

Московский Физико-Технический Институт
(государственный университет)

Тестирование подсистемы диагностики и компенсации неисправностей встроенных памятей системы на кристалле.

Магистерская диссертация
студента 418 группы
Тимина Леонида Сергеевича

Научный руководитель: Гурин Константин Львович

Москва
2010

Встроенная память

- Компактность
- Производительность
- 20-30% площади кристалла
- ~50% от общего числа транзисторов
- Full custom IP-блоки
- Малый размер транзистора
- Функционирование вблизи порогов переключения
- Огромное число транзисторов
- А техпроцессы все тоньше и тоньше...

Методы диагностики и компенсации неисправностей

- **Встроенное самотестирование и восстановление (BIST и BISR)**

Автоматически генерируемая последовательность обращений к памяти с целью обнаружить возможные неисправности, в случае обнаружения неисправного блока он может быть заменен запасным

- **Коды коррекции ошибок**

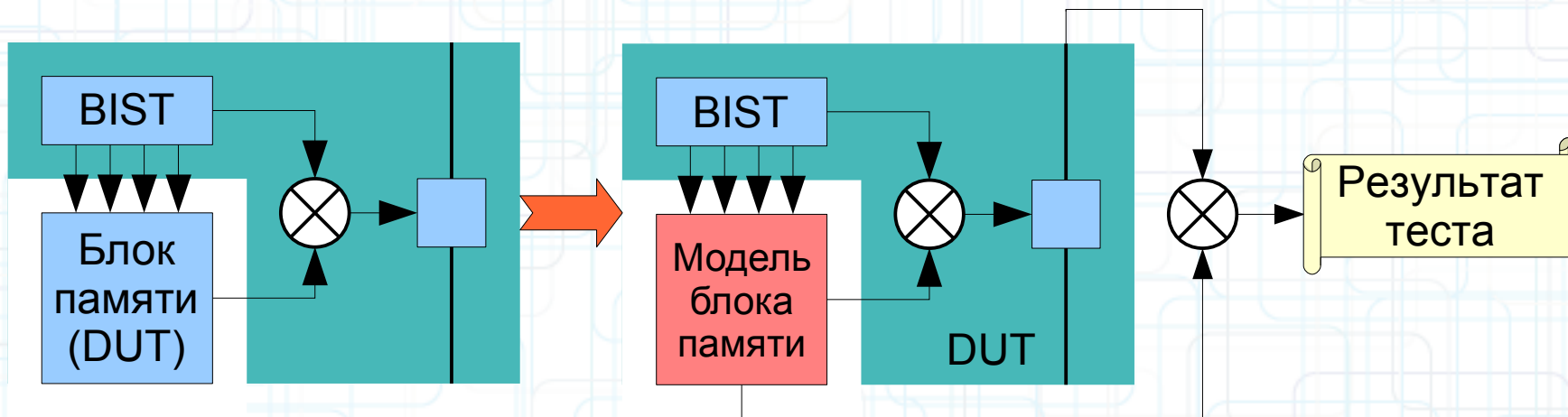
При записи в память к данным добавляются контрольные разряды, позволяющие обнаружить испорченные данные и восстановить их

- **Контрольные суммы**

При записи в память к данным добавляются контрольные разряды, позволяющее обнаружить испорченные данные

Цель работы

- Система контроля реализации встроенного самотестирования
 - Работа в среде Verilog-симуляции
 - Поддержка широкого класса неисправностей
 - Адаптируемость к различным проектам
- Использование полученной системы для тестирования проекта «Эльбрус-S»



