиболее часто используемым для верификации на кристалле задержек СЦЭ. Его точность достигает 1 пс, а занимаемая площадь измерительных схем значительно меньше, чем, например, у большинства реализаций метода линии Вернье [9]. Однако получение столь высокого временного разрешения влечет за собой использование дополнительных схем калибровки и без того сложных аналоговых цепей таких, как генератор опорного напряжения, блоки дискретизации и схемы выборки-хранения. Кроме того, дискретизированные временные диаграммы нуждаются в постобработке, что требует дополнительных временных затрат на разработку соответствующего программного обеспечения.

D. Гомодинный преобразователь

Самым точным из предложенных на настоящий методов измерения задержек является момент гомодинный преобразователь время-код [10]. позволяющий получить временное разрешение порядка десяти фемтосекунд. В основе данного метода лежит гомодинный смеситель, преобразующий разность фаз между входными сигналами в пропорциональное постоянное напряжение, используя анапоговый умножитель и фильтр нижних частот, как показано на рис. 2. Генерация постоянного напряжения Vdc происходит за счет модуляции первого входного сигнала вторым и последующей фильтрации,

удаляющей переменную составляющую модулированного сигнала. Далее напряжение Vdc оцифровывается сигма-дельта АЦП, которое получает код, соответствующий задержке между интересующими сигналами.



Рис. 2. Структурная схема гомодинного преобразователя время-код [11]

Недостатком данной схемы является сложность связанная, реализации. частности, в С нетривиальностью схемы сигма-дельта АЦП. Проектирование устройства такой сложности требует больших трудозатрат и соблюдения специальных правил цифро-аналогового проектирования. Кроме того, в случае большого количества верифицируемых элементов, возникает проблема симметрирования сигналов с сохранением разности фаз, что требует дополнительных введения аналоговых преобразователей.

Е. Управляемый кольцевой генератор

С давних пор кольцевые генераторы используются для верификации и оценки вариаций временных параметров логических элементов [12]. Имеется множество работ [13]-[15], показывающих высокую точность и легкость реализации данного метода. Тем не менее классический кольцевой генератор не позволяет определить задержку конкретного одиночного элемента, ограничиваясь лишь средним по всем звеньям значением задержки.

Для определения задержки одиночного элемента было предложено [16]-[17] использовать разностный подход, а именно – рассчитывать задержки по разностям периодов кольцевых генераторов, находящихся в разных состояниях. В основе данного лежит кольцевой генератор, звенья метода (измерительные ячейки) которого управляются с помощью сигналов S[1]...S[N] (рис. 3). Каждая измерительная ячейка (ИЯ) состоит из тестируемого устройства (I₁, ТУ), мультиплексора (M₁), буфера (B₁) и нагрузочных элементов (I₂, M₂, B₂), как показано на рис. 4. Под ТУ понимается тестируемый СЦЭ, путь между входом и выходом которого является комбинационным (метод не предусматривает измерение задержек последовательных схем). Управляющий сигнал S[i] регулирует прохождение сигнала через мультиплексор со входа ИЯ на ее выход через ТУ, либо в обход него.



Рис. 3. Кольцевой генератор на основе измерительных ячеек. Для генератора на основе инвертирующих ячеек число N – четное



Рис. 4. Схема измерительной ячейки

В основе предложенного метода лежит вычисление разности задержек распространения сигнала со входа на выход ИЯ:

$$t_{S[i]=0} - t_{S[i]=1} = t_{pi} + M_A - M_B, \qquad (1)$$

где $t_{S[i]=0}$ и $t_{S[i]=1}$ – времена распространения сигнала через ИЯ соответственно при низком и высоком логическом уровне сигнала на линии S[i]; t_{pi} – задержка распространения сигнала через ТУ; M_A и M_B – задержки распространения сигнала через мультиплексор M_1 соответственно со входов A и B на выход Y.

Аналогично формуле (1) имеет место связь длительностей импульса и паузы на выходе кольцевого генератора (рис. 3) и задержек нуля и единицы t_{p10i} и

*t*_{*p*01*i*} ТУ *i*-ой измерительной ячейки:

$$t_{p10_i} = (T_{ON}^{S_{0i}} - T_{ON}^{S_{1i}}) - (M_{Ai}^{10} - M_{Bi}^{10}), \qquad (2)$$

$$t_{p01_i} = (T_{OFF}^{S_{0i}} - T_{OFF}^{S_{1i}}) - (M_{Ai}^{01} - M_{Bi}^{01}), \qquad (3)$$

где S_{0i} и S_{1i} – статусные вектора, в которых соответственно S[i] = 0 и S[i] = 1, а остальные биты имеют одинаковые значения (например, $S_{04} = \{1110\}$ и $S_{14} = \{1111\}$); $T_{ON}^{S_{0i}}$ и $T_{ON}^{S_{1i}}$ – длительности импульса на выходе УКГ соответственно в состояниях S_{0i} и S_{1i} ; $T_{OFF}^{S_{0i}}$ и $T_{OFF}^{S_{1i}}$ – длительности паузы на выходе УКГ соответственно в состояниях S_{0i} и S_{1i} ; $T_{OFF}^{S_{0i}}$ и $T_{OFF}^{S_{1i}}$ – длительности паузы на выходе УКГ соответственно в состояниях S_{0i} и S_{1i} ; M_{Ai}^{10} и M_{Bi}^{10} – времена распространения логического нуля соответственно со входов А и В на выход Y мультиплексора *i*-ой измерительной ячейки; M_{Ai}^{01} и

 M_{Bi}^{01} – времена распространения логической единицы соответственно со входов А и В на выход Ү мультиплексора *i* -ой измерительной ячейки.

Второй член в скобках правой части формул (2) и (3) определяет асимметрию мультиплексора *i*-ой измерительной ячейки и может быть минимизирован схемотехническими и топологическими методами.

Для УКГ, состоящего из инвертирующих измерительных ячеек, не все комбинации статусных битов *S* являются допустимыми, поэтому определение задержек логической единицы и логического нуля является затруднительным, однако имеется возможность получить среднюю задержку

$$t_{pi} = \frac{t_{p01_i} + t_{p10_i}}{2},$$

рассчитываемую по периодам УКГ в различных состояниях шины *S*.

Несмотря на проблему измерения задержек инвертирующих элементов и последовательных схем, метод УКГ прост в реализации и обладает хорошей точностью. Кроме того, малый объем вспомогательных блоков позволяет эффективно использовать площадь на кристалле.

III. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ

В табл. 1 приведено сравнение рассмотренных методов измерения временных интервалов. Каждый из этих методов имеет свои достоинства и недостатки, однако с точки зрения авторов, наравне с точностью, важнейшим критерием выбора метода верификации на кристалле является простота реализации. В методе управляемых кольцевых генераторов необходимость ручного проектирования сводится к минимуму, поскольку все элементы, кроме мультиплексора, являются библиотечными, а трассировка сигнальных линий производится автоматически с помощью САПР.

Следующая глава описывает разработанную реализацию данного метода.

Таблица 1

| Название | Временное разрешение, пс | S _{изм} *, мм ² / Техпроцесс, нм | Недостатки |
|--|--------------------------------|--|--|
| ЛЗ [18] ВИФЭ [19] | 10 | _ | • Стоимость |
| Линия Вернье [20] Линия Вернье (Flash) [4] | 5 | 6 / 700 2,25 / 180 | Большая занимаемая площадьСложность проектирования |
| Встроенный осциллограф (с дискретизацией в детерменированные [9] и псевдослучайные [5] моменты времени) | 1 | 0,25 / 90 0,15 / 90 | • Сложность проектирования |
| Гомодинный преобразователь [10] | 0.04 | 0,1 / 120 | Большой объем ручного проектированияПроблема симметрирования сигналов |
| УКГ [17] | 1,5 | 0,05 / 65 | Отсутствует возможность раздельного измерения t_{p01} и t_{p10} неинвертирующих элементов. Измеряется средняя задержка t_p Отсутствует возможность измерения задержек последовательных схем |

Сравнение методов верификации задержек СЦЭ

* - оценка площади кристалла, занимаемой измерительной схемой, для верификации задержек 50 СЦЭ (без учета площади верифицируемых СЦЭ)



Рис. 5. Точность определения задержек распространения логической единицы (а) и нуля (б) неинвертирующих СЦЭ и средней задержки (в) инвертирующих СЦЭ по методу УКГ с использованием разработанной схемы. *укг* – задержка рассчитанная по методу УКГ; *ист* – истинное значение задержки, полученное по временным диаграммам входа и выхода ТУ

IV. РАЗРАБОТАННАЯ ТЕСТОВАЯ СТРУКТУРА

верификации имеющихся Для библиотечных элементов был разработан блок УКГ, позволяющий измерить задержки 72 СЦЭ. Всего на кристалле vстановлено 4 таких блока для оценки пространственно-коррелированных вариаций. Тестовый кристалл изготовлен по объемной КМОП технологии с проектными нормами 90 нм. С учетом шин земли и питания, а также площади, занимаемой верифицируемыми элементами, размеры блока составили 105×145 мкм².

В ходе анализа возможных источников погрешности измерения, вносимой измерительной схемой, было установлено, что наибольший вклад в эту погрешность вносит мультиплексор M_1 . Проведенное моделирование разработанной схемы показало, что максимальная абсолютная погрешность измерения задержек, вносимая схемой, составляет 3 пс. На рис. 5 приведены гистограммы, иллюстрирующие точность определения задержек на примере 8 верифицируемых элементов. Результирующая ошибка измерений Δt рассчитывается следующим образом:

$$\Delta t = |t_{ucm} - t_{yk2}| + \Delta t_{yk23\sigma}$$

где t_{yx2} – задержка полученная с использованием разработанной схемы (без учета вариаций в схеме мультиплексора), t_{ucm} – истинное значение этой задержки, полученное по временным диаграммам входа и выхода ТУ, $\Delta t_{yx23\sigma}$ – отклонение задержки (на уровне 3 σ), полученной по методу УКГ, от своего среднего значения, вызванное вариациями в схеме мультиплексора.

Временное разрешение данной схемы может быть улучшено путем перепроектирования схемы мультиплексора, а именно увеличением размеров транзисторов и дополнительной симметризацией топологии. Общий вид топологии тестового кристалла представлен на рис. 6.



Рис. 6. Топология тестового кристалла, в который включены блоки УКГ

V. Заключение

Проведен сравнительный анализ применимых для верификации задержек СЦЭ методов измерения временных интервалов. Для верификации имеющихся библиотек на тестовом кристалле выбран метод управляемых кольцевых генераторов, который прост в реализации, а малый объем вспомогательных блоков позволяет эффективно использовать площадь кристалла.

Тестовый кристалл изготовлен по объемной КМОП технологии с проектными нормами 90 нм и содержит блок УКГ, который занимает 105×145 мкм², позволяя верифицировать задержки 72 СЦЭ. Установлено, что абсолютная погрешность измерений, вносимая схемой, составляет не более 3 пс.

Литература

[1] Parrassin T., Larre P., Dudit S. et al. From EBT to LVP, from 130nm to 28nm node, internal timing characterization

evolution // Proceedings from the 38th International Symposium for Testing and Failure Analysis (ISTFA). 2012. P. 232–238.

- [2] Boit C., Schlangen R., Glowacki A. et al. Physical IC debug – backside approach and nanoscale challenge // Advances in Radio Science. 2008. V. 6. P. 265-272.
- [3] Abdel-Hafeez S., Harb S.M., Lee K.M. On-Chip Jitter Measurement Architecture Using A Delay-Locked Loop With Vernier Delay Line, To The Order Of Giga Hertz // Proceedings of 18th International Conference MIXDES. 2011. P. 502-506.
- [4] Levine P.M., Roberts G.W. High-Resolution Flash Time-To-Digital Conversion and Calibration For System-On-Chip Testing // IEE Proceedings Computers and Digital Techniques. 2005. V. 152. № 3. P. 415-426.
- [5] Bhatti R.Z., Chugg K.M., Draper J. Standard Cell based Pseudo-Random Clock Generator for Statistical Random Sampling of Digital Signals // Proceedings of 50th Midwest Symposium on Circuits and Systems. 2007. P. 1110-1113.
- [6] Maggioni S., Veggetti A., Bogliolo O., Croce L. Random sampling for on-chip characterization of standard-cell propagation delay // IEEE International Symposium on Quality Electronic Design. 2003. P. 41-45.
- [7] Churayev S.O., Matkarimov B.T., Paltashev T.T. On-chip Measurements of Standard-Cell Propagation Delay // Proceedings of Design & Test Symposium (EWDTS). 2010. P.179-181.
- [8] Zhang X., Ishida K., Takamiya M., Sakurai T. An On-Chip Characterizing System for With-in-Die Delay Variation Measurement of Individual Standard Cells in 65-nm CMOS // Proceedings of the ASP-DAC. 2011. P. 109-110.
- [9] Inagaki K., Antono D., Takamiya M. et al. A 1-ps Resolution On-Chip Sampling Oscilloscope with 64:1 Tunable Sampling Range Based on Ramp Waveform Division Scheme // Symposium on VLSI Circuits. Digest of Technical Papers. 2006. P. 61-62.
- [10] Collins M., Al-Hashimi B., Wilson P. On-chip timing measurement architecture with femtosecond resolution // Proceedings of 11th IEEE European Test Symposium. 2006. P.103-110.
- [11] Collins M. On-Chip Time Measurement Architectures and Implementation. Thesis for the degree of Doctor of Philosophy. May, 2009. Southampton, UK. – 160 p.

- [12] Cassard J.M. A Sensitivity Analysis of SPICE Parameters Using an Eleven-Stage Ring Oscillator // IEEE Transactions on Electron Devices. 1984. V. 31. № 2. P. 264-269.
- [13] Masuda H., Ohkawa S., Kurokawa A. et al. Challenge: variability characterization and modeling for 65- to 90-nm processes // Proceedings of the IEEE Custom Integrated Circuits Conference. 2005. P. 593-600.
- [14] Bhushan M., Gattiker A., Ketchen M.B. et al. Ring oscillators for CMOS process tuning and variability control // IEEE Transactions on Semiconductor Manufacturing. 2006. V. 19. № 1. P. 10–18.
- [15] Ketchen M.B., Bhushan M. Product-representative "at speed" test structures for CMOS characterization // IBM Journal of Res. and Dev. 2006. V. 50. № 4/5. P. 451–468.
- [16] Das B.P., Onodera H. Area-efficient reconfigurable-arraybased oscillator for standard cell characterization // IEEE Transactions on Circuits and Systems - II: Express Briefs. 2014. V. 61. № 3. P. 429-436.
- [17] Das B.P., Onodera H. On-Chip Measurement of Rise/Fall Gate Delay Using Reconfigurable Ring Oscillator // IEEE Transactions On Circuits And Systems. 2014. V. 61. № 3. P. 183-187.
- [18] Kindereit U. Fundamentals and Future Applications of Laser Voltage Probing // Proc. of IEEE International Reliability Physics Symposium. 2014. P. 3F.1.1 - 3F.1.11.
- [19] Frohmann S., Dietz E., Dittrich H., Hübers H.W. Picosecond imaging of signal propagation in integrated circuits // Advanced Optical Technologies. 2017. V. 6. № 2. P. 137– 142.
- [20] Dudek P., Szczpanksi S., Hatfield J.V. A high-resolution CMOS time-to-digital converter utilizing a Vernier delay line // IEEE Transactions on Solid-State Circuits. 2000. V. 35. № 2. P. 240–247.

On-chip Standard Cell Delay Verification Techniques

A.V. Kobylyatskiy, D.K. Sergeev

NRNU MEPhI, Moscow, AVKobylyatskiy@mephi.ru, DKSergeev@mephi.ru

Abstract — Standard cell libraries are complete products, functionality of which must be silicon-proven. However, the issue of standard cell delay validation is not enough highlighted. The concepts of on-chip delay measurement are well-known, yet there is no systematized information about existing techniques that are suitable for measuring standard cell propagation delays. In this work we present comparative analysis of such techniques that are referred to date. The techniques compared are as follows: on-chip oscilloscope, random sampling, Vernier delay line, flash, homodyne conversion, reconfigurable ring oscillator and some off-chip techniques. The benchmarks chosen are delay measurement accuracy, design complexity and area overhead. We also give a short description for each technique. From all the diversity of presented techniques the reconfigurable ring oscillator approach was chosen for implementing on our test chip. The chosen technique is area-efficient, very simple to design and provides decent accuracy (authors of the concept report 1.5 ps time resolution). The concept does not involve any analog circuits and can be designed without much effort. The test chip has been fabricated in a bulk 90-nm CMOS process. The CAD simulation of the designed structure shows maximum delay measurement error to be approximately 3 ps. We assume that the discrepancy is due to multiplexer circuit. It is supposed that timing resolution could be improved by enlarging transistor sizes and adding more symmetry to the MUX layout. The expected measurement accuracy should not be greatly less than the simulated one since possible IR drop, noise and self-heating effects were accounted.

Keywords — VLSI, SoC, standard cell, timing, propagation delay, on-chip verification, test structure, characterization, post-silicon validation, reconfigurable ring oscillator

REFERENCES

- Parrassin T., Larre P., Dudit S. et al. From EBT to LVP, from 130nm to 28nm node, internal timing characterization evolution // Proceedings from the 38th International Symposium for Testing and Failure Analysis (ISTFA). 2012. P. 232–238.
- [2] Boit C., Schlangen R., Glowacki A. et al. Physical IC debug – backside approach and nanoscale challenge // Advances in Radio Science. 2008. V. 6. P. 265-272.
- [3] Abdel-Hafeez S., Harb S.M., Lee K.M. On-Chip Jitter Measurement Architecture Using A Delay-Locked Loop With Vernier Delay Line, To The Order Of Giga Hertz // Proceedings of 18th International Conference MIXDES. 2011. P. 502-506.
- [4] Levine P.M., Roberts G.W. High-Resolution Flash Time-To-Digital Conversion and Calibration For System-On-Chip Testing // IEE Proceedings Computers and Digital Techniques. 2005. V. 152. № 3. P. 415-426.
- [5] Bhatti R.Z., Chugg K.M., Draper J. Standard Cell based Pseudo-Random Clock Generator for Statistical Random Sampling of Digital Signals // Proceedings of 50th Midwest Symposium on Circuits and Systems. 2007. P. 1110-1113.
- [6] Maggioni S., Veggetti A., Bogliolo O., Croce L. Random sampling for on-chip characterization of standard-cell propagation delay // IEEE International Symposium on Quality Electronic Design. 2003. P. 41-45.
- [7] Churayev S.O., Matkarimov B.T., Paltashev T.T. On-chip Measurements of Standard-Cell Propagation Delay // Proceedings of Design & Test Symposium (EWDTS). 2010. P.179-181.
- [8] Zhang X., Ishida K., Takamiya M., Sakurai T. An On-Chip Characterizing System for With-in-Die Delay Variation Measurement of Individual Standard Cells in 65-nm CMOS // Proceedings of the ASP-DAC. 2011. P. 109-110.

- [9] Inagaki K., Antono D., Takamiya M. et al. A 1-ps Resolution On-Chip Sampling Oscilloscope with 64:1 Tunable Sampling Range Based on Ramp Waveform Division Scheme // Symposium on VLSI Circuits. Digest of Technical Papers. 2006. P. 61-62.
- [10] Collins M., Al-Hashimi B., Wilson P. On-chip timing measurement architecture with femtosecond resolution // Proceedings of 11th IEEE European Test Symposium. 2006. P.103-110.
- [11] Collins M. On-Chip Time Measurement Architectures and Implementation. Thesis for the degree of Doctor of Philosophy. May, 2009. Southampton, UK. – 160 p.
- [12] Cassard J.M. A Sensitivity Analysis of SPICE Parameters Using an Eleven-Stage Ring Oscillator // IEEE Transactions on Electron Devices. 1984. V. 31. № 2. P. 264-269.
- [13] Masuda H., Ohkawa S., Kurokawa A. et al. Challenge: variability characterization and modeling for 65- to 90-nm processes // Proceedings of the IEEE Custom Integrated Circuits Conference. 2005. P. 593-600.
- [14] Bhushan M., Gattiker A., Ketchen M.B. et al. Ring oscillators for CMOS process tuning and variability control // IEEE Transactions on Semiconductor Manufacturing. 2006. V. 19. № 1. P. 10–18.
- [15] Ketchen M.B., Bhushan M. Product-representative "at speed" test structures for CMOS characterization // IBM Journal of Res. and Dev. 2006. V. 50. № 4/5. P. 451–468.
- [16] Das B.P., Onodera H. Area-efficient reconfigurable-arraybased oscillator for standard cell characterization // IEEE Transactions on Circuits and Systems - II: Express Briefs. 2014. V. 61. № 3. P. 429-436.
- [17] Das B.P., Onodera H. On-Chip Measurement of Rise/Fall Gate Delay Using Reconfigurable Ring Oscillator // IEEE Transactions On Circuits And Systems. 2014. V. 61. № 3. P. 183-187.
- [18] Kindereit U. Fundamentals and Future Applications of Laser Voltage Probing // Proc. of IEEE International Reliability Physics Symposium. 2014. P. 3F.1.1 - 3F.1.11.
- [19] Frohmann S., Dietz E., Dittrich H., Hübers H.W. Picosecond imaging of signal propagation in integrated circuits // Advanced Optical Technologies. 2017. V. 6. № 2. P. 137– 142.
- [20] Dudek P., Szczpanksi S., Hatfield J.V. A high-resolution CMOS time-to-digital converter utilizing a Vernier delay line // IEEE Transactions on Solid-State Circuits. 2000. V. 35. № 2. P. 240–247.

Методы обеспечения переносимости тестовых сценариев между различными верификационными окружениями

А.В. Андрианов

ЗАО НТЦ "Модуль", г. Москва, aerodronich@yandex.ru

Аннотация — При разработке и верификации IP блоков встает задача переноса тестовых сценариев между различными окружениями для проверки корректности блока стадиях работы ня BCCX маршрута В проектирования. статье описаны методы, позволяющие обеспечить переносимость тестовых сценариев между различными верификационными окружениями, такими как: верификационное окружение IР блока, ПЛИС-прототип (под управлением linux и без операционной системы), RTL модель СБИС и реальная микросхема.

Ключевые слова — верификация, СБИС, переносимость, тестовое окружение.

I. Введение

При разработке IP блока и последующей интеграции в СБИС необходимо контролировать всех корректность его работы на этапах проектирования. Некоторые тестовые последовательности работы с блоком являются схожими для различных окружений и при соблюдении некоторых правил могут быть перенесены между различными тестовыми окружениями без каких-либо изменений.

Возможность переноса тестовых сценариев между различными окружениями может существенно сократить время верификации. В первую очередь, это может достигаться за счет исключения необходимости повторной разработки тестовых сценариев под новое Во-вторых, возможность окружение. переноса тестовых сценариев между окружениями упрощает отладку проблем и исключает случаи внесения дополнительных ошибок при ручном переносе тестового сценария, вызвавшего проблему, между окружениями.

В данной статье будет описан ряд методов, которые позволяют обеспечить переносимость тестовых сценариев между окружениями, начиная от самых простых (на уровне программных интерфейсов) и заканчивая гибридным верификационным окружением.

II. ВЕРИФИКАЦИОННЫЕ ОКРУЖЕНИЯ

В этой части статьи приведено краткое описание типичных верификационных окружений, использующихся на разных этапах проектирования СБИС. Так как ошибки могут быть выявлены на различных этапах разработки, возможность быстрого переноса между указанными окружениями может позволить быстро определить причину проблемы (например, ошибка в логике работы блока, интеграции блока или настройке окружения).

А. Верификационное окружение ІР блока

С этого окружения начинается разработка нового IP блока. В зависимости от сложности блока и наличия внешних интерфейсов это окружение может содержать помимо самого IP блока интерфейс системной шины, поведенческую модель памяти. Если это интерфейс передачи данных, то какую-либо ответную часть.

В данном окружении тесты обычно разрабатываются на языках Verilog/SystemVerilog, иногда с применением методологии верификации иvm. Основная цель верификации на данной стадии – проверка корректности работы блока.

В силу отсутствия процессорного ядра использование программных тестов на языках С/С++, описывающих обращение к регистрам устройства, затруднено. Именно в этом окружении необходимо воспроизводить ошибки, связанные с логикой работы разрабатываемого устройства, в случае их обнаружения на более поздних фазах разработки СБИС.

В. RTL модель СБИС

В данном окружении, как правило, присутствует одно или несколько процессорных ядер. На этом этапе возможна разработка тестового сценария на языке Assembler/C/C++, моделирующего работу с блоком. На данном этапе важно, в первую очередь, проверить корректность интеграции IP блока в состав СБИС. Помимо этого, важна проверка возможности доступа DMA контроллером блока (если он имеется) во все области необходимые памяти, правильность подключения к системной шине, корректность подключения к буферам ввода-выода микросхемы, и т.п. Также желательны: проверка типичных сценариев работы с блоком, производительности, тест на максимальное энергопотребление.

Схематично это окружение представлено на рис. 1. Для наглядности рассмотрен случай, когда верифицируемое устройство и процессорное ядро находятся на разных системных шинах, а в системе присутствует несколько блоков памяти, подключенных к разным системным шинам. В виде примера таких СБИС можно привести отечественную СБИС К1879ХБ1Я [1] и СБИС 1888ТХ018 [2]

Верификационное окружение СБИС



Рис. 1. Упрощенная схема верификационного окружения СБИС

С. ПЛИС-прототип

При разработке некоторых типов периферийных устройств трудно обойтись без фазы ПЛИСпрототипирования. ПЛИС-прототип позволяет проверить работу разрабатываемого блока с устройствами, реальными ответными а также разработать полноценный драйвер для операционной системы, которая будет использоваться в дальнейшем на готовой микросхеме. При этом набор библиотечных элементов. таких как буфера ввода-вывода, макроблоки памяти и системная шина. могут отличаться от тех, что используются в реальной микросхеме или в верификационном окружении IP блока.

D. Реальная микросхема

На данном этапе важна возможность быстрого переноса и запуска имеющихся тестов, позволяющая проверить работоспособность первых образцов микросхемы. Помимо этого тестовые сценарии могут служить готовыми примерами для разработчиков программного обеспечения.

III. МЕТОДЫ ОБЕСПЕЧЕНИЯ ПЕРЕНОСИМОСТИ ТЕСТОВЫХ СЦЕНАРИЕВ

А. Обеспечение переносимости на уровне программных интерфейсов (API)

Это один из самых очевидных и простых в реализации методов обеспечения переносимости между окружениями. Однако он требует более тщательного проектирования программного кода для

работы с блоком и сам по себе не может обеспечить переносимости в тестовое окружение IP блока, хотя и существенно упрощает повторное использование кода. Ниже приведены самые основные способы обеспечения переносимости кода на уровне программного интерфейса.

1) Разделение тестового сценария на библиотечную часть и основную

Такой подход позволяет скрыть внутреннюю логику работы с блоком от разработчика и сократить сам тестовый сценарий. Более того, библиотечная часть, описывающая типовые сценарии работы с блоком, может применяться в дальнейшем на итоговой микросхеме, ПЛИС-прототипе и при написании драйверов. Использование системы документирования кода doxygen [3] для библиотечной части также очень желательно.

2) Применение функций-оберток для доступа к регистрам устройства и абстракции над используемым контроллером прерываний.

Использование в коде функций-оберток вместо прямого доступа к регистрам IP блока через указатель языка C/C++ дает ряд преимуществ. Помимо очевидного удобства сбора трассы чтения/записи регистров это позволяет без труда переносить код в окружения, где отсутствует прямой доступ к регистрам блока. Эти обертки можно определить как 'inline' функции, тем самым избегая накладных расходов, вызванных генерацией компилятором вызова подпрограммы с сохранением стека. Пример (для компилятора gcc) представлен на Листинге 1.

| <pre>static inline uint32_t ioread32(uint32_t const base_addr) { return *((volatile uint32_t*)(base_addr));</pre> |
|---|
| <pre>} static inline void iowrite32(uint32_t const value, uint32_t const</pre> |
| base addr) { |

*((volatile *uint32_t**)(**base_addr**)) = **value**;

Листинг 1. Пример функций-оберток для доступа к регистрам устройств

Слой абстракции над контроллером прерываний будет также крайне удобен, так как используемые контроллеры прерываний в ПЛИС-прототипе и проектируемой СБИС могут различаться.

3) Специализированная библиотека выделения памяти для DMA операций

Одной из самых частых задач при верификации IP обладающих встроенным контроллером блоков, прямого доступа к памяти, является проверка возможности обращения DMA всем ко присутствующим в СБИС блокам памятей. В различных окружениях может присутствовать разное количество регионов памяти, доступных для DMA операций, поэтому желательно применять слой абстракции для управления выделением памяти. Это позволит вносить минимум изменений в исходный код тестового сценария при его переносе.

Прибегать к функции выделения памяти из стандартной библиотеки языка С (например, newlib) нежелательно, т.к.

- аллокатор памяти стандартной библиотеки языка С работает только с одной общей областью памяти (heap),
- даже самая простая реализация в библиотеке языка C newlib требует достаточно много времени при моделировании,
- освобождение выделенной памяти в большинстве тестовых сценариев не требуется.

сравнения времени выделения памяти Лля использовалась RTL модель СБИС, содержащая процессорное ядро с архитектурой ARM, работающее на частоте 800 Mhz, SRAM память, и набор периферийных блоков. Динамическое выделение блока памяти в 512 байт, используя реализацию из newlib, занимает около 3800 нс (лучший случай, первый выделяемый блок, фрагментация отсутствует), в то упрощенный линейный аллокатор время как производит выделение памяти за 2000 нс (всегда).

Таким образом, для большинства тестов на RTL модели СБИС достаточно использовать упрощенный аллокатор, который выделяет память линейно из нескольких имеющихся в системе регионов.

Для более сложных задач, где требуется получить максимальную производительность, может возникнуть необходимость воспользоваться специализированным аллокатором памяти [4].

Наличие слоя абстракции над управлением памятью позволит без проблем использовать различные библиотеки для выделения памяти в различных окружениях.

4) Использовать функции-обертки для ожидания при блокирующих операциях

Естественный подход большинства разработчиков – использовать **блокирующий** API для ожидания выполнения длительных операций. Так как на RTL модели СБИС операционная система чаще всего отсутствует, то ожидание отклика аппаратуры в таком окружении производится либо через циклический опрос значения в регистре статуса устройства, либо через ожидание прерывания, которое также может сводиться к опросу общей структуры данных, выступающей здесь примитивом синхронизации.

Упрощенный пример приведен на Листинге 2.

подход может существенно Этот усложнить повторное использование кода при разработке драйверов для операционных систем, так как циклический опрос в большинстве случаев в драйвере недопустим, а во время ожидания отклика оборудования ресурсы процессора используются другими программами.

Решением этой проблемы может стать применение функций-оберток для блокирующих операций, которые впоследствии могут быть легко заменены на функции операционной системы.

Пример переработанного кода из Листинга 2 приведен на Листинге 3. Здесь реализована упрощенная обертка над ожиданием, используя вызов wait_event, идентичный по синтаксису соответствующему вызову ядра OS Linux.

```
void dma_copy(void *to, void *from, size_t size)
{
    iowrite32(from, DMA_FROM);
    iowrite32(to, DMA_TO);
    iowrite32(size, DMA_LENGTH);
    iowrite32(1, DMA_START);
    while(ioread32(DMA_STATUS) &
    DMA_STATUS_WORKING) {};
}
```

Листинг 2. Копирование памяти через DMA, блокирующий API

```
#ifndef LINUX
typedef uint32_t wait_queue_head_t;
#define wait_event(wq, condition) \
    while(!condition) { }
```

#endif

void dma_copy(wait_queue_head_t *wq, void *to, void *from, size_t size)
{
 iowrite32(from, DMA_FROM);
 iowrite32(to, DMA_TO);
 iowrite32(size, DMA_LENGTH);

```
iowrite32(1, DMA_START);
wait_event(&wq, !ioread32(DMA_STATUS) &
DMA_STATUS_WORKING));
```

Листинг 3. Копирование памяти через DMA, блокирующий API с использованием обертки для ожидания

5) Применять неблокирующий программный интерфейс

Функции-обертки для примитивов синхронизации диспетчеризации процессов могут немного упростить портирование кода, но не помогут, если потребуется обеспечить работу В окружении, ориентированном асинхронную на молель проектирования (asynchronous design pattern). Более того, разные операционные системы имеют разные АРІ и примитивы синхронизации.

Использование неблокирующего АРІ позволяет решить эти проблемы.

В случае неблокирующего API разработчикам драйверов достаточно использовать существующий код, обернув его необходимыми примитивами синхронизации, специфичными для операционной системы. К плюсам неблокирующего API стоит также отнести то, что с его использованием проще создавать тестовые сценарии, задействующие сразу несколько устройств, например для тестов на максимальную производительность коммутационной среды и максимальное энергопотребление.

На Листинге 4 приведен код с Листинга 2, переписанный в неблокирующем стиле.



Листинг 4. Копирование памяти через DMA, неблокирующий API с блокирующей оберткой

В. Перенесение программы генерации тестового воздействия на хост-компьютер (гибридное верификационное окружение)

Тщательное проектирование архитектуры библиотек кода для работы с IP блоками может позволить повторно использовать разработанный код в дальнейшем на ПЛИС-прототипе и реальной микросхеме, однако не позволяет быстро перенести сценарий, вызвавший проблему, в верификационное окружение IP блока.

Для достижения этой цели необходимо либо существенно усложнить верификационное окружение этого блока, добавив процессорное ядро и блоки памяти, воспользоваться решениями на основе эмулятора с открытым исходным кодом QEMU (например, решения QEMU Cosim [6], FEMU[7]), либо применить гибридное верификационное окружение [8].

Гибридное верификационное окружение предполагает, что тестовый сценарий компилируется и исполняется независимо от модели СБИС. В этом случае информация об операциях чтения и записи регистров, системной памяти, а также о произошедших прерываниях, передается по TCP/IP соединению или через unix socket. Схематично это окружение изображено на рис.2. Направление обмена данными изображено в виде пунктирной линии.

На стороне моделирования используется VPI расширение языка verilog, которое обеспечивает чтение транзакций из сетевого сокета, и трансляцию их в тестовое окружение СБИС.



Рис. 2. Схема гибридного верификационного окружения IP блока

Преимущество данного метода в том, что тестовый сценарий представляет собой обычное приложение для ПК, а значит разработчику доступны все современные средства разработки и отладки, такие как пошаговая отладка и трассировка. Более того, многие операции, которые на модели СБИС занимают продолжительное время (форматированный вывод, копирование и памяти). исполняются заполнение локально И работают намного быстрее. Технически также можно реализовать возможность перезапуска сценария без перезапуска моделирования и возможность запуска тестового сценария на другом компьютере, нежели моделирование СБИС.

К недостаткам стоит отнести отсутствие прямого доступа к памяти RTL-модели и необходимость всегда использовать слой абстракции для доступа к ресурсам аппаратуры. Для сохранения возможности переноса на верификационное окружение СБИС необходимо избегать использования в тестовых сценариях функций языков С/С++, которые могут быть недоступны в окружении без операционной системы (например, работа с файловой системой, дескрипторами ввода вывода и т.п.).

Этот подход также можно использовать при верификации СБИС, так как он позволяет существенно упростить отладку тестовых сценариев, а также

верифицировать интерфейсы, дающие доступ в адресное пространство СБИС (например, PCI и PCI Express). Схематично такие верификационные окружения приведены на рис. 3 и 4.



Рис. 3. Схема гибридного верификационного окружения СБИС через внутреннюю память

К недостаткам такого окружения стоит отнести то, что тестовый сценарий будет скомпилирован для архитектуры хост-компьютера (как правило, это intel x86/x86_64, порядок байт - little endian). Таким образом порядок байт и размеры некоторых базовых типов данных в языках C/C++ могут отличаться.

Решением данной проблемы может служить либо повышение требований к качеству кода, как уже было упомянуто выше, так и использование эмулятора qemu в режиме "usermode emulation".

Этот режим позволяет запускать программы пространства пользователя linux, скомпилированные под другую архитектуру, как если бы они были скомпилированы для хост-архитектуры.

Такой режим эмуляции требует намного меньше ресурсов, нежели эмуляция всей системы (режим

системной эмуляции QEMU, применяемый решениями FEMU/QEMU Cosim) и позволяет компилировать тестовый сценарий в условиях, максимально приближенных к RTL модели СБИС.



Программа генерации тестового воздействия

Рис. 4. Схема гибридного верификационного окружения СБИС через интерфейс РСІ

С. Применение высокоуровневого скриптового языка для разработки тестовых сценариев

При использовании гибридного верификационного окружения выбор языков программирования для разработки тестового покрытия не ограничивается С/С++. Существует возможность написания тестовых сценариев на высокоуровневых языках, таких как lua, python, perl и любых других.

Применение высокоуровневых языков может существенно сократить временные затраты на написание тестовых сценариев за счет наличия богатого набора библиотек, наличия более гибкого и современного синтаксиса, автоматического управления памятью, также отсутствием проблем а переносимости, связанных с размерами типов данных и порядком байт.

Олнако необходимость переноса тестовых сценариев в окружение без операционной системы для верификации готовых микросхем накладывает существенные ограничения на спектр высокоуровневых языков программирования, которые можно задействовать для решения данной задачи. В этой части статьи приведена информация о трех высокоуровневых популярных языках программирования, которые возможно запустить в окружении без операционной системы, И. соответственно, которые могут быть использованы при верификации СБИС.

популярный Lua это легковесный интерпретируемый язык, ориентированный на встраивание в приложения. Этот язык широко используется в большом количестве проектов от разработки компьютерных игр до прототипирования встраиваемых систем[9] и задач машинного обучения[10]. Интерпретатор языка lua можно скомпилировать и использовать в окружении без операционной системы даже при малом объеме доступной оперативной памяти. Для интерпретатора lua версии 5.1 существует проект lua2с, который лополнительно позволяет транслировать кол. написанный на lua, в код на С, использующий вызовы библиотеки интерпретатора lua.

Python – один из наиболее известных современных высокоуровневых языков общего назначения. Его отличают минималистичный синтаксис, богатый набор стандартной библиотеки и функций огромное количество сторонних библиотек. Проект micropython [11] позволяет работать интерпретатору этого языка программирования в окружении без операционной системы, однако поддерживает далеко не полный набор функций стандартной библиотеки. К преимуществам этого языка стоит отнести также расширения myhdl[12], наличие позволяющего разрабатывать и верифицировать синтезируемые цифровые блоки используя python как основной язык для описания аппаратуры и создания тестов.

Javascript (baremetaljs) – хотя изначально основной нишей данного языка программирования была вебразработка с появлением таких платформ, как node.js, этот язык стал активно применяться в виде языка общего назначения. Проект baremetaljs позволяет использовать данный язык на микроконтроллерах и в окружении без операционной системы, что делает его технически пригодным для использования в виде высокоуровневого языка программирования для написания верификационных сценариев.

IV. Заключение

Представленные в данной статье методы, все вместе или по отдельности, могут позволить разработчикам переносить тестовые сценарии между различными окружениями, сводя к минимуму вероятность внесения дополнительных ошибок, добиваясь повторного использования уже написанного кода на дальнейших этапах проектирования СБИС. Возможность быстрого переноса тестовых сценариев между окружениями может упростить поиск ошибок. Перенос и запуск этих сценариев в верификационном окружении IP блока позволяет учитывать их при расчете суммарного покрытия блока тестами.

Важно отметить, что упомянутые в этой статье методы подходят для верификации периферийных блоков и интерфейсов, однако едва ли будут достаточными для полноценной верификации непосредственно ядра процессора.

Дополнительная возможность применять высокоуровневые языки программирования для верификации IP блоков может ускорить разработку тестовых сценариев.

ЛИТЕРАТУРА

[1] «Отечественная СБИС декодера цифрового телевизионного сигнала К1879ХБ1Я (PDF)», Шевченко П.А., Цифровая обработка сигналов и её применение — DSPA'2011, Москва 2011 URL:

http://www.module.ru/upload/images/1354523271art_dspa2 011_1.pdf (дата обращения 01.04.2016)

- [2] И.П. Филимонова, И.В. Безкоровайный, Д.И. Дрягалкин, Г.О. Чумаченко, В.Ю. Залётов, А.В. Андрианов: «Мультимедийная СНК с процессорными ядрами PowerPC и NMC3», Международный форум «Микроэлектроника 2017», Алушта 2017.
- [3] URL: http://www.doxygen.org (дата обращения: 08.04.2018)
- [4] «Гибридный метод аллокации массивов памяти в аппаратных платформах с разветвленной структурой памяти на базе процессора NeuroMatrix® DSP», С.В. Мушкаев, А.В. Андрианов
- [6] URL: https://www.aldec.com/en/solutions/functional_verification/ gemu co sim (дата обращения 01.04.2016)
- [7] "FEMU: A firmware-based emulation framework for SoC verification", 2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) URL: http://ieeexplore.ieee.org/document/5751510/ (дата обращения 01.04.2016)
- [8] "Методика гибридной верификации СБИС "Системана-Кристалле", Андрианов А.В., Шагурин И.И. "Датчики и системы", 2018г., №2, с. 14-18.
- [9] URL: http://www.eluaproject.net/ (дата обращения 01.04.2016)
- [10] URL: http://torch.ch/ (дата обращения 01.04.2016)
- [11] URL: https://micropython.org/ (дата обращения 01.04.2016)
- [12] URL: http:// myhdl.org/ (дата обращения 01.04.2016)

Methods of Achieving Test Scenario Portability Between Different Verification Environments

A.V. Andrianov

Research Centre "Module" JSC, Moscow, aerodronich@yandex.ru

Abstract —During development and verification of IP cores it is often needed to port test scenarios between different environments to check that the IP core works correctly.

Development and verification of IP cores is done in a set of distinct environments: the IP core testbench, the FPGA-prototype, RTL model of the SoC containing the IP core and finally the real SoC.

These environments differ drastically and porting test scenarios between environments can be very useful during debugging.

The article describes a set of useful methods that can make verification scenarios written in C/C++ or a higher level scripted language become portable between different environments and even be reused later as part of the software development kit for the SoC (SDK).

The very first method of providing portability is a careful design of application programming interfaces (API). Apart from obvious splitting of the test scenarios into a library part and the actual test, using wrapper functions for all register operations, a specialized memory allocation library for managing DMA memory is highly recommended over allocating test data arrays statically. This is especially useful when the SoC has a set of different memory blocks DMA access to which needs to be verified.

Busy-wait loops in code should be either avoided by using a non-blocking API or wrapped in functions, mimicking the operating system scheduling API.

However, careful API abstractions cannot bring the test scenarios written in C/C++ to the verification testbench of the IP core that usually lacks a CPU required to execute them. To avoid complicating the testbench by adding a CPU and memories, or using a QEMU for CPU emulation, a hybrid approach is suggested. In hybrid verification environment the test scenario is executed as a user space process, communicating with the actual simulation over a TCP/IP channel or a unix socket.

This also allows comfortable usage of high-level languages for verification purposes instead of C/C++. To retain portability to the target SoC, languages like python (micropython), lua (elua) and javascript (baremetaljs) are recommended, since they provide a way to run the test scenario in the "bare-metal" environment, without any operating system.

Keywords — SoC, verification, verification environment, portability.

REFERENCES

 «Otechestvennaja SBIS dekodera cifrovogo televizionnogo signala K1879HB1Ja (PDF)», (Digital television broadcast devoder IC) Shevchenko P.A. Cifrovaja obrabotka signalov i ejo primenenie — DSPA'2011, Moscow 2011 URL: http://www.module.ru/upload/images/1354523271art_dspa2

011_1.pdf (access date: 01.04.2016)

- [2] I.P. Filimonova, I.V. Bezkorovajnyj, D.I. Drjagalkin, G.O. Chumachenko, V.Ju. Zaljotov, A.V. Andrianov: «Mul'timedijnaja SNK s processornymi jadrami PowerPC i NMC3», (Multimedia SoC with PowerPC processor cores and NMC3 DSP), Mezhdunarodnyj forum «Mikrojelektronika 2017», Alushta 2017.
- [3] URL: http://www.doxygen.org (access date: 08.04.2018)
- [4] «Gibridnyj metod allokacii massivov pamjati v apparatnyh platformah s razvetvlennoj strukturoj pamjati na baze processora NeuroMatrix® DSP», ("Hybrid method of array allocation in branched memory organisation platforms based on NeuroMatrix® DSP") S.V. Mushkaev, A.V. Andrianov
- [6] URL: https://www.aldec.com/en/solutions/functional_verification/ gemu_co_sim (access date: 01.04.2016)
- [7] "FEMU: A firmware-based emulation framework for SoC verification", 2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) URL: http://ieeexplore.ieee.org/document/5751510/ (access date: 01.04.2016)
- [8] "Metodika gibridnoj verifikacii SBIS "Sistema-na-Kristalle", (Hybrid "System-On-Chip" verification methodology) Andrianov A.V., Shagurin I.I. "Datchiki i sistemy", 2018., №2, p. 14-18.
- [9] URL: http://www.eluaproject.net/ (access date: 01.04.2016)
- [10] URL: http://torch.ch/ (access date: 01.04.2016)
- [11] URL: https://micropython.org/ (access date: 01.04.2016)
- [12] URL: http://myhdl.org/ (access date: 01.04.2016)

Верификация алгоритма арбитража потоков запросов к памяти

М.Е. Барских, О.И. Эсула

Научно-исследовательский институт системных исследований РАН, г. Москва, barskikh@cs.niisi.ras.ru, olgaesula@rambler.ru

Аннотация — Рассмотрен опыт верификации алгоритма арбитража потоков запросов к памяти от нескольких устройств, проведённой с помощью UVM (Universal Verification Metodology). Описана организация используемого тестового окружения. Представлены преимущества выбранного способа верификации. Обосновывается возможность повторного использования созданного тестового окружения.

Ключевые слова — верификация, арбитраж, QoS, UVM.

I. Введение

Основной функцией контроллера памяти микропроцессора является обеспечение доступа различных устройств к внешней памяти. Одновременно в память может быть направлено несколько запросов, но исполняться они будут по порядку, определяемому арбитром. В соответствии с алгоритмом арбитража выставленным на входы арбитра запросам назначаются приоритеты, что обеспечивает доступ к памяти только одному запросу. При выборе алгоритма арбитража запросов в память учитываются требования технического задания. а именно, пропускная способность подсистемы динамической памяти, тактовая частота работы контроллера памяти, перечень устройств, обращающихся к памяти, и их требования к ширине пропускания канала памяти.

При разработке микропроцессора 1890ВМ9 (далее упоминается только номер процессора, без серии) с архитектурой КОМДИВ64 оказалось неприемлемым использовать алгоритм арбитража потоков запросов, реализованный для микропроцессора предыдущей версии BM8. Так, на ПЛИС-прототипе при разрешении 1024х168 и частоте памяти DDR3 75МГц при интенсивном обращении нескольких устройств к памяти на экране наблюдалось непериодическое дрожание изображения. Пропускная способность канала памяти была повышена за счёт изменения алгоритма арбитража, а для операций вывода на экран установлен более высокий приоритет. По этой причине возникла задача тщательной проверки алгоритма работоспособности арбитража, разработанного для микропроцессора ВМ9.

Тесты основного маршрута верификации, применяемых для проектов НИИСИ РАН, являются программами, выполняемыми процессором. В случае проверки алгоритма арбитража запросов в память этот подход является сложным и медленным в силу следующего ряда причин:

• тест должен запрограммировать контроллеры всех устройств, обращающихся к памяти, подготовить дескрипторы и данные в память;

• в проекте должна быть модель этих устройств, имитирующая их поведение;

• тест должен идти достаточное время, чтобы все устройства выполнили требуемые обращения в память.

На время тестирования, кроме длительности самого теста, влияет скорость моделирования, которая будет невысокой из-за объема и сложности моделируемого проекта. Поэтому было принято решение в качестве мастеров на шине контроллера памяти использовать UVM-агенты (рис. 1), а их поведение описывать используя библиотеку последовательностей настраиваемых запросов. В качестве языка реализации выбран Specnam *e*, который, с одной стороны, позволяет использовать методологию UVM, а с другой стороны, объем исходного кода агентов и всего тестового окружения по сравнению с UVM-SV (Universal Verification Metodology SystemVerilog) у него меньше.



Рис. 1. Подключение мастеров к шине контроллера памяти (MC – контроллер памяти; DDR – модель внешней памяти; USB, SATA – контроллеры USB, SATA и их модели; eUVM – UVM-агенты; seq lib – библиотека последовательностей настраиваемых запросов)

II. АЛГОРИТМ АРБИТРАЖА ЗАПРОСОВ В ПАМЯТЬ

Задача распределения приоритетов между несколькими одновременно выставленными запросами

к одному ресурсу имеет известные решения [1]. Реализация арбитража с помощью фиксированных требует минимум приоритетов ресурсов, но гарантирует доступ запросу только с самым высоким приоритетом. Алгоритм арбитража Round-Robin [2] обеспечивает равные условия доступа всем запросам, образом вообще не учитывается таким их приоритетность. В микропроцессоре ВМ8 из-за большого количества устройств, обращающихся в память, реализовано 2 уровня арбитража [3]. На первом уровне вне контроллера памяти используется алгоритм Round-Robin, а на втором для арбитража запросов внутри контроллера памяти реализован алгоритм распределения приоритетов Least Recently Used (LRU) [4]. Реализация LRU не потребовала сложной логики с большим количеством элементов. Этот алгоритм отлаживался с помощью тестов основного маршрута верификации, для прохождения которых оказалось достаточным нескольких часов.

Перечисленные алгоритмы арбитража запросов к памяти не позволяли обеспечить необходимый уровень качества обслуживания устройств (Quality of Service, QoS) во всех режимах, требуемых в техническом задании для микропроцессора BM9. По этой причине был реализован усложнённый механизм LRU в области предоставляемой устройству ширины полосы пропускания [3], кратко описанный ниже.

Контроллер памяти микропроцессора ВМ9 имеет 7 каналов, работающих по протоколу AXI3 (Advanced eXtensible Interface). Каждому каналу соответствует определенный приоритет от 1 до 7, меняющийся по алгоритму LRU. В любой момент времени все приоритеты разные. Дополнением к алгоритму LRU является присвоение каждому каналу контроллера памяти некоторого числа (обозначается далее n_i, где i – номер канала). n_i определяет количество запросов, передаваемых по каналу і и имеющих неизменный приоритет. Ниже описан механизм изменения n_i для канала і и его влияние на значение приоритета. При одновременно выставленных запросах по каналу і и какому-то другому каналу сравниваются приоритеты каналов. Если приоритет канала і ниже, то значение увеличивается. Иначе приоритета запросу предоставляется доступ в память, при этом приоритет остаётся неизменным, а n_i = n_i - 1. Когда n_i достигает 0 приоритет становится минимальным, а n_i принимает начальное значение, которое является настраиваемой величиной и задаётся программистом. Реализованный алгоритм арбитража позволяет изменять ширину полосы конкретного канала в зависимости от выполняемой вычислительной задачи в процессе работы.

III. Повышение эффективности верификации

Верификация правильности RTL-описания реализованного алгоритма арбитража запросов в память проводилась на основе плана тестирования и управлялась тестовым покрытием. План тестирования содержал 3 раздела:

1) Интерфейсная часть: контроль корректности реализации входных и выходных интерфейсов контроллера Так использовался памяти. как стандартный интерфейс АХІЗ, который реализован в большом количестве блоков и проектов, и его верификация не являлась основной частью описанной в статье работы, то данный раздел содержал в основном ссылки на формальные утверждения (assertion). Для контроля корректности использовалась библиотека ARM [5].

2) Функциональное ядро: описание тестирования именно механизма арбитража, учитывающее особенности его реализации (т.е. блок рассматривался как «белый ящик» – *White Box*). Это позволяло перечислить состояния RTL-модели, которые должны быть достигнуты во время тестов.

3) Специальные случаи: здесь описывались так называемые «corner case» и критические ситуации, которые должны быть покрыты тестами. Часть из этих случаев может быть описана во втором разделе, но здесь они перечисляются именно с целью выделить те особые ситуации (в запросах и/или поведении контроллера), которые необходимо проконтролировать «вручную».

Для описания формальных утверждений (assertion) и функционального покрытия (functional coverage) выбран язык SystemVerilog, так как его использование упрощает проверку при выполнении тестовых программ в рамках основного маршрута верификации.

План верификации организован как древовидная структура записей, уточняющихся по мере углубления уровня вложенности, и завершающаяся записью, которая имеет ссылку на пункт технического задания.

В соответствии с планом верификации написан список утверждений и критических ситуаций. Т.е. каждой завершающей записи плана верификации сопоставляется или assertion или пункт functional coverage. Достижение этих точек в процессе тестирования позволяет не только видеть прогресс тестирования, но корректировать тестовые И воздействия для скорейшего достижения нужного результата. Тест может корректироваться двумя способами: или за счет добавления новых тестовых последовательностей (например, пакетная передача данных с декрементом адреса) или за счет изменения ограничений (constrain) на генерацию параметров. Полное выполнение тестового плана служит объективным критерием окончания процесса тестирования.

