УДК 681.326.7

А.П. Занкович, В.Н. Ярмолик

ВСТРОЕННАЯ АППАРАТУРА НЕРАЗРУШАЮЩЕГО САМОТЕСТИРОВАНИЯ ДЛЯ СХЕМ ОЗУ НА ОСНОВЕ ЛОКАЛЬНО-СИММЕТРИЧНЫХ ТЕСТОВ

Дается сравнительный анализ нескольких схем встроенной аппаратуры самотестирования оперативных запоминающих устройств (ВАСТ ОЗУ). Рассмотренные ВАСТ ОЗУ отличаются сложностью реализуемых тестов, количеством использованных сигнатурных анализаторов и количеством обнаруживаемых неисправностей. Предлагается алгоритм поиска минимального количества сигнатурных анализаторов, необходимых для реализации новых локально-симметричных тестов в виде ВАСТ ОЗУ, и генерации последовательности их использования.

Введение

Высокая степень интеграции современных цифровых систем позволяет реализовывать на одном кристалле сложные проекты, состоящие из нескольких миллионов вентилей. Это относится как к заказным схемам большой и сверхбольшой интеграции, так и к программируемым логическим интегральным схемам (ПЛИС), причем значительная часть кристалла (50–75 % общей площади) отводится под ОЗУ. Рост информационного объема ОЗУ и высокая плотность размещения запоминающих элементов приводят к проблемам технологического характера, вызывающим увеличение вероятности появления дефектов. Результаты исследований [1] свидетельствуют о том, что отказы ОЗУ составляют до 70 % от общего числа отказов системы в целом. Это объясняет высокую актуальность задачи периодической проверки работоспособности ОЗУ.

Для решения данной задачи в 1980-х гг. были предложены маршевые тесты [2] – высокоэффективные алгоритмы, позволяющие со 100 %-ной вероятностью обнаруживать наиболее распространенные неисправности ОЗУ, обладая при этом сравнительно небольшим временем выполнения. Их сложность оценивается как $O(N)$, где N – количество ячеек ОЗУ. Однако маршевые тесты в традиционном виде полностью разрушают текущее содержимое памяти, что ограничивает их применение входным контролем и моментами включения устройства.

Для критичных к показателям надежности приложений возникает проблема тестирования в промежутках между периодами нормального функционирования устройства. Примерами таких систем могут служить телекоммуникационные устройства, маршрутизаторы, микропроцессоры автономных систем управления и пр. Основным требованием, предъявляемым к периодическим тестам, является восстановление исходного состояния ОЗУ после сеанса тестирования.

При тестировании встроенных ОЗУ возникают определенные проблемы, связанные с невозможностью непосредственного подключения к информационным и управляющим входам и выходам схемы извне для организации диагностического эксперимента. С этой целью применяется подход, основанный на использовании встроенной аппаратуры самотестирования ОЗУ. Его основным преимуществом является возможность тестирования устройства на внутренней частоте. В данной работе проведено исследование существующих неразрушающих ВАСТ ОЗУ и представлен алгоритм генерации структуры ВАСТ, которые реализуют новые локально-симметричные тесты [3].

1. Неразрушающие маршевые тесты ОЗУ

1.1. Традиционные неразрушающие маршевые тесты ОЗУ

Для решения задачи периодического неразрушающего тестирования ОЗУ Николаидисом был предложен метод [5], позволяющий преобразовать классические маршевые тесты к их неразрушающему виду. В отличие от классических маршевых тестов, оперирующих с предопределенными значениями, неразрушающие тесты анализируют текущее содержимое ОЗУ и активизиру-

ют неисправности некоторым обратимым способом. В методе Николаидиса для этого используется инвертирование данных с последующим их восстановлением.

Обнаружение неисправностей производится путем сравнения наборов данных, прочитанных до и после инвертирования. Хранение сравниваемых наборов целиком нецелесообразно, используется их компактное представление, получаемое с помощью сигнатурных анализаторов на базе линейных сдвиговых регистров с обратной связью (LFSR – Linear Feedback Shift Registers). Соответственно тест включает две фазы: фазу вычисления начальной сигнатуры, состоящую только из операций чтения, и фазу базового неразрушающего теста, активизирующую возможные неисправности с помощью операций записи и формирующего рабочую сигнатуру. Начальная и рабочая сигнатуры данных в случае отсутствия ошибок будут совпадать.

Рассмотрим две фазы неразрушающего маршевого теста *TMarch C-*, полученного с помощью метода Николаидиса:

$$\Uparrow(r_a);\Uparrow(r_a);\Downarrow(r_a);\Downarrow(r_a);\Uparrow(r_a); \tag{1}$$

$$\Uparrow(r_a, w_{\bar{a}});\Uparrow(r_{\bar{a}}, w_a);\Downarrow(r_a, w_{\bar{a}});\Downarrow(r_{\bar{a}}, w_a);\Uparrow(r_a). \tag{2}$$

Каждую фазу составляют пять маршевых элементов. Они включают набор из операций чтения или записи, последовательно применяемых ко всем ячейкам памяти в определенном порядке – от старших адресов к младшим ($\Uparrow$) или наоборот ($\Downarrow$). При реализации операций чтения значения из ОЗУ передаются в прямом (r_a) или обратном ($r_{\bar{a}}$) коде во временный буфер, размер которого равен размеру слова памяти. При реализации операций записи (w_a и $w_{\bar{a}}$) данные помещаются соответственно в прямом или обратном коде из буфера в текущее слово ОЗУ.

Дальнейшее развитие неразрушающих тестов привело к созданию глобально- и локально-симметричных тестов [3, 6]. За счет отказа от использования фазы вычисления начальной сигнатуры они выполняются на 30–40 % быстрее тестов, полученных по методу Николаидиса. Обе сравниваемые сигнатуры данных получаются в процессе выполнения базового неразрушающего теста, причем ошибки могут исказить как первую, так и вторую сигнатуру. В обоих случаях нарушается симметрия сжимаемых данных, что позволяет сделать вывод о наличии неисправности.

