УДК 681.326.7

СОВРЕМЕННЫЕ НЕРАЗРУШАЮЩИЕ МЕТОДЫ И АЛГОРИТМЫ ДИАГНОСТИРОВАНИЯ ОПЕРАТИВНЫХ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

А.А. ИВАНЮК, Д.С. ПЕТРОНЕНКО

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 4 июля 2004

В статье приведен обзор современных неразрушающих алгоритмов самодиагностирования оперативных запоминающих устройств. Рассмотрено применение адаптивного сигнатурного анализатора при построении встроенных неразрушающих схем самодиагностирования памяти. Проанализирована эффективность дихотомических алгоритмов самодиагностирования запоминающих устройств.

Ключевые слова: маршевые тесты, адаптивный сигнатурный анализатор, неразрушающая самодиагностика, дихотомические алгоритмы.

Введение

На протяжении последних лет наблюдается устойчивая тенденция по совершенствованию технологии производства цифровых устройств [1]. С каждым годом все большее распространение получают встроенные системы и системы на кристалле [2], когда практически вся аппаратура устройства расположена на одном чипе. В структуре таких систем доминирующее положение постепенно начинают занимать оперативные запоминающие устройства (ОЗУ) [2].

Постоянное увеличение емкости и уменьшение технологических норм производства оперативной памяти приводит к значительному увеличению количества сбоев и отказов запоминающих устройств в процессе эксплуатации цифровой техники [1]. Для повышения отказоустойчивости и надежности оперативных запоминающих устройств применяется ряд мер, позволяющих обнаружить, локализовать и исправить возникающие неисправности и ошибки. К таким мерам относятся средства контроля, тестирования, диагностирования и ремонта запоминающих устройств [1–20]. При рассмотрении методов повышения надежности схем памяти имеет смысл уточнить основные термины и понятия теории диагностирования ОЗУ.

Неисправность — отказ памяти, возникающий как на стадии изготовления запоминающего устройства в результате несовершенства конструкции и технологии, так и на этапе эксплуатации.

Диагностирование ЗУ — проверка работоспособности ОЗУ с целью выявления места, причины и характера неисправностей запоминающего устройства.

Современные методы самодиагностики

В ходе эволюционного развития средств диагностирования (самодиагностирования) первыми наибольшее распространение получили средства тестирования [4]. Поэтому логичным является развитие диагностических алгоритмов запоминающих устройств, основанных на уже разработанных средствах тестирования и самотестирования [5, 8, 10]. При таком подходе па-

мять тестируется встроенной схемой самотестирования, а информация о неисправных ячейках памяти в сжатом виде передается на внешнюю диагностическую аппаратуру. Несмотря на ряд достоинств, такой метод диагностирования ОЗУ обладает существенными недостатками: ограничение на передаваемый объем информации от схемы встроенного самотестирования к внешней диагностической аппаратуре, необходимость учета ограничения числа выводов микросхем, необходимость использования внешнего оборудования. Развитие научной мысли, а также постоянное совершенствование технологии производства цифровых устройств позволило в последнее время для диагностирования массива запоминающих элементов применять аппаратуру встроенного диагностирования (BISD — *built-in self diagnosis*). Для этого на кристалле располагается дополнительная аппаратура, обеспечивающая генерацию тестовых наборов и обработку результатов диагностирования. По сравнению с внешним диагностированием встроенное диагностирование обладает рядом преимуществ:

для диагностирования устройства не надо дорогостоящее внешнее оборудование;

диагностирование памяти может проводиться как на этапе производства, так и на этапе эксплуатации устройства конечным пользователем;

аппаратура диагностирования может работать на более высокой внутренней частоте устройства;

одновременно может диагностироваться несколько модулей памяти, имеющих встроенную аппаратуру самодиагностирования.

В случае встроенного ОЗУ из-за ограничений, накладываемых на количество внешних контактов кристалла и доступность схем памяти, применение аппаратуры встроенного самодиагностирования является единственно приемлемым решением. При построении схем встроенного самодиагностирования, как и в случае внешнего диагностирования, алгоритмы диагностирования основываются на уже разработанных функциональных тестах ЗУ.

