УДК 004.33.054
https://doi.org/10.37661/1816-0301-2021-18-1-25-42

Оригинальная статья
Original Paper

Построение и применение маршевых тестов для обнаружения кодочувствительных неисправностей запоминающих устройств

В. Н. Ярмолик[1]✉, В. А. Леванцевич[1], Д. В. Деменковец[1], И. Мрозек[2]

[1]*Белорусский государственный университет информатики и радиоэлектроники, ул. П. Бровки, 6, 220013, Минск, Беларусь*
✉*E-mail: yarmolik10ru@yahoo.com*

[2]*Белостоцкий технический университет, ул. Вейска, 45А, 15-351, Белосток, Польша*

Аннотация. Показывается актуальность задачи тестирования запоминающих устройств современных вычислительных систем. Исследуются математические модели неисправностей запоминающих устройств и используемые методы тестирования наиболее сложных из них на базе классических маршевых тестов. Выделяются пассивные кодочувствительные неисправности (*PNPSFk*), в которых участвуют произвольные k из N ячеек памяти, где $k << N$, а N представляет собой емкость памяти в битах. Для этих неисправностей приводятся аналитические выражения минимальной и максимальной полноты покрытия, которые достижимы в рамках маршевых тестов. Определяется понятие примитива, описывающего в терминах элементов маршевого теста условия активизации и обнаружения неисправностей *PNPSFk* запоминающих устройств. Приводятся примеры построения маршевых тестов, имеющих максимальную полноту покрытия, а также маршевых тестов с минимальной временной сложностью, равной $18N$. Исследуется эффективность однократного применения тестов типа *MATS++*, *March C–* и *March PS* для различного количества $k \leq 9$ ячеек памяти, участвующих в неисправности *PNPSFk*. Обосновывается применимость многократного тестирования с изменяемыми адресными последовательностями, в качестве которых предлагается применять случайные последовательности адресов. Приводятся аналитические выражения для полноты покрытия сложных неисправностей *PNPSFk* в зависимости от кратности теста. Кроме того, даются оценки среднего значения кратности тестов *MATS++*, *March C–* и *March PS*, полученные на основании математической модели, которая описывает задачу собирателя купонов, и обеспечивающие обнаружение всех $k2^k$ неисправностей *PNPSFk*. Экспериментально показывается справедливость аналитических оценок и подтверждается высокая эффективность обнаружения неисправностей *PNPSFk* тестами типа *March PS*.

Ключевые слова: тестирование вычислительных систем, запоминающие устройства, маршевые тесты памяти, многократное тестирование, задача собирателя купонов

Для цитирования. Построение и применение маршевых тестов для обнаружения кодочувствительных неисправностей запоминающих устройств / В. Н. Ярмолик [и др.] // Информатика. – 2021. − Т. 18, № 1. – С. 25–42. https://doi.org/10.37661/1816-0301-2021-18-1-25-42

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 04.12.2020
Подписана в печать | Accepted 12.01.2021
Опубликована | Published 26.03.2021

Construction and application of march tests for pattern sensitive memory faults detection

Vyacheslav N. Yarmolik[1]✉, Vladimer A. Levantsevich[1], Denis V. Demenkovets[1], Ireneusz Mrozek[2]

Belarusian State University of Informatics and Radioelectronics,
st. P. Brovki, 6, 220013, Minsk, Belarus
✉*E-mail: yarmolik10ru@yahoo.com*
[2]*Bialystok University of Technology,*
st. Wiejska, 45A, 15-351, Białystok, Poland

Abstract. The urgency of the problem of testing storage devices of modern computer systems is shown. The mathematical models of their faults and the methods used for testing the most complex cases by classical march tests are investigated. Passive pattern sensitive faults (*PNPSFk*) are allocated, in which arbitrary *k* from *N* memory cells participate, where $k << N$, and *N* is the memory capacity in bits. For these faults, analytical expressions are given for the minimum and maximum fault coverage that is achievable within the march tests. The concept of a *primitive* is defined, which describes in terms of march test elements the conditions for activation and fault detection of *PNPSFk* of storage devices. Examples of march tests with maximum fault coverage, as well as march tests with a minimum time complexity equal to 18*N* are given. The efficiency of a single application of tests such as *MATS ++*, *March C–* and *March PS* is investigated for different number of $k \le 9$ memory cells involved in *PNPSFk* fault. The applicability of multiple testing with variable address sequences is substantiated, when the use of random sequences of addresses is proposed. Analytical expressions are given for the fault coverage of complex *PNPSFk* faults depending on the multiplicity of the test. In addition, the estimates of the mean value of the multiplicity of the *MATS++*, *March C–* and *March PS* tests, obtained on the basis of a mathematical model describing the problem of the coupon collector, and ensuring the detection of all $k2^k$ *PNPSFk* faults are given. The validity of analytical estimates is experimentally shown and the high efficiency of *PNPSFk* fault detection is confirmed by tests of the *March PS* type.

Key words: testing of computer systems, storage devices, march tests of memory, multiple testing, coupon collector problem

For citation. Yarmolik V. N., Levantsevich V. A., Demenkovets D. V., Mrozek I. Construction and application of march tests for pattern sensitive memory faults detection. *Informatics*, 2021, vol. 18, no. 1, pp. 25–42 (in Russian). https://doi.org/10.37661/1816-0301-2021-18-1-25-42

Conflict of interest. The authors declare of no conflict of interest.

