Анализ и синтез маршевых тестов запоминающих устройств

В. Н. Ярмолик , д. т. н., профессор, профессор кафедры программного обеспечения информационных технологий

E-mail: Yarmolik@bsuir.by

Белорусский государственный университет информатики и радиоэлектроники, ул. П. Бровки, д. 6, 220013, г. Минск, Республика Беларусь

В. А. Леванцевич, м. т. н., старший преподаватель кафедры программного обеспечения информационных технологий

E-mail: Lvn@bsuir.by

Белорусский государственный университет информатики и радиоэлектроники, ул. П. Бровки, д. 6, 220013, г. Минск, Республика Беларусь

Д. В. Деменковец, м. т. н., старший преподаватель кафедры программного обеспечения информационных технологий

E-mail: Demenkovets@bsuir.by

Белорусский государственный университет информатики и радиоэлектроники, ул. П. Бровки, д. 6, 220013, г. Минск, Республика Беларусь

Аннотация. В статье показывается актуальность тестирования запоминающих устройств современных вычислительных систем. Представляются математические модели неисправностей запоминающих устройств и эффективность их обнаружения, в частности, сложных кодочувствительных неисправностей типа PNPSFk, на базе классических маршевых тестов. Приводятся предельные оценки полноты покрытия подобных неисправностей в зависимости от количества запоминающих ячеек, участвующих в неисправности. Обосновывается необходимость синтеза маршевых тестов, характеризующихся высокой эффективностью обнаружения PNPSFk неисправностей. Определяется понятие примитива, обеспечивающего условия активизации и обнаружения различных видов PNPSFk. Приводятся примеры анализа и синтеза маршевых тестов, имеющих различную полноту покрытия PNPSFk неисправностей. Синтезируется маршевый тест March OP, характеризующийся максимальной полнотой покрытия неисправностей PNPSFk и имеющий минимальную временную сложность по сравнению с известными маршевыми тестами, обеспечивающими такую же полноту покрытия сложных неисправностей запоминающих устройств.

Ключевые слова: тестирование вычислительных систем, неисправности памяти, кодочувствительные неисправности, маршевые тесты, псевдоисчерпывающие тесты

Для цитирования: Ярмолик, В. Н. Анализ и синтез маршевых тестов запоминающих устройств / В. Н. Ярмолик, В. А. Леванцевич, Д. В. Деменковец // Цифровая трансформация. – 2021. – № 2 (15). – С. 45–55.

Analysis and Synthesis March Memory Tests

V. N. Yarmolik , Doctor of Science (Technical), Professor, Professor of Information Technology Software Department

E-mail: Yarmolik@bsuir.by

Belarusian State University of Informatics and Radioelectronics, 6 P. Brovka St., 220013 Minsk, Republic of Belarus

V. A. Levantsevich, Master of Science (Technology), Senior Lecturer of the Information Technology

E-mail: Lvn@bsuir.by

Belarusian State University of Informatics and Radioelectronics, 6 P. Brovka St., 220013 Minsk, Republic of Belarus

D. V. Demenkovets, Master of Science (Technology), Senior Lecturer of the Information Technology

E-mail: Demenkovets@bsuir.by
Belarusian State University of Informatics and Radioelectronics,
6 P. Brovka St., 220013 Minsk, Republic of Belarus

Abstract. The paper shows the relevance of testing storage devices in modern computing systems. Mathematical models of memory device faults and the efficiency of their detection, in particular, complex pattern sensitive faults of the PNPSFk type, based on classical march memory tests are presented. Limit estimates are given for the completeness of coverage of such faults depending on the number of memory cells involved in the fault. The necessity of synthesis of memory march tests characterized by high efficiency of PNPSFk failure detection is substantiated. The concept of a primitive providing conditions for activation and detection of various types of PNPSFk is defined. Examples of analysis and synthesis of memory march tests with different coverage of PNPSFk faults are given. The March OP memory test is synthesized, which is characterized by the maximum completeness of PNPSFk fault coverage and has the lowest time complexity compared to the known memory march tests, which provide the same comprehensiveness of coverage of complex memory device faults.

Key words: computer systems testing, memory faults, pattern sensitive faults, march tests, pseudo-exhaustive tests
For citation: Yarmolik V. N., Levantsevich V. A., Demenkovets D. V. Analysis and Synthesis March Memory Tests. *Cifrovaja transformacija* [Digital transformation], 2021, 2 (15), pp. 45–55 (in Russian).

Введение. Проблема тестирования запоминающих устройств современных вычислительных систем, таких как *встроенные системы (embedded systems), системы на кристалле (systems-on-a-chip) и сети на кристалле (nets-on-a-chip)* является весьма актуальной задачей [1–3]. Для подобных вычислительных систем характерным является неуклонное увеличение емкости их запоминающих устройств, удельный вес которых достигает 94% занимаемой системой площади кристалла [4, 5]. Наряду с увеличением емкости запоминающих устройств (памяти) возрастают требования к их надежности, что достигается применением эффективных методов тестового диагностирования [6].

Тестирование запоминающих устройств с целью обнаружения различных их неисправностей предполагает запись в запоминающее устройство всевозможных его состояний и их считывание, что приводит к нереально большой сложности теста, так как сложность самой тестовой процедуры пропорциональна величине 2^N, где N – емкость памяти в битах. Поэтому в настоящее время широко применяются на практике и по-прежнему разрабатываются тесты, которые имеют существенно меньшую сложность, как правило, линейно зависящую от емкости памяти N [6]. Подобные тесты имеют общее название *маршевые тесты (march tests)* и доминируют в практических приложениях [7, 8]. Подобные тесты наряду с приемлемым уровнем обнаружения неисправностей памяти характеризуются простотой реализации, как временной, так и аппаратурной, что весьма важно для *встроенных средств самотестирования (built-in self-test – BIST)* [8].