Глобально-симметричные тесты ОЗУ [6] основаны на разбиении базового неразрушающего теста на две части, считывающие взаимно-симметричные последовательности данных. В случае необходимости к базовому неразрушающему тесту добавляют дополнительные операции чтения. Приведем в качестве примера такого теста *UDMarch C-*:

$$\underline{\Downarrow(r_a)};\Uparrow(r_a, w_{\bar{a}});\Uparrow(r_{\bar{a}}, w_a);|\ \Downarrow(r_a, w_{\bar{a}});\Downarrow(r_{\bar{a}}, w_a);\Uparrow(r_a). \tag{3}$$

Здесь вертикальной линией помечена граница симметричных последовательностей, добавленная операция подчеркнута. Легко заметить, что первая и вторая фазы теста *UDMarch C-* считывают взаимосимметричные последовательности данных (последовательность второй фазы отличается обратным порядком элементов и тем, что они проинвертированы).

1.2. Локально-симметричные неразрушающие маршевые тесты ОЗУ

При создании локально-симметричных тестов [3] рассматривается симметрия не всего теста, а его отдельных операций чтения или их групп. Главным требованием при выделении таких симметрий является обнаружение максимального количества неисправностей. Дополнительными требованиями могут являться простота аппаратной реализации полученного теста, возможность обнаружения активизированных неисправностей при его прерывании, минимизация количества маскируемых в результате наложения друг на друга ошибок. Основным понятием, используемым при построении локально-симметричных тестов, является симметричный элемент.

Симметричным элементом назовем два набора операций чтения маршевого неразрушающего теста, считывающих из ОЗУ симметричные между собой последовательности данных. Составляющие симметричный элемент наборы операций назовем его первой и второй симметричными фазами. Пусть $D = (D_0, \ldots, D_{N-1})$ – некоторая последовательность данных. Обратной назовем последовательность $D^* = (D_{N-1}, \ldots, D_0)$, инверсной – последовательность $\overline{D} = (\overline{D}_0, \ldots, \overline{D}_{N-1})$, а обратной инверсной – последовательность $\overline{D}^* = (\overline{D}_{N-1}, \ldots, \overline{D}_0)$. Всего возможны четыре типа симметрии последовательностей данных. Очевидно, что локально-симметричные тесты могут включать любой тип симметрии, но с точки зрения затрат на аппаратную реализацию сигнатурного анализа целесообразно сравнивать две идентичные и обратные инверсные последовательности, поскольку в этом случае выполняется непосредственное сравнение полученных сигнатур без дополнительных вычислений [4]. Назовем их последовательностями, обладающими первым и вторым типом симметрии соответственно. В статье [3] показано, как преобразовать тест с различными типами симметрии в тест с рассматриваемыми типами без потери покрывающей способности.

Обозначим через sig(g_0, P, O) сигнатуру последовательности данных O, получаемую на сигнатурном анализаторе, задаваемом примитивным полиномом P, с начальным значением g_0. Тогда выражения для сигнатурного анализа двух последовательностей O_1 и O_2, обладающих симметрией первого и второго типа, запишутся соответственно как

$$\mathrm{sig}(g_0, P, O_1) = \mathrm{sig}(g_0, P, O_2); \tag{4}$$

$$\mathrm{sig}(\mathrm{sig}(g_0, P, O_1)^*, P^{-1}{}^*, O_2^*) = g_0^*. \tag{5}$$

Рассмотрим два варианта локально-симметричных тестов ОЗУ: *SMarch C*-$_1$ и *SMarch C*-$_2$ (рис. 1). В основе каждого из них лежит базовый неразрушающий тест *TMarch C*- (2). Дугами соединены операции чтения, образующие маршевый элемент (оба теста содержат по четыре симметричных элемента).

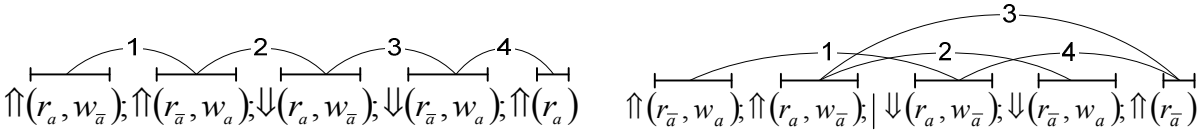

Рис. 1. Неразрушающие локально-симметричные тесты *SMarch C*-$_1$ и *SMarch C*-$_2$, соответственно

Воспользуемся более компактной записью указанных тестов:

SMarch C-$_1$ $\quad \Uparrow\left(r_a^{[1]}, w_{\bar{a}}\right);\ \left|^{[1]}\ \Uparrow\left(r_{\bar{a}}^{[1,2]}, w_a\right);\right|^{[2]}\ \Downarrow\left(r_a^{[2,3]}, w_{\bar{a}}\right);\ \left|^{[3]}\ \Downarrow\left(r_{\bar{a}}^{[3,4]}, w_a\right);\ \right|^{[4]}\ \Uparrow\left(r_a^{[4]}\right),$

SMarch C-$_2$ $\quad \Uparrow\left(r_a^{[1]}, w_{\bar{a}}\right);\ \left|^{[1]}\ \Uparrow\left(r_{\bar{a}}^{[2,3]}, w_a\right);\ \right|^{[2]}\ \Downarrow\left(r_a^{[1,4]}, w_{\bar{a}}\right);\ \left|^{[3]}\ \Downarrow\left(r_{\bar{a}}^{[2]}, w_a\right);\ \right|^{[4]}\ \Uparrow\left(r_a^{[3,4]}\right).$

Здесь к операциям чтения, входящим в симметричные элементы, дописывается верхний индекс с его порядковым номером в квадратных скобках (если номеров несколько, они перечисляются через запятую), а также граница первой и второй фазы из каждого симметричного элемента отмечается вертикальной чертой с его порядковым номером.

Локально-симметричные тесты можно рассматривать как обобщение рассмотренных выше тестов. Единственный симметричный элемент глобально-симметричных тестов охватывает все операции базового неразрушающего теста. В случае тестирования по методу Николаидиса первая фаза симметричного элемента формируется алгоритмом вычисления эталонной сигнатуры, вторая – базовым неразрушающим тестом. Соответственно аппаратный или программный модули, реализующие тестирование с помощью локально-симметричных тестов, можно использовать также и для реализации любого из существующих неразрушающих маршевых тестов.