Введение. Быстрый прогресс в области полупроводниковых технологий привел к созданию разнообразных типов запоминающих устройств большой емкости, широко используемых в различных приложениях [1, 2]. Так, запоминающие устройства современных вычислительных систем занимают до 94 % площади кристалла системы, что приводит к возрастанию требований к их надежности [3–5]. Косвенным результатом такого роста емкости запоминающих устройств является не только увеличение вероятности отказов памяти, но и увеличение сложности неисправностей запоминающих устройств, которая обусловлена высокой плотностью размещения запоминающих ячеек, а также разнообразными механизмами неисправностей [1, 6]. Общепринятой моделью неисправностей запоминающих устройств, покрывающей более простые виды неисправностей памяти, являются так называемые кодочувствительные неисправности (*pattern sensitive faults, PSF*) [6, 7].

Неисправности *PSF* вызваны аномалиями запоминающего устройства и паразитными эффектами, которые проявляются в зависимости как от хранимых в памяти данных, так и от их изменений и последовательностей обращений (операций чтения и записи) к памяти [6–8]. В запоминающем устройстве емкостью *N* бит может храниться 2^N различных наборов двоичных данных, надежность записи и считывания которых обеспечивает исправное поведение памяти. Показано, что для определения неисправности *PSF* запоминающего устройства минимальная сложность тестовой последовательности определяется выражением $2^N(3N^2 + 2N)$ [7, 9]. Это приводит к тому, что для больших объемов современных запоминающих устройств временная сложность

реализации алгоритмов тестирования памяти чрезвычайно велика [6–10]. Поэтому задача тестирования запоминающих устройств с целью обнаружения кодочувствительных неисправностей на протяжении последних десятилетий и до настоящего времени являлась и является весьма актуальной [11–15].

Кодочувствительные неисправности *PSF* описывают поведение нескольких ячеек памяти вплоть до *N* ячеек, где *N* – емкость памяти в битах [6]. Для подобных неисправностей логическое состояние одной ячейки памяти, называемой базовой (*base cell*), может зависеть от содержимого (0 или 1) или от логических переходов из 1 в 0 или из 0 в 1 в соседних ячейках (*neighborhood cells*) запоминающего устройства. Под логическим переходом из 1 в 0, обозначаемым символом ↓, понимают изменение единичного состояния ячейки памяти на нулевое состояние, а под переходом из 0 в 1 – изменение нулевого состояния на единичное, которое обозначается символом ↑ [6, 7]. Различают два вида кодочувствительных неисправностей: неограниченные (*unrestricted*) и ограниченные (*restricted*), или граничные (*neighborhood, NPSF*). Под *NPSF* понимают такие разновидности неисправностей *PSF*, для которых вводится ограничение как на количество ячеек памяти, участвующих в неисправности, так и на их физическое расположение в матрице запоминающих ячеек. При тестировании современных запоминающих устройств обычно придерживаются последней, более реальной модели кодочувствительных неисправностей, для которой рассматривается небольшое число $k \le 9$ ячеек памяти, входящих в неисправность *NPSFk*, а их местоположение может быть произвольным [5, 6, 16]. Существуют три классические модели неисправностей *NPSFk*, а именно активные (*ANPSFk*), пассивные (*PNPSFk*) и статические (*SNPSFk*) [6]. Активными, или динамическими, являются такие *NPSFk*, для которых базовая ячейка изменяет свое содержимое из-за изменения в наборе, хранящемся в соседних k–1 ячейках. Пассивными называются такие *NPSFk*, для которых содержимое базовой ячейки не может быть изменено из-за определенного набора данных в соседних k–1 ячейках. Статические *SNPSFk* – это неисправности, при которых содержимое базовой ячейки принудительно переводится в одно из двух состояний 0 или 1 из-за определенного набора в соседних k–1 ячейках. Отметим, что все перечисленные неисправности возникают при изменении содержимого памяти, т. е. при выполнении операции записи в ячейку памяти [5, 6].

В качестве объекта исследования чаще всего рассматриваются пассивные кодочувствительные неисправности (*PNPSFk*), где k обозначает количество произвольных ячеек памяти 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 моделью наиболее трудно обнаруживаемых неисправностей памяти, покрывающей другие виды неисправностей [5, 6].

В настоящей статье основное внимание уделено анализу эффективности существующих тестов для обнаружения кодочувствительных неисправностей *PNPSFk*, синтезу однократных маршевых тестов с максимальной полнотой покрытия таких неисправностей и оценке их эффективности при многократном применении.

Анализ существующих разновидностей моделей неисправностей *PNPSFk*. Существуют различные виды кодочувствительных неисправностей *NPSFk* в зависимости от тех ограничений, которые накладываются на обобщающую модель *PSFk*. Первым необходимым ограничением является количество запоминающих элементов, участвующих в неисправности *NPSFk* [5, 6]. Как правило, k не превышает 9. Это следует из того факта, что для тестирования подобных неисправностей необходимо время, пропорциональное величине 2^k. Вторым ограничением является физическое соседство ячеек памяти. На практике обычно используют модели кодочувствительных неисправностей *NPSFk* с числом k, равным 3, 5 или 9, конфигурации и обозначения которых показаны на рис. 1 [5, 6, 11, 17].

