АЛГОРИТМЫ И МЕТОДЫ ОПТИМАЛЬНОГО КОДИРОВАНИЯ АППАРАТУРЫ САМОТЕСТИРОВАНИЯ ДЛЯ ОПЕРАТИВНЫХ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Белорусский государственный университет информатики и радиоэлектроники г. Минск, Республика Беларусь

Лукьянов А.А.

Иванюк А.А. – д.т.н., доцент

Оперативные запоминающие устройства (ОЗУ) - энергозависимая часть системы компьютерной памяти, в которой во время работы устройства хранится исполняемый машинный код (программы), а также входные, выходные и промежуточные данные, обрабатываемые процессором. Современные операционные системы, активно используют оперативную память, для хранения и обработки в ней важных и часто используемых данных. Если бы в электронных устройствах не использовалась оперативная память, то все операции происходили бы гораздо медленней и для считывания с постоянного источника памяти (ПЗУ), требовалось бы значительно больше времени. Из за высокой нагрузки на ОЗУ, важным вопросом является тестирование памяти для обеспечения надежности ее работы.

С увеличением сложности цифровых устройств наиболее актуальным и распространенным способом тестирования стала встроенная аппаратура самотестирования (BIST — Built-In Self-Test). Ядро BIST передает тестовые последовательности тестируемому устройству, считывает результаты и сравнивает их с ожидаемыми, и таким образом определяет, исправно ли тестируемое устройство. Существуют различные подходы к проектированию BIST. BIST может реализовывать один определенный алгоритм тестирования. В таком случае аппаратные затраты на BIST минимальны и скорость применения теста является максимальной, но данный подход является наименее гибким, так как он определяет единственный алгоритм тестирования, способный обнаруживать фиксированный набор неисправностей. Изменение алгоритма тестирования невозможно без необходимости перепроектирования BIST. Альтернативным подходом является BIST с микропрограммным управлением P-MBIST (Programmable Memory BIST), в которой алгоритм тестирования определяется микрокодом, хранящимся в памяти микропрограмм. P-MBIST позволяет изменять алгоритм тестирования во время жизненного цикла тестируемого устройства.

Было разработано много алгоритмов тестирования памяти для покрытия FFM (functional fault model), большинство из которых имели теоретическое происхождение. Традиционные ad-hoc тесты использовались в прошлом для неисправных устройств. Широкоизвестные тесты проход 1/0, GALPAT, Butterfly, Zero-one тест и Checkerboard. Однако, временная сложность первых двух тестов совершенно неприемлема в данный момент; в то время как покрытие ошибок последних трех тестов неприемлемы в промышленности. Маршевые тесты были введены для обнаружения Inversion Coupling Faults (CFin), Idempotent Coupling Faults CFid (CFid), а также SFs [1]. В таблице 1 представлено покрытие дефектов маршевыми тестами. В таблице, например, «a/b» означает, что тест обнаруживает «a», из «b» FP (fault primitive) соответствующего FFM. Например, March C обнаруживает оба FP TF, в то время как MATS+ обнаруживает только одну из них. CFin не включен в таблицу, т.к. имеет теоретическое происхождение и никогда не проявлялся в реальных конструкциях.

FFM	March Tests							
	MATS+	March C-	March B	PMOVI	March U	March LR	March Sr	March SS
SF	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
TF	1/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
WDF	0/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2
RDF	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
DRDF	0/2	0/2	0/2	2/2	0/2	0/2	2/2	2/2
IRF	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
CFst	4/8	8/8	6/8	8/8	8/8	8/8	8/8	8/8
CFid	3/8	8/8	8/8	7/8	8/8	8/8	8/8	8/8

Табл. 1

Один из возможных способов реализации маршевых тестов на P-MBIST - микропрограммный подход. В отличие от BIST в нем используется дополнительный запоминающий модуль, предназначенный для хранения тестовых наборов и микрокодов маршевых операций. Набор микроопераций также должен включать в себя операции сравнения, команды переходов, автодекремента и инкремента, остановки тестирования [2]. Маршевый тест можно представить как совокупность предопределенных компонентов, представляющих собой целую фазу теста [3]. В маршевых тестах из табл. 1 можно выделить следующие компоненты: SM0 (wd) , SM1 ($r\bar{d}$, wd) , SM2 ($r\bar{d}$, wd, rd, $w\bar{d}$), SM3 ($r\bar{d}$, wd, $w\bar{d}$), SM4 ($r\bar{d}$, $r\bar{d}$, $r\bar{d}$), SM5 ($r\bar{d}$) , SM6 ($r\bar{d}$, wd, $w\bar{d}$, wd), SM7 ($r\bar{d}$, wd, rd). С учетом программной реализации можно составить микропрограмму (пример на рис. 1), которая воспроизведет маршевый тест для аппаратуры встроенного самотестирования ОЗУ.