2. Оптимизация локально-симметричных тестов ОЗУ для реализации в виде ВАСТ

2.1. Алгоритм FindBISTMin *поиска минимальной структуры ВАСТ для реализации заданного локально-симметричного теста*

Сигнатурный анализ идентичных последовательностей данных (первый тип симметрии) требует двух одинаковых сигнатурных анализаторов или одного сигнатурного анализатора и одного регистра хранения вычисленной сигнатуры. Сигнатурный анализ последовательностей данных второго типа симметрии требует двух сигнатурных анализаторов (прямого и обратного) или одного сигнатурного анализатора с возможностью прямого и обратного счета. Особенностью традиционных неразрушающих тестов является то, что в них используется только один тип симметрии и, следовательно, может использоваться аппаратный модуль тестирования с фиксированной структурой. Аппаратная реализация локально-симметричных тестов требует введения ограничений на количество анализируемых симметричных элементов. Приведем предлагаемый алгоритм нахождения минимального количества сигнатурных анализаторов, необходимых для реализации заданного локально-симметричного теста. Входными данными для него являются:

– базовый неразрушающий тест $A = (d_1(o_{11}, o_{12}, \ldots, o_{1P_1})$; $d_2(o_{21}, o_{22}, \ldots, o_{2P_2})$; …; $d_K(o_{K1}, o_{K2}, \ldots, o_{KP_K}))$, где K – количество маршевых элементов в тесте; P_i – количество операций в i-м маршевом элементе; o_{ij} – j-я операция i-го маршевого элемента ($o_{ij} \in \{r_a, r_{\bar{a}}, w_a, w_{\bar{a}}\}$); d_i – направление адресации i-го маршевого элемента ($d_i \in \{\Uparrow, \Downarrow\}$);

– список симметричных элементов $S = (s_1, s_2, \ldots, s_M)$, где s_i ($i = 1, 2, \ldots, M$) имеет вид $(O_i^I)(O_i^{II}) = (o_{i1}^I, o_{i2}^I, \ldots, o_{iR}^I)(o_{i1}^{II}, o_{i2}^{II}, \ldots, o_{iR}^{II})$, а M – количество симметричных элементов в тесте. Символами o_{ij}^I и o_{ik}^{II} ($j, k = 1, 2, \ldots, R$) обозначаются операции чтения, составляющие первую и вторую фазы симметричного элемента соответственно, а R равно их количеству;

– список $T = (t_1, t_2, \ldots, t_M)$, хранящий типы симметрии для каждого симметричного элемента из S ($t_i \in \{1, 2\}$).

Предположим, что каждый сигнатурный анализатор имеет фиксированную структуру, задаваемую прямым или обратным полиномом, и может использоваться как для вычисления сигнатуры, так и для ее хранения, а значения сигнатур могут пересылаться между любыми двумя сигнатурными анализаторами. Запишем алгоритм *FindBISTMin*, выполняющий поиск минимального количества сигнатурных анализаторов, которые необходимы для реализации заданного локально-симметричного теста в виде ВАСТ, и генерирующий последовательность их использования.

1. Сформировать список $G = (g_1, g_2, \ldots, g_{2M})$ всех вычисляемых тестом сигнатур: если $t_i = 1$, то $g_{2i-1} = \mathrm{sig}(g_0, P, O_i^I)$ и $g_{2i} = \mathrm{sig}(g_0, P, O_i^{II})$, иначе $g_{2i-1} = \mathrm{sig}(g_0, P, O_i^I)$ и $g_{2i} = \mathrm{sig}(g_{2i-1}, P^{-1}*, O_i^{II})$ ($i = 1, \ldots, M$). Здесь g_0 обозначает начальную сигнатуру, P и $P*$ – прямой и обратный полиномы для сигнатурного анализатора соответственно.

2. На основе G сформировать список $L = (l_1, l_2, \ldots, l_{2M})$ начальных значений для вычисления каждой сигнатуры: если $i/2 \neq 0$ или $i/2 = 0$ и $t_{i/2} = 1$, то $l_i = g_0$, иначе $l_i = g_{i-1}$ ($i = 1, \ldots, 2M$).

3. Для каждого симметричного элемента ($i = 1, \ldots, M$) сформировать список сравнений $C = (c_1, c_2, \ldots, c_M)$ для определения значения, сравниваемого с сигнатурой его второй фазы (если $t_i = 1$, то $c_i = g_0$, , иначе $c_i = g_{2i-1}$); сформировать список хранения сигнатур $W = (w_1, w_2, \ldots, w_M)$, задающий для каждой сигнатуры список операций чтения, во время которых вычисленную сигнатуру или ее промежуточное значение запрещено модифицировать.

4. Проанализировать список G с целью удаления сигнатур одинаковых последовательностей данных. Модифицировать списки C и W: в списке сравнений заменить удаленные сигнатуры на их аналоги, список хранения для оставляемой сигнатуры построить как пересечение списков хранения всех ее дубликатов.

5. Составить матрицу вычисления сигнатур Q с точностью до одного маршевого элемента (это вызвано тем, что сигнатуры пересылаются и сравниваются только на границах маршевых элементов). Матрица Q хранит соответствие между сигнатурой и использованным для ее хранения или вычисления регистром (z^s) или сигнатурным анализатором, заданным прямым (z) или

обратным (z^*) полиномом. Количество столбцов в матрице Q равно K, количество строк соответствует количеству используемых сигнатурных анализаторов и определяется в процессе заполнения. Для этого из списка G выбираются все сигнатуры g_i и помещаются в свободные строки, соответствующие z или z^*. Если строки со свободным сигнатурным анализатором не найдено, то добавляется новая. Затем на основе списка W в матрицу Q добавляются строки, соответствующие регистрам хранения сигнатуры z^s.

6. Выполнить минимизацию количества строк в матрице Q с учетом того, что предложенная архитектура позволяет использовать любой сигнатурный анализатор в качестве регистра хранения и пересылать данные между любыми двумя анализаторами.