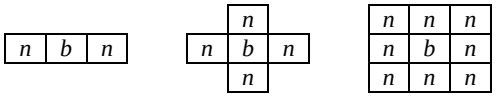

Рис. 1. Модели кодочувствительных неисправностей

Как следует из рис. 1, в каждой из приведенных неисправностей в явном виде выделяется базовый запоминающий элемент *b* и соседние запоминающие элементы *n*. Соседние ячейки запоминающего устройства являются физическими соседями по отношению к базовому элементу. Исследования, посвященные построению тестов для обнаружения *NPSF*3 при известной физической топологии, достаточно полно представлены в работах [5, 6, 13, 15, 17]. Наиболее распространенными моделями *NPSF* являются *NPSF*5 и *NPSF*9, которые получили название неисправностей типа 1 (*type* 1) и типа 2 (*type* 2) [6, 7, 10]. Существуют и другие конфигурации *NPSFk* как по взаимному расположению запоминающих элементов, так и по их количеству. Одной из таких распространенных разновидностей неисправностей является *NPSFk*, для которой все *k* запоминающих элементов принадлежат одному столбцу или одной строке матрицы ячеек памяти [6, 18, 19], а максимальное значение *k* достигает значения 25 [5, 6, 15, 17]. Среди указанного множества неисправностей выделяют *neighborhood word-line sensitive faults* (*NWSF*), т. е. неисправности, связанные с шиной данных [20, 21]. В работе [22] был предложен вариант соседства с четырьмя ячейками (*T-type*), которое обеспечивало обнаружение неисправностей, чувствительных к двоичному набору окружения битовой шины памяти (*bit-line neighborhood pattern sensitive faults, NBLSFs*). Достаточно полно модели кодочувствительных неисправностей (*row* (*column*) *pattern sensitive faults*), связанные с топологией строк и столбцов матрицы запоминающих элементов, исследованы в работе [18]. Весьма интересной моделью кодочувствительных неисправностей являются неисправности, чувствительные к операции чтения (*disturb neighborhood pattern sensitive faults*), впервые представленные и проанализированные в работе [23].

Описание неисправностей *NPSFk*, когда уточняются топологические особенности памяти, является идеальным с точки зрения решения проблемы тестирования запоминающих устройств. Однако на практике физическое месторасположение запоминающих элементов памяти и их логические адреса в основном не совпадают и на уровне пользователя являются недоступной информацией [5, 6, 15, 17, 24, 25]. Как правило, логически соседние запоминающие элементы физически располагаются удаленно друг от друга и, наоборот, физически соседние ячейки имеют адреса, удаленные друг от друга в используемом адресном пространстве памяти [17, 25]. Причиной данного факта является ряд технологических приемов, используемых при производстве современных запоминающих устройств [6, 17, 18].

Выделяют *k* различных классов *PNPSFk* в зависимости от местоположения в адресном пространстве памяти базовой ячейки *b* по отношению к соседним ячейкам *n*. Например, для $k = 5$ существует пять классов неисправностей *PNPSF*5: $b_{i_4} n_{i_3} n_{i_2} n_{i_1} n_{i_0}$, $n_{i_4} b_{i_3} n_{i_2} n_{i_1} n_{i_0}$, $n_{i_4} n_{i_3} b_{i_2} n_{i_1} n_{i_0}$, $n_{i_4} n_{i_3} n_{i_2} b_{i_1} n_{i_0}$ и $n_{i_4} n_{i_3} n_{i_2} n_{i_1} b_{i_0}$, где адреса ячеек в адресном пространстве имеют следующее соотношение: $i_0 < i_1 < i_2 < i_3 < i_4$. Каждый класс неисправностей включает как минимум две неисправности *PNPSFk* в зависимости от состояния базового элемента, которое не может быть изменено на противоположное состояние. Например, классу $n_{i_4} b_{i_3} n_{i_2} n_{i_1} n_{i_0}$ неисправностей *PNPSF*5 для конкретного набора (0, 0, 0, 0) в соседних ячейках $n_{i_4} n_{i_2} n_{i_1} n_{i_0}$ принадлежат две неисправности, имеющие вид $\langle 0,\uparrow,0,0,0\rangle$, $\langle 0,\downarrow,0,0,0\rangle$. В обеих неисправностях состояния соседних ячеек с адресами i_0, i_1, i_2 и i_4 принимаются нулевыми. При этих состояниях в случае неисправности $\langle 0,\uparrow,0,0,0\rangle$ базовая ячейка не может выполнить переход из нулевого состояния в единичное, а для второй неисправности $\langle 0,\downarrow,0,0,0\rangle$ – из единичного в нулевое. В общем случае в соседних ячейках возможны любые из 2^{k-1} двоичных наборов, каждый из которых определяет два неисправных поведения базовой ячейки. Соответственно, общее количество *PNPSFk*, относящихся к одному конкретному классу из *k* возможных, равняется $2 \cdot 2^{k-1} = 2^k$. В то же время для фиксированных *k* ячеек памяти, для которых существует *k* классов неисправностей, имеем общее количество $Q_{TK}(PNPSFk)$ неисправностей *PNPSFk*, равное $k2^k$. Число всевозможных неисправностей *PNPSFk* для памяти емкостью *N* бит определяется выражением [17]

$$Q_{TN}(PNPSFk) = k \cdot 2^k \cdot \binom{N}{k}. \quad (1)$$

Последний сомножитель в равенстве (1) представляет собой количество сочетаний по k ячеек памяти из общего их количества, определяемого емкостью N памяти, причем каждое такое сочетание соответствует определенной конфигурации неисправностей *PNPSFk*, определяемой ячейками памяти, участвующими в ней. Емкость современных запоминающих устройств N велика и является существенно большей по сравнению с количеством k ячеек, участвующих в рассматриваемых кодочувствительных неисправностях. Поэтому для реальных значений N приведенная оценка (1) принимает большие значения, показывающие сложность проблемы синтеза тестов для обнаружения кодочувствительных неисправностей запоминающих устройств.