Утверждения разделены на две группы – проверка корректности выдачи ответов на внешние воздействия по протоколу AXI3 и проверка внутренних критических ситуаций, связанных непосредственно с работой алгоритма арбитража запросов в память. Все необходимые для описания внутренних ситуаций сигналы находятся в одном модуле, поэтому решено разместить эти утверждения в нём же, не выделяя их в отдельный файл. Утверждения, проверяющие обмен информацией по протоколу AXI3, размещены в тестовом окружении (рис. 2).

Для проверки корректности работы алгоритма арбитража применен уже известный набор утверждений [6]. Например, любой выставленный на вход контроллера памяти запрос должен быть обработан или нельзя выдавать разрешение на обработку более, чем одному запросу. Также описаны критические ситуации, соответствующие данной реализации. Например, пока число n_i не нулевое, соответствующий каналу i приоритет не меняется.



Рис. 2. Схема размещения проверочных утверждений. test_env – тестовое окружение; wr/rd_mon[i] – мониторы UVM-агентов канала записи/чтения, i = 0, ..., 6; checker – проверочные утверждения; arbiter – модуль, описывающий алгоритм арбитража запросов в память; cov & assert – набор критических ситуаций и проверочных утверждений

IV. ОРГАНИЗАЦИЯ ТЕСТОВ

Внешними воздействиями для контроллера памяти являются запросы, посылаемые 7 UVM-агентами по независимым каналам записи и чтения, работающим по протоколу AXI3. Все запросы формируются при помощи шаблона запроса, у которого есть набор необходимых параметров (табл. 1). Перечень параметров шаблона запроса

Таблина 1

| Название параметра | Описание параметра |
|-----------------------|--|
| Па | раметры запроса |
| id | Идентификационный номер |
| unalign | Признак, что адрес не выровнен |
| | (необходим для расчёта значения маски данных) |
| size | Число, определяющее |
| | размерность шины данных (= 2 ^(size+3)) |
| length | Число, определяющее количество фаз данных (= length+1) |
| address | Адрес |
| mask | Маска данных |
| data | Данные |
| Параметры, регулир | ующие плотность выдачи запросов |
| delay_before_address | Длительность паузы перед |
| dalay bafora data | адресной фазой |
| ueray_berore_uata | длительность паузы перед |

delay in data Длительность пауз между фазами данных Проверка корректности реализации интерфейсов контроллера памяти осуществлялась с помощью изменения параметров запроса (длина, адрес, данные и др.). В шаблоне кроме логических параметров собственно запроса присутствуют параметры. отвечающие на задержки в выдаче запросов. Эти параметры учитываются драйвером шины и дают возможность в тесте регулировать загрузку шины, что делает тестовые сценарии более гибкими и разнообразными. На рис. 3 продемонстрирован пример части теста, состоящего из трёх запросов по записи со

случайными параметрами плотности выдачи запросов.

началом передачи данных



Рис. 3. Диаграмма нагрузки шины записи тремя запросами с разными параметрами плотности выдачи запросов. clock – сигнал тактовой частоты, AW channel – адресная фаза запроса по записи, W channel – фаза данных, T1 – временной интервал между запросами (между фазами передачи адреса двух разных запросов), T2 – временной интервал до записи данных (между фазами передачи адреса и данных одного запроса), T3 – временной интервал между фазами передачи данных одного запроса), T3 –

Последовательности настраиваемых запросов определяют входные потоки запросов для тестируемого модуля. Сами последовательности в свою очередь

реализованы в виде шаблонов в библиотеке, к которой обращается тест при формировании тестового сценария (рис. 4). В соответствии с методологией UVM шаблоны

реализованы иерархически, позволяя формировать несколько запросов в соответствии с заложенными в них алгоритмами. Для проверки работоспособности алгоритма арбитража запросов в память выбраны три типа последовательностей:

1) SINGLE – последовательность, состоящая из одного запроса;

 MULTIPLE – последовательность, состоящая из нескольких запросов (по умолчанию 16), все запросы имеют разный размер и обращаются по разным адресам;

3) BURST - последовательность, состоящая из нескольких запросов (по умолчанию 16), имеющих одну длительность и идущие «подряд» адреса.



Рис. 4. Схема создания теста (item – шаблон настраиваемого запроса; seq lib – библиотека последовательностей запросов; test - тест; write/read_driver[i] – драйвера независимых каналов записи или чтения для каждого UVM-агента, i = 0, ..., 6)

Тест считается успешно прошедшим, если данные по чтению совпали с ранее записанными (или теми, что инициализировали память), и при этом не было ошибок в ситуациях, описанных утверждениями. Механизм сравнения данных сделан универсальным для трёх значений параметра протокола AXI3 «size» = 4, 3, 2 [7], так как именно они используются при имитации входных запросов в память. Рассматриваемое в статье тестирование направлено не на проверку когерентности данных, а на проверку алгоритма арбитража запросов в память. Т.е. проверяется не арбитраж обращения нескольких устройств к одной области памяти, а только разделение доступа к ней. Поэтому для упрощения генерации тестов решено каждому UVM-агенту выделить свой диапазон адресов с учётом структуры DDR, размера страниц и банков памяти. Кроме того, принятые меры позволили упростить механизм сравнения данных по записи и чтению, что позволило ускорить процесс верификации.

Более полное покрытие RTL-модели тестами обеспечено использованием виртуальных последовательностей, позволяющих верификатору контролировать выбор типа последовательностей запросов и параметры запросов, а также другие настройки тестового сценария.

V. ПРОЦЕСС МОДЕЛИРОВАНИЯ

Моделирование проводилось с помощью симулятора Cadence Incisive Enterprise Simulator. Запуск

тестов осуществлялся с помощью программы Emanager, входящей в комплект поставки. В командном файле выбирались тесты UVM, требуемые для запуска, и параметры самого запуска (такие как количество итераций и значение ядра случайного генератора (seed) для синтезатора). В Emanager также велась статистика запусков. Помимо контроля ошибочных тестов и регрессионного тестирования это позволяло собрать и объединить покрытия (покрытие кода и функциональное) для различных запусков.

На основе анализа собранной информации менялись ограничения для запросов и сценарии тестов. Используемые средства создания тестовых сценариев и оценки функционального покрытия позволили добиться высокого качества отладки.

VI. РЕЗУЛЬТАТЫ

Описанный в работе пример верификации позволил обнаружить 7 ошибок в RTL-модели на языке Verilog алгоритма арбитража запросов в память лпя микропроцессора ВМ9. Также 5 ошибок в реализации механизма расщепления запросов при наличии перехода через ранки памяти. Кроме того обнаружены ещё 4 ошибки в контроллере памяти: первая - при работе в режиме включения кода Хемминга, вторая - в блоке, меняющем формат запроса с АХІЗ на понятный контроллеру памяти, третья – при имитации запросов с параметром «size» = 3 и случайными масками. Последняя ошибка проявлялась при тестировании проекта на ПЛИС (Программируемая Логическая Интегральная Схема) фирмы Altera, но точного места ошибки не удавалось найти в течение нескольких дней. Благодаря коротким, управляемым и возобновляемым тестовым сценариям, описанным выше, по указанным разработчиком параметрам запросов на поиск ошибки понадобилось 4 часа. При этом большая часть времени ушла на воссоздание требуемой ситуации.

Реализованный алгоритм арбитража позволил решить поставленную задачу повышения пропускной способности канала памяти с приоритетом для операций вывода на экран. На ПЛИС-прототипе при разрешении 1024х168 и частоте памяти DDR3 75МГц при интенсивном обращении нескольких устройств к памяти на экране более не наблюдалось дрожание изображения [3].

VII. ПРЕИМУЩЕСТВА

Можно выделить три основных преимущества представленного в статье способа верификации по сравнению с тестированием проекта на ПЛИС фирмы Altera:

1) Сокращение времени, затрачиваемого на поиск ошибки, с 13 часов до 4 часов, так как невыполнение хотя бы одного утверждения из всего набора сразу останавливает процесс тестирования;

2) Уменьшение объёма используемой памяти (рис. 5) за счёт того, что моделировалась не вся система целиком, а только её часть и тестовое окружение в виде UVM-агентов и библиотеки; Сокращение времени моделирования благодаря использованию коротких направленных тестов и моделированию только интересующей части системы, как и для предыдущего пункта.



Рис. 5. Сравнение объёма используемой памяти. 1 тестирование проекта на ПЛИС фирмы Altera, 2 тестирование проекта описанным в статье способом

VIII. ЗАКЛЮЧЕНИЕ

Необхолимость верификации аппаратной реализации нового алгоритма арбитража запросов в память для микропроцессора ВМ9 привела к созданию тестового окружения с несколькими агентами, работающими по протоколу АХІЗ, и с универсальным механизмом проверки данных чтению. по Использование проверочных утверждений позволило сделать проверку более эффективной. Гибкая система настройки тестовых сценариев дала возможность покрыть все перечисленные критические ситуации.

Проверка представленным способом позволила обнаружить ошибки в обособленных участках проекта за меньшее время в сравнении с основным методом верификации проекта.

Тестовая система организована так, что её использование стало возможным при проверке других частей проекта, и планируется её использование при верификации нового проекта.

ЛИТЕРАТУРА

- Mohan S., Joseph A. A Dynamic Priority Based Arbitration Algorithm // International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2013. V. 3, № 1. P. 232-233.
- [2] Nosrati M., Karimi R., Hariri M.. Task Scheduling Algorithms Introduction // World Applied Programming. 2012. V. 2, № 6. P. 394-398.
- [3] Корниленко А.В., Эсула О.И. Оптимизация подсистемы памяти вычислительной системы с помощью предоставления гарантированной полосы пропускания канала памяти // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №3. С. 136-140.
- [4] Aravind A.A.. An arbitration algorithm for multiport memory systems. // IEICE Electronics Express. 2013. V. 2, № 19. P. 1-7.
- [5] URL: https://silver.arm.com/download/download.tm?pv= 1242915&p=1073545 (дата обращения 21.03.2018).
- [6] Foster H., Krolnik A. Creating Assertion-Based IP // Springer. 2008. P. 318.
- [7] URL: http://infocenter.arm.com/help/index.jsp?topic= /com.arm.doc.ihi0022f.b/index.html (дата обращения 21.03.2018).

Verification of Memory Requests Arbitration Algorithm

M.E. Barskikh, O.I Esula

SRISA RAS, Moscow, barskikh@cs.niisi.ras.ru, olgaesula@rambler.ru

Abstract — Memory controller in microprocessor provides access to external memory for a number of system components. Some requests can be set in the same time. So there is a request arbiter in the memory controller that orders requests by priorities. Arbitration algorithm [1] depends on technical requirements. It was impossible to reuse in microprocessor 1890VM9 the arbitration algorithm that was designed for the previous version 1890VM8. The main verification path in SRISA RAS is a list of programs executed by the processor. This verification path is difficult and slow. The article considers arbitration algorithm verification that uses UVM. Masters on memory controller bus are UVMagents, their behavior is described by using request sequence library. We used Specman *e* because agents' initial code and test environment are less in comparison with UVM-SV.

Microprocessor 1890VM9 has 2 levels of arbitration. The first level uses Round-Robin [2] algorithm. The second level [3] uses Least Recently Used (LRU) [4] arbitration algorithm with some modification that allows to change the bandwidth of a particular memory controller channel depending on the computational task in progress [3]. Arbitration verification was based on a test plan and managed by test coverage. Test plan had references to technical documentation and assertion or functional coverage written in SystemVerilog. The ARM library [5] was used to check correctness. We used well-known set of assertions [6] and project specified assertions to check the correctness of the arbitration algorithm. Achieving of these points allows monitoring the progress of testing and adjusting the test effects to achieve the desired result as soon as possible. The test can be adjusted by adding new test sequences or by changing constraints on parameter generation. Full execution of the test plan serves as an objective criterion for the end of the testing process.

UVM agents sent read or write requests to the memory controller. All requests were generated using a request template that had a set of required parameters. A test is successful if the read data matches the previously written data (or initial memory data) and there are no errors in the situations described by the assertions. Emanager conducted statistics runs. In addition to error test control and regression testing, this allowed us to collect and combine coverages (code coverage and functional) for different runs. Based on the analysis of the collected information, request and test scenario limits were changed.

Verification described in the article allowed to detect 7 errors in the RTL-model of the memory controller request arbitration algorithm. In addition, 9 errors were detected in the memory controller that were not related to arbitration algorithm.

The test system can be reused for verification other part of the project or in other project.

Keywords - verification, arbitration, QoS, UVM

REFERENCES

 Mohan S., Joseph A. A Dynamic Priority Based Arbitration Algorithm // International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2013. V. 3, № 1. P. 232-233.

- [2] Nosrati M., Karimi R., Hariri M.. Task Scheduling Algorithms Introduction // World Applied Programming. 2012. V. 2, № 6. P. 394-398.
- [3] Kornilenko A.V., Esula O.I. Optimizatsiya podsistemy pamyati vychislitel'noj sistemy s pomosch'ju predostavlenija garantirovannoj polosy propuskanija kanala pamyati (Computer memory subsystem optimization by providing guaranteed memory bandwidth) // Problemy razrabotki perspestivnyh mikro- i nanojelectronnyh sistem (MES). 2016. №3. P. 136-140.
- [4] Aravind A.A.. An arbitration algorithm for multiport memory systems. // IEICE Electronics Express. 2013. V. 2, № 19. P. 1-7.
- [5] URL: https://silver.arm.com/download/download.tm?pv= 1242915&p=1073545 (access date 21.03.2018).
- [6] Foster H., Krolnik A. Creating Assertion-Based IP // Springer. 2008. P. 318.
- [7] URL: http://infocenter.arm.com/help/index.jsp?topic= /com.arm.doc.ihi0022f.b/index.html (access date 21.03.2018).

Быстрый алгоритм нахождения доступных вершин графа управления при ограничениях траекторий

А.С. Щербаков

Университет Мельбурна, Австралия, andreas@softwareengineer.pro

Аннотация — Предлагается эвристически обоснованный алгоритм быстрого вычисления множества доступных вершин графа управления программы, допускающий задание произвольного списка "запрещенных" узлов в динамическом режиме. Алгоритм предназначен для ускорения автоматического принятия решений при моделировании и тестировании программно-аппаратных комплексов.

Ключевые слова — граф управления, поиск путей в графе, тестирование программ, моделирование программ, тестирование аппаратно-программных комплексов, направленное тестирование, гамаки, кучи. Heaps.

I. Введение

Тестирование программного обеспечения является одной из наиболее сложных и дорогостоящих задач в процессе разработки аппаратно-программных Несмотря комплексов. на наличие технологий оптимального планирования тестов, в том числе методов и средств направленного тестирования, учитывающих граф управления программы И символические представления условий перехода в ней [1,13], ряд принципиальных недостатков ограничивают возможности достижения высоких показателей покрытия кода и поиска ошибок в ПО. В частности, разрушающие присвоения значений переменных в коде программы, наличие и последовательность которых зависит от конкретного пути выполнения [2,6,7], значительно затрудняют учет возможных ситуаций при различных входных стимулах. Такие последовательности, как правило, не учитываются явно при рассмотрении символических представлений. Доминирующий в настоящее время подход к данной проблеме основан на рассмотрении зависимости порядка выполнения операторов от присвоений значений данных как "черного" ящика, поведение которого описывается рядом сильных и слабых поведенческих гипотез, автоматически генерируемых в процессе итеративном моделирования или тестирования [3,5]. Данный способ фактически игнорирует возможность непосредственного учета участков кода, зависящих друг от друга через присвоения данных. В то же время информация о таких зависимостях может быть достаточно легко получена в процессе тестирования, например, путем одновременного динамического контроля за доступом к адресам памяти для чтения данных и исполняемого

кода. Другим вариантом является инструментирование кода, которое сегодня доступно для большинства платформ. Однако использование зависимостей участков кода при планировании тестов сопряжено с существенной технической проблемой – обеспечением быстрого поиска точек ветвлений в графе управления, измерение пути выполнения в которых может приводить к существенному для тестирования изменению символической формы вычисляемых в программе выражений.

Можно сформулировать указанную проблему в более общем виде как поиск ответа на вопрос о целесообразности изменения траектории в том или ином ветвлении программы. Рассмотрение значений данных абстрактных результатов как последовательности выполнения операторов присвоения приводит к задаче о поиске в графе узлов, доступных из узла v^+ по траекториям, не проходящим через узлы из множества V, и, наоборот, недоступным по траекториям, проходящим из узлов V не через v^+ [9]. При использовании алгоритмов обхода графа для принятия решения о целесообразности изменения траектории в той или иной точке ветвления необходимо затратить время, сопоставимое по порядку с временем выполнения программы, что значительно такого метода. В данной снижает применимость работе рассматривается подход к оптимизации решения данной задачи поиска, основанной на типичной структуре графа управления в виде вложенных блоков с одним входом и одним или выходами (мы будем называть их несколькими термином «квазигамаки»). Предлагается оптимизированный алгоритм учета, использующий возможно, типичную там, где это структуру программы виде совокупности вложенных в квазигамаков.

II. Постановка задачи

Для простоты рассматривается функция $D(v^+, V)$, соответствующая одному из двух составляющих утверждений упомянутой во введении задачи поиска вершин. Для вершины v^+ и множества вершин Vграфа управления программы $D(v^+, V)$ возвращает множество узлов, доступных из v^+ по траекториям, не проходящим ни через одну вершину, принадлежащую V. В дальнейшем изложении мы будем называть вершины, принадлежащие множеству V, вершинамиисключениями. Предполагается, что число вершинисключений сравнительно невелико (O(1)).

Сложность вычисления при каждом обращении к D должна быть по возможности минимизирована в предположении случайности (и равновероятности) каждого выбора параметров. Предполагается, что вспомогательные структуры могут быть построены однократно для каждого графа управления, при этом допустимая максимальная сложность таких построений не должна превышать $O(N \log N)$, где N сумма числа ребер и графов управления. Поскольку перечисление элементов множеств само по себе может вносить сложность O(N) для каждого вычисления D, допускается использовать сжатые представления подмножеств множества-результата (в данном случае это интервалы номеров вершин в соответствии с выбранной в рамках алгоритма нумерацией).

III. ИСПОЛЬЗОВАНИЕ ОСТОВНЫХ ДЕРЕВЬЕВ ДЛЯ ОПТИМИЗАЦИИ НАХОЖДЕНИЯ ДОСТУПНЫХ ВЕРШИН

управления программы, Граф являющийся результатом компиляции исходного кода, написанного на языке программирования высокого уровня, в основном состоит из вложенных квазигамаков. Следует предположить, что, если ввести нумерацию вершин графа управления, в которой непрерывные диапазоны номеров соответствуют квазигамакам, то поиск доступных узлов в графе может быть сведен к перечислению сравнительно небольшого количества диапазонов. Однако такая задача не имеет прямого как одной вершине решения, так может соответствовать неограниченное множество квазигамаков, и нахождение оптимального разбиения которых на диапазоны представляет собой задачу оптимизации. Сложность решения такой задачи при использовании динамического программирования кубическая по отношению к размеру графа управления¹.

Чтобы избежать неприемлемого усложнения вспомогательных построений, в настоящей работе предлагается использовать сопоставимый по эффективности вычисления *D* жадный подход к построению иерархии квазигамаков.

Для разбиения графа управления на множество вложенных структур используется остовное дерево [8] с корнем в его входной вершине. При этом поддерево остовного дерева с корневой вершиной v считается (единственным рассматриваемым) квазигамаком со входной вершиной v. В дальнейшем для обозначения такого поддерева используется термин конус (вершины v). Следует заметить, что построение остовного дерева в общем случае неоднозначно, однако, это не оказывает существенного влияния на эффективность алгоритма. Для обоснования возможности использования остовного дерева для выделения квазигамаков вначале следует показать, что квазигамак полностью покрывается конусом остовного дерева, имеющим ту же входную вершину, независимо от способа построения остовного дерева.

Теорема 1. Пусть G – ориентированный граф; $r \in G$ – его вершина (вход); T - остовное дерево с корнем r, полностью покрывающее G; g - область графа G; $r \notin g$; $v \in g$ – единственная вершина, принадлежащая этой области, в которую входят ребра из (одной или более) вершин, не принадлежащих g. Тогда все вершины области g принадлежат поддереву дерева T с корнем в вершине v.

Доказательство. Для каждой вершины $x \in g$ существует последовательность $[x_i]$ вершин, составляющих путь в дереве T из r в x (где $i=0.k, x_0=r, x_k=x$). Так как x принадлежит. g, а r - нет, то существует вершина $x_i \in g$, в которую входит ребро из x_{i-1} , не принадлежащей g. Согласно условию теоремы, такой вершиной может являться только v (так как множество ребер T входит во множество ребер G). Таким образом, все вершины $x \in g$ являются потомками v в дереве T, то есть принадлежат его поддереву с корнем в вершине v.

Рассматривая соотношение конуса и произвольного квазигамака с той же входной вершиной, можно заметить, что конус является дополнением данного квазигамака (возможно, пустым) множеством других квазигамаков. В среднем ожидается, что (1)включаемые в конус квазигамаки подобны по статистическим свойствам; (2) для группы вложенных квазигамаков в практической программе характерно частое использования общих выходных вершин (самым распространенным примером является выход из программы или функции). Поэтому, по подобию с квазигамаком, следует ожидать малого в среднем числа внешних (не входящих в некоторый конус) вершин, соединенных ребрами графа управления с вершинами, входящими в конус. Экспериментальное подтверждение данной гипотезы являлось одной из целей настоящего исследования.

В предлагаемом алгоритме множество R(x) всех доступных из вершины x вершин графа управления распадается на два подмножества:

$$R(x) = I(x) \cup \xi(x), \tag{1}$$

где I(x) - множество узлов, содержащихся в конусе остовного дерева с входной вершиной x; $\xi(x)$ - множество узлов, не принадлежащих данному конусу, соединенных входящим рёбрами хотя бы с одной вершиной, принадлежащей данному конусу.

Если пронумеровать узлы в порядке обхода остовного дерева поиском в глубину, можно поставить в соответствие каждому конусу непрерывный целочисленный диапазон номеров. В данной работе

¹ Сложность однократных построений в постановке задачи не ограничена, однако уже квадратичная их сложность часто неприемлема при современных вычислительных мощностях.

используется нумерация с инкрементом в фазе прямого перехода по ребру.



Рис. 1. Пример построения остовного дерева на графе управления с вычисленными и для его вершин.

Вычисление множества ξ производится на обратном проходе поиска остовного дерева в глубину путём фильтрации и объединения аналогичных множеств, вычисленных для вершин-потомков в остовном дереве. Кроме того, в ξ добавляются графе вершины-потомки данной вершины в управления, не являющиеся, однако, ее потомками в остовном дереве. При этом по мере перехода по направлению к корню остовного дерева из ξ должны быть удалены вершины, оказавшиеся внутри текущего конуса І. Способ эффективного построения множеств ξ описан в разделе VI. На рис. 1 представлен пример назначения вершинам графа управления значений *I* и ξ. Ребра, входящие в остовное дерево, показаны сплошными стрелками, а не вошедшие в него (и таким образом порождающие новые элементы в ξ) штриховыми.

Вернемся к вычислению D(x,V). В тех случаях, когда конус с входной вершиной x не содержит узловисключений из V, достаточно использовать в его качестве R(x), вычисленное в соответствии с уравнением (1). Если же конус с вершиной x содержит вершину-исключение, вычисленные значения I(x) и $\xi(x)$ игнорируются. Вместо этого вычисляется объединение доступных вершин по всем исходящим из x ребрам, что, по сути, означает «обычный» обход графа управления без какой либо оптимизации применительно к данной вершине. В общем случае получаем

$$D(x, V^{-}) = \begin{cases} I(x) \cup \xi(x), \text{ если } V^{-} \cap I(x) = \emptyset \\ \bigcup_{y \in o(x)} D(y, V^{-}), \text{ иначе} \end{cases}$$
(2)

где o(x) — множество непосредственных потомков вершины x в графе управления, не входящих в конус вершины x в остовном дереве.

В следующем разделе рассматриваются дополнительные меры по снижению вероятности использования второй ветви уравнения (2) при практическом вычисления $D(v^+, V)$.

IV. ВЫЧИСЛЕНИЕ ИНТЕРВАЛОВ ВЕРШИН В АРХИТЕКТУРЕ ПЛАНИРОВАНИЯ ТЕСТОВ

Предполагается использование молуля. реализующего алгоритм вычисления $D(v^+, V)$, в комплексе с планировщиком тестов, использующим стратегию, основанную на оценке целесообразности выбора той или иной ветви потока управления в зависимости от набора входящих в D вершин. Планировщик инициирует запрос на вычисление $D(v^+, V)$, получает результат и на его основе решает вопрос о целесообразности выбора ветви. Хотя в планировщику простейшем случае может возвращаться полностью вычисленное значение D, архитектура более эффективной является С приоритетной очередью интервалов номеров вершин, в которой приоритет соответствует размеру интервала, т.е. количеству вершин в нём. При этом вычисленные согласно уравнению (1) интервалы поступают в очередь без немедленной рекурсии для наследуемых вершин. По мере готовности ресурсов наибольший интервал из очереди поступает как на вход планировщика, так и на вход вычислителя D (для дальнейшего перечисления составляющих D(i)интервалов). При этом потребление вычислительных значительно ресурсов уменьшается благодаря первоочередной обработке наибольших интервалов и исключению покрываемых ими интервалов меньшей длины из дальнейшего рассмотрения.

V. Использование «запасных» двоичных деревьев

В случае, если в конусе вершины х оказывается вершина-исключение, как показано выше, интервал *I*(*x*), вычисляемый при построении остовного дерева, неприменим, и требуется произвести итерацию вершин в глубину дерева. При этом даже одна вершинаисключение может порождать до *d* рекурсий уравнения (1), где *d* - глубина вершины-исключения в остовном дереве относительно х. Следует заметить, что остовное дерево, как правило, отнюдь не является сбалансированным, поэтому оно зачастую имеет глубину порядка общего количества узлов в нём. Применение альтернативных способов построения остовного дерева (например, с учетом весов ребер) может не приводить к положительному результату изза наличия «узких мест» в графе управления. На рис. 2 показан пример графа, в котором из-за исключения узла №8 интервалы, ассоциированные с вершинами, становятся непригодными для вычисления $D(N_{2}j, \{N_{2}8\})$, где j=0...7, в результате необходимо выполнить 8 итераций и фактически произвести обход графа управления без какой-либо оптимизации. Чтобы избежать увеличения сложности предлагается использовать дополнительное двоичное дерево, построенное один раз для данного графа управления. Каждая вершина (кроме листьев) бинарного дерева соответствует диапазону номеров узлов в графе управления, а ее вершины-потомки - разбиению интервала на 2 подинтервала. Листья соответствуют узлам графа управления. Для каждой вершины бинарного дерева вычисляются I и ξ аналогично тому, как это делается для остовного дерева. Пример бинарного дерева показан на рис. 3 (соответствует примеру графа управления, показанному на рис. 2).



Рис. 2. Пример построения (несбалансированного) остовного дерева в случае, когда граф управления представляет собой цепочку гамаков

Конусу в остовном дереве соответствует в общем случае множество ИЗ не более $\log_2(K)$ непересекающихся интервалов в бинарном дереве, где К - число вершин в конусе. В то же время глубина соответствующих конусу поддеревьев также Таким ограничена логарифмом Κ. образом, использование бинарного дерева для вычисления в случае присутствия вершин-исключений в рассматриваемом интервале гарантирует логарифмический верхний предел сложности.



Рис. 3. Пример построения бинарного дерева для остовного дерева графа управления (соответствует примеру, показанному на Рис. 2)

VI. ВЫЧИСЛЕНИЕ МНОЖЕСТВ «ВНЕШНИХ» УЗЛОВ

В вышеприведенном алгоритме использовалось построение множеств ξ путем объединения множеств, построенных для вершин-потомков остовного дерева. Вычисление и хранение отдельных контейнеров для таких множеств внесло бы сложность порядка $O(K \cdot N)$ по времени и по размеру памяти, где К - средний размер множества, N - число вершин. Чтобы vменьшить сложность до ~ $O(N \cdot \log(K))$, применяется куча (heap) [12,13], построенная с применением совместно используемого множеством значений графа выражений [15]. Для представления экземпляра кучи, являющегося результатом операций (добавления элемента, удаления элемента или объединения куч) там, где возможно, вместо копирования структур используются ссылки на уже построенные ранее структуры. В результате сложность определяется количеством вершин графа выражений, которые следует изменить или добавить, имея в наличии граф выражений, построенный для узлов-потомков вершины графа управления. Такая сложность для сбалансированной кучи составляет $O(\log(K))$. Автором разработан код для реализации куч с графом проведенных экспериментах выражений. В сравнивается численная сложность построения множеств ξ для различных типов куч.

1) В используемом алгоритме мы отказываемся от исключения из кучи элементов, номера которых оказываются внутри интервала I(x) при переходе от вершины-потомка к вершине-родителю в остовном дереве графа управления. Это необходимо для обеспечения пригодности ранее построенных подвыражений куч к повторному использованию. Вместо исключения мы производим последующую фильтрацию по минимальному номеру вершины в момент использования выражения для {, то есть используем итерацию элементов кучи с ограничением по минимальному значению элемента. Можно показать, что такая итерация с нижней границей в бинарной куче имеет сложность порядка имеющегося количества элементов, удовлетворяющих нижней границе, то есть не вносит дополнительной сложности.

2) Перед объединением $\xi(a)$ и $\xi(b)$, где *a* и *b* потомки некоторой вершины в остовном дереве, необходимо исключить повторяющиеся элементы (чтобы элементы результата были уникальны). Это может быть сделано без вычисления пересечения куч. Достаточно, обходя остовное дерево в глубину, запоминать для каждой включаемой в $\xi(x)$ вершины *z* соответствующую вершину *x*. Если при этом существует предыдущая *x* (назовем ее *x'*) для той же *z*, то *z* удаляется из аккумулированного в данный момент $\xi(h)$, где *h* – ближайший общий предок *x* и *x'* в остовном дереве (поиск которого имеет сложность порядка O(log *d*), где *d* - глубина остовного дерева).

VII. Достоинства алгоритма с точки зрения интеграции

Предложенный алгоритм вычисления множеств доступных узлов обладает рядом свойств, важных для его реализации в аппаратно-программных комплексах

для тестирования систем управления или сбора данных выполненных на СБИС, в том числе в реальном времени.

1) Задействует простую и быструю независимую элементарную процедуру поиска вершин, удовлетворяющей требованиям *RESTful* интерфейсов [15], благодаря чему легко встраивается в различные параллельные и последовательные архитектурные решения, в том числе распределенные.

2) Алгоритм исключает динамическое выделение памяти переменного размера, что облегчает синхронизацию процессов и распределение ресурсов.

3) Возможна непосредственная привязка вычисляемых множеств вершин к логическим адресам памяти — отсутствие программного инструментирования обращений к памяти увеличивает производительность в 2-5 раз.

4) Возможно применение аппаратных приоритетных очередей для ускорения вычислений (при моделировании фактор возможного ускорения оценивается в ~2.5).

VIII. Обработка вызовов

Тестируемый код обычно содержит вызовы функций, что усложняет обработку графа управления, так как вершина, в которую функция возвращает управление, зависит от вершины, в которой она вызывается. Кроме того, управление может возвращаться в различные вершины, если данная функция вырабатывает исключения или использует дальние передачи управления (в том числе прерывание или завершение работы всей программы) [16,17].



Рис. 4. Пример обработки вызовов функции. Штриховыми линиями обозначены рёбра, относящиеся к вызовам

Вызов функции (инлайнинг). Если при статическом анализе оказывается, что функция вызывается из единственной точки кода, возможно включение тела вызываемой функции в конус вершины вызова при построении остовного дерева.

Вызов функции (общий случай). В предлагаемом методе для каждого экземпляра вызова функции с помощью статического анализа находятся все возможные вершины-получатели управления при выходе из вызываемой функции (в том числе при возможной обработке исключений). Затем в граф управления добавляются ребра от вызывающей функцию вершины ко всем таким вершинам. Таким образом, вызов функции заменяется на условные переходы во все возможные вершины-получатели управления по выходе из нее (условия переходов считаются абстрактными). Чтобы учесть множество доступных из ее входной вершины вершин внутри ее тела (которое соответствует конусу вершины-входа функции), достаточно иметь в графе ребро, соединяющее вершину вызова с входной вершиной тела функции (рис. 4).

Возврат из функции. В зависимости от тактики планирования тестов при наличии в некоторой вершине оператора возврата из функции происходит вычисление D(e,V) либо для всех возможных вершин e- получателей управления (упомянутых в предыдущем пункте), либо динамический выбор одной вершиныполучателя на основании имеющегося или планируемого стека вызовов. Полученные множества (рекурсивно) объединяются с $D(v^+, V)$.

IX. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Для экспериментальной проверки гипотезы о низкой практической сложности предложенного алгоритма был создан специализированный исследовательский код², моделирующий решение задачи вычисления $D(v^+, V)$ для множества тестовых наборов (v^+, V) . Также разработан интерфейс для встраивания вычислителя D в системы тестирования.

Код включает в себя сборшик и вычислитель. На сборщик получает промежуточные входе представления (LLVM IR) кода модулей тестируемой программы, генерируемые компилятором CLANG [18] из исходных кодов. После анализа шаблонов IR и функций с известным сборки (linking) всех промежуточным представлением в общий граф управления (с учетом вызовов функций) строится остовное дерево и множества внешних узловнаследников его конусов. Вычислитель – резидентная программа (сервис), обеспечивает вычисление D по запросам в параллельном режиме.

Х. Эксперименты

молелирования практической сложности Для вычисления D(v+V-)были использованы 60 приложений с открытым исходным кодом. В каждом раунде функция вычислялась для всех имеющихся в графе управления ребер $v \rightarrow u$, при этом полагалось v+, а в качестве V использовались все прямые наследники v, кроме u. Если V оказывалось пустым множеством, ребро исключалось из рассмотрения. Таким образом, моделировались вычисления, необходимые в случае решения задачи о целесообразности единичных альтернаций пути выполнения в каждом ветвлении тестируемого кода.

Эксперименты показывают, что среднее количество внешних по отношению к конусу остовного дерева вершин, являющихся потомками

² https://github.com/andreas-softwareengineer-pro/cfgsearch

вершин, покрытых данным конусом, составляет около 4.5 и не показывает существенного роста при увеличении размера конуса. В двоичном дереве такой показатель больше, около 8.5, наблюдается его примерно логарифмический рост с увеличением размера покрываемых интервалов. В среднем D вычисляется за примерно 21 итерацию при среднем размере графа управления в 1340 вершин. Таким образом, алгоритм обеспечивает достаточно низкую вычислительную сложность для использованной выборки тестируемых программ.

Также было проведено сравнение эффективности различных реализаций контейнеров (куч) для хранения множеств внешних узлов. Несмотря на отсутствие гарантированной сбалансированности и, как результат, весьма низкой ожидаемой производительности для худшего случая, куча *skew heap* [10] показывает многократно лучшее время построения и доступа, чем ближайшие по таким показателям другие исследованные типы куч [11].

XI. Смежные подходы

из известных методов Одним организации комбинирования исполняемых участков кода с учетом зависимости условий от предшествующих присвоений переменных является партиционирование кода [4]. Основная идея метода сводится к разделению кода на участки (партиции) и присвоение номеров каждому участку так, чтобы последовательность исполнения участков с различными номерами могла быть поставлена в соответствие символическому виду выражений условий перехода. Сложность собственно незначительна. партиционирования Однако партиционирования в таком виде есть существенный недостаток – в реальной программе партиций оказывается слишком много, и их комбинаторный перебор при тестировании не может быть осуществлен за разумное время. Такой метод пригоден лишь для разреженных связей по данным, например, если предварительно выделены "интересующие" целевые условия или применяется специально разработанная абстрактная модель потока данных. Предложенный в настоящей работе метод фактически является компромиссом между полным партиционированием кода и динамическим поиском зависимых участков по графу управления. При этом используются структурные особенности программ, написанных на языках высокого уровня.

С другой стороны, предложенный метод является компромиссом в смысле качества структур, на которые разбивается граф управления. Например, структурирование в виде гамаков [14] позволило бы уменьшить число пересечений ребрами графа управления границ конусов остовного дерева, однако в результате дублирования вершин общая ожидаемая сложность поиска оказывается худшей.

XII. ЗАКЛЮЧЕНИЕ

Предложенный алгоритм поиска доступных вершин в графе управления с динамическими

ограничениями служит для ускорения эффективного планирования тестов аппаратно-программных комплексов и ПО общего назначения. Малая средняя сложность алгоритма связана с типичной структурой графов управления и подтверждена экспериментально. Улучшение процедуры разбиения графа в рамках метода является предметом дальнейших исследований.

ЛИТЕРАТУРА

- Patrice Godefroid. Compositional dynamic test generation // Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. New York, 2007.
- [2] Uday Khedker, Amitabha Sanyal, Bageshri Sathe. Data Flow Analysis: Theory and Practice // CRC Press, 2009.
- [3] Chakrabarti A, Godefroid P. Software partitioning for effective automated unit testing // Proceedings of the 6th ACM & IEEE International conference on Embedded software, 2006. - P. 262-271.
- [4] Majumdar R. and Ru-Gang Xu. Reducing test generation with information partitions // Lecture Notes in Computer Science, 2009. – V. 5643/2009. - P. 555-569.
- [5] Cimatti A., Griggio A. Software model checking via IC3 // International Conference on Computer Aided Verification 2012, Berlin, Heidelberg, Springer. 2012. P. 277-293).
- [6] Allen FE, Cocke J. A program data flow analysis procedure. Communications of the ACM. 1976 Mar 1;19(3):137.
- [7] Damaggio, Elio, Deutch, Alin, et Vianu, Victor. Artifact systems with data dependencies and arithmetic. ACM Transactions on Database Systems (TODS), 2012, vol. 37, no 3, p. 22.
- [8] Ложкин СА. Лекции по основам кибернетики. М.: Издат. отд. Фак. ВМиК МГУ им. МВ Ломоносова; 2004.
- [9] Щербаков А.С. Быстрый алгоритм учета зависимостей данных при анализе и тестировании программного обеспечения СБИС // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №2. С. 76-83.
- [10] Sleator DD, Tarjan RE. Self-adjusting heaps. SIAM Journal on Computing. 1986 Feb;15(1):52-69.
- [11] Carlsson S, Munro JI, Poblete PV. An implicit binomial queue with constant insertion time. InScandinavian Workshop on Algorithm Theory 1988 Jul 5 (pp. 1-13). Springer, Berlin, Heidelberg.
- [12] Staples J. Computation on graph-like expressions. Theoretical Computer Science. 1980 Feb 1;10(2):171-85.
- [13] Patrice Godefroid. DART: Directed Automated Random Testing (joint work with Nils Klarlund and Koushik Sen) // Proceedings of PLDI'2005 (ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation), Chicago, June 2005. - P. 213-223.
- [14] Zhang, Fubo et D'Hollander, Erik H. Using hammock graphs to structure programs. // Software Engineering, IEEE Transactions on, 2004, vol. 30, no 4. - p. 231-245.
- [15] Richardson L, Ruby S. RESTful web services. // "O'Reilly Media, Inc.", 2008.
- [16] Grove, D., DeFouw, G., Dean, J., and Chambers, C. 1997. Call graph construction in object-oriented languages. SIGPLAN Not. 32, 10, Oct. 1997. – P. 108-124.
- [17] Callahan, D.; Carle, A.; Hall, M.W.; Kennedy, K., Constructing the procedure call multigraph. Software Engineering, IEEE Transactions on, vol.16, no.4pp.483– 487, Apr 1990
- [18] Lattner C. LLVM and Clang: Next generation compiler technology // The BSD Conference. 2008. P. 1-2.
A fast Algorithm for Finding Vertices Accessible via Constrained Paths in a Control Flow Graph

A.S. Scherbakov

The University of Melbourne, Australia, andreas@softwareengineer.pro

Abstract — Modern techniques of directed software and firmware testing widely use symbolic exploration of conditions for program execution branch control. However, they are usually ignorant of possible influence of data value assignments to the symbolic representation of an expression. Instead, they tend to use behavioral observations for building iteratively converging sets of weak and strong hypotheses. The cause lays in the fact that although data assignment dependency information is relatively easy to extract, deriving reasonable solutions on desirable branch alternation at program branching points is a rather complicated task. One of the main challenges is high complexity of finding branch points that control interdependent assignment/usage code fragments execution patterns in a control flow graph. It requires considering graph vertex search tasks with some control vertices being restricted (in order to avoid repeating already seen paths). We propose a fast and scalable algorithm for finding sets of accessible vertices in a control flow graph being given a start vertex and a dynamically supplied list of excepted vertices. This empirically supported algorithm essentially relies on the typical structure of a program compiled from a source code. We prove experimentally that the expected complexity of each query processing is lower than O(log N), where N is total number of code fragments. The algorithm is based on substitution of a control flow graph with its spanning tree, and considering a set of vertices accessible from somewhere as a union of a vertex interval and a set of outstanding (out-of-cone) vertices. To collect the latter at a low complexity, we propose the usage of specially designed expression-graph powered version of heap containers. We also consider building (more) balanced back-off versions of spanning tree to ensure a lower complexity when walking around an excepted vertex. As well, we consider the alignment of back-off tree to its control flow graph.

Keywords — control flow graph, graph traversing, software testing, software modeling, firmware testing, data dependency, variable assignments, symbolic simulation, directed testing, hammocks, heap, expression graph.

REFERENCES

- Godefroid P. Compositional dynamic test generation // Acm Sigplan Notices. 2007. V. 42. № 1. P. 47-54.
- [2] Khedker U., Sanyal A., & Sathe B. Data Flow Analysis: Theory and Practice. // CRC Press. 2009. 385 p.
- [3] Chakrabarti A, Godefroid P. Software partitioning for effective automated unit testing. InProceedings of the 6th

ACM & IEEE International conference on Embedded software 2006 Oct 22 (pp. 262-271). ACM.

- [4] Majumdar, R., Xu, R. G. Reducing test generation with information partitions // Lecture Notes in Computer Science, 2009. V. 5643/2009. P. 555-569.
- [5] Cimatti A, Griggio A. Software model checking via IC3. InInternational Conference on Computer Aided Verification 2012 Jul 7 (pp. 277-293). Springer, Berlin, Heidelberg.
- [6] Allen FE, Cocke J. A program data flow analysis procedure. Communications of the ACM. 1976 Mar 1;19(3):137.
- [7] Damaggio, E., Deutch, A., Vianu, V. Artifact systems with data dependencies and arithmetic. // ACM Transactions on Database Systems (TODS). 2012. V. 37. № 3. p. 22.
- [8] Ложкин СА. Лекции по основам кибернетики. М.: Издат. отд. Фак. ВМиК МГУ им. МВ Ломоносова; 2004.
- [9] Shcherbakov A.S. A fast algorithm for data dependency tracking in Software and Firmware analysis and testing // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2016. Part2. P. 76-83.
- [10] Sleator DD, Tarjan RE. Self-adjusting heaps. SIAM Journal on Computing. 1986 Feb;15(1):52-69.
- [11] Carlsson S, Munro JI, Poblete PV. An implicit binomial queue with constant insertion time. InScandinavian Workshop on Algorithm Theory 1988 Jul 5 (pp. 1-13). Springer, Berlin, Heidelberg.
- [12] Staples J. Computation on graph-like expressions. Theoretical Computer Science. 1980 Feb 1;10(2):171-85.
- [13] Godefroid P., Klarlund N., Sen K. DART: Directed Automated Random Testing (joint work with Nils Klarlund and Koushik Sen) // Proceedings of PLDI'2005 (ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation),. Chicago, June 2005. P. 213-223.
- [14] Zhang F., D'Hollander E.H. Using hammock graphs to structure programs // IEEE Transactions on Software Engineering. 2004. V. 30. № 4. P. 231-245.
- [15] Richardson L, Ruby S. RESTful web services. // " O'Reilly Media, Inc.", 2008.
- [16] Grove D., DeFouw G., Dean J., Chambers C. Call graph construction in object-oriented languages // SIGPLAN Notes, V. 32. № 10. Oct. 1997. P. 108-124.
- [17] Callahan D., Carle A.; Hall M.W., Kennedy K. Constructing the procedure call multigraph // IEEE Transactions on Software Engineering. Apr 1990. V. 16. №.4. P. 483–487.
- [18] Lattner C. LLVM and Clang: Next generation compiler technology // The BSD Conference. 2008. P. 1-2.

Using Formal Coverage Analyzer for Code Coverage Improvement

Y.A. Tatarnikov

Independent Contractor, yuritatar@gmail.com

Abstract — This presentation summarizes the results of using the tool grounded on formal proof technics (in this case the tool is Synopsys Formal Coverage Analyzer (FCA)) to improve code coverage for two design Blocks.

The goal is to find unreachable coverage constructs (UNRs) in the target design Blocks and remove them from the list of uncovered constructs. The removal of UNRs saves the Designers and Verification Engineers the time needed to achieve high level of code coverage.

FCA became part of the design and verification methodology within our organization following its successful evaluation.

Keywords — Design Verification, Formal Verification, SystemVerilog, SVA.

I. INTRODUCTION

At the end of verification, when you created and debugged all tests from your test plan your block has some level of coverage, but you need to reach $\sim 100\%$ code coverage.

Most popular code coverage metrics are: line, toggle, condition, FSM. Ultimately, you have to have 100% for each of them.

It is responsibility of block Designer to get high level code coverage. For it Designer makes the decisions of: adding some tests OR to exclude some not covered constructs. Verification guy helps him/her providing existing coverage data base and coverage reports, implementing new tests scenarios and/or excluding coverage constructs. This process is iterative and takes pretty much time, because usually you start with thousands of uncovered constructs.

Any means to shorten the list of uncovered constructs are welcome. Fortunately, we can get help from Vendors Formal Proof tools, which usually have the special mode to get unreachable coverage constructs (called UNRs), which cannot be covered by any test for this particular block. One of the tools is Synopsys Formal Coverage Analyzer (FCA), which is part of Verification Compiler [1, 2].

Goal of this paper is to show briefly how to use this tool and which results we got, why we included this tool in our Design flow.

II. USE FLOW FOR GETTING UNRS

• You (verification guy) – run regression with coverage enabled to get Coverage Data Base.

- FCA has Coverage Data Base as its input.
- You provide clock(s), reset(s) and reset durability OR give simulation snapshot, which represents your Design initial state.
- FCA selects uncovered coverage constructs (for line, toggle, condition, FSM metrics) and tries to generate timing diagram, which will cover them.
- If it can not find this construct is uncoverable.
- FCA generates file with all unreachable constructs (UNR).
- You put this file in the command, which reports coverage, to exclude UNRs.

III. WHEN IS PROPER TIME TO DO THIS JOB

For FCA - less uncovered constructs - better.

For Verification Engineer proper time is when you have finished the creation and debugging tests according to your Test Plan. With good, detailed test plan code coverage metrics (line, toggle, condition, FSM) might be on the level 70%-80%.

IV. BLOCKS USED FOR FCA EVALUATION

There are 2 blocks, which were under development at the time of evaluation. Let's call them "block A" and "block B".

In the terms of coverage constructs, block A looks like:

- 1. Lines 1835
- 2. Conditions 473
- 3. Signal bits (for toggle) 27974
- 4. **FSM states 61**

As you can see this Block is pretty small, but functionally not easy.

Block B:

- 1. Lines 43988
- 2. Conditions 13470
- 3. Signal bits (for toggle) 672904
- 4. FSM states 72

Block B looks ~ 20-30 times greater than block A.

V. BLOCKS A AND B COVERAGE BEFORE FCA

Table below shows coverage numbers of both blocks. Fractions xxx/yyy in the table mean: xxx – not covered constructs, yyy- total amount of constructs, (zz%) – percentage of uncovered constructs.

Table 1

| | Block A | Block B |
|------------------|---------------|------------------|
| Lines | 123/1835 (7%) | 7035/43988 (16%) |
| Conditions | 57/473 (12%) | 2377/13470 (18%) |
| T l_(1.4) | 315/27974 (| 9013/672904 |
| Toggle (bits) | 1%) | (1.5%) |
| FSM states | 1/61 (1.5%) | 3/72 (4%) |
| Total | 496/30343 | 18428/730434 |

Code coverage before FCA

Comments to the table 1 (above):

- **toggle** is counted for each bit of each block signal. If bit has both transitions: 0->1 and 1->0 this bit is counted as toggled. If one transition or no transitions no toggle for this bit.
- **condition** is one particular combination of input signals for given expression. If, for example, we have expression in source code (a && b) and for coverage we have 3 combinations of input signals: 11, 01, 10 in this case we have 3 conditions.
- smaller block has better coverage –very usual situation
- most concern to improve coverage has to be about condition coverage UNRs determined by FCA for blocks A and B

| I | a | bl | le | 2 |
|---|---|----|----|---|
|---|---|----|----|---|

| FCA results | | | | |
|------------------|----------|------------|--|--|
| | Block A | Block B | | |
| Lines: | | | | |
| found | 123 | 7035 | | |
| coverable | 93 | 7035 | | |
| uncoverable | 30 | 0 | | |
| Conditions: | | | | |
| found | 57 | 2377 | | |
| coverable | 42 | 131 | | |
| uncoverable | 15 | 2246 | | |
| | | - | | |
| Signal bits: | | | | |
| found | 315 | 9013 | | |
| coverable | 269 | 8925 | | |
| uncoverable | 46 | 88 | | |
| FSM states: | | | | |
| found | 1 | 3 | | |
| coverable | 1 | 3 | | |
| uncoverable | 0 | 0 | | |
| FSM transitions: | | | | |
| found | 1 | 27 | | |
| coverable | 1 | 23 | | |
| uncoverable | 0 | 4 | | |
| Total: | | | | |
| found | 509 | 18455 | | |
| uncoverable | 91 (18%) | 2338 (12%) | | |

Comments to the table above:

- most desired result of FCA is to get **uncoverable** constructs to exclude them from coverage report saving Designer and Verification Engineer time. As you can see relative number of UNR is not impressive, but look from other side: for block B we excluded 2338 coverage constructs!!! Plenty of manual analyzing time saved !
- Designer has to consider each uncoverable construct (especially line and toggle) as the potential source of Design redundancy
- FCA determined minimal amount of uncoverable constructs, because we did not constrain any Design inputs. In reality Design has some interfaces with specific protocols, which cause input ports dependencies, and some input combinations become impossible. It will potentially increase amount of UNRs, but requires additional manual work to be done by creating and debugging some constraints.

VI. COMPUTATIONAL RESOURCES

FCA run was done locally (not on computer farm), on one Linux workstation, with 8 Xeon processors, each with 4 cores. Memory – 32GB.

FCA job used 1 processor with 4 cores, virtual memory – up to 17GB.

For block A elapsed time is ~ 1 day job run.

For block B ~ 7 days job run.

VII. CONCLUSIONS

- It is worth to use remember: found 2338 uncoverable constructs for block B.
- Preparation for FCA run is very minimal ~ 10 min for me.
- FCA job is highly paralleled and running on network can shorten job time.
- This tool became part of the design and verification methodology within our organization following its successful evaluation.
- This presentation is first step in Formal methods use for verification. Next step has been done [3].

ACKNOWLEDGEMENTS

Author thanks my colleagues – Verification Engineers of company SK Hynix memory solutions (San Jose, CA, the USA), who reviewed previous versions of my presentation and applied this approach to get their blocks coverage improvements, sometimes getting even more impressive results.

REFERENCES

 VC Formal Coverage Analyzer User Guide, Version K-2015.09, September 2015, Synopsys

- [2] VC Formal Verification User Guide, Version K-2015.09, September 2015, Synopsys
- [3] Tatarnikov Y., Labib K. Next step of Formal Verification utilization Available at

https://www.synopsys.com/community/snug/snug-siliconvalley/location-proceedings-2018.html (accessed 03.05.2018).

УДК 519.714

Использование формального метода для улучшения покрытия проекта оцениваемого с помощью метрики «code coverage»

Ю.А. Татарников

Независимый контрактор, yuritatar@gmail.com

Аннотация — Данная презентация представляет результаты использования ППП «Формальный Анализатор» компании Синопсис. ППП основан на методах формальных доказательств.Определяются конструкции дизайна, представленного на регистровом уровне, которые не могут быть обнаружены любым тестом. Исключение этих конструкций улучшает показатели тестового покрытия дизайна. Анализ – составная часть технологии разработки логического дизайна СБИС

Ключевые слова — СБИС, формальная верификация, моделирование, RTL, SystemVerilog, SVA.

ЛИТЕРАТУРА

- VC Formal Coverage Analyzer User Guide, Version K-2015.09, September 2015, Synopsys.
- [2] VC Formal Verification User Guide, Version K-2015.09, September 2015, Synopsys.
- [3] Tatarnikov Y., Labib K. Next step of Formal Verification utilization. Available at https://www.synopsys.com/community/snug/snug-siliconvalley/location-proceedings-2018.html (accessed 03.05.2018).