В общем случае маршевый тест состоит из конечного числа *маршевых элементов (march elements)* [6–11]. В свою очередь, каждый маршевый элемент содержит символ, определяющий порядок формирования *адресной последовательности (address sequence)* для тестируемого запоминающего устройства, где символ ⇧ определяет последовательный перебор адресов памяти по возрастанию, символ ⇩ определяет последовательный перебор адресов по убыванию и сочетание двух символов ⇧⇩ означает формирование адресов по убыванию либо по возрастанию. Следует отметить, что убывающая последовательность адресов (⇩) представляет собой последовательность, которая формируется в обратном порядке по сравнению с возрастающей последовательностью (⇧). Маршевый элемент содержит последовательность операций чтения (*read* – *r*) и записи (*write* – *w*), заключенных в круглые скобки и разделяемых точкой с запятой. Каждая операция представляет собой элемент из следующего набора: *r*0 – операция чтения содержимого запоминающего элемента (ячейки) с ожидаемым значением 0, *r*1 – операция чтения с ожидаемым значением 1, *w*0 – операция записи 0 в ячейку памяти, *w*1 – операция записи 1 в ячейку. Переход к следующей ячейке осуществляется только после выполнения всех операций в текущем маршевом элементе [6–11]. Тест *MATS* является простейшим примером семейства маршевых тестов и имеет следующий вид: {⇧⇩(*w*0); ⇧(*r*0,*w*1); ⇩(*r*1)}. Данный тест состоит из трех маршевых элементов и имеет сложность 4*N*. Первый маршевый элемент ⇧⇩(*w*0) называется *фазой инициализации (initialization phase)*, применяемой для записи *начального состояния*

(background) запоминающего устройства. Остальные две фазы теста ⇧(*r*0,*w*1) и ⇩(*r*1) являются основными фазами, обеспечивающими обнаруживающую способность данного теста. Таким образом, элементы маршевого теста, а именно, их вид, количество и порядок использования в тесте и определяют эффективность теста обнаруживать различные неисправности памяти.

Целью настоящей статьи является анализ эффективности маршевых тестов обнаруживать сложные кодочувствительные неисправности памяти, а также разработка новых маршевых тестов минимальной сложности.

Математические модели неисправностей запоминающих устройств. Существующие модели неисправностей запоминающих устройств можно представить двумя основными группами [6–11]: неисправности электронного обрамления запоминающего устройства, включающего дешифратор адреса, и неисправности массива элементов памяти. Неисправностям массива запоминающих элементов уделяется основное внимание, так как, показано, что тесты, обнаруживающие простейшие неисправности запоминающих ячеек, покрывают неисправности электронного обрамления памяти [6]. Все неисправности матрицы запоминающих ячеек памяти делятся на несколько подмножеств. Чаще всего используется деление неисправностей матрицы запоминающих ячеек в зависимости от их количества в описании конкретной неисправности. К таким неисправностям запоминающих устройств, в первую очередь, относят неисправности, включающие одну ячейку запоминающего устройства, две ячейки и несколько ячеек устройства, в общем случае более чем две, без ограничений на их количество [6].

К неисправностям, затрагивающим одну ячейку ЗУ, относят: константные неисправности *(stuck-at faults (SAF)*). Данные неисправности характеризуются тем, что состояние конкретной ячейки ЗУ (неисправной) постоянно находится в одном из возможных состояний, в состоянии нуля (0) *(stuck-at 0 (SAF0)*) или состоянии (1) *(stuck-at 1 (SAF1)*) независимо от выполненных ранее операций записи в данную ячейку противоположного значения [6, 8]. Переходные неисправности *(transition faults (TF)*) характеризуются невозможностью логического перехода состояния ячейки ЗУ (неисправной) из 0 в 1 (*transition up*) $\langle\uparrow\rangle$ или из 1 в 0 (*transition down*) $\langle\downarrow\rangle$ при выполнении соответствующих операций записи [6].

Среди неисправностей, в которых участвуют две ячейки ЗУ, выделяют инверсные неисправности взаимного влияния (*inversion coupling faults (CFin)*). В подобной неисправности участвует две ячейки ЗУ a_i и a_j $i \neq j$, одна из которых a_i называется агрессором (*aggressor*), а вторая a_j жертвой (*victim*). Расположение агрессора и жертвы в адресном пространстве ЗУ произвольно. При наличии данной неисправности переход из 1 в 0 или из 0 в 1 в агрессоре a_i приводит к инверсии логического значения жертвы a_j [3, 6, 8]. Таким образом, различают два вида инверсных неисправностей $\langle\uparrow, a_j\rangle$ и $\langle\downarrow, a_j\rangle$. Для представления соотношения адресов агрессора (*i*) и жертвы (*j*), в адресном пространстве ЗУ, используют символы ∧ и ∨, причем символ ∧ означает факт того, что адрес агрессора меньше адреса жертвы $i < j$, а символ ∨, наоборот, i > j, больше. Тогда имеем четыре различных инверсных неисправности $\wedge\langle\uparrow, a_j\rangle$, $\wedge\langle\downarrow, a_j\rangle$, $\vee\langle\uparrow, a_j\rangle$ и $\vee\langle\downarrow, a_j\rangle$.

При неисправности прямого действия (*idempotent coupling faults (CFid)*) во время логического перехода из 1 в 0 или из 0 в 1 во влияющей a_i ячейке происходит принудительная установка определенного логического значения 0 или 1 в ячейке жертве a_j, на которую оказывается влияние агрессора [3, 6, 8]. Различают восемь неисправностей прямого действия $\wedge\langle\uparrow,0\rangle$, $\wedge\langle\uparrow,1\rangle$, $\wedge\langle\downarrow,0\rangle$, $\wedge\langle\downarrow,1\rangle$, $\vee\langle\uparrow,0\rangle$, $\vee\langle\uparrow,1\rangle$, $\vee\langle\downarrow,0\rangle$ и $\vee\langle\downarrow,1\rangle$. При анализе эффективности тестов ЗУ анализируется их покрывающая способность для всех 12 неисправностей, в которых участвуют две ячейки (*2-coupling faults*) [12, 13].

Неисправности, затрагивающие несколько ячеек памяти, называются кодочувствительными неисправностями (*pattern sensitive faults (PSF)*) [6]. Для подобных неисправностей логическое состояние одной ячейки ЗУ, называемой базовой (*base cell*), может зависеть от содержимого (0 или 1) или от логических переходов из 1 в 0 или из 0 в 1 соседних ячеек (*neighborhood cells*) ЗУ. Различают два вида кодочувствительных неисправностей: неограниченные (*unrestricted*) и ограниченные (*restricted*), или граничные (*neighborhood (NPSF)*) кодочувствительные неисправности. При тестировании массива ячеек ЗУ обычно придерживаются последней более реальной модели кодочувствительных неисправностей, для которой рассматривается небольшое число $k < 10$ ячеек ЗУ входящих в кодочувствительную неисправность [3,6, 8,14–17]. Среди их многообразия выделяют пассивные кодочувствительные неисправности (*passive NPSF (PNPSF)*), для которых состояние базовой ячейки

не может быть изменено для определенного кода в $k-1$ соседних ячейках [3, 6, 8]. Такие неисправности также называются *k-coupling faults* [18].

В качестве объекта исследования, чаще всего, рассматриваются пассивные кодочувствительные неисправности (*PNPSFk*), где *k* обозначает количество произвольных ячеек ЗУ емкостью N бит, участвующих в конкретной неисправности. Отметим, что результаты, полученные для *PNPSFk*, легко обобщаются и для других классов кодочувствительных неисправностей, в силу того, что *PNPSFk* является наиболее трудно обнаруживаемой моделью неисправностей ЗУ, покрывающей другие виды неисправностей [3, 6].