В связи с большим объемом запоминающих устройств их тестирование оказывается возможным только с применением маршевых тестов (*march tests*), временная сложность реализации которых линейно зависит от емкости N памяти [6]. Маршевый тест состоит из конечной последовательности маршевых элементов, называемых фазами, каждая из которых представляет конечную последовательность операций записи и чтения, применяемых к каждой ячейке памяти перед переходом к следующей ячейке. Операция может состоять из записи 0 в ячейку ($w0$), записи 1 в ячейку ($w1$), чтения ожидаемого 0 из ячейки ($r0$) и чтения ожидаемой 1 из ячейки ($r1$). Каждый маршевый элемент содержит символ, определяющий порядок формирования адресной последовательности (*address sequence*), где символ $\Uparrow$ определяет последовательный перебор адресов памяти по возрастанию, символ $\Downarrow$ – последовательный перебор адресов по убыванию, а сочетание двух символов $\Uparrow\Downarrow$ означает формирование адресов по убыванию либо по возрастанию. Убывающая последовательность адресов ($\Downarrow$) представляет собой последовательность, которая формируется в обратном порядке по сравнению с возрастающей последовательностью ($\Uparrow$), при этом в качестве возрастающей последовательности адресов может быть использована любая последовательность, состоящая из всех N адресов. Тест *MATS*: {$\Uparrow\Downarrow(w0)$; $\Uparrow(r0,w1)$; $\Downarrow(r1)$} служит простейшим примером семейства маршевых тестов и имеет временную сложность $4N$. Часто используемыми маршевыми тестами являются тесты *MATS++*: {$\Uparrow\Downarrow(w0)$; $\Uparrow(r0,w1)$; $\Downarrow(r1,w0,r0)$} и *March C−*: {$\Uparrow\Downarrow$ $(w0)$; $\Uparrow(r0,w1)$; $\Uparrow(r1,w0)$; $\Downarrow(r0,w1)$; $\Downarrow(r1,w0)$; $\Uparrow\Downarrow(r0)$}, имеющие временную сложность $6N$ и $10N$ соответственно.

Не нарушая общности последующих рассуждений, предположим, что всегда для любого маршевого теста используется фаза инициализации вида $\Uparrow\Downarrow(w0)$. Тогда, учитывая последовательное обращение к запоминающим ячейкам и одинаковые операции записи используемого маршевого теста, применяемые к ним, можно заключить, что содержимое памяти после выполнения каждой из фаз будет либо исходным нулевым, либо инверсным, когда в каждой ячейке будет записана единица. Аналогичное заключение можно сделать и для k ячеек памяти с упорядоченными адресами $i_0 < i_1 < i_2 < \ldots < i_{k-1}$, участвующих в неисправности *PNPSFk* $n_{i_{k-1}} n_{i_{k-2}} n_{i_{k-3}} \ldots n_{i_{j+1}} b_{i_j} n_{i_{j-1}} \ldots n_{i_2} n_{i_1} n_{i_0}$. Такая особенность маршевых тестов позволяет определить множество обнаруживаемых неисправностей, которое соответствует следующему утверждению.

Утверждение. *К множеству обнаруживаемых маршевыми тестами неисправностей PNPSFk относятся такие их разновидности, для которых в соседних ячейках* n, *расположенных до базовой ячейки* b, *находятся только нулевые значения или только единичные значения, а в соседних ячейках после базовой также находятся одинаковые значения (нулевые или единичные), причем независимо от значений в соседних ячейках до базовой ячейки.*

Согласно приведенному утверждению для случая кодочувствительных неисправностей *PNPSFk* $n_{i_{k-1}} n_{i_{k-2}} n_{i_{k-3}} \ldots n_{i_{j+1}} b_{i_j} n_{i_{j-1}} \ldots n_{i_2} n_{i_1} n_{i_0}$ произвольного j-го класса, $j \in \{1, 2, \ldots, k-2\}$, множество обнаруживаемых неисправностей будет состоять из восьми видов неисправностей каждого из этих классов. Для нулевого (i_0) и $k-1$-го (i_{k-1}) классов в силу отсутствия соседних ячеек слева или справа обнаруживаемыми будут только четыре вида неисправностей *PNPSFk*.

В качестве примера обнаруживаемых неисправностей на базе маршевых тестов в табл. 1 приведены все виды неисправностей для каждого из $k = 5$ классов.