7. Сформировать матрицы L' и C' выполнения пересылок между сигнатурными анализаторами и сравнения их значений на основе матрицы Q и списков G, L, C. Сгенерировать матрицу активности сигнатурных анализаторов для каждой операции чтения Q'. Символ «+» в любом элементе этой матрицы q_{ij}^{l} обозначает, что работа i-го сигнатурного анализатора разрешена при выполнении j-й операции чтения теста, а символ «s» – что он используется в этот момент как регистр хранения.

Симметричный сигнатурный анализ позволяет при сжатии обеих фаз любого симметричного элемента заменить сигнатурный анализатор z на z^* и наоборот. При параллельном анализе нескольких симметричных элементов такая замена может позволить сократить количество используемых сигнатурных анализаторов. Поэтому шаги 3–7 следует повторить, рассмотрев все комбинации замен сигнатурных анализаторов на обратные.

2.2. Результаты работы алгоритма FindBISTMin

Рассмотрим работу алгоритма *FindBISTMin* на примере теста *SMarch C-*$_2$. Исходные данные: базовый неразрушающий тест приведен в формуле (2); список $S = (((o_{1,1}), (o_{3,1})), ((o_{2,1}), (o_{4,1})), ((o_{2,1}), (o_{5,1})), ((o_{3,1}), (o_{5,1})))$; $T = (2, 2, 1, 2)$.

1. Строим список вычисляемых сигнатур G:

$$g_1 = (g_0, P, (o_{1,1})), \quad g_3 = (g_0, P, (o_{2,1})), \quad g_5 = (g_0, P, (o_{2,1})), \quad g_7 = (g_0, P, (o_{3,1})),$$
$$g_2 = (g_1, P^*, (o_{3,1})), \quad g_4 = (g_3, P^*, (o_{4,1})), \quad g_6 = (g_0, P, (o_{5,1})), \quad g_8 = (g_7, P^*, (o_{5,1})).$$

2. Формируем список $L = (g_0, g_1, g_0, g_3, g_0, g_0, g_0, g_7)$.

3. Формируем списки $C = (g_0, g_0, g_5, g_0)$, $W = ((o_{2,1}), (o_{3,1}), (o_{3,1}, o_{4,1}, o_{5,1}), (o_{4,1}))$.

4. Удаляем из списка G сигнатуру g_5, совпадающую с g_3. Заменяем g_5 на g_3 в списках сравнений и хранения: $C = (g_0, g_0, g_3, g_0)$, $W = ((o_{2,1}), (o_{3,1}, o_{4,1}, o_{5,1}), –, (o_{4,1}))$.

5. Составляем матрицу вычисления сигнатур Q. В процессе ее построения были добавлены четыре строки, соответствующие одному сигнатурному анализатору z, одному z^* и двум регистрам хранения z^s:

$$Q = \begin{array}{c} \\ z_1 \\ z_2^* \\ z_3^s \\ z_4^s \end{array} \begin{array}{c} \begin{array}{ccccc} M_1 & M_2 & M_3 & M_4 & M_5 \end{array} \\ \begin{bmatrix} g_1 & g_3 & g_7 & & g_6 \\ & & g_2 & g_4 & g_8 \\ & g_1 & g_3 & g_3 & g_3 \\ & & & g_7 & \end{bmatrix} \end{array}.$$

6. Минимизируем количество строк матрицы Q путем переноса значений из нижних строк в свободные места верхних строк. Это позволяет избавиться от регистра хранения z_4^s:

$$Q = \begin{array}{c} \\ z_1 \\ z_2^* \\ z_3^s \end{array} \begin{array}{c} \begin{array}{ccccc} M_1 & M_2 & M_3 & M_4 & M_5 \end{array} \\ \begin{bmatrix} g_1 & g_3 & g_7 & g_7 & g_6 \\ & & g_2 & g_4 & g_8 \\ & g_1 & g_3 & g_3 & g_3 \end{bmatrix} \end{array}.$$

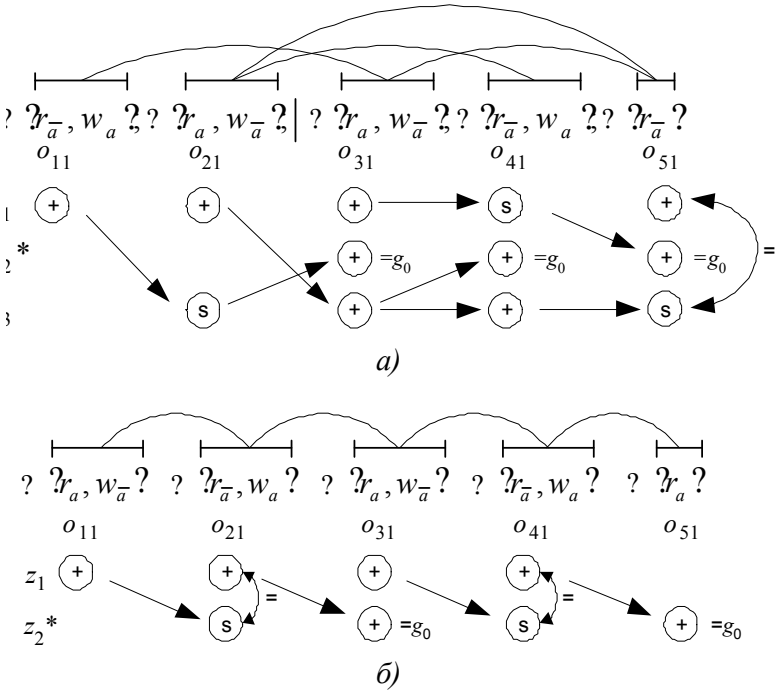

Рис. 2. Схема работы BACT для неразрушающих тестов *SMarch C-*:
а) $BACT_3$ для теста *SMarch C-*$_2$; *б)* $BACT_2$ для теста *SMarch C-*$_1$

Таблица 1

Локально-симметричные неразрушающие тесты для BACT

Тип BACT	Название	Формальная запись теста
$BACT_2$	*sMATS*$_1$	$\Uparrow\left(r_a^{[1]}, w_{\bar{a}}\right);\left\vert^{[1]}\right. \Uparrow\left(r_{\bar{a}}^{[1]}\right)$
	sMATS$_{2x}$	$\Downarrow\left(r_{\bar{a}}^{[1]}\right);\left\vert^{[1]}\right. \Uparrow\left(r_a^{[1,2]}, w_{\bar{a}}\right);\left\vert^{[2]}\right. \Uparrow\left(r_{\bar{a}}^{[2]}\right)$
	sMATS+$_1$	$\Uparrow\left(r_a^{[1]}, w_{\bar{a}}\right);\left\vert^{[1]}\right. \Uparrow\left(r_{\bar{a}}^{[1]}, w_a\right)$
	sMATS+$_{2x}$	$\Downarrow\left(r_a^{[1]}\right);\left\vert^{[1]}\right. \Uparrow\left(r_a^{[1,2]}, w_{\bar{a}}\right);\left\vert^{[2]}\right. \Downarrow\left(r_{\bar{a}}^{[2]}, w_a\right)$
	sMarch X$_1$	$\Uparrow\left(r_a^{[1,2]}, w_{\bar{a}}\right);\left\vert^{[1]}\right. \Downarrow\left(r_{\bar{a}}^{[2]}, w_a\right);\left\vert^{[1]}\right. \Downarrow\left(r_a^{[2]}\right)$
	sMarch X$_{2x}$	$\Uparrow\left(r_a^{[1]}\right);\Uparrow\left(r_a^{[1]}, w_{\bar{a}}\right);\left\vert^{[1]}\right. \Downarrow\left(r_a^{[2]}, w_a\right);\Downarrow\left(r_a^{[1]}\right)$
	sMarch Y$_1$	$\Uparrow\left(r_a, w_{\bar{a}}, r_a^{[1]}\right);\left\vert^{[1]}\right. \Downarrow\left(r_a^{[2]}, w_{\bar{a}}, r_{\bar{a}}^{[1]}\right);\left\vert^{[2]}\right. \Downarrow\left(r_{\bar{a}}^{[2]}\right)$
	sMarch Y$_{2x}$	$\Uparrow\left(r_a^{[1,2]}\right);\left\vert^{[1]}\right. \Uparrow\left(r_a, w_{\bar{a}}, r_a\right);\Downarrow\left(r_{\bar{a}}^{[1]}, w_a, r_{\bar{a}}\right);\left\vert^{[2]}\right. \Downarrow\left(r_a^{[2]}\right)$
	sMarch C-$_3$	$\Uparrow\left(r_a^{[1]}, w_{\bar{a}}\right);\Uparrow\left(r_a^{[1]}, w_{\bar{a}}\right);\left\vert^{[1]}\right. \Downarrow\left(r_{\bar{a}}^{[1,2]}, w_a\right);\left\vert^{[2]}\right. \Downarrow\left(r_a^{[1]}, w_{\bar{a}}\right);\Downarrow\left(r_{\bar{a}}^{[2]}\right)$
	sMarch U$_1$	$\Uparrow\left(r_a^{[1]}, w_{\bar{a}}, r_a^{[1]}, w_{\bar{a}}\right);\left\vert^{[1]}\right. \Uparrow\left(r_a^{[2]}, w_{\bar{a}}\right);\left\vert^{[2]}\right. \Downarrow\left(r_a^{[1]}, w_{\bar{a}}, r_a^{[1]}, w_{\bar{a}}\right);\Downarrow\left(r_{\bar{a}}^{[2]}, w_a\right)$
$BACT_3$	*sMATS*++$_1$	$\Uparrow\left(r_a^{[1]}, w_{\bar{a}}\right);\left\vert^{[1]}\right. \Downarrow\left(r_a^{[1,2]}, w_{\bar{a}}, r_a^{[2]}\right);\left\vert^{[2]}\right.$
	sMATS++$_{2x}$	$\Downarrow\left(r_a^{[1]}\right);\left\vert^{[1]}\right. \Uparrow\left(r_a^{[1,2]}, w_{\bar{a}}\right);\left\vert^{[2]}\right. \Downarrow\left(r_{\bar{a}}^{[2,3]}, w_a, r_{\bar{a}}^{[3]}\right);\left\vert^{[3]}\right.$
	sMarch X$_{3x}$	$\Uparrow\left(r_a^{[1]}\right);\left\vert^{[1]}\right. \Uparrow\left(r_a^{[1,2]}, w_{\bar{a}}\right);\left\vert^{[2]}\right. \Downarrow\left(r_a^{[2,3]}, w_{\bar{a}}\right);\left\vert^{[3]}\right. \Uparrow\left(r_a^{[3]}\right)$
	sMarch Y$_3$	$\Uparrow\left(r_a^{[1]}, w_{\bar{a}}, r_a^{[1,2]}\right);\left\vert^{[1]}\right\vert^{[2]} \Downarrow\left(r_a^{[2,3]}, w_{\bar{a}}, r_{\bar{a}}^{[3,4]}\right);\left\vert^{[3]}\right\vert^{[4]} \Uparrow\left(r_{\bar{a}}^{[3]}\right)$
	sMarch U$_2$	$\Uparrow\left(r_a^{[1,2]}, w_{\bar{a}}, r_a^{[1]}, w_{\bar{a}}\right);\left\vert^{[1]}\right\vert^{[2]} \Uparrow\left(r_a^{[2,3]}, w_{\bar{a}}\right);\left\vert^{[3]}\right. \Downarrow\left(r_a^{[3,4,5]}, w_{\bar{a}}, r_a^{[4]}, w_{\bar{a}}\right);\left\vert^{[4]}\right\vert^{[5]} \Downarrow\left(r_{\bar{a}}^{[5]}, w_a\right)$

7. На основе матрицы Q и списков G, L, C получаем

$$L' = \begin{array}{c} \\ z_1 \\ z_2{*} \\ z_3^s \end{array} \begin{array}{c} \begin{array}{ccccc} M_1 & M_2 & M_3 & M_4 & M_5 \end{array} \\ \begin{bmatrix} & & & z_1 & \\ & & z_3 & z_3 & z_1 \\ & z_1 & z_1 & z_3 & z_3 \end{bmatrix} \end{array}; \quad C' = \begin{array}{c} \\ z_1 \\ z_2{*} \\ z_3^s \end{array} \begin{array}{c} \begin{array}{ccccc} M_1 & M_2 & M_3 & M_4 & M_5 \end{array} \\ \begin{bmatrix} & & & & \\ & & = g_0 & = g_0 & = g_0 \\ & & & & = z_1 \end{bmatrix} \end{array};$$

$$Q' = \begin{array}{c} \\ z_1 \\ z_2{*} \\ z_3^s \end{array} \begin{array}{c} \begin{array}{ccccc} o_{11} & o_{21} & o_{31} & o_{41} & o_{51} \end{array} \\ \begin{bmatrix} + & + & + & \text{s} & + \\ & & + & + & + \\ & + & + & \text{s} & \text{s} \end{bmatrix} \end{array}.$$

Повторение шагов 3–7 после замены сигнатурных анализаторов на анализаторы, задаваемые обратными полиномами, не привело к нахождению способа реализации *SMarch C-*$_2$ с помощью меньшего их количества.

Значения из полученных матриц L', C' и Q' позволяют однозначно формировать последовательность команд по сжатию данных на сигнатурных анализаторах, пересылке данных между ними и сравнению получаемых сигнатур. Эти команды определяют алгоритм работы ВАСТ для реализации локально-симметричного неразрушающего теста. Схема работы ВАСТ для *SMarch C-*$_2$ показана на рис. 2, *а*. Данные, считываемые первыми операциями чтения o_{11} и o_{21}, сжимаются только сигнатурным анализатором z_1, операцией o_{31} – анализаторами z_1 и z_2*, и т. д. Полученные сигнатуры могут передаваться между любыми двумя сигнатурными анализаторами, сравниваться между собой или с начальной нулевой сигнатурой. Так, сигнатура, полученная после первого маршевого элемента, передается в сигнатурный анализатор z_3. После третьего маршевого элемента сигнатура из z_2* сравнивается с начальной сигнатурой g_0, а после последнего маршевого элемента сигнатуры из анализаторов z_1 и z_3 сравниваются между собой.

Алгоритм *FindBISTMin* был реализован и опробован для различных сгенерированных ранее локально-симметричных тестов. Схема работы ВАСТ для реализации *SMarch C-*$_1$ показан на рис 2, *б*. В результате автоматического анализа были выделены два типа ВАСТ, способных реализовать большинство из существующих неразрушающих тестов: с двумя (прямым и обратным) и тремя (двумя прямыми и одним обратным) сигнатурными анализаторами. Обозначим эти реализаторы ВАСТ_2 и ВАСТ_3 и приведем список тестов, реализуемых с их помощью (табл. 1). Рассмотренный выше тест *SMarch C-*$_2$ может быть реализован только на ВАСТ_3, а *SMarch C-*$_1$ – на ВАСТ_3 и ВАСТ_2.

3. Архитектура встроенной аппаратуры самотестирования ОЗУ

Реализуем ВАСТ_2 и ВАСТ_3 аппаратно и сравним их эффективность с существующими ВАСТ, реализующими традиционные неразрушающие тесты [5, 6]. В статье [3] было показано, что покрывающая способность локально-симметричных тестов не ниже, чем у существующих неразрушающих тестов. Поэтому основным параметром, позволяющим оценить эффективность реализации предлагаемых ВАСТ, являются аппаратные затраты на их реализацию.

Существует два подхода к реализации ВАСТ ОЗУ, отличающихся типом устройства управления – с программируемым или жесткозаданным алгоритмом. ВАСТ второго типа (hardware-based) обычно реализуется в виде конечного автомата [8, 9]. Однако существенным недостатком такого подхода является потеря гибкости. При любых, даже незначительных, изменениях теста требуется изменение устройства управления, что приводит к изменению всей ВАСТ. ВАСТ ОЗУ с микропрограммным управлением (microcode-based) [10, 11] позволяет проводить тестирование на основании заданного набора инструкций (микропрограммы). Данный тип ВАСТ допускает изменения теста без изменения самого контроллера путем перепрошивки памяти программ. Этот факт определил более широкое распространение перепрограммируемых ВАСТ [12].

В состав обобщенной архитектуры ВАСТ ОЗУ с микропрограммным управлением (рис. 3) входят: генератор тестовых данных (ГТД), анализатор тестовых данных (АТД), устройство управления (УУ). ГТД формирует тестовые воздействия, подаваемые на входы ОЗУ в режиме тестирования. Он состоит из двух главных частей – генератора адресов и генератора данных. Задачей анализатора тестовых реакций является поиск ошибок в выходных данных схемы ОЗУ. Устройство управления контролирует работу всех компонентов ВАСТ. Память команд используется для хранения кодов микропрограмм для ВАСТ.

Часть интерфейсных линий ВАСТ подключается к тестируемой схеме ОЗУ: к входным шинам адреса, данных и управления, к выходной шине данных. Значения на выходах ОЗУ доступны ВАСТ для анализа на наличие ошибок и формирования новых тестовых данных. Другая часть линий используется для управления самой ВАСТ: «Запуск» инициирует работу схемы тестирования, для сигнализации о завершении работы используется выход «Завершено», а выход «Ошибка» – для выставления признака обнаружения неисправностей в тестируемой схеме.

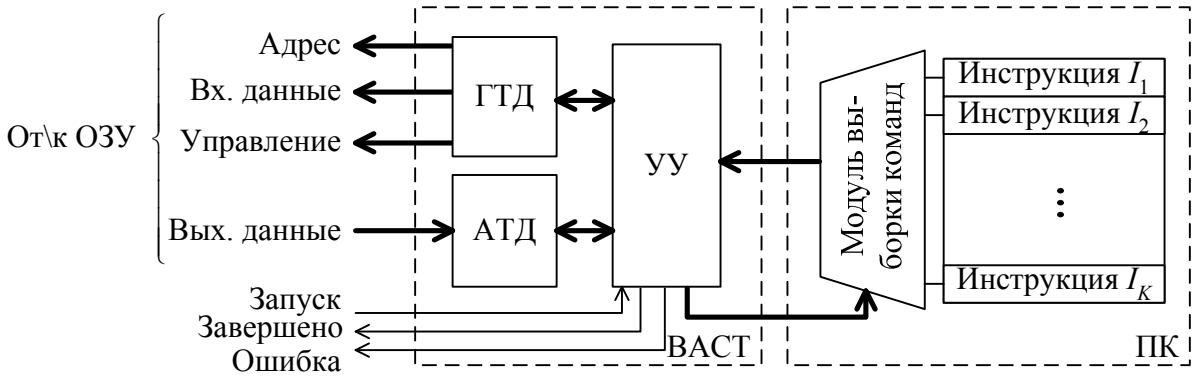

Рис. 3. Обобщенная архитектура ВАСТ ОЗУ

Аппаратные затраты для модулей УУ и ГТД не будут существенно различаться для рассматриваемых тестов, поскольку в их основу положены базовые неразрушающие тесты. Поэтому большое внимание при проектировании неразрушающих ВАСТ было уделено модулю АТД.

Минимальными затратами характеризуется ВАСТ с одним сигнатурным анализатором с возможностью прямого и обратного счета (назовем ее $BACT_1$). Такая схема описана в работе [5], она позволяет реализовать любой традиционный неразрушающий тест. Кроме сигнатурного анализатора модуль АТД в этом случае должен содержать схему сравнения полученной сигнатуры с начальной. $BACT_2$ пригодна для реализации более сложных тестов за счет использования двух сигнатурных анализаторов, задаваемых прямым и обратным полиномами. АТД такой ВАСТ можно использовать для работы с последовательностями данных обоих типов симметрии (см. 1.2), причем последовательности могут частично перекрываться, так что вторая часть последовательности первого типа сжимается одновременно с первой частью последовательности второго типа или одновременно сжимаются первые части последовательностей любого типа.

Схема модуля АТД $BACT_3$ представлена на рис. 4 (управляющие линии не показаны). Сигнатурные анализаторы z_1 и z_2 задаются прямыми полиномами, а z_3* – обратным. Данные для сжатия от ОЗУ поступают на их входы D, а входы LD используются для загрузки значения сигнатуры с других анализаторов. Выход АТД формируется либо в результате сравнения значений двух сигнатур (для последовательностей с первым типом симметрии), либо в результате сравнения значения сигнатуры с начальным значением g_0. Главным преимуществом такой ВАСТ является возможность одновременного анализа до трех симметричных элементов.

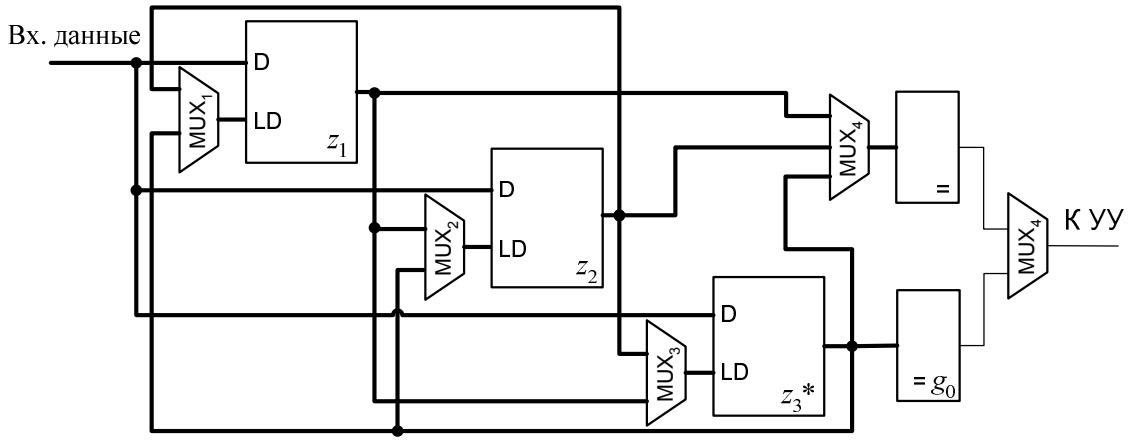

Рис. 4. Схема модуля АТД с тремя сигнатурными анализаторами

4. Оценка эффективности различных схем неразрушающей ВАСТ ОЗУ

Для описания всех ВАСТ, рассматриваемых в данном исследовании, использовался язык VHDL [7], а для синтеза – коммерческий синтезатор полузаказных СБИС Synplicity Synplify ASIC версии 3.0.4. Выдаваемая этим синтезатором информация о площади, занимаемой компонентами, приводится в условных единицах и не может быть использована для получения абсолютных оценок, но вполне пригодна для сравнительного анализа.

Результаты сравнения площади кристалла, занимаемой различными компонентами спроектированных ВАСТ, показаны на рис. 5. Размер круговых диаграмм соответствует затратам на реализацию всей ВАСТ. Очевидно, что дальнейшее увеличение количества сигнатурных анализаторов ведет к росту аппаратных ресурсов и является нецелесообразным.

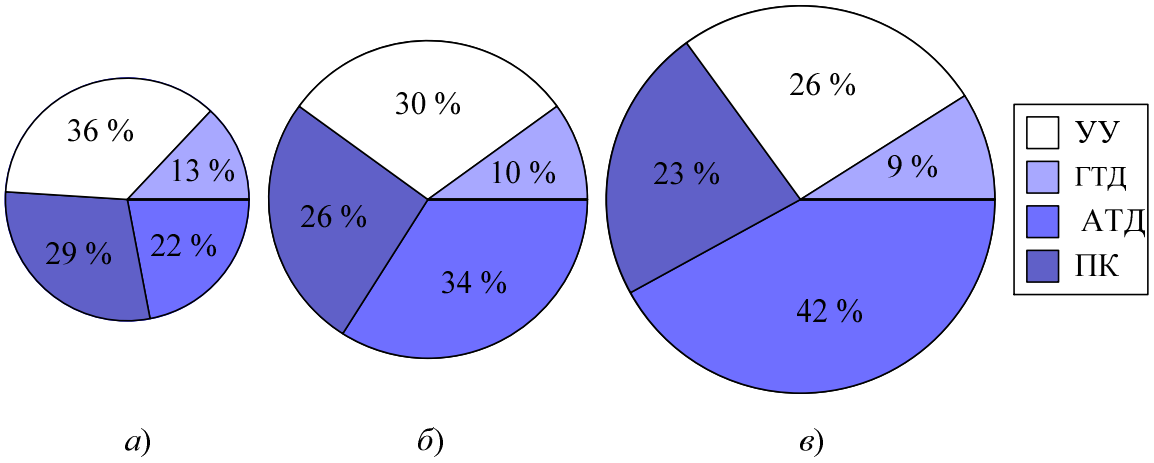

Рис. 5. Соотношение аппаратных затрат на реализацию компонентов ВАСТ: *а*) $ВАСТ_1$; *б*) $ВАСТ_2$; *в*) $ВАСТ_3$

Важным параметром, который характеризует эффективность ВАСТ, является площадь кристалла, занимаемая ею, по отношению к площади кристалла ОЗУ. В табл. 2 приведены результаты синтеза созданных ВАСТ и модулей ОЗУ небольшого объема. $ВАСТ_1$, применяемая для реализации традиционных неразрушающих тестов, требует минимальных аппаратных затрат по сравнению с другими ВАСТ. Однако ресурсоемкость $ВАСТ_2$ отличается от $ВАСТ_1$ лишь незначительно и, учитывая большую эффективность локально-симметричных тестов, применение ее целесообразно. Дальнейшее увеличение объема тестируемого ОЗУ еще больше снижает относительные затраты на реализацию ВАСТ, что приводит к возможности применения и $ВАСТ_3$.

Таблица 2

Относительная площадь кристалла, занимаемая ВАСТ, для ОЗУ разных объемов

Объем ОЗУ, кБ	Площадь кристалла, занимаемая ВАСТ; % от общей площади ОЗУ		
	$ВАСТ_1$	$ВАСТ_2$	$ВАСТ_3$
1	0,98	1,15	1,52
2	0,43	0,57	0,76
4	0,21	0,29	0,38

Заключение

На основе полученных данных можно сделать вывод о целесообразности использования $ВАСТ_2$ даже для ОЗУ небольшого объема. Незначительный, по сравнению с $ВАСТ_1$, рост аппаратных затрат компенсируется сокращением времени и повышением достоверности тестирования. При необходимости еще большего повышения достоверности за счет снижения вероятности маскирования ошибок можно использовать $ВАСТ_3$.

Список литературы

1. Mazumder P., Chakraborty K. Testing and Testable Design of High-Density Random-Access Memories. – Chichester: Addison-Wesley, 1997. – 448 p.

2. Marinescu M. Simple and Efficient Algorithms for Functional RAM Testing // IEEE International Test Conference: Proc. IEEE Computer Society. – Philadelphia, PA, USA, 1982. – P. 236–239.

3. Занкович А.П., Ярмолик В.Н. Неразрушающее тестирование ОЗУ на основе анализа симметрии выходных данных // Автоматика и телемеханика. – 2003. – № 9. – С. 141–154.

4. Zankovich A.P., Yarmolik V.N. Aliasing minimization of transparent memory testing // Новые информационные технологии = New Information Technologies: Мат. V Междунар. науч. конф., Минск, 29–30 октября 2002 г.: В 2 т. – Минск, 2002. – Т. 1. – С. 71–76.

5. Nicolaidis M. Transparent BIST for RAMs // IEEE International Test Conference: Proc. – Baltimore, Maryland, USA, 1992. – P. 596–607.

6. Hellebrand S., Wundelich H.-J., Yarmolik V.N. Symmetric Transparent BIST for RAMs // IEEE Design Automation and Test in Europe Conference (DATE'99): Proc. – Munich, Germany, 1999. – P. 702–707.

7. Бибило П.Н. Основы языка VHDL: 2-е изд. – М.: Солон-Р, 2002. – 224 с.

8. Zoriah Y., Dey S., Rodgers M. Test of Future System-on Chip // IEEE International Test Conference: Proc. – Atlantic City, NJ, USA, 2000. – P. 392–398.

9. Industrial BIST for Embedded RAMs / P. Camurati, P. Prinetto, M.S. Reorda et al. // IEEE Design and Test of Computers. – 1995. – Vol. 12. – № 3. – P. 86–95.

10. Saluja K.K., Sng S.H., Kinoshita K. Built-In Self-Test RAM: A practical alternative // IEEE Design and Test of Computer. – 1987. – Vol. 4. – № 1. – P. 42–51.

11. A Programmable BIST Core For Embedded DRAM / C.-T. Huang, J.-R. Huang, C.-F. Wu et al. // IEEE Design and Test of Computers. – Vol. 16. – № 1. – 1999. – P. 59–70.

12. Zarrineh K., Upadhyaya S.J. On Programmable Memory Built-In Self Test Architectures // IEEE Design Automation and Test in Europe Conference (DATE'99): Proc. – Munich, Germany, 1999. – P. 708–713.

Поступила 10.06.05

Белорусский государственный университет
информатики и радиоэлектроники,
Минск, Бровки, 6
e-mail: kanc@bsuir.unibel.by

A.P. Zankovich, V.N. Yarmolik

TRANSPARENT BUILD-IN SELF-TEST FOR RAMS BASED ON LOCAL SYMMETRICAL MARCH TESTS

The article presents comparative analysis of several RAM Build-In Self-Test (BIST) types based on transparent March tests. Algorithm of generating BIST for local symmetrical algorithms is proposed. Two types of BIST that are suitable to perform most of existing transparent tests were obtained: $BIST_2$ and $BIST_3$. Estimation of hardware cost for $BIST_2$ and $BIST_3$ shows advisability of their use for embedded memory testing.