Выделяют *k* различных классов *PNPSFk* в зависимости от местоположения в адресном пространстве памяти базовой ячейки (*b*) по отношению к соседним ячейкам (*n*). Например, для $k = 4$ существует четыре класса *PNPSF4:* $b_{i0}\, n_{i1}\, n_{i2}\, ni3$; $ni0\, b_{i1}\, n_{i2}\, n_{i3}$; $n_{i0}\, n_{i1}\, b_{i2}\, n_{i3}$; $n_{i0}\, n_{i1}\, n_{i2}\, b_{i3}$, где адреса ячеек в адресном пространстве имеют следующее соотношение $i0 < i1 < i2 < i3$. Каждый класс включает две *PNPSFk* в зависимости от состояния базового элемента, которое не может быть изменено. Например, классу $n_{i0}\, b_{i1}\, n_{i2}\, n_{i3}$ неисправности *PNPSF4*, наряду с другими неисправностями, принадлежат следующие их конкретные виды: $\langle 1,\uparrow,0,0\rangle$, $\langle 0,\uparrow,1,1\rangle$, $\langle 1,\downarrow,0,0\rangle$, $\langle 0,\downarrow,1,1\rangle$. В соседних ячейках возможны любые из 2^{k-1} двоичных наборов, каждый из которых определяет конкретные два неисправных поведения базовой ячейки. В отличие от ранее описанных неисправностей взаимного влияния символы ↑ и ↓ в *PNPSFk* неисправности, так же как и в *TF* неисправности, означают невозможность выполнения соответствующего перехода из нуля в единицу (↑) и наоборот (↓). Соответственно, общее количество *PNPSFk* относящихся к *k* ячейкам ЗУ равняется $2\kappa \times 2^{k-1} = k \times 2^k$, а число всевозможных *PNPSFk* для памяти емкостью *N* бит определяется согласно выражению [3].

$$Q_T(PNPSFk) = 2k2^{k-1}\binom{N}{k} = k2^k\binom{N}{k} \quad (1)$$

Емкость современных запоминающих устройств, как правило, велика и является существенно большей по сравнению с количеством ячеек *k*, участвующих в рассматриваемых кодочувствительных неисправностях. Поэтому всегда справедливо следующее неравенство $N >> k$, которое позволяет оценить количество *PNPSFk*, определяемое соотношением (1). Тогда, уже для $k = 3$ получим, что $Q_T(PNPSF3) \approx N^3$. Для реальных значений *N* приведенная оценка принимает большие значения, показывающие сложность проблемы синтеза тестов запоминающих устройств для обнаружения кодочувствительных неисправностей.

Оценка эффективности обнаружения кодочувствительных неисправностей маршевыми тестами. Как правило, эффективность маршевых тестов оценивается для каждого вида неисправности независимо от эффективности по отношению к другим их разновидностям. При этом показывается только факт 100%-го их обнаружения конкретным тестом, либо невозможность 100%-го обнаружения, без оценки полноты покрытия теста для исследуемого вида неисправности [6].

В работах [6, 9–11] были проведены оценки обнаруживающих способностей маршевых тестов относительно одиночных неисправностей различных классов, приведенных в табл. 1.

Таблица 1. Эффективность обнаружения неисправностей
Table 1. Fault detection efficiency

Название теста	Сложность теста	Обнаруживаемые неисправности								
		AF	*SAF*	*TF*	*CFin*	*CFid*	*TF-CF*	*CFid-CFid*	*CFin-CFin*	*CFid-CFin*
MATS	4N		+							
MATS+	5N	+	+							
MATS++	6N	+	+	+						
March *X*	6N	+	+	+	+					
March *Y*	8N	+	+	+	+		+			
Marching *1/0*	14N	+	+	+	+		+			
March *C*	11N	+	+	+	+	+			+	
March *C–*	10N	+	+	+	+	+			+	
March *A*	15N	+	+	+	+	+		+		
March *B*	17N	+	+	+	+	+	+	+		
Algorithm *B*	17N	+	+	+	+	+	+	+		

Как показывает анализ покрывающих способностей, не все маршевые тесты способны в полной мере обнаруживать различные классы неисправностей. Из приведенной табл. 1 покрывающих способностей тестов видно, что, несмотря на различную структуру и сложность маршевых тестов, открытой проблемой остается обнаружение в полной мере неисправностей взаимного влияния, сложных связанных неисправностей, а также кодочуствительных неисправностей [6].

Развитие методов многократного применения маршевых тестов, с изменяемыми начальными состояниями ячеек памяти и адресной последовательностью, привели к появлению псевдо исчерпывающих тестов памяти [19]. Сущность подобных тестов заключается в формировании в произвольных k из N ячейках памяти всевозможных 2^k двоичных комбинаций. Основой эффективности таких тестов является формирование различных видов орбит, представляющих собой набор двоичных комбинаций в произвольных k из N ячейках памяти. В [19] показано, что в рамках маршевых тестов возможным является формирование 8 видов орбит представленных в табл. 2 и 3.

При описании предыдущей и текущей фаз теста, приведенных в табл. 2 и 3, показана последняя операция записи в фазе, после которой могут быть только операции чтения. В текущей фазе также показана первая операция записи, перед которой возможно использование только операций чтения. Отметим, что орбиты O_0, O_1, O_2 и O_3 формируются фазами, которые инвертируют содержимое запоминающего устройства. Представленные орбиты хорошо описывают основные свойства тестов и широко представлены в классических маршевых тестах [19]. Например, фаза ⇧(r0,w1,w0,w1) теста March *A* и фаза ⇧(r0,w1,w0,w1,r1) теста March LA, так же как и фаза ⇧(r0,w1) теста *MATS ++*, формируют орбиту O_0 [19].

Формирование маршевыми тестами различного рода орбит обеспечивает условие активизации многообразных неисправных состояний памяти. Причем выполнение условия генерирования псевдоисчерпывающих комбинаций в произвольных k из N ячеек памяти, с последующей их проверкой, гарантирует 100%-ое обнаружение различных подмножеств неисправностей [3, 19]. Отметим, что для определения конкретной неисправности необходимо выполнение условия

Таблица 2. Орбиты, формируемые тестами типа MATS ++ и March C–
Table 2. Orbits formed by march tests MATS ++ and March C–

Орбита	O_0	O_1	O_2	O_3
Предыдущая фаза теста	(…, w0, …)	(…, w0, …)	(…, w1, …)	(…, w1, …)
Фаза теста	⇧(…, w1, …)	⇩(…, w1, …)	⇧(…, w0, …)	⇩(…, w0, …)
P_0	000…00	000…00	111…11	111…11
P_1	000…01	100…00	111…10	011…11
P_2	000…11	110…00	111…00	001…11
…	…	…	…	…
P_{k-1}	011…11	111…10	100…00	000…01
P_k	111…11	111…11	000…0	000…00

Таблица 3. Орбиты, формируемые тестами типа March A
Table 3. Orbits formed by march tests March A

Орбита	Q_0	Q_1	Q_2	Q_3
Предыдущая фаза теста	(…, w0, …)	(…, w0, …)	(…, w1, …)	(…, w1, …)
Фаза теста	⇧(…, w1, …, w0, …)	⇩(…, w1, …, w0, …)	⇧(…, w0, …, w1, …)	⇩(…, w0, …, w1, …)
P_0	000…00	000…00	111…11	111…11
P_1	000…01	100…00	111…10	011…11
P_2	000…10	010…00	111…01	101…11
…	…	…	…	…
P_{k-1}	010…00	000…10	101…11	111…01
P_k	100…00	000…01	011…11	111…10

ее *активизации (sensitization)* и ее *обнаружения (detection)* [6].