Таблица 1

Обнаруживаемые неисправности *PNPSF*5

Классы *PNPSF*5	Виды неисправностей *PNPSF*5
$n_{i_4} n_{i_3} n_{i_2} n_{i_1} b_{i_0}$	⟨0,0,0,0,↑⟩, ⟨0,0,0,0,↓⟩, ⟨1,1,1,1,↑⟩, ⟨1,1,1,1,↓⟩
$n_{i_4} n_{i_3} n_{i_2} b_{i_1} n_{i_0}$	⟨0,0,0,↑,0⟩, ⟨0,0,0.↓,0⟩, ⟨1,1,1,↑,1⟩, ⟨1,1,1,↓,1⟩, ⟨0,0,0,↑,1⟩, ⟨0,0,0,↓,1⟩, ⟨1,1,1,↑,0⟩, ⟨1,1,1,↓,0⟩
$n_{i_4} n_{i_3} b_{i_2} n_{i_1} n_{i_0}$	⟨0,0,↑,0,0⟩, ⟨0,0.↓,0,0⟩, ⟨1,1,↑,1,1⟩, ⟨1,1,↓,1,1⟩, ⟨0,0,↑,1,1⟩, ⟨0,0,↓,1,1⟩, ⟨1,1,↑,0,0⟩, ⟨1,1,↓,0,0⟩
$n_{i_4} b_{i_3} n_{i_2} n_{i_1} n_{i_0}$	⟨0,↑,0,0,0⟩, ⟨0,↓,0,0,0⟩, ⟨1,↑,1,1,1⟩, ⟨1,↓,1,1,1⟩, ⟨0,↑,1,1,1⟩, ⟨0,↓,1,1,1⟩, ⟨1,↑,0,0,0⟩, ⟨1,↓,0,0,0⟩
$b_{i_4} n_{i_3} n_{i_2} n_{i_1} n_{i_0}$	⟨↑,0,0,0,0⟩, ⟨↓,0,0,0,0⟩, ⟨↑,1,1,1,1⟩, ⟨↓,1,1,1,1⟩

Соответственно, согласно утверждению 1 максимально возможное число $Q_{MAX}(PNPSFk)$ обнаруживаемых неисправностей *PNPSFk* при применении однократного маршевого теста определяется равенством

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

Соотношение (2) показывает предельные возможности маршевых тестов, которые достижимы при однократном их применении.

Анализ известных разновидностей тестов для обнаружения неисправностей *PNPSFk*. Как отмечалось в предыдущих разделах, объектом исследований в части сложных неисправностей памяти, как правило, являются неисправности *PNPSFk* [6, 17].

Для нахождения определенной неисправности памяти, в том числе и неисправности *PNPSFk*, в используемом для этих целей маршевом тесте необходимо выполнить как условия ее активизации (*sensitization*), так и условия обнаружения (*detection*) [3, 5, 6]. Процесс активизации неисправности требует предварительной инициализации (*initialization*) содержимого в k ячейках памяти, для которого заданная неисправность *PNPSFk* будет активизирована. Инициализация необходима как для соседних ячеек, участвующих в неисправности в соответствии с приведенным утверждением, так и для базовой ячейки и определяется одним из восьми видов неисправностей *PNPSFk*. Процедура инициализации заключается в предварительном обнулении содержимого памяти либо записи во все ячейки памяти единичных значений, а также выборе в последующей фазе теста соответствующего направления адресации, возрастающего либо убывающего. Выполнение этих двух условий позволит сформировать (инициализировать) содержимое в k произвольных ячейках памяти, соответствующее одному из восьми видов неисправностей *PNPSFk*. Отметим временную зависимость указанных процедур, которая заключается в последовательной реализации в маршевом тесте инициализации, затем активизации и, наконец, обнаружения неисправности.

В случае классических маршевых тестов условия инициализации, активизации и обнаружения неисправности *PNPSFk* очевиднее всего реализуются тремя последовательными фазами маршевого теста, которые в дальнейшем будем называть предыдущей, текущей и последующей фазами, а их совокупность – примитивом. Все восемь видов неисправностей *PNPSFk* и необходимые для их обнаружения примитивы, реализуемые маршевыми тестами, приведены в табл. 2.

Отметим, что все обнаруживаемые неисправности соответствуют приведенному утверждению и поэтому могут быть выявлены маршевым тестом. Например, маршевый тест *MATS* реализует примитив, состоящий из предыдущей фазы ⇑⇓($w0$), текущей фазы ⇑($r0$,$w1$) и последующей фазы ⇓($r1$), что обеспечивает условия активизации и обнаружения неисправности вида ⟨0,0,0,…,0,↑,1,1, 1,…,1⟩ для всех k классов неисправностей *PNPSFk*. Для случая $k = 4$ фиксированных ячеек памяти тест *MATS* обнаружит четыре неисправности ⟨0,0,0,↑⟩, ⟨0,0,↑,1⟩, ⟨0,↑,1,1⟩ и ⟨↑,1,1,1⟩, а остальные $k\cdot 2^k - k = 4\cdot 2^4 - 4 = 60$ неисправностей *PNPSF*4 в этих ячейках данный тест не обнаружит.

Из табл. 2 видно, что примитив может состоять также из двух последовательных фаз маршевого теста, как и для случая ранее рассмотренной неисправности вида ⟨0,0,0,…,0,↑, 1,1,1,…,1⟩, однако это требует усложнения текущей фазы, в которой необходима реализация процедуры обнаружения неисправности. В таком случае примитив будет состоять только из двух фаз: предыдущей и следующей за ней текущей фазы.

Таблица 2

Виды неисправности *PNPSFk* и необходимые примитивы для их обнаружения

Обнаруживаемые виды неисправностей *PNPSFk*	Примитив		
	Предыдущая фаза	Текущая фаза	Последующая фаза
⟨0,0,0,…,0,↑,1,1,1,…,1⟩	⇑⇓(…,*w*0,…)	⇑(*r*0,*w*1)	⇑⇓(*r*1,…)
	⇑⇓(…,*w*0,…)	⇑(*r*0,*w*1,*r*1)	–
⟨0,0,0,…,0,↓,1,1,1,…,1⟩	⇑⇓(…,*w*1,…)	⇓(*r*1,*w*0)	⇑⇓(*r*0,…)
	⇑⇓(…,*w*1,…)	⇓(*r*1,*w*0,*r*0)	–
⟨1,1,1,…,1,↑,0,0,0,…,0⟩	⇑⇓(…,*w*0,…)	⇓(*r*0,*w*1)	⇑⇓(*r*1,…)
	⇑⇓(…,*w*0,…)	⇓(*r*0,*w*1,*r*1)	–
⟨1,1,1,…,1,↓,0,0,0,…,0⟩	⇑⇓(…,*w*1,…)	⇑(*r*1,*w*0)	⇑⇓(*r*0,…)
	⇑⇓(…,*w*1,…)	⇑(*r*1,*w*0,*r*0)	–
⟨0,0,0,…,0,↑,0,0,0,…,0⟩	⇑⇓(…,*w*0,…)	⇑⇓(*r*0,*w*1,*r*1,*w*0)	–
⟨0,0,0,…,0,↓,0,0,0,…,0⟩	⇑⇓(…,*w*0,…)	⇑⇓(*r*0,*w*1,*r*1,*w*0)	⇑⇓(*r*0,…)
	⇑⇓(…,*w*0,…)	⇑⇓(*r*0,*w*1,*r*1,*w*0,*r*0)	–
⟨1,1,1,…,1,↓,1,1,1,…,1⟩	⇑⇓(…,*w*1,…)	⇑⇓(*r*1,*w*0,*r*0,*w*1)	–
⟨1,1,1,…,1,↑,1,1,1,…,1⟩	⇑⇓(…,*w*1,…)	⇑⇓(*r*1,*w*0,*r*0,*w*1)	⇑⇓(*r*1,…)
	⇑⇓(…,*w*1,…)	⇑⇓(*r*1,*w*0,*r*0,*w*1,*r*1)	–

В общем случае тест *MATS* характеризуется минимальной полнотой покрытия неисправностей, так как он позволяет обнаруживать только один вид неисправности *PNPSFk*, а именно ⟨0,0,0,…, 0,↑,1,1,1,…,1⟩. Общее количество $Q_{Dk}(PNPSFk)$ обнаруживаемых тестом *MATS* неисправностей *PNPSFk* в *k* фиксированных ячейках равняется *k*, а во всей памяти емкостью *N* бит определяется выражением

$$Q_{DN}(PNPSFk) = Q_{Dk}(PNPSFk)\cdot\binom{N}{k} = k\cdot\binom{N}{k}. \qquad (3)$$

Таким образом, полнота покрытия однократного маршевого теста *MATS* как процентное отношение количества $Q_{DN}(PNPSFk)$ обнаруживаемых неисправностей *PNPSFk* к их общему числу $Q_{TN}(PNPSFk)$ (см. (1)) принимает вид

$$FC_{MATS}(PNPSFk) = \frac{Q_{DN}(PNPSFk)}{Q_{TN}(PNPSFk)}\cdot 100\,\% = \frac{Q_{Dk}(PNPSFk)\cdot\binom{N}{k}}{Q_{Tk}(PNPSFk)\cdot\binom{N}{k}}\cdot 100\,\% = \frac{1}{2^k}\cdot 100\,\%. \qquad (4)$$

Анализ соотношения (4) позволяет сделать следующие выводы. Во-первых, полнота покрытия неисправностей *PNPSFk* произвольного маршевого теста для всей памяти емкостью *N* бит совпадает с полнотой покрытия для этих же неисправностей в *k* произвольных фиксированных ячейках памяти. Во-вторых, полнота покрытия $FC_{MATS}(PNPSFk)$ является минимально возможной полнотой покрытия, достигаемой маршевыми тестами. Второй вывод следует из того факта, что произвольный маршевый тест обязательно содержит фазу инициализации, а в последующих фазах – операцию записи инверсного значения и проверку правильности ее выполнения. В совокупности это составляет по меньшей мере один примитив, обеспечивающий обнаруже-

ние как минимум одного из видов неисправностей *PNPSFk*, приведенных в табл. 2. Более сложные маршевые тесты по сравнению с тестом *MATS* в части числа их фаз и количества операций чтения (записи) в фазах, как правило, имеют бóльшую полноту покрытия $FC_{Test}(PNPSFk)$. Например, тест *MATS*++, имеющий сложность теста 6*N*, которая больше сложности 4*N* теста *MATS*, характеризуется полнотой покрытия $FC_{MATS++}(PNPSFk) = 1/2^{k-1} \cdot 100\,\%$. Это следует из того факта, что данный тест обнаруживает *k* неисправностей *PNPSFk* вида ⟨0,0,0,…,0,↑,1,1,1,…, 1⟩ и *k* неисправностей вида ⟨0,0,0,…,0,↓,1,1,1,…, 1⟩, так как содержит соответствующие примитивы. Примитив ⇑⇓(*w*0), ⇑(*r*0,*w*1), ⇓(*r*1,*w*0,*r*0) теста *MATS*++ обеспечивает обнаружение ⟨0,0,0,…,0,↑,1,1,1,…,1⟩, а примитив ⇑(*r*0,*w*1), ⇓(*r*1,*w*0,*r*0) реализует условия покрытия неисправностей ⟨0,0,0,…,0,↓, 1,1,1,…,1⟩. Более сложный тест *March C*− обеспечивает обнаружение уже четырех видов *PNPSFk*, приведенных в табл. 3 с соответствующими примитивами теста.

Таблица 3

Виды *PNPSFk* и примитивы, формируемые тестом *March C*− для их обнаружения

Примитив	Предыдущая фаза	⇑⇓ (*w*0)	⇑(*r*0,*w*1)	⇑(*r*1,*w*0)	⇓(*r*0,*w*1)
	Текущая фаза	⇑(*r*0,*w*1)	⇑(*r*1,*w*0)	⇓(*r*0,*w*1)	⇓(*r*1,*w*0)
	Последующая фаза	⇑(*r*1,*w*0)	⇓(*r*0,*w*1)	⇓(*r*1,*w*0)	⇑⇓(*r*0)
Виды обнаруживаемых неисправностей *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⟩

Максимальной полнотой покрытия неисправностей *PNPSFk* характеризуются маршевые тесты, синтезированные для обнаружения сложных кодочувствительных неисправностей, которые при однократном применении обнаруживают все восемь видов указанных ранее неисправностей *PNPSFk* (см. табл. 2). Среди наиболее известных тестов выделяют *March PS*: {⇑⇓(*w*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); ⇑(*r*0,*w*1,*r*1,*w*0,*r*0)} [26], сложность которого равняется 23*N*, и *March* 17*N*, часто называемый *Cheng test* [12]: {⇑⇓(*w*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)}. Указанные тесты достигают максимально возможной полноты покрытия неисправностей *PNPSFk*, определяемой выражением [17]

$$FC_{MAX}(PNPSFk) = \frac{Q_{MAX}(PNPSFk)}{Q_{TN}(PNPSFk)} \cdot 100\,\% = \frac{k-1}{k \cdot 2^{k-3}} \cdot 100\,\%. \qquad (5)$$

Отметим, что 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 величины, определяемой выражением (5) [3, 17].

Построение однократных маршевых тестов с максимальной полнотой покрытия неисправностей *PNPSFk*. Первоначально рассмотрим ряд особенностей примитивов, необходимых для обнаружения неисправностей *PNPSFk*, приведенных в предыдущем разделе. Для конкретного вида неисправностей *PNPSFk*, например ⟨0,0,0,…,0,↑,1,1,1,…,1⟩, примитив, обеспечивающий их обнаружение, может состоять из трех или двух фаз (см. табл. 2). В обоих случаях минимальная временная сложность реализации обоих примитивов (количество операций чтения (записи)) равняется 4*N*. Однако с учетом того, что конкретная фаза маршевого теста может участвовать в нескольких примитивах, средняя и суммарная сложности примитивов, описывающих обнаружение более чем одного вида неисправностей *PNPSFk*, может быть существенно меньше. Например, временная сложность теста *March C*−, равная суммарной сложности примитивов, составляет 10*N*, а сам тест обнаруживает четыре вида неисправностей *PNPSFk*. Поэтому при построении тестов необходимо, чтобы каждая фаза участвовала как можно в большем числе примитивов. В этом случае достигается минимальная сложность синтезируемого теста. Примерами таких тестов могут быть приведенный ранее тест *March 17N* и тест *March*

PNPSFk [17]: {⇑⇓(*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)}, сложность которого равна 18*N*.

Второй особенностью примитивов является возможность их построения с использованием в текущей фазе более одной операции записи *w*0 или *w*1, как это видно, например, из табл. 2. Однако независимо от сложности фаз теста примитивы, включающие такие фазы, могут обеспечить условия активизации и обнаружения не более чем двух видов неисправностей *PNPSFk*. Это следует из того, что текущая фаза может сформировать только два перехода (↑, ↓) в базовой ячейке для одного из четырех состояний в соседних ячейках (см. утверждение). Для формирования двух переходов (↑, ↓) необходимы как минимум две операции записи. Применение еще одной операции записи необходимо только для случая инвертирования ею содержимого памяти. Примером построения подобного примитива является случай его синтеза для обнаружения двух видов неисправностей *PNPSFk*, а именно ⟨0,0,0,…,0,↑,1,1,1,…,1⟩ и ⟨0,0,0,…,0,↓,1,1,1,…,1⟩. Соответствующий примитив может состоять из предыдущей фазы ⇑⇓(…,*w*0,…) и текущей вида ⇑(*r*0,*w*1,*r*1,*w*0,*r*0,*w*1) (см. тест *March PS*). Примитив, представленный фазами ⇑⇓(…,*w*1,…) и ⇓(*r*1,*w*0,*r*0,*w*1,*r*1,*w*0), также описывает обнаружение неисправностей ⟨0,0,0,…,0,↑,1,1,1,…,1⟩ и ⟨0,0,0,…,0,↓,1,1,1,…,1⟩. В то же время два примитива ⇑⇓(…,*w*0,…), ⇓(*r*0,*w*1,*r*1,*w*0,*r*0,*w*1) и ⇑⇓(…,*w*1,…), ⇑(*r*1,*w*0,*r*0,*w*1, *r*1,*w*0) 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 выше примитивов, можно построить маршевый тест, имеющий такую же полноту покрытия неисправностей *PNPSFk*, как и тест March *C*−. Примером подобного теста может быть тест {⇑⇓(*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)}, обнаруживающий, так же как и March *C*−, четыре вида неисправностей *PNPSFk*.

Основываясь на понятиях примитивов, приведенных в настоящей статье, можно построить маршевый тест, обеспечивающий требуемую полноту покрытия неисправностей *PNPSFk*.

Для случая тестов типа *March PS* процедура синтеза будет состоять из выбора набора примитивов, каждый из которых обеспечивает обнаружение двух видов неисправностей *PNPSFk*, а их совокупность, представляющая искомый тест, – всех восьми видов неисправностей. Анализ примитивов и обнаруживаемых ими пар неисправностей, приведенных в табл. 4, свидетельствует о том, что тест будет состоять как минимум из четырех фаз, кроме фазы инициализации.

Таблица 4

Неисправности *PNPSFk* и необходимые примитивы для их обнаружения

Обнаруживаемые пары неисправностей *PNPSFk*	Примитив		
	Предыдущая фаза	Текущая фаза	Последующая фаза
⟨0,0,0,…,0,↑,1,1,1,…,1⟩	⇑⇓(…,*w*0,…)	⇑(*r*0,*w*1,*r*1,*w*0,*r*0,*w*1)	–
⟨0,0,0,…,0,↓,1,1,1,…,1⟩	⇑⇓(…,*w*1,…)	⇓(*r*1,*w*0,*r*0,*w*1,*r*1,*w*0)	–
⟨1,1,1,…,1,↑,0,0,0,…,0⟩	⇑⇓(…,*w*0,…)	⇓(*r*0,*w*1,*r*1,*w*0,*r*0,*w*1)	–
⟨1,1,1,…,1,↓,0,0,0,…,0⟩	⇑⇓(…,*w*1,…)	⇑(*r*1,*w*0,*r*0,*w*1,*r*1,*w*0)	–
⟨0,0,0,…,0,↑,0,0,0,…,0⟩	⇑⇓(…,*w*0,…)	⇑⇓(*r*0,*w*1,*r*1,*w*0,*r*0)	–
⟨0,0,0,…,0,↓,0,0,0,…,0⟩	⇑⇓(…,*w*0,…)	⇑⇓(*r*0,*w*1,*r*1,*w*0)	⇑⇓(*r*0,…)
⟨1,1,1,…,1,↑,1,1,1,…,1⟩	⇑⇓(…,*w*1,…)	⇑⇓(*r*1,*w*0,*r*0,*w*1,*r*1)	–
⟨1,1,1,…,1,↓,1,1,1,…,1⟩	⇑⇓(…,*w*1,…)	⇑⇓(*r*1,*w*0,*r*0,*w*1)	⇑⇓(*r*1,…)

Итогом такого синтеза могут быть следующие тесты типа *March PS*:

{⇑⇓(*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)},
{⇑⇓(*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)},
{⇑⇓(*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,*w*1,*r*1,*w*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,*w*1,*r*1,*w*0,*r*0,*w*1)}.

Приведенные тесты являются результатом эвристического подхода к синтезу тестов, заключающегося в подборе необходимых фаз и их сочетания для обеспечения максимальной полноты покрытия тестом неисправностей *PNPSFk*. Отметим, что полученный результат обеспечивает максимальную эффективность полученных тестов только по отношению к неисправностям *PNPSFk* и не гарантирует их эффективность по отношению к другим видам неисправностей, в том числе и более простым их разновидностям. Последнее утверждение также справедливо и для менее сложных тестов, в которых используется особенность фаз теста участвовать более чем в одном примитиве. Приведем примеры оптимальных с точки зрения временной сложности тестов:

$\{\Uparrow\Downarrow(w0); \Uparrow(r0,w1); \Uparrow(r1,w0); \Downarrow(r0,w1); \Downarrow(r1,w0,r0,w1); \Downarrow(r1,w0); \Uparrow(r0,w1,r1,w0,r0)\}$,
$\{\Uparrow\Downarrow(w0); \Uparrow(r0,w1); \Uparrow(r1,w0,r0,w1); \Downarrow(r1,w0); \Uparrow(r0,w1,r1,w0); \Downarrow(r0,w1); \Downarrow(r1,w0,r0)\}$,
$\{\Uparrow\Downarrow (w1); \Uparrow(r1,w0); \Uparrow(r0,w1,r1,w0); \Uparrow(r0,w1); \Downarrow(r1,w0); \Downarrow(r0,w1); \Uparrow(r1,w0,r0,w1); \Uparrow\Downarrow(r1)\}$.

Очевидно, что, применяя рассмотренную методику, можно построить достаточно большое множество тестов, характеризующихся максимальной полнотой покрытия неисправностей *PNPSFk* (5). Среди их многообразия выделим тест *March OP*:

$$\{\Uparrow\Downarrow(w0); \Uparrow(r0,w1); \Uparrow(r1,w0); \Downarrow(r0,w1); \Downarrow(r1,w0,r0,w1); \Downarrow(r1,w0); \Uparrow(r0,w1,r1,w0); \Uparrow\Downarrow(r0)\}. \quad (6)$$

Тест *March OP* является расширением теста *March C−* в части добавления двух примитивов: $\Uparrow\Downarrow(r1,w0,r0,w1)$, $\Uparrow\Downarrow(r1,\ldots)$ и $\Uparrow\Downarrow(r0,w1,r1,w0)$, $\Uparrow\Downarrow(r0,\ldots)$, обеспечивающих обнаружение четырех дополнительных видов неисправностей *PNPSFk* (см. табл. 4). Временная сложность $18N$ теста *March OP* соответствует минимально достижимой сложности известных тестов с максимальной полнотой покрытия для указанных неисправностей. Тест *March OP* по аналогии с тестом *Cheng* может быть адаптирован для обнаружения неисправностей *PNPSFk* при заданных значениях k и стандартных начальных состояний памяти, необходимых для их обнаружения. Это достигается путем удаления первой операции чтения, что приводит к уменьшению сложности теста до $17N$. Однако для неразрушающего тестирования памяти, при котором фаза инициализации отсутствует, указанная первