УДК 621.382.2

Методика встроенного тестирования субмикронных цифровых КМОП СБИС

М.С. Ладнушкин

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

***Аннотация* – Предложена комплексная методика разработки встроенного тестирования КМОП СБИС, которая предусматривает создание иерархической модели СЦТ с компрессией, использование моделей ОЗУ для тестирования связей блоков памяти, применение технологии Pattern Map для тестирования блоков ОЗУ небольшой глубины и технологии MBIST для остальных блоков ОЗУ. Применение данной методики снизило время разработки на 20%, на 5% увеличило тестовое покрытие. Также повышена эффективность и сокращена площадь кристалла на 0,05% за счет применения технологии Pattern Map вкупе с MBIST.**

***Ключевые слова* – отбраковка, тестирование, микросхема, ОЗУ, сканирующие цепочки триггеров, технология Pattern Map.**

I. Введение

В современных высокопроизводительных СБИС, изготовленных по субмикронным техническим нормам, задача быстрой и точной отбраковки микросхем приобретает особую актуальность, особенно, если речь идет о высоконадежных системах. Разработка встроенных средств отбраковки (СО) и средств тестирования (СТ) микросхем является необходимым этапом в маршруте проектирования системы на кристалле, так как она позволяет с большой точностью отбраковывать кристаллы без использования длительных функциональных тестов [1]. В имеющемся высокопроизводительном процессоре [2] используемые средства тестирования являются недостаточно эффективными с точки зрения разработки как самих СО, так и микросхемы в целом.

В связи с переходом на технологический процесс 65 нм и усложнением функциональной схемы процессора актуальной задачей является анализ существующих методик проектирования СО и разработка методик по созданию новой архитектуры средств тестирования. Новая методика создания СО должна обеспечить блочную организацию СТ [4], которая позволит сократить время как на разработку СТ, так и время создания конечной микросхемы, и при этом не будет уступать в эффективности тестирования используемой в настоящее время архитектуре, а также позволит тестировать связи блоков ОЗУ и ячейки ОЗУ [5]. Таким образом, новая комплексная методика встроенного тестирования предполагает создание блочной архитектуры, обеспечивающей высокое тестовое покрытие и позволяющей тестировать связи и ячейки блоков ОЗУ с помощью сканирующих цепочек триггеров, сохраняя при этом малую длительность теста и малую площадь тестовой логики на кристалле. В данной статье рассматриваются производственные неисправности типа «залипание» (“stuck-at faults”).

II. Тестирование логики с помощью сканирующих цепочек триггеров

В настоящее время для отбраковки микросхем используется архитектура СТ, представляющая собой сканирующие цепочки триггеров (СЦТ) (см. рис. 1), а также встроенные средства тестирования ОЗУ (Memory Build-in-Self-Test, MBIST) для большей части блоков памяти на кристалле [1]. СЦТ предусматривают загрузку/выгрузку данных через единый блок компрессии тестовых сигналов [3], [4]. Технология MBIST предусматривает создание в проекте дополнительных функциональных блоков, контроллеров MBIST, которые осуществляют тестирование блоков ОЗУ по заданным алгоритмам. В связи с этим возрастает площадь кристалла СБИС.

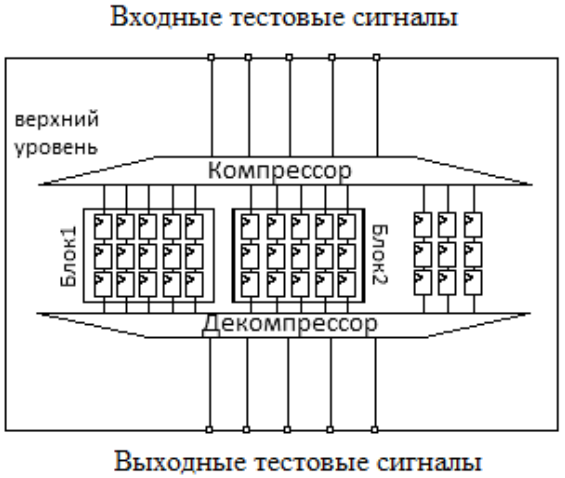

Рис. 1. Архитектура сканирующих цепочек с общим блоком компрессии

Создание средств тестирования осуществляется с помощью САПР фирмы Synopsys. Управление процессом создания архитектуры СТ, задание настроек и вывод результатов задается в файлах-сценариях, представляющих собой последовательность команд, написанных на языке TCL. Методика создания существующей архитектуры предусматривает построение СЦТ на верхнем уровне иерархии для всего проекта (см. рис. 2).