Неразрушающие алгоритмические ОЗУ

Все современные подходы встроенного самодиагностирования (как разрушающего, так и неразрушающего) базируются на функциональных тестах схем памяти [2, 5, 8]. Поэтому рассмотрение методов встроенного самодиагностирования без приведения обзора функциональных тестов запоминающих устройств было бы недостаточным.

Важным этапом в эволюции функциональных тестов ОЗУ стало применение неразрушающих методов тестирования [20]. Необходимость в их использовании обусловлена критичностью современных приложений (сетевые серверы, системы управления) к показателям надежности. В таких случаях необходимо использовать тестовые алгоритмы, которые не разрушали бы содержимое ОЗУ. В первых разработках систем периодического тестирования подобная проблема решалась при помощи резервной памяти [3]. Перед началом тестирования содержимое основной памяти сохранялось в резервной области, а после тестирования информация переписывалась обратно в основную память. Такой подход требует наличия резервной памяти, больших временных затрат и его невозможно применять для тестирования встроенной памяти в силу трудностей, связанных с доступом к ее содержимому. Поэтому в последнее время получили распространение методы неразрушающего тестирования ОЗУ в виде аппаратуры встроенного самотестирования [9]. Одной из первых работ, посвященной неразрушающему тестированию ОЗУ, был доклад Б. Конемана на семинаре "*Design For Testability*" 1986 года. Технология, предложенная Конеманом, основывалась на линейности сигнатурного анализатора и состояла из следующих шагов.

Содержимое памяти сжималось на сигнатурном анализаторе. Полученная таким образом сигнатура $s(old)$ записывалась в регистр эталонной сигнатуры.

Новое содержимое памяти вычислялось исходя из формулы $new=\text{old}\oplus\text{tp}$, где *new* — новое содержимое памяти, *old* — начальное содержимое памяти, *tp* — тестовая маска; символ $\oplus$ — поразрядная операция "исключающее ИЛИ".

Содержимое памяти опять сжималось на сигнатурном анализаторе. Полученная сигнатура $s(new)$ записывалась в регистр рабочей сигнатуры.

После этого восстанавливалось исходное состояние памяти (возможность этой операции обеспечивается свойством линейности операции "исключающее ИЛИ").

Считалось, что память не содержит неисправностей, если выполнялось равенство: $s(2) \oplus s(1) = s(tp)$, где $s(tp)$ — сигнатура тестовой маски, символ $\oplus$ — поразрядная операция "исключающее ИЛИ".

Описанная технология имеет ряд недостатков:

адекватная работа данной технологии возможна только при использовании линейного сигнатурного анализатора для сжатия содержимого памяти;

в некоторых случаях имеет место маскирование неисправностей. Например, если какая-либо отражается в ошибку двойной кратности (т.е. ошибку, которая будет присутствовать в памяти во время выполнения шагов 1 и 3), то, представив ошибку как маску E, накладываемую на память, получим:

$$s(old \oplus E) \oplus s(new \oplus E) = s(old \oplus E \oplus new \oplus E) = s(tp),$$

т.е. произошло маскирование неисправности.

большинство моделей неисправностей, использующихся для описания поведения реальных дефектов, достаточно сложны и для их обнаружения необходимо использовать намного более сложные тесты. Напротив, в алгоритме Конемана используется обычное инвертирование содержимого памяти, что не может обеспечить необходимую покрывающую способность.