В составе системы
на кристалле

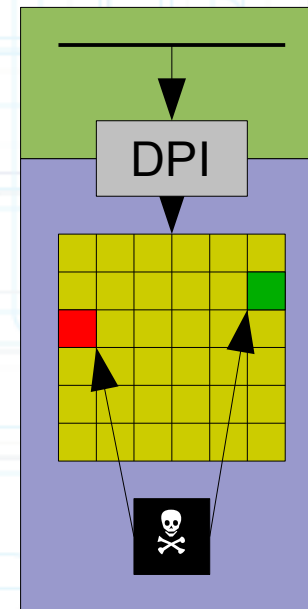
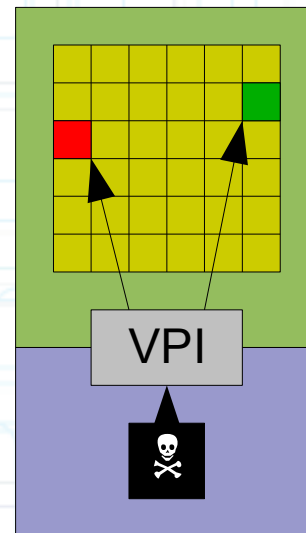
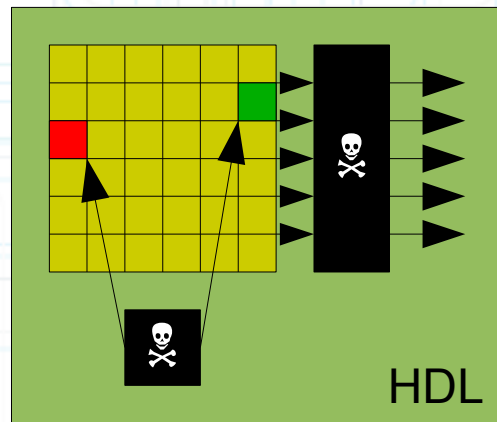
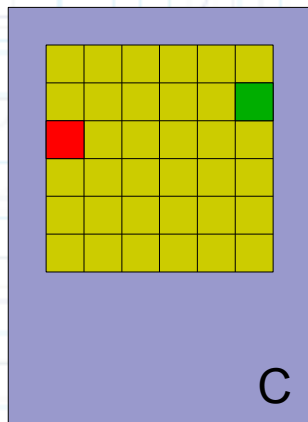
В составе тестового окружения

Порядок тестирования

- Моделирование неисправной встроенной памяти
- Запуск подсистемы диагностики
- Анализ реакции на неисправность
 - Проверка получения прерываний
 - Проверка корректности восстановления данных
 - Проверка синдромов и сигнатур

Существующие подходы к моделированию неисправностей памяти

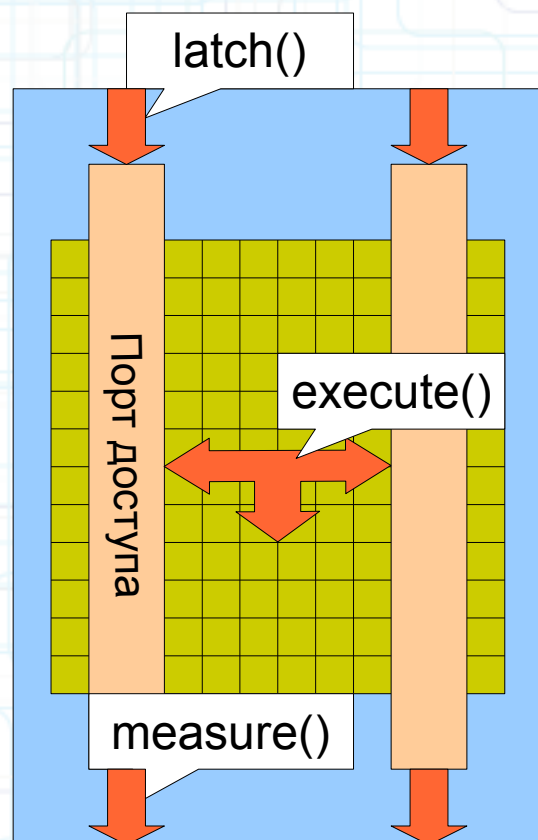
- Моделирование на языке программирования
- Моделирование на языке описания аппаратуры
- Смешанные методы, использование интерфейсов взаимодействия языков описания аппаратуры и языков программирования
- Для реализации выбрана программная модель, взаимодействующая с SystemVerilog осуществляется посредством DPI



Моделирование исправной памяти (1/2)

- Разнообразиие памятей
 - Размер слова и число слов
 - Число и функциональность портов доступа
 - Пословный и побитовый доступ
- Разнообразиие поведенческих моделей памятей
 - Интерфейс
 - Внутреннее устройство
- По результатам анализа были разработаны структура и интерфейс программной модели

Моделирование исправной памяти (2/2)



Интерфейс

- Input: chip_en, write_en, address, data_in, bit_mask
- Output: data_out

Стадии работы модели

- latch() - «захват» и декодирование входных сигналов
- execute() - исполнение полученных команд, взаимодействие портов доступа с массивом данных и друг другом
- measure() - определение выходного значения

Неисправности памяти

- **Разнообразие**
 - Расположение дефекта
 - Статические и динамические
 - Связанные с одной или несколькими ячейками памяти
 - Постоянные, проходящие, проявляющиеся
- **Нет единого метода описания**
 - <S/F/R>-нотация и ее разновидности
 - Модель конечных автоматов

Нужно проанализировать существующие и разработать подходящий метод описания неисправностей

Описание неисправностей массива данных (предпосылки)

- Массив данных — однородная структура
- Несколько неисправностей могут существовать в одном устройстве
- Неисправность — достаточно редкое явление
- Общепринято, что неисправное поведение демонстрирует только одна ячейка — «жертва»
- «Агрессоры» — исправные ячейки, влияющие на проявление неисправности
- Различные типы неисправностей могут быть объединены в параметризуемые классы

Описание неисправностей массива данных (состав)

- Список затронутых ячеек
 - Содержит одну «жертву» и несколько «агрессоров»
- Функциональная модель неисправности — функция
 - Текущего состояния неисправности
 - Номера порта
 - Времени
 - Текущих значений ячеек и операций над ними

Возвращает изменение содержимого «жертвы», изменение ответа «жертвы» на операцию чтения и следующее состояние неисправности.

$$(s_{(n+1)}, \Delta m_{vtm}, \Delta o_{vtm}) = f_{\bar{p}}(s_n, N_{port}, t, (m, op)_i)$$

Описание неисправностей массива данных (реализация)

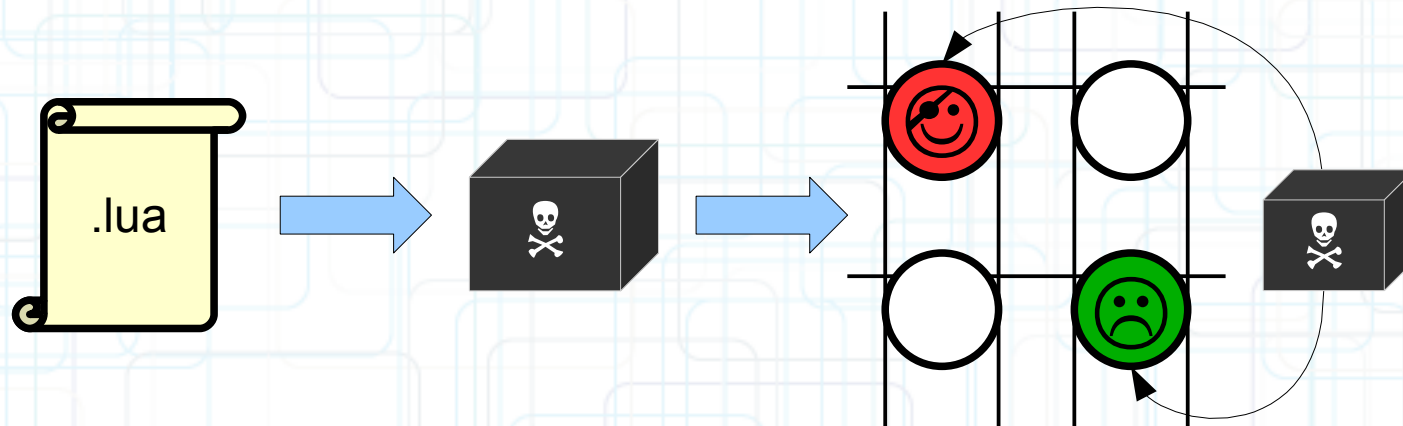
- Для реализации функциональных моделей был выбран язык Lua

Простой, компактный, достаточно мощный, устоявшийся и широко используемый, создавался как встраиваемый язык

- Состав описания функциональной модели неисправности
 - Список ячеек, занятых в описании с указанием их ролей
 - Список параметров неисправности и их значений по умолчанию
 - Функция инициализации состояния неисправности
 - Функция, реализующая модель неисправности

Моделирование неисправностей массива данных

- Описание компилируется в Lua-байткод
- Байткод загружается на новую виртуальную машину
- Виртуальная машина внедряется в модель исправной памяти
- Модель неисправности вызывается на execute()-стадии работы модели памяти
- Изменения применяются на execute()- и measure()-стадиях



Применение для тестирования системы на кристалле «Эльбрус-S»

- Модели памяти были внедрены в существующий дизайн
- Организовано автоматическое добавление неисправностей в модели памяти
- Создано окружение, запускающее процедуру встроенного самотестирования через JTAG-интерфейс
- Проведены исследования свойств используемого алгоритма тестирования памяти
 - Алгоритм тестирования выявил все неисправности, связанные с одной ячейкой
 - Обнаружены неисправности, связанные с двумя ячейками, которые не могут быть выявлены алгоритмом тестирования

Результаты

- Создана программная модель исправной памяти
- Предложен способ описания неисправностей памяти
- Создана библиотека функциональных описаний неисправностей
- Проведены исследования свойств алгоритма тестирования памяти, используемого в проекте «Эльбрус-S»

Направления дальнейшей работы

- Добавление ранее неизвестных типов неисправностей
- Тестирование реализации контрольных сумм и кодов коррекции ошибок
- Использование моделей исправной памяти для расширения диагностических возможностей при функциональном тестировании как отдельных блоков, так и вычислительной системы в целом

**Спасибо за внимание.
Вопросы.**