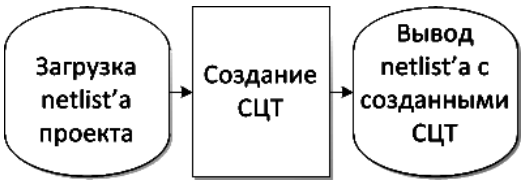

Рис. 2. Блок-схема, описывающая методику создания СЦТ с общим блоком компрессии

Данная архитектура достаточно эффективна при использовании в проектах с небольшой степенью интеграции за счет высокого тестового покрытия и простоты реализации. Но, необходимо отметить, что в проектах с высокой степенью интеграции (более 100 тысяч триггеров на одном кристалле) возникает ряд проблем [3]: архитектура не позволяет вести разработку СТ крупных блоков проекта параллельно для сокращения времени на создание СТ и всего проекта; изменения, вносимые разработчиками RTL-моделей в схему одного из блоков, потребуют повторного создания СЦТ для всего проекта, а значит потребуется заново выполнять топологию проекта – этап проектирования, который требует значительного времени (2 недели); нет возможности тестирования отдельных блоков проекта; нет возможности тестирования некоторых блоков ОЗУ (в которых не реализован MBIST); нет возможности тестирования связей блоков ОЗУ посредством СЦТ.

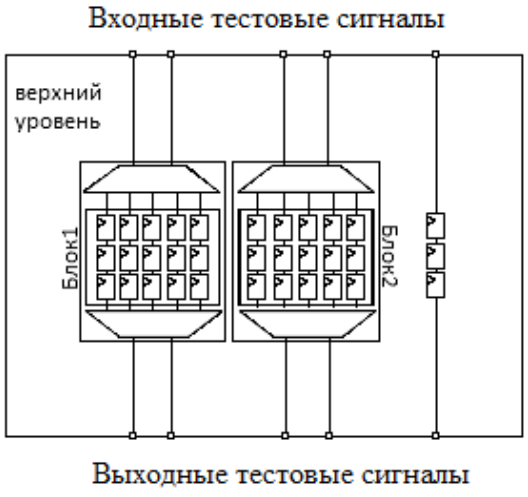

Рис. 3. Архитектура сканирующих цепочек с несколькими блоками компрессии

Современные САПР поддерживают возможность создания блочной архитектуры СТ, которая обычно применяется для тестирования высокопроизводительных микросхем с высокой степенью интеграции. Отличие блочной архитектуры от используемой заключается в возможности создания на кристалле независимых блоков тестирования, что позволяет распараллелить разработку СТ, а также упрощает дальнейшую модификацию проекта и позволяет тестировать блоки независимо друг от друга (см. рис. 3) [3].

Были разработаны методики, представляющие собой последовательность выполняемых действий в САПР Synopsys (см. рис. 4).