Наиболее важный результат в области неразрушающего тестирования был получен М. Николаидисом, который предложил следующие основные правила трансформации произвольного маршевого теста в его неразрушающую версию [20] (AL_0 — исходный маршевый тест):

удалить фазу инициализации из алгоритма AL_0 (получаем алгоритм AL_1);

все операции записи алгоритма AL_1 заменяют их неразрушающими аналогами. Пусть x — это значение, считанное из памяти предыдущей операцией чтения, а y — значение, записываемое операцией записи. Тогда, если $x=y$ то данная операция записи заменяется операцией w_{-a}, иначе операцией w_a (получаем алгоритм AL_2);

заменить все операции чтения в алгоритме AL_2 на операцию r^* (получаем алгоритм AL_3). Алгоритм AL_3 представляет неразрушающую версию исходного алгоритма AL_0. Он называется базовым неразрушающим тестом. На 4 шаге получается алгоритм AL_4, который используется для вычисления эталонной сигнатуры;

удалить все операции записи из алгоритма AL_1 и заменить все операции $r0$ и $r1$ на операцию r^*;

Рассмотрим описанную выше процедуру на примере маршевого теста *March B*:

AL_0: ↕($w0$); ↑($r0$, $w1$, $r1$, $w0$, $r0$, $w1$); ↑($r1$, $w0$, $w1$); ↓($r1$, $w0$, $w1$, $w0$); ↓($r0$, $w1$, $w0$).

На первом шаге удаляется фаза инициализации. При этом получается следующий алгоритм:

AL_1: ↑(r0, w1, r1, w0, r0, w1); ↑(r1, w0, w1); ↓(r1, w0, w1, w0); ↓(r0, w1, w0).

На втором шаге заменяют все операции записи:

AL_2: ↑($r0$, w_{-a}, $r1$, w_{-a}, $r0$, w_{-a}); ↑($r1$, w_{-a}, w_a); ↓($r1$, w_{-a}, w_a, w_{-a}); ↓($r0$, w_{-a}, w_a).

На третьем шаге заменяются все операции чтения:

AL_3 ↑(r^*, w_{-a}, r^*, w_{-a}, r^*, w_{-a}); ↑(r^*, w_{-a}, w_a); ↓(r^*, w_{-a}, w_a, w_{-a}); ↓(r^*, w_{-a}, w_a).

Данный алгоритм является базовым неразрушающим тестом.

На четвертом шаге получается алгоритм чтения эталонной сигнатуры:

AL_4 ↑(r^*, r^*, r^*); ↑(r^*); ↓(r^*); ↓(r^*).

Теперь опишем полную процедуру тестирования памяти:

вычисляется эталонная сигнатура C^{REF} путем выполнения алгоритма AL_4;

вычисляется рабочая сигнатура C_{TEST} путем выполнения неразрушающего теста AL_3;

сигнатуры C_{REF} и C_{TEST} сравниваются. Результат сравнения определяет наличие или отсутствие неисправностей в ОЗУ

Сложность данного теста определяется общим количеством циклов обращения к памяти в процессе выполнения обеих процедур. Анализ последнего примера показывает, что неразрушающий маршевый тест имеет большую сложность по сравнению с обычным маршевым тестом. Оценки сложности неразрушающих тестов и их оригиналов приведены в таблице.

Сравнительная сложность стандартных и неразрушающих маршевых тестов

Название теста	Сложность теста	Сложность неразрушающего теста		
		Фазы вычисления C_{REF}	Фазы тестирования и вычисления C_{TEST}	Общая сложность
MATS	4*n*	2*n*	3*n*	5*n*
March C-	10*n*	5*n*	9*n*	14*n*
Algarithm B	17*n*	6*n*	16*n*	22*n*

Описанной технологии неразрушающего тестирования свойствен ряд недостатков:

данная технология существенно увеличивает сложность теста (на 40–50 % для большинства маршевых тестов) в силу необходимости вычисления эталонной сигнатуры перед началом тестирования;

данная технология не гарантирует стопроцентную покрывающую способность даже для однократных неисправностей из-за эффекта маскирования;

наблюдается ухудшение диагностической способности аппаратуры к самотестированию, основанной на классическом неразрушающем тестировании. Это вызвано сложностью вычисления адреса неисправной ячейки из разности эталонной и рабочей сигнатур.