Разработка конструкции гибридных сенсорных мультисборок с элементами радиочастотной идентификации

А.И. Власов, П.В. Григорьев, В.А. Шахнов

Московский Государственный Технический Университет им. Н. Э. Баумана, shakhnov@iu4.bmstu.ru

Статья посвящена рассмотрению Аннотация особенностей конструкции гибридных сенсорных мультисборок элементами радиочастотной с обзор идентификации. Представлен топологий беспроводных сенсорных сетей, на основании которого предложено использование топологии с несколькими главными узлами. В статье дается описание подхода к разработке топологии сенсорной сети с радиочастотной идентификацией, а также описана элементная база сенсорной сети. На основании сравнения беспроводных модулей связи и считывателей разработана структура блока управления сенсорной системы. Представлен вариант компоновки блока схемы усиления и обработки сигнала, состоящей из ячеек преобразователя напряжения, преобразователя интерфейсов и преобразователя сигналов с чувствительных элементов (ЧЭ), собранные на базе микропроцессора Миландр 1986BE93Y. Авторами предложено использование гибридной схемы компоновки, что позволит минимизировать габаритные характеристики датчика. Описана концепция схемы контрольно-измерительного сенсора на основе RFID-метки, как базового элемента разработки линейки сенсоров для контрольноизмерительных защитных систем, что может служить основой для различных модификаций сенсоров, адаптированных к широкому спектру параметров измеряемых сред. В заключении дана оценка областей применения гибридных сенсорных мультисборок с элементами радиочастотной идентификации.

Ключевые слова — сенсорные мультисборки, беспроводные каналы связи, обработка сигналов, беспроводные сенсорные сети.

I. Введение

Беспроводные сенсорные сети (БСС, Wireless Sensor Networks), состоящие из беспроводных сенсоров и управляющих устройств и способные к самоорганизации с помощью интеллектуальных алгоритмов, демонстрируют масштабные перспективы использования для контроля состояния окружающей среды, функционирования производственных и транспортных систем, состояния здоровья человека, учета различных ресурсов и др. [1]. Также они имеют интерфейсы для внешних подключений, что может расширить и без того богатые функциональные возможности этих устройств. В основном они используются как системы мониторинга и контроля. Большим преимуществом БСС является возможность их использования как внутри помещений, так и за его пределами - в окружающей среде [2].

В зависимости от среды передачи сигналов датчики могут быть проводными И беспроводными. Применение проводных систем не всегда эффективно стоимости из-за монтажных высокой и пусконаладочных работ, а также технического обслуживания. Кроме того, в некоторых ситуациях установка проводных датчиков вообще невозможна по технологическим или организационным причинам [3]. Достоинствами беспроводных датчиков являются минимальные ограничения по их размещению, возможность внедрения и модификации сети таких эксплуатируемом объекте датчиков на без процесс функционирования, вмешательства в надежность и отказоустойчивость всей системы в целом при нарушении отдельных соединений между узлами [4]. Кроме того, в некоторых ситуациях установка проводных датчиков вообще невозможна по технологическим или организационным причинам. Достоинствами беспроводных датчиков являются минимальные ограничения по их размещению, возможность внедрения и модификации сети таких датчиков на эксплуатируемом объекте без процесс функционирования, вмешательства в надежность и отказоустойчивость всей системы в целом при нарушении отдельных соединений между узлами [5].

На сегодняшний день вопрос сбора и обработки информации с сенсорных элементов мультисборок является одним из основополагающих направлений исследований ведущих зарубежных научных институтов. Опубликован ряд работ зарубежных авторов, посвященных вопросу применения беспроводных каналов связи в различных датчиках при мониторинге объектов. Тем не менее, универсального решения, применимого в любых условиях и обладающего высокой точностью, на настоящий момент не получено, поэтому данная область актуальна для проведения дальнейших научных исследований [6-8].

Таблица 1

Достоинства и недостатки топологий БС

| Критерий | Сетевая топология с одним главным | Сетевая топология с несколькими главными |
|------------|---|---|
| сравнения | узлом | узлами |
| 1 | 2 | 3 |
| Надежность | Выход из строя главного узла выведет из | Выход из строя одного из главных устройств не |
| | строя всю сеть | выведет сеть из строя, т.к. его функции может |
| | | выполнять другое устройство |
| Стоимость | Сложное, мощное вычислительное | Имеется возможность распределить часть |
| | устройство в одном узле – высокая | функций по более простым устройствам, что |
| | стоимость | снижает стоимость |
| Энергопо- | Установка мощных приемо-передающих | Устройства работают только в момент передачи |
| требление | устройств на базовой станции. Высокий | или приема данных, что уменьшает |
| | расход энергии | энергопотребление до 90% |
| Масштаби- | Усложнена из-за лимита приёма-передачи | Поддержка тысяч индивидуальных устройств. |
| руемость | информации по расстоянию главным узлом | |

II. МЕТОДИКА ВЫБОРА ТОПОЛОГИИ СЕНСОРНОЙ СЕТИ

При разработке структуры сенсорных сетей важную роль играет архитектура и топология построения сети.

Топологии построения сети можно разделить на 2 группы:

- с одним главным узлом (single-hop);
- с несколькими главными узлами (multi-hop).

Достоинства и недостатки каждой топологии приведены в табл. 1. На рис. 1 представлена графическая интерпретация топологий сенсорных сетей.



Рис. 1. Графическая интерпретация топологий сенсорных сетей

Одни из последних достижений в этой области продемонстрировал проект «SmartDust» (умная пыль), который нацелен на создание сверхминиатюрных устройств – узлов сенсорной сети [9].

III. Разработка топологии сенсорной сети с радиочастотной идентификацией

Для реализации топологии сенсорной сети с несколькими главными узлами, было выделено несколько категорий узлов:

- центральный узел;
- узел-координатор;
- узел-сенсор.

Центральный узел соединен с узломкоординатором проводной связью для обеспечения надежности передачи данных. Узел-координатор разворачивает главную подсеть и инициализируется для приема данных. Узлы-сенсоры, попавшие в зону устойчивого приема главной сети или подчиненной подсети, подключаются к узлу-координатору.

Центральный узел занимается сбором, обработкой и передачей данных, собранных с помощью сенсорной сети. Этот элемент сети является самым важным, так как в нем хранится вся информация, собранная в ходе работы.

Узел-координатор выступает в роли маршрутизатора. Его основная цель – сбор данных от узлов-сенсоров и передача в центральный узел. Общая схема приведена на рис. 2.



Рис. 2. Структурная схема сенсорной сети

Узел-сенсор выступает в роли датчика, он собирает параметры об объекте исследования и передает на узел-координатор. Для увеличения энергоэффективности датчик (его коммуникационный модуль) должен находиться в режиме сна до тех пор, пока к нему не обратятся.

IV. Разработка структуры блока управления сенсорной сети

Вопросы проектирования блока управления сенсорных систем для различных условий эксплуатации подробно описаны в литературе [10]. Для создания более гибкой системы применена модульная структура блока управления. На рис. 3 приведена обобщенная структурная схема узласенсора [11].



Рис. 3. Структурная схема узла-сенсора

В качестве беспроводного протокола передачи данных предложено применить протокол ZigBee, поскольку при малом энергопотреблении дальность передачи данных достигает 200 метров в помещениях и 500 на открытой местности, а скорость передачи 250 кбит/с. Малое энергопотребление достигается за счет использования двух режимов работы. В одном режиме устройство работает при передаче данных, в другом – устройство может быть активно всегда или находиться в спящем состоянии [12].

Каждый узел сети может выполнять две функции: он либо собирает данные с датчиков или исполняет роль коммутатора узлов. Данная схема позволяет гибко настраивать сенсорную систему под конкретные нужды в производстве.

Конечная структурная схема сенсорной сети с радиочастотной идентификацией показана на рис.4.

Для обеспечения автономности работы датчика и RFID-антенны необходимо использовать микросхему, которая сможет питаться пассивно от RFID-антенны. Это позволит не заботиться о питании к датчику и повысит энергоэффективность всей системы.

Для связи между центральным узлом и узломкоординатором (1) будем использовать технологию Ethernet, это позволит увеличить пропускную способность канала, по сравнению с беспроводной технологией Wi-Fi, в 10 раз. Узел-координатор связан с узлом-сенсором (2) посредством технологии ZigBee. Проблемным местом такой связи является ограниченное расстояние опроса узлов-сенсоров, для этого узлы-сенсоры могут принимать информацию от других узлов-сенсоров и передавать к следующему. Связь (3) между узлом-сенсором и считывателем происходит по протоколу UART. Для обеспечения связи между считывателем и RFID-антенной используется радиоканал на частоте 868 МГц [13].



Рис. 4. Структурная схема сенсорной сети с радиочастотной идентификацией

После ответа RFID-метки выполняется временное соединение ее с базовой станцией для дальнейшего обмена данными. Как только соединение произошло, ожидается передача пакета данных, содержащего идентификационный номер RFID-метки и информацию об уровне мощности принятого сигнала (RSSI - Received Signal Strength Indicator). Если соединения метки и базовой станции не произошло, базовая станция через некоторое время выдаст вторую команду соединения. После того как был принят пакет данных от одной RFID-метки, базовая станция переходит в режим приема ответа следующей RFIDметки.

Связь (5) между датчиком и RFID-антенной происходит при попадании RFID-антенны в поле излучения считывателя. RFID-антенна заряжает внутренний конденсатор, датчик производит замеры и отправляет цифровой сигнал на RFID-антенну. Далее излучаемый сигнал получает считыватель.

V. Разработка элемента радиочастотной идентификации

Основная функция узлов БСС – опрос датчиков и передача измерений центрам БСС. Исходя из анализа областей применения, можно выделить используемые БСС типы датчиков:

- пассивные, всенаправленные датчики;
- пассивные, узконаправленные датчики;
- активные датчики.

Пассивные, всенаправленные датчики. Эти датчики производят измерения физической величины в одной точке, не воздействуя на окружающую среду, поэтому и называются пассивными. Кроме того, многие датчики этого типа не нуждаются в питании, потребляя энергию окружающей среды, питание нужно только для усиления их сигналов. Среди таких датчиков: термометры, датчики света, вибрации, микрофоны, тензодатчики, детекторы дыма, датчики химических соединений и т.д.

Пассивные, узконаправленные датчики. Эти датчики тоже не воздействуют на окружающую среду, но обладают направлением измерения. Примером таких датчиков является камера [14]. Она может производить измерения в одном направлении и быть повернута при необходимости.

Активные датчики. Производят воздействия на окружающую среду для измерения. Среди них: сонары, радары и прочие.

Все перечисленные датчики доступны в различных реализациях. В зависимости от приложения необходимо разрешить противоречие между: точностью, потребляемой мощностью, стоимостью, размером, доступностью. Чаше прочих используются пассивные всенаправленные датчики.

Датчики предназначены для работы в системах автоматического контроля, регулирования и управления технологическими процессами, в приборах измерения уровня, расхода, силы и обеспечивают непрерывное преобразование значения измеряемого параметра в унифицированный токовый сигнал дистанционной передачи [15].

В данной работе основное внимание уделяется миниатюризации, обеспечению термокомпенсации и надежности разрабатываемых датчиков. От оптимизации расположения элементов зависит не уменьшение только занимаемого объема, последующая трассировка проводников и адаптация линий связи, но также и надежность всего электронного модуля, что объясняется наличием взаимного влияния (например, теплового) соседних элементов друг на друга. А в случае нежелательного теплового влияния на чувствительный элемент датчика давления, появляется дополнительная температурная погрешность измерений [16]. Поэтому актуально решение задачи по определению такого взаимного расположения электронных элементов, которое было бы оптимально по принятым критериям. Основными метрическими критериями являются минимальная суммарная длина межсоединений, минимальная площадь (объем) области размещенных элементов, а также их производные. К топологическим критериям относят такие критерии, которые учитывают взаимное расположение элементов И соединений на коммутационном поле, например: близость расположения друг к другу тепловыделяющих элементов, минимум числа пересечений соединений и др.

На рис. 5 представлен вариант компоновки блока схемы усиления и обработки сигнала, состоящей из ячеек преобразователя напряжения, преобразователя интерфейсов и преобразователя сигналов ЧЭ, собранные на базе микропроцессора Миландр 1986ВЕ93У.



Рис. 5. Вариант компоновки блока схемы усиления и обработки сигнала, состоящей из ячеек преобразователя напряжения, преобразователя интерфейсов и преобразователя сигналов ЧЭ, собранные на базе микропроцессора Миландр 1986ВЕ93У

Представленный вариант компоновки позволяет разместить все узлы схемы в миниатюрном корпусе, а разнесённая конструкция обеспечивает удобство обслуживания. Данная схема обладает достаточно широким функционалом, но её недостатком является большая масса и габариты (диаметр корпуса – 60 мм, масса датчика примерно 750 г).

Задачи конструкторско-технологического синтеза микромеханических сенсоров, как правило, являются NP-трудными.

Используя технологии производства микросборок, можно минимизировать габаритные характеристики датчика. Также предложен вариант построения датчика, вычислительной схемы и элементов усиления на базе бескорпусного микроконтроллера по гибридной схеме компоновки.

Гибридная схема датчика давления содержит базовые элементы для осуществления полной функции преобразования: мембрану чувствительного элемента со стеклянной подложкой; полную мостовую тензорезистивную схему; электронную схему на усилителях; интегральных операционных для микроконтроллер получения стандартного выходного сигнала, температурной компенсации, балансировки и т.д.



Рис. 6. Конструкция микросборки датчика давления

На рис. 6 представлена конструкция микросборки датчика давления. Конструкция состоит из крышки (1), трубки для подачи давления (2) и микроплаты в виде керамической подложки (3). На подложке (3) также расположены один или два операционных усилителя (4), выполненных на отдельных полупроводниковых пластинах, и мембранный ЧЭ (5), изготовленный в виде отдельного элемента микроконтроллер (6) и необходимые для работы схемы пассивные элементы (7).

Концепция схемы контрольно-измерительного сенсора на основе RFID-метки как базового элемента разработки линейки сенсоров для контрольноизмерительных защитных систем может служить основой для различных модификаций сенсоров, адаптированных к различным параметрам измеряемых сред, различным способам измерения определённого параметра, использованию в различных средах окружающей среды. Например, лавление в трубопроводе можно измерять, во-первых, подсоединившись к содержимому трубопровода, а вочувствительность вторых, используя высокую монокристаллического кремния посредством измерения звукового давления, которое оказывает жидкость или газ на стенки трубопровода, прикрепив трубопровода. сенсор на поверхность Ланная гибридная конструкция интегрируется в конструкцию на рис. 8.



Рис. 8. Эскиз схемы контрольно-измерительного сенсора на RFID-метке (1 – RFID-метка, 2 – плата обработки и передачи сигнала, 3 – антенна RFID-метки, 4 – RFIDчип, 5 – стеклянная подложка чувствительного элемента, 6 – кристалл монокристаллического кремния с элементами измерения параметра и интегральной схемой обработки выходного сигнала)

Учитывая, что за базу конструкторской разработки модификаций сенсоров принимается чувствительный элемент, разработанный на монокристаллическом кремнии, имеющий различный измерительный инструмент, но единую интегральную схему обработки выходного сигнала, можно унифицировать схемы обработки и оцифровки данных и их передачу. Что, в свою очередь, позволит унифицировать детали и сборочные узлы.

Унификация деталей и сборочных узлов упрощает технологический процесс их изготовления и контроля и технологический процесс сборки, тарировки и контроля сенсоров. Позволяет автоматизировать отдельные процессы и обеспечивает максимальное использование мощностей высокоэффективного оборудования.

Заключение

Представлен анализ существующих топологий сенсорных сетей и дано описание основных узлов сети. Предложена топология сенсорной сети с несколькими главными узлами, а также структуры блока управления.

Проведен анализ способов синтеза гибридных сенсорных мультисборок с элементами радиочастотной идентификации.

Описана концепция и представлен эскиз схемы контрольно-измерительного сенсора на основе RFIDметки как базового элемента.

На основании полученных в ходе исследований данных разработана общая концепция сенсорной сети на базе RFID-технологии, получено представление об основных узлах и элементах, входящих в ее состав. Предметом дальнейшего исследования остается взаимная увязка функциональных узлов и элементов сети, реализация коммутационных структур между ними.

Отдельные результаты получены при финансовой поддержке МОН РФ по Соглашению №2.4176.2017/4.6

ЛИТЕРАТУРА

- Andrey I. Vlasov, Anton V. Yudin, Maria A. Salmina, Vadim A. Shakhnov and Konstantin A. Usov Design Methods of Teaching the Development of Internet of Things Components with Considering Predictive Maintenance on the Basis of Mechatronic Devices // International Journal of Applied Engineering Research. — 2017. — Volume 12, Number 20. — PP. 9390-9396.
- [2] Григорьев П.В., Власов А.И. Оценка потенциального объема рынка систем на базе радиочастотной идентификации и его динамика // В сборнике: Энергосбережение и эффективность в технических системах Материалы IV Международной научнотехнической конференции студентов, молодых ученых и специалистов. — Тамбовский государственный технический университет. — 2017. — С. 389-390.
- [3] Власов А.И., Григорьев П.В., Жалнин В.П. Применение методов и средств радиочастотной идентификации в корпоративных информационных производственных системах // Труды Международного симпозиума «Надежность и качество». — 2017. Том 1. — С.272-277.
- [4] Аваева Л.Г., Милешин С.А., Сергеева Н.А., Цивинская Т.А. Математическое моделирование сенсора давления повышенной надежности при эксплуатации в экстремальных условиях // Труды Международного симпозиума «Надежность и качество». — 2017. Том 1. — С.233-237.
- [5] Кальнов В.А., Шахнов В.А., Денисов А.А. Проектирование наносенсоров — М.: Изд-во МГТУ им.Н.Э.Баумана. 2011. Том 6. Сер. Библиотека "Наноинженерия".
- [6] Luís M. L. Oliveira, Joel J. P. C. Rodrigues Wireless Sensor Networks: a Survey on Environmental Monitoring // Journal of communications. — 2011. — Volume 6, Number 2.
- [7] G. Mois, S. Folea, T. Sanislav Analysis of Three IoT-Based Wireless Sensors for Environmental Monitoring // IEEE Transactions on Instrumentation and Measurement. — 2017. — Volume 66, Issue 8.
- [8] M. Arslan, Z. Riaz, S. Azhar Real-time environmental monitoring, visualization and notification system for construction h&s management // Journal of Information Technology in Construction.— 2014. — Volume 19, Page 74.

- [9] B. Warneke, M. Last, B. Liebowitz, K.S.J. Pister Smart Dust: communicating with a cubic-millimeter computer // Computer. — 2001. — Volume 34, Pages 44-51.
- [10] Дшхунян В.Л., Шаньгин В.Ф. Электронная идентификация. -NTPress, Москва, 2004. – 345 с.
- [11] Balanis K. Antenna Theory: Analysis and Design. 2nd Edition. -John Wiley and Sons, 2001. - 68 c.
- [12] Датчики: Справочное пособие / В.М. Шарапов, Е.С. Полищук, Н.Д. Кошевой, Г.Г. Ишанин, И.Г. Минаев, А.С. Совлуков. -Москва: Техносфера. 2012. 624 с.
- [14] Nikitin P. V. and Rao K. V. S. Performance of RFID Tags with Multiple RF Ports. - Proc. Int. Symp. IEEE AP. - June 2007, Honolulu, HI, USA. - C. 5459-5462
- [15] ШарфельдТ. Системы RFID низкой стоимости. Москва, 2006.
 423 с.
- [16] Анфилатов В.С., Емельянов А.А., Кукушкин А.А. Системный анализ в управлении. Учебное пособие / под ред. А.А. Емельянова. — М.: Финансы и статистика. 2002. — 368 с.

Development of Hybrid Sensor Multi-Assemblies Construction with Elements of Radio Frequency Identification

A.I. Vlasov, P.V. Grigoryev, V.A. Shakhnov

Moscow State Technical University N.E. Bauman, shakhnov@iu4.bmstu.ru

Abstract — The article is devoted to the consideration of the peculiarities of the construction of hybrid sensor multiassemblies with elements of radio frequency identification. An overview of the topologies of wireless sensor networks is presented, on the basis of which it is proposed to use a topology with several main nodes. The article describes the approach to the development of the topology of the sensor network with radio frequency identification, and also describes the element base of the sensor network. Based on the comparison of wireless communication modules and readers, the structure of the sensor system control unit has been developed. The variant of arrangement of the block of the circuit of amplification and signal processing consisting of cells of the voltage converter, the converter of interfaces and the converter of signals CHE, collected on the basis of microprocessor Milander 1986E93Y is presented. The authors proposed the use of a hybrid layout scheme, which minimize the overall characteristics of the sensor. The concept of a RFID-based measuring sensor circuit is described as a basic element in the development of a sensor line for monitoring and measuring protective systems, which can serve as the basis for various modifications of sensors adapted to a wide range of parameters of the measured media. In conclusion, an assessment is given of the application areas of hybrid sensor multi-assemblies with **RFID** elements.

Keywords — sensor multi-assemblies, wireless communication channels, signal processing, wireless sensor networks.

REFERENCES

- Andrey I. Vlasov, Anton V. Yudin, Maria A. Salmina, Vadim A. Shakhnov and Konstantin A. Usov Design Methods of Teaching the Development of Internet of Things Components with Considering Predictive Maintenance on the Basis of Mechatronic Devices // International Journal of Applied Engineering Research. — 2017. — Volume 12, Number 20. — PP. 9390-9396.
- [2] Grigoryev P.V., Vlasov A.I. Ocenka potencialnogo obyema rynka sistem na baze radiochastotnoi identifikacii i ego dinamika (Assessment of potential market volume of systems based on radio frequency identification and its dynamics) // V sbornike: Energosberejenieieffektivnost v tehnicheskih sistemah Materialy IV Mejdunarodnoinauchno-tehnicheskoi konferencii studentov, molodyhuchenyh i specialistov. — Tambovskii gosudarstvenny i tehnicheskii universitet.

- [3] Vlasov A.I., Grigoryev P.V., Jalnin V.P. Primenenie metodov i sredst v radiochastotnoi identifikacii v korporativnyh i nformacionnyh proizvodstvennyh sistemah (Application of methods and means of radio-frequency identification in corporate information production systems) // Trudy Mejdunarodnogo simpoziuma «Nadejnost i kachestvo». — 2017. Tom 1. — S.272-277.
- [4] Avaeva L.G., Milewin S.A., Sergeeva N.A., Civinskaya T.A. Matematicheskoe modelirovanie sensoradavleniya povyshennoinadejnostipri ekspluatacii v ekstremalnyh usloviyah (Mathematical modeling of the pressure sensor of increased reliability in operation under extreme conditions) // Trudy Mejdunarodnogo simpoziuma «Nadejnostikachestvo». — 2017. Tom 1. — S.233-237.
- [5] Kalnov V.A., Shakhnov V.A., Denisov A.A. Proektirovanienanosensorov (Designing of nanosensors) — M.: Izd-vo MGTU im.N.E.Baumana. 2011. Tom 6. Ser. Biblioteka "Nanoinjeneriya".
- [6] Luís M. L. Oliveira, Joel J. P. C. Rodrigues Wireless Sensor Networks: a Survey on Environmental Monitoring // Journal of communications. — 2011. — Volume 6, Number 2.
- [7] G. Mois, S. Folea, T. Sanislav Analysis of Three IoT-Based Wireless Sensors for Environmental Monitoring // IEEE Transactions on Instrumentation and Measurement. — 2017. — Volume 66, Issue 8.
- [8] M. Arslan, Z. Riaz, S. Azhar Real-time environmental monitoring, visualization and notification system for construction h&s management // Journal of Information Technology in Construction.— 2014. — Volume 19, Page 74.
- [9] B. Warneke, M. Last, B. Liebowitz, K.S.J. Pister Smart Dust: communicating with a cubic-millimeter computer // Computer. — 2001. — Volume 34, Pages 44-51.
- [10] Dwhunyan V.L., Wangin V.F. Elektronnayaidentifikaciya(Electronic identification) - Press, Moskva, 2004. – 345 s.
- [11] Balanis K. Antenna Theory: Analysis and Design. 2nd Edition. -John Wiley and Sons, 2001. - 68 s.
- [12] Datchiki: Spravochnoeposobie (Sensors: legal aid) / V.M. Warapov, E.S. Politshuk, N.D. Kowevoi, G.G. Iwanin, I.G. Minaev, A.S. Sovlukov. - Moskva: Tehnosfera. 2012. 624 s.
- [13] Nikitin P. V. and Rao K. V. S. Performance of RFID Tags with Multiple RF Ports. - Proc. Int. Symp. IEEE AP. - June 2007, Honolulu, HI, USA. - 5459-5462 s.
- [14] Sharfeld T. Sistemy RFID nizkoistoimosti (RFID low cost systems). Moskva, 2006. $423~{\rm s}$
- [15] Anfilatov V.S., Emelyanov A.A., Kukuwkin A.A. Sistemnyianaliz v upravlenii. Uchebnoeposobie (System analysis in management. Tutorial) / pod red. A.A. Emelyanova. — M.: Finansyistatistika. 2002. — 368 s.

Анализ потребляемой мощности схем суммирования сигналов сопоставления КМОП 65-нм регистров ассоциативной памяти

А.В. Антонюк^{1,2}, П.В. Степанов¹

¹ ФГУ ФНЦ НИИ системных исследований РАН

² Национальный исследовательский ядерный университет "МИФИ", antonyuk@cs.niisi.ras.ru, stepanov@cs.niisi.ras.ru

Аннотация — Проведен анализ потребления мощности двух регистров ассоциативной памяти на основе сбоеустойчивых ячеек STG DICE. Рассмотрены регистры с различными схемами суммирования лвумя комбинационной логической схемой и схемой на основе линии сопоставления. Анализ результатов моделирования показал, что потребление схемы суммирования на основе комбинационной логики зависит от количества N входов схемы, изменивших свое состояние. Потребление схемы суммирования на основе линии сопоставления практически не зависит от N и соответствует потреблению схемы суммирования на основе комбинационной логики при изменении состояний половины входов схемы. Задержка выходного сигнала схемы с линией сопоставления на 22% меньше задержки выходного сигнала комбинационной логической схемы суммирования, а площадь, занимаемая схемой с линией сопоставления на кристалле, меньше на 35%.

Ключевые слова — ассоциативная память, комбинационная логика, логический элемент, моделирование, мощность, проектирование, топология.

I. Введение

Ассоциативная память (content addressable memory - САМ) используется в высокопроизводительных системах поточной обработки данных и в буферах ассоциативной трансляции (translation lookaside buffer -TLB) микропроцессоров. Регистры САМ осуществляют сопоставление хранящихся слов данных с входным словом и состоят из элементов сопоставления (ячеек ассоциативной памяти), элементов маскирования (ячеек маски) и схемы суммирования сигналов сопоставления. Блоки САМ обладают повышенной потребляемой мощностью по сравнению с традиционными ОЗУ из-за одновременного переключения элементов сопоставления и элементов схем суммирования во всех регистрах САМ. Уменьшение энергопотребления схем суммирования - приоритетная задача при проектировании блоков САМ.

Традиционно в регистре САМ сигналы сопоставления суммируются схемой на основе линии сопоставления (match line – ML), к которой подключаются все элементы сопоставления. Выходной сигнал сопоставления регистра формируется в зависимости от напряжения на линии ML. На сегодняшний день существуют схемы суммирования [1], [2] регистров САМ, позволяющие уменьшить потребление в 5 раз и более по сравнению с традиционными регистрами САМ. В работе [1] предзаряд линии ML осуществляется коротким импульсом тока, а дифференциальный усилитель формирует выходной сигнал сопоставления в зависимости от напряжения на ML. В работе [2] предзаряд линии ML производится только при совпадении первых семи бит в регистре, что сокращает мощность, потребляемую цепями предзаряда в накопителе. В статье [3] проанализированы проблемы проектирования регистров САМ с линией сопоставления при увеличении разрядности: увеличение задержек, разделение заряда. Также обоснованы преимущества использования комбинационной логики в регистрах САМ по отношению к традиционной линии сопоставления. В работе [4] представлена модифицированная схема регистра с линией сопоставления, позволяющая снизить энергопотребление и избавиться от проблемы разделения заряда.



Рис. 1. Принципиальная схема элемента сопоставления на основе ячейки памяти STG DICE и логического элемента XOR на основе двух инверторов с третьим состоянием TRInv 1 и TRInv 2. Элемент сопоставления – половина элемента "2 CAM cells"

Традиционно блоки ассоциативной памяти проектируются на основе 6-транзисторных ячеек памяти (6T). В работе [5] описаны методы, использованные при проектировании 28-нм КМОП TLB. С уменьшением проектных норм до 65 нм и ниже увеличилась чувствительность 6Т ячеек к воздействию одиночных ядерных частиц. Новые ячейки STG DICE (Spaced Transistor Groups Dual Interlocked Cells) [6], [7] имеют повышенную устойчивость к эффектам воздействия одиночных ядерных частиц и являются перспективным решением для использования в регистрах САМ с повышенной сбоеустойчивостью.

В работе проведен анализ потребляемой мощности регистров САМ на базе ячеек STG DICE, представлено сравнение схем суммирования сигналов сопоставления: схемы на основе комбинационной логики (combinational logic – CL) и схемы на основе линии сопоставления ML.

II. Элементы регистра ассоциативной памяти

Регистр ассоциативной памяти включает в себя элементы сопоставления и ячейки маски. Схема элемента сопоставления на основе ячейки памяти STG DICE [8] представлена на рис. 1. Сравнение хранимых данных с данными, поступающими на входы Input 1 и Input 2, осуществляет логический элемент "Исключающее ИЛИ" (XOR), выполненный на основе двух инверторов с третьим состоянием TRInv 1 и TRInv 2. В зависимости от совпадения или несовпадения бита, хранящегося в ячейке, и бита, поданного на вход элемента, формируется состояние выхода элемента сопоставления (Output).

Транзисторы элемента сопоставления разделены на два блока так, что воздействие одиночной ядерной частицы с линейными потерями энергии на треке до $60 \text{ M} \cdot \text{B} \times \text{сm}^2/\text{м} \text{г}$ лишь на один блок не приводит к сбою ячейки, а приводит только к кратковременному импульсу помехи на выходе элемента сопоставления, что подтверждается TCAD моделированием [9]. На топологии между двумя блоками транзисторов одного элемента сопоставления расположен блок соседнего элемента сопоставления расположен блок соседнего элемента сопоставления расположен блок соседнего за



Рис. 2. Принципиальная схема ячейки маски (mask cell) на основе ячейки памяти STG DICE с декодером для считывания данных в стационарном и нестационарном состояниях STG DICE

мента сопоставления. При размерах блока 2.4×2.45 мкм такое чередование обеспечивает расстояние между взаимно чувствительными узлами, находящимися в разнесенных блоках, более 4 мкм. Это практически исключает вероятность одновременного воздействия частицы на два блока одного элемента сопоставления. Два элемента сопоставления с чередованием блоков на топологии образуют элемент "2 CAM cells".

Ячейка маски (mask cell) состоит из ячейки памяти STG DICE и декодера для считывания данных в нестационарных состояниях STG DICE, состоящего из двух инверторов Inv 1 и Inv 2, и двух инверторов с третьим состоянием TRInv 1 и TRInv 2 [10]. Схема ячейки маски приведена на рис. 2.

III. Суммирование сигналов сопоставления в регистрах ассоциативной памяти

Выходной сигнал сопоставления регистра ассоциативной памяти формируется схемой суммирования сигналов сопоставления всех элементов в составе регистра. Рассматриваемые в работе регистры ассоциативной памяти включают в себя 8 ячеек маски, каждая из которых, в зависимости от ее логического состояния, обеспечивает маскирование результата сопоставления двух определенных бит слова.

Регистр со схемой суммирования на основе комбинационной логики включает в себя три блока сопоставления (block of matching – BM), а также два блока сопоставления и маскирования (block of matching and masking – BMM) [11]. Блок BM состоит из восьми элементов сопоставления (т.е. четырех элементов "2 CAM cells") и логического элемента с компенсацией помехи 8NAND [12]. Схема блока BM представлена на рис. За. Блок BMM состоит из четырех элементов "2 CAM cells", четырех ячеек маски, а также комбинационной логической схемы, выполняющей суммирование сигналов сопоставления восьми элементов сопоставления блока BMM с учетом маскирования. Схема ¹/₄ части BMM представлена на рис. 36. Схема регистра ассо-



Рис. 3. Функциональные схемы блоков регистра с комбинационной логикой (CL): (а) – блок сопоставления (block of matching – BM); (b) – ¼ часть блока сопоставления и маскирования (block of matching and masking – BMM)



Рис. 4. Регистр ассоциативной памяти со схемой сопоставления на основе комбинационной логики

циативной памяти с комбинационной логикой приведена на рис. 4. Суммирование сигналов сопоставления блоков ВМ и ВММ выполняется логическим элементов 5OR, формирующим сигнал сопоставления для регистра – Output bit of matching. Таким образом, схема суммирования на основе комбинационной логики состоит из элементов 8NAND в составе ВМ, логических элементов в составе ВММ и логического элемента 5OR регистра и представляет из себя многовходовой комбинационный логический элемент И-НЕ (NAND) с возможностью логического маскирования отдельных входов.



Рис. 5. Схема регистра ассоциативной памяти с линией сопоставления ML, включающего три блока сопоставления данных ВМ_{ML} и два блока сопоставления и маскирования данных ВММ_{ML}



Рис. 6. Схемы блоков регистра ассоциативной памяти с линией сопоставления ML: (а) – ¼ часть блока сопоставления данных BM_{ML} ; (б) – ¼ часть блока сопоставления и маскирования данных BMM_{ML}

В регистре с линией сопоставления ML реализована схема "гонки токов" (сиггепt гасе scheme), предложенная в [4]. Схема регистра с ML представлена на рис. 5. Регистр с линией сопоставления включает в себя три блока сопоставления BM_{ML} и два блока маскирования и сопоставления BM_{ML} . На рис. 6а и бб представлены схемы блоков BM_{ML} и BMM_{ML} соответственно. Блок сопоставления включает в себя четыре элемента "2 CAM cells", выходные сигналы которых поступают на затворы *P*MOII транзисторов, соединяющих ML с шиной питания. Блок BMM_{ML} включает в себя четыре элемента "2 CAM cells" и четыре ячейки маски "mask cell". Для реализации маскирования используются дополнительные РМОП транзисторы, включенные последовательно с транзисторами, управляемыми маскируемыми элементами сопоставления. В зависимости от состояния ячейки маски, дополнительные транзисторы закрываются, исключая влияние маскируемых бит на результат сопоставления в регистре. Элементы сопоставления "2 САМ cells" и ячейки маски "mask cell" в регистре с ML такие же, как и в регистре с CL.

При сопоставлении данных в регистре, линия ML, предварительно заряженная до напряжения питания, по высокому уровню сигнала SE подключается к общей шине через транзистор N_{SE}. При совпадении слова, хранящегося в регистре, с входным словом шины Input for lookup все РМОП транзисторы блоков ВМ_М и ВММ_М закрыты, ML разряжается до низкого логического уровня, а выход переходит в состояние логической "1". При несовпадении битов в элементе сопоставления, РМОП транзистор, управляемый данным элементом открыт, и линия ML подключена к шине питания. Тогда через ML протекает сквозной ток с шины питания на общую шину пока сигнал SE находится в высоком логическом уровне. Затем SE переходит в низкий уровень, транзистор N_{SE} закрывается, и линия ML возвращается в высокое логическое состояние, а выход – в состояние логического "0".

IV. Анализ потребления комбинационной логики и линии сопоставления

В регистре с комбинационной логической схемой суммирования мощность потребляется элементами сопоставления и комбинационной логической схемой. При изменении данных, поданных на входы Input 1, Input 2 элемента сопоставления, выход элемента сопоставления изменяет свое логическое состояние на противоположное. Таким образом, мощность, потребляемая элементами сопоставления регистра, зависит от количества элементов сопоставления, изменивших выходное состояние в данном цикле поиска. Мощность, потребляемая комбинационной логикой, зависит от количества переключившихся логических вентилей. Это количество зависит от количества элементов сопоставления, изменивших свое состояние.

В регистре с линией сопоставления мощность также потребляется элементами сопоставления и самой линией сопоставления. Потребление элементов сопоставления в двух различных регистрах равно, поскольку элементы в этих регистрах одинаковы. Потребление

Таблица 1

Значения потребляемой мощности регистров с комбинационной логикой (CL) и линией сопоставления (ML) при различных состояниях выходов элементов сопоставления, мкВт

| Состояния выходов элементов сопоставления | Элементы со- поставления | CL | ML | Регистр с CL | Регистр с ML |
|---|-----------------------------|-----|----|-----------------|-----------------|
| Несовпадение всех бит → совпадение всех бит | 268 | 97 | 48 | 360 | 316 |
| Совпадение всех бит → несовпадение всех бит | 99 | 104 | 57 | 202 | 155 |
| Несовпадение всех бит → несовпадение 1 бита | 264 | 75 | 52 | 345 | 316 |
| Несовпадение всех бит → совпадение 1 бита | 6 | 3 | 57 | 9 | 63 |
| Совпадение 1 бита → несовпадение всех бит | 3 | 2 | 56 | 5 | 58 |

схемы с ML зависит от результата сопоставления в регистре. При совпадении данных ML отключена от шины питания и разряжается через транзистор N_{SE}. В таком случае мощность, потребляемая схемой с ML, ограничена значением $P_{ML} = V_{DD}^2 \cdot C_{ML} / T_S$, где V_{DD} – напряжение питания, C_{ML} – емкость линии сопоставления, T_S – длительность операции поиска. При несовпадении в регистре, через РМОП транзисторы линии сопоставления и транзистор N_{SE} протекает сквозной ток с шины питания на общую шину. В таком случае потребляемая мощность определяется длительностью импульса SE, отпирающего транзистор N_{SE}, и величиной сквозного тока, которая определяется параметрами транзисторов элементов BM_{ML} и BMM_{ML} и транзистора N_{SE}, а также количеством открытых РМОП транзисторов, которое равно количеству элементов сопоставления, в которых произошло несовпадение бит.



Рис. 7. Зависимость потребляемой мощности P схем суммирования с комбинационной логикой и линией сопоставления от количества N входов схемы суммирования, изменивших свое состояние. Графики "0 \rightarrow 1" соответствуют переходу N входов из "0" в "1", остальные входы остаются в изначальном состоянии "0". График "1 \rightarrow 0" соответствует переходу N входов из "1" в "0", остальные входы остаются в изначальном состоянии "1"

В табл. 1 представлены значения потребляемой мощности схем суммирования и регистров для разных выходных состояний элементов сопоставления. К примеру, запись "несовпадение всех бит \rightarrow совпадение всех бит" соответствует случаю, в котором все элементы сопоставления изначально находились в состоянии несовпадение проводилось в симуляторе Spectre CADENCE для структур, спроектированных по объемной КМОП 65 нм технологии, при температуре +25°С, напряжении питания 1.0 В и тактовой частоте 1 ГГц. Значения мощностей, потребляемых элементами сопоставления одинаковы для обоих регистров, так как сами элементы сопоставления в регистрах с CL и ML одинаковы.

Мощность, потребляемая комбинационной логической схемой, зависит от числа переключившихся логических вентилей в составе схемы в виду изменения состояния входов. По этой причине значение мощности, потребляемой схемой с CL, зависит от последовательности данных, поступающих с шины Input for lookup и варьируется от 2 до 104 мкВт. Минимальное значение мощности соответствует случаю "совпадение 1 бита → несовпадение всех бит", когда только один вход схемы суммирования изменяет свое логическое состояние. Максимальное значение мощности соответствует случаю "совпадение всех бит → несовпадение всех бит", когда все входы схемы суммирования изменяют свое логическое состояние. Мощность, потребляемая схемой с ML зависит от количества совпавших бит только в текущем цикле сопоставления и варьируется от 48 до 57 мкВт. Минимальное значение мощности соответствует случаю "несовпадение всех бит \rightarrow совпадение всех бит", когда все РМОП транзисторы линии ML закрыты, и ML разряжается до уровня логического "0". Максимальное значение мощности соответствует случаю "совпадение всех бит → несовпадение всех бит", когда все РМОП транзисторы линии ML открыты, и сквозной ток через ML максимален. Диапазон значений мощности, потребляемой схемой с ML меньше диапазона значений мощности, потребляемой схемой с CL в 11.3 раза.

На рис. 7 представлена зависимость потребляемой мощности Р схем суммирования (с CL и с ML) от количества N входов схемы, изменивших свое состояние. Для комбинационной логики приведены два графика. Графики "0-1" соответствуют начальному состоянию всех входов схемы – логический "0" и переходу N входов в состояние логической "1", график "1→0" соответствует начальному состоянию всех входов схемы логическая "1" и переходу N входов в состояние логического "0". В случае "1→0" схема с СL потребляет больше мощности, чем в случае "0→1", поскольку в случае "1→0", при том же количестве входов, изменивших свое состояние, переключается большее количество логических вентилей в составе CL. Например, если все входы элемента И-НЕ находятся в состоянии логической "1", изменение состояния даже одного *i*-го входа на "0" приведет к изменению состояния выхода схемы. В этом случае произойдет переключение всех логических вентилей в составе схемы И-НЕ, состояние которых зависит от состояния *i*-го входа. В случае переключения входов из "0" в "1" все входы изначально находятся в состоянии логического "0", и переключение одного *i*-го входа в состояние "1" не изменит состояния выхода схемы. В таком случае не переключится ни один логический вентиль в составе схемы И-НЕ.

Как видно из рис. 7, при переключении более половины входов схемы значение мощности, потребляемой схемой с CL больше, чем значение мощности, потребляемой схемой с ML. Количество входов схемы суммирования, изменивших состояние, равно количеству бит шины Input for lookup, изменивших состояние. Если слова данных, поступающие на шину поиска Input for lookup блока CAM в среднем отличаются менее, чем половиной бит, то регистр с CL будет потреблять меньше мощности, чем регистр с ML.

Таблица 2

Параметры схем суммирования с комбинационной логикой (CL) и линией сопоставления (ML)

| Параметр | CL | ML |
|-------------------------|------|------|
| N _{TR} | 280 | 108 |
| S, мкм ² | 110 | 72 |
| Р _{МІN} , мкВт | 2 | 48 |
| Р _{МАХ} , мкВт | 104 | 57 |
| t _{DEL} ,HC | 0.23 | 0.18 |

В табл. 2 представлены параметры схем суммирования. Здесь N_{TR} – количество транзисторов, S – площадь, P_{MIN} – минимальная потребляемая мощность в зависимости от входных данных, P_{MAX} – максимальная потребляемая мощность в зависимости от входных данных, t_{DEL} – задержка выходного сигнала сопоставления. Диапазон значений мощности, потребляемой схемой с ML, меньше диапазона значений мощности, потребляемой схемой с CL, в 11.3 раза. Задержка сигнала сопоставления ML меньше задержки CL на 22%, а площадь, занимаемая схемой с ML на кристалле меньше на 35%.

V. Заключение

Проведен анализ потребления мощности регистров ассоциативной памяти с двумя различными схемами суммирования – комбинационной логической схемой и схемой на основе линии сопоставления. Мощность, потребляемая схемой на основе комбинационной логики, зависит от количества N входов схемы, изменивших свое состояние. Мощность, потребляемая схемой на основе линии сопоставления, практически не зависит от N и соответствует потреблению комбинационной логической схемы при переключении половины входов схемы. Если на этапе проектирования блока ассоциативной памяти известно, что слова данных, поступающие на шину поиска, в среднем отличаются меньше, чем половиной бит, то использование схемы суммирования на основе комбинационной логики позволит снизить потребляемую мощность.

Анализ топологии и результатов моделирования показал, что схема суммирования на основе линии сопоставления занимает на 35% меньше площади на кристалле и обладает на 22% меньшей задержкой по сравнению со схемой суммирования на основе комбинационной логики.

ЛИТЕРАТУРА

[1] Do A.T., Yin C., Velayudhan K., Lee Z.C., Yeo K.S., Kim T.T-H. 0.77 fJ/bit/search content addressable memory using

small match line swing and automated background checking scheme for variation tolerance // IEEE Journal of Solid-State Circuits. 2014. V. 49. № 7. P. 1487–1498.

- [2] Zukowski C.A., Wang S.-Y. Use of selective precharge for lowpower content-addressable memories // Proc. of IEEE International Symposium of Circuits Syst (ISCAS). Hong Kong, 1997. V. 3. P. 1788–1791.
- [3] Соловьева Л.А. Проектирование гибридного регистра ассоциативной памяти // Проблемы разработки перспективных микро- и наноэлектронных систем – 2016. Сб. трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2016. Ч. 3. С. 171-177.
- [4] Arsovski I., Chandler T., Sheikholeslami A. A ternary content addressable memory (TCAM) based on 4T static storage and including a current-race sensing scheme // IEEE Journal of Solid-State Circuits. 2003. V. 38. № 1. P. 155– 158.
- [5] Кириченко П.Г., Соловьева Л.А., Тарасов И.В. Проектирование 14-портового регистрового файла и буфера трансляции адресов со сниженным потреблением с учетом особенностей технологии 28 нм // Проблемы разработки перспективных микро- и наноэлектронных систем – 2016. Сб. трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2016. Ч. 3. С. 129-135.
- [6] Стенин В.Я., Катунин Ю.В., Степанов П.В. Сбоеустойчивые ОЗУ на основе STG DICE элементов памяти с разделенными на две группы транзисторами // Микроэлектроника. 2016. Т. 45. № 6. С. 456–470.
- [7] Катунин Ю.В., Стенин В.Я. ТСАD моделирование эффектов воздействия одиночных ядерных частиц на ячейки памяти STG DICE // Микроэлектроника. 2018. Т. 47. № 1. С. 23-37.
- [8] Антонюк А.В., Стенин В.Я. Моделирование переходных процессов в 65 нм КМОП логическом элементе сравнения для ассоциативных запоминающих устройств при воздействии одиночных ядерных частиц // Вестник НИЯУ МИФИ. 2016. Т. 5. № 5. С. 445–453.
- [9] Katunin Yu.V., Stenin V.Ya. TCAD Simulation of Single-Event Transients in the 65-nm CMOS Element of Matching for a Content-Addressable Memory // Proc. of 25th Telecommunication Forum TELFOR – 2017. Belgrade, 2017. P. 1–4.
- [10] Stenin V.Ya., Antonyuk A.V., Stepanov P.V., Katunin Yu.V. Design of the 65-nm CMOS Translation Lookaside Buffer on the Hardened Elements // Proc. of 25th Telecomunication Forum TELFOR – 2017. Belgrade, 2017. P. 1–4.
- [11] Stenin V.Ya., Antonyuk A.V., Katunin Yu.V., Stepanov P.V. Design of logical elements for the 65-nm CMOS translation lookaside buffer with compensation of single events effects // Proc. of International Siberian Conference on Control and Communication SIBCON. Astana, 2017. P. 1–6.
- [12] Katunin Yu.V., Stenin V.Ya., Antonyuk A.V. Design of Logical Elements with Single-Event Compensation for the 28-nm CMOS Decoders // Proc. of 24th Telecomunication Forum TELFOR – 2016. Belgrade, 2016. P. 617–620.

Analysis of Power Consumption of Matching Signals Summation Circuits for 65 nm CMOS Associative Memory Registers

A.V. Antonyuk^{1,2}, P.V. Stepanov¹

¹Scientific Research Institute for System Analysis, Russian Academy of Sciences

²National Research Nuclear University "MEPhI", antonyuk@cs.niisi.ras.ru,

stepanov@cs.niisi.ras.ru

Abstract — Power consumptions of two registers of contentaddressable memory (CAM) based on STG DICE memory cells were analyzed. Considered CAM registers have different summation circuits of matching signals - the circuit based on combinational logic (CL) and the circuit based on match line (ML). The elements of matching (CAM cells) and the elements of masking (mask cells) based on upsethardened STG DICE memory cells are the same in register with CL and in register with ML, so they consume the same amount of power. CL circuit includes three 8-input NAND elements with compensation of single event effects, 5-input OR element and logical gates that provide logical masking of certain inputs. Inputs of CL circuit are connected to the outputs of elements of matching. Circuit with match line consists of PMOS precharge transistor, NMOS discharge transistor and match line ML that is connected to the supply through PMOS transistors that are controlled by elements of matching. In search operation, NMOS transistor opens trying to discharge ML. In case of match in all elements of matching, ML voltage becomes low. In case of miss in one or more elements, there is a short-through current in ML, and ML voltage remains high. Power consumption of CL circuit depends on number N of inputs of the circuit that change their logical states, because the number of switched logical gates inside CL depends on the N value. Analysis of the simulation results showed, that the value of power consumption of CL differs from 2 to 104 µW as a function of N value. Power consumption of ML depends on the matching result in register - match or miss. In case of miss, ML power depends on the value of short-through current that depends on number of mismatched bits. Value of ML power consumption differs from 48 to 57 μ W and it practically equals the value of CL power consumption when a half of the CL inputs switches. So if in the step of CAM development, it is known that input data words of CAM differs from each other in average by less than a half of all bits, then CL is preferable to save power in CAM. Analysis of topology and simulation results showed, that ML takes 35% less area on the chip and has 22% less output bit of matching delay compared to CL.

Keywords — combinational logic, content-addressable memory, design, logical element, power consumption, simulation, topology

REFERENCES

- [1] Do A.T., Yin C., Velayudhan K., Lee Z.C., Yeo K.S., Kim T.T-H. 0.77 fJ/bit/search content addressable memory using small match line swing and automated background checking scheme for variation tolerance // IEEE Journal of Solid-State Circuits. 2014. V. 49. № 7. P. 1487–1498.
- [2] Zukowski C.A., Wang S.-Y. Use of selective precharge for lowpower content-addressable memories // Proc. of IEEE

International Symposium of Circuits Syst (ISCAS). Hong Kong, 1997. V. 3. P. 1788–1791.

- [3] Solovyeva L.A. Design of the hybrid CAM register // Problemy razrabotki perspektivnyh mikro- i nanoehlektronnyh sistem – 2016. Sb. trudov / pod obshch. red. akademika RAN A.L. Stempkovskogo. M.: IPPM RAN, 2016. Ch. 3. S. 171-177.
- [4] Arsovski I., Chandler T., Sheikholeslami A. A ternary content addressable memory (TCAM) based on 4T static storage and including a current-race sensing scheme // IEEE Journal of Solid-State Circuits. 2003. V. 38. № 1. P. 155– 158.
- [5] Kirichenko P.G., Solovyeva L.A., Tarasov I.V. Design of Power Efficient 14-port Register File and Translation Lookaside Buffer in 28-nm Process // Problemy razrabotki perspektivnyh mikro- i nanoehlektronnyh sistem – 2016. Sb. trudov / pod obshch. red. akademika RAN A.L. Stempkovskogo. M.: IPPM RAN, 2016. Ch. 3. S. 129-135.
- [6] Stenin V.Ya., Katunin Yu.V., Stepanov P.V. Sboeustojchivye OZU na osnove STG DICE ehlementov pamyati s razdelennymi na dve gruppy tranzistorami (Upset-hardened RAM based on STG DICE memory cells with transistors separated into two groups) // Mikroehlektronika. 2016. T. 45. № 6. S. 456–470.
- [7] Katunin Yu.V., Stenin V.Ya. TCAD modelirovanie effektov vozdejstviya odinochnyh yadernyh chastic na yachejki pamyati STG DICE (TCAD simulation of effects of single nuclear particles influence on STG DICE memory cells) // Mikroelektronika. 2018. T. 47. № 1. S. 23-37.
- [8] Antonyuk A.V., Stenin V.Ya. Simulation of single-event transients in 65-nm CMOS logical elements of comparison for an associative memory // Vestnik of NRNU MEPhI. 2016. T. 5. № 5. S. 445–453.
- [9] Katunin Yu.V., Stenin V.Ya. TCAD Simulation of Single-Event Transients in the 65-nm CMOS Element of Matching for a Content-Addressable Memory // Proc. of 25th Telecommunication Forum TELFOR – 2017. Belgrade, 2017. P. 1–4.
- [10] Stenin V.Ya., Antonyuk A.V., Stepanov P.V., Katunin Yu.V. Design of the 65-nm CMOS Translation Lookaside Buffer on the Hardened Elements // Proc. of 25th Telecommunication Forum TELFOR – 2017. Belgrade, 2017. P. 1–4.
- [11] Stenin V.Ya., Antonyuk A.V., Katunin Yu.V., Stepanov P.V. Design of logical elements for the 65-nm CMOS translation lookaside buffer with compensation of single events effects // Proc. of International Siberian Conference on Control and Communication SIBCON. Astana, 2017. P. 1–6.
- [12] Katunin Yu.V., Stenin V.Ya., Antonyuk A.V. Design of Logical Elements with Single-Event Compensation for the 28-nm CMOS Decoders // Proc. of 24th Telecommunication Forum TELFOR – 2016. Belgrade, 2016. P. 617–620.