В общем случае, для кодочувствительных неисправностей условие активизации будет состоять из операции записи 1 в базовую ячейку, выполняющую переход из 0 в 1, и, наоборот, записи 0, осуществляющей переход из 1 в 0, для всех возможных 2^{k-1} состояний в *k–1* соседних ячеек. После выполнения каждого из переходов необходимо выполнение операции чтения из базовой ячейки, что и будет являться условием обнаружения конкретной неисправности. Причем операция чтения может быть как в текущей фазе, так и в последующей, важным является то, что между изменением содержимого базовой ячейки (выполнением перехода) и чтением ее содержимого не было промежуточных операций записи. При подобном тестировании базовой ячейки можно определить все пассивные и статические кодочувствительные неисправности. Для случая активных кодочувствительных неисправностей необходимо для базовой ячейки осуществить операции *r*0 и *r*1 при всевозможных изменениях состояний соседних ячеек. Более подробно рассмотрим эффективность обнаружения кодочувствительных неисправностей *PNPSFk* однократными маршевыми тестами.

Учитывая последовательное обращение к запоминающим ячейкам и единообразных операций записи для них, согласно структуре любого однократного маршевого теста памяти, можно выделить обнаруживаемые и необнаруживаемые подмножества *PNPSFk*. Обнаруживаемые неисправности определяются видом рассмотренных ранее орбит, генерируемых тестом. Например, формирование орбиты O_0 фазой ⇧(*r*0, *w*1) в произвольных *k* ячейках реализует условие активизации *k* неисправностей *PNPSFk* следующего вида ⟨ 0, 0, 0, ...0,↑, 1,1,1,...,1⟩. При наличии соответствующей операции чтения, обеспечивающей условие обнаружения, эти *PNPSFk* являются обнаруживаемыми. При условии, что тест формирует только одну орбиту O_0 фазой ⇧(*r*0,*w*1), остальные $k2^k$–*k PNPSFk* будут относиться к множеству не обнаруживаемых данным тестом неисправностей. Отметим, что любой маршевый тест, по определению, формирует как минимум одну орбиту, которая обеспечивает условие активизации k неисправностей *PNPSFk*. Примером такого теста может быть тест MATS, генерирующий орбиту O_0 фазой ⇧(*r*0,*w*1).

Таким образом, минимальная полнота покрытия однократного маршевого теста, как процентное отношение количества $Q_{MIN}(PNPSFk) = k \times \binom{N}{k}$, обнаруживаемых PNPSFk к их общему числу *QT(PNPSFk)*, принимает вид:

$$FC_{MIN}(PNPSFk) = \frac{Q_{MIN}(PNPSFk)}{Q_T(PNPSFk)} \times 100\% = 1/2^k \times 100\% \quad (2)$$

Маршевые тесты MATS+: {⇧⇩(*w*0); ⇧(*r*0,*w*1); ⇩(*r*1,*w*0)} и MATS++: {⇧⇩(*w*0); ⇧(*r*0,*w*1); ⇩(*r*1,*w*0,*r*0)} формируют по две одинаковые орбиты O_0 и O_3, обеспечивающие активизацию *k* неисправностей *PNPSFk* вида ⟨0,0,0,...0,↑,1,1,1,..., 1⟩ и *k* неисправностей вида ⟨0,0,0,...0,↓,1,1,1,..., 1⟩. Однако, только *MATS++* обеспечивает обнаружение *2k* неисправностей *PNPSFk*, так как, в отличие от *MATS+*, в данном тесте обеспечены условия обнаружения активизированных неисправностей. Операция чтения *r*1 в третьей фазе ⇩(*r*1,*w*0,*r*0) обеспечивает обнаружение неисправностей, ⟨0,0,0,...0,↑,1,1,1,..., 1⟩ активизированных во второй фазе ⇧(*r*0,*w*1), а операция *r*0 в третьей фазе обнаруживает неисправности ⟨0,0,0,...0,↓,1,1,1,..., 1⟩, активизированные в этой же фазе. Тогда полнота покрытия теста *MATS+* будет равняться минимальной полноте покрытия $FC_{MATS+}(PNPSFk) = FC_{MIN}(PNPSFk) = 2^{-k} \times 100\%$, а теста *MATS++* будет в два раза выше, т.е. $FC_{MATS++}(PNPSFk) = 2^{1-k} \times 100\%$. Такую же полноту покрытия *PNPSFk* можно достичь, применяя следующий тест {⇧⇩(*w*0); ⇧(*r*0,*w*1,*r*1,*w*0,*r*0,*w*1)}, во второй фазе которого формируются условия активизации и обнаружения k неисправностей *PNPSFk* вида ⟨0,0,0,...0,↑,1,1,1,..., 1⟩ и k неисправностей вида ⟨0,0,0,...0,↓,1,1,1,..., 1⟩.

Каждая из орбит, представленных в табл. 2, обеспечивает условие активизации конкретного вида *PNPSFk*, а именно O_0: ⟨0,0,0,...,0,↑,1,1,1,...,1⟩, O_1: ⟨1,1,1,...,1,↓,0,0,0,..., 0⟩, O_2: ⟨1, 1, 1, ... , 1, ↓, 0, 0, 0, ... , 0⟩, O_3: ⟨0,0,0,...,0,↓,1,1,1,...,1⟩. В тоже время, орбиты Q_0, Q_1, Q_2 и Q_3 обеспечивают обнаружение двух видов неисправностей, каждая, а именно орбиты Q_0, Q_1, формируют условие активизации неисправностей ⟨0,0,0,...,0,↑,0,0,0,...,0⟩ и ⟨0,0,0,...,0,↓,0,0,0,..., 0⟩, а Q_2, Q_3 ⟨1,1,1,...,1,↑,1,1,1,...,1⟩ и ⟨1,1,1,...,1,↓,1,1,1,...1⟩. Анализ орбит, представленных в табл. 2 и 3 и обнаруживаемых с их помощью *PNPSFk*, позволяет сформулировать следующее утверждение, определяющее множество, состоящее из восьми неисправностей *PNPSFk*, обнаруживаемых однократными маршевыми тестами.

Утверждение 1. К множеству обнаруживаемых маршевыми тестами неисправностей

Таблица 4. Обнаруживаемые неисправности PNPSF4
Table 4. Detected faults PNPSF

Классы *PNPSF4*	Неисправности *PNPSF4*
$b_{i0}\, n_{i1}\, n_{i2}\, n_{i3}$:	⟨↑,0,0,0⟩, ⟨↓,0,0,0⟩, ⟨↑,1,1,1⟩, ⟨↓,1,1,1⟩
$n_{i0}\, b_{i1}\, n_{i2}\, n_{i3}$:	⟨0,↑,0,0⟩, ⟨0,↓,0,0⟩, ⟨1,↑,1,1⟩, ⟨1,↓,1,1⟩, ⟨0,↑,1,1⟩, ⟨0,↓,1,1⟩, ⟨1,↑,0,0⟩, ⟨1,↓,0,0⟩
$n_{i0}\, n_{i1}\, b_{i2}\, n_{i3}$:	⟨0,0,↑,0⟩, ⟨0,0.↓,0⟩, ⟨1,1,↑,1⟩, ⟨1,1,↓,1⟩, ⟨0,0,↑,1⟩, ⟨0,0,↓,1⟩, ⟨1,1,↑,0⟩, ⟨1,1,↓,0⟩
$n_{i0}\, n_{i1}\, b_{i2}\, n_{i3}$:	⟨0,0,0,↑⟩, ⟨0,0,0,↓⟩, ⟨1,1,1,↑⟩, ⟨1,1,1,↓⟩

PNPSFk относятся такие их разновидности, для которых в соседних ячейках до базовой ячейки находятся значения все 0, либо все 1, после базовой также находятся одинаковые значения, причем независимо от значений в соседних ячейках до базовой ячейки.

Для случая k = 4, к обнаруживаемым неисправностям *PNPSF4* относятся неисправности четырех различных классов, представленные в табл. 4. Остальные 40 неисправностей *PNPSF4*: ⟨↑,0,0,1⟩,⟨↑,0,1,0⟩, ⟨↑,0,1,1⟩,.. , ⟨1,1,0,↓⟩ относятся к подмножеству не обнаруживаемых однократными маршевыми тестами неисправностей.

В случае произвольного k для класса с нулевым индексом (i0) базовой ячейки и класса с индексом базовой ячейки (i(k–1)) обнаруживаемыми являются только по 4 неисправности *PNPSFk*, а для остальных классов по 8 неисправностей, как это видно, например, из табл. 4. Соответственно, максимально возможное число *QMAX(PNPSFk*) обнаруживаемых неисправностей *PNPSFk* равняется

$$Q_{MAX}(PNPSFk) = (4 + 8 \times (k-2) + 4) \times \binom{N}{k} = 8 \times (k-1) \times \binom{N}{k} \quad (3)$$

Таким образом, максимально достижимая полнота покрытия для однократного маршевого теста принимает вид

$$FC_{MAX}(PNPSFk) = \frac{Q_{MAX}(PNPSFk)}{Q_T(PNPSFk)} \times 100\% = \frac{k-1}{k \times 2^{k-3}} \times 100\% \quad (4)$$

Вне зависимости от вида и структуры маршевого теста, однократное его применение для произвольного начального состояния памяти и применяемой адресной последовательности не позволяет достичь полноты покрытия больше, чем величина определяемая выражением (4) [3, 8]. Значения полноты покрытия экспоненциально убывают с ростом k, как это видно из табл. 5.

Существенный разброс полноты покрытия неисправностей от минимального до максимального значения, в особенности с ростом k, предопределяет необходимость синтеза тестов обеспечивающих максимальную их эффективность при обнаружении сложных кодочувствительных неисправностей.

Построение маршевых тестов обеспечивающих максимальную полноту покрытия PNPSFk. При синтезе маршевого теста с максимальной полнотой покрытия FC_{MAX}(*PNPSFk*) (4) неисправностей PNPSFk необходимым является обеспечение условий активизации и обнаружения каждого из восьми их видов, определенных в утверждении 1. Эти условия задаются *примитивами*, определяемыми тремя последовательными фазами теста, которые назовем *предыдущая* фаза, *текущая* фаза и *последующая* фаза.

Во-первых, отметим, что предыдущая фаза однозначно определяет начальное состояние k произвольных ячеек памяти, участвующих в конкретной неисправности PNPSFk. Текущая фаза, как правило, реализует условие активизации неисправности и, возможно, ее обнаружения. Третья фаза может не входить в примитив для конкретной неисправности, так как она реализует только условие ее обнаружения, которое может быть реализовано в текущей фазе.

В качестве примера рассмотрим случай неисправности ⟨0,0,0,…,0,↑,1,1,1,…,1⟩. Как

Таблица 5. Полнота покрытия (%) FC_{MAX}(PNPSFk) и FC_{MIN}(PNPSFk)
Table 5. Entirety of coverage (%) FC_{MAX}(PNPSFk) and FC_{MIN}(PNPSFk)

	PNPSF2	*PNPSF3*	*PNPSF4*	*PNPSF5*	*PNPSF6*	*PNPSF7*
FC_{MAX}(PNPSFk)	100	66,66	37,5	20,0	10,41	5,35
FC_{MIN}(PNPSFk)	25	12,5	6,25	3,125	1,5625	0,78125

было показано в предыдущем разделе, для обнаружения данной неисправности необходимо, чтобы предыдущая фаза вида ⇧⇩(…,*w*0,…) обеспечила запись во все ячейки нулевых начальных значений. Также как и в случае орбит, приведенная фаза может иметь произвольный вид при одном ограничении, что последняя операция записи должна быть w0, после которой могут применяться только операции чтения. Текущая фаза ⇧(*r*0,*w*1) обеспечивает условие активизации *k* неисправностей *PNPSFk* вида ⟨0,0,0,…,0,↑,1,1,1,…,1⟩. В последующей фазе ⇧⇩(*r*1,…) операция чтения *r*1 обеспечивает условие обнаружения всех *k* неисправностей ⟨0,0,0,…,0,↑,1,1,1,…,1⟩. Применение такого примитива, состоящего из трех последовательных фаз ⇧⇩(…,w0,…), ⇧(r0,w1) и ⇧⇩(r1,…), гарантирует обнаружение всех *k* неисправностей вида ⟨0,0,0,…,0,↑,1,1,1,…,1⟩. В частном случае, приведенный примитив, представляет собой ранее рассмотренный маршевый тест *MATS*: {⇧⇩(*w*0); ⇧(*r*0,*w*1); ⇩(*r*1)}.

Следует отметить, что конкретная фаза маршевого теста может участвовать в различных примитивах, однако не более чем в трех. Например, в тесте March *C*−: {⇧⇩ (w0); ⇧(*r*0,*w*1); ⇧(*r*1,*w*0); ⇩(*r*0,*w*1); ⇩(*r*1,*w*0);⇧⇩(*r*0)} вторая фаза ⇧(*r*0,*w*1) является предыдущей фазой, третья фаза ⇧(r1,w0) – текущей, и четвертая ⇩(r0,w1) – последующей фазой примитива обеспечивающего активизацию и обнаружение неисправности ⟨1,1,1,…,1,↓,0,0,0,…,0⟩. Та же четвертая фаза ⇩(r0,w1) является текущей фазой примитива, описывающего обнаружение неисправности ⟨1,1,1,…,1,↑,0,0,0, …,0⟩, и предыдущей в примитиве соответствующем неисправности ⟨0,0,0,…,0,↓,1,1,1,…,1⟩. Все множество примитивов и описываемые ими обнаруживаемые *PNPSFk* приведены в табл. 6.

Для конкретного вида PNPSFk неисправностей ⟨0,0,0,…,0,↑,1,1,1,…,1⟩, примитив, обеспечивающий их обнаружение, может состоять из двух фаз, предыдущей фазы ⇧⇩(*w*0) и текущей ⇧(r0,w1,r1), в которой обеспечивается и активизация и обнаружение указанных неисправностей. В обоих случаях минимальная временная сложность примитивов равняется 4*N*. Однако с учетом того, что конкретная фаза маршевого теста может участвовать в нескольких примитивах, средняя и суммарная сложности примитивов, описывающие обнаружение более чем одной неисправности *PNPSFk*, существенно меньше, как это видно на примере теста March *C*−. Временная сложность данного теста равняется 10*N*, а сам тест обнаруживает четыре *PNPSFk*.

Более сложные примитивы могут обеспечить условия активизации и обнаружения не более чем двух видов *PNPSFk*. Это следует из того, что текущая фаза может сформировать только два перехода (↑, ↓) в базовой ячейке для каждого из четырех состояний в соседних ячейках (см. Утверждение 1). Примером подобного примитива является случай обнаружения двух видов *PNPSFk*, а именно ⟨0,0,0,…,0,↑,1,1,1,…,1⟩ и ⟨0,0,0,…,0,↓,1,1,1,…,1⟩, состоящим из предыдущей фазы ⇧⇩(…,*w*0,…) и текущей, вида ⇧(*r*0,*w*1,*r*1,*w*0,*r*0,*w*1). Примитив, представленный фазами ⇧⇩(…,*w*1,…) и ⇩(*r*1,*w*0,*r*0,*w*1,*r*1,*w*0), описывает обнаружение этих же неисправностей ⟨0,0,0,…,0,↑,1,1,1,…,1⟩ примитива ⇧⇩(…,w0,…), ⇩(r0,w1,r1,w0,r0,w1) и ⇧⇩(…,w1,…), и ⇧(r1,w0,r0,w1,r1,w0) обеспечивают обнаружение следующих двух неисправностей ⟨1,1,1,…,1,↑,0,0,0,…,0⟩ и ⟨1,1,1,…,1,↓,0,0,0,…,0⟩.

Используя два из четырех приведенных примитивов, оказывается возможным построение маршевого теста, имеющего такую же покрывающую неисправности PNPSFk способность, как и тест March C−. Примером такого теста может быть тест {⇧⇩(*w*0); ⇧(r0,w1,r1,w0,r0,w1); ⇧(r1,w0,r0,w1,r1,w0)}, обнаруживающий, так же как и March *C*−, четыре неисправности *PNPSFk*.

Для получения максимально возможной полноты покрытия (4), достижимой в рамках однократного маршевого теста, необходимо использовать примитивы, обеспечивающие покрытие *PNPSFk* соответствующих орбитам Q_0, Q_1, Q_2 и Q_3. Аналогично, как и для случая неисправностей, описываемых орбитами O_0, O_1, O_2 и O_3, получим примитивы, состоящие из двух фаз. Примитив, состоя-

Таблица 6. Примитивы, формируемые тестом March C–
Table 6. Primitives generated by the March C–

Примитив	⇧⇩ (w0), ⇧(r0,w1), ⇧(r1,w0)	⇧(r0,w1), ⇧(r1,w0), ⇩(r0,w1)	⇧(r1,w0), ⇩(r0,w1), ⇩(r1,w0)	⇩(r0,w1), ⇩(r1,w0), ⇧⇩(r0)
PNPSFk	⟨0,0,0,…,0,↑, 1, 1,1,…,1⟩	⟨1, 1 ,1,…,1,↓, 0, 0, 0,…, 0⟩	⟨1,1,1,…,1,↑, 0, 0,0,…,0⟩	⟨0,0,0,…,0,↓, 1, 1,1,…,1⟩

щий из предыдущей фазы ⇧⇩(…,*w*0,…) и текущей ⇧⇩(*r*0,*w*1,*r*1,*w*0,*r*0) фазыы, описывает условия обнаружения неисправностей ⟨0,0,0,…,0,↑,0,0,0,…,0⟩, и ⟨0,0,0,…,0,↓,0,0,0,…,0⟩, а примитив ⇧⇩(…,w1,…) и ⇧⇩(*r*1,*w*0,*r*0,*w*1,*r*1) неисправностей ⟨1,1,1,…,1,↑,1,1,1,…,1⟩ и ⟨1,1,1,…,1,↓,1,1,1,…,1⟩. Отметим, что в двух предыдущих примитивах обнаружение неисправностей обеспечивается в текущей фазе каждого из них за счет соответствующих операций чтения. В случае первого примитива, состоящего из предыдущей фазы ⇧⇩(…,*w*0,…) и текущей ⇧⇩(*r*0,*w*1,*r*1,*w*0,*r*0), факт наличия неисправности ⟨0,0,0,…,0,↑,0,0,0,…,0⟩ определяет операция чтения *r*1, а наличие неисправности ⟨0,0,0,…,0,↓,0,0,0,…,0⟩, вторая операция чтения r0 текущей фазы. Присутствие неисправности ⟨0,0,0,…,0,↓,0,0,0,…,0⟩ может быть определено и в последующей фазе, которая должна начинаться с операции чтения r0. Соответственно, можно показать, что примитив, состоящий из предыдущей фазы ⇧⇩(…,*w*0,…), текущей фазы ⇧⇩(*r*0,*w*1,*r*1,*w*0) и последующей ⇧⇩(*r*0,…), обеспечивает ее обнаружение. В тоже время обнаружение неисправностей ⟨1,1,1,…,1,↑,1,1,1,…,1⟩ и ⟨1,1,1,…,1,↓, 1,1,1⟩ также может быть обеспечено примитивом, состоящим из трех фаз, а именно, предыдущей фазы ⇧⇩(…,*w*1,…), текущей фазы ⇧⇩(*r*1,*w*0,*r*0,*w*1) и последующей фазы ⇧⇩(*r*1,…).

Описанные ранее процедуры построения примитивов и их конкретные виды, приведенные выше, позволяют синтезировать маршевые тесты, однократное применение которых позволяет обеспечить максимальную полноту покрытия *PNPSFk*, определяемую соотношением (4). Примером результата такого синтеза может быть тест {⇧⇩(*w*0); ⇧(*r*0,*w*1,*r*1,*w*0,*r*0); ⇩(*r*0,*w*1,*r*1,*w*0,*r*0,*w*1); ⇩(*r*1,*w*0,*r*0,*w*1,*r*1); ⇧(*r*1,*w*0,*r*0,*w*1,*r*1,*w*0)}, сложность которого равняется сложности 23*N* известного теста March PS, имеющего максимальную полноту покрытия в классе *PNPSFk* [20].

Объединение примитивов различной сложности позволяет синтезировать более простой маршевый тест March *OP*, имеющий меньшую временную сложность равную 18*N*. Это тест имеет вид (5), а его примитивы, обеспечивающие обнаружение всевозможных неисправностей *PNPSFk* приведены в табл. 7.

March *OP* {⇧⇩(*w*0); ⇧(*r*0,*w*1); ⇧(*r*1,*w*0); ⇩(*r*0,*w*1); ⇩(*r*1,*w*0,*r*0,*w*1); ⇩(*r*1,*w*0); ⇧(*r*0,*w*1,*r*1,*w*0,*r*0)} (5)

Результатом применения различных комбинаций примитивов могут быть тесты запоминающих устройств, обладающие требуемой полнотой, как правило, максимальной, в классе PNPSFk и имеющие различную временную сложность. Однако, тестом с максимальной полнотой покрытия *PNPSFk* и имеющим минимальную временную сложность, по мнению авторов, является тест *March OP* (5). Следует отметить, что, используя понятия примитивов, можно синтезировать тесты запоминающих устройств, позволяющих обнаруживать и другие разновидности их сложных неисправностей.

Заключение. В представленной статье показана взаимосвязь между орбитами, формируемыми маршевыми тестами, и их способностью обнаруживать сложные кодочувствительные неисправности *PNPSFk*. Показано, что наличие минимального подмножества из восьми типов орбит при реализации маршевого теста является необходимым условием достижения максимальной полноты покрытия тестом PNPSFk неисправностей. Определены понятия примитивов и по-

Таблица 7. Примитивы, формируемые тестом March OP
Table 7. Primitives generated by the March OP

Фазы	Предыдущая	⇧⇩ (w0)	⇧(r0,w1)	⇧(r1,w0)	⇩(r0,w1)	⇩ (r1,w0,r0, w1),	⇩(r1,w0)
	Текущая	⇧(r0,w1)	⇧(r1,w0)	⇩(r0,w1)	⇩ (r1,w0, r0,w1);	⇩(r1,w0);	⇧ (r0,w1, r1,w0,r0)
	Последующая	⇧(r1,w0)	⇩(r0,w1)	⇩ (r1,w0, r0,w1)	⇩(r1,w0);	⇧ (r0,w0, r1,w0,r0)	-
Обнаруживаемые PNPSFk		⟨0,0,0,…, 0,↑,1, 1,1,…,1⟩	⟨1,1,1,…, 1,↓,0, 0,0,…,0⟩	⟨1,1,1,…, 1 , ↑ , 0 , 0,0,…,0⟩	⟨1,1,1,…, 1 , ↓ , 1 , 1,1,…,1⟩ ; ⟨1,1,1,…, 1 , ↑ , 1 , 1,1,…,1⟩	⟨0,0,0,…, 0 , ↓ , 1 , 1,1,…,1⟩	⟨0,0,0,…, 0,↑,0, 0,0,…,0⟩ ; ⟨0,0,0,…, 0,↓,0, 0,0,…,0⟩ ;

казаны их примеры, а также результаты синтеза маршевых тестов обеспечивающих требуемую полноту покрытия заданных видов PNPSFk. Приведен пример маршевого теста с максимальной полнотой покрытия, имеющий минимальную временную сложность.