Несмотря на недостатки описанного подхода неразрушающего тестирования данная методика (с различными модификациями) активно применяется при построении схем неразрушающего встроенного самодиагностирования. При использовании такого подхода значительное влияние на диагностическую способность оказывают обнаруживающая способность применяемого функционального теста, а также качество анализа выходных последовательностей ЗУ.

Адаптивный сигнатурный анализ выходных последовательностей ОЗУ

Как уже отмечалось выше, в последнее время получили распространение методы неразрушающего диагностирования ОЗУ в виде аппаратуры встроенного самодиагностирования. Все современные технологии неразрушающего тестирования и диагностирования для встроенной памяти основываются на использовании процедуры сжатия содержимого памяти [17, 19]. Поэтому важным элементом схем неразрушающего тестирования и диагностирования является схема получения сигнатуры памяти.

Чаще всего в качестве схемы вычисления сигнатуры используются классические сигнатурные анализаторы, преимущество которых заключается в малых аппаратурных затратах и высокой обнаруживающей способности. Однако главным недостатком классических сигнатурных анализаторов является невозможность быстрой коррекции сигнатуры памяти при изменении содержимого даже одной ячейки памяти. Вследствие этого недостатка был предложен подход, который в случае изменения содержимого памяти позволяет производить коррекцию сигнатуры, не используя сжатие всего содержимого памяти.

По сравнению со стандартным сигнатурным анализатором адаптивный сигнатурный анализатор (АСА) [19] позволяет корректировать сигнатуру памяти, содержимое которой изменяется в процессе работы. При применении АСА новое значение сигнатуры вычисляется исходя из предыдущего значения сигнатуры, содержимого ячейки памяти и записываемого значения. При этом для вычисления сигнатуры бит-ориентированной памяти используя АСА можно применять простую процедуру побитного сложения по модулю 2 адресов тех ячеек памяти, в которых записано значение "1".

Адаптивный сигнатурный анализ обладает следующими свойствами:

сигнатуры для произвольного состояния памяти и обратного ему равны между собой;

конечное значение сигнатуры не зависит от способа и направления движения по памяти при вычислении сигнатуры;

при использовании АСА в качестве сигнатурного анализатора все однократные неисправности памяти будут обнаруживаться и диагностироваться. При этом адрес неисправности вычисляется путем побитого сложения по модулю 2 двух сигнатур, полученных до и после проявления неисправности;

все двукратные неисправности обнаруживаются.

В случае применения АСА вычисление эталонной сигнатуры состоит из единственной фазы $\uparrow(r^*)$.

Полная процедура тестирования памяти на основе АСА состоит из следующих этапов:

вычисляется эталонная сигнатура *Cref* путем выполнения алгоритма $\uparrow(r^*)$. Полученная эталонная сигнатура записывается в регистр эталонной сигнатуры

выполняется вычисление рабочей сигнатуры *Ctest* путем выполнения первой фазы неразрушающего маршевого теста;

полученные сигнатуры (*Cref* и *Ctest*) сравниваются между собой. Результат сравнения определяет наличие или отсутствие неисправностей в памяти;

повторяются шаги 2–3 для всех остальных фаз неразрушающего маршевого теста

Подобная процедура тестирования памяти обеспечивает более высокую покрывающую способность по сравнению с классическим неразрушающим тестированием в силу значительного уменьшения длины сжимаемой последовательности. Благодаря тому факту, что значения эталонной и рабочей сигнатур сравниваются после выполнения каждой фазы маршевого теста, длина сжимаемой последовательности уменьшается в κ раз, где κ — количество фаз маршевого теста. При этом достигается 100 %-ная покрывающая способность для однократных и двукратных неисправностей, обнаруживаемых данным маршевым тестом.

Описанный адаптивный сигнатурный анализатор часто применятся при построении современных встроенных неразрушающих схем самодиагностирования памяти.