Четырехканальный мультистандартный адаптивный последовательный приемопередатчик для диапазона 1.25–10.3 Гб/с по технологии КМОП 65нм

А.В. Ларионов, О.Н. Буякова, О.В. Сысоева, С.Э. Осина, С.О. Задябин, П.А. Алексан, И.В. Тарасов, Ю.Б. Рогаткин, В.В. Мастеров

Научно-исследовательский институт системных исследований PAH, alar@cs.niisi.ras.ru

— В статье представлен полностью Аннотация адаптивный последовательный четырехканальный приемопередатчик для стандартов 10G Base-R, SATA, РСІ Express, FC, RapidIO, DisplayPort. Блок функционирует в диапазоне 1,25÷10,3125 Гб/с с вероятностью битовых ошибок BER менее 10⁻¹². Суммарный джиттер выходного сигнала передатчика ТЈтх < 26пс@10,3125 Гб/с. Толерантность приемника к высокочастотному джиттеру JTHF_{RX} > 0,3UI (30% единичного интервала) при приеме данных с затуханием 24дБ@10,3125Гб/с для полинома PRBS31. Приемопередатчик демонстрирует безошибочную обработку данных на скорости 10,3125 Гб/с для канала с суммарным затуханием 33 дБ. Блок спроектирован по технологии КМОП 65 нм и имеет потребляемую мощность 760мВт@10,3125Гб/с. К особенностям данного приемопередатчика относятся: эквалайзер с решающей обратной связью и прямым доступом с уменьшенной потребляемой мощностью, контроллер поискя положения фазы вспомогательного оптимального тактового сигнала для увеличения толерантности псевдодифференциальный приемника. касколный выходной буфер и контроллер минимально допустимого напряжения выходного сигнала передатчика.

Ключевые слова — приемопередатчик, приемник, передатчик, эквалайзер, выходной буфер, входной буфер, бинарный алгоритм наименьших квадратов.

I. Введение

Большое разнообразие стандартов для коммутации устройств через электрическое соединение (кабель/печатная плата) приводит к увеличению площади, потребляемой мощности, количества выводов интегральной микросхемы. Данная проблема решается путем проектирования приемопередатчика, покрывающего максимально возможное количество требуемых стандартов.

Степень деградации передаваемого сигнала в коммутирующем канале существенно варьируется в зависимости от ряда физических эффектов. Для обеспечения корректной работы приемопередатчик должен обладать широким диапазоном толерантности, адаптивно подстраивая систему в зависимости от целостности сигнала. В главе II представлена архитектура приемопередатчика верхнего уровня. Реализации приемника и передатчика показаны в главах III и IV, соответственно. Результаты измерений приведены в главе V.

II. АРХИТЕКТУРА ПРИЕМОПЕРЕДАТЧИКА

Архитектура приемопередатчика, реализованного в данной работе, показана на рис. 1.



Рис. 1. Структурная схема четырехканального приемопередатчика

Две двухгенераторные LCPLL с центральными частотами 5,0, 6,3, 8,0, 10,1 ГГц и диапазоном перестройки ±12,5% покрывают частотный диапазон от 4,5 до 11.3 ГГц. Сигнал делится на два, формируя квадратурный тактовый сигнал CLKI/CLKQ,

поступающий на вход каждого канала. В процессе распространения скважность и квадратура CLKI/CLKQ деградируют. Блок DQC восстанавливает тактовый сигнал, способствуя улучшению линейности интерполяторов.

Блок SER передатчика мультиплексирует входную параллельную шину данных TIN с разрядностью N (40/32/20/16/10/8) в 2-х битную последовательность. Связка блоков SREG, PRE, DRIV образует FFE эквалайзер 3 порядка, основанный на принципе нерекурсивного фильтра с конечной импульсной характеристикой. SREG - сдвиговый регистр, PRE мультиплексирующая матрица, перераспределяющая вклад каждого порядка фильтра в выходном потоке, DRIV - выходной буфер с операцией суммирования. SWC регулирует размах выходного сигнала. Цифровая машина DLL подстраивает фазу внутреннего тактового сигнала под фазу входного сигнала TCLKIN, управляя интерполятором передатчика. На выход интерполятора добавлен дополнительный корректор скважности DCC, который нивелирует остаточные эффекты в процессе преобразования импульсного тактового сигнала в цифровой сигнал. Программируемый выходной делитель DIV на 1, 2 или 4 позволяет формировать тактовые сигналы для обработки данных в диапазоне 1.25÷10,3125 Гб/с.

Входной тракт приемника состоит из блоков TERM, VGA и CTLE. Блок TERM согласует вход приемника с каналом и передатчиком, обеспечивает достаточный уровень электростатической защиты, осуществляет регулировку уровня постоянной составляющей. Автоматическая регулировка усиления VGA обеспечивает оптимальный размах, а линейный эквалайзер CTLE компенсирует МСИ (межсимвольную интерференцию) в средней части частотного спектра входного сигнала. Далее сигнал подается на вход эквалайзера с решающей обратной связью DFE 4 порядка, состоящего из трех трактов: Boundary, Data Auxiliary, тактируемых И синхросигналами CLKB, CLKD CLKA, И соответственно. Блок способен компенсировать нелинейные затухания входного сигнала без усиления шума и перекрестных помех. Затем зафиксированные данные демультиплексируются и в низкочастотном режиме обрабатываются в блоке восстановления синхронизации CDR и блоке контроля коэффициентов эквалайзеров SSLMS. Блок CDR формирует управляющие коды BCode, DCode и ACode для трех независимых интерполяторов PI, подстраивая частоту и фазу синхросигналов CLKB, CLKD и CLKA. Блок SSLMS формирует управляющие коды VGACode, CTLECode, DFECode для регулировки амплитуды VGA, глубины эквалайзера СТLE, порогового уровня и весовых коэффициентов DFE, постоянной составляющей.

III. ПРИЕМНИК

А. Входной тракт

На рис. 2 показан входной тракт приемника. Все параметры тракта подстраиваются цифровым образом. Вход TERM имеет дифференциальную катушку T, которая уменьшает потери на отражения, вызванные



Рис. 2. Принципиальная схема входного тракта приемника и АЧХ каскадов

емкостью электростатической защиты. Терминатор R_{TERM} автоматически калибруется в диапазоне ±25% от номинала, компенсируя технологический разброс. Барьерный конденсатор Св формирует независимую постоянную составляющую на входе VGA, что позволяет улучшить линейность И полосу пропускания. Желаемое значение постоянной составляющей определяется резистивным делителем R_A/R_B. Поскольку в печатных платах скин-эффект может проявляться уже на частоте 50 МГц, блок TERM также выполняет роль низкочастотного эквалайзера, ослабляя низкочастотную составляющую сигнала на величину 3,5 дБ. Коэффициент передачи VGA регулируется в диапазоне от -4,5 до 7,5 дБ резистором R_{VGA}. Для предотвращения влияния паразитного нуля в режиме аттенюатора на выходе VGA добавлен дополнительный конденсатор C_{NVGA}, величина которого зависит от текущего коэффициента передачи. Эквалайзер CTLE (на рис. 2 показан только один из двух каскадов) имеет фиксированный коэффициент передачи по постоянному сигналу. Это позволяет уменьшить либо требуемый коэффициент передачи VGA для каналов с большим затуханием, либо нелинейность DFE для каналов с малым затуханием. Регулировка коэффициента усиления CTLE осуществляется конденсатором достигая C_{CTLE}, максимального усиления около 11 дБ на частоте 5 ГГц.

В. Эквалайзер с решающей обратной связью

Блок DFE в работе [1] реализован на токовой логике CML (Current Mode Logic), что требует больших энергетических затрат. В работе [2] потребление снижено за счет замены резистивной нагрузки на активную индуктивность, потребление блока при этом все еще значительно. В [3] эквалайзер полностью реализован на логике с предзарядом, что существенно снижает потребление. Однако такой подход приводит к потере быстродействия. Более того, схема тактируется синхросигналами сложной формы, увеличивая потребление формирователя тактовых сигналов.

На рис. 3 показана схема DFE, реализованная в данном приемнике. Эквалайзер содержит лва параллельных идентичных конвейера, что позволяет вдвое снизить требования к полосе пропускания элементов. Конвейер EVEN обрабатывает четные, а ODD нечетные импульсы входной последовательности данных. В процессе проектирования DFE важно выполнить временные ограничения, накладываемые на формирование данных в петле обратной связи. Для первого порядка (k = 1) общее время задержки должно быть в пределах UI, для остальных порядков (k ≥ 2) – в пределах 2UI. Предлагается распараллелить путь данных deven(2n) для формирования первого и остальных порядков фильтра (аналогично для dodd(2n+1)). Путь первого порядка показан пунктирной линией и полностью реализован на CML логике с активной индуктивностью в нагрузке. Для остальных порядков фильтра используются триггеры на основе усилителя

считывания SA (Sense Amplifier) [4], что позволяет существенно снизить затраты по мощности. Для 1-ого порядка:

$$T_{TAP1} = T_{SUM} + T_{BUF} + T_{SETUP_L} < UI,$$

где T_{SUM} – задержка сумматора, T_{BUF} – задержка буфера, T_{SETUP_L} – время предустановки данных защелки L. Для 2-ого и более высоких порядков:

$$T_{TAPK} = T_{SUM} + T_{BUF} + T_{SETUP_SA} + T_{C2Q_SA} < 2UI,$$

где T_{SETUP_SA} – время предустановки триггера SA, T_{C2Q_SA} – время задержки триггера SA.



Рис. 3. Блок-схема эквалайзера с решающей обратной связью

Если СLKD подавать на защелку L и триггер SA одновременно, то временное ограничение в один UI будет уменьшено на величину $T_{SETUP_L}-T_{SETUP_SA}$. Это связано с тем, что эти значения не совпадают, T_{SETUP_L} – положительно, а T_{SETUP_SA} – отрицательно. Для компенсации этой временной разницы синхросигнал на защелку L подается с задержкой относительно триггера SA.

С. Адаптивные алгоритмы приемника

В основу алгоритмов, необходимых для восстановления целостности входного сигнала, заложен бинарный метод наименьших квадратов SSLMS [3]. Пороговый уровень и коэффициенты DFE подстраиваются исходя из уравнений:

$$h(n,0) = h(n-1,0) - \mu * sign[e(n)],$$

$$h(n,k) = h(n-1,k) - \mu * sign[e(n)] * d(n-k),$$

где k – порядковый номер фильтра от 1 до 4, µ – коэффициент передачи и sign[e(n)] – знак ошибки.

Коррекция постоянной составляющей основана на выравнивании количества ошибок конвейера EVEN (где проходит проверка для позитивных данных (единиц)), с количеством ошибок конвейера ODD (где проверяются негативные данные (нули)):

$$h_{OFFC}(n) = h_{OFFC}(n-1) - \mu^* (-1)^n * sign[e(n)].$$

Из АЧХ на рис. 2 видно, что СТLE усиливает в диапазоне от 0,6 до 5 ГГц. Достаточно аккумулировать 8 значений данных, предшествующих текущему значению. Регулировка глубины СТLE:

$$h_{CTLE}(n) = h_{CTLE}(n-1) - \mu_{CTLE} * sign[e(n)] * \sum_{k=1}^{8} d(n-k),$$

µ_{CTLE} – коэффициент передачи линейного эквалайзера.

После подстройки коэффициентов СТLE, DFE и постоянной составляющей осуществляется однократная проверка h(0) на условие:

$$h_{MIN}(0) \le h(0) \le h_{MAX}(0).$$

Если условие выполнено, то коэффициент передачи VGA остается неизменным (значение по умолчанию). В противном случае он меняется в зависимости от текущего значения h(0), используя аппаратную прошивку. Затем приемник сбрасывает все адаптивные настройки и подстраивается заново с учетом актуального значения коэффициента передачи VGA.

D. Контроллер поиска оптимальной фазы вспомогательного тактового сигнала

На рис. 4 показана диаграмма сигналов конвейера EVEN после сходимости всех алгоритмов приемника. Информация фиксируется тремя синхросигналами. CLKD фиксирует deven(2n) по центру, формируя исходные переданные по каналу данные. CLKB фиксирует deven(2n) в моменты переключений, формируя данные, необходимые для синхронизации входного сигнала с внутренней тактовой системой. CLKA фиксирует eeven(2n) по центру, формируя данные для восстановления целостности входного сигнала. В момент выборки CLKD размах сигнала deven(2n) будет соответствовать эталонному уровню h(0), а в момент выборки CLKA размах сигнала eeven(2n) будет равен нулю.

Из рис. 4 видно, что неправильная позиция CLKA вызовет ошибку Δe . Эта ошибка приводит к тому, что регистрируемая системой величина МСИ входного

сигнала z(n) будет казаться больше, чем есть на самом деле. Суммарное значение h(k) будет завышено, а значение h(0) – занижено. Неоптимальные коэффициенты h(k) снизят эффективность системы. Возможные причины возникновения ошибки:

1) квадратура СLКА к СLКВ может быть нарушена, например, в силу технологического разброса,

2) истинность фазы CLKB также не гарантирована. Причина в возможном ассиметричном распределении детерминированного джиттера входного сигнала [7],

3) задержки компаратора и буфера на рис. 3 не идентичны.



Рис. 4. Диаграмма работы DFE после сходимости всех алгоритмов приемника

Методика поиска оптимальной фазы синхросигнала CLKA, вспомогательного предложенная в данной работе, основана на мониторинге порогового уровня h(0). Суть методики сводится к поиску максимального значения h(0) путем принудительного варьирования фазы СЬКА в диапазоне, перекрывающем возможное отклонение от идеальной позиции. Положение фазы CLKA, при котором пороговое напряжение h(0) будет максимальным, означает, что ошибка e0 минимизирована и коэффициенты h(k) оптимальны.

В блок SSLMS встроен контроллер ААСС, задача которого регулировать фазу CLKA через блок CDR. Схема CDR на рис. 5 состоит из бинарного фазового детектора PD, мажоритарной схемы MV, интегральнопропорционального фильтра PIF, формирователя сдвига фазы SHIFT и декодера TC. Используя данные из Boundary и Data, фазовый детектор вырабатывает информацию о направлении сдвига синхросигналов приемника. Мажоритарная схема выполняет децимацию, снижая частоту обработки данных. Цифровой фильтр накапливает информацию для фазы синхросигналов в контроля частоты И интегральном И пропорциональном путях, соответственно. Коэффициенты К_I и К_Р задают полосы пропускания в этих путях. Блок ТС декодирует бинарный код в термо-код BCode, DCode, ACode,

понятный интерполяторам. Коэффициенты K_D и K_A задают сдвиг – соответственно DCode и ACode – относительно BCode.

нормальном режиме работы CDR B все коэффициенты - константы. Когда контроллер ААСС активен коэффициент КА – переменная величина. Значение h(n,0) используется контроллером для оценки направления сдвига CLKA. Общий принцип работы отражен в табл. 1. На каждом шаге анализируется совокупность трех переменных: текущее значение h(n,0), предыдущее значение h(n-1,0) и знак коэффициента приращения sign[a(n-1)] на предыдущем шаге. Исходя из этих данных формируется текущий знак sign[a(n)], увеличивая или уменьшая значение коэффициента K_A(n). Отметим, что коэффициент КА изменяется на каждом шаге, даже когда h(n,0) = h(n-1,0), что позволяет стимулировать машину состояний к поиску. По истечении определенного времени контроллер останавливается, а значение КА фиксируется.



Рис. 5. Упрощенная блок-схема CDR

На рис. 6 показана реализация контроллера. Компаратор сравнивает два семибитных кода – текущий h(n,0) и h(n-1,0). Логический ноль или единица коэффициента приращения a(n) отражают его знак, зависящий от результата на выходе компаратора и предыдущего значения a(n-1). Коэффициент K_{IA} задает полосу пропускания контроллера. Данные аккумулируются в 24-разрядном знаковом сумматоре с насыщением, где 7 старших битов отражают коэффициент $K_A(n)$.

Таблица 1

Алгоритм работы контроллера ААСС

| Сравнение | sign[a(n-1)] | Тенденция | sign[a(n)] |
|-------------------|--------------|---------------|------------|
| h(n,0) > h(n-1,0) | +1 | улучшение | +1 |
| h(n,0) > h(n-1,0) | -1 | улучшение | -1 |
| h(n,0) = h(n-1,0) | +1 | без изменений | +1 |
| h(n,0) = h(n-1,0) | -1 | без изменений | -1 |
| h(n,0) < h(n-1,0) | +1 | ухудшение | -1 |
| h(n,0) < h(n-1,0) | -1 | ухудшение | +1 |



Рис. 6. Блок-схема контроллера подстройки фазы вспомогательного синхросигнала

IV. ПЕРЕДАТЧИК

К особенностям передатчика, реализованного в данной работе, относятся: контроллер минимально допустимого напряжения выходного сигнала [5] и псевдодифференциальный касколный выходной буфер, схема которого показана на рис. 7. Аналогично входу приемника в буфер встроены калибруемый терминатор R_{TERM} и дифференциальная катушка Т. Буфер представляет собой токовый сумматор из 64 одинаковых параллельно соединенных ячеек. Транзисторы M1/M2 выполнены на толстом окисле и формируют основной ток буфера. Ключи М3/М6 соединены с М1/М2 последовательно. Особенность буфера – вспомогательный источник тока М7, контролирующий потенциалы в узлах V1 и V2 через ключи М4/М5. Это дает два преимущества. Во-первых, снижается частотно-зависимый характер работы выходного буфера путем ликвидации остаточного заряда в V1/V2 в процессе ухода в отсечку M3/M6. Вовторых, позволяет избежать возможные перегрузки на стоках М3/М6.



Рис. 7. Принципиальная схема псевдодифференциального каскодного выходного буфера

V. РЕЗУЛЬТАТЫ ИЗМЕРЕНИЙ

Блок спроектирован и изготовлен по технологии КМОП 65 нм. Топология показана на рис. 8 и имеет геометрические размеры 2,22 х 1,3 мм. Блок LCPLL и выходной буфер передатчика работают от напряжения питания 2,5 В, остальные блоки от 1 В. Исследовалась микросхема 1890ВГ19Я, имеющая в своем составе 10 четырехканальных приемопередатчиков [6].

Толерантность приемника измерена прибором J-BERT N4903B or Agilent Technologies. PRBS generator формирует последовательность данных и подает её на вход приемника, PRBS checker снимает данные с выхода передатчика и осуществляет проверку на истинность, при этом сам приемопередатчик работает в режиме параллельной петли обратной связи. В этом режиме управляющие коды на вход интерполятора передатчика поступают из блока CDR приемника (а не из DLL, как на рис. 1). На рис. 9 показана полученная зависимость для скорости 10,3125 Гб/с. Данные, имеющие полиномом PRBS31, подаются на вход приемника через встроенную в J-BERT печатную плату (материал Nelco 4000-6) длиной 40см (16 дюймов). Суммарное затухание канала на частоте Найквиста составляет около 24 дБ (18,4 дБ для Nelco 4000-6 и 5,6 дБ для соединителей, дочерней платы и корпуса). Запас толерантности к высокочастотному джиттеру JTHF_{RX} составил 0.3UI (30% единичного интервала).

Рис. 10 демонстрирует эффективность работы контроллера поиска оптимальной фазы вспомогательного тактового сигнала. Измерения проведены для 32 приемников. Результат каждой выборки представлен как разница толерантности выключенным приемника включенным с И контроллером. В 20-ти случаях улучшение не регистрируется. Максимальный эффект составил 15*10-3UI. Отрицательного влияния контроллера на толерантность приемника не зарегистрировано. Оценка эффективности контроллера через среднеарифметическое значение составляет приблизительно 2,6*10⁻³UI.

На рис. 11 показан совокупный анализ измерений, демонстрирующий запас толерантности приемника к высокочастотному (> 20 МГц) синусоидальному джиттеру для четырех скоростей с различными длинами печатной платы Nelco 4000-6. Например, для 5 Гб/с потока, прошедшего через плату в 110 см, запас составляет 0,36UI.

Дифференциальные потери на отражения для приемника измерены с помощью Tektronix DSA8300 и показаны на рис. 12. В диапазоне от 0 до 6 ГГц отражения лежат ниже -10 дБ.

На рис. 13 показана глазковая диаграмма, снятая с выхода передатчика с помощью Keysight DSAZ254A. Суммарный джиттер для 10,3125 Гб/с потока составляет 26 пс для BER = 10⁻¹², что менее 27% от единичного интервала. Отметим, что с целью компенсации затухания корпуса и печатной платы, эквалайзер передатчика был запрограммирован на величину 4,1 дБ на частоте Найквиста.

На заключительном этапе тестирования два модуля были соединены между собой через печатную плату длиной 50 см. Суммарное затухание канала на частоте 5,15 ГГц составило около 33 дБ (23 дБ – тестовая плата на рис. 14, 10 дБ – остальные компоненты). Эквалайзер передатчика запрограммирован на величину 7,2 дБ. Приемопередатчики продемонстрировали безошибочную работу в течение 72 часов, обмениваясь данными на скорости 10,3125 Гб/с.

В табл. 2 приведены основные характеристики четырехканальных приемопередатчиков для этой и ранее представленных работ.

Таблица 2

Основные характеристики четырехканальных приемопередатчиков

| Дизайн | Эта раб. | [1] | [7] | [8] |
|-----------------------|------------|--------------|------------|------------|
| Тех., нм | 65 CMOS | 32 CMOSLP | 90 CMOS | 90 CMOS |
| Скор., Гб/с | 1,25÷10.3 | 14,025 | 8 | 1,25÷10,3 |
| Затухание канала, дБ | 33 | 22 | 36,8 | 41 |
| Площ.,мм ² | 2,89 | 2,925 | 2,56 | 3,04 |
| Пит., В | 1,0/2,5 | 1,0 | 1,2 | 1,2/2,5 |
| Мощ., Вт | 0,76 | 0,88 | 0,928 | 1,264 |



Рис. 8. Топология четырехканального приемопередатчика



Рис. 9. Зависимость толерантности приемника от частоты синусоидального джиттера для 10,3125 Гб/с



Рис. 10. Статистическое распределение разности толерантности приемника к синусоидальному джиттеру на частоте 20 МГц с выключенным и включенным контроллером ААСС



Рис. 11. Зависимость толерантности приемника к высокочастотному (> 20 МГц) синусоидальному джиттеру от длины печатной платы Nelco 4000-6



Рис. 12. Зависимость дифференциальных потерь на отражение от частоты на входе приемника



Рис. 13. Глазковая диаграмма на выходе передатчика



Рис. 14. Зависимость затухания от частоты для тестовой печатной платы длиной 50 см

VI. Заключение

Спроектирован мультистандартный полностью адаптивный последовательный четырехканальный приемопередатчик для электрических коммутационных сред. Блок покрывает диапазон скоростей от 1,25 до 10,3125 Гб/с и способен функционировать в канале с затуханием 33 дБ с вероятностью ошибок менее 10⁻¹².

Литература

- Erba S. Multi-standard transceiver for NRZ serial communication up to 14Gbps // IEEE Signal and Power Integrity. 2012. P. 17-19.
- [2] Ларионов А.В. Эквалайзер с решающей обратной связью и активной индуктивностью для высокоскоростного приемника // VII Всероссийская научно-техническая конференция «Проблемы разработки перспективных микро- и наноэлектронных систем-2016 (МЭС-2016)» Сборник научных трудов. / Под ред. А.Л. Стемпковского – М.: ИППМ РАН, 2016. Часть III - С. 2-7.
- [3] Zhong F., Quan S., Liu W., et al. A 1.0625 ~ 14.025 Gb/s multi-media transceiver with full-rate source-seriesterminated transmit driver and floating-tap decisionfeedback equalizer in 40nm CMOS // IEEE Journal of Solid-State Circuits. 2011. V. 46. № 12. P. 3126-3139.
- [4] Nikolic B., Oklobdzija V., Stojanovic V. Improved senseamplifier-based flip-flop: design and measurements // IEEE Journal of Solid-State Circuits. 2000. V. 35. № 6. P. 876-884.
- [5] Ларионов А. В. Адаптивный эквалайзер с контроллером минимально допустимого дифференциального напряжения выходного сигнала и псевдодифференциальным каскодным выходным буфером для 10 Гб/с передатчика по технологии 65 нм // Микроэлектроника. 2015. Т. 44. № 6. С. 464-471.
- [6] Алексан П. А. Методика тестирования приемопередатчика РСІ Express // Труды НИИСИ РАН. 2016. Т. 6. № 1. С. 98-101.
- [7] Fucuda K., Yamashita H., Yuki M., et al. An 8Gb/s transceiver with 3x-oversampling 2-threshould eye-tracking CDR circuit for -36.8dB loss backplane // IEEE International of Solid-State Circuits Conference. 2008. SES. 5. P. 98-99.
- [8] Hidaka Y., Horie T., Koyanagi Y., et al. A 4-channel 10.3Gb/s transceiver with adaptive phase equalizer for 4-to-41dB loss PCB channel // IEEE International of Solid-State Circuits Conference. 2011. SES. 20. P. 346-347.

A 4-channel Multi-Standard Adaptive Serial Transceiver for the Range 1.25–10.3Gb/s in CMOS 65nm

A.V. Larionov, O.N. Buyakova, O.V. Sysoeva, S.E. Osina, S.O. Zadiabin, P.A. Aleksan, I.V. Tarasov, Yu.B. Rogatkin, V.V. Masterov

Scientific Research Institute for the System Analysis, alar@cs.niisi.ras.ru

Abstract — The article presents a fully adaptive serial 4channel transceiver for the standards 10G Base-R, SATA, PCI Express, FC, RapidIO, DisplayPort. The unit is capable of operating in the range 1.25+10.3125Gb/s with a bit error rate less 10⁻¹². The total jitter of the output signal transmitter <26ps@10.3125Gb/s for BER=10⁻¹². At 10.3125Gb/s receiver tolerance to high-frequency jitter >0.3UI (30% of the unit interval) with channel loss 24dB for polynomial PRBS31. The transceiver demonstrates error-free data processing at a speed 10.3125Gb/s for a channel with a total attenuation of 33dB. The unit is designed using CMOS 65nm technology and has power consumption of 760mW@10.3125Gb/s. The features of this transceiver include: 1) the decision feedback equalizer with direct access with reduced power consumption, 2) the controller for searching for the optimum position of the auxiliary clock phase to increase the receiver's tolerance, 3) the pseudo-differential cascode output buffer, 4) the controller of the minimally admissible differential voltage of the output buffer [5].

Power hungry direct feedback DFE (Decision-Feedback Equalizer) in [1] is implemented on CML (Current Mode Logic). In [2] consumption is reduced by replacing the resistive load of the CML elements with the active inductor, but the power consumption is still significant. In [3] DFE is fully implemented on domino logic, significantly reducing the power. However, this approach leads to a decrease in the speed. Moreover, the circuit is clocked by complex-shaped clock signals, increasing the consumption of the clock driver. DFE implemented in this work, forms data in the feedback loop for the first tap and the remaining taps in parallel. The path of the critical first tap is implemented in CML logic with an active inductance in the load. The remaining paths use the Sense Amplifier, significantly reducing power costs without degrading performance.

The article presents a technique that allow finding the optimal position of the auxiliary clock phase, which allows to register more actual information about the ISI (Inter Symbol Interference) value of the input signal. The error of in the calculated coefficients DFE and CTLE (Continuous Time Linear Equalizer) is reduced, to increase the receiver's tolerance.

The feature of our pseudo-differential output buffer is an auxiliary current source that reduces the frequencydependent nature of the circuit, which leads to a decrease in the deterministic jitter of the output signal. *Keywords* — equalizer, decision-feedback equalizer, receiver, transceiver, DFE, direct-feedback architecture, active inductor, CML, SSLMS, sign-sign least mean square.

REEFERENCES

- [1] Erba S. Multi-standard transceiver for NRZ serial communication up to 14Gbps // IEEE Signal and Power Integrity. 2012. P. 17-19.
- [2] Larionov A.V. Jekvalajzer s reshajushhej obratnoj svjaz'ju i aktivnoj induktivnost'ju dlja vysokoskorostnogo priemnika (A high-speed receiver with adaptive auxiliary clock for high loss backplane channels) // VII Vserossijskaja nauchno-tehnicheskaja konferencija «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem-2016 (MJeS-2016)» Sbornik nauchnyh trudov. / Pod red. A.L. Stempkovskogo – M.: IPPM RAN, 2016. Chast' III - S. 2-7.
- [3] Zhong F., Quan S., Liu W., et al. A 1.0625 ~ 14.025 Gb/s multi-media transceiver with full-rate source-seriesterminated transmit driver and floating-tap decisionfeedback equalizer in 40nm CMOS // IEEE Journal of Solid-State Circuits. 2011. V. 46. № 12. P. 3126-3139.
- [4] Nikolic B., Oklobdzija V., Stojanovic V. Improved senseamplifier-based flip-flop: design and measurements // IEEE Journal of Solid-State Circuits. 2000. V. 35. № 6. P. 876-884.
- [5] Larionov A. V. Adaptivnyj jekvalajzer s kontrollerom minimal'no dopustimogo differencial'nogo naprjazhenija vyhodnogo signala i psevdo-differencial'nym kaskodnym vyhodnym buferom dlja 10 Gb/s peredatchika po tehnologii 65 nm (Adaptive equalizer with a controller of a minimally admissible differential voltage of the output signal and pseudodifferetial cascode output buffer for the 10-Gb/s transmitter according to the 65-nm CMOS technology) // Mikrojelektronika. 2015. T. 44. № 6. S. 464-471.
- [6] Aleksan P. A. Metodika testirovanija priemoperedatchika PCI Express (Methodology of testing of the PCI Express transceiver) // Trudy NIISI RAN. 2016. T. 6. № 1. S. 98-101.
- [7] Fucuda K., Yamashita H., Yuki M., et al. An 8Gb/s transceiver with 3x-oversampling 2-threshould eye-tracking CDR circuit for -36.8dB loss backplane // IEEE International of Solid-State Circuits Conference. 2008. SES. 5. P. 98-99.
- [8] Hidaka Y., Horie T., Koyanagi Y., et al. A 4-channel 10.3Gb/s transceiver with adaptive phase equalizer for 4-to-41dB loss PCB channel // IEEE International of Solid-State Circuits Conference. 2011. SES. 20. P. 346-347.

Развитие структуры и алгоритма работы устройства встроенного саморемонта статической оперативной памяти

Л.А. Щигорев

ЗАО НТЦ «Модуль»

Национальный исследовательский ядерный университет «МИФИ», l.shchigorev@module.ru

Аннотация — Предложена новая структура устройства саморемонта встроенного памяти с резервными элементами и алгоритм проведения операции поиска корректной конфигурации резервных элементов. Используется резервирование запасными столбцами. Показано преимущество разбиения информационного слова, благодаря которому уменьшается время проведения саморемонта для различных способов замены поврежденных организации элементов. Проведено функциональное моделирование и логический синтез предложенных моделей. Приведены оценки дополнительных затрат на аппаратуру и задержек прохождения сигнала для различных вариантов реализации блока памяти.

Ключевые слова — ремонт памяти, анализатор ремонта, самотестирование памяти, система на кристалле (СнК), резервирование, запасные столбцы, статическая оперативная память (СОЗУ).

I. Введение

памяти современных Элементы систем на кристалле (СнК) занимают значительную площадь. Эксперты группы Semico Research Corp дают оценку, что в ближайшем будущем до 75% площади современных СнК будут заняты элементами статических оперативных запоминающих устройств (СОЗУ) [1]. В различных реализациях реальных устройств эта величина может варьироваться от 50% до 95% площади микросхемы. Они становятся основными источниками дефектов, определяющими выход годных (ВГД) микросхем. Поэтому в процессе разработки подсистем памяти, в частности для блоков объемом от единиц килобайт, необходимо прибавлять основным элементам памяти резервные. к Неиспользованные после производства резервные элементы могут заменить поврежденные ячейки памяти во время эксплуатации микросхемы [2]. Также для интегральных микросхем, изготовленных по суб-100-нм технологическим процессам, наличие резервных элементов помогает нивелировать резкий рост отказывающих ячеек СОЗУ при падении напряжения питания. На рис. 1 [3] приведена зависимость вероятности повреждения ячейки памяти Р от напряжения питания Vdd, из которой авторами сделан вывод о 10-кратном росте вероятности отказа ячейки памяти при 50 мВ падении напряжения питания.



Рис. 1. Зависимость вероятности отказа ячейки памяти от напряжения питания

Снижение влияния понижения напряжения питания при помощи резервных элементов памяти использовано, в частности, при проектировании L3-кэш памяти объемом 20 МБ в процессоре Intel Xeon[4]. Поэтому резервные элементы могут как повышать ВГД микросхем, так и служить фактором, улучшающим отказоустойчивость.

II. САМОРЕМОНТ ПАМЯТИ

А. Способы хранения конфигурации резервных элементов

От способа хранения конфигурации резервных зависит многократной элементов возможность реконфигурации памяти. Программный способ выполнения замены поврежденных элементов памяти резервными (soft repair) отличается от аппаратного (hard repair) возможностью многократно генерировать конфигурацию резервных элементов. При реализации способа аппаратного конфигурация резервных элементов генерируется однократно, обычно в процессе производственного контроля, и записывается в энергонезависимую память при помощи плавких предохранителей или перемычек, электронных или лазерных. Программный же способ предполагает хранение информации о конфигурации резервных элементов в энергозависимом регистре. Информация о реконфигурации памяти попадает туда после выполнения операции самотестирования в результате

исполнения алгоритма, расположенного в энергонезависимом ПЗУ, или работы встроенного блока саморемонта памяти. После выключения или перезагрузки микросхемы информация о конфигурации резервных элементов исчезает из энергозависимых регистров.

В. Устройства встроенного самотестирования

Устройства встроенного самотестирования (УВСТ) призваны диагностировать нуждающиеся в ремонте блоки памяти. Они могут отличаться набором производимых тестов памяти, но в задаче саморемонта главным аспектом является не это. Существенные отличия в построении структуры и алгоритма работы устройства встроенного саморемонта (УВСР) вносит способ, при помощи которого УВСТ информирует о результате проведенного тестирования. Если УВСТ снабжен выходным вектором ошибки, то УВСР может по результатам первого тестирования сделать вывод о присутствии дефектов в основных элементах. При наличии дефектов можно сделать вывол 0 достаточности резервных элементов для замены. Если число дефектов не превышает количество резервных элементов, то память будет сконфигурирована для замены поврежденных элементов, а повторное подтвердит работоспособность тестирование комбинации основных и резервных ячеек. Однако УВСТ может не обладать таким диагностическим выходным портом, а выдавать только однобитный статус тестирования: «Без ошибок / С ошибками». В этом случае поиск неисправных основных элементов памяти придется осуществлять методом перебора [5]. Описываемое в данной работе УВСР булет ориентировано на УВСТ с бинарной индикацией результата самотестирования.

С. Время, необходимое для поиска конфигурации резервных элементов

Отсутствие выходного вектора ошибки УВСТ увеличивает время поиска в зависимости от размера информационного слова [6]. Однако время будет также зависеть от типов и количества резервных элементов. В данном случае рассматриваются резервные столбцы.

Количество возможных комбинаций резервных следовательно, и максимальное элементов, а, количество операций самотестирования будет зависеть от способа замены. Рассматриваются два способа замены: сдвиг [7] и мультиплексирование [8]. При использовании сдвига блок памяти делится на области, элементы которых могут быть заменены одним резервным. Т.е. по количеству резервных элементов организуются внутренние части с одним резервным элементом. Например, для двух резервных столбцов блок будет разделен на младшую и старшую части информационного слова. В каждой из частей можно заменить один основной элемент. Максимальное число самотестирования операций для сдвига Zsh определяется по формуле 1:

$$Z_{sh} = \left(\frac{N}{r}\right)^r + 1 , \qquad (1)$$

где N – размерность информационного слова, а r – количество резервных элементов.

При использовании мультиплексирования замена происходит по принципу «любой на любой», т.е. любой основной элемент может быть заменен любым резервным. Максимальное число операций самотестирования для мультиплексирования Z_{mux} будет равно числу сочетаний из основных элементов по количеству резервных. Значение вычисляется по формуле 2:

$$Z_{mux} = \frac{N!}{r!(N-r)!} + 1, \qquad (2)$$

где N – размерность информационного слова, а r – количество резервных элементов.

Чем больше размерность информационного слова, тем больше возможных конфигураций резервных элементов. Разбиение блока памяти на блоки, содержащие меньшие по размеру информационные слова, приводит к общему сокращению максимального числа операций самотестирования и, как следствие, времени поиска поврежденных основных элементов.

Пусть Y_{sh} – величина, равная отношению Z_{sh} при реализации блока памяти из одного массива к Z_{sh} при реализации блока памяти той же размерности из нескольких массивов меньшей размерности. Эту величину можно вычислить по формуле 3:

$$Y_{sh} = \frac{\left(\frac{kN}{r}\right)^r + 1}{\left(\frac{N}{r}\right)^r + 1} \approx k^r , \qquad (3)$$

где N – размерность информационного слова, k – количество блоков меньшей размерности, из которых собран блок памяти, а r – количество резервных элементов.

Пусть Y_{mux} – величина, равная отношению Z_{mux} при реализации блока памяти из одного массива к Z_{mux} при реализации блока памяти той же размерности из нескольких массивов меньшей размерности. Эту величину можно вычислить по следующей формуле:

$$Y_{mux} = \frac{\frac{(kN)!}{r!(kN-r)!+1}}{\frac{(N)!}{r!(N-r)!+1}} \approx \prod_{i=0}^{r-1} \frac{kN-i}{N-i}.$$
 (4)

Таблица 1

Z, Y при различных N и r

| N | 8 | 8 | | 32 | | 128 |
|---------------------|----|----|-----|-------|------|----------|
| r | 2 | 4 | 2 | 4 | 2 | 4 |
| Сдвиг | | | | | | |
| Z_{sh} | 17 | 17 | 257 | 4097 | 4097 | 1048577 |
| \mathbf{Y}_{sh} | 1 | 1 | 16 | 256 | 256 | 65536 |
| Мультиплексирование | | | | | | |
| Z _{mux} | 29 | 71 | 497 | 35961 | 8129 | 10668001 |
| Y _{mux} | 1 | 1 | 17 | 513 | 290 | 152400 |



Рис. 2. Схема соединения УВСР, УВСТ и М массивов памяти с резервными элементами

В табл. 1 представлены величины Z и Y для различных N (8, 32 и 128 бит) и г (2 и 4 резервных столбцов). По эти данным можно сделать вывод, что разбиение блока памяти на несколько массивов меньшей разрядности может уменьшить максимальное число операций самотестирования для поиска корректной конфигурации резервных элементов в десятки тысяч раз для сдвига и сотни тысяч раз для мультиплексирования.

III. ЦЕЛЬ ИССЛЕДОВАНИЯ

Разрабатываемое УВСР должно осуществлять саморемонт памяти, имея информацию только об успешном/неуспешном прохождении операции самотестирования при заранее установленной конфигурации основных и резервных элементов памяти с помощью шины конфигурации резервных элементов (ШКРЭ). Поиск расположения поврежденных элементов производится путем перебора.

В соответствии с формулами 3 и 4 разбиение блока памяти на несколько блоков, содержащих меньшие по разрядности информационные слова, сократит максимальное количество операций саморемонта. В связи с этим актуальна разработка УВСР, предназначенных для групп массивов. В ходе работы будут получены данные по аппаратурным затратам и максимальным задержкам прохождения сигнала.

Целью данного исследования является развитие структуры и алгоритма работы УВСР, изложенных в [9].

IV. УСТРОЙСТВО ВСТРОЕННОГО САМОРЕМОНТА ПАМЯТИ

В составе СнК УВСР соединено с УВСТ и блоками СОЗУ с резервными элементами по схеме, изображенной на рис. 2. В качестве интерфейса с управляющим устройством (УУ) выступают входной сигнал старта операции саморемонта (СРСТРТ), выходные сигналы окончания операции саморемонта (СРФНШ) и статуса (СРСТАТ). УВСР соединен с УВСТ сигналами старта (СТСТРТ) и окончания (СТФНШ) операции самотестирования, а также статуса каждого массива памяти (СТСТАТ[i]). УВСР соединено с каждым из массивов памяти (ШКРЭ[i]). УВСТ соединено с каждым из массивов шиной записи тестовых данных (ТДз), шиной тестового адреса (ТА) и шинами чтения тестовых данных (ТДч).

Предлагаемая структура УВСР показана на рис. 3. Управление операцией саморемонта осуществляется при помощи автомата состояний (АС). Для анализа операции самотестирования введен статуса Mэлемент И. Для генерации входовый новой конфигурации резервных элементов необходимы операций самотестирования счетчик (COCT), дешифратор (ДШ СОСТ-КРЭ), счетчик обработанных блоков (СОБ), М регистров хранения конфигурации резервных элементов (РКРЭ). Для разрешения записи новой конфигурации резервных элементов введены демультиплексор из 1 в М (DMUX), М элементов 2входовое-И.



Рис. 3. Структура УВСР

Новая структура УВСР отличается от предлагаемой ранее отсутствием регистров хранения статус операции самотестирования (теперь они содержатся в УВСТ), наличием М-входового элемента И, а также отсутствием счетчика исправных блоков (СИБ).

Алгоритм работы предлагаемого УВСР изображен на рис. 4. АС принимает сигнал от УУ о провести COCT необходимости саморемонт. устанавливается в 0. После завершения первой (и любой последующей) операции самотестирования АС устанавливает СОБ в 0 и анализирует выход Мвходового элемента И. Если он равен 1, то саморемонт завершился успешно (СРФНШ = 1, СРСТАТ = 1). После первой операции самотестирования ЭТО означает, что все основные ячейки памяти работают корректно и замену производить не надо. После любой последующей это означает идентификацию и замену всех неисправных элементов. Если выход М-входового элемента И равен 0, то это означает наличие ошибок в памяти. АС начинает инкрементировать СОБ. Для массивов с отрицательным статусом прохождения теста в соответствующие РКРЭ будут записаны новые конфигурации резервных элементов. Если тест завершился успешно, то РКРЭ не будет перезаписан. Для этого введена схема разрешения записи в РКРЭ, состоящая из DMUX и М элементов 2-входовое-И. По достижении СОБ максимального количества блоков (МКБ) будет инкрементирован СОСТ. Затем будет осуществлена проверка достижения СОСТ максимального числа операций самотестирования (МОСТ). Если все возможные комбинации резервных элементов протестированы, то AC сигнализирует о неуспешной операции саморемонта (СРФНШ = 1, СРСТАТ = 0). В противном случае будет инициирована новая операция самотестирования.

V. МОДЕЛИРОВАНИЕ И АНАЛИЗ РЕЗУЛЬТАТОВ

Для исследования был выбран блок памяти, хранящий 4К 128-разрядных информационных слов. Этот блок был собран пятью способами: (А) из 16-и 8разрядных блоков, (Б) 8-и 16-разрядных, (В) 4-х 32разрядных, (Г) 2-х 64-разядных и (Д) 1-го 128разрядного массивов. Каждый из блоков содержит два резервных столбца, замена осуществляется сдвигом.

Был проведен синтез моделей УВСР, разработанных на языке Verilog HDL, для оценки аппаратурных затрат и максимальных задержек прохождения сигнала. Результаты получены в САПР Cadence Encounter RTL Compiler для проектнотехнологической нормы 28 нм КМОП при нормальных условиях ($V_{\text{пнт}} = 1,0$ В; T = 25 °C). Результаты приведены в табл. 2. Данные предоставлены с учетом аппаратурных затрат на УВСТ, реализующего алгоритм тестирования памяти March-LR [10].





Таблица 2

| Способ реализаци и блока 4К*128 | S _{увср+ув} ст, МКМ ² | t ₃ , пс | $S_{ybcp+ybct+p3}$ / $S_{память}$, % | $S_{yBCP+yBCT/} S_{\Pi AM STL}, \%$ |
|--|---|---------------------|--|-------------------------------------|
| А | 2291 | 385 | 17,92 | 1,22 |
| Б | 2164 | 319 | 13,14 | 1,55 |
| В | 1858 | 300 | 9,32 | 1,60 |
| Г | 1729 | 292 | 6,93 | 1,67 |
| Д | 1622 | 290 | 5,07 | 1,52 |

Экспериментальные результаты

В таблице 2 представлены две относительные величины. Первая – отношение площади всей дополнительной аппаратуры, необходимой для реализации резервирования с учетом дополнительных к площади незащищенной столбцов, памяти. Наибольшие аппаратурные затраты на резервирование – 18% – для способа А в связи с тем, что при этой реализации число резервных столбцов максимальное – 32. Наименьшие – 5% – для способа Д – всего 2 резервных столбца. Вторая относительная величина отношение площади блоков УВСР и УВСТ к площади, занимаемой матрицей памяти с учетом резервных столбцов. В этом случае величина не превышает 1,7% для любой реализации. Наименьшее значение – 1,22% – у способа А, т.к. площадь блока памяти при реализации из 16-и 8-разрядных массивов наибольшая [11].

В ходе синтеза показано. что плошаль дополнительной аппаратуры при реализации способом А наибольшая – 2291 мкм². Это связано с наибольшим количеством массивов – 16, образующих блок 4000 x 128. Наибольший критический путь, вносящий задержку прохождения сигнала, также наблюдается для реализации способом А. Однако именно этот вариант реализации требует наименьшего максимального количества операций самотестирования для поиска поврежденных основных элементов и корректной конфигурации резервных – 17.

VI. Выводы

В данной статье рассмотрены новые структура и алгоритм работы УВСР, ориентированного на бинарную индикацию результата операции самотестирования.

Новая структура УВСР отличается от предлагаемой ранее наличием М-входового элемента И. Благодаря ему определение исправности всех блоков происходит непосредственно после окончания тестирования. Поэтому УВСР не нужны регистры хранения статуса операции самотестирования, а также счетчик исправных блоков. Предложенные VBCP описаны на языке Verilog HDL. Проведено функциональное моделирование и синтез для проектно-технологической нормы 28 нм КМОП. Представлены абсолютные величины площадей, занимаемые VBCP и VBCT, а также максимальные величины задержек на критических путях. Показано, что дополнительные затраты на резервные элементы памяти и блоки VBCP и VBCT не превышают 18%.

ЛИТЕРАТУРА

- URL: http://www.semico.com/content/worldwide-socmarket-forecast-approach-200-billion-2019-says-semicoresearch (дата обращения: 01.03.2018)
- [2] Краснюк А.А., Петров К.А. Особенности применения методов помехоустойчивого кодирования в суб-100нм микросхемах памяти для космических систем // Проблемы разработки перспективных микро- и наноэлектронных систем - 2012. Сб. тр. / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2012. №1. С. 638-641.
- [3] B. Zimmer, P.-F. Chiu, B. Nikolić and Krste Asanović Reprogrammable redundancy for SRAM-based cache V_{min} reduction in a 28-nm RISC-V processor // IEEE Journal of Solid-State Circuits. 2017. V. 52. № 10. P. 2589–2600.
- [4] M. Huang, M. Mehalel, R. Arvapalli and S. He An Energy Efficient 32-nm 20-MB Shared On-Die L3 Cache for Intel® Xeon® Processor E5 Family // IEEE Journal of Solid-State Circuits. 2013. V. 48. № 8. P. 1954–1962.
- [5] Nordholz P., Otterstedt J., Niggemeyer D. A Defect-Tolerant Word-Oriented Static RAM with Built-In Self-Test and Self-Reconfiguration // 8th Annual IEEE International Conference of Innovative Systems In Silicon. 1996. P. 124-132.
- [6] Щигорев Л.А. Применение шины диагностики в задаче саморемонта блоков статической оперативной памяти // Нано- и микросистемная техника 2018. Т.20. №2. С. 98-106.
- [7] Щигорев Л.А. Методы исправления последствий отказов в блоках статической оперативной памяти // Тр. НИИСИ РАН. 2017. Т.2, № 2. С. 110-114.
- [8] Nicolaidis M., Achouri N., Boutobza S. Optimal reconfiguration functions for column of data-bit built-in self-repair // Design, Automation and Test in Europe Conference and Exhibition. Proc. 2003. P. 590-595.
- [9] Щигорев Л.А. Организация саморемонта блоков статической оперативной памяти с резервными элементами // Проблемы разработки перспективных микро- и наноэлектронных систем - 2016. Сб. тр. / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН. 2016. №3. С. 178-185.
- [10] Ad J. Van de Goor, G. N. Gaydadjiev, V. G. Mikitjuk, V. N. Yarmolik March LR: a test for realistic linked faults // 14th VLSI Test Symosium, Proceedings. 1996. P. 272-280.
- [11] Shchigorev L.A., Shagurin I.I. Combined methods of tolerance increasing for embedded SRAM // 1st International Telecommunication Conference "Advanced Micro- and Nanoelectronic Systems and Technologies" / IOP Conference Series: Materials Science and Engineering. 2016 V. 151, №1.

Structure and Algorithm Development of Built-in Self-repair for SRAM

L.A. Shchigorev

RC Module

National Research Nuclear University «MEPHI», 1.shchigorev@module.ru

Abstract — Experts from the Semico Research Corp group estimate that soon up to 75% of the area of modern systemon-chips (SoC) will be occupied by elements of static random access memory (SRAM). Therefore, in the process of developing memory subsystems, it is necessary to add redundant memory elements. It's the most commonly used technique for memory yield improvement. But if the existing redundancy was not used in such a way, it can be used for the replacement of the faulty cells in the future. Also, for integrated circuits manufactured using sub-100-nm process, the availability of spare elements helps to level out the sharp growth of the failing SRAM cells number with the operating voltage reducing. To be able to reconfigure the memory during operation, only soft repair method can be used. This article is devoted to the repair of SRAM with redundant columns.

Two methods of substitution are considered: shift and multiplexing. When using a shift, the memory block is divided into areas whose elements can be replaced by a single spare one. When multiplexing is used, the replacement takes place according to the principle of "anyone on any," i.e., any primary element can be replaced by any reserve element.

The self-repair operation is preceded by the self-testing operation. The method of redundancy analysis depends on the way of the built-in self-test (BIST) status information producing. The proposed built-in self-repair (BISR) scheme interacts with the BIST unit, which has the only single-bit status signal. Therefore, the search of CRV is executed by using the exhaustive search. The equations for calculating the maximum number of self-test operations are presented in the article. For reducing the maximum number of test iterations, the word width dividing is provided.

Improved structure and algorithm for the device self-repair are presented in the article. For the area and timing penalties estimation, 4Kx128 memory block was selected. Five variants of memory organization were investigated: consisting from 8, 16, 32, 64 and 128-bit arrays. All blocks have two redundant columns – one column per each half of the word. BISR block was written in Verilog HDL. Functional modeling and synthesis for the for 28 nm process CMOS were carried out. The absolute values of the areas occupied by the BISR and BIST are presented, as well as the maximum values of the delays on critical paths. It is shown that the additional area costs for the redundant memory elements and the BISR and BIST units do not exceed 18%.

Keywords — memory repair, memory repair analyzer, selftest, system-on-chip (SoC), redundancy, spare columns, SRAM.

REFERENCES

- URL: http://www.semico.com/content/worldwide-socmarket-forecast-approach-200-billion-2019-says-semicoresearch (access data: 01.03.2018)
- [2] Krasnyuk A.A., Petrov K.A. Osobennosti primenenija metodov pomehoustojchivogo kodirovanija v sub-100nm mikroshemah pamjati dlja kosmicheskih sistem (Features of application ECC methods in sub-100 nm SRAMs for space systems) // Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem - 2012. Sb. tr. / pod obshh. red. akademika RAN A.L. Stempkovskogo. M.: IPPM RAN, 2012. №1. S. 638-641.
- [3] B. Zimmer, P.-F. Chiu, B. Nikolić and Krste Asanović Reprogrammable redundancy for SRAM-based cache Vmin reduction in a 28-nm RISC-V processor // IEEE Journal of Solid-State Circuits. 2017. V. 52. № 10. P. 2589–2600.
- [4] M. Huang, M. Mehalel, R. Arvapalli and S. He An Energy Efficient 32-nm 20-MB Shared On-Die L3 Cache for Intel® Xeon® Processor E5 Family // IEEE Journal of Solid-State Circuits. 2013. V. 48. № 8. P. 1954–1962.
- [5] Nordholz P., Otterstedt J., Niggemeyer D. A Defect-Tolerant Word-Oriented Static RAM with Built-In Self-Test and Self-Reconfiguration // 8th Annual IEEE International Conference of Innovative Systems In Silicon. 1996. P. 124-132.
- [6] Shchigorev L.A. Primenenie shiny diagnostiki v zadache samoremonta blokov staticheskoj operativnoj pamjati (Built-in Self-Repair Application of a Diagnostic Bus) // Nano- i mikrosistemnaja tehnika 2018. T.20. №2. S. 98-106.
- [7] Shchigorev L.A. Metody ispravlenija posledstvij otkazov v blokah staticheskoj operativnoj pamjati (Methods of failures eliminating in SRAM) // Tr. NIISI RAN. 2017. T.2, № 2. S. 110-114.
- [8] Nicolaidis M., Achouri N., Boutobza S. Optimal reconfiguration functions for column of data-bit built-in self-repair // Design, Automation and Test in Europe Conference and Exhibition. Proc. 2003. P. 590-595.
- [9] Shchigorev L.A. Organizacija samoremonta blokov staticheskoj operativnoj pamjati s rezervnymi jelementami (Built-in self-repair for SRAM with redundant elements) // Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem - 2016. Sb. tr. / pod obshh. red. akademika RAN A.L. Stempkovskogo. M.: IPPM RAN. 2016. №3. S. 178-185.
- [10] Ad J. Van de Goor, G. N. Gaydadjiev, V. G. Mikitjuk, V. N. Yarmolik March LR: a test for realistic linked faults // 14th VLSI Test Symosium, Proceedings. 1996. P. 272-280.
- [11] Shchigorev L.A., Shagurin I.I. Combined methods of tolerance increasing for embedded SRAM // 1st International Telecommunication Conference "Advanced Micro- and Nanoelectronic Systems and Technologies" / IOP Conference Series: Materials Science and Engineering. 2016 V. 151, №1.