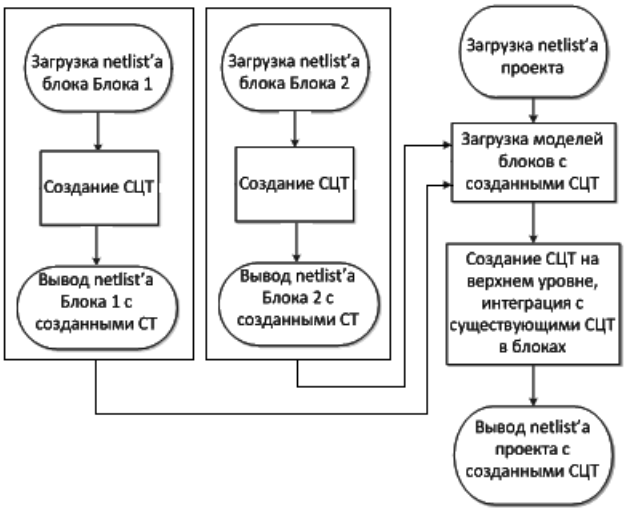

Рис. 4. Блок-схема, описывающая методику создания СЦТ с несколькими блоками компрессии

В отличие от известной архитектуры с общим блоком компрессии на верхнем уровне в каждом из блоков создаются независимые структуры СЦТ, которые впоследствии подключаются при создании СТ на верхнем уровне иерархии. Разработанная методика предусматривает параллельное создание СТ в каждом из блоков, что экономит время на разработку СТ.

Сценарий создания СЦТ в каждом из блоков представляет собой следующую последовательность действий: 1) загрузка netlist Блока 1; 2) создание тестовых портов; 3) описание тестовых сигналов; 4) описание сканирующих цепочек; 5) задание параметров создания сканирующих цепочек; 6) задание тестовых режимов; 7) задание параметров компрессии; 8) создание тестового протокола; 9) создание сканирующих цепочек в данном блоке; 10) вывод тестового протокола; 11) вывод отчетов о результатах создания СЦТ; 12) вывод тестовой модели Блока 1; 13) вывод netlist Блока1 с созданными СЦТ.

В результате создания СЦТ в Блоке 1 и Блоке 2 были созданы netlist этих блоков, а также тестовые модели. На верхнем уровне иерархии блоки с созданными СЦТ интегрируются в проект в виде тестовых моделей, а тестовые сигналы блоков выводятся на

верхнем уровне к портам ввода/вывода. Также создаются СЦТ для остальной логики на верхнем уровне.

Для получения конечного netlist всего проекта необходимо удалить тестовые модели блоков и подключить все netlist. Сценарий создания СЦТ на верхнем уровне выглядит следующим образом:

1) загрузка netlist верхнего уровня; 2) удаление из проекта Блока1 и Блока2; 3) загрузка тестовых моделей Блока1 и Блока2; 4) создание тестовых портов; 5) подключение тестовых портов блоков к портам ввода/вывода на верхнем уровне; 6) описание тестовых сигналов; 7) описание сканирующих цепочек; 8) задание параметров создания сканирующих цепочек; 9) задание тестовых режимов; 10) задание параметров компрессии и метода подключения блоков с созданными СЦТ; 11) создание тестового протокола; 12) создание сканирующих цепочек в проекте на верхнем уровне; 13) удаление тестовых моделей Блока1 и Блока2; 14) загрузка netlist Блока1 и Блока2; 15) вывод тестового протокола; 16) вывод отчетов о результатах создания СЦТ; 17) вывод netlist верхнего уровня с созданными СЦТ.

В результате выполнения сценариев, созданных на основе разработанных методик, были получены результаты, приведенные в табл. 1. Туда же для сравнения включены и результаты, полученные при создании СТ по применяемой в настоящее время методике с использованием общего компрессора на верхнем уровне. Из анализа табл. 1 следует, что предложенная архитектура немного уступает в тестовом покрытии (~1%), но при этом занимаемая площадь меньше на 5%. Тестовое покрытие составило 91,1%, что является хорошим результатом, учитывая тот факт, что связи блоков ОЗУ и сами ОЗУ не тестируются средствами СЦТ.

Таблица 1

Сравнение результатов создания различных архитектур СЦТ

Параметр	Общий компрессор	Несколько компрессоров
Тестовое покрытие, %	92,5	91,1
Площадь СТ, 10^3 мкм^2	934	878
Степень сжатия	10	8

III. Тестирование ОЗУ с помощью СЦТ

Тестирование ОЗУ в существующем проекте выполняется средствами MBIST. Достоинствами MBIST являются автономность и скорость тестирования ОЗУ. При этом MBIST реализуется посредством создания логического блока, а значит требует ресурсов на разработку и отладку RTL-модели, а также увеличивает площадь кристалла микросхемы. В современных САПР (таких как TetraMAX) реализована возможность тестирования ОЗУ с помощью СЦТ. Для этого необходимо внести ряд изменений в проект СБИС.

Нужно ввести дополнительный тестовый порт на верхнем уровне иерархии, управляющий сигналами разрешения записи (Write Enable) всех ОЗУ в режиме теста. Для этого в проекте создается новый тестовый порт TEST_WE, который коммутируется со всеми портами разрешения записи ОЗУ с помощью мультиплексоров, управляемых сигналом TEST_MODE (режим теста). В режиме теста (TEST_MODE = 1) порты разрешения записи ОЗУ управляются портом верхнего уровня TEST_WE.

Необходимо исключить возможность записи данных в ОЗУ по время «сдвига» СЦТ, что можно сделать с помощью сигнала TEST_SE (сигнал переключения фаз захвата/сдвига СЦТ в режиме теста), что осуществляется путем добавления логического элемента ”И” в цепь сигнала разрешения записи ОЗУ. На рис. 5 показано, как выглядит тестовая обвязка ОЗУ с учетом перечисленных изменений.

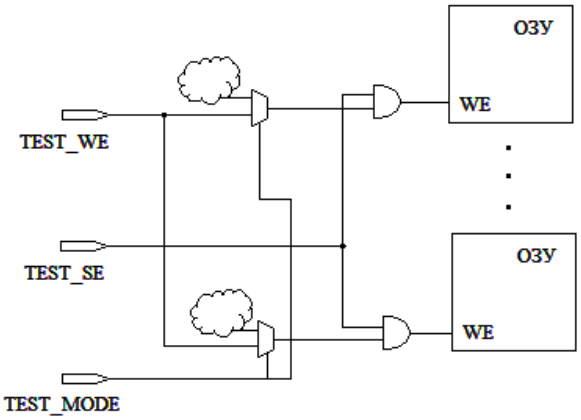

Рис. 5. Необходимая логика для тестирования ОЗУ

В TetraMAX существует две основные технологии генерации тестовых векторов для тестирования ОЗУ: с использованием моделей и Pattern Map (отображение векторов) [5].

A. Генерация векторов с помощью моделей

Автоматическая генерация тестовых векторов для СЦТ при тестировании ОЗУ возможна в том случае, если имеются специальные Verilog-файлы (модели), описывающие поведение ОЗУ (обычно поставляются в комплекте с библиотеками на ОЗУ [6]).

В результате, загружая проект с внесенными изменениями модели ОЗУ и задавая последовательный режим автоматической генерации тестовых векторов (Sequential ATPG), получаем результаты, отраженные в табл. 2. Из анализа данных следует, что тестовое покрытие с использованием моделей ОЗУ было увеличено на 5%.

Следует заметить, что при подсчете тестового покрытия ОЗУ САПР учитывались только неисправно-

сти портов памятей, сами же ячейки памяти не учитывались. В данном случае тестируются связи ОЗУ на залипание и окружающая их логика, ячейки ОЗУ практически не тестируются. Тем не менее, данный механизм позволяет в первом приближении оценивать работоспособность блоков ОЗУ.

Таблица 2

Сравнение результатов тестирования

Параметр	Без моделей ОЗУ	С моделями ОЗУ
Тестовое покрытие, %	91,1	96,3
Количество векторов	6000	42000

B. Технология Pattern Map

Технология Pattern Map позволяет, в отличие от тестирования с помощью моделей, проводить тесты ячеек памяти путем последовательной загрузки/выгрузки сканирующих векторов для последовательных операций чтения и записи в блоки ОЗУ. Данная технология не предусматривает создание дополнительных логических блоков на кристалле (как MBIST), но загрузка/выгрузка векторов требует больше времени, нежели непосредственный доступ к ОЗУ с контроллера MBIST. Создание тестовых векторов проходит в два этапа. Первый - создание модели теста ОЗУ на языке Verilog HDL. В результате моделирования теста в программе-симуляторе создается VCD-файл (Value Change Dump), который содержит последовательности воздействий и значения на выходах ОЗУ. Второй этап – загрузка VCD файла в программу TetraMAX для генерации сканирующих векторов. Создание VCD-файлов занимает до 10% от всего времени разработки СТ.

Последовательности воздействий представляют собой типичные алгоритмы тестирования, используемые при тестировании памяти по технологии MBIST: маршевый тест (MATS++), адресный тест, тесты «бегущая единица» и «бегущий ноль». В качестве тестируемых ОЗУ были выбраны регистровые двухпортовые ОЗУ (один порт для чтения и один порт для записи) с конфигурациями, различающимися по глубине: 32×8, 128×67 и 512×35. Результаты тестирования, а также площадь блоков MBIST для различных конфигураций ОЗУ представлены в табл. 3.

Из табл. 3 следует, что время тестирования с помощью СЦТ ОЗУ небольшой глубины, таких как 32×8, относительно невелико, хотя и заметно больше времени теста с помощью MBIST.

В результате анализа полученных результатов была предложена методика разработки СТ для СБИС: создание блоков MBIST нецелесообразно для ОЗУ небольших размеров, таких, как 32×8. Для тестирования таких ОЗУ стоит использовать СЦТ. Для остальных блоков ОЗУ следует применять технологию MBIST.

Таблица 3

Сравнение результатов тестирования ОЗУ с помощью технологий Pattern Map и MBIST

Параметр	32×8	128×67	512×35
Количество тестовых векторов для тестирования данной ОЗУ	306	1316	7407
Время выполнения теста с помощью СЦТ, мкс	16×10^6	70×10^6	393×10^6
Площадь блоков MBIST относительно площади ОЗУ, %	10	7	5
Время выполнения теста с помощью MBIST, мкс	Менее 70	Менее 3000	Менее 5000

IV. Заключение

На основе анализа экспериментальных данных по применению различных инструментов тестирования составлена комплексная методика разработки встроенного тестирования КМОП СБИС. Методика предусматривает создание иерархической модели сканирующих цепочек триггеров с компрессией, использование моделей ОЗУ для тестирования связей блоков памяти, применение технологии Pattern Map для тестирования блоков ОЗУ небольшой глубины и технологии MBIST для остальных ОЗУ. В результате применения предложенной методики уменьшено время разработки на 10% за счет распараллеливания процесса создания СЦТ с учетом времени, потраченного на создание VCD-файлов для тестирования ОЗУ; на 5% увеличено тестовое покрытие благодаря тестированию обвязки ОЗУ; повышена эффективность и сокращена площадь кристалла на 0,05% за счет применения технологии Pattern Map совместно с контроллерами MBIST.

Литература

[1] Crouch A. Design-for-Test for Digital IC’s and Embedded Core Systems. New Jersey: Prentice-Hall, 1999, pp. 95-108.

[2] Власов А.О., Евлампиев Б.Е., Кириченко П.Г., Кочнов А.А. Оптимизация некоторых этапов маршрута проектирования процессора КОМДИВ64-РИО // Сборник трудов «Проблемы разработки перспективных микро- и наноэлектронных систем». М.: ИППМ, 2010. С. 394-399.

[3] Alessandro V., Abuhamdeh Z. A Case Study of Hierarchical Scan Compression Implementation Using Synopsys' DFT MAX // SNUG Boston. 2006.

[4] Kaliamoorthy P., Kotari K. Hierarchical Adaptive Scan Synthesis for multi-million gate designs // SNUG Bangalore, India. 2007.

[5] Morton G. The Novel Use of TetraMAX Pattern Mapping Applied To A New Scan-Based Programmable BIST Architecture // SNUG, Europe. 2003.

[6] Lakamsani N. ATPG and Tetramax Memory Models // SNUG, San Jose. 2002.