Одним из важных вопросов программирования аппаратуры самотестирования является оптимизация микрокода. В работе [3] предложен способ кодирования, при котором каждый элемент может быть представлен 5 значениями: AO - порядок обхода, LO - флаг, показывающий является ли операция последней

в текущем SMx компоненте, OT - тип операции (r/w), dt - эталонное значение, NOE - количество SMx компонентов в маршевом тесте. Каждый маршевый тест может быть закодирован:

$$V' = [log_2(max(n_e))] + n_e + 2n_o$$

Таким образом, большинство маршевых тестов будет занимать не более 42 бит.

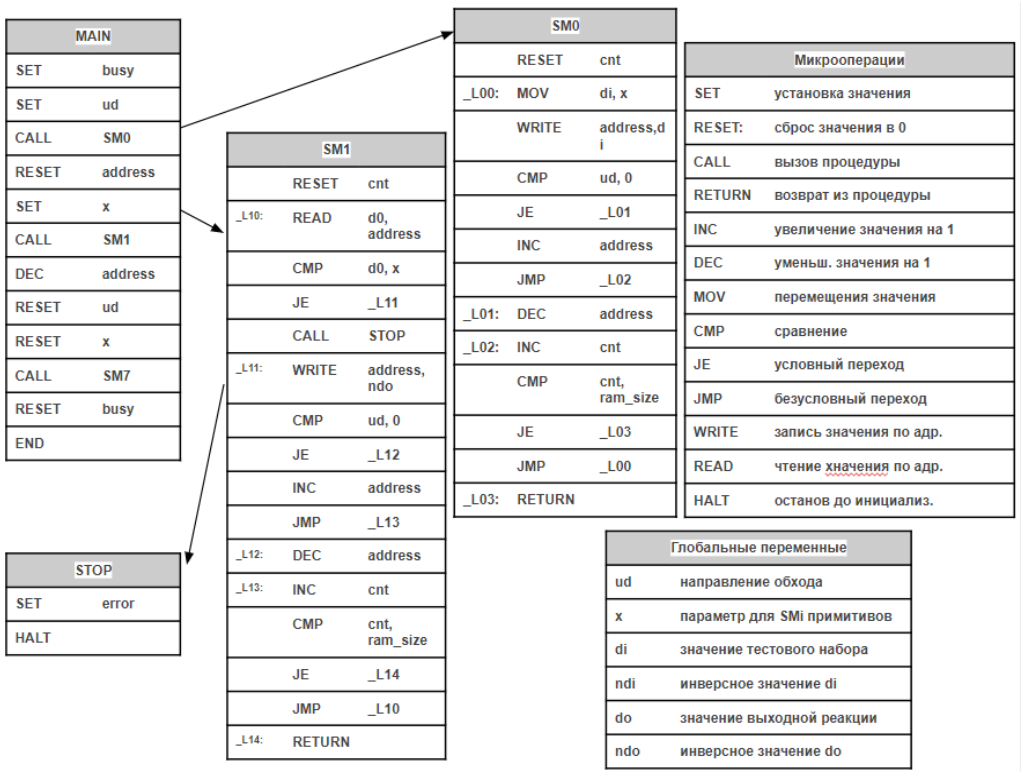

Рис. 1

Альтернативой может являться метод программируемого конечного автомата. Этот метод заключается в реализации маршевых примитивов как последовательностей состояний управляющего автомата. Входные параметры и порядок следования могут хранится в небольшой памяти. Это решение обладает меньшими аппаратными затратами, но в то же время менее гибкое.

Развитие средств встроенного самотестирования ОЗУ связано с появлением так называемых неразрушающих маршевых тестов, идея применения которых базируется на использовании сигнатурного анализа. Маршевый тест может быть использован для проведения процедуры неразрушающего тестирования, которая состоит из трех частей [4]:

- Вычисление значения эталонной сигнатуры SL;
- Применение неразрушающего теста с вычислением значения рабочей сигнатуры SW;
- Анализ значений двух сигнатур SL и SW, по результату которого можно судить о состоянии ОЗУ;

В работе рассмотрены некоторые методы и алгоритмы, применимые для тестирования ОЗУ. Так как одна из проблем тестирования ОЗУ заключается в небольшом количестве памяти, доступной для хранения, в дальнейшем планируется рассмотрение и создание методов оптимизации микрокода встроенной аппаратуры самотестирования ОЗУ.

Списокиспользованныхисточников:

1. Said Hamdioui, Ad J. van de Goor “March SS: A Test for All Static Simple RAM Faults” In proc IEEE International Workshop on Memory Technology, Design and Testing, 2002

2. Иванюк А.А., Ярмолик В.Н. // Проектирование контролепригодных цифроввых устройств. 2006. С. 193–205

3. Ivaniuk A. // Optimal Memory Tests Coding for Programmable BIST Architecture. 2008.

4. Zarrineh K. Upadhyaya S.J. // Design, Automation and Test in Europe (DATE’99): proc. of IEEE Int. Conf. Munich, Germany. Mar. 9–12, 1999. P. 709–713.