Самосинхронный D-триггер с «защелкой»

А.А. Старых, Е.Б. Лукьяненко

Инжиниринговый центр приборостроения, радио- и микроэлектроники ЮФУ, г. Taranpor anastasya.staryh@mail.ru, luk101010@mail.ru

Аннотация — Рассмотрены одно- и двухступенчатые самосинхронные D-триггеры с «защелкой». Предложена реализация схемы индикатора переходных процессов результаты Приведены блочным метолом. **D-триггеров**, моделирования самосинхронных выполненных на стандартных логических элементах «И», «ИЛИ», «И-НЕ», «ИЛИ-НЕ» и разработанных самосинхронных D-триггеров с «защелкой». Показана перспективность использования предложенных Dтриггеров в самосинхронных последовательностных схемах.

Ключевые слова — самосинхронный элемент, D-триггер, блочная структура, индикатор переходных процессов, рассеиваемая мощность, быстродействие схемы, энерготопологический критерий.

I. Введение

Главными преимуществами самосинхронных схем (СС-схем) над синхронными являются их свойства отказобезопасности и отсутствия «гонок сигналов» при любых конечных задержках элементов. Использование самосинхронной схемотехники (СС-схемотехники) является действенным методом увеличения быстродействия цифровых электронных схем, повышения их энергоэффективности.

В последовательностных СС-схемах D-триггер принимается за элементарную самосинхронную запоминающую ячейку (ССЗЯ). В СС-схемотехнике к ССЗЯ предъявляются следующие основные требования [1]:

1) ССЗЯ должна хранить один бит информации после окончания фазы, в которой произошла запись этого бита.

2) Окончание переходных процессов должно индицироваться в обеих фазах – рабочей и спейсере.

3) ССЗЯ должна иметь максимальное быстродействие и/или минимальную сложность (в транзисторах) среди имеющихся вариантов.

Исходя из вышеупомянутых требований видно, что оптимизация ССЗЯ направлена на увеличение их быстродействия снижения транзисторной И избыточности. В книгах известного специалиста в области СС-схем В.И. Варшавского [2, 3] довольно подробно рассмотрены вопросы построения последовательностных СС-схем. Однако со стороны практической схемотехники материалы эти недостаточны, а иногда и неприемлемы. В них

отсутствует анализ схем по критериям оптимальности. Некоторые предлагаемые схемы либо вообще не могут быть использованы в двухфазной дисциплине, либо неэффективны по быстродействию и/или затратам [1].

Примем во внимание известную классификацию ССЗЯ по основным признакам, существенным для ССсхемотехники [1]:

1) Длительность хранения записанной информации – в каких последующих фазах она хранится: только в рабочей фазе (одноступенчатая ячейка) или в спейсере и рабочей фазе (двухступенчатая ячейка).

- 2) Наличие управляющего сигнала.
- 3) Значение входного спейсера.

Все рассматриваемые в работе самосинхронные Dтриггеры имеют двухпроводную организацию, их входы – D, \overline{D} являются парафазными (ПФС– входами), а выходы – Q, \overline{Q} бистабильными (БС– выходы). На выходе каждого D-триггера имеется индикатор переходных процессов, необходимый СС– схемам для фактического определения окончания рабочей и спейсерной фаз.

II. САМОСИНХРОННЫЙ ОДНОСТУПЕНЧАТЫЙ D– ТРИГГЕР НА СТАНДАРТНЫХ ЛОГИЧЕСКИХ ЭЛЕМЕНТАХ

Схема одноступенчатой ячейки – управляемого Dтриггера [1, 2] приведена на рис. 1.



Рис. 1. Схема самосинхронного D-триггера на стандартных логических элементах

На рис. 1 обозначено: S – управляющий сигнал, чередующий фазы записи и хранения информации; D информационный сигнал; Q – выходной сигнал; I – индикаторный сигнал, показывающий окончание рабочего процесса в тригтере.

Схема состоит из синхронизирующего элемента (микросхемы DD1, DD2), запоминающего элемента (микросхемы DD3, DD4) и схемы «Исключающее ИЛИ» с инверсией (микросхемы DD5 – DD7), служащей для формирования индикаторного сигнала. Входной управляющий сигнал «S» позволяет реализовать в D-триггере двухфазный протокол работы – записать данные в ячейку в требуемой фазе (в данном случае - спейсерной), хранить данные в ячейке в рабочей фазе и предотвратить «гонки сигналов». Индикаторный сигнал «I» индицирует входные (D, \overline{D}) и выходные (Q, \overline{Q}) сигналы, регистрируя окончание переходных процессов в D-триггере.

III. Разработанный одноступенчатый самосинхронный D-триггер с «зашелкой»

Аналогичное функционирование можно получить, используя для запоминания «защелку», выполненную на двух инверторах с положительной обратной связью, а для управления – набор транзисторов, которые могут иметь на выходе высокоомное состояние Z (рис. 2).



Рис. 2. Схема самосинхронного одноступенчатого D-триггера с «защелкой»

Транзисторы VT1, VT4 служат для управления режимами записи и хранения. В режиме записи (логический «0» на входе управляющего сигнала «S») транзисторы VT1, VT4 открываются и происходит запись информационного сигнала D в «защелку», выполненную на инверторах DD1, DD2. В режиме хранения (логическая «1» на входе S) транзисторы VT1, VT4 закрываются, и «защелка» запоминает входной сигнал D в прямой и инверсной форме.

Чтобы работоспособность обеспечить запоминающего элемента необходимо, чтобы токи насыщения МОП-транзисторов VT1 - VT4 и транзисторов инвертора DD1 были в 3-4 раза больше DD2. токов транзисторов инвертора Такое соотношение обеспечивается выбором токов топологических размеров транзисторов. Для транзисторов VT1 - VT4 и инвертора DD1 ширина затвора может быть выбрана равной 2,0 мкм и 1,2 мкм (для р- и п-канальных транзисторов, соответственно). Тогда МОП-транзисторы инвертора DD2 могут иметь ширину затвора 0,8 мкм (р-канальный транзистор) и 0,45 мкм (п-канальный транзистор). При этом длина канала всех транзисторов выбирается равной 0,18 мкм. Токи таких МОП-транзисторов равны 0,8 мА и 0,25 мА, соответственно. Значения токов получены в программе OrCAD при использовании модели МОПтранзисторов BSIM3. Периметр и площадь областей стока и истока рассчитаны по методике, предложенной в [4].

Схему «Исключающее ИЛИ» выполним по блочному методу без выходного инвертора (рис. 3) [5].



Рис. 3. Блочная схема «Исключающее ИЛИ» с инверсией

Использование блочного метода повышает энергоэффективность схемы.

IV. РЕЗУЛЬТАТЫ СХЕМОТЕХНИЧЕСКОГО МОДЕЛИРОВАНИЯ ОДНОСТУПЕНЧАТОГО D-ТРИГГЕРА

Работа самосинхронного одноступенчатого Dтриггера с «защелкой» исследовалась в САПР OrCAD и показана на рис. 4.



Рис. 4. График работы одноступенчатого D-триггера с «защелкой»

Как видно из графика, при переключении управляющего сигнала «S» из логической «l» в логический «0» происходит запись информационного сигнала D на выход триггера. При этом окончание рабочего процесса индицируется положительным фронтом индикаторного сигнала «I», а изменения входного сигнала «D» индицируются отрицательным фронтом индикатора «I».

Параметры разработанного одноступенчатого Dтриггера с «защелкой» и D-триггера на стандартных логических элементах измерялись в непрерывном режиме переключения входных сигналов и приведены в табл. 1.

D-триггеры сравниваются по быстродействию, средней рассеиваемой мощности, количеству транзисторов, работе переключения И энерготопологическому критерию. Последний параметр представляет собой произведение работы переключения на количество транзисторов и является обобщающей величиной, характеризующей энергоэффективность исследуемой схемы [5]. Как видно из таблицы 1 по всем измеренным параметрам D-триггер с «защелкой» превосходит известный аналог на стандартных логических элементах.

Быстродействие разработанного D-триггера с «защелкой» выше аналога в 2,03 раза, рассеиваемая мощность – в 2,06 раз, энергоэффективность – в 9,5 раз.

Таблица 1

Параметры самосинхронных одноступенчатых D-триггеров

| Элементы | D-триггер на | Разработанны |
|-----------------------------|--------------|---------------|
| | стандартных | й D-триггер с |
| Параметры | элементах | «защелкой» |
| Е пит, В | 5 | 5 |
| Рассеиваемая | | |
| мощность, Р _{ср} , | 0,33 | 0,16 |
| мВт | | |
| Кол-во | | |
| транзисторов в | 36 | 16 |
| схеме, N, шт. | | |
| Задержка | | |
| распространения, | 0,55 | 0,27 |
| τ _{3.p} , нс | | |
| Работа | | |
| переключения, А, | 0,18 | 0,04 |
| лДж | | |
| Энерготопологиче | | |
| ский критерий, L, | 6,53 | 0,69 |
| пДж∙шт | | |

V. САМОСИНХРОННЫЙ ДВУХСТУПЕНЧАТЫЙ D– ТРИГГЕР С «ЗАЩЕЛКОЙ»

Включая последовательно два одноступенчатых Dтригтера получим двухступенчатый тригтер с «защелкой», срабатывающий по отрицательному фронту. Для обеспечения самосинхронного режима необходимо определить окончание переходного процесса в первой ступени, переключить режим записи и хранения в обеих ступенях на противоположный и определить окончание переходного процесса во второй ступени. Все эти действия обеспечивает схема, показанная на рис. 5.



Рис. 5. Схема самосинхронного двухступенчатого Dтриггера с «защелкой»

Первая ступень триггера состоит из транзисторов VT1–VT4 и микросхем DD1, DD2, образующих «защелку». Выходные сигналы с первой ступени

снимаются с выходов DD1 (прямой сигнал Q1) и выхода микросхемы DD2 (инверсный сигнал O1). Вторая ступень триггера состоит из транзисторов VT5-VT8 и микросхем DD3, DD4 («защелка»). Выходные сигналы второй ступени – Q2 и Q2 Управление самосинхронной работой двухступенчатого триггера производится микросхемой DD5, выполняющей функцию «Исключающее ИЛИ-НЕ» над сигналами Q1 и Q2 и микросхемой DD6 логическое «И» между управляющим сигналом «S» и индикаторным сигналом «І». Эти микросхемы обеспечивают синхронизацию переключения режимов записи и хранения с длительностью переходного процесса в первой ступени триггера и выработку индикаторного сигнала, положительный фронт которого указывает на окончание рабочего процесса в триггере.

VI. РЕЗУЛЬТАТЫ СХЕМОТЕХНИЧЕСКОГО МОДЕЛИРОВАНИЯ ДВУХСТУПЕНЧАТОГО D-ТРИГГЕРА С «ЗАЩЕЛКОЙ»

Проверка работоспособности схемы проведена моделированием в САПР OrCAD.

Для наглядности результатов моделирования логические микросхемы DD5, DD6 выполнены на идеальных элементах, взятых из библиотеки Dig_abm, а в каждую ступень триггера введена дополнительная задержка 0,5 нс. Результаты моделирования приведены на рис. 6.



Рис. 6. График работы одноступенчатого D-триггера с «защелкой»

Работа тригтера начинается с появлением в момент времени t_0 спейсерного сигнала S положительной полярности. В это время индикаторный сигнал I равен логической единице, так как Q1=Q2=0. Поэтому сформированный спейсер S1 равен логической единице. При таком значении спейсера первая ступень D-тригтера находится в режиме записи, а вторая – в режиме хранения. В момент времени t_1 данные D записываются в первую ступень и появляются на выходе Q1. Теперь Q1=1, Q2=0, и индикаторный сигнал I становится равным логическому нулю. Тогда первая ступень триггера переходит в режим хранения, а вторая – в режим записи. В момент времени t_2 сигнал Q1 записывается во вторую ступень и выходной сигнал Q2 становится равным логической единице. Аналогично протекает процесс при записи логического нуля.

Параметры разработанного двухступенчатого Dтриггера с «защелкой» сравнивались с параметрами Dтриггера на стандартных логических элементах, приведенного на рис. 7 [1].



Рис. 7. Схема самосинхронного двухступенчатого Dтриггера на логических элементах

Первая ступень триггера включает микросхемы DD1, DD2, вторая – микросхемы DD3, DD4. Микросхема DD5 формирует индикаторный сигнал.

Параметры самосинхронных двухступенчатых Dтриггеров (разработанного с «защелкой» и на стандартных логических элементах) измерялись в непрерывном режиме переключения входных сигналов и приведены в табл. 2.

Таблица 2

Параметры самосинхронных двухступенчатых Dтриггеров

| Элементы | D-триггер на | Разработанны |
|-----------------------------|--------------|---------------|
| | стандартных | й D-триггер с |
| Параметры | элементах | «защелкой» |
| Е пит, В | 5 | 5 |
| Рассеиваемая | 0,75 | 0,41 |
| мощность, Р _{ср} , | | |
| мВт | | |
| Кол-во | 80 | 30 |
| транзисторов В | | |
| схеме, N, шт. | | |
| Задержка | 0,95 | 0,6 |
| распространения, | | |
| τ _{3.p} , нс | | |
| Работа | 0,71 | 0,25 |
| переключения, А, | | |
| пДж | | |
| Энерготопологиче | 57 | 7,4 |
| ский критерий, L, | | |
| пДж∙шт | | |

Как видно из таблицы 2 разработанный самосинхронный двухступенчатый D-триггер с «защелкой» имеет лучшие параметры, чем триггер на логических элементах. Быстродействие такого D-триггера с «защелкой» выше аналога в 1,58 раз,

рассеиваемая мощность – в 1,83 раза, энергоэффективность – в 7,7 раз.

VII. Инициализация состояния D-триггера с «защелкой»

Во многих практических приложениях необходимо задать начальное состояние триггера. Рассмотрим вопросы сброса и установки для разработанного Dтриггера. Особенностью схем с «защелкой» является то, что при подаче инициализирующего сигнала не должно возникать сквозного протекания тока в управляющей цепочке транзисторов. На рис. 8 приведена схема D-триггера с «защелкой», имеющая вход R для сброса в логический ноль обеих ступеней триггера.



Рис. 8. Схема D-триггера со сбросом

Сброс схемы в логический ноль происходит, если вход R находится в состоянии логического нуля. При этом благодаря транзисторам VT5 и VT11 цепочка транзисторов в нижней полуплоскости оказывается выключенной при любых сочетаниях сигналов на входах транзисторов VT3, VT4 и VT9, VT10. Транзисторы же VT6 и VT12 открываются, создавая на выходах Q1 и Q2 состояния логического нуля.

На рис. 9 приведен график работы D-тригтера с «защелкой» при наличии сигнала сброса R.



Рис. 9. График работы D-триггера с сигналом сброса

При нулевом значении сигнала R напряжения на выходах Q1 и Q2 равны нулю и не зависят от уровня входных сигналов. В момент времени t_1 значение R становится равным логической единице, транзисторы VT5, VT11 открываются, а транзисторы VT6, VT12 – закрываются, и схема начинает нормально функционировать.
Сбросить схему в логический ноль можно в любой момент времени. На рис. 10 приведен график работы D-триггера с «защелкой», в котором импульс сброса действует в течение времени от t₁ до t₂.



Рис. 10. График работы D-триггера с импульсным сигналом сброса

Как видно из графика при нулевом значении сигнала R сбрасываются в логический ноль выходы Q1 и Q2 триггера.

Для установки выходов D-триггера в логическую единицу можно применить схему, показанную на рис. 11.



Рис. 11. Схема D-триггера с установкой

Активный уровень сигнала установки (Set) – логическая единица. При значении сигнала Set, равном логической единице, транзисторы VT1, VT7 закрываются, а транзисторы VT6, VT12 открываются, благодаря чему выходные сигналы Q1 и Q2 устанавливаются в логическую единицу. Транзисторы VT1, VT7 защищают управляющие транзисторы от сквозного протекания тока.

График работы D-триггера с сигналом установки приведен на рис. 12.

Как видно из рис. 12 в моменты времени от t_1 до t_2 значение сигнала Set равно логической единице и выходные сигналы обеих ступеней триггера Q1 и Q2 устанавливаются в логическую единицу. В момент времени t_3 подан импульс установки положительной полярности длительностью 1 нс. Этот сигнал также

устанавливает выходные состояния триггера в логическую единицу.



Рис. 12. График работы D-триггера с сигналом установки Set

Схемы сброса и установки можно применить совместно. Тогда D-триггер будет полнофункциональным, имея функции сброса и установки.

VIII. Выводы

Разработка оптимизированного D-триггера с «зашелкой» дала возможность получить от предложенного устройства улучшенные рабочие характеристики. Из полученных результатов видно, что предложенные схемы D-триггеров, реализованные на основе «защелки» и схемы с Z-состоянием, превосходят D-триггеры на основе стандартных параметрам. логических элементов по всем Разработанная схема D-триггера имеет преимущества перед схемами с использованием RS - триггеров, поскольку в ней отсутствует запрещенное состояние.

Для разработанного самосинхронного D-триггера выполняются все требования, предъявляемые к ССЗЯ, поэтому его использование в последовательностных схемах можно считать перспективным.

ЛИТЕРАТУРА

- [1] Плеханов Л.П. Основы самосинхронных электронных схем. М.: БИНОМ. Лаборатория знаний, 2013. 208 с.
- [2] Варшавский В.И. Автоматное управление асинхронными процессами в ЭВМ и дискретных системах. М.: Наука, 1986. 398 с.
- [3] Астановский А.Г., Варшавский В.И., Мараховский В.Б. Апериодические автоматы. М: Наука, 1976. 424 с.
- [4] Ракитин В.В. Интегральные схемы на КМОПтранзисторах. М.: 2007. 307 с.
- [5] Старых А.А. Метод синтеза функциональных блоков комбинационных схем с использованием минтермов и макстермов // Электронная техника. Серия 2. Полупроводниковые приборы. 2015. № 1. С. 63–69.

Self-Timed D-Trigger with «Load/Latch»

A.A. Starykh, E.B. Lukyanenko

Engineering center of instrument making, radio- and microelectronics, Southern federal university, Taganrog anastasya.staryh@mail.ru, luk101010@mail.ru

Abstract — One- and two-step self-timed D-triggers with a "load/latch" are considered. All self-timed D-triggers considered in the work have a two-wire organization, their inputs are D, D are paraphase, and outputs are Q, Obistable. At the output of each D-trigger, there is an indicator of the transient processes necessary for the selftimed circuits to actually determine the end of the working and spacer phases. The realization of the circuit of the indicator of transients by the block method is proposed. The simulation results of self-timed D-triggers performed on standard logic elements «AND», «OR», «AND-NO», «OR-NO» and developed self-timed D-triggers with «load/latch» are presented. The development of an optimized D-trigger with a "load/latch" made it possible to obtain improved performance from the proposed device. The developed circuit of the D-trigger has advantages over the circuits using RS-triggers, since there is no forbidden state in it. For the developed self-timed D-trigger, all the requirements for selftimed store cell are the same, therefore it's usage in sequential circuits can be considered as promising. The twostep D-trigger with «load/latch», that triggers on the negative front, is obtained by the successive inclusion of two one-step D-triggers. To provide a self-timed mode, the end of the transient process is determined at the first stage, the recording and storage mode in both stages is switched to the opposite one and the end of the transient process in the second stage is determined. D-triggers are compared by speed, average dissipated power, number of transistors, switching operation and energy efficiency. The latter parameter is the product of the switching operation by the number of transistors and it is the generalizing value, which characterizes the energy efficiency of the circuit of interest. The performance of developed D-trigger with «load/latch»

and D-trigger on standard logic elements were measured in a continuous mode of switching input signals. The speed of developed one-step D-trigger with «load/latch» is higher than the compatible of 2,03 times, dissipated power -2,06 times, energy efficiency -9,5 times. The speed of developed twostep D-trigger with «load/latch» is higher than the compatible of 1,58 times, dissipated power -1,83 times, energy efficiency -7,7 times.

Keywords — self-timed element, D-trigger, block structure, indicator of transient, power dissipation, performance of the circuit, energy-topological criterion.

References

- [1] Plehanov L.P. Osnovy samosinhronnyh jelektronnyh shem (The basics of self-timed electronic circuits). M.: BINOM. Laboratorija znanij, 2013. 208 s.
- [2] Varshavskij V.I. Avtomatnoe upravlenie asinhronnymi processami v JeVM I diskretnyh sistemah (Self-timed control of concurrent processes: the design of aperiodic logical circuits in computers and discrete systems). M.: Nauka, 1976. 398 s.
- [3] Astanovskij A.G., Varshavskij V.I., Marahovskij V.B. Aperiodicheskie avtomaty (Aperiodic automats). M: Nauka,1976. 424 s.
- [4] Rakitin V.V. Integral'nye shemy na KMOP-tranzistorah (Integrated circuits in CMOS transistors). M.:, 2007.307 s.
- [5] Staryh A.A. Method sinteza funkcional'nyh blokov kombinacionnyh shem s ispol'zovaniem mintermov i makstermov (The method for the synthesis functional blocks of combinational circuits with the use minterms and maxterms) // Jelektronnaja tehnika. Serija 2. Poluprovodnikovye pribory. 2015. № 1. S. 63–69.

Способ организации автомата Мура с повышенной устойчивостью к мягким отказам

И.В. Егоров

Санкт-Петербургский политехнический университет Петра Великого, г. Санкт-Петербург ig-ego@mail.ru

Аннотация: снижение проектной нормы в производстве полупроводниковых структур повышает чувствительность цифровых устройств к попаданию частиц высоких энергий (в частности, при работе в условиях радиации), что приводит к возникновению мягких отказов – искажению информации при сохранении работоспособности аппаратуры. Известно, что наиболее действенным средством защиты от мягких является периодическая отказов перезапись искаженных данных корректными (восстановление информации). По этой причине традиционные способы повышения надежности, основанные на использовании структурной избыточности, оказываются малоэффективными в условиях мягких отказов. Цель: разработка новых способов структурной организации автомата Мура, работающего при потоке мягких отказов. Результаты: разработана структура автомата Мура с троированием памяти, мажорированием сигналов памяти и восстановлением выходных информации на каждом такте, а также оснащенная средствами регистрации количества мягких отказов, произошедших в ходе работы автомата.

Ключевые слова — автомат с памятью, комбинационная схема, анализ надежности, синхронизация, мягкие отказы, структурное резервирование, восстанавливаемые системы, вероятность безотказной работы.

I. Введение

Под мягким отказом понимается явление, при котором в элементе памяти вычислительной системы происходит искажение бита данных. Элемент памяти при этом остается работоспособным. Возникновение мягких отказов наиболее характерно при работе устройства в условиях повышенной радиации [1].

В опубликованных работах [2-4]проанализированы процессы в логических элементах и триггерах, выполненных по технологии КМОП (CMOS Fabrication), протекающие при воздействии радиации. Выявлено, что основной причиной возникновения мягких отказов является попадание частицы высокой энергии в МОП-транзистор, в результате которого происходит ионизация подзатворной области полупроводника, что в свою очередь приводит к проскакиванию импульса тока на выходе вентиля, в схему которого входит транзистор. Длительность импульса зависит от величины заряда неосновных

носителей, а он – от энергии частицы и технологических параметров вентиля. Обычно длительность импульса находится в диапазоне до 1-2 нс. При микронной и субмикронной технологии производства ИС эти импульсы были неопасны вследствие инерционности элементов. При современной нанотехнологии производства ИС с проектной нормой меньше 0,1 мкм, такие наведенные сравнимы импульсы ложные с полезными импульсными сигналами и могут привести к искажению полезной информации в СБИС. Статистически установлено [5], что на текущем уровне технологии именно мягкие отказы наиболее часто (примерно в 95% случаев) являются причиной выхода из строя космических аппаратов.

Известны [6] методы борьбы с мягкими отказами на различных уровнях организации системы:

а) на уровне проектирования и изготовления ИС;

б) на уровне стандартных элементов, входящих в библиотеку САПР;

в) на уровне функциональной организации СБИС.

Установлено, что текущие средства защиты на уровне изготовления ИС (технология SOI) и на уровне библиотек элементов (путем искусственного снижения быстродействия триггеров или использования DICEячеек) приносят количественный эффект, но не позволяют решить проблему мягких отказов на качественном уровне.

Также для повышения надежности широко применяются способы структурного троирования с мажорированем – triple modular redundancy (TMR) [7], однако они приводят к увеличению структурной сложности схемы и уменьшению ее быстродействия. Рост структурной сложности ведет к увеличению площади, занимаемой схемой на кристалле, что в условиях повышенной радиации повышает вероятность попадания в нее частицы высокой энергии и возникновения мягкого отказа. Это понижает эффективность данного метода защиты [8].

При разработке сложных информационных систем с повышенной устойчивостью к мягким отказам особое распространение получил подход, называемый "доменной организацией" системы [9, 10]. Его суть заключается в декомпозиции системы на

"отказоустойчивые ячейки" – структурные блоки, надежность которых повышается за счет применения различных методов резервирования (в частности, TMR) и организации восстановления таким образом, что отказы в каждом из блоков происходят независимо друг от друга. По функциональному назначению эти блоки разделяют на два типа: автомат с памятью и запоминающее устройство. Для типовой организации отказоустойчивой ячейки опубликован метод оценки надежности [11], из которого следует важный вывод: частота возникновения мягкого отказа восстанавливаемого блока обратно пропорциональна организованному в нем периоду восстановления. Эта зависимость обосновывает актуальность задачи организации блоков обоих типов, обеспечивающей периодическое восстановление их состояния, так как это позволит значительно повысить устойчивость блока к мягким отказам и, следовательно, надежность устройства при воздействии радиации. Данная работа посвящена организации структуры конечного автомата с памятью с периодическим самовосстановлением.

В ходе предварительных исследований автором были получены оценки вероятности возникновения мягкого отказа при попадании частицы высокой энергии в транзистор для различных известных реализаций конечного автомата со структурной избыточностью [12, 6]. Эти оценки могут быть использованы для дальнейшего их сравнения с характеристиками надежности разработанной автором отказоустойчивой структуры конечного автомата с восстановлением на каждом такте и регистрацией отказов, описываемой в данной работе.

II. МОДЕЛЬ И ТРАДИЦИОННАЯ СТРУКТУРА АВТОМАТА МУРА

Традиционно модель абстрактного конечного автомата S представляется следующей математической структурой:

$S = \langle A, B, R, \delta, \lambda, r_0 \rangle$, где

А – множество состояний входа (входной алфавит),

В – множество состояний выхода (выходной алфавит),

R – множество внутренних состояний,

 $r_0 \in R$ – начальное состояние, в которое автомат приводится сигналом начальной установки,

 $\delta: A \times R \rightarrow R - функция переходов,$

 λ : R \rightarrow B – функция выходов.

В проектировании цифровых устройств используются три типа абстрактных конечных автоматов: автоматы Мили, Мура и Медведева [13]. Для всех трех типов автоматов функция переходов имеет одинаковое теоретико-множественное представление: δ:А×R→R.

Типы автоматов различаются представлением функции выхода λ . Для автомата Мили λ : А×R→B, для

автомата Мура: λ : R \rightarrow B, для автомата Медведева: λ : B=R.

Как средства формализованного представления алгоритма эти модели равномощны: для каждого автомата одного типа можно построить эквивалентный автомат другого типа. С точки зрения реализации на электронных схемах типы автоматов имеют различия в двух отношениях: по качеству выходных сигналов; по затратам триггеров на память автомата.

Наилучшее качество выходных сигналов имеет автомат Медведева. В этих автоматах значение выходного сигнала устанавливается сразу после переключения триггера синхронно с фронтом либо спадом тактового импульса и сохраняется в течение всего такта.

В автоматах Мура качество выходных сигналов несколько хуже, так как при их формировании выходные сигналы триггеров подвергаются некоторым функциональным преобразованиям, соответствующим функциям выходов λ . Но следует отметить, что глубина распространения сигналов в соответствующей комбинационной схеме невелика, поэтому переходные процессы распространения сигналов быстро завершаются в начале такта после переключения триггеров.

В автоматах Мили качество выходных сигналов хуже, чем в других типах автоматов. Это связано с тем, что при их формировании глубина распространения сигналов по сети элементов наиболее велика. Она включает процессы во внешней схеме, формирующей входные сигналы автомата, переключение триггеров и процессы в схеме, реализующей функции выходов λ. Эта схема имеет существенно большую сложность и глубину распространения сигналов в сравнении с автоматом Мура. У функций выхода автомата Мили больше число аргументов, как это видно из приведенного выше теоретико-множественного представления функций λ для разных типов автоматов. В связи с этим выходные сигналы в автомате Мили устанавливаются только в конце такта.

По затратам триггеров минимальное число имеет автомат Мили, максимальное – автомат Медведева. Таким образом, с точки зрения рассмотренных показателей автомат Мура дает компромиссное решение. Следует отметить, что автоматы Мура чаще всего и применяются при проектировании цифровой аппаратуры.

Традиционная структурная схема автомата Мура представлена на рис. 1.

Комбинационная схема КС1 реализует функцию переходов δ , КС2 реализует функцию выходов λ . Для реализации блока памяти (П) автомата использовано *s* триггеров (ТТ) типа D, синхронизируемых спадом тактового сигнала *C*. Входы триггеров: R – сигнал сброса, D – входные данные, C – вход синхронизации. НУ – сигнал начальной установки. Связи между блоками соответствуют функциям в автомате Мура: δ :

 $\{X\} \times \{Q\} \rightarrow \{Q\}; \lambda: \{Q\} \rightarrow \{Y\},$ где разрядность вектора **X** равна *m*, а разрядность вектора **Y** равна *n*. τ_1 , τ_2 , τ обозначают задержки в электронных схемах, реализующих блоки KC1, KC2 и II, соответственно.



Рис 1. Структурная схема автомата Мура

Для данной структуры в [6] автором получена формула, позволяющая вероятность отсутствия мягких отказов $\overline{P}_{\text{м.o.a}}$ в автомате в течение некоторого времени выполнения задачи T_3 при известной частоте $q_{n.ч.т}$ попадания заряженных частиц в один из транзисторов конечного автомата:

$$\overline{P}_{\text{M.O.a}} = e^{-(q_{\text{M.O.II}} + q_{\text{M.O.KC1}})T_3},$$

где qмол – частота возникновения мягкого отказа автомата по причине попадания заряженной частицы в область памяти; qможе – частота возникновения мягкого отказа автомата по причине попадания заряженной частицы в область КС1.

При оценке вероятности возникновения мягкого отказа необходимо учитывать, что ложный импульс, распространяющийся с выходов КС1, влечет за собой изменение состояния П (мягкий отказ) только в том случае, если он совпадет по времени с моментом записи данных (спадом синхроимпульса С) в один из триггеров П. Этот интервал времени занимает периода незначительную долю синхронизации автомата. Таким образом, с точки зрения надежности КС1 имеет меньшую структурную значимость, нежели П, где ложное изменение состояния любого из триггеров непосредственно приводит к мягкому отказу. Однако КС1 может содержать большее по сравнению с П число логических элементов, что увеличивает вероятность попадания заряженных частиц в ее область и приводит к необходимости разработки механизма борьбы с ложными импульсами на выходах КС1.

III. СТРУКТУРА АВТОМАТА МУРА С ТРОИРОВАНИЕМ, МАЖОРИРОВАНИЕМ И САМОВОССТАНОВЛЕНИЕМ

Применительно к рассматриваемой задаче построения автомата с повышенной устойчивостью к мягким отказам автором было определено расширение функций автомата дополнительно к основной функции реализации алгоритма:

a) блокирование прохождения мягкого отказа на выход автомата;

 б) восстановление состояния отказавшего экземпляра автомата без прерывания выполнения основной функции;

в) выявление, регистрация и подсчет числа мягких отказов в автомате.

Перечисленные расширения реализованы разработанной автором и защищенной патентом [14] структурной схеме автомата Мура с троированием, мажорированием самовосстановлением, И приведенной на рис. 2. Принцип реализации этих расширений применим для последовательностных схем, соответствующих не только автомату Мура, но также автоматам Мили и Медведева. Отличия в обеспечении защиты от мягких отказов для других типов автомата несущественны, и предложенные способы защиты могут быть в них использованы очевидным образом.



Рис. 2. Структурная схема автомата Мура с троированием, мажорированием и самовосстановлением

Рассмотрим состав схемы и ее отличительные особенности в сравнении с традиционной (рис. 1) структурой автомата Мура без резервирования.

Входные сигналы, аналогичные рис.1:

Х – m-разрядный вектор входных информационных сигналов;

С – тактовый импульс синхронизации;

R – сигнал сброса (начальной установки).

Дополнительные входные сигналы:

ENA (enable) – сигнал разрешения работы автомата: при ENA=1 работа разрешена; при ENA=0 автомат остановлен, при переходе сигнала ENA $0 \rightarrow 1$ автомат продолжит работу с того состояния, в котором был остановлен;

R_{БРО} – сигнал сброса блока регистрации ошибок БРО. В восстанавливаемых системах периодически проводится мониторинг состояния блоков (проверка количества отказов, зарегистрированных БРО), в конце каждого цикла мониторинга производится сброс состояния БРО на начальное. Выходные сигналы автомата:

 Y – п-разрядный вектор информационных выходных сигналов (аналогично сигналам на рис.1);

О – код числа мягких отказов в автомате за период мониторинга.

Перечень блоков структуры:

КС1 – комбинационная схема, реализующая функцию переходов δ, аналогичная рис.1;

КС2 – комбинационная схема, реализующая функцию выходов λ, аналогичная рис.1;

П – блок памяти автомата, содержащий 3 экземпляра П1, П2, П3 (каждый аналогичен блоку памяти П на рис.1);

М1 – блок мажорирования троек соответствующих выходных сигналов блоков П1, П2, П3;

MUX – мультиплексор, переключающий на информационные входы блоков памяти (D) сигнал Φ_{π} перехода с выходов комбинационной схемы KC1, либо сигнал Q с выходов блока M1;

БРО – блок регистрации ошибок (мягких отказов в автомате);

СТ1, СТ2, СТ3 – три экземпляра синхронных счетчиков для подсчета числа мягких отказов;

M2 – блок мажорирования сигналов с выходов счетчиков и формирования сигнала О – числа мягких отказов за период мониторинга;

 $\tau_1, \tau_2, \tau_3, \tau_4$ – элементы задержки.

IV. ПРИНЦИП РАБОТЫ АВТОМАТА МУРА С ТРОИРОВАНИЕМ, МАЖОРИРОВАНИЕМ И САМОВОССТАНОВЛЕНИЕМ

Раскроем функции новых по сравнению с традиционной структурой автомата Мура (рис. 1) блоков.

Отличие данного структурного решения от TMR заключается в применении троирования не устройства памяти, а памяти как части конечного автомата. Такие автоматы в вычислительных системах играют роль управляющих блоков и операционных блоков. Особенность троирования состоит в том, что троируется не весь автомат, а только память состояний и предусмотрено периодическое (каждый такт) восстановление информации в ней, а влияние мягких отказов в комбинационной схеме КС1 устраняется другим оригинальным способом, описанным ниже.

Блок мажорирования М1 содержит s (s – число триггеров в памяти автомата) мажоритарных элементов. Возможная схема, реализующая мажоритарный элемент, представлена на рис. 3.

Помимо этого блок M1 содержит схему, реализующую функцию E(Q1, Q1, Q3) выявления мягкого отказа, возвращающую логическую 1 в том случае, если сигналы на x₁, x₂, x₃ на входах М1 не одинаковы.



Рис. 3. Структурная схема мажоритарного элемента

Сигнал Е поступает на вход блока регистрации ошибок БРО. В БРО для подсчета числа мягких отказов (ошибок) используется синхронный счетчик, например, соответствующий стандартному счетчику К1533ИЕ10. Счетчик имеет информационные входы записи (D₀D₁D₂D₃), вход РЕ разрешения записи (при PE=0), входы CE разрешения прибавления единицы (при CE=1), вход синхронизации С и вход сброса R. Поскольку в самом счетчике под воздействием радиации могут возникать мягкие отказы, то в блоке БРО применяется троирование счетчика мажорирование. Блок мажорирования M2 содержит 4 соответственно мажоритарных элемента числу разрядов в счетчике. Выходы блока мажорирования подключены к внешнему выходу О автомата и к информационным входам (D₀D₁D₂D₃) всех 3-x счетчиков. Управляющие входы РЕ и СЕ всех трех счетчиков соединены и подключены к выходу Е блока мажорирования M1. Таким образом, при E=1 в счетчиках по спаду С прибавляется 1, при Е=0, по спаду С записывается код с выхода М2, и каждый такт в счетчиках БРО происходит самовосстановление информации.

При работе автомата в режиме реализации алгоритма каждый такт выполняется переход в новое состояние. В случае возникновения отказа в одном из экземпляров памяти он не проявляется на выходе Q мажоритара, и на информационные входы элементов П каждый такт поступает верная информация. Таким образом, каждый такт осуществляется самовосстановление данных в П.

В некоторых случаях в системах организуется работа автомата в стартстопном режиме. Для этого используется сигнал ENA (enable), вырабатываемый управляющим блоком системы. Режим временной приостановки работы автомата может быть достаточно длительным. За это время в памяти автомата также могут возникать мягкие отказы. С целью организации самовосстановления памяти автомата в этом режиме выходные сигналы Q с выхода блока M1 поступают не только на входы комбинационной схемы КС1, но и на вход мультиплексора MUX. При этом, если ENA=0,

каждый такт происходит самовосстановление памяти П.

В работе [8] аргументирована необходимость борьбы с ложными импульсами на выходе КС1, так как они являются источниками мягких отказов в памяти состояний. Также доказана низкая эффективность мажорирования на уровне КС1, поскольку в этом случае источником ложных импульсов, искажающим данные памяти состояний, становится мажоритарный элемент. Поэтому для уменьшения влияния ложных сигналов на выходах КС1, подключенных к информационным входам триггеров памяти автомата, предложен следующий способ. Как отмечено в [8], ложный импульс, поступающий на информационный вход триггера, может изменить его состояние, если этот импульс попадает в интервал т=t_{SU}+t_H, где t_{SU} - время предустановки триггера, t_н - время удержания триггера, на практике составляющий малую долю от периода синхронизации T_C. С учетом этого в троированной схеме памяти автомата П (рис. 2) в цепь передачи синхроимпульсов С введены элементы задержки: т₁=т для экземпляра памяти П2 и т₂=2т для экземпляра ПЗ. Благодаря этому, переключение всех триггеров в П2 происходит по сравнению с П1 с задержкой т, а в П3 – с задержкой 2т.

Положительное влияние введенных задержек на надежность автомата проиллюстрировано на рис 4.



Рис. 4. Принцип уменьшения влияния ложных сигналов на выходах КС1

На рис. 4 использованы следующие обозначения:

D – сигнал на информационных входах D триггеров блока памяти П (рис. 2);

С1, С2, С3 – входы синхронизации С экземпляров блока памяти П1, П2, П3 соответственно (рис.2);

ТС – период синхронизации триггеров.

На входы D поступает кратковременный ложный импульс длительностью t_{л.и.} Для каждого экземпляра

блока памяти (П1, П2, П3) существует временной интервал (1, 2 и 3 соответственно) длительностью t_{SU}+t_H, в течение которого сигнал на входе D влияет на состояние триггеров данного экземпляра. Эти интервалы между собой не пересекаются из-за внесенных задержек т1, т2. Таким образом, ложный импульс, попадающий только в интервал 1, вызывает мягкий отказ в П1, но не оказывает влияния на П2 и ПЗ. А поскольку выходы троированного блока памяти П подключены к мажоритару М1, мягкий отказ в одном из экземпляров блока памяти не вызывает искажения выходного вектора Q. Таким образом, распространение мягкого отказа заблокировано.

Искажение состояния троированной памяти может произойти только в том случае, если в течение одного такта ложный сигнал на входах D захватит минимум два интервала длительностью $t_{SU}+t_{H}$, отмеченных выше. Назовем это сложное событие возникновением неисправленного отказа в памяти из-за ложных сигналов на входе D. Зависимость вероятности $P_{H.o.D}$ этого события от вероятности $P_{л.c.D}$ появления ложного сигнала на информационном входе D как:

$$P_{\mathrm{H.o.}D} = (P_{\mathrm{n.c.}D} \frac{\tau}{T_C})^2,$$

Где P_{HOD} собой выражение представляет появления вероятность ложного сигнала на информационном входе D в течение одного такта и попадания его в интервал т при спаде импульса С. Очевидно, что на практике значение P_{н.o.D} << 1, что практически исключает воздействие ложных импульсов на входе D на работу автомата.

Аналогичный способ введения задержек ($\tau_3=\tau$, $\tau_4=2\tau$) в цепь передачи синхроимпульсов используется и в блоке БРО.

V. Анализ надежности автомата Мура с троированием, мажорированием и самовосстановлением

В работе [6] автором оценивалась вероятность сохранения конечным автоматом работоспособности в течение времени Т₃ решения алгоритмической задачи. Для аналогичной функциональной спецификации конечного автомата проведем оценку надежности разработанной структуры автомата Мура с троированием, мажорированием и самовосстановлением.

В ходе анализа будем считать, что все отказы в элементах автомата являются мягкими (восстанавливаемыми), возникающими по причине попадания заряженных частиц в транзисторы автомата с некоторой известной интенсивностью **q**_{п.ч.т.} Работоспособность автомата считается утраченной, если на выход автомата У (рис. 2) поступают данные. При оценке не будем искаженные рассматривать блок БРО (рис. 2), так как он реализует

дополнительную функцию, не связанную с решением основной задачи автоматом.

Проанализируем, в результате чего может исказиться выходной сигнал автомата. Основной причиной являются мягкие отказы в элементах памяти, которые приведут к потере работоспособности, если за один такт работы автомата в одинаковых битах двух различных экземпляров памяти П произойдет мягкий отказ. В противном случае побитный мажоритар М1 по цепи обратной связи передаст корректные данные на входы П1, П2, П3, и на следующем такте состояние памяти будет автоматически восстановлено. Для получения оценки вероятности отказа в памяти ограничимся рассмотрением ситуаций, когда за такт работы автоматы возникает от нуля до трех отказов (остальными случаями пренебрежем, так как вероятность их возникновения на несколько порядков меньше). Тогда, обозначив событие мягкого отказа бита памяти за Апі, (где і – номер экземпляра памяти, ј порядковый номер бита памяти в экземпляре), рассмотрим возможные комбинации событий, которые приведут к отказу блока памяти.

В условиях текущей задачи каждый блок памяти содержит 3 информационных бита (s=3, что определяется функциональной спецификацией). В случае отсутствия искаженных бит памяти, либо при наличии только одного искаженного бита, отказа рассматриваемой структуры не происходит. При наличии двух искаженных бит к отказу приведут следующие комбинации: $A_{n1,1}A_{n2,1}$, $A_{n1,1}A_{n3,1}$, $A_{n2,1}A_{n3,1}$, $A_{n1,2}A_{n2,2}$, $A_{n1,2}A_{n3,2}$, $A_{n2,2}A_{n3,2}$, $A_{n1,3}A_{n2,3}$, $A_{n2,3}A_{n3,3}$, $A_{n1,3}A_{n3,3}$. Итого 9 комбинаций для случая двух искаженных бит к отказу приводят 57 возможных комбинаций событий.

Общую вероятность возникновения отказа в памяти автомата оценим как сумму вероятностей возникновения рассмотренных несовместных комбинаций событий одновременного отказа двух или трех бит в блоке памяти.

$$\begin{split} P_{\rm m.o.a} &= 9(P_{\rm m.o.\,Ghta})^2 (1-P_{\rm m.o.\,Ghta})^7 + 57(P_{\rm m.o.\,Ghta})^3 (1-P_{\rm m.o.\,Ghta})^6. \end{split}$$

Определим зависимость вероятности искажения бита данных в памяти Р_{м.о.бита} в (1) от интенсивности q_{п.ч.т} попадания заряженной частицы в транзистор автомата. Причин, вследствие которых может возникнуть искажение, две. Первая - попадание заряженной частицы непосредственно в область памяти. Так как один бит памяти реализуется триггером, состоящим ИЗ 5 транзисторов, интенсивность возникновения этого события равна Вторая причина искаженной 5q_{п.ч.т}. _ запись информации вследствие попадания заряженной частицы в элементы КС1 или MUX, подключенные к соответствующему биту памяти. Исходя из расчетов, произведенных в [6], интенсивность появления ложных импульсов на выходах КС1 равна 3,4qп.ч.т, а на выходах мультиплексора – 1,25q_{п.ч.т.} Для оценки

интенсивности появления пожных импульсов оказывающих влияние на П, эти величины необходимо умножить на коэффициент K=0,15, так как элементы КС1 и MUX влияют на работу памяти только в момент спада синхроимпульса (обычно занимающего не больше 15% от длительности такта). Суммарная интенсивность искажений одного бита данных соответственно равна 5qп.ч.т+0,15(3,4qп.ч.т+1,25qп.ч.т)=5,7qп.ч.т. Исходя из этого, вероятность искажения бита памяти Р_{м.о.бита} в течение одного такта синхронизации ТС автомата выражается через интенсивность попадания заряженной частицы в транзистор q_{п.ч.т} следующим образом:

$$P_{\rm m.o.6uta} = 1 - e^{-5.7q_{\rm m.u.t}T_{\rm c}}.$$
 (2)

Подставив выражение (2) в (1) получим вероятность отказа всей структуры в течение одного такта работы автомата.

Поскольку в начале каждого такта происходит восстановление состояния системы, вероятность безотказной работы автомата в течение п последовательных тактов вычисляется как произведение вероятностей безотказной работы в течение каждого такта:

$$\overline{P_{\text{M.O.a}}}(n) = (1 - P_{\text{M.O.a}})^n.$$

Используя полученные выражения, построим графики функции вероятности безотказной работы на интервале T_3 (n=100 тактов) для исследованной структуры (структура 4) и сравним его с графиками, построенными для трех известных отказоустойчивых структур, проанализированных автором в [8]:

Структура 1 – автомат без структурного резервирования (рис. 1);

Структура 2 – автомат с троированными блоками и троированными входными мажоритарами без периодического восстановления информации;

Структура 3 – автомат с троированными блоками и троированными входными мажоритарами и периодическим восстановлением информации. Восстановление искаженного состояние происходит за счет формирования сигнала начальной установки в конце каждого цикла работы автомата (период восстановления соответствует длительности цикла алгоритма работы автомата).

Графики функций работоспособности проанализированных структур приведены на рис. 5.

Ось абсцисс обозначает текущее время t решения задачи, измеряемое в количестве тактов работы автомата. По оси Y расположена вероятность нахождения автомата в работоспособном состоянии (1 – гарантированно работоспособен, 0 – гарантированно неработоспособен).



Рис. 5. Функции работоспособности анализируемых структур

Итоговая вероятность успешного решения задачи предложенной структуры 4 равна 0.99. лля Аналогичная величина, рассчитанная для структуры 3 [8] лучшие (показавшей в характеристики надежности), равна 0,84 что наглядно демонстрирует преимущество предложенной структуры при работе в условиях мягких отказов. Это превосходство обеспечивается меньшим по сравнению со структурой 3 периодом восстановления - оно происходит на каждом такте, а не только по окончании цикла работы алгоритма.

Заключение

Определены расширения функциональности традиционной структуры автомата Мура, позволяющие повысить надежность устройства в случае функционирования при потоке мягких отказов.

Разработана структура автомата Мура с троированием памяти, мажорированием выходных сигналов памяти и восстановлением информации в каждом такте. Данная структура обладает повышенной защитой от ложных импульсов, возникающих как причине попадания частиц высокой энергии в области комбинационных схем, так памяти состояний автомата. Также она оснащена средствами регистрации количества мягких отказов, произошедших в ходе работы автомата.

Оценки надежности разработанной структурой демонстрируют превосходство разработанной структуры над другими известными отказоустойчивыми структурами автомата Мура с точки зрения устойчивости к возникновению мягких отказов.

ЛИТЕРАТУРА

[1] Егоров И.В., Мелехин В.Ф. Анализ проблемы повышения радиационной стойкости информационноуправляющих систем на этапе функциональнологического проектирования // Информационноуправляющие системы. 2016. № 1(80). С. 26–31.

- [2] Edmonds D.L., Barnes C.E., Scheick L.Z. An introduction to space radiation effects on microelectronics. Pasadena, USA: NASA, Jet propulsion laboratory, California institute of technology, 2000.
- [3] Schwank J.R., Shaneyfelt M.R., Dodd P.E. Radiation hardness assurance testing of microelectronic devices and integrated circuits: Test guideline for proton and heavy ion single-event effects // IEEE Transactions on Nuclear Science. 2013. Vol. 60, № 3. P. 2101–2118.
- [4] Amusan O.A. et al. Single event upsets in deepsubmicrometer technologies due to charge sharing // IEEE Transactions on Device and Materials Reliability. 2008. Vol. 8, № 3. P. 582–589.
- [5] Koons H.C. et. al The impact of the space environment on space systems // 6th Spacecraft Charging Technology. 1998. P. 7–11.
- [6] Максименко С.Л., Мелехин В.Ф., Филиппов А.С. Анализ проблемы построения радиационно-стойких информационно-управляющих систем // Информационно-управляющие системы. 2012. № 2(57). С. 18–25.
- [7] Oliveira R., Jagirdar A., Chakraborty T.J. A TMR Scheme for SEU Mitigation in Scan Flip-Flops. IEEE, 2007. P. 905– 910.
- [8] Егоров И.В., Мелехин В.Ф. Анализ показателей надежности и сложности реализации различных вариантов структур автомата с памятью при потоке мягких отказов // Информационно-управляющие системы. 2017. № 3(88). С. 34–46.
- [9] Глухих М.И. Разработка методов синтеза информационно-управляющих систем специального назначения со структурным резервирование: дис. ... канд. техн. наук. СПб, 2006.
- [10] Abraham J.A., Siewiorek D.P. An algorithm for the accurate reliability evaluation of triple modular redundancy networks // IEEE Transactions on Computers. 1974. Vol. C-23(7). P. 682–692.
- [11] Максименко С.Л., Мелехин В.Ф. Анализ надежности функциональных узлов цифровых СБИС со структурным резервированием и периодическим восстановлением работоспособного состояния // Информационно-Управляющие Системы. 2013. № 2(63). С. 18–23.
- [12] Егоров И.В., Мелехин В.Ф. Анализ процессов в конечном автомате при воздействии радиации. Оценка вероятности искажения информации // Информационно-управляющие системы. 2016. № 3 (82). С. 24–33.
- [13] Kaeslin H. Digital integrated circuit design. from vlsi architectures to cmos fabrication / H. Kaeslin, Cambridge University Press, 2008. 879 p.
- [14] Пат. 174640 RU, МПК G06F 11/07 (2006.01). Отказоустойчивый цифровой преобразователь информации для управления дискретными процессами / И. В. Егоров (RU), В. Ф. Мелехин (RU). – № 174640/25– 08; заявл. 14.06.2017; опубл. 24.10.2017, Бюл. № 30. – 7 с.

A Method for Organizing the Soft Error Tolerant Moore Finite State Machine

I.V. Egorov

Peter the Great St. Petersburg Polytechnic University, Saint-Petersburg, ig-ego@mail.ru

Abstract — up-to-date design rules used in computer engineering make hardware unreliable when working under radiation. A hit of a charged particle causes a "soft failure" a situation, when hardware elements remain in usable condition but the information transmitted or stored in memory is corrupted. This problem appeared after the introduction of nanotechnology in production of integrated circuits. With old submicron technologies, soft failures occurred much less frequently, because false impulses caused by charged particles were invisible due to the inertia of the logic elements. Now we need to develop new circuitry solutions, which would increase the resistance of hardware (especially of the finite states machines) to soft failures. Known research revealed, that soft failures occur more often in memory units, than in combinational circuits, and can be eliminated by the periodic recovery of the memory state. In spite of this, the most of known reliable structures of finite state machines are based on structural redundancy, which is not enough efficient in case of soft failures, and don't use self-recovery for memory units. Purpose: suggest new technical solutions to increase the reliability of a Moore automaton working in case of periodic soft failures. Results: the developed structure of Moore automaton with triple redundancy of internal memory and output signals and with self-recovery on each synchronization clock period has increased resistance to the soft errors occurred both in combinational circuits and internal memory. This structure also contains the tools to register a count of soft failures occurred during exploitation, which allow to control the state of the automaton and detect the dangerous state, when soft failures occur too often. The reliability characteristic of the developed structure has been estimated. The reliability analysis has revealed the advantage of the developed structure against known fault-tolerant structures of the Moore automaton - in case of periodic soft failures it's operating time to failure is greater by an order of magnitude.

Keywords — finite state machine, combinational circuit, reliability analysis, synchronization, soft failures, structural redundancy, recoverable systems, probability of non-failure.

REFERENCES

- Egorov I.V., Melekhin V.F. Analiz problemy povyshenija radiacionnoj stojkosti informacionno-upravljajushhih sistem na jetape funkcional'no-logicheskogo proektirovanija (Analysis of Radiation Resistance Improvement Issue for Information and Control Systems at the Stage of Functional and Logical Design) // Informacionno-upravljajushhie sistemy. 2016. № 1(80). S. 26–31.
- [2] Edmonds D.L., Barnes C.E., Scheick L.Z. An introduction to space radiation effects on microelectronics. Pasadena, USA: NASA, Jet propulsion laboratory, California institute of technology, 2000.
- [3] Schwank J.R., Shaneyfelt M.R., Dodd P.E. Radiation hardness assurance testing of microelectronic devices and

integrated circuits: Test guideline for proton and heavy ion single-event effects // IEEE Transactions on Nuclear Science. 2013. Vol. 60, № 3. P. 2101–2118.