Неразрушающая самодиагностика

Практически все современные методы встроенной неразрушающей самодиагностики основаны на использовании адаптивного сигнатурного анализатора [19] в качестве схемы сжатия выходных последовательностей диагностируемой памяти, а также на использовании симметрий маршевых тестов, так как большинство неразрушающих маршевых тестов для схем встроенного самотестирования памяти получают информацию на операциях чтения с определенной степенью симметрии [18]. Пример с симметрией можно рассмотреть на классическом маршевом тесте *MATS+* $\{\uparrow(rd, wdc); \downarrow(rdc, wd)\}$, с помощью которого можно показать, как использование симметрии может увеличить эффективность встроенного самотестирования. Далее будет использоваться следующая нотация: *dc* есть комплиментарное значение к *d*; $\uparrow$ — указывает на увеличение адресов при работе теста, $\downarrow$ — указывает на уменьшение адресов, $\updownarrow$ — направление изменения адресов не важно (т.е. может быть или $\downarrow$ или $\uparrow$); *wdc*, *wd* — запись *dc* и *d* значений в ячейку памяти; *rdc*, *rd* — чтение из ячейки памяти ожидаемых значений *dc* и *d* и сжатие прочитанных значений ячеек памяти на сигнатурном анализаторе.

В соответствии с первой фазой *MATS+* теста для бит-ориентированной памяти размером 4 бита выходная последовательность на операции чтения будет, например, иметь следующее значение *d*=(1,0,1,1). Эта последовательность соответственно сожмется на сигнатурном анализаторе. При работе второй фазы теста выходная последовательность будет прочитана в обратном порядке и состоит из комплиментарных значений относительно первой последовательности, т.е. мы получим d=(0,0,1,0).

Такой тип симметрии очень удобно использовать при реализации схем тестирования, в которых в качестве элемента получения сигнатуры используется АСА. Общая схема построения маршевого теста с применением АСА и свойства симметричности маршевых тестов показана на рис. 1. В качестве неразрушающего маршевого теста взят неразрушающий маршевый тест *March C-*.

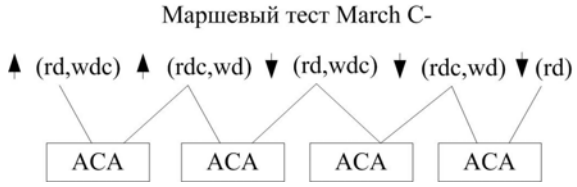

Рис. 1. Построение маршевого теста с применением АСА

При таком методе сжатия последовательности адресов ячеек памяти, содержащих "1", отпадает необходимость подсчета эталонной сигнатуры. В случае проявления неисправностей на определенной операции чтения соответствующий сигнатурный анализатор будет иметь с определенной доверительной вероятностью ненулевое значение, указывающее на наличие неисправностей в ОЗУ. При наличии в памяти только однократных неисправностей содержимое ненулевого АСА будет указывать на адрес неисправности.

Диагностические алгоритмы, основанные на симметричных неразрушающих тестах, обладают следующими свойствами, важными при их реализации в схемах встроенного неразрушающего самодиагностирования запоминающих устройств:

проведение диагностирования ОЗУ не изменяет содержимое памяти;

эталонная сигнатура, так же как и рабочая сигнатура, для исправной памяти равна 0 и не зависит от размера памяти и ее содержимого, выбранного симметричного неразрушающего теста, генерирующего полинома для сигнатурного анализатора;

покрывающая способность диагностических алгоритмов, основанных на симметричных неразрушающих маршевых тестах, является как минимум не хуже обычных неразрушающих маршевых тестов.

Дихотомические неразрушающие диагностические алгоритмы

Для неразрушающего диагностирования многократных неисправностей, возникающих в схемах ОЗУ, были предложены дихотомические алгоритмы диагностирования [17], основанные на симметричных неразрушающих тестах [18].

Рассмотрим общую схему работы таких диагностических алгоритмов на примере неразрушающего диагностического алгоритма "*Divide and Check*" [17].

Входными данными для алгоритма "*Divide and Check*" являются размер блока памяти (который будет диагностироваться) n и его начальный адрес l.

Начало.