Список литературы

1. Bushnell, M. L. Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits / M. L. Bushnell, V. D. Agrawal. – New York: Kluwer Academic Publishers, 2000. – 690 p.
2. Wang, L. T. VLSI Test Principles and Architectures: Design for Testability / L. T. Wang, C. W. Wu, X. Wen. – Amsterdam: Elsevier, 2006. – 808 p.
3. Ярмолик, В. Н. Контроль и диагностика вычислительных систем / В. Н. Ярмолик. – Минск: Бестпринт, 2019. – 387 с.
4. The International Technology Roadmap for Semiconductors: 2003 Edition (ITRS'2003). – San Jose: Semiconductor Industry Association, 2003. – 65 p.
5. Sharma, A. K. Advanced Semiconductor Memories: Architectures, Designs, and Applications / A. K. Sharma. – London: John Wiley & Sons, 2003. – 652 p.
6. Goor, A. J. Testing Semiconductor Memories: Theory and Practice / A. J. Goor. – Chichester: John Wiley & Sons, 1991.
7. Ярмолик, В. Н. Неразрушающее тестирование запоминающих устройств / В. Н. Ярмолик [и др.]. – Минск: Бестпринт, 2005. – 230 с.
8. Ярмолик, С. В. Маршевые тесты для самотестирования ОЗУ / С. В. Ярмолик, А. П. Занкович, А. А. Иванюк. – Минск: Бестпринт, 2009. – 270 с.
9. Marinescu, M. Simple and Efficient Algorithms for Functional RAM Testing: IEEE Int. Test Conf. / M. Marinescu. – IEEE Computer Society Press, 1982. – P. 236–239.
10. Nair, C. Efficient Algorithms for Testing Semiconductor Random-Access Memories: IEEE Int. Test Conf. / C. Nair, S. Thatte, J. Abraham. – IEEE Transactions on Computers, 1978. – vol. 27, no. 6. – P. 572-576.
11. Suk, D. S. A March Test for Functional Faults in Semiconductor Random-Access Memories: IEEE Trans. on Computers / D. S. Suk, S. M. Reddy. – IEEE Transactions on Computers, 1981. – vol. 30, no. 12. – P. 982–985.
12. Goor, A. J. March LR: A test for Realistic Linked Faults: 14th VLSI Test Symposium / A. J. Goor, G. N. Gaydadjiev, V. N. Yarmolik, V. G. Mikitijuk. – IEEE Computer Society Press, 1996. – P. 272–280.
13. Goor, A. J. March LA: A test for Linked Memory Faults: Proc. of the 1997 European Design and Test Conference (ED&TC'97) / A. J. Goor, G. N. Gaydadjiev, V. N. Yarmolik, V. G. Mikitjuk. – IEEE Computer Society Press, 1997. – P. 627.
14. Cheng, K. L. Efficient neighborhood pattern-sensitive fault test algorithms for semiconductor memories: 19th VLSI Test Symposium / K. L. Cheng, M. F. Tsai, C. W. Wu. – IEEE Computer Society Press, 2001. – P. 225–237.
15. Cascaval, P. Efficient March test for 3-coupling faults in random access memories / P. Cascaval, S. Bennett. // Microprocessors and Microsystems. – Elsevier, 2001. – vol. 24, no. 10. – P. 501–509.
16. Kang, D. C. An efficient built-in self-test algorithm for neighborhood pattern sensitive faults in high-density memories: Proc. 4th Korea-Russia Int. Symposium on Science and Technology / D. C. Kang, S. B. Cho. – IEEE Service Center, 2000. – vol. 2 – P. 218–223.
17. Cheng, K. L. Neighborhood pattern-sensitive fault testing and diagnostics for random-access memories / K. L. Cheng, M. F. Tsai, C.W. Wu. // IEEE Transactions on Computer – IEEE Press, 2002. – vol. 21. no. 11. – P. 1328–1336.
18. Cockburn, B. F. Deterministic tests for detecting single V-coupling faults in RAMs / B. F. Cockburn. // Journal of Electronic Testing: Theory Applicat. – Springer, 1994. – vol. 5, no. 1. – P. 91–113.
19. Ярмолик, В. Н. Псевдоисчерпывающее тестирование запоминающих устройств на базе маршевых тестов типа March A / В. Н. Ярмолик, И. Мрозек, С. В. Ярмолик. // Информатика. – Минск, 2020. – № 2 (17). – С. 54–70.
20. Yarmolik, V. N. March ps(23n) test for DRAM pattern-sensitive faults: Proc. of the 7th AsianTest Symposium / V. N. Yarmolik, Y. Klimets, S. Demidenko. – IEEE Computer Society, 1998. – P. 354–357.

References

1. M. L. Bushnell, V. D. Agrawal. Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits. New York: Kluwer Academic Publishers, 2000. 690 p.
2. L. T. Wang, C. W. Wu, X. Wen. VLSI Test Principles and Architectures: Design for Testability. Amsterdam: Elsevier, 2006. 808 p.
3. V. N. Yarmolik. Kontrol i diagnostika vychislitelnykh system [Control and diagnostics of computing systems]. Minsk: Bestprint, 2019. 387 p. (in Russian).
4. The International Technology Roadmap for Semiconductors: 2003 Edition (ITRS'2003). San Jose: Semiconductor Industry Association, 2003. 65 p.
5. A. K. Sharma. Advanced Semiconductor Memories: Architectures, Designs, and Applications. London: John Wiley & Sons, 2003. 652 p.
6. A. J. Goor. Testing Semiconductor Memories: Theory and Practice. Chichester: John Wiley & Sons, 1991.
7. V. N. Yarmolik. Nerazrushayushcheye testirovaniye zapominayushchikh ustroystv [Nondestructive testing of storage devices]. Minsk: Bestprint, 2005. 230 p. (in Russian).
8. S. V. Yarmolik, A. P. Zankovich, A. A. Ivanyuk. Marshevyye testy dlya samotestirovaniya OZU [Marching tests for self-testing of RAM]. Minsk: Bestprint, 2009. 270 p. (in Russian).
9. M. Marinescu. Simple and Efficient Algorithms for Functional RAM Testing: IEEE Int. Test Conf. IEEE Computer Society Press, 1982. pp. 236–239.
10. C. Nair, S. Thatte, J. Abraham. Efficient Algorithms for Testing Semiconductor Random-Access Memories: IEEE Int. Test Conf. IEEE Transactions on Computers, 1978. vol. 27, no. 6, pp. 572-576.
11. D. S. Suk, S. M. Reddy. A March Test for Functional Faults in Semiconductor Random-Access Memories: IEEE Trans. on Computers. IEEE Transactions on Computers, 1981. vol. 30, no. 12, pp. 982–985.
12. A. J. Goor, G. N. Gaydadjiev, V. N. Yarmolik, V. G. Mikitijuk. March LR: A test for Realistic Linked Faults: 14th VLSI Test Symposium. IEEE Computer Society Press, 1996. pp. 272–280.
13. A. J. Goor, G. N. Gaydadjiev, V. N. Yarmolik, V. G. Mikitjuk. March LA: A test for Linked Memory Faults: Proc. of the 1997 European Design and Test Conference (ED&TC'97). IEEE Computer Society Press, 1997. 627 p.
14. K. L. Cheng, M. F. Tsai, C. W. Wu. Efficient neighborhood pattern-sensitive fault test algorithms for semiconductor memories: 19th VLSI Test Symposium. IEEE Computer Society Press, 2001. pp. 225–237.
15. P. Cascaval, S. Bennett. Efficient March test for 3-coupling faults in random access memories. Microprocessors and Microsystems, Elsevier, 2001. vol. 24, no. 10, pp. 501–509.
16. D. C. Kang, S. B. Cho. An efficient built-in self-test algorithm for neighborhood pattern sensitive faults in high-density memories: Proc. 4th Korea-Russia Int. Symposium on Science and Technology. IEEE Service Center, 2000. vol. 2, pp. 218–223.
17. K. L. Cheng, M. F. Tsai, C.W. Wu. Neighborhood pattern-sensitive fault testing and diagnostics for random-access memories. IEEE Transactions on Computer, IEEE Press, 2002. vol. 21, no. 11, pp. 1328–1336.
18. B. F. Cockburn. Deterministic tests for detecting single V-coupling faults in RAMs. Journal of Electronic Testing: Theory Applicat, Springer, 1994. vol. 5, no. 1. pp. 91–113.
19. V. N. Yarmolik, I. Mrozek, S. V. Yarmolik. Psevdoischerpyvayushcheye testirovaniye zapominayushchikh ustroystv na baze marshevykh testov tipa March A [Pseudo-exhaustive memory testing based on March A]. Informatika. Minsk, 2020. № 2 (17). pp. 54–70. (in Russian).
20. V. N. Yarmolik, Y. Klimets, S. Demidenko. March ps(23n) test for DRAM pattern-sensitive faults: Proc. of the 7th AsianTest Symposium. IEEE Computer Society, 1998. pp. 354–357.

Received: 23.11.2020 *Поступила: 23.11.2020*