- [4] Amusan O.A. et al. Single event upsets in deepsubmicrometer technologies due to charge sharing // IEEE Transactions on Device and Materials Reliability. 2008. Vol. 8, № 3. P. 582–589.
- [5] Koons H.C. et. al The impact of the space environment on space systems // 6th Spacecraft Charging Technology. 1998. P. 7–11.
- [6] Maksimenko S.L., Melehin V.F., Filippov A.S. Analiz problemy postroenija radiacionno-stojkih informacionnoupravljajushih sistem (Analysis of the Problem of Radiation-Tolerant Information and Control-Systems Implementation) // Informacionno-upravljajushie sistemy. 2012. № 2(57). S. 18–25.
- [7] Oliveira R., Jagirdar A., Chakraborty T.J. A TMR Scheme for SEU Mitigation in Scan Flip-Flops. IEEE, 2007. P. 905– 910.
- [8] Egorov I.V., Melekhin V.F. Analiz pokazatelej nadezhnosti i slozhnosti realizacii razlichnyh variantov struktur avtomata s pamjat'ju pri potoke mjagkih otkazov (Analysis of Reliability and Complexity Characteristics for Various Structures of a Finite State Machine Working in Case of Soft-Failure Flow) // Informacionno-upravljajushhie sistemy. 2017. № 3(88). S. 34–46.
- [9] Glukhih M.I. Razrabotka metodov sinteza informacionnoupravljajushhih sistem special'nogo naznachenija so strukturnym rezervirovanie: dis. ... kand. tehn. nauk. (Development of methods of synthesis of information and control systems of a special purpose with structural redundancy: PhD thesis) Saint-Petersburg, 2006.
- [10] Abraham J.A., Siewiorek D.P. An algorithm for the accurate reliability evaluation of triple modular redundancy networks // IEEE Transactions on Computers. 1974. Vol. C–23(7). P. 682–692.
- [11] Maksimenko S.L., Melehin V.F. Analiz nadezhnosti funkcional'nyh uzlov cifrovyh SBIS so strukturnym rezervirovaniem i periodicheskim vosstanovleniem rabotosposobnogo sostojanija (Analysis of Reliability of Functional Nodes of Digital VLSI Circuits with Structural Redundancy and Periodic Operational State Recovery) // Informacionno-Upravljajushhie Sistemy. 2013. № 2(63). S. 18–23.
- [12] Egorov I.V., Melekhin V.F. Analiz processov v konechnom avtomate pri vozdejstvii radiacii. Ocenka verojatnosti iskazhenija informacii (Analysis of Processes in a Finite State Machine under Radiation. Probabilistic Assessment of Information Distortion) // Informacionno-upravljajushhie sistemy. 2016. № 3 (82). S. 24–33.
- [13] Kaeslin H. Digital integrated circuit design. from vlsi architectures to cmos fabrication / H. Kaeslin, Cambridge University Press, 2008. 879 p.
- [14] Pat. 174640 RU, MPK G06F 11/07 (2006.01). Otkazoustojchivyj cifrovoj preobrazovatel' informacii dlja upravlenija diskretnymi processami (The failure-safe information digitizer for management of discrete processes) / I. V. Egorov (RU), V. F. Melehin (RU). № 174640/25–08; zajavl. 14.06.2017; opubl. 24.10.2017, Bjul. № 30. 7 s.

Детектор свободных участков радиочастотного спектра

В.А. Жмылев

МИЭТ, г. Москва, vz-75@yandex.ru

Аннотация — В статье рассматриваются программноаппаратные средства, способные определять свободные участки радиочастотного спектра. При помощи этих средств можно устанавливать сеансы связи на временно свободных участках радиочастотного спектра. Представлены требования, предъявляемые к детектору. Предложены различные варианты аппаратных средств. Рассмотрены способы анализа, применяемые на различных аппаратных средствах. Подробно описан алгоритм формирования гистограммы энергии, определения порога принятия решения.

Ключевые слова — программно-конфигурируемое радио, когнитивное радио, вторичные сети связи, концепция вторичных пользователей, детектор энергии, анализатор эфира.

I. Введение

В настоящее время весь радиочастотный спектр распределён между различными пользователями. По данным международной статистики относительная занятость участков радиочастотного спектра даже в пиковые часы не превышает 16% [4]. Поскольку все работы распределены, новых частоты для пользователей существует проблема доступа к свободным участкам радиочастотного спектра. использования Эффективность радиочастотного спектра можно повысить за счет использования временно свободных участков спектра, что позволяет в том числе и повысить помехоустойчивость, например, ППРЧ работать в режиме (псевдослучайная перестройка радиочастоты), оперативно изменять рабочую полосу частот, чтобы повысить скорость передачи данных и т.п. Важно, чтобы при работе на вторичной основе не нарушалась работа первичных пользователей. Для организации вторичных сетей необходим детектор свободных каналов, связи позволяющий с высокой степенью достоверности определять свободные участки радиочастотного спектра.

Развитие цифровых технологий привело к идеологии систем связи основанной на илее Software программно-конфигурируемого радио – Defined Radio (SDR). Системы связи программнорадио конфигурируемого получили широкое распространение во многих областях техники. Зачастую, на одной аппаратной платформе приходится совмещать различные стандарты связи, работать в различных режимах, модернизировать протоколы и т.д. В аппаратуру систем связи программноконфигурируемого радио закладывается элементная

база с запасом производительности (вычислительных ресурсов). Конфигурация таких систем связи осуществляется, в основном, путем обновления программного обеспечения. Программными управляются средствами такие параметры радиостанция как: вид модуляции, скорость передачи информации, способы кодирования, режим работы и т.д.

В настоящее время параметры радиоканала (ширина и центральная частота канала) строго закреплены за пользователем, и пользователь может работать только в пределах выделенных участков радиочастотного спектра. Управление параметрами радиоканала, а именно установка сеансов связи на временно свободных участках спектра позволяет повысить эффективность использования спектра, помехоустойчивость повысить связи при необходимости увеличить скорость передачи данных и Поэтому реализация детектора свободных тп участков радиочастотного спектра представляется актуальной задачей.

II. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Технические требования, предъявляемые к детектору, продиктованы постановкой задачи – определять свободные участки радиочастотного спектра.

Диапазон частот, в котором осуществляется анализ, и ширину канала задает пользователь. Это требование объясняется тем, что свободные участки спектра определяются для нужд пользователя, и в зависимости от своих нужд пользователь устанавливает требуемую ширину канала и диапазон частот, в котором желает установить сеанс связи.

Детектор определяет свободные участки радиочастотного спектра независимо от типа сигналов. В радиочастотном спектре присутствуют не только сигналы различных радиостанций, но и стационарные помехи, например исходящие от производственных станков. Наличие в канале стационарной помехи или сигнала сторонней радиостанции должно приводить к решению о занятости канала, поскольку такой канал не пригоден для установки сеанса связи.

Время анализа сравнимо с временем стационарности радиоканала, поскольку состояние канала за время измерения не должно меняться.

Детектор является детектором энергии. Энергия является общим параметром для любых сигналов. Поскольку параметры обнаруживаемых сигналов заранее неизвестны, то сигналы детектируются по средней энергии.

Обнаружение сигналов осуществляется на основе критерия Неймана-Пирсона. Вычисление порогового значения происходит по измеренным значениям средних энергий в каждом канале и по заданной (допустимой) вероятности ложной тревоги.

III. АППАРАТНЫЕ СРЕДСТВА

Аппаратные средства необходимые для реализации детектора свободных участков радиочастотного спектра представляют собой собственный приемник радиостанции программно-конфигурируемого радио или специализированный приемник.

Реализовать детектор свободных участков радиочастотного спектра можно на аппаратной базе собственного приемника радиостанции программно-конфигурируемого радио путем обновления программного обеспечения.

Специализированный приемник представляет собой отдельное устройство. В процессе разработки, можно заложить широкую полосу ВЧ (высокочастотного) тракта, что способствует меньшему времени анализа.

Функциональная схема приемника программноконфигурируемого радио представлена на рис. 1.



Рис. 1. Функциональная схема приемника

MIIIV (малошумящий усилитель) усиливает входной высокочастотный сигнал, квадратурный демодулятор преобразовывает высокочастотный сигнал в baseband-диапазон и выделяет I/Q, АЦП (аналого-цифровой преобразователь) оцифровывает baseband-сигналы. После этого сигнал поступает в устройство цифровой обработки сигналов. Синтезатор частоты генерирует высокочастотный сигнал, с частотой ω_r .

Функция аппаратных средств заключается в преобразовании высокочастотного участка спектра в baseband-диапазон и представлении baseband-сигналов в цифровом виде для измерения значений средних энергий в каждом канале анализируемого диапазона частот.

IV. Способы анализа

Существует два способа анализа – последовательный и параллельный. При

последовательном анализе (рис. 2), что характерно для собственного приемника радиостанции, средняя энергия каждого канала вычисляется последовательно. Частота гетеродина устанавливается в середину первого канала с полосой W_{ch2} . Высокочастотный участок спектра с помощью демодулятора переносится на нулевую ПЧ (промежуточная частота). Измеряется Ν временных отсчетов, которые ограничиваются фильтром основной селекции с шириной полосы равной ширине канала W_{ch2} . После этого временные отсчеты усредняются. Полученное значение есть средняя энергия первого канала из анализируемой полосы частот BW. Для вычисления средней энергии следующего канала из анализируемой полосы частот выполняется оперативная перестройка частоты гетеродина на величину ширины канала W_{ch2} И аналогичным образом вычисляется средняя энергия следующего канала из анализируемой полосы частот BW.



Рис. 2. Последовательный способ анализа

В результате анализа имеются значения средней энергии для каждого канала из анализируемой полосы частот *BW*. Важно отметить, что время измерения средней энергии одного канала не должно превышать время стационарности радиоканала.



Рис. 3. Параллельный способ анализа

При параллельном анализе (рис. 3) частота гетеродина устанавливается в середину всего анализируемого диапазона частот *BW*. С помощью демодулятора весь спектр частот *BW* переносится на нулевую ПЧ. Измеряется k групп по N отсчетов на канал, где k – количество каналов. По сигналу, состоящему из k*N временных отсчетов, выполняется БПФ (быстрое преобразование Фурье). Частотные отсчеты, попавшие в один и тот же канал W_{ch2} , линейно усредняются. Полученные значения представляют собой среднюю энергию в каждом канале анализируемого диапазона частот.

В результате анализа, как и в случае последовательного анализа, имеются значений средней энергии для каждого канала из анализируемой полосы частот *BW*. В случае параллельного анализа время измерения средних энергий всех каналов не должно превышать время стационарности радиоканала.

При использовании аппаратной базы собственного приемника радиостанции обычно можно использовать только последовательный способ анализа, который занимает больше времени по сравнению c параллельным анализом. Параллельный способ анализа можно применить в специализированном приемнике. Отличительной особенностью такого приемника является широкая полоса ВЧ тракта, поскольку преобразуется сразу весь анализируемый диапазон частот. При широкой полосе ВЧ тракта потребуются широкополосные элементы, высокоскоростные АЦП и т.п. Аппаратные затраты повлияют на характеристики приемника, например на параметр энергопотребления, при этом характеристики обнаружения повысятся, в частности увеличится скорость анализа.

V. АЛГОРИТМ РАБОТЫ ДЕТЕКТОРА

Детектор предназначен для принятия решения о состоянии канала – свободно или занято. Функциональная схема алгоритма работы детектора представлена на рис. 4.



Рис. 4. Функциональная схема детектора

Исходными данными для детектора служат baseband-сигналы I/Q квадратур. По I/Q сигналам, полученным с помощью приемника, вычисляется средняя энергия в каждом канале анализируемой полосы частот.

При последовательном анализе средняя энергия в каждом канале определяется в полосе частот, ограниченной фильтром основной селекции. При параллельном способе анализа частотные отсчеты, полученные после преобразовании Фурье, попавшие в один канал, линейно усредняются, и полученное значение представляет собой среднюю энергию в канале. Величина средней энергии в канале, и при параллельном и при последовательном способе анализа, определяется нормированной суммой квадратурных компонент демодулированного сигнала (1)

$$E_{ch} = \frac{1}{N} \sum_{i=0}^{N-1} \left(l_i^2 + Q_i^2 \right)$$
(1)

где E_{ch} – средняя энергия в канале, N – количество отсчетов на канал, I/Q – квадратуры демодулированного сигнала.

После вычисления средней энергии в каждом формируется гистограмма канале энергии. Гистограмма средней энергии является основой для статистически достоверного определения свободных участков спектра, поскольку не зависит от типа сигналов и шумов. Гистограмма отражает текущую плотность распределения вероятности средней энергии в месте проведения анализа в данный момент времени. По полученной гистограмме энергии и по заданному значению вероятности ложной тревоги определяется пороговое значение. Вероятность ложной тревоги в случае обнаружения сигналов есть вероятность того, что канал, в котором нет сигнала, определяется как занятый. Далее принимается решение о состоянии каждого канала путем сравнения средней энергии в каждом канале и порогового значения. В случае, когда средняя энергия превышает пороговое значение, канал считается занятым. Когда средняя энергия не превышает пороговое значение, канал считается свободным.

VI. ГИСТОГРАММА ЭНЕРГИИ

Рассмотрим процесс формирования гистограммы энергии. Измеренные значения средней энергии E_{ch} во всех каналах расположены в диапазоне $E_{\min}...E_{\max}$. Для формирования гистограммы область возможных значений средней энергии в этом диапазоне разбивается на *m* интервалов усреднения. Оптимальное количество интервалов усреднения *m* оценивается следующим выражением (2)

$$m \approx \operatorname{int}\left[\sqrt{K_{ch}}\right] + 1$$
 (2)

где K_{ch} – количество каналов.

Аргумент гистограммы E_k представляет граничное значение энергии *k*-го интервала усреднения (3)

$$E_k = \frac{k^* (E_{\max} - E_{\min})}{m} + E_{\min}$$
(3)

где m – количество интервалов усреднения, E_{\min} и E_{\max} – минимальное и максимальное значение средней энергии, k = 0...m.

Функция гистограммы p_k представляет нормированную плотность энергии. Дискретное значение плотности энергии $h_{n,k}$ на каждом *k*-ом интервале усреднения определяется относительным

количеством значений средней энергии E_{ch} , находящихся в этом интервале усреднения (4)

$$p_{k} = \frac{1}{K_{ch}} \sum_{n=0}^{K_{ch}-1} h_{n,k}, \quad h_{n,k} = \begin{cases} 1 & \text{при } E_{k} < E_{ch n} \le E_{k+1} \\ 0 & \text{остальных} \end{cases}$$
(4)

где K_{ch} – количество каналов, E_{chn} – средняя энергия в *n*-ом канале, $h_{n,k}$ – дискретное значение плотности энергии.

Для определения порогового значения до начала формирования гистограммы следует удалить из рассмотрения большие значения энергий. Механизм удаления сигналов с большими энергиями следующий. По всем значениям средней энергии находится среднее значение. После этого из рассмотрения удаляются сигналы, средняя энергия которых превышает среднее значение на 3 дБ. Затем проводиться вторая итерация удаления сигналов таким же образом.

Статистика (1) для нормального шума имеет распределение центральное хи-квадрат. Для распределения хи-квадрат при значениях N=16 или N=32, которые являются оптимальными, как установлено в ходе проведения исследований, все значения, которые превышают среднее значения на 3 дБ, составляют порядка 3-5% от общего числа значений. В случае если мощные детерминированные сигналы отсутствуют, потеря 3-5% значений не сильно точность повлияет на информативность И В гистограммы. очередь, свою если детерминированные сигналы присутствуют, то они удаляться из рассмотрения на первой или второй итерации. Величину 3 дБ можно изменять в зависимости от решаемых задач. Если предполагается наличие мощных детерминированных сигналов на первой итерации рационально изменить величину с 3 дБ до 6 дБ. Таким образом, уменьшится количество шумовых сигналов, которые удалятся из рассмотрения. В реальной жизни шумовое распределение не совпадает с распределением Гаусса, но это не влияет на качество отображения гистограммы. В общем случае всегда останется достаточное количество сигналов точного определения шумовых для порогового значения.

Как видно из рис. 5а (гистограмма по всем энергиям) в области малых значений энергии плотность энергии высокая, что соответствует большому количеству шумовых сигналов. В области больших энергий плотность энергии низкая, что соответствует малому количеству мошных детерминированных сигналов. При наличии мощных детерминированных сигналов все энергии, соответствующие шумовым сигналам, попадут в несколько (в пределе в один) начальных интервалов усреднения, и вычисление порогового значения по такой гистограмме будет способствовать низкой вероятности правильного обнаружения (что повлечет за собой нарушение работы первичных пользователей). Гистограмма на рис. 5б, сформированная с ограничением энергии, представляет собой гистограмму шума, поскольку в основном состоит из энергий соответствующим шумовым сигналам.



Рис. 5. Гистограмма энергии

По гистограмме шума и по заданному значению вероятности ложной тревоги вычисляется пороговое значение в соответствии с выражением (5)

$$p_{FA}(w_T) = 1 - \int_{0}^{w_T} p_k(w) dw$$
 (5)

где $p_k(w)$ – плотность распределения средней энергии шума, w_T – пороговое значение.

Решение о состоянии каждого канала (свободно или занято) принимается путем сравнения средней энергии каждого канала и порогового значения. Если средняя энергия канала превышает пороговое значение, канал считается занятым, иначе канал считается свободным.

VII. ЗАКЛЮЧЕНИЕ

В статье представлены технические требования, которые предъявляются к детектору свободных участков радиочастотного спектра. Рассмотрены различные варианты аппаратных средств и способов анализа. Приведен алгоритм работы детектора. Подробно рассмотрен процесс формирования гистограммы энергии и алгоритм определения порогового значения.

ЛИТЕРАТУРА

- Mahmood A.K. Abdulsattar, Zahir A. Hussein Energy Detector with Baseband Sampling for Cognitive Radio: Real-Time Implementation // Wireless Engineering and Technology, 2012, 3, 229-239.
- [2] Галкин В.А. Анализатор состояния эфира в радиостанциях адаптивного радио // Радиотехника. 2016. №9. С.146-155.
- [3] Отчет МСЭ-R SM.2256 (09/2012) Измерения и оценка занятости спектра.
- [4] M.A.Mchenry NSF spectrum occupancy measurement project summary. Shared Spectrum Company Report, Aug.2005.

Detector of Free Parts of Radio Frequency Spectrum

V.A. Zhmylev

MIET, Moscow, vz-75@yandex.ru

Abstract — This article presents hardware and software, with the help of which it is possible to establish communication sessions on temporarily free parts of the radio frequency spectrum. The efficiency of radio frequency spectrum can be improved by using temporarily free parts of the spectrum. Using temporarily free parts of the spectrum also allows to increase noise immunity, for example, working in FHSS (frequency-hopping spread spectrum) mode, etc. Secondary users' activity should not disrupt the primary users. For the organization of secondary communication networks it is necessary to implement the detector of free parts of radio frequency spectrum. The decision on a condition of the channel is made by comparison of a threshold and average energy of the channel. Basis for statistically reliable determination of threshold value is the histogram of energy. The histogram represents experimentally measured density of distribution of average energy in the channel for all channels in all range of frequencies. In this article requirements for the detector are presented. There are many variants of the hardware. Methods of analysis are considered. The algorithm of forming a histogram of average energy and the algorithm for determining the threshold value are described in detail.

Keywords — software defined radio, cognitive radio, secondary communication networks, concept of secondary users, energy detector, radio band analyzer.

Keywords — software defined radio, cognitive radio, secondary communication networks, concept of secondary users, energy detector, analyzer of ether.

REFERENCES

- Mahmood A.K. Abdulsattar, Zahir A. Hussein Energy Detector with Baseband Sampling for Cognitive Radio: Real-Time Implementation // Wireless Engineering and Technology, 2012, 3, 229-239.
- [2] Galkin V.A. Analizator sostojanija jefira v radiostancijah adaptivnogo radio (Ether of analyzer in radio stations of cognitive radio) // Radiotehnika. – 2016. - №9. – C.146-155.
- [3] Otchet MSE -R SM.2256 (09/2012) Izmerenija i ocenka zanjatosti spektra (Measurements and assessment of employment of a range).
- [4] M.A.Mchenry NSF spectrum occupancy measurement project summary. Shared Spectrum Company Report, Aug.2005.

Повышение скорости работы многоразрядного двоичного умножителя

А.Н. Якунин¹, Аунг Мьо Сан²

Национальный исследовательский университет «МИЭТ», г. Москва ¹yakunin.alexey@gmail.com; ²aungmyosan61028@gmail.com

Аннотация — В данной статье рассмотрены два типа многоразрядных двоичных умножителей, реализующих арифметическое умножение двух положительных чисел с фиксированной точкой: ведический умножитель и модифицированный древовидный умножитель. В работе проведено моделирование обеих архитектур для 8×8, 16×16, 32×32 - разрядных двоичных умножителей в среде САПР Ouartus II на базе ПЛИС Altera EP2SGX30DF780C3 семейства Stratix-II-GX. Выполнен их сравнительный анализ по аппаратным и временным затратам. По сравнений при реализации результату 32×32 двоичного умножителя умножитель, разрядного предложенный в этой работе, даёт выигрыш по скорости до 23% по сравнению с ведическим умножителем. Аппаратные затраты в предложенном умножителе снижаются на 22% по сравнению с ведическим умножителем. Кроме того, предложенную структуру можно масштабировать на большее количество разрядов, например 64×64, 128×128, 256×256 битов и т.д.

Ключевые слова - двоичный умножитель; полусумматор; полный сумматор; параллельно-префиксный сумматор (ППС); сумматор с запоминанием переноса (СЗП); ведический умножитель (ВУ); ALUTs; время задержки.

I. Введение

На сегодняшний день в цифровой схемотехнике большое внимание уделяется быстродействию работы устройств, реализующих арифметические операции, такие как сложение, вычитание, умножение, деление и т.д. [1]. Аппаратная реализация арифметических операций над двоичными числами является важным архитектурным элементом в микропроцессорах, цифровых сигнальных процессорах, математических сопроцессорах и других приложений [2]. Многие арифметические операции опираются на сложение, поэтому имея аппаратную структуру сумматора становится возможным реализовать умножение путём повторного сложения, вычитание путём логического отрицания одного операнда и деление путём повторного вычитания [3]. При схемной реализации выполнения арифметических операций чем меньше время задержки поступления данных на выход действующего устройства, тем выше его скорость работы. Одним из способов сокращения временных затрат является переход к параллельной архитектуре действующего устройства, в которой применяются базовые логические элементы: «НЕ», «И», «ИЛИ», «И-НЕ», «ИЛИ-НЕ» и т.д.

Умножитель – это логический комбинационный узел, выполняющий операцию умножения двух двоичных чисел. Умножители входят в состав более сложных цифровых устройств, например арифметикологических устройств (АЛУ). Поэтому аппаратная реализация эффективного умножителя необходима для повышения быстродействия АЛУ и, следовательно, процессора в целом.

В последнее десятилетие было изучено И разработано множество структур многоразрядных двоичных умножителей, построенных на основе различных методов ускорения умножения. По принципу суммирования частичных произведений, которые формируются путём умножения отдельных разрядов множителя на всё множимое, двоичные vмножители подразделяются на матричные и древовидные. В обоих структурах для суммирования частичных произведений широко применяются массивы взаимосвязанных комбинационных двоичных сумматоров. В реализации матричных умножителей двоичные сумматоры формируются в виде матрицы, а в древовидных – в виде дерева. Существующими вариантами матричных структур являются умножитель Брауна, умножитель Бо-були и умножитель Пезариса [4]. В них операция умножения сводится к параллельному формированию битов из п-разрядных частичных произведений с последующим их суммированием с помощью матриц сумматоров.

Сокращение времени выполнения суммирования частичных произведений в матричных умножителях реализуется в схемах, построенных по древовидной структуре. Если в матричных умножителях для суммирования n частичных произведений требуется n строк сумматоров, то в древовидных схемах количество каскадов сумматоров пропорционально log₂ n. Это приводит к сокращению времени вычисления суммирования частичных произведений. Однако аппаратной реализации при таких умножителей требуются дополнительные связи для объединения разрядов, имеющих одинаковых вес. Изза этого их аппаратные затраты выше. К древовидным умножителям относятся умножитель со схемой

перевернутого дерева [4], умножитель Уоллеса [4, 5], умножитель Дадда [4, 6].

В настоящее время к умножителям, реализующим аппаратные методы ускорения, относятся ведические умножители. Такой тип умножителя построен на основе ведической математики. Подробности ведической математики и реализации ведического умножителя рассмотрены в работе [7]. В ней проведено сравнение аппаратных и временных затрат ведического умножителя с другими умножителями, такими как матричный умножитель, умножитель с деревом Уоллеса и умножитель Бута. В результате сравнения в этой работе выявлено, что ведический умножитель обладает минимальной задержкой по сравнению с другими умножителями.

В данной работе рассмотрены два типа многоразрядных двоичных умножителей: ведический умножитель и модифицированный древовидный умножитель. Моделирование умножителей обеих схем для разрядностей 8×8, 16×16 и 32×32 проведено в среде САПР Quartus II на базе ПЛИС Altera EP2SGX30DF780C3 семейства Stratix-II-GX и выполнен их сравнительный анализ. Для оценки аппаратных и временных затрат обоих умножителей следующие параметры: аппаратная определены сложность по количеству логических блоков ПЛИС (единицей измерения в Quartus II являются так называемые ALUTs [8]) и максимальное время задержки. Достоверность выполнения операции умножения двоичных чисел подтверждена временными диаграммами результатов моделирования.

При аппаратной реализации многоразрядных двоичных умножителей сумматоры играют важную роль для выполнения суммирования частичных произведений. Поэтому прежде всего в работе проанализированы полусумматор и полный сумматор для сложения двух одноразрядных двоичных чисел. Затем для сложения двух многоразрядных двоичных чисел реализован параллельно-префиксный сумматор.

II. АППАРАТНАЯ РЕАЛИЗАЦИЯ ДВОИЧНЫХ СУММАТОРОВ

А. Полусумматор и полный сумматор

Одноразрядные сумматоры выполняют сложение двух одноразрядных двоичных слагаемых. По числу выходов одноразрядные двоичные входов и сумматоры можно разделить на полусумматор и полный сумматор. Полусумматор имеет два входа (А и B) и два выхода (S и $C_{\it out}$). S - это сумма A и B . Если А и В равны 1, то выход S должен стать равным 2, но такое число не может быть представлено в виде одного двоичного разряда. В этом случае результат указывается вместе с переносом C_{out} в следующий разряд. Таблица истинности отражает работу полусумматора, а логические уравнения и схемная реализация полусумматора показаны на рис.1a. Буквы НА (Half Adder) используются для обозначения полного сумматора.

Полный сумматор имеет три входа: A и B – разряды слагаемых и C_{in} – перенос из предыдущего (младшего) разряда; и два выхода: S – сумма по данному разряду и C_{out} – перенос в следующий разряд.



Рис. 1. а – полусумматор; б – полный сумматор

Работу полного сумматора отражает таблица истинности, а схемная реализация и логические уравнения полного сумматора показаны на рис.16. Буквы FA (Full Adder) используются для обозначения полного сумматора.

В. Параллельно-префиксный сумматор (ППС)

Сложение двух многоразрядных двоичных чисел может быть представлено схемой, состоящей из трёх каскадов (рис. 2) [9]: каскада предвычисления, каскада формирования параллельно-префиксного дерева и каскада формирования результата. На первом каскаде осуществляется предварительное вычисление битов $g_i = A_i B_i$, генерирующих сигнал переноса, битов $h_i = A_i \oplus B_i$, передающих сигнал переноса для любого входных операндов, $0 \le i \le n-1$.

На втором каскаде группа сигналов генерации переноса $G_{i:k}$ и распространения переноса $H_{i:k}$ вычисляется для каждого бита $0 \le i \le n-1$ по следующим логическим уравнениям:

$$G_{[i:k]} = \begin{cases} g_i; & ecnu \ i = k \\ G_{[i:j]} + H_{[i:j]} \cdot G_{[j-1:k]}; \ e \ npomub hom \ cny uae \end{cases}$$
(1)
$$H_{[i:k]} = \begin{cases} h_i; & ecnu \ i = k \\ H_{[i:j]} \cdot H_{[j-1:k]}; \ e \ npomub hom \ cny uae \end{cases}$$
(2)

Используя уравнения (1) и (2), и $G_{i:k}$ и $H_{i:k}$ используют входы из верхней части, охватывающей биты i: j, и из нижней части, охватывающей биты j-1:k. Затем эти части объединяются для формирования дерева сигналов генерации и распространения всего переноса, охватывающего биты i:k.

Используя сигналы $G_{i:k}$ из второго каскада, на последнем каскаде вычисляются выходные биты суммы S_i и бит выходного переноса по следующим формулам:

$$P_i = G_{[i:k]} \tag{3}$$

$$S_i = h_i \oplus P_{i-1} \tag{4}$$

где P_{-1} – входной сигнал $P_{in} = 0$.

На рис. 2 представлены 8-разрядный ППС и реализация используемых базовых схематичных узлов. Введены схематичные узлы: чёрный узел, белый узел, прямоугольник и треугольник, применяемые для большей наглядности при построении ППС.



Рис. 2. 8-разрядный ППС

В данной схеме сначала вычисляются g_i и h_i для пар разрядов первого каскада, далее для блоков из 4-х разрядов, затем для узлов из каскада формирования префиксного дерева, пока сигнал P_i не будет известен для каждого разряда. После этого результат S_i операции сложения вычисляется вместе с P_i В последнем каскаде приведённым выше по формулам (3), (4). Время задержки такого ППС пропорционально количеству уровней на каскаде формирования префиксного дерева. На этом каскаде количество уровней соответствует $\log_2 n$. С наращиванием разрядности входных операндов на рис. 3 показана схема параллельно-префиксного сумматора для 16-разрядных операндов.



Рис. 3. 16-разрядный параллельно-префиксный сумматор

С. Сумматор с запоминанием переноса (СЗП)

Одноразрядный сумматор с запоминанием переноса [10] представляет собой одноразрядный полный сумматор с входом С_і, переименованным в С, и выход C_{out} переименованным в Р. Идея состоит в том, чтобы входные складываемые 3 числа А, В и С преобразовать в два таких числа P и S, чтобы A + B + C = P + S. Ha puc. 4a показано, как nсумматоров с запоминанием переноса расположены так, чтобы при сложении трёх *n*-разрядных двоичных чисел A, B и C сформировать результат в виде P и S. Схема параллельного суммирования четырёх входных операндов с п-разрядами приведена на рис. 4б. Чтобы получить окончательный результат суммирования, необходимо сложить Р и S по каждому разряду. Все полные сумматоры в трёхоперандном СЗП с *n*разрядами независимы друг от друга, поэтому вся схема имеет задержку только полного сумматора.



Рис. 4. *а* - Трёхоперандный СЗП с *n* – разрядами; б-Четырёхоперандный СЗП с *n* - разрядами

В схеме трёхоперандного СЗП с n - разрядами видно, что в нулевом разряде первый полный сумматор выдаёт сумму трёх операндов S_0 и перенос в следующий разряд P_1 , а не P_0 . Поэтому при сложении в нулевом разряде значение переноса $P_0 = 0$. Такая же концепция может быть применена к схеме четырёхоперандного СЗП с n – разрядами. Следует обратить внимание на то, что в показанных схемах желаемый полный сумматор может быть заменён на полусумматор без третьего операнда C.

III. АППАРАТНАЯ РЕАЛИЗАЦИЯ ВЕДИЧЕСКОГО УМНОЖИТЕЛЯ

В данном разделе рассмотрим алгоритм умножения десятичных и двоичных чисел на основе ведической математики. Чтобы проиллюстрировать этот алгоритм, возьмём умножение двух десятичных чисел (A=25, B=76) и двух двоичных чисел (A=11 и B=11) по методу Урдхва Тирякхьяма [7], как показано на рис. 1.



Рис. 5. Алгоритм умножения двух десятичных и двоичных чисел на основе ведической математике

Из рис. 6 видно, что сначала нужно взять две цифры (A_0 и B_0) в нулевом разряде как от множителя, так и от множимого. Затем умножить их между собой и получить первый бит результата R_0 и перенос P_1 , поступающий на следующий шаг. На втором шаге умножаем цифры ($A_0 \times B_1$) и ($A_1 \times B_0$) путём умножения крест-на-крест и выполняем сложение ($A_0 \times B_1$) + ($A_1 \times B_0$) + P_1 . Получаем второй бит ответа R_1 и перенос P_2 на следующий шаг. На последнем шаге умножаем A_1 на B_1 и выполняем сложение ($A_1 \times B_1$) + P_2 . Получаем S_2 и P_3 . Сумма S_2 является третьим соответствующим битом, а перенос P_3 становится четвертым битом конечного ответа.

По этому методу можно создать схему (2×2) ведического умножителя (ВУ) с использованием четырех двухвходовых логических элемента «И» и двух полусумматоров. Схема (2×2) ведического умножителя показана на рис. 6.



Рис. 6. (2×2) ведический умножитель

Описанный метод может быть расширен на большее количество входных операндов, но для этого потребуются дополнительные аппаратные затраты. Ведический умножитель (4×4) разряда создается с использованием четырёх блоков (2×2) умножения. Кроме этого для сложения промежуточных результатов из каждого (2×2) умножителя требуются сумматоры с запоминанием переноса, а также для генерации окончательных результатов необходимо применять параллельно-префиксный сумматор.

Для наращивания разрядности входных операндов ведического умножителя на рис. 7 и рис. 8 приведены схемы (4×4), (8×8), (16×16) и (31×31) ведических умножителей.



Рис. 7.а - (4×4) ведический умножитель; б - (8×8) ведический умножитель



Рис. 8.а - (16×16) ведический умножитель; б - (31×31) ведический умножитель

А. Реализация модифицированного древовидного умножителя

В рамках данной работы авторами предложена схема модифицированного умножителя, в котором используется древовидная структура. Для реализации этой схемы на рис. 9 проведена точечная структура умножения двух 16×16-разрядных двоичных чисел, которая используется для реализации модифицированного умножителя.



Рис. 9. Точечная структура для выполнения 16×16разрядного умножения

Принцип работы данного умножителя виден из рисунка и состоит из следующих шагов:

1-й шаг: 16-разрядный множитель B15:0 (второй операнд) разбивается на 4 группы по четыре бита: B15:12, B11:8, B7:4 и B3:0.

2-й шаг: Умножается 16-разрядное множимое А (первый операнд) на каждую группу множителя В, и получаются 4 частичные произведения для каждой группы.

3-й шаг: Выполняется суммирование частичных произведений из каждой группы и выдаются значения S_{180}^{l} и P_{180}^{l} .

4-й шаг: Приписываются значения $S_{18:0}^{I}$ и $P_{18:0}^{I}$ из второй и четвёртой группы к значениям $S_{18:0}^{I}$ и $P_{18:0}^{I}$ из третьей и первой группы со сдвигом на 4 бита. Затем они суммируются, и выдаются значения $S_{23:0}^{2}$ и $P_{23:0}^{2}$ для первой и третьей группы.

4-й шаг: Приписываются значения S_{230}^2 и P_{230}^2 из третьей группы к значениям S_{230}^2 и P_{230}^2 из первой группы со сдвигом на 8 битов. Затем они суммируются, и выдаются значения S_{310}^3 и P_{310}^3 для первой группы.

5-й шаг: Наконец складываются S_{310}^3 и P_{310}^3 для первой группы, и получается 2n- разрядный результат.

На рис. 10 представлена схема (16×4)-блока, состоящего из матрицы логических элементов «И» (16×4) для выделения частичных произведений и четырёх-операндный СЗП для сложения частичных произведений.



Рис. 10. Схема (16×4)-блока



Рис. 11. 16×16-разрядный модифицированный умножителя



Рис. 12. 32×32-разрядный модифицированный умножителя

Используя рассмотренный принцип на рис. 11 предложены 16×16-разрядный модифицированный умножитель, содержащий 4 (16×4)-блоков, три 4операндных СЗП и 32-разрядный ППС. На рис. 12 представлен 32×32-разрядный модифицированный умножитель.

IV. СРАВНИТЕЛЬНЫЙ АНАЛИЗ И РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ УМНОЖИТЕЛЕЙ

Моделирование схем двух умножителей для выполнения 8×8, 16×16 и 32×32-разрядных двоичных чисел проведено в среде САПР *Quartus II* на базе ПЛИС Altera *EP2SGX30DF780C3* семейства *Stratix-II-GX*. Для оценки аппаратных и временных затрат определим значение комбинационных логических блоков *ALUTs* (H_{ALUTs}) и максимальное время задержки t_{max} по результату моделирования схем умножителей. Значение t_{max} может быть получено из формулы: $t_{max} = t_{cell} + t_L$; где, t_{cell} - общая задержка их межсоединений. В таблице 1 приведены оценки аппаратных и временных затрат двух умножителей для вычисления 8×8, 16×16, 32×32-разрядных двоичных чисел, которые получены в результате моделирования.

Таблица 1

Оценки аппаратных и временных затрат двух двоичных умножителей

| разрядность | Ведический умножтель | | Модифицированный умножитель | |
|-------------|-------------------------|--------------------------|--------------------------------|--------------------------|
| | H _{ALUT} | t _{max} (нс) | $H_{\scriptscriptstyle ALUT}$ | t _{max} (HC) |
| 8×8 | 151 | 15.0 | 119 | 14.9 |
| 16×16 | 666 | 28.7 | 553 | 18.9 |
| 32×32 | 2852 | 31.6 | 2201 | 24.3 |

Анализ результатов моделирования показывает, что аппаратные затраты у модифицированного умножителя меньше, чем у ведического умножителя, в частности у 32×32-рязрядных умножителей эта разница равна 651 *ALUTs* (22%). Кроме того, модифицированный умножитель даёт выигрыш в скорости до 23% по сравнению с ведическим умножителем. Результаты показывают, что чем больше разрядность умножителей, тем больше процентная разность скорости работы и аппаратной сложности между ними.

Достоверность выполнения операции 32×32разрядного двоичного умножения подтверждена результатами моделирования (временными диаграммами), полученными в среде САПР *Quartus II* и представленными на рис. 13.

| 0 p Nam | s 10.0 | ns | 20.0 ns | 30. <mark>0</mark> |
|------------|------------------------|-------------------|------------------|--------------------|
| | 4294967295 X | 585992471 | X 42558 | χ |
| B B | 4294967295 X | 115267568 | X 557409634 | χ |
| ∎ R C | 18446744065119617025 X | 67545926998480528 | X 23722239203772 | |

Рис. 13. Временная диаграмма результатов моделироания (32×32)-разрядного модифицированного умножителя

V. Заключение

В данной работе приведены схемные реализации полусумматора, полного сумматора, параллельнопрефиксного сумматора И мульти-операндного сумматора с запоминанием переноса для выполнения арифметического сложения двоичных чисел. Реализован ведический двоичный умножитель, Предложен модифицированный умножитель И проведён их сравнительный анализ. Результаты сравнения показали, что предложенный умножитель имеет лучшие параметры по аппаратным и временным затратам по сравнению с ведическим. Структуру умножителя можно масштабировать, спроектировав умножители с большим количеством разрядов, таких как 64, 128, 256 бит и т.д. Интересным направлением разработки дальнейшей является исследование возможности применения многоразрядного высокоскоростного умножителя для модулярной арифметики.

ЛИТЕРАТУРА

- [1] Червяков Н.И., Ляхов П.А., Валуева М.В., Криволапова О.В. Сравнительный анализ аппаратной реализации сумматороа на FPGA // «Науки. Инновации. Технологии» Северо-Кавказский федеральный университет. 2016. № 4. С. 98.
- [2] Акаш Кумар, Дипика Шарма. Анализ производительности различных типов сумматоров для высокоскоростного 32-битного умножения и накапливания. 2013. -V. 3. С. 1460.
- [3] Проф. Рашми Рахул Кулкарни. Сравнение между различными сумматорами // Журнал IOSR СБИС и обработка сигналов. 11-12, 2015. V 5. С. 2319-4197.
- [4] С. А. Орлов., Б. Я. Цилькер. Организация ЭВМ и системы. Учебник для вузов. 2-е издание, 2011. С. 192-206.
- [5] Bhupender Pratap Singh., Rakesh Kumar. Design and Implementatin 8-bit Wallace Tree Multiplier. International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering. Vol. 5, Issye 4, April 2016. P. 2307-2311.
- [6] Palaniappan Ramanathan., Ponnisamy Thangapandian Vanathi., Sundeepkumar Agarwal. High Speed Multiplier Design Using Decomposition Logic. Serbian journal of electrical engineering. Vol. 6, No. 1, May 2009, 33-42.
- [7] Abhijeet Kumar., Dilip Kumar., Siddhi. Hardware Implementation of 16*16 bit Multiplier and Square using Vedic Mathematics. International Conference on Signal, Image and Video Processing (ICSIVP) 2012. P. 309-314.
- [8] Altera Corporation. Stratix II Device Handbook, Volume 1. 2005// URL: http://www.ece.iastate.edu/~zambreno/classes/ cpre583/documents/altera/stx2_sii5v1_01.pdf. P. 20-22.
- [9] Дэвид Харрис. Таксономия параллельных префиксных сетей. Колледж Харви Мадд. IEEE 2013 / Р. 2213-2214.
- [10] Рето Циммерманн. Архитектуры двоичного сумматора для клеточного СБИС и их синтеза. Швейцарский федеральный институт технологии, Цюрих, 1997. С. 34-39.

Increasing the Speed of a Multi-bit Binary Multiplier

A.N. Yakunin¹, Aung Myo San²

National Research University "MIET", Moscow

¹yakunin.alexey@gmail.com; ²aungmyosan61028@gmail.com

Abstract — In this article, we consider two types of multi-bit binary multipliers realizing the arithmetic multiplication of two positive numbers with a fixed point: a Vedic multiplier and a modified tree multiplier. Both structures are modeled for (8×8) , (16×16) , (32×32) -bit binary multipliers in the Quartus II CAD environment based on the Altera EP2SGX30DF780C3 FPGAs of the Stratix-II-GX family. Their comparative analysis on hardware and time costs is performed. Based on the result of the comparisons, with the implementation of the (32×32) -bit binary multiplier, the multiplier proposed in this work gives a gain in speed up to 23% in comparison with the Vedic multiplier. In terms of hardware costs, the proposed multiplier reduces by 22% compared to the Vedic multiplier. In addition, the proposed structure can be scaled to a larger number of bits, for example (64×64), (128×128), (256×256) bits, and so on.

Keywords — binary multiplier; half-adder; full adder; parallel-prefix adder (PPA); a carry-save adder (CSA); Vedic multiplier (VM); ALUTs; delay time.

REFERENCES

- [1] Chervyakov N.I., Lyakhov P.A., Valueva M.V., Krivolapova O.V., Comparative analysis of the hardware implementation of the adder on FPGA // "Nauki. Innovation. Technologies »North-Caucasian Federal University. 2016. № 4. C. 98.
- [2] Akash Kumar, Deepika Sharma. Analyze the performance of various types of adders for high-speed 32-bit multiplication and accumulation. 2013.-V.. 3. C. 1460.
- [3] Prof. Rashmi Rahul Kulkarni. Comparison between different adder // Journal of IOSR VLSI and Signal Processing. 11-12, 2015. V 5. P. 2319-4197.
- [4] SA Orlov., B. Ya. Zilker. Organization of computers and systems. Textbook for high schools. 2nd edition, 2011. pp. 192-206.
- [5] Bhupender Pratap Singh., Rakesh Kumar. Design and Implementatin 8-bit Wallace Tree Multiplier. International Journal of Advanced Research in Electrical Electronics and

Instrumentation Engineering. Vol. 5, Issye 4, April 2016. P. 2307-2311.

- [6] Palaniappan Ramanathan., Ponnisamy Thangapandian Vanathi., Sundeepkumar Agarwal. High Speed Multiplier Design Using Decomposition Logic. Serbian journal of electrical engineering. Vol. 6, No. 1, May 2009, 33-42.
- [7] Abhijeet Kumar., Dilip Kumar., Siddhi. Hardware Implementation of 16*16 bit Multiplier and Square using Vedic Mathematics. International Conference on Signal, Image and Video Processing (ICSIVP) 2012. P. 309-314.
- [8] Altera Corporation. Stratix II Device Handbook, Volume 1. 2005// URL: http://www.ece.iastate.edu/~zambreno/classes/ cpre583/documents/altera/stx2 sii5v1 01.pdf. P. 20-22.
- [9] David Harris. Taxonomy of parallel prefix networks. Harvey Mudd College. IEEE 2013 / P. 2213-2214.
- [10] Reto Zimmermann. The architecture of the binary adder for the cell-based VLSI and their synthesis. Swiss Federal Institute of Technology, Zurich, 1997. pp. 34-39.

Цифровой измеритель частоты с повышенной точностью и быстродействием для доплеровского измерителя скорости

Е.В. Ливенцев, А.Л. Переверзев, Е.В. Примаков, Д.В. Рыжкова, А.М. Силантьев

Национальный исследовательский университет «МИЭТ», г. Москва, olmer.aod@gmail.com

Аннотация — В статье рассмотрена задача быстрого частоты низкочастотного сигнала измерения в доплеровском измерителе скорости. Приведён анализ методов измерения частоты и предложена структура аппаратного блока частотомера с уточнением перехода через ноль для системы на основе ПЛИС, управляющей В измерителем. статье приведено сравнение погрешности способа измерения частоты с уточнением перехода через ноль с погрешностью аналогичного способа без аппроксимации. В статье приведены результаты анализа характеристик разработанного блока на примере реального сигнала. В ходе работы было проведено моделирование и получены экспериментальные зависимости погрешности предложенного способа измерения от соотношения частот измеряемого и опорного сигнала.

Ключевые слова — частотомер, ИУС, ПЛИС, FPGA.

I. Введение

В настоящее время задача измерения скорости движущихся объектов актуальна для систем помощи водителям, систем автоматического управления летательными аппаратами и других информационноуправляющих систем. Повышение точности и измерения скорости быстродействия позволяет уменьшить время реакции, которое во многих случаях является критическим параметром всей системы управления в целом. Для измерения скорости могут использоваться фотокамеры, радары и смешанные системы. В случае применения радаров скорость может определяться по частоте доплеровского смещения отражённого от объекта сигнала.

В рамках информационно-управляющей системы (ИУС) частотомер может быть реализован различными способами. Это может быть как отдельный блок, спроектированный на базе дискретных компонентов, так и функциональный блок системы на кристалле или ПЛИС. В случае, когда ИУС базируется на ПЛИС, заказной или полузаказной интегральной схеме, частотомера реализация В виде сложно функционального блока позволяет удешевить изделие за счёт отказа от части дискретной компонентной базы, а также снизить энергопотребление.

В настоящей работе приведён анализ методов измерения частоты и проведена оценка их погрешностей. На основе оценки погрешностей выбран способ измерения с уточнением перехода через

предложена нопь и структура спожно реализующего функционального блока, ланный способ. Блок был разработан и применён в доплеровском измерителе скорости. В статье приводятся схема проверки цифрового блока измерителя частоты и экспериментальные данные измерений. Выбор способа измерения производился исходя из требований к блоку по точности определения частоты 0,5% для сигнала в полосе 10 -16 КГц. При этом время измерения не должно превышать 2 мс.

II. МЕТОДЫ ИЗМЕРЕНИЯ ЧАСТОТЫ

В процессе проектирования блока частотомера был проведён анализ методов измерения частоты. Ниже приводится краткий обзор рассмотренных методов, их методическая погрешность и краткие комментарии о применимости для измерений с повышенной точностью и быстродействием.

А. Быстрое преобразование Фурье

Для определения частоты сигнала может быть использовано быстрое преобразование Фурье (БПФ) [1]. Оценка абсолютной погрешности по частоте, получаемая при использовании БПФ, фактически равна половине разрешающей способности и вычисляется как

$$\Delta F = \frac{F_s}{N/2},\tag{1}$$

где F_s — частота следования выборок дискретного сигнала, N — число точек БПФ.

Использование БПФ позволяет снизить влияние шумов (в том числе фазовых) на измерение, но обладает рядом недостатков:

• необходимость в накоплении выборки,

• быстрый рост выборки и вычислительной сложности при увеличении точности.

Эти особенности не позволяют использовать БПФ для получения частотомера с требуемой точностью и скоростью измерения.

В. Электронно-счетный частотомер

Электронно-счётный частотомер вычисляет измеряемую частоту путём подсчёта количества

импульсов, сформированных из входного сигнала (чаще всего получаемых с помощью компаратора) за заранее определённое время. Интервал времени задаётся с помощью опорного генератора. Частота измеряемого сигнала вычисляется как

$$F = \frac{N}{T_{U}},$$
(2)

где N — количество подсчитанных импульсов, а T_{H} — временной интервал измерения.

Методическая погрешность этого метода измерения обусловлена несовпадением времени появления импульсов опорной частоты с появлением импульсов, сформированных из сигнала, и не превышает периода одного импульса опорной частоты [2].

Тогда справедлива оценка абсолютной погрешности измеренной частоты:

$$\Delta F = \frac{\Delta N * F}{N} = \frac{F}{N} = \frac{1}{T_{H}},\tag{3}$$

где ΔN – погрешность измерения в периодах импульсов опорной частоты.

Для получения более точных значений измеряемой частоты требуется не только обеспечить стабильность опорного генератора, но и расширять временной интервал, на котором проводится измерение, что приводит к низкому быстродействию. Для уменьшения погрешности измерения интервал должен быть тем больше, чем ниже измеряемая частота.

Этот метод измерения частоты также не удовлетворяет поставленным перед блоком частотомера требованиям к точности и скорости.

С. Измеритель периода сигнала

Вычисление частоты с использованием данного метода заключается в подсчёте количества импульсов опорного генератора сигналов, поступающих за единичный период измеряемого [3].

Точность измерения зависит от отношения измеряемой и опорной частоты. Оценка погрешности имеет вид

$$\Delta F = \frac{2F_R^2}{F_s - 2F_R},\tag{4}$$

где F_s — частота следования импульсов опорного генератора, F_R — частота измеряемого сигнала.

Преимуществами метода являются простота вычислений и реализации, а также скорость, так как для проведения измерения требуется всего один период измеряемого сигнала. Однако оценка частоты по единичному периоду не гарантирует достоверности измерения, если измеряемый сигнал модулирован по частоте или зашумлён.

III. Способ измерения частоты с аппроксимацией

Как видно из приведённого выше обзора наиболее подходящим методом для реализации быстродействующего измерителя частоты является метод измерения периода сигнала. Для повышения точности измерений предлагается использовать способ измерения периода с уточнением перехода через ноль [4, 5, 6].

Так как измеритель обрабатывает дискретный сигнал и использует для измерения периода опорную частоту, то прямое уточнение положения перехода через нулевой уровень невозможно.



Рис. 1. Целочисленная и дробная часть измеренного периода в тактах опорной частоты

В качестве решения предлагается использовать аппроксимацию сигнала вблизи нуля для вычисления дробной части отсчётов опорной частоты, в которых произошел переход сигнала через нулевой уровень (рис. 1) [5, 6]. Тогда уточнённый период сигнала можно представить как:

$$T = T_1 + T_2 + T_3, (5)$$

где T_1 – дробное число тактов опорной частоты, прошедшее с предыдущего перехода через ноль, T_2 – измеренная длительность периода в тактах опорной частоты, T_3 – дробное число тактов опорной частоты, предшествующих следующему переходу через ноль.

Применив линейную аппроксимацию для сигнала вблизи нуля (рис. 2) получим:

$$T_1 = 1 - \frac{S_0 - S(t_0)}{S(t_1) - S(t_0)},$$
(6)

$$T_3 = \frac{S_0 - S(t_0)}{S(t_1) - S(t_0)}.$$
(7)

При этом вычисленное при переходе сигнала через ноль значение T_3 является составляющей частью предыдущего измеренного периода, а T_1 – текущего.



Рис. 2. Линейная аппроксимация сигнала вблизи нуля

Такой способ позволяет уточнить период измеряемого сигнала, но не компенсирует чувствительности метода к фазовым и амплитудным шумам.

IV. ОЦЕНКА МЕТОДИЧЕСКОЙ ТОЧНОСТИ СПОСОБА ИЗМЕРЕНИЯ С УТОЧНЕНИЕМ ПЕРЕХОДА ЧЕРЕЗ НОЛЬ

Для сравнительной оценки методической точности измерений периода сигнала с аппроксимацией и без программные неё были использованы модели соответствующих измерителей. Необходимо отметить, что погрешность измерения может варьироваться в зависимости от фазового соотношения измеряемого и опорного сигналов. Поэтому для оценки погрешностей использованы максимальные значения разности фаз. Рис. 3 демонстрирует зависимость максимальной методической погрешности от соотношения частот опорного сигнала и измеряемого сигнала.



Рис. 3. Сравнительная оценка методической точности способов измерения периода

Измеритель с аппроксимацией вблизи уровня нуля демонстрирует меньший уровень методической погрешности, особенно при соотношении частот сигналов в 5 раз и более, сохраняя при этом многие достоинства реализации без аппроксимации.

V. АППАРАТНАЯ РЕАЛИЗАЦИЯ

Для управления устройством измерения скорости была спроектирована система на ПЛИС, включающая цифровой фильтр, процессорное ядро, блок управления СВЧ, контроллеры периферии, блок амплитудного детектора и блок измерителя частоты (рис. 4). Система была реализована на ПЛИС Xilinx Artix-7 25T. Система построена на основе софт-процессора uRV[7] (разработан CERN) архитектуры RISC-V со встроенной памятью данных и команд. Она функционирует следующим образом: блок управления СВЧ настраивает и формирует управляющие сигналы для СВЧ части измерителя скорости, результатом работы которого является НЧ сигнал доплеровской частоты. Сигнал поступает на АЦП далее на цифровой полосовой фильтр и затем в блоки детектора и счётчика частоты.

Цифровой фильтр применён для подавления внеполосных паразитных гармоник и помех, которые, как было замечено выше, негативно влияют на работу измерителя частоты. В измерителе скорости был применён КИХ фильтр 55-го порядка с полосой пропускания 10 – 16 кГц и запиранием на уровне -60 дБ. Групповая задержка фильтра при частоте следования выборок 100 кГц составляет 0.23 мс.

Использование в системе аппаратного блока обнаружителя позволяет сократить время от обнаружения сигнала до момента измерения. Результаты измерения считываются и обрабатываются на процессорном ядре.

Сложно функциональный блок измерителя частоты обеспечивает доступ к настройкам и результатам измерений через контроллер системной шины. Такое решение позволило осуществлять настройку блока и управление его работой, а также получить доступ к измеренным значениям со стороны программного обеспечения. В качестве механизма сигнализации блоком окончания вычислений было решено использовать прерывание.

Стоит отметить, что обработка прерывания не происходит мгновенно. Из этого следует необходимость обеспечить такую частоту вызова прерываний, чтобы время, необходимое для обработки полученных данных, было меньше её периода. В случае, когда частота сигнала превышает скорость обработки прерываний, можно путем введения в сложно функциональный блок памяти полученных значений сократить число вызовов прерывания.



Рис. 4. Структурная схема аппаратуры неконтактного датчика скорости



Рис. 5. Структурная схема блока измерения периода сигнала с аппроксимацией сигнала вблизи уровня нуля

Блок измерения периода сигнала (рис. 5) функционирует следующим образом:

Сигнал поступает на вход детектора перехода через нулевой уровень. На второй вход детектора поступает предыдущий отсчет сигнала. Детектор перехода через нуль представляет собой цифровую схему на основе компараторов. При обнаружении перехода сигнала через нулевой уровень детектор формирует строб, запускающий работу блоков вычисления целой части и дробной части.

Блок вычисления целой части представляет собой целочисленный измеритель периода сигнала на основе счётчика, в качестве опорной частоты использующий частоту дискретизации АЦП.

Блок вычисления дробной части принимает на вход два отсчёта, между которыми был обнаружен переход через нулевой уровень, и производит интерполяцию по формулам (6), (7), а вычисленный на предыдущем периоде сигнала результат сохраняется и используется ресурсоёмкой вычисления (5). Наиболее лпя операцией, реализуемой блоком вычисления дробной части, является деление. Для оптимизации аппаратных затрат и повышения максимальной тактовой частоты используется конвейеризованный делитель. По причине более высоких задержек в тракте вычисления дробной части возникает необходимость в синхронизации потоков выходных данных из двух блоков. Для этой цели использована буферная память типа "очередь".

Вычисленные компоненты T1, T2 и T3 поступают в сумматор, формирующий окончательный результат измерения в формате с фиксированной точкой, который затем передаётся в блок управления. Блок управления координирует работу вычислительных блоков, генерирует сигналы прерываний и реализует связь с процессором по системной шине. Прерывания вырабатываются сразу по окончании вычисления значения одного периода сигнала.

VI. ХАРАКТЕРИСТИКИ АППАРАТНОЙ РЕАЛИЗАЦИИ

В рамках работы были получены экспериментальные сравнительные результаты измерений частоты спроектированным блоком и блоком без уточнения положения перехода через ноль.

Схема экспериментальной измерительной установки представлена на рис. 6.



Рис. 6. Схема измерительной установки

установка включает Измерительная в себя лабораторный генератор низкочастотных сигналов, а также макетную плату с АЦП (разрешение 8 бит, частота следования выборок - 100 кГц) и ПЛИС, в цифровая которой реализована проверяемая подсистема измерителя. Сигнал с эталонной частотой из генератора поступает на вход макетной платы с АЦП. После оцифровки и фильтрации сигнал поступает в измеритель. Измеренные значения частоты передаются на персональный компьютер и затем статистически обрабатываются на нём. Диапазон измеряемых частот соответствует рабочему диапазону устройства и составляет 10 – 16 кГц с шагом в 0.5 кГц.

С целью компенсации чувствительности блока измерителя частоты к шумам и повышения точности в программном обеспечении цифровой подсистемы измерителя скорости был реализован алгоритм фильтрации импульсных помех: после проведения 10 измерений периода отбрасывается минимальное и максимальное значение, а за результат принимается среднее арифметическое от оставшихся значений.

Статистическая обработка выполнялась над выборкой в 3000 результатов измерений. За меру погрешности измерения принято максимальное зафиксированное отклонение измеренной частоты от эталонной генератора). Графическая (частоты интерпретация результатов измерения и обработки данных представлена на рис. 7.



Рис. 7. Сравнительная оценка экспериментальной точности способов измерения периода

В результате эксперимента установлено, что измеритель частоты сигнала с аппроксимацией уровня нуля предлагает выгодное соотношение точности и быстродействия. В рассмотренной реализации для проведения одного измерения необходимо накопление 10 периодов сигнала, что соответствует 1 мс на нижней границе рабочей полосы частот. С учетом групповой задержки цифрового полосно-пропускающего фильтра максимальная оценка времени одного измерения - 1.5 учётом пост-обработки. Относительная мс с погрешность определения частоты не превысила 0.41%. Кроме того, подтверждена возможность работы измерителя на относительно низких частотах дискретизации, что позволяет уменьшить требования к характеристикам АЦП и тем самым упростить и удешевить конечное устройство.

VII. Выводы и направления дальнейших исследований

Был реализован аппаратный блок измерителя частоты, основанный на способе измерения периода с аппроксимацией перехода через ноль. Для анализа погрешности метода была построена математическая Результаты измерения модель. точностных характеристик характеристик быстродействия И реализованного блока показали соответствие модели. Продемонстрировано, что применение выбранного способа позволяет существенно снизить требования к аппаратной части, сохраняя высокую точность и быстродействие.

В дальнейшем планируется провести исследование влияния длительности выборки и применения различных алгоритмов пост-обработки на точностные характеристики и быстродействие измерителя.

ЛИТЕРАТУРА

- [1] Баскаков С.И. Радиотехнические цепи и сигналы. М.: Высшая школа, 2000. С. 396-397.
- [2] Lyons R.G. Understanding Digital Signal Processing. Boston, M.A.: Addison-Wesley, 1996. P. 738-739.
- [3] Нефедов В.И., Сигов А.С., Битюков В.К., Самохина Е.В. Электрорадиоизмерения. М.: Форум, 2018. С. 200-203.
- [4] Terauds M., Zuters K. Low Complexity DSP Block Design Methodology For Road Speed Monitoring // Advances in Wireless and Optical Communications (RTUWO). Riga. 2017, P. 162-166.
- [5] Kim S., Nguyen C. On the development of a multifunction millimeter-wave sensor for displacement sensing and lowvelocity measurement // IEEE Transactions on Microwave Theory and Techniques. 2004. P. 2503-2512.
- [6] Plantier G., Servagent N., Sourice A., Bosch T. Real-time parametric estimation of velocity using optical feedback interferometry // IEEE Transactions on Instrumentation and Measurement. 2001. P. 915-919.
- [7] URL: https://www.ohwr.org/projects/urv-core/wiki (дата обращения: 25.05.2018).

Accurate High-Speed Frequency Meter for Doppler Initial Velocity Measurement

E.V. Liventsev, A.L. Pereverzev, E.V. Primakov, D.V. Ryzhkova, A.M. Silantiev

National Research University of Electronic Technology, Moscow, olmer.aod@gmail.com

Abstract — The article considers the issue of high-speed frequency measurement of low-frequency signal for initial velocity estimation in Doppler velocity meter. Authors reviewed existing frequency measurement techniques and proposed another measurement method with a structure of the hardware frequency meter block for a system on FPGA controlling Doppler meter. The article gives a measurement error calculation for proposed method and analysis of developed structure characteristics on real signal.

Keywords — frequency meter, control system, FPGA.

REFERENCES

- Baskakov S.I. Radiotekhnicheskiye tsepi i signaly (Radio engineering circuits and signals). Moscow: Higher School, 2000. P. 396-397.
- [2] Lyons R.G. Understanding Digital Signal Processing. Boston, M.A.: Addison-Wesley, 1996. P. 738-739.
- [3] Nefedov V.I., Sigov A.S., Bityukov V.K., Samokhina E.V. Radioelektronnyye izmereniya (Electroradiometry). M .: Forum, 2018. P. 200-203.
- [4] Terauds M., Zuters K. Low Complexity DSP Block Design Methodology For Road Speed Monitoring // Advances in

Wireless and Optical Communications (RTUWO). Riga. 2017, P. 162-166.

- [5] Kim S., Nguyen C. On the development of a multifunction millimeter-wave sensor for displacement sensing and lowvelocity measurement // IEEE Transactions on Microwave Theory and Techniques. 2004. P. 2503-2512.
- [6] Plantier G., Servagent N., Sourice A., Bosch T. Real-time parametric estimation of velocity using optical feedback interferometry // IEEE Transactions on Instrumentation and Measurement. 2001. P. 915-919.
- [7] URL: https://www.ohwr.org/projects/urv-core/wiki (access date: 25.05.2018)

Опыт разработки радиационно-стойкого контроллера накопителя для бортовой космической аппаратуры

А.В. Руткевич, Д.И. Воронков, И.Ю. Сысоев, Н.Н. Хайло, А.А. Вейков

ООО «НПП «Цифровые решения», г. Москва, igor@dsol.ru

Аннотация — В работе описан опыт разработки контроллера памяти, позволяющий на базе микросхем NAND Flash создавать накопители ёмкостью до 256 Гбайт и скоростью передачи данных до 150 Мбайт/с. Контроллер поддерживает обмен информацией по интерфейсу сериалайзера TLK2711-SP, асинхронному интерфейсу статической памяти, последовательным интерфейсам и PCI. Стойкость контроллера к предельной накопленной дозе 100 крад и порог возникновения тиристорного эффекта не менее 67,9 МэВ×см²/мг.

Ключевые слова — твердотельный накопитель информации, контроллер памяти, РСКН, NAND Flash, SLC, ONFI, PCI, PC/104-Plus, космическая аппаратура, FTL, ЭКБ, TLK2711-SP.

I. Введение

На сегодняшний день основным элементом для энергонезависимого хранения больших объёмов данных является твердотельная память. К наиболее доступной твердотельной памяти относится NAND Flash память. Микросхемы NAND Flash памяти позволяют достичь наилучших значений таких параметров накопителя как: информационная ёмкость, энергопотребление, габариты, масса, надёжность, скорость записи и воспроизведения, стойкость к механическим воздействиям.

Управление твердотельной памятью типа NAND Flash связано с рядом ограничений, задаваемых внутренним устройством микросхем хранения. Микросхемы состоят из двух основных частей - массива для хранения и контроллера. При записи данные сначала попадают в контроллер, затем отдельной командой переносятся в массив. Запись и чтение из массива могут осуществляться только постранично. Типовые размеры страниц современных микросхем: 4 кбайта, 8 кбайт и 16 кбайт. Операция записи – это установка в нулевое логическое значение соответствующих бит страницы. Операции записи всегда предшествует операция стирания. Во время операции стирания все биты блока устанавливаются в единичное логическое значение. Операция стирания применима только к блокам. Ёмкость блока, как правило, составляет несколько мегабайт. Блок содержит примерно нескольких сотен страниц. Причём страницы, принадлежащие одному блоку, должны записываться только по порядку. Дополнительно при работе с NAND Flash памятью нужно учитывать, что количество циклов «стирание-запись» для каждого блока ограничено. Для SLC памяти, наиболее надёжной, оно составляет от 60 000 до 100 000 циклов. Также технология изготовления ячеек NAND Flash памяти не исключает возможность ошибки при считывании информации из массива.

Исходя из вышеперечисленных ограничений следует, что значения таких параметров, как скорость случайных и последовательных записи и чтения, равномерное распределение нагрузки, целостность информации и надёжность зависят от радиационностойкого контроллера накопителя (РСКН).

С развитием технологии в течение последних пятнадцати лет в космической сфере наблюдается переход на твердотельные накопители информации. В мировой промышленной сфере, благодаря организации комитетов лидерами отрасли, получилось выработать общие требования и стандарты к РСКН. Они касаются как внешних интерфейсов, так и поддерживаемой функциональности. Это отразилось в РСКН массового применения от таких компаний как Intel, Marvell, Phison и Seagate (paнee – SandForce).

Космическая сфера отличается большей консервативностью и закрытостью, чем промышленная. Новые разработки должны быть совместимыми с уже применяемыми решениями. При этом изделия, разработанные различными организациями, могут быть несовместимы между собой. Каждая организация придерживается собственной логики построения сложных систем, набора архитектур устройств и интерфейсов. Результатом этого является самостоятельная разработка каждой организацией своего собственного контроллера и программного обеспечения для него, как правило, на базе программируемой логической интегральной схемы. Как следствие, при низкой унификации повышается стоимость результата и ухудшаются характеристики используемого решения.

Компания ООО «НПП «Цифровые решения», основываясь на собственном опыте разработки РСКН для космического применения на базе ПЛИС, а также требованиях потребителей, предложила универсальный РСКН для космической аппаратуры ЦКРФ.467316.001.

Разработанный РСКН предназначен для широкого применения в спутниках дистанционного зондирования Земли (ДЗЗ), системах телеметрии, чёрных ящиках, промежуточных высокоскоростных буферах данных высокой ёмкости.

II. ТРЕБОВАНИЯ К РСКН

А. Поддержка ранее разработанных систем на базе PC/104-Plus

Ранее разработанные и применяемые интерфейсы отличаются относительно невысокими скоростными характеристиками. Верхняя граница для записи и чтения данных составляет 30 Мбайт/сек. Также ёмкость ячейки запоминающего устройства, необходимая для функционирования системы, не превышает нескольких гигабайт.

Такие невысокие скоростные и ёмкостные требования, с учётом современной номенклатуры проверенных процессорных устройств, позволяют строить системы, в которых основная функциональная нагрузка ложится на процессор.

В космической аппаратуре подобного рода распространённым решением стало использование микрокомпьютеров на базе стандарта PC/104-Plus. В данной системе все модули подключаются по единому интерфейсу PCI, по которому передаются как команды управления, так и данные.

Таким образом разрабатываемый РСКН должен поддерживать шину РСІ, обеспечивать скорость записи и чтения данных не менее, чем 30 Мбайт/с, позволять подключать массив NAND Flash памяти не менее 10-ти Гбайт и при этом вся ячейка на базе РСКН должна удовлетворять требованиям конструктива РС/104-Plus.

В. Поддержка специализированных систем с асинхронной памятью

Другим распространённым устоявшимся решением является использование управляющего процессорного устройства повышенной сбоеустойчивости в качестве основного узла. Скорости передачи данных в таких системах также не превышают 30 Мбайт/с, а ёмкость не превышает 10-ти Гбайт.

С точки зрения периферии такие процессорные устройства обязательно содержат выводы GPIO, интерфейсы UART, SPI и интерфейс для подключения ОЗУ или ПЗУ (как правило, SRAM, MRAM, ROM, EEPROM). Что касается остальных интерфейсов, то нельзя выделить общие решения.

Из вышеизложенного следует, что единственно правильным решением для обеспечения унификации при заданных скоростных требованиях будет реализация в РСКН интерфейса асинхронной памяти для подключения к внешнему процессорному устройству (ВПУ).

В этом случае для ВПУ управление РСКН будет выглядеть как запись или чтение команд и данных в определённые переменные карты памяти. РСКН тогда находится на стороне ведомого устройства, а источником и потребителем данных со стороны ведущего устройства является ВПУ.

С. Поддержка ранее разработанных систем на базе набора последовательных интерфейсов

Другой распространённый класс систем состоит из ВПУ и независимых от него источников (датчиков) и потребителей (ретрансляторов) данных. В таком случае ВПУ выполняет только функцию управления, без анализа принимаемых и передаваемых данных.

Устройства, являющиеся источниками и потребителями пользовательских данных, как правило, имеют невысокие скоростные характеристики, не более 50 Мбит/с, и интерфейсы, в которых основной упор сделан на минимизацию точек межсоединений, сигнальных и управляющих линий.

Стандартный состав такого интерфейса включает в себя линию тактирования сигнала, одну линию данных и одну линию управления для возможности остановки внешнего потока информации. При этом передача данных осуществляется только в одну сторону.

На этапе разработки РСКН была заложена возможность поддержки 4-х последовательных интерфейсов на приём и 4-х последовательных интерфейсов на передачу на скорости 50 Мбит/с. Также в целях гибкости РСКН должен обеспечивать механизм модификации потока данных с участием ВПУ. Поддержка работы с 8-ю внешними устройствами в целях снижения накладных расходов на обработку прерываний должна быть реализована с помощью системы дескрипторов.

D. Поддержка перспективных систем на базе радиационно-стойкого сериалайзера TLK2711-SP

Современные перспективные высокоскоростные системы оперируют потоками информации на символьной скорости не менее 2,5 Гбит/с и скорости приёма и передачи пользовательских данных не менее 100 Мбайт/с.

Высокоскоростная передача в сложных системах невозможна без использования дифференциального потока данных и, как следствие, устройства сериализации и десериализации потока информации. Сериализация данных на большой скорости возможна только с использованием сложных аналоговых устройств, что может привести к невозможности применения изделия в космической аппаратуре. Поэтому номенклатура решений не отличается разнообразием.

Стандартной схемой для такой задачи является использование микросхем сериалайзера TLK2711-SP [3] или разработки сериалайзера на базе радиационностойкой ПЛИС. Поэтому на этапе разработки РСКН была заложена совместимость с микросхемой TLK2711-SP и возможность передачи и приёма данных на скорости до 150 Мбайт/с.

Дополнительно, РСКН должен обеспечивать возможность подключения массива NAND Flash памяти ёмкостью 256 Гбайт.

Е. Требования к NAND Flash интерфейсу

Благодаря разработанному стандарту ONFI и его широкой поддержке, требования к NAND Flash интер-

фейсу РСКН для космической аппаратуры практически совпадают с требованиями к РСКН для коммерческих изделий.

ООО «НПП «Цифровые решения» имеют большой опыт по разработке контроллеров NAND Flash памяти [1]. Интерфейс NAND Flash памяти РСКН должен поддерживать одновременную работу по 4-м каналам данных, поддержку массивов NAND Flash памяти на базе наиболее надёжных микросхем типа SLC, возможность работы по любым 2-м или 1-му каналу данных, поддерживать ёмкость массива энергонезависимой памяти не менее 256 Гбайт. В целях помехоустойчивости в соответствии с требованиями производителя микросхем NAND Flash памяти, РСКН должен осуществлять автоматическое обнаружение и коррекцию до 12-ти битовых ошибок кодами БЧХ [2] в блоках размером 512 байт. Для снижения нагрузки на внутренний процессор контроллер NAND Flash памяти в составе ТНИ должен включать аппаратные ускорители для отправки, приёма команд и функцию троирования служебных данных.

F. Требования к аппаратной поддержке алгоритмов работы с NAND Flash памятью

При использовании NAND Flash памяти устройство управления должно обеспечивать возможность реализации следующих алгоритмов, относящихся к уровню взаимодействия с NAND Flash памятью (Flash Transition Layer) [3]:

- запись и чтение по секторам размером 1 кбайт;
- организация механизмов отображения логических блоков на область физических блоков (Logic Unit Number Table);
- управление неисправными блоками (Bad Block Management);
- контроль выравнивания износа (Wear Leveling);
- организация кэш-буферов для последовательной записи (Serial Logic Block);
- организация кэш-буферов для случайной записи (Random Logic Block);
- дефрагментация данных в массиве NAND Flash памяти (Garbage Collection);
- защита таблицы логических блоков от внезапного выключения питания (Power-Off Protection).

III. ОПИСАНИЕ ФУНКЦИЙ РСКН

А. Структурная схема РСКН

Структурная схема РСКН показана на рис. 1.

РСКН состоит из следующих основных блоков:

- RISC CPU микропроцессорное ядро, реализованное по гарвардской архитектуре с сокращённым набором инструкций;
- IМ память инструкций микропроцессорного ядра объёмом 32 кбайта;



Рис. 1. Структурная схема РСКН

- Loader загрузчик исполняемой программы микропроцессорного ядра из массива NAND Flash памяти;
- RAM ОЗУ микропроцессорного ядра объёмом 8 кбайт;
- LUN таблица отображения логических блоков на физические блоки массива NAND Flash памяти;
- FI-0..3 контроллеры интерфейсов массива NAND Flash памяти;
- TLK контроллер интерфейса TLK2711-SP;
- РСІ контроллер РСІ;
- MCU контроллер интерфейса асинхронной шины;
- SRL TX контроллер передатчика последовательного интерфейса;
- SDL RX-0..3 контроллеры последовательных интерфейсов приёма данных;
- DMA контроллер прямого доступа к памяти;
- DB 2x32 2 буфера данных ёмкостью 32 кбайта каждый для передачи информации по шинам TLK2711-SP, MCU, PCI и шинам последовательного интерфейса к внешним принимающим устройствам;
- DB 128 буфер данных ёмкостью 128 кбайт для приёма информации по последовательным интерфейсам от внешних передающих устройств;
- UART, SPI, I2C, GPIO контроллеры интерфейсов к вспомогательной периферии.

IV. Сценарии использования РСКН

Рассмотрим основные сценарии взаимодействия ВПУ и РСКН.

А. Режим работы по асинхронному интерфейсу

Асинхронный интерфейс (MCU) предназначен для подключения РСКН к ВПУ по шине асинхронной памяти. РСКН поддерживает работу с 8, 16 и 32 битной шиной данных MCU.

Большинство современных процессоров содержат встроенный контроллер внешний памяти, поддерживающий работу с асинхронной памятью. Рассмотрим работу РСКН с радиационно-стойким процессором 5023BC016 «Спутник» [4] в качестве ВПУ, подключённого к РСКН по интерфейсу МСU. Встроенное ОЗУ размером 128 кБ и контроллер прямого доступа к памяти позволяют снять с процессорного ядра задачу переноса массива данных из ОЗУ в буфер РСКН и обратно.

В режиме MCU задачей процессора «Спутник» является управление PCKH, а также запись и чтение полезных данных. В этом режиме процессор имеет непосредственный доступ к буферам данных и регистрам управления PCKH. Это позволяет самостоятельно заполнять буфер по мере поступления данных, либо использовать механизм прямого доступа к памяти.

Минимальный размер блока данных, записываемого в массив NAND Flash памяти – одна страница размером 32 кбайт при работе в четырёхканальном режиме. Общий объем буфера данных РСКН составляет 64 кбайт. Это позволяет организовать двойную буферизацию под управлением ВПУ. При заполнении первой половины буфера ВПУ отдаёт команду на перенос данных в массив NAND Flash памяти и приступает к заполнению второй половины. После заполнения второй части буфера ВПУ проверяет, переданы ли данные из первой части, и, если данные переданы, отдаёт команду на перенос данных из второй части. Это позволяет организовать непрерывный процесс записи данных в накопитель. Аналогичным образом организован механизм двойной буферизации при чтении данных из накопителя.

Несомненным преимуществом работы в режиме MCU является возможность изменения данных внутри буфера со стороны BПУ. Это позволяет обойти ограничение NAND Flash памяти на страничную запись. При необходимости записи части страницы BПУ отдаёт команду на чтение страницы из Flash памяти, изменяет необходимые данные в буфере и отдаёт команду на запись данных. Данная функция также может быть полезна при необходимости сопровождения записываемых данных служебной информацией (функция журналирования).

В. Режим работы по интерфейсу РСІ

РСКН поддерживает подключение к ВПУ по шине РСІ на частоте 33 МГц. Наличие интерфейса РСІ позволяет легко интегрировать систему хранения данных на базе РСКН в готовую или вновь разрабатываемую систему стандарта PC/104-Plus или CompactPCI. Модульность этих систем позволяет использовать в новых системах ранее разработанные изделия без внесения каких-либо изменений, а также наращивать объем системы хранения данных за счёт увеличения количества подключаемых модулей.

Кроме того, шина PCI используется в качестве системной шины некоторых специализированных процессоров, что позволяет подключать устройства на базе шины PCI непосредственно к процессору без использования дополнительных преобразователей. Примером могут служить радиационно-стойкие процессоры семейства «Комдив» [5] и радиационно-стойкий процессор AT697E фирмы Atmel.

РСКН поддерживает подключение к шине РСІ в режимах «Master» и «Target».

При работе РСКН в режиме «Target» адресное пространство буфера данных РСКН отображается на адресное пространство шины РСІ и ВПУ имеет непосредственный доступ к буферу данных. Процесс записи и чтения данных происходит аналогично режиму МСU. ВПУ заполняет буфер данных и формирует команды для переноса данных во Flash память. Задача организации двойной буферизации ложится на ВПУ.

При работе РСКН в режиме «Master» ВПУ указывает начальный адрес в пространстве шины РСІ и размер пакета данных. За передачу данных отвечает контроллер прямого доступа к памяти, входящий в состав РСКН. По окончании передачи блока данных РСКН формирует прерывание и возвращает результат выполнения команды.

С. Режим работы с сериалайзером TLK2711-SP

РСКН поддерживает интерфейс подключения к радиационно-стойкому сериалайзеру TLK2711-SP (интерфейс TLK). Данный интерфейс используется для записи и чтения данных совместно с одним из интерфейсов управления (MCU или PCI).

Сериалайзер TLK2711-SP представляет собой приемо-передатчик для построения высокоскоростных систем двунаправленной передачи данных типа «точка-точка» со скоростью передачи полезных данных от 1,6 до 2 Гбит/с. Он позволяет при соединении двух устройств заменить параллельную шину на последовательный интерфейс со средой передачи в виде печатного монтажа, коаксиального или оптоволоконного кабеля.

Интерфейс подключения к сериалайзеру представляет собой две параллельных шины: шину передатчика и шину приёмника. Каждая шина состоит из 16 бит данных, двух бит признака служебной информации и одного сигнала тактовой частоты. Частота параллельного интерфейса от 80 до 125 МГц. В сериалайзере применяется кодирование 8/10 и механизм передачи служебной информации в виде К-символов.

Разделение интерфейсов управления и передачи информации позволяет получить максимальную скорость записи/чтения полезных данных. Как правило, поточная передача данных подразумевает последовательную запись и чтение больших массивов информации. При работе в режиме TLK ВПУ передаёт одну команду на запись или чтение большого массива данных (вплоть до полного объёма накопителя) и ожидает прерывания по завершении выполнения команды. Дальнейшая работа по обеспечению буферизации и переносу данных из буфера в массив NAND Flash памяти и обратно выполняется РСКН.

При записи данных в накопитель РСКН игнорирует все принимаемые К-символы, а полезные данные записывает в буфер данных и автоматически переносит их во Flash память по мере заполнения буфера. Объем записываемых данных и адрес для записи данных в массив Flash памяти предварительно передаются ВПУ с помощью команды по интерфейсу управления.

При чтении данных из накопителя РСКН формирует поток данных в соответствии с форматом, заданным ВПУ с помощью специальной команды настройки приемо-передатчика интерфейса ТLК. При этом заданными К-символами в потоке выделяются начало и конец посылки, а также происходит прореживание потока данных указанным количеством К-символов для снижения средней скорости передачи полезных данных.

D. Режим с приёмом и передачей данных по последовательным интерфейсам

РСКН поддерживает работу со специализированным последовательным интерфейсом, который состоит из четырёх каналов ввода и четырёх каналов вывода информации (интерфейс SRL). Данный интерфейс используется для записи и чтения данных совместно с одним из интерфейсов управления (МСU или PCI).

Каждый канал ввода и вывода информации состоит из 3-х сигналов: сигнал данных, сигнал синхроимпульса и сигнал разрешения передачи. Одновременно могут работать все четыре канала ввода информации. Одновременно может работать один из четырёх каналов вывода данных.

Так как при записи данных по последовательному интерфейсу источники данных не синхронизированы между собой, то возникает сложная задача управления потоками данных, поступающими из разных каналов. Как правило, данные каждого канала сохраняются по разным адресам во Flash памяти. Поэтому в РСКН они накапливаются в отдельных частях общего буфера и, по мере накопления, переносятся во Flash память.

Для управления этим процессом используется механизм дескрипторов и задач. ВПУ по интерфейсу управления формирует в отдельной области памяти РСКН очередь дескрипторов и очередь задач для каждого канала.

Дескриптор содержит информацию для конкретного приёмника о том, какая область буфера данных выделена для записи принимаемой информации, и каков её объем. Дескрипторы обрабатываются приёмником последовательного интерфейса.

Задача содержит информацию о том, какой адрес в массиве Flash памяти соответствует данным конкрет-

ного дескриптора. Задачи обрабатываются Flash контроллером.

По команде от ВПУ приёмник приступает к обработке дескрипторов. После получения указанного в дескрипторе объёма данных приёмник сигнализирует о завершении обработки текущего дескриптора и приступает к выполнению следующего. После этого Flash контроллер приступает к обработке задачи и переносит данные, соответствующие обрабатываемым задаче и дескриптору, во Flash память по адресу, указанному в задаче.

Функцией ВПУ является заполнение очередей дескрипторов и задач, отслеживание их состояния и своевременное пополнение очередей во избежание потери полезных данных.

При чтении данных используется аналогичный механизм. Сначала, в соответствии с задачей, происходит перенос данных из Flash памяти в буфер данных РСКН. После этого ВПУ при необходимости дополняет данные служебной информацией и формирует дескриптор. Передатчик обрабатывает дескриптор и передаёт соответствующую информацию по последовательному интерфейсу.

V. Схемы применения

Рассмотрим несколько типовых схем систем хранения данных на базе РСКН с использованием разных наборов интерфейсов и устройств управления.

А. Схема с использованием TLK2711-SP

Использование отдельных интерфейсов для управления и передачи данных позволяет достичь наибольших скоростей передачи данных. На рис. 2 показан пример системы, где в качестве ВВУ используется процессор «Спутник», подключённый к РСКН по интерфейсу МСU. Источник данных подключён к системе по интерфейсу TLK.



Рис. 2. Схема работы в режиме TLK

Подобная схема может быть применена в спутниках связи или дистанционного зондирования Земли, где накопитель информации не может быть размещён в непосредственной близости от источника данных (приёмника, фотокамеры, регистрирующей аппаратуры) и получателя данных (передатчика).

B. PC/104-Plus

На рис. 3 показан пример бортовой системы сбора телеметрической информации, где в качестве накопителя используется система на базе РСКН. Такая система может быть построена на базе модулей формата PC/104-Plus. В данной системе шина PCI используется как для управления PCKH, так и для записи/чтения.



Рис. 3. Схема применения в конструктиве PC/104-Plus

Такая система может быть построена на базе модулей формата PC/104-Plus. В данной системе шина PCI используется как для управления PCKH, так и для записи/чтения данных. Большой выбор готовых модулей формата PC/104-Plus позволяет легко модернизировать подобные системы под используемые в аппаратуре интерфейсы.

С. Схема с последовательными интерфейсами

На рис. 4 приведён пример системы, в которой для записи данных в накопитель используются четыре последовательных канала ввода информации (интерфейс Serial RX). Источником данных для подобного интерфейса может служить аппаратура сбора данных спутника Д33: фотокамеры, датчики излучения, система телеметрии спутника.



Рис. 4. Пример схемы с использованием последовательных интерфейсов

Как правило, накопленная информация передаётся на Землю в течение сеанса связи. Для этого может использоваться один из четырёх последовательных каналов вывода информации (интерфейс Serial TX).

D. С процессором «Спутник»

Процессор «Спутник» может использоваться совместно с РСКН не только в качестве устройства управления, но и в качестве устройства чтения и записи данных. Используемые в процессоре «Спутник» интерфейсы позволяют разрабатывать как самостоятельные системы на его основе, так и использовать его в качестве контроллера подсистемы хранения данных в составе других систем.

На рис. 5 представлен пример системы на базе процессора «Спутник», которая может быть использована в качестве устройства управления или сбора телеметрической информации.



Рис. 5. Схема с использованием процессора «Спутник»

В данном устройстве в качестве накопителя используется система на базе модуля РСКН, подключённого к шине контроллера внешней памяти процессора по интерфейсу MCU. Большой выбор периферийных интерфейсов позволяет использовать процессор «Спутник» для решения широкого круга задач.

VI. Методы повышения стойкости к воздействию ИИ КП в РСКН

Для повышения стойкости к воздействию ИИ КП в РСКН применён ряд топологических, схемотехнических и алгоритмических решений [6].

А. Топологические методы

Для защиты от тиристорного эффекта (Single Event Latch-up, SEL) и деградации от накопленной дозы (Total Ionizing Dose, TID) в библиотеке элементов, на базе которой разработан РСКН, используется специализированная библиотека элементов. В основе топологии элементов лежат следующие решения:

 элементы выполнены по технологии КМОП 180 нм, оптимальной с точки зрения сбоеустойчивости и стойкости к накопленной дозе;

 для повышения стойкости к тиристорному эффекту используются охранные кольца;

 для повышения стойкости к накопленной дозе используются кольцевые затворы.

В. Схемотехнические методы

При функционировании в условиях воздействия ИИ КП сбои, исправляемые схемотехническими методами, можно классифицировать следующим образом:

- одиночный сбой (Single Event Upset, SEU);
- множественный сбой (Multibit Upset, MBU);

- функциональный отказ (Single Event Functional Interrupt, SEFI);
- пробой затвора (Single Event Gate Rupture, SEGR).

Перечень схемотехнических решений в РСКН, повышающих сбоеустойчивость:

1) защита расширенным кодом Хэмминга блоков статической памяти для защиты от SEU;

2) повышение стойкости памяти LUN периодическим сканированием с использованием аппаратного блока (Scrubbing) для защиты от SEU;

 использование кодирования данных кодами БЧХ в NAND Flash памяти с корректирующей способностью, в 1,5 раза превышающей рекомендации производителей NAND Flash для защиты от SEU;

4) троирование управляющих регистров контроллера интерфейса NAND Flash памяти для защиты от SEU;

5) защита контрольной суммой, многократным резервированием экземпляров, кодами БЧХ, расположением в различных физических областях образов программы в NAND Flash памяти для защиты от SEFI;

6) троирование служебной информации о блоках для защиты от SEFI;

 предоставление возможности контроля со стороны ВПУ программного счётчика для определения нештатной ситуации для защиты от SEFI; 8) предоставление возможности контроля со стороны ВПУ возникновения некорректируемых ошибок в памяти инструкций и ОЗУ микропроцессорного ядра для защиты от SEFI;

9) возможность коррекции конфигурации массива NAND Flash памяти после изготовления аппаратуры за счёт изменения набора используемых каналов, режимов работы каналов (1-/2-/4-х канальные режимы) и набора используемых выводов выбора микросхем (Chip Select) памяти для защиты от SEGR.

С. Программные методы

В программном обеспечении для РСКН предусмотрены следующие алгоритмические решения:

1) троирование команд и параметров команд от ВПУ по интерфейсам MCU и PCI для защиты от SEU;

2) использование сторожевого таймера для контроля попадания микропроцессорного ядра в бесконечный цикл для защиты от SEFI;

 заполнение неиспользуемой области памяти инструкций безусловными переходами в функцию для обработки нештатных ситуаций;

4) создание таблицы неисправных блоков (Bad Block Management).

Таблица 1

| Функциональные параметры РСКН | | | | |
|--|----------|--------------------|--|--|
| | Ед. изм. | Значение | | |
| Поддерживаемые ёмкости массива NAND Flash памяти | Гбайт | 16/32/ 64/128/256 | | |
| Размер сектора | кбайт | 1 | | |
| Максимальная скорость записи и чтения по интерфейсу TLK2711-SP | Мбайт/с | 150 | | |
| Максимальная скорость записи и чтения по интерфейсу РСІ | Мбайт/с | 60 | | |
| Максимальная скорость записи и чтения по МСИ | Мбайт/с | 60 | | |
| Максимальная скорость записи по последовательному интерфейсу | Мбит/с | 50 | | |
| Максимальная скорость чтения по последовательному интерфейсу | Мбит/с | 50 | | |
| Интерфейс РСІ | версия | 2.2 | | |
| Стандарт ONFI | версия | 2.2 | | |
| Количество экземпляров ПО в массиве NAND Flash памяти | ШТ. | 4096 | | |
| Корректирующая способность кодов БЧХ | | 12 бит на 512 байт | | |
| Поддержка функции перепрограммирования | | есть | | |

Таблица 2

Надёжность РСКН

| | Единицы измерения | Значение |
|---|-------------------------|---------------|
| Диапазон рабочих температур | °C | от -60 до125 |
| Группа аппаратуры по ГОСТ РВ 20.39.304-98 | | 5.3 |
| Срок активного существования | лет. | 15 |
| Предельная накопленная доза | крад | не менее 100 |
| Порог возникновения тиристорного эффекта | МэВ×см ² /мг | не менее 67,9 |
| Порог возникновения SEU | МэВ×см ² /мг | 6,2 |
| Интенсивность сбоев на ГСО | шт./(с×бит) | 10-9 |

VII. ПАРАМЕТРЫ РСКН

Параметры разработанного РСКН показаны в табл. 1 и 2.

VIII. ЗАКЛЮЧЕНИЕ

Компанией ООО «НПП «Цифровые решения» был разработан РСКН, предназначенный для широкого

применения в космической аппаратуре. Он позволяет модернизировать уже разработанные системы и обеспечить перспективные космические аппараты высокоскоростными и ёмкими ячейками памяти. Партия опытных образцов прошла предварительные испытания и КД присвоена литера «О». Внешний вид РСКН показан на рис. 6.



Рис. 6. Внешний вид радиационно-стойкого контроллера накопителя для бортовой космической аппаратуры

Изделие реализовано на 2-х кристаллах. Основной кристалл имеет размеры 10х10 мм. Аналоги по внешним интерфейсам отсутствуют. В отечественной и зарубежной космической аппаратуре задачи управления твердотельным накопителем, как правило, решаются с

помощью программируемых логических интегральных схем.

Благодаря заложенным конструктивным, топологическим, схемотехническим и алгоритмическим решениям РСКН соответствует современным требованиям, применяемым к электронной компонентной базе.

Использование РСКН в космической аппаратуре позволит унифицировать архитектуру, а также снизить стоимость и сроки её разработки.

ЛИТЕРАТУРА

- Руткевич А.В., Поляков Е.А., Сысоев И.Ю. СФ-блок контроллера массива NAND Flash памяти / Сб. трудов «МЭС-2014». Ч. 4 М.: ИППМ РАН. 2014 С. 7-12;
- [2] Micheloni R., Marelli A., Ravasio R. Error Correction Codes for Non-Volatile Memories, 2008, 388 p.
- [3] Micheloni R., Crippa L., Marelli A. Inside NAND Flash Memories, 2010, 582 p.
- [4] URL: http://dsol.ru/projects/plis_n_sbis/sputnik/ (дата обращения: 15.04.2018);
- [5] URL: https://www.niisi.ru/devel.htm (дата обращения: 15.04.2018);
- [6] Чумаков А.И. Действие космической радиации на интегральные схемы. М: Радио и связь, 2004. 320 с.

The Experience of Rad-Hard Controller SSD Development for Space Application

A.V. Rutkevich, D.I. Voronkov, I.Y. Sysoev, N.N. Khaylo, A.A. Veykov

Institute Digital Solutions, SPE, LLC, igor@dsol.ru

Abstract — The study deals with implementation of solid state memory controller. Solid state storage device consists of two main units: controller and NAND Flash memories. A maximum storage capacity is 256 Gbytes. A maximum data transfer rate is 150 Mbyte/s. The controller has the following set of interfaces: PCI, asynchronous interface of static random access memory, serial interfaces, interface of serdes TLK2711-SP. The controller is rad-hard: TID equals 100 krad. SEL threshold is not less than 67,9 MeV×cm²/mg. By the means of wide range of rad-hard microprocessors one can extend a set of interfaces on the spacecraft. Asynchronous interface and PCI support data transfer rate up to 60 Mbyte/s. For example, using processor 5023BC016 ("Sputnik") one can set up the system with CCSDS, SpaceWire or MIL-STD-1553 buses. The controller implemented in ceramic dimpled grid array package with 399 pads. There are a lot of topological, schematic and algorithmic methods implemented in the controller to achieve rad-hard tolerance. Topological solutions include 180 nm standard cell process and guard rings. Circuit methods include SEC-DEC Hamming coding for SRAM, scrubbing, triple register map, CRC and multiplication for binary program image, an availability for external processor to read program counter and error counter, NAND Flash array reconfiguration (set of using channels and set of using CS signals). Algorithmic solutions include triple set of command code and command parameters during task initialization, watchdog

timer for infinite loop detection, filling of unused instruction area of un-conditional jump into a special function, creation of bad block table. The proposed controller has no functional analogue, as such problems are usually solved with FPGA. Using of the application specified integrated circuit allows to reduce overall dimensions, time, cost, a number of used components and power consumption of spacecraft.

Keywords — solid state drive, memory controller, controller SSD, NAND Flash, SLC, ONFI, PCI, spacecraft, FTL, components, TLK2711-SP.

REFERENCES

- Rutkevich A.V., Polyakov E.A., Sysoev I.Y. NAND Flash memory controller IP-core / Sb.trudov «MES-2014». CH 4. M.: IPPM RAN. 2004. S. 7-12.
- [2] Micheloni R., Marelli A., Ravasio R. Error Correction Codes for Non-Volatile Memories, 2008, 388 p.
- [3] Micheloni R., Crippa L., Marelli A. Inside NAND Flash Memories, 2010, 582 p.
- [4] URL: http://dsol.ru/projects/plis_n_sbis/sputnik/ (access date: 15.04.2018);
- [5] URL: https://www.niisi.ru/devel.htm (access date: 15.04.2018);
- [6] Chumakov A.I. Deystvie cosmicheckoy radiacii na integralnye schemy. M: Radio i svyaz, 2004. 320 p.
Нечувствительный к задержкам блок умножения-сложения-вычитания с плавающей точкой

И.А. Соколов, Ю.В. Рождественский, Ю.Г. Дьяченко, Ю.А. Степченков, Н.В. Морозов, Д.Ю. Степченков, Д.Ю. Дьяченко

Институт проблем информатики Федерального исследовательского центра «Информатика и управление» Российской академии наук (ФИЦ ИУ РАН)

{YRogdest, YDiachenko, YStepchenkov, NMorozov, DStepchenkov}@ipiran.ru

Аннотация — Представлено устройство совмещенного умножения-сложения-вычитания, независящее от задержек в элементах и проводниках. Оно полностью соответствует стандарту IEEE 754 и реализует одновременно операции сложения и вычитания третьего операнда из произведения первых двух. Каждый 64разрядный операнд содержит либо одно число двойной точности, либо два числа одинарной точности. Для увеличения быстродействия умножитель, реализующий модифицированный алгоритм Бута, разбит на две ступени конвейера с ускоренным переключением в спейсер. Схема кодера Бута интегрирована во входное FIFO. Выполнение сложения и вычитания в троичном избыточном коде обеспечивает сокращение аппаратных затрат всего блока. С целью сокращения энергопотребления, блок построен как одноканальное устройство. Блок разработан на базе объемной КМОП технологии с проектными нормами 65 нм с использованием библиотеки стандартных элементов, дополненной самосинхронными элементами, и обеспечивает производительность на уровне 3 гигафлопс.

Ключевые слова— избыточное кодирование, троичный сумматор, "дерево" Уоллеса, эквихронная зона, FIFO.

I. Введение

Аппаратное совмещение умножения двух входных операндов и последующего сложения-вычитания с третьим входным операндом в одном устройстве может выполняться с промежуточным округлением произведения (типично для сигнальных процессоров первых поколений) или с одним округлением общего результата. Версия с одним округлением получила название fused multiply-add (FMA). Она стала де-факто стандартной операцией современных центральных процессоров, так как обеспечивает более высокую точность вычислений по сравнению с вариантом с промежуточным округлением.

Известные в настоящее время реализации данного устройства в подавляющем большинстве являются синхронными [1]-[3]. Асинхронные решения, претендующие на принадлежность к классу самосинхронных устройств[4]-[5], не отслеживают в полной мере завершение переключения во всех элементах схемы перед переходом в следующую фазу работы. Поэтому они не могут считаться устройствами, правильное функционирование которых не зависит от задержек в элементах и проводниках, т.е. нечувствительных к задержкам (НЧЗ, Delay-Insensitive) при любых условиях эксплуатации.

Сохранение работоспособности НЧЗ схем при сверхмалых значениях питающих напряжений открывает широкие перспективы для применения в портативных изделиях с аккумуляторным питанием и создания бортовых комплексов, не требовательных к объему и стабильности энергоресурсов. Устойчивая работа в экстремальных условиях достигается за счет аппаратной избыточности и дополнительных временных затрат на индикацию и фазу «спейсера» в работе НЧЗ схем. Однако грамотное проектирование НЧЗ схем позволяет существенно снизить эту избыточность, а в ряде случаев, например, в отказоустойчивых устройствах [6], получить результаты лучше, чем в синхронных аналогах.

Ранее авторами уже предпринимались попытки разработки устройства FMA гигафлопсного класса, независимого от задержек в элементах, – SIFMA [7]-[8] и SIFPC [9]-[10]. Однако для достижения предельного быстродействия в SIFMA использовался принцип спекулятивной индикации, не обеспечивавший его стопроцентной самопроверяемости, а SIFPC был построен как двухканальное устройство с двухступенчатым конвейером с общим входом и выходом и адаптивной индикацией, не учитывавшей реальных размеров эквихронной зоны (ЭЗ) [11].

В данной статье излагаются результаты проектирования 64-разрядного НЧЗ вычислительного устройства, выполняющего операции умножения со сложением и умножения с вычитанием с плавающей точкой в соответствии со стандартом IEEE 754 (Delay Insensitive Fused Multiply-Add-Subtract, DIFMAS). Оценки сложности реализации и временные характеристики обсуждаются в разделах II и III. В качестве прототипа математической модели вычислений, включая избыточное представление сомножителей, была выбрана синхронная реализация блока FMA [12], поскольку она обеспечивает наилучшие характеристики при использовании самосинхронного схемотехнического базиса. Методологические аспекты построения самосинхронного устройства FMA были подробно рассмотрены в [7].

II. Особенности DIFMAS

Как и в предшествующих разработках, каждый из трех обрабатываемых операндов содержит либо одно число двойной точности, либо два числа одинарной точности. В последнем случае выполняются две независимые операции: FMA и FMS, – над двумя тройками операндов одинарной точности.

А. Структурная схема DIFMAS

Разработка современных вычислительных средств ведется в направлении обеспечения минимального энергопотребления при достаточно высокой производительности. Это определяется тенденцией к использованию сравнительно невысокой тактовой частоты и большого числа вычислительных узлов на одну СБИС для высокопроизводительных компьютеров.

Наличие двух фаз в работе любой НЧЗ схемы (активной – рабочей и паузы – спейсерной) наталкивает на идею использовать два параллельных канала, фазы работы которых чередуются. Именно такая реализация была представлена в работах [9]-[10]. Она позволила достичь среднестатистической производительности на уровне 3,15 Гфлопс при работе с синхронным окружением и 3,90 Гфлопс при отсутствии непроизводительного ожидания отклика от синхронного окружения об успешном считывании результата с выхода FMA. Однако это вылилось в большие аппаратурные затраты и относительно большое энергопотребление устройства.

В связи с этим была предложена новая структурная схема устройства, вычисляющего сумму (FMA) и разность (FMS) произведения двух первых операндов и третьего операнда, – НЧЗ сопроцессора с плавающей точкой (DIFMAS), показанная на рис. 1.



Помимо ядра DIFMAS, она содержит входное и выходное FIFO, которые повышают быстродействие DIFMAS при работе с синхронным окружением [7] за счет буферизации потока данных. Входное и выходное FIFO реализованы как регистровая самосинхронная память емкостью 4 слова.

Схема ядра DIFMAS представлена на рис. 2. В отличие от предыдущих разработок устройства FMA [7]-[10], в ней нет дублирующихся каналов или блоков умножителя. Тем не менее, она обладает лучшим, по сравнению с предшествующими реализациями, соотношением "быстродействие / аппаратурные затраты" за счет инновационной индикации блока умножителя и более широкого использования избыточного самосинхронного кодирования.



В. Блок умножения DIFMAS

Блок умножения является наиболее ресурсоёмким (55–60% от всего устройства), энергопотребляющим и времязатратным блоком среди всех функциональных блоков DIFMAS. В данной работе за основу выбран базовый вариант блока умножения, использующийся в подавляющем большинстве современных вычислительных устройств. Это чисто комбинационная схема, состоящая из кодера Бута (Radix-2) и "дерева" Уоллеса на сумматорах с избыточным кодированием (redundant binary representation, RBR) [12]-[13].

Описанные в работах [7]-[10] варианты самосинхронного блока FMA включали в себя 2 блока умножения для достижения требуемой производительности. В качестве самосинхронного кода использовались парафазный для кодера Бута и избыточный (троичный) для "дерева" Уоллеса. Оба кода имеют двухфазную дисциплину кодирования, состоящую из рабочей и спейсерной фаз. Оптимизация самосинхронной дисциплины взаимодействия между ступенями конвейера FMA позволила обеспечить латентность вычислений согласно выражению:

$T_L = N^* T_W + T_S,$

где: T_L – латентное время работы конвейера FMA; T_W – длительность рабочей фазы ступени конвейера; T_S – длительность спейсерной фазы ступени конвейера; N – количество ступеней конвейера.

Однако длительность цикла работы одной ступени конвейера осталась по-прежнему равной суммарной длительности двух фаз и это определило производительность самосинхронного FMA. При реализации блока умножения в виде одной ступени конвейера в 65-нм КМОП технологии длительность среднестатистического цикла составила 1,5 нс без учёта топологической реализации. Использование двух параллельных блоков умножения в SIFMA и двух параллельных каналов обработки троек операндов в SIFPC позволило существенно ускорить работу блока самосинхронного FMA за счет увеличения аппаратурных затрат.

В то же время, простое разбиение блока умножения на две ступени конвейера позволяет уменьшить длительность каждой ступени до 1 наносекунды без учёта трассировки, что является явно недостаточным. Дальнейшее увеличение числа ступеней конвейера в блоке умножения будет чрезвычайно затратным и не приведёт к заметному сокращению длительности цикла работы ступени конвейера. Основной причиной этого являются существенные затраты времени на индикацию завершения всех процессов переключений элементов на каждой ступени конвейера при высокой разрядности обрабатываемых операндов. Дополнительные затраты привносятся и за счет организация самосинхронных регистров для хранения промежуточных результатов.

Для индикации только одного выходного регистра одной ступени конвейера блока умножения требуется пятислойное индикаторное "дерево" на трехвходовых гистерезисных триггерах (Г-триггерах, [11]). Суммарная задержка формирования индикаторных сигналов в двух фазах для 106 выходных разрядов составляет в 65-нм КМОП технологии 300-350 пс. Индикация регистров привносит ещё 100-150 пс. В результате суммарная длительность рабочей и спейсерной фаз приближается к 1 нс, в то время как максимальная длительность цикла работы ступени конвейера для обеспечения гигафлопсной производительности без учёта топологии не должна превышать 800-850 пс. В разработанном варианте НЧЗ блока умножения удалось решить эту задачу с помощью совместного использования целого ряда специальных методов:

 принудительного ускорения спейсерной фазы самосинхронного цикла;

- ввода дополнительного регистра временного хранения промежуточных данных во второй стадии "дерева" Уоллеса;

 перехода от последовательной индикации, в пределах библиотечных элементов, к общей параллельной индикации внутренних переменных с последующей оптимизацией временных параметров;

- применения более быстрых троичных сумматоров в "дереве" Уоллеса.

Рис. 3 демонстрирует схему одноразрядного троичного сумматора из "дерева" Уоллеса блока DIFMAS. Схема сумматора на рис. 3 более сложная (на 18%), но обладает существенно лучшим быстродействием (на 27%) в составе конвейера в сравнении со схемой одноразрядного троичного сумматора-прототипа из предшествующих вариантов блока FMA [8].



Вход FS на рис. 3 обеспечивает ускоренное переключение одноразрядного сумматора и всего "дерева" Уоллеса в спейсер. Три индикаторных выхода Ind1 – Ind3 объединяются в один индикаторный выход "дерева" Уоллеса распределенной индикаторной подсхемой, учитывающей соотношение задержек их формирования в разных разрядах и каскадах "дерева" Уоллеса. Реализация этого подхода позволила дополнительно повысить быстродействие умножителя на 12%.

Разработанный с использованием предложенных методов блок умножения DIFMAS реализуется в виде двух ступеней конвейера и обеспечивает суммарную длительность рабочей и спейсерной фазы 850-900 пс без учёта топологической реализации. Это в 1,7 раза меньше исходного классического варианта, что позволяет получить DIFMAS гигафлопсного класса с использованием одного умножителя. Дополнительные аппаратные затраты, обеспечивающие повышение бы-

стродействия умножителя, составляют 12-13%. Однако суммарные затраты на умножитель в составе DIFMAS в результате снижаются на 40-45% в сравнении с SIFMA и SIFPC за счет использования одного умножителя вместо двух. Пропорционально снизились энергетические затраты на операцию умножения и площадь топологии блока умножения DIFMAS.

При построении схемы умножителя использовались следующие принципы формирования парафазных входов комбинационной схемы (ступени конвейера):

 Входы могут переключаться в рабочую фазу, если схема завершила переключение в спейсер и предыдущая и последующая ступени конвейера разрешили ей переключение в рабочую фазу; входы могут переключаться в спейсер, если схема завершила переключение в рабочую фазу и предыдущая и последующая ступени конвейера разрешили ей переключение в спейсер. Вход разрешения переключения в противоположную фазу работы формируется индикаторами данной, предыдущей и следующей ступеней конвейера.

2. Вход разрешения переключения в противоположную фазу работы является аналогом локального синхросигнала и может быть реализован в виде древовидной структуры типа "дерева тактового сигнала".

3. Формирователем входов является регистр на выходе предшествующей ступени конвейера. Разряд регистра реализуется схемой из двух Г-триггеров и индикаторного элемента (2И-НЕ или 2ИЛИ-НЕ в зависимости от типа спейсера входов).

Выходной регистр всей НЧЗ схемы реализуется двухтактными RS-триггерами с парафазным информационным входом, входом разрешения записи и бифазным выходом для обеспечения интерфейса с синхронным окружением.

С. Входное и выходное FIFO

Входное и выходное FIFO, использованные в предшествующих реализациях блока FMA [7]-[10], обладали одним существенным недостатком – относительно большим энергопотреблением. Это обусловлено их схемотехнической реализацией на основе полуплотного регистра сдвига [11, рис. 11.9]: слово данных, записанное во входную головку регистра, автоматически продвигается к выходной головке регистра до ближайшей не занятой ячейки.

С одной стороны, это обеспечивает строгую последовательность слов данных, записываемых в FIFO и считываемых из него, без дополнительных аппаратурных затрат на механизм адресации текущей выходной ячейки FIFO. С другой стороны, на пути к выходной головке FIFO слово данных вынужденно проходит через все промежуточные ячейки FIFO, вызывая перезаряд их паразитных емкостей, что приводит к дополнительному энергопотреблению.

С целью сокращения энергопотребления блока DIFMAS была предложена новая реализация входного и выходного FIFO, основанная на регистровом файле и указателях активного регистра для операций записи и чтения FIFO. Разряды регистров FIFO реализованы на самосинхронном однотактном триггере с унарным информационным входом, схема которого показана на рис. 4, что отвечает требованиям как синхронного, так и самосинхронного интерфейсов DIFMAS с окружением и упрощает управление FIFO. Здесь *D* – информационный унарный вход; *E* – вход разрешения записи; *I* – индикаторный выход.



Структурная схема FIFO показана на рис. 5. Сигналы WrEn (разрешение записи), Wr (инициация записи) и *Full* (признак заполнения FIFO) обеспечивают взаимодействие с синхронным устройством, формирующим входные операнды (*Op1*, *Op2*, *Op3*) для блока DIFMAS. Выходы *OK* (готовность операндов *X*, *Y*, *Z* на выходе FIFO) и *RdReq* (запрос на чтение следующей тройки операндов из FIFO со стороны ядра DIFMAS) реализуют запрос-ответное взаимодействие FIFO с остальной частью DIFMAS.



Рис. 5. Структурная схема FIFO

Регистровый файл рассчитан на хранение n троек операндов и сопровождающих их признаков операций. Блоки адреса чтения и записи реализованы на НЧЗ счетчиках. Схема управления выполняет арбитраж операций записи и чтения и предупреждает формирователь входных операндов о необходимости приостановки накачки FIFO (сигнал *Full*). Мультиплексор МХ выбирает из регистрового файла тройки операндов в порядке их записи в FIFO с помощью счетчика адреса чтения.

Предложенная реализация FIFO в сравнении с реализацией на основе самосинхронного полуплотного регистра [8] обладает в 2,6 раза меньшим энергопотреблением при практически одинаковых аппаратурных затратах и быстродействии.

D. Индикация DIFMAS

DIFMAS является устройством, обрабатывающим многоразрядные данные и занимающим достаточно большую площадь на кристалле СБИС. В нем реализованы принципы оптимальной индикации на локальном уровне, отработанные в предшествующем варианте блока FMA (SIFPC) [10]. Однако на уровне крупных функциональных блоков, входящих в состав DIFMAS, индикация реализована как минимально необходимая с точки зрения принципов индикации самосинхронных схем и не учитывает ограничений, накладываемых на подсхему индикации размером ЭЗ [11].

Под ЭЗ понимаются фрагменты схемы, компоненты которых функционируют в "одном времени" [11, стр. 10] и разница в задержках сигналов в цепях межсоединений после разветвления не превышает минимальной задержки переключения произвольного элемента библиотеки стандартных элементов, использованной для реализации данной СБИС.

На рис. 6 показан пример разветвления сигнала для случая трех приемников сигнала, формируемого некоторым элементом U0. Задержки распространения сигнала с выхода элемента U0 до входов элементов U1, U2 и U3 ($t_{3д1}$, $t_{3д2}$ и $t_{3д3}$ соответственно) определяются технологией изготовления микросхемы и ее топологической реализацией: длинами отрезков цепи сигнала с выхода элемента U0 до элементов U1, U2 и U3 после точки разветвления сигнала A и их физической реализацией (на каких слоях трассировки проведены соответствующие отрезки трассы межсоединения). Если выполняются одновременно соотношения:

$$\begin{cases} |t_{_{3}\chi_{1}}-t_{_{3}\chi_{2}}| < t_{_{MHH,3}\chi_{*}}, \\ |t_{_{3}\chi_{2}}-t_{_{3}\chi_{3}}| < t_{_{MHH,3}\chi_{*}}, \\ |t_{_{3}\chi_{1}}-t_{_{3}\chi_{3}}| < t_{_{MHH,3}\chi_{*}}, \end{cases}$$
(1)

где $t_{\text{мин.зд}}$ – минимальная задержка переключения выхода любого из элементов U1, U2, U3 относительно соответствующего входа, то схема на рис. 6 считается расположенной целиком в Э3, и индицировать сигнал A можно на входе любого из элементов U1 – U3. В противном случае индицировать сигнал A необходимо на конце отрезка цепи с максимальной задержкой. Поскольку реальные задержки выходного сигнала элемента U0 после точки разветвления сигнала A определяются конкретной топологической реализацией (взаимным расположением элементов, подключенных к одной цепи, и использованными технологическими слоями трассировки), достоверный анализ соотношений (1), во-первых, возможен только после топологической реализации схемы и, во-вторых, необходим после каждой коррекции топологии схемы.



Рис. 6. Пример разветвления сигнала для случая трех приемников

Анализ влияния паразитных емкостей и сопротивлений в стандартной 65-нм КМОП технологии показывает, что использование для проведения трасс межсоединений слоев металла второго и третьего уровней в наихудшем случае приводит к появлению паразитных емкостей с погонным значением 202 фФ/мм, а при проведении той же трассы в слоях металла четвертого и пятого уровней – 198 фФ/мм. С учетом того, что типовая емкость входа стандартного библиотечного элемента не превышает 1,5 фФ, получается, что основной вклад в задержку распространения сигнала в микросхеме, изготовленной по 65-нм КМОП технологии, вносят трассы межсоединений.

Результаты моделирования схемы, показанной на рис. 6, с учетом паразитных параметров, восстановленных из ее топологической реализации, показывают следующее:

1. Для элементов с одинарной нагрузочной способностью разность задержек распространения сигнала по трассам, отличающимся по длине на 60 мкм, составляет около 5 пс, что соответствует задержке переключения одинарного инвертора для данной технологии (t_{мин.зд.} = 5 пс). При этом разность задержек не зависит от типа элемента и сложности выполняемой им функции.

2. Для элементов с повышенной нагрузочной способностью разность задержек распространения сигнала по таким же трассам оказывается даже больше, чем для элементов с одинарной нагрузочной способностью (например, для инвертора с 40-кратной нагрузочной способностью сами задержки сокращаются, но разность между ними оказывается порядка 6 пс).

Таким образом, размер ЭЗ для КМОП технологии с проектными нормами 65 нм не превышает 60 мкм для элементов с одинарной нагрузочной способностью и 50 мкм для элементов с большой нагрузочной способностью. Следует, однако, учесть, что размер ЭЗ ассоциируется с радиусом окружности, центр которой располагается в точке разветвления трассы.

Размер ЭЗ – понятие довольно условное. Элементприемник может находиться достаточно близко от элемента-передатчика, но трасса к нему после точки разветвления может "плутать" по топологии, из-за чего ее длина и соответствующая ей паразитная емкость, определяющая задержку сигнала, будут относительно большими. Поэтому целесообразно говорить об эквихронных трассах (ЭТ), имея в виду их длину (и задержку распространения сигналов по ним) после точки разветвления.

Для обеспечения свойства НЧЗ необходимо соблюдение следующих условий:

1. Парафазные сигналы должны индицироваться на входе их приемника, на конце самой длинной (задержанной) трассы.

2. При наличии трасс, выходящих за пределы подмножества ЭТ данного сигнала, индицироваться должны и сигналы на концах всех этих трасс.

Проверка описанных условий выполняется при анализе схемы на самосинхронность с помощью программы анализа [14] с учетом реальных паразитных параметров трасс, извлеченных из топологической реализации анализируемой схемы.

III. ПАРАМЕТРЫ DIFMAS

DIFMAS был спроектирован в стандартной 65 нм КМОП объемной технологии с 6 слоями металлизации. Параметры DIFMAS в сравнении с синхронным аналогом близкой производительности [15] приведены в таблице. Временные и энергетические параметры получены на основе моделирования без учета паразитных параметров топологической реализации для статистически достоверного набора комбинаций входных операндов двойной и одинарной точности.

Таблица

| Наименование параметра | Аналог | DIFMAS |
|--|---------------------------|---------------------------------|
| Частота работы, ГГц | 1,03 | 1,02 |
| Площадь топологии, мм ² | 0,312 | 0,468 |
| Латентность, нс | 10,8 | 2,94 |
| Производительность, Гфлопс | 2,06 | 3,06 |
| Эффективность площади, мм ² /Гфлопс | 0,151 | 0,153 |
| Диапазон работоспособности по напряжению питания V _{пит} | $V_{\text{пит}}{\pm}10\%$ | $V_{\text{пор}}V_{\text{проб}}$ |
| Обнаружение константных неисправностей | - | + |

Параметры DIFMAS

Быстродействие определялось для типовых условий эксплуатации (1,0 В напряжения питания, 25°С), так как производительность НЧЗ схем всегда соответствует текущим условиям эксплуатации, а сами НЧЗ схемы не требуют учета наихудшего случая для обеспечения работоспособности схемы во всем гарантированном диапазоне изменений напряжения питания и температуры окружающей среды.

Следует отметить, что DIFMAS обладает большей функциональностью по сравнению с аналогом: за один цикл он способен обработать одну тройку операндов двойной точности или две тройки операндов одинарной точности, вычисляя при этом не только сумму, но и разность произведения первых двух операндов и третьего операнда (это учтено в показателе производительности). Кроме того, он имеет намного более широкий диапазон работоспособности, ограниченный лишь пороговыми напряжениями КМОП транзисторов (V_{пор}) и напряжением пробоя полупроводниковых структур (V_{проб}), и прекращает работу при обнаружении константных неисправностей [11]. Платой за эти преимущества является большая сложность реализации и, в связи с этим, большее энергопотребление. Энергопотребление может быть снижено до требуемой величины за счет уменьшения питающего напряжения при соответствующем снижении производительности. За счет меньшего числа ступеней конвейера латентность DIFMAS в 3,7 раз меньше, чем у синхронного аналога.

Таким образом, представленный вариант DIFMAS обеспечивает производительность на уровне 3,06 Гфлопс. Он реализует современный тренд в построении вычислительных средств высокой производительности: использование большего количества процессоров с относительно низкой производительностью.

IV. ЗАКЛЮЧЕНИЕ

DIFMAS с одним блоком умножителя, разработанный по КМОП технологии с проектными нормами 65 нм, демонстрирует высокую среднюю производительность (3,06 Гфлопс при типовых условиях) и хорошую латентность (менее 3 нс).

Использование избыточного самосинхронного кодирования, двухступенчатой реализации умножителя и ускорения переключения умножителя в спейсер обеспечило разработку 64-разрядного вычислителя, реализующего FMA и FMS операции, соответствующего по производительности современным образцам синхронных аналогов и обладающего всеми преимуществами НЧЗ устройств: полной самопроверяемостью относительно константных неисправностей, сохранением работоспособности при сверхмалых значениях питающих напряжений.

Направлением дальнейших исследований является изучение возможности сокращения длительности спейсерной фазы всех ступеней конвейера DIFMAS за счёт разработки локализованной поразрядной системы ускорения перехода элементов блока в спейсерное состояние и рабочую фазу за счёт разработки более быстрого регистра дополнительной внутренней памяти.

Поддержка

Исследование выполнено при частичной поддержке Программы фундаментальных исследований Прези-

диума РАН № 14 "Исследование инновационных методов автоматизации проектирования СБИС и систем на кристалле на 2018-2020 годы" (проект 0063-2018-0004) в Институте проблем информатики ФИЦ ИУ РАН.

ЛИТЕРАТУРА

- [1] R.V.K. Pillai, S.Y.A. Shah, A.J. Al-Khalili, and D. Al-Khalili, Low power floating point MAFs – A comparative study / Sixth International Symposium on Signal Processing and its Applications, Kuala Lumpur, 2001, V. 1.P. 284-287.
- [2] P.-M. Seidel, Multiple path IEEE floating-point Fused Multiply-Add / Proc. 46th IEEE International Midwest Symposium on Circuits and Systems, Cairo, Egypt, 2003.P. 1359–1362.
- [3] T. M. Bruintjes. Design of a Fused Multiply-Add Floating-Point and Integer Datapath. Master's thesis, University of Twente, Enschede, the Netherlands, 2011. 154 p.
- [4] J.R. Noche, and J.C. Araneta, An asynchronous IEEE floating-point arithmetic unit / Science Diliman, Philippines. 2007. V.19. No.2.P. 12–22.
- [5] R. Manohar, and B.R. Sheikh, Operand-optimized asynchronous floating-point units and method of use therefor, US patent, № 20130124592. May 2013.
- [6] Y. Stepchenkov, Y. Diachenko, V. Zakharov, Y. Rogdestvenski, N. Morozov, and D. Stepchenkov, Self-Timed Computing Device for High-Reliable Applications / Proc. International Workshop on power and timing modeling, optimization and simulation (PATMOS'2009), Delft, Netherlands, 2009.P. 276–285.
- [7] Соколов И.А., Степченков Ю.А., Рождественский Ю.В., Дьяченко Ю.Г. Самосинхронное устройство умножения-сложения гигафлопсного класса: методологические аспекты // Проблемы разработки перспективных микро- и наноэлектронных систем -2014. Сб. трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2014. Ч. IV. С. 51-56.
- [8] Степченков Ю.А., Рождественский Ю.В., Дьяченко Ю.Г., Морозов Н.В., Степченков Д.Ю., Сурков А.В. Самосинхронное устройство умножения-сложения гигафлопсного класса: варианты реализации //

Проблемы разработки перспективных микро- и наноэлектронных систем - 2014. Сб. трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМРАН, 2014. Ч. IV. С. 57-60.

- [9] Yuri Stepchenkov, Victor Zakharov, Yuri Rogdestvenski, Yuri Diachenko, Nikolai Morozov and Dmitri Stepchenkov. Speed-Independent Fused Multiply Add and Subtract Unit // Proceedings of IEEE EastWest Design & Test Symposium (EWDTS'2016), Yerevan, October, 14 - 17, 2016. P. 150-153.
- [10] Ю.А. Степченков, Ю.В. Рождественский, Ю.Г. Дьяченко, Н.В. Морозов, Д.Ю. Степченков, Б.А. Степанов, Д.Ю. Дьяченко, А.В. Рождественскене. Самосинхронное устройство умножения-сложения с плавающей точкой // Проблемы разработки перспективных микро- и наноэлектронных систем -2016. Сб. трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМРАН, 2016. Часть 3. С. 149-156.
- [11] Варшавский В.И. и др. Автоматное управление асинхронными процессами в ЭВМ и дискретных системах. М.: Наука, 1986. 400 с.
- [12] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "An 8.8-ns 54x54-bit multiplier with high speed redundant binary architecture" // IEEE Journal of Solid-State Circuits.1996. V. 31. No. 6, pp. 773-783.
- [13] Stepchenkov Y.A., Zakharov V.N., Rogdestvenski Y.V., Diachenko Y.G., Morozov N.V., Stepchenkov D.Y. Speed-Independent Floating Point Coprocessor / IEEE Eeast-West Design and Test Symposium, Batumi, Georgia, September 26-29, 2015. P. 111- 114.
- [14] Рождественский Ю.В., Морозов Н.В., Рождественскене А.В. Подсистема событийного анализа самосинхронных схем АСПЕКТ // Проблемы разработки перспективных микро- и наноэлектронных систем - 2010. Сб. трудов / под общ. ред. академика А.Л.Стемпковского. М.:ИППМ РАН, 2010. С. 26-31.
- [15] S. Galal, and M. Horowitz, Energy-Efficient Floating-Point Unit Design // IEEE Transactions on computers. 2011. V. 60. No.7. P. 913–922.

Delay-Insensitive Floating Point Multiply-Add-Subtract Unit

I.A. Sokolov, Y.V. Rogdestvenski, Y.G. Diachenko, Y.A. Stepchenkov, N.V. Morozov, D.Y. Stepchenkov, D.Y. Diachenko

Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences (IPI FRC CSC RAS), IPI RAS

{YRogdest, YDiachenko, YStepchenkov, NMorozov, DStepchenkov}@ipiran.ru

Abstract — The subject of this paper is a floating point unit implementing fused multiply-add-subtract operation. It belongs to the delay-insensitive self-timed circuits which do not depend on delays both in cells and on wires. It is fully compliant with IEEE 754 Standard and processes both a sum and difference between product of first two operands and third operand. Each 64-bit input operand contains either one double precision number, or two single precision numbers. Thus presented unit calculates either one operation with double precision numbers, or two simultaneous operations with single precision numbers. Multiplier utilizes modified Booth algorithm. In order to increase its performance, it is divided into two pipeline stages with accelerated forced switching to a spacer phase. Booth encoder circuit is integrated into an input FIFO. FIFO is implemented as a register file with an output multiplexer and read/write address counters. Using ternary redundant self-timed code for multiplying, adding and subtracting provides a reduction of unit's complexity. Indication subcircuit considers the constrains imposed by an equichronous zone for chosen fabrication technology. For decreasing energy consumption, the fused multiply-add-subtract unit implements one-channel pipeline. The unit is designed for 65-nm CMOS bulk technology using an industrial standard cell library supplemented by selftimed cells. It provides 3 Gflops performance and 2.9-ns latency.

Keywords — redundant coding, ternary adder, Wallace tree, equichronous zone, FIFO.

REFERENCES

- R.V.K. Pillai, S.Y.A. Shah, A.J. Al-Khalili, and D. Al-Khalili, Low power floating point MAFs – A comparative study / Sixth International Symposium on Signal Processing and its Applications, Kuala Lumpur, 2001, V. 1. P. 284-287.
- [2] P.-M. Seidel, Multiple path IEEE floating-point Fused Multiply-Add / Proc. 46th IEEE International Midwest Symposium on Circuits and Systems, Cairo, Egypt, 2003. P. 1359–1362.
- [3] T. M. Bruintjes. Design of a Fused Multiply-Add Floating-Point and Integer Datapath. Master's thesis, University of Twente, Enschede, the Netherlands,2011. 154 p.
- [4] J.R. Noche, and J.C. Araneta, An asynchronous IEEE floating-point arithmetic unit / Science Diliman, Philippines. 2007. V.19. No.2.P. 12–22.
- [5] R. Manohar, and B.R. Sheikh, Operand-optimized asynchronous floating-point units and method of use therefor, US patent, № 20130124592. May 2013.
- [6] Y. Stepchenkov, Y. Diachenko, V. Zakharov, Y. Rogdestvenski, N. Morozov, and D. Stepchenkov, Self-Timed Computing Device for High-Reliable Applications / Proc. International Workshop on power and timing modeling, optimization and simulation (PATMOS'2009), Delft, Netherlands, 2009.P. 276–285.
- [7] Sokolov I.A., Stepchenkov Yu.A., Rozhdestvenskij Yu.V., Diachenko Yu.G. Samosinhronnoe ustroystvo umnojeniyaslojeniya gigaflopsnogo klassa: metodologicheskie aspektyi (Speed-Independent Fused Multiply-Add Unit of Gigaflops Rating: Methodological Aspects) // Sb. trudov "Problemyi razrabotki perspektivnyih mikro- i nanoelektronnyih sistem". M.: IPPM RAN, 2014. Ch. IV. S. 51-56.

- [8] Stepchenkov Yu.A., Rozhdestvenskij Yu.V., Diachenko Yu.G., Morozov N.V., Stepchenkov D.Yu., Surkov A.V. Samosinhronnoe ustroystvo umnojeniya-slojeniya gigaflopsnogo klassa: variantyi realizatsii (Speed-Independent Fused Multiply-Add Unit of Gigaflops Rating: Implementation Variants) // Sb. trudov "Problemyi razrabotki perspektivnyih mikro- i nanoelektronnyih sistem". M.: IPPM RAN, 2014. Ch. IV. S. 57-60.
- [9] Yuri Stepchenkov, Victor Zakharov, Yuri Rogdestvenski, Yuri Diachenko, Nikolai Morozov and Dmitri Stepchenkov. Speed-Independent Fused Multiply Add and Subtract Unit // Proceedings of IEEE EastWest Design & Test Symposium (EWDTS'2016), Yerevan, October, 14 - 17, 2016. P. 150-153.
- [10] Stepchenkov Yu.A., Rozhdestvenskij Yu.V., DiachenkoYu.G., Morozov N.V., Stepchenkov D.Yu., Stepanov B.A., Diachenko D.Y., Rozhdestvenskene A.V. Samosinhronnoe ustroystvo umnojeniya-slojeniya s plavayuschey tochkoy (Self-Timed Floating Point Multiply-Add Unit) // Sb. trudov "Problemyi razrabotki perspektivnyih mikro- i nanoelektronnyih sistem". M.: IPPM RAN, 2016. Ch 3. S. 149-156.
- [11] Varshvskij V.I. i dr. Avtomatnoe upravlenie asinhronnyimi protsessami v EVM i diskretnyih sistemah (Automatic control of the asynchronic processes in the computers and discrete systems). M.: Nauka, 1986. 400 s.
- [12] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "An 8.8-ns 54x54-bit multiplier with high speed redundant binary architecture" // IEEE Journal of Solid-State Circuits.1996. V. 31. No. 6, pp. 773-783.
- [13] Stepchenkov Y.A., Zakharov V.N., Rogdestvenski Y.V., Diachenko Y.G., Morozov N.V., Stepchenkov D.Y. Speed-Independent Floating Point Coprocessor / IEEE Eeast-West Design and Test Symposium, Batumi, Georgia, September 26-29, 2015. P. 111- 114.
- [14] Rozhdestvenskij Yu.V., Morozov N.V., Rozhdestvenskene A.V. Podsistema sobyitiynogo analiza samosinhronnyih shem ASPEKT (ASPECT – a Subsystem of Event Analysis of Self-Timed Circuits) // Sb. trudov "Problemyi razrabotki perspektivnyih mikro- i nanoelektronnyih sistem". M., IPPM RAN, 2010. S. 26-31.
- [15] S. Galal, and M. Horowitz, Energy-Efficient Floating-Point Unit Design // IEEE Transactions on computers. 2011. V. 60. No.7. P. 913–922.

Сложнофункциональный блок сетевого коммутатора Ethernet для радиорелейной системы связи

Р.С. Кобяков^{1,2}, А.А. Шевченко¹, М.В. Махлышев¹, Р.О. Масленников¹

¹ООО «Радио Гигабит», г. Нижний Новгород

²Нижегородский государственный университет им. Н.И. Лобачевского, г. Нижний Новгород,

roman.kobyakov@radiogigabit.com

Аннотация — В статье рассматривается разработка сложнофункционального блока трехпортового сетевого коммутатора Ethernet для реализации на ПЛИС. Коммутатор предназначен для внутриканального управления радиорелейной станцией и объединения внешнего сетевого интерфейса, интерфейса данных радиомодема и сетевого интерфейса центрального процессора. Блок коммутатора поддерживает скорости передачи данных 10/100/1000 Мбит/с, коммутацию Ethernet кадров на канальном уровне на основе МАС адресов, технологии виртуальных сетей (VLAN) - IEEE 802.1Q и port-based. При разработке коммутатора учтены особенности архитектуры РРС и характеристики передаваемых данных с целью минимизации объема используемых ресурсов ПЛИС. Представлено описание требований, архитектуры и особенностей аппаратной реализации коммутатора, а также схемы его интеграции в состав «системы-накристалле». Для разработанного сложнофункционального блока приведены характеристики использования аппаратных ресурсов ПЛИС и результаты тестирования.

Ключевые слова — СФ-блок, сетевой коммутатор Ethernet, ПЛИС, система на кристалле, радиорелейная система связи.

I. Введение

Радиорелейные системы связи широко применяются в существующих и перспективных сетях передачи данных. При этом реализация цифровой части радиорелейной станции (РРС), как правило, выполняется на программируемых логических интегральных схемах (ПЛИС) и использует архитектуру «системы-накристалле» (СнК), объединяющую центральный процессор (ЦПУ) и различные периферийные аппаратные блоки. Основным таким блоком является цифровой радиомодем, обеспечивающий широкополосную передачу и прием данных в радиоканале. В качестве внешнего сетевого интерфейса в РРС наиболее часто применяется интерфейс сети Ethernet 0, поддерживающий передачу данных со скоростью до 1 Гбит/с. Поэтому в состав СнК, как правило, включается аппаратный блок Ethernet MAC (Medium Access Control).

Управление РРС производится (аналогично другому сетевому оборудованию) удаленно с применением специализированных приложений, выполняемых на ЦПУ станции и использующих различные сетевые протоколы, такие как SNMP, Telnet и др. При этом возникает необходимость в отдельном управляющем сетевом интерфейсе для ЦПУ, который также должен быть подключен к физической среде Ethernet. В РРС возможны два варианта организации такого доступа процессора к Ethernet сети. Первый вариант подразумевает использование отдельного физического интерфейса. В этом случае трафик данных и служебный трафик передаются по двум физически независимым каналам. При этом управляющий интерфейс называется внеканальным (outband).

Второй вариант использует единый физический интерфейс Ethernet для передачи данных и управления РРС. При этом оба типа трафика передаются по одному каналу, но так как данные управляющих приложений ЦПУ занимают малую часть от общей полосы пропускания Ethernet интерфейса, то влияние на скорость передачи данных оказывается несущественным. Такой подход к управлению РРС называется внутриканальным (inband). Преимуществом внутриканального подхода является отсутствие необходимости организации второго физического интерфейса Ethernet. Однако реализация внутриканального управления требует использования сетевого коммутатора в составе РРС. Коммутатор производит распределение Ethernet кадров между интерфейсом радио, внешним сетевым интерфейсом и ЦПУ.

Ethernet коммутатор может быть реализован путем использования специализированной микросхемы, что имеет такие недостатки как увеличение стоимости системы и необходимость использования двух физических Ethernet интерфейсов между ПЛИС и микросхемой коммутатора. Альтернативой является использование аппаратного СФ-блока коммутатора в составе СнК ПЛИС. При этом применение интегрированного коммутатора в составе СнК РРС может значительно увеличить общий объем занимаемых аппаратных ресурсов. Уменьшение объема используемых ресурсов возможно путем оптимизации архитектуры коммутатора с учетом специальных требований РРС и особенностей передаваемых данных. Разработка такого Ethernet коммутатора для использования в составе СнК РРС и оптимизированного для минимизации потребляемых аппаратных ресурсов рассматривается в данной статье.

II. ТРЕБОВАНИЯ К СЕТЕВОМУ КОММУТАТОРУ РРС

На рис.1 представлена общая схема сетевой части РРС, включающая сетевой коммутатор Ethernet.



Рис. 1. Общая схема сетевого интерфейса РРС

Радиомодем производит передачу и прием кадров данных через беспроводной канал связи. Внешний Ethernet интерфейс обеспечивает доступ к физическому уровню сети Ethernet. ЦПУ осуществляет контроль станции и взаимодействие с удаленными системами управлениям. Сетевой коммутатор производит распределение Ethernet кадров между интерфейсами трех указанных блоков, подключенных к его портам. Передача данных между коммутатором и остальными блоками выполняется по шине AMBA AXI4-Stream 0.

К сетевому коммутатору РРС предъявляются следующие требования. Во-первых, коммутацию Ethernet кадров необходимо выполнять между тремя портами согласно технологии «прозрачного моста» на основе МАС адресов [3]. Во-вторых, максимальная скорость приема и передачи данных для каждого порта должна составлять 1 Гбит/с, а также должны поддерживаться скорости 10 Мбит/с и 100 Мбит/с. Скорости работы разных портов могут быть разные. Также необходима поддержка механизма контроля потока данных (Ethernet Flow Control) 0 для обеспечения передачи данных без потери кадров в том случае, если пропускная способность исходящего канала ниже, чем скорость поступления входящих данных на соответствующем порту. Дополнительно требуется поддержка виртуальных сетей (VLAN) согласно протоколу IEEE 802.1Q 0 и коммутация Ethernet кадров на основе VLAN тегов с возможностью их добавления и удаления «на лету» в процессе передачи данных. Кроме того, необходима возможность программного определения логических связей между портами коммутатора по технологии порт-ориентированных виртуальных сетей (port-based VLAN).

У разрабатываемого коммутатора, используемого в составе СнК РРС, есть следующие особенности, которые могут быть учтены при оптимизации его аппаратной архитектуры. Во-первых, основной объем данных передается между радиомодемом и внешним интерфейсом Ethernet. Кадры внутриканального управления составляют малую часть от общего объема трафика. Также к порту ЦПУ подключено только одно сетевое устройство, использующее сообщения размера, не превышающего стандартный максимальный Ethernet кадр в 1518 байт. При этом априорное знание о единственном абоненте, подключенном к ЦПУ порту, допускает уменьшение размера коммутационной таблицы МАС, по сравнению с общим случаем трехпортового Ethernet коммутатора. Кроме того, приемник радиомодема имеет в своем составе буфер данных, в котором хранятся Ethernet кадры, что также может быть учтено при разработке коммутатора.

III. АППАРАТНАЯ АРХИТЕКТУРА СЕТЕВОГО КОММУТАТОРА

А. Общая архитектура коммутатора

Аппаратная архитектура сетевого коммутатора представлена на рис. 2. В состав сетевого коммутатора входят три входных порта, три выходных порта, коммутационная таблица, матрица межсоединений и регистровый файл.



Рис. 2. Аппаратная архитектура сетевого коммутатора

Каждый из входных портов выполняет прием Ethernet кадров, адаптивную буферизацию данных, распознавание Ethernet заголовков и извлечение контрольной информации, необходимой для работы коммутационной таблицы. Кроме того, входной порт осуществляет поддержку функции контроля потока данных. Этот механизм обеспечивает защиту от переполнения буфера входного порта путем формирования запросов об остановке и возобновлении входящего потока данных, либо временно прекращая прием кадров, если остановка трафика невозможна. Для порта, подключенного к внешнему интерфейсу Ethernet, запросы на остановку и возобновление реализуются согласно Ethernet Flow Control протоколу 0. Если сетевое устройство, формирующее поток данных на внешний интерфейс Ethernet, поддерживает данный протокол, то прием кадров на данном порту выполняется без потерь.

Коммутационная таблица определяет номера портов, на которые должен быть отправлен принятый Ethernet кадр. Вычисление номера порта выполняется по алгоритму прозрачного моста 0 с учетом VLAN конфигурации 0 и логических связей между входными и выходными портами. При определении выходных портов учитываются MAC адрес получателя, идентификатор VLAN группы, а также номер входного порта, принявшего кадр.

Матрица межсоединений выполняет коммутацию (соединение) входных и выходных портов на основе решений коммутационной таблицы для передачи Ethernet кадров.

Выходной порт производит считывание Ethernet кадра из буфера входного порта согласно решению коммутационной таблицы и осуществляет контроль передачи Ethernet кадра на подключенный к нему сетевой интерфейс. В процессе передачи кадра может выполняться удаление, перезапись или добавление VLAN тега (согласно конфигурации порта), а также производиться вычисление и замена контрольной суммы кадра, если данные в заголовке кадра при прохождении через коммутатор были изменены.

Управление параметрами сетевого коммутатора выполняется с помощью регистрового файла. Чтение и запись регистров производится ЦПУ по шине данных, использующей интерфейс AMBA AXI4-Lite 0. Через регистровый файл определяются общие настройки коммутатора, структура логических связей между портами (port-based VLAN), конфигурация VLAN IEEE 802.1Q, а также предоставляется доступ к статистике передачи данных через каждый порт.

В. Архитектура входного порта

На рис. 3 представлена аппаратная архитектура входного порта сетевого коммутатора. Обработку Ethernet кадров осуществляют следующие блоки входного порта коммутатора: контроллер записи, буфер данных, контроллер принятых кадров, анализатор заголовка и контроллер VLAN.

Разработанный коммутатор использует гибридную схему буферизации Ethernet кадров на входных портах. Данный алгоритм производит запись в память заголовка Ethernet кадра для анализа контрольной информации необходимой для работы коммутационной таблицы. После этого производится обращение к коммутационной таблице, которая определяет, на какие выходные порты должен быть передан кадр, и информирует каждый соответствующий выходной порт о наличии кадра для передачи. Если данный выходной порт свободен, то отправка данных начинается, не дожидаясь завершения приема кадра входным портом (частичная буферизация). Процедуры записи оставшейся части кадра и чтения уже принятых данных будут выполняться одновременно. В этом случае в памяти хранится только небольшая часть кадра. Запись всего кадра в память порта (полная буферизация) выполняется, только если выходной порт занят, либо скорость приема кадра ниже скорости его передачи выходным портом.



Рис. 3. Аппаратная архитектура входного порта

Контроллер записи управляет процессом размещения входящих Ethernet кадров в буфере данных, а также выполняет его защиту от переполнения.

Буфер данных является блоком памяти, используемым для хранения принятых Ethernet кадров, и поддерживает функцию множественного доступа на чтение содержимого выходными портами. Размеры буфера данных выбраны для каждого входного порта согласно особенностям, рассмотренным в разделе II. Буфер порта ЦПУ использует 4096 байт и позволяет одновременно располагать в нем два максимальных кадра размера 1518 байт так как ЦПУ не использует Ethernet кадры большего размера. Размер буфера данных входящего порта внешнего интерфейса Ethernet составляет 32768 байт, что позволяет хранить до трех кадров размера 9600 байт одновременно. 9600 байт является максимальным размером Ethernet кадра, поддерживаемым коммутатором. Входной порт радио задействует 16384 байт под хранение данных. Размер буфера выбран таким образом, чтобы полностью записывать один кадр максимального размера 9600 байт, передаваемый по беспроводному каналу. Получение кадров, хранящихся в буфере данных радиомодема управляется входным портом с помощью механизма контроля потока. Если оценка свободного пространства буфера радио порта меньше 9600 байт, то прием новых кадров временно прекращается до освобождения необходимого объема. Это позволяет задействовать память радиомодема, последовательно подгружая для выполнения коммутации кадры, принятые по беспроводному каналу.

Контроллер принятых кадров выполняет учет Ethernet кадров с определением и отслеживанием их текущего статуса, которое может принимать три состояния: принят, ожидает отправки, отправлен. Анализатор заголовка производит распознавание и извлечение из каждого кадра контрольной информации необходимой коммутационной таблице: MAC адреса получателя, MAC адреса отправителя, наличия VLAN тега согласно протоколу IEEE 802.1Q, номера VLAN группы. Контроллер VLAN выполняет обработку кадра согласно настройкам VLAN данного входного порта. Модулем принимается решение о дальнейшей передаче или фильтрации (отбрасывании) кадра в зависимости от режима работы порта и наличия VLAN тега.

С. Архитектура коммутационной таблицы

На рис. 4 представлена аппаратная архитектура коммутационной таблицы. Основными компонентами блока являются контроллер MAC адресов, таблица MAC адресов, контроллер VLAN групп, VLAN таблица и порт-ориентированный VLAN контроллер.



Рис. 4. Аппаратная архитектура коммутационной таблицы

В работе коммутационной таблицы используется алгоритм «прозрачного моста», определенный в стандарте IEEE 802.1D 0. Также на процедуру коммутации накладываются ограничения VLAN в соответствии со спецификацией IEEE 802.1Q 0 и логическими связями между портами коммутатора. Для всех Ethernet кадров, которые не были отфильтрованы входными портами, коммутационной таблицей определяется номер или последовательность номеров выходных портов, через который текущий кадр будет передан далее. Результатом работы является коммутационный вектор *EgressPortVec*. Размерность данного вектора и всех рассматриваемых далее векторов равна трем (числу портов коммутатора), а каждый бит, принимая значения «0» или «1», определяет должен ли Ethernet кадр быть отправлен на порт с номером, равным его индексу.

Контроллер МАС адресов определяет порт назначения кадра согласно МАС адресу получателя. Результатом является вектор *ATVec* либо содержащий одну «1» и все остальные нули, если данный адрес не является широковещательным и был найден в таблице МАС адресов, либо все «1» в противном случае.

Таблица МАС адресов содержит соответствие между МАС адресом и номером порта, к которому подключено сетевое устройство с данным адресом. Размер таблицы МАС адресов составляет 2048 записей, что является достаточным для РРС с учетом априорного знания о единственном устройстве порта ЦПУ. Вся таблица разбита на 256 подтаблиц размера 8. Поиск в таблице производится комбинированным способом в два этапа. На первом этапе с помощью хеш-функции по МАС адресу вычисляется номер подтаблицы. На втором этапе в выбранной подтаблице производится линейный поиск МАС адреса.

Заполнение таблицы может производиться как аппаратным способом (путем обучения по коммутируемого Ethernet кадрам), так и через программный интерфейс регистрового файла. При аппаратном обучении выполняется поиск записи, соответствующей MAC адресу отправителя кадра. Если результат поиска отрицательный, то данный MAC адрес заносится в таблицу, а в соответствие ему ставится порт, который принял кадр.

Актуальность содержимого таблицы поддерживается на аппаратном уровне путем удаления неактивных MAC адресов, если соответствующие сетевые устройства в течение заданного периода не передавали или не принимали Ethernet кадры. Период «устаревания» таблицы MAC адресов определяется через регистровый файл и может варьироваться в диапазоне от нескольких секунд до десяти минут.

Контроллер VLAN группы производит определение подмножества портов, на которые может быть отправлен кадр с данным номером VLAN группы. Результатом является вектор *VlanVec*, где для портов, входящих в VLAN группу, соответствующие биты равны «1».

VLAN таблица определяет принадлежность портов коммутатора к VLAN группам. Размер таблицы равен максимальному числу VLAN групп и содержит 4095 элементов. Заполнение VLAN таблицы выполняется через программный интерфейс регистрового простран-

ства, оставляя за ЦПУ задачу конфигурации виртуальных сетей.

Порт-ориентированный VLAN контроллер определяет подмножество выходных портов, на которые с данного входного порта может быть передан кадр с учетом заданных логических ограничений (port-based VLAN). Результатом является вектор *InPortVec*, где ненулевые биты определяют номера выходных портов, связанных с данным входным портом.

Представленные выше блоки параллельно обрабатывают контрольную информацию Ethernet кадра, и результирующий коммутационный вектор *EgressPortVec* вычисляется операцией побитового умножения полученных векторов:

EgressVec = ATVec & VlanVec & InPortVec

Вычисленный таким образом вектор передается на все выходные порты, а также на входной порт, принявший кадр, для информирования о том, какие порты будут выполнять чтение данных. Если сетевое устройство с MAC адресом получателя подключено к порту, с которого кадр поступил на коммутатор, то *EgressPortVec* вектор будет содержать все нули, и кадр будет отфильтрован по окончанию приема.

D. Архитектура выходного порта

Общая схема выходного порта коммутатора представлена на рис. 5. Блок выходного порта состоит из контроллера отправки кадра, контроллера VLAN и блока пересчета CRC.



Рис. 5. Аппаратная архитектура выходного порта

Управление передачей данных выполняется контроллером отправки кадра. Модуль производит чтение принятых Ethernet кадров из буфера данных входных портов и выполняет отправку по мере готовности подключенного сетевого интерфейса. С целью экономии аппаратных ресурсов выходной порт не имеет собственной памяти для хранения кадров. Удаление или изменение VLAN тега производится контроллером VLAN, если это требуется для данного кадра согласно VLAN правилам задействованных портов. Блок пересчета CRC выполняет вычисление и замену в передаваемых данных контрольной суммы Ethernet кадра.

IV. РЕЗУЛЬТАТЫ СИНТЕЗА И ТЕСТИРОВАНИЯ

Аппаратная реализация СФ-блока сетевого коммутатора выполнена с использованием языка описания цифровой аппаратуры Verilog HDL. Разработанный блок не использует какие-либо особенности и встроенные функции конкретной аппаратной платформы и может быть применен в СнК, реализуемых на различных технологиях ПЛИС и заказных интегральных схем.

Отладка и тестирование СФ-блока производилось на ПЛИС XC7Z020T семейства Zynq-7000 компании Xilinx, включающей программируемую логику с низким энергопотреблением и двухъядерный процессор ARM Cortex-A9 со встроенными периферийными модулями, такими как интерфейсы UART, Gigabit Ethernet MAC, контроллер памяти DDR и др. Наличие встроенного процессора с широким набором интерфейсных блоков, не использующих ресурсы ПЛИС, в сочетании с высокопроизводительной программируемой логикой позволяет рассматривать данное семейство микросхем в качестве целевой аппаратной платформы для радиорелейных систем связи. В табл. 1 представлены результаты синтеза рассматриваемого СФ-блока. Также для сравнения представлена оценка требуемых аппаратных ресурсов для СФ-блока коммутатора компании COMCORES аналогичной конфигурации. Результаты синтеза получены для ПЛИС Xilinx 7VX690TFFG1761-2 для корректности сравнения рассматриваемых блоков. Представленные результаты демонстрируют существенный выигрыш рассматриваемого СФ-блока сетевого коммутатора.

Таблица 1

| Т | Использовано, шт. (%) | | |
|--------------------------------|----------------------------|---------------------|--|
| і ин элементов | Рассматриваемый СФ-блок | СФ-блок COMCORES | |
| Регистры (Flip Flops) | 5202 (0.006) | 15391(1.78) | |
| Таблицы истинности (LUT) | 2588 (0.006) | 22177(5.12) | |
| Блоки памя- ти (BRAM) | 21 (0.01) | 61.5(4.18) | |

Оценка использования аппаратных ресурсов ПЛИС Xilinx 7VX690TFFG1761-2

Тестирование разработанного блока производилось в составе РРС, использующей указанную ПЛИС. Передача данных в радиоканале выполнялась в дуплексном режиме со скоростью 1 Гбит/с в каждом направлении, дополнительно выполнялся мониторинг состояния радио соединения на основе SNMP сообщений, отправляемых ЦПУ РРС. Также производилось тестирование радиорелейного соединения согласно спецификации RFC 2544 0 сетевым анализатором Метротек Беркут-ЕТ. Полученные результаты демонстрируют 100% передачу Gigabit Ethernet трафика и отсутствия потери пакетов в процессе тестирования на всех длинах Ethernet кадров. Кроме того было проведено успешное тестирование реализованной в коммутаторе поддержки технологии виртуальных локальных сетей (IEEE 802.1Q и port-based).

V. Заключение

В работе представлено описание аппаратной архитектуры и особенностей реализации СФ-блока сетевого коммутатора для радиорелейной системы связи (РРС). Разработанный блок является частью системы на кристалле PPC и обеспечивает коммутацию Ethernet кадров между встроенным ЦПУ, внешним Ethernet интерфейсом и цифровым радиомодемом на скорости до 1 Гбит/с. Адаптация архитектуры коммутатора под особенности сетевого интерфейса РРС и тип предаваемого трафика позволяет обеспечивать высокую производительность и необходимый сетевой функционал при относительно малом объеме используемых аппаратных ресурсов в сравнении с аналогичными решениями других производителей. Представленные результаты интеграции и тестирования в составе прототипа коммерческой РРС демонстрируют эффективность и промышленную применимость данного СФблока.

ЛИТЕРАТУРА

- IEEE Standard for Ethernet," in IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008), vol., no., pp.1-3747, Dec. 28 2012
- [2] ARM AMBA Specification url: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.d oc.set.amba/index.html (дата обращения: 02.04.2018).
- [3] IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges," in IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998), vol., no., pp.1-281, June 9 2004
- [4] IEEE Standards for Local and Metropolitan Area Networks: Supplements to Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Specification for 802.3 Full Duplex Operation and Physical Layer Specification for 100 Mb/s Operation on Two Pairs of Category 3 Or Better Balanced Twisted Pair Cable (100BASE-T2)," in IEEE Std 802.3x-1997 and IEEE Std 802.3y-1997 (Supplement to ISO/IEC 8802-3: 1996; ANSI/IEEE Std 802.3, 1996 Edition), vol., no., pp.0_1-324, 1997
- [5] IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks," in IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011), vol., no., pp.1-1832, Dec. 19 2014
- [6] Bradner, S. 'Benchmarking terminology for net-work interconnection devices', Internet Engineering Task Force, Network Working Group, March 1999 url: https://tools.ietf.org/html/rfc2544 (дата обраще-ния: 02.04.2018

Ethernet Switch IP-Core for Wireless Backhaul Systems

R.S. Kobyakov^{1,2}, A.A. Shevchenko¹, M.V. Makhlyshev¹, R.O. Maslennikov¹

¹Radio Gigabit, Nizhny Novgorod

²Nizhny Novgorod State University N.I. Lobachevsky, Nizhny Novgorod,

roman.kobyakov@radiogigabit.com

Abstract — This article presents design of a 3-port Ethernet switch IP-core for a point-to-point wireless backhaul system. The IP-core is dedicated for implementation of in-band management via switching of Ethernet frames between the external Ethernet interface, the data interface of the radio modem and the network interface of the CPU. The switch supports 10/100/1000 Mbps data rates and routes Layer-2 frames based on their Ethernet MAC addresses, virtual network identifiers (IEEE 802.1Q VLAN ID) and logical interconnections between Ethernet MAC interfaces (port-based VLANs). The switch architecture and FPGA hardware implementation features are presented.

The switch architecture is optimized for specific requirements of the in-band management application and characteristics of the transmitted traffic. The following main requirements are taken into account:

• Switching of the Ethernet frames between three ports based on MAC addresses 0.

- The major part of traffic is transmitted between the Ethernet interface and the radio modem. Therefore, frame commutation is based on a hybrid scheme that combines a "cut-through" approach for most frames and full buffering in the cases of a flooding or a target output port occupation by another transmission.
- The maximum input data rate for each port is 1 Gbps with independent support on different ports of 10 and 100 Mbps.
- The receiver of the radio modem includes a buffer for input data therefore the size of ingress port buffers can be decreased up to several maximum frame lengths.
- Ethernet Flow Control standard 0 is supported for lossless data transmission in case of exceeding the egresses port maximum bandwidth that protects input buffers against an overflow and received frames damaging.
- The network switch topology can include different virtual LANs and frames are routed between ingress and egress ports via VLAN header information according to

IEEE 802.1Q specification 0 and software-defined rules for each port and VLAN ID.

• Port-Based VLAN determines a logical interconnection between each switch port pair and allows or denies data transferring in each direction.

The switch IP core was prototyped and verified using the Xilinx XC7Z020T FPGA device. The resource utilization estimates are the following:

- Flip-Flop count: 5202 (4.89%)
- LUT count: 2588 (4.86%)
- BRAM count: 21 (12.29%)

Presented results that demonstrate efficient hardware resource utilization allow to consider as appropriate for practical implementation.

Keywords — IP-core, Ethernet switch, FPGA, system-on-a-chip, wireless backhaul.

REFERENCES

 IEEE Standard for Ethernet," in IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008), vol., no., pp.1-3747, Dec. 28 2012

- [2] ARM AMBA Specification url: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.d oc.set.amba/index.html (accessed: 02.04.2018).
- [3] IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges," in IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998), vol., no., pp.1-281, June 9 2004
- [4] IEEE Standards for Local and Metropolitan Area Networks: Supplements to Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Specification for 802.3 Full Duplex Operation and Physical Layer Specification for 100 Mb/s Operation on Two Pairs of Category 3 Or Better Balanced Twisted Pair Cable (100BASE-T2)," in IEEE Std 802.3x-1997 and IEEE Std 802.3y-1997 (Supplement to ISO/IEC 8802-3: 1996; ANSI/IEEE Std 802.3, 1996 Edition), vol., no., pp.0_1-324, 1997
- [5] IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks," in IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011), vol., no., pp.1-1832, Dec. 19 2014
- [6] Bradner, S. 'Benchmarking terminology for net-work interconnection devices', Internet Engineering Task Force, Network Working Group, March 1999 url: https://tools.ietf.org/html/rfc2544 (accessed: 02.04.2018)

ИМЕННОЙ УКАЗАТЕЛЬ АВТОРОВ СТАТЕЙ

-A-

Алексан П.А. – 115 Андрианов А.В. – 79 Антонюк А.В. – 109 Аунг Мьо Сан – 149

– Б –
 Барских М.Е. – 86
 Буякова О.Н. – 115

– В –
Вейков А.А. – 162
Власов А.И. – 103
Воронков Д.И. – 162

Г –
Гаращенко А.В. – 9
Гревцев Н.А. – 52
Григорьев П.В. – 103

– Д – Дьяченко Д.Ю. – 170 Дьяченко Ю.Г. – 170

– **E** – Евдокимов А.П. – 23 Егоров И.В. – 136

– **Ж** – Жмылев В.А. – 144

– **3** – Задябин С.О. – 115

— **И** — Иванников А.Д. — 46

- K -

Камкин А.С. – 2 Кобыляцкий А.В. – 72 Кобяков Р.С. – 178

– Л – Ладнушкин М.С. – 64

Ларионов А.В. – 115 Ливенцев Е.В. – 156 Лукьяненко Е.Б. – 130 – M – Масленников Р.О. – 178 Мастеров В.В. – 115 Махлышев М.В. – 178 Меликов А.В. – 23 Морозов Н.В. – 170 Мосин С.Г. – 59 – H – Николаев А.В. - 9 -0-Осина С.Э. – 115 $-\Pi -$ Переверзев А.Л. – 156 Примаков Е.В. – 156 Проценко А.С. – 2 Путря Ф.М. – 9 – P – Рогаткин Ю.Б. – 115 Рождественский Ю.В. – 170 Руткевич А.В. – 162 Рыжкова Д.В. – 156 Рябцев В.Г. – 23 – C – Сардарян С.С. – 9 Сергеев Д.К. – 72 Силантьев А.М. – 156 Смирнов А.В. – 31 Смолов C.A. - 2Соколов И.А. – 170 Сохацкий А.А. 16 Старых А.А. – 130 Степанов П.В. – 109 Степченков Д.Ю. – 170 Степченков Ю.А. – 170 Сысоев И.Ю. – 162 Сысоева О.В. – 115 – T –

Тарасов И.В. – 115 Татарников А.Д. – 2 Татарников Ю.А. – 99 Тихомиров М.В. – 39 Травкин Д.Н. – 39 — **X** — Хайло Н.Н. — 162

Чибисов П.А. – 31, 52

-III -

Шалумов А.С. – 39 Шахнов В.А. – 103 Шевченко А.А. – 178 – Щ – Щербаков А.С. – 92 Щигорев Л.А. – 123

– Э – Эсула О.И. – 86

– **Я** – Якунин А.Н. – 149

Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС)

2018. Выпуск II

Под общей редакцией академика РАН А.Л.Стемпковского

Федеральное государственное бюджетное учреждение науки Институт проблем проектирования в микроэлектронике Российской академии наук

Подписано в печать 27.07.2018 г. Формат 60х84/8. Печать офсетная. Бумага офсетная. Печ. л. 24,75. Усл. печ. л. 23,02. Тираж 175. Заказ № 32.

Издательство «Манускрипт» 248000, г. Калуга, ул. Баженова, 2. Тел. (4842) 57-31-87. E-mail: id_manuskript@mail.ru