$w=n/2$, $r=l$, где n — размер диагностируемого блока памяти с начальным адресом l, w — размер окна памяти с начальным адресом r;

запустить алгоритм диагностирования на базе СНТ (симметричный неразрушающий тест) для блока памяти (n, l), данные сжимать только внутри окна памяти (w, r). Если *Ctest*(w, r)_FAIL и $w>1$; $w=w/2$, перейти на начало пункта 2; если $w=1$ — перейти на пункт 3 алгоритма. Если *Ctest*(w, r)_PASS и $w>1$, $r=r+w$, $w=w/2$, перейти на начало пункта 2; если $w=1$; $r=r+1$ и перейти к 3 пункту;

адрес неисправной ячейки памяти равен значению r. Передать информацию о неисправной ячейке и запустить алгоритм ремонта ячейки;

запустить тест на блок памяти (n,l). Если *Ctest*(n,l)_FAIL, то перейти к пункту 1, иначе завершить работу диагностического алгоритма.

Как видно из последовательности действий для диагностического алгоритма "*Divide and Check*", дихотомические диагностические алгоритмы основаны на последовательной сходимости проверяемого блока памяти к одному адресу неисправной ячейки. Недостатком дихотомических диагностических алгоритмов является относительно большая сложность диагностической процедуры при увеличении общей кратности неисправностей, присутствующих в памяти.

Дальнейшим развитием неразрушающих дихотомических алгоритмов стало использование в этих алгоритмах адаптивного сигнатурного анализатора (в качестве средства сжатия выходных тестовых последовательностей) и бита четности для каждой пары комплиментарных операций чтения [22] (рис. 2). На рис. 2 СП блок представляет собой сигнатурный анализатор и

бит четности для одной пары комплиментарных операций чтения. При проведении процедуры диагностирования нулевое значение бита четности при ненулевом значении сигнатурного анализатора будет указывать на нахождение в памяти четного числа неисправностей.

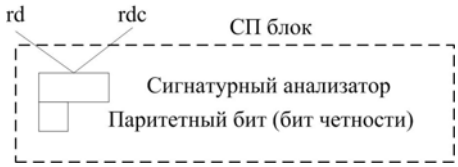

Рис. 2. СП блок для пары комплиментарных операций чтения

В процессе работы симметричного неразрушающего теста выходная тестовая последовательность сжимается только внутри тестового окна. Если диагностируемая память не содержит неисправностей, то все значения сигнатурных анализаторов и битов четностей СП блоков будут равны нулю. В случае однократной неисправности все ненулевые сигнатурные анализаторы будут содержать адрес неисправной ячейки памяти, а соответствующие биты четности будут равны единице. Во всех остальных случаях или все сигнатурные анализаторы равны нулю или биты четности при ненулевых сигнатурных анализаторах не равны нулю и соответственно мы должны продолжить деление текущего окна тестирования.

Таким образом, применение СП блоков в дихотомических алгоритмах позволяет уменьшить общее количество операций деления тестового окна благодаря возможности предсказания адреса однократной неисправности.

Дихотомический алгоритм с использованием СП блоков определяет адреса неисправных ячеек памяти как в случае *SAF* и *TF* неисправностей различной кратности, так и адреса ячеек-жертв при возникновении связных неисправностей. Данный алгоритм содержит в себе два диагностических подтеста: Неразрушающий Тест для Константных неисправностей и неисправностей перехода (НТК) и Неразрушающий Тест для Связных неисправностей (НТС) (рис. 3).

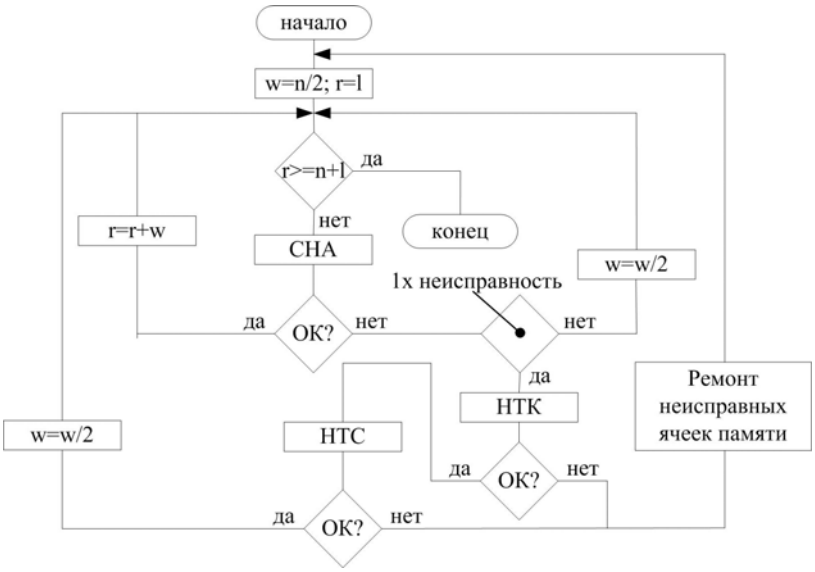

Рис. 3. Схема модифицированного дихотомического алгоритма

НТК алгоритм определяет константные неисправности и неисправности переходи для текущей ячейки памяти и имеет сложность 8 операций чтения и записи. НТС алгоритм находит ячейку-агрессор для предполагаемой жертвы. Средняя сложность НТС алгоритма составляет 4N, где N — общее количество ячеек памяти.

На рис. 3 используются следующие обозначения: n — размер тестируемого блока памяти; l — начальный адрес блока памяти; w — размер окна памяти, в котором происходит сжатие выходных тестовых последовательностей; r — начальный адрес окна памяти.

Описание неразрушающего теста для константных неисправностей и неисправностей перехода:

Данный алгоритм состоит из единственного маршевого элемента, который определяет константные неисправности и неисправности перехода: (*rd*, *wdc*, *rdc*, *wd*, *rd*). Результатом работы данного алгоритма являются 3 случая:

все значения операций чтения равны. В этом случае текущая ячейка памяти содержит константную неисправность;

значение, считанное на первой операции чтения, не совпадает со значениями второй и третьей операций чтения — ячейка содержит неисправность перехода;

значение, считанное на второй операции чтения, не совпадает со значениями первой и третьей операций чтения — текущая ячейка памяти не содержит ни константной неисправности, ни неисправности перехода.

Неразрушающий тест для связанных неисправностей состоит из последовательности следующих операций:

сохранение значения ячейки жертвы;

запуск маршевого элемента (*rd*, *wdc*, *wd*) для каждой ячейки памяти за исключением ячейки жертвы. При проведении операции сравнивается сохраненное значение и текущее значение ячейки жертвы. Если равенство не выполняется, то устанавливается сигнал "Ошибка теста" и тест завершается. Текущий адрес памяти указывает на ячейку агрессор:

инвертирование значение ячейки жертвы;

сохранение значения ячейки жертвы;

повторение операции 2.

На рис. 4 приведены результаты сравнительной оценки сложности стандартного дихотомического алгоритма и дихотомического алгоритма с применением АСА в зависимости от размера памяти при наличии в памяти однократной неисправности. При оценке сложности в качестве основного теста использовалась неразрушающая версия *March C-* теста.

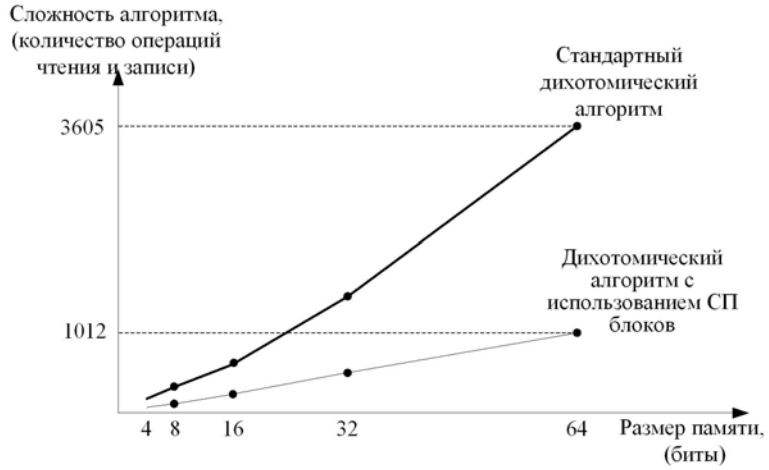

Рис. 4. Сложность дихотомических алгоритмов для однократных неисправностей

Применение симметричных неразрушающих тестов с использованием адаптивного сигнатурного анализатора при проведении дихотомического диагностирования позволяет резко увеличить скорость схождения алгоритма к неисправной ячейке памяти, вследствие чего сокращается общее время, необходимое на проведение диагностирования и ремонта памяти, что является критичным при проведении неразрушающего дихотомического диагностирования схем памяти.

Заключение

В представленном обзоре рассмотрены современные методы обеспечения надежной и отказоустойчивой работы оперативных запоминающих устройств. Основное внимание уделено способам встроенного неразрушающего самодиагностирования, которое, по мнению авторов,

является наиболее перспективным направлением развития средств технической диагностики запоминающих устройств.

STATE-OF-THE-ART TRANSPARENT MEMORY DIAGNOSTIC METHODS AND ALGORITHMS

A.A. IVANIUK, D.S. PETRONENKO

Abstract

Overview of state-of-the-art transparent memory build-in diagnostic algorithms is represented in the paper. Diagnosis for embedded RAM based on Self-Adjusting Output Data Compression is also considered. Important results of dichotomy algorithms efficiency are introduced at the end of the article.

Литература

1. Quality/reliability handbook. Micron technology, Inc., 1995.
2. *Chackraborty K., Mazumder P.* Fault-Tolerance and Reliability Techniques for High-Density Random-Access memory, Prentice Hall, 2002.
3. *Maly W., Naik S.* // Proc. Int'l Test Conf. 1989. P. 527–532.
4. *Chen J.T. et al.* // Rec. of VLSI Test Symp. 2001.
5. *Yarmolik V.N., Klimets Y.V., van de Goor A.J., Demidenko S.N.* // Proc. IEEE Int. Workshop on Memory Technology, Design and Testing (MTDT). 1996. P. 100–102.
6. *Bergfeld T.J., Niggemeyer D., Rudnick E.M.* // Proc. Design, Automation and Test in Europe (DATE). Paris, Mar., 2000. P. 305–309.
7. *Li J.-F., Wu C.-W.* // Proc. Design, Automation and Test in Europe (DATE). Munich, Mar., 2001. P. 97–101.
8. *Wang C.-W., Wu C.-F., Li J.-F., et al.* // Proc. Ninth IEEE Asian Test Symp. (ATS). Taipei, Dec. 2000. P. 45–50.
9. *Dekker R., Beenker F., and Thijssen L.* // IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 9(6) June 1990. P. 567–572.
10. *Yarmolik V.N., Ivaniuk A.A.* // Proceedings of the 7th International Conference, MIXDES 2000, P. 547–550.
11. *Yarmolik V.N., Hellebrand S., Wunderlich H.-J.* // DATE99. Munich, Germany, March 9–12, 1999. P. 702–707.
12. *Yarmolik V.N., Ivaniuk A.A, Krips M.* // Fifth International Workshop IEEE DDECS 2002. Brno, Czech Republic, April 17-19, 2002, P. 360–365
13. *Nicolaidis M.* // IEEE Int. Test Conference. 1992. P. 598–607.
14. *Ivaniuk A.A., Petronenko D.S., Sokol B.* // Proceedings of International Conference on Computer Information Systems and Industrial Management Applications (CISIM'2003), Elk, June 26-28, 2003. P. 212–218
15. *Yarmolik V.N., Buslowska E.* // Proceedings of International Conference on Computer Information Systems and Industrial Management Applications (CISIM'2003), Elk, June 26-28, 2003. P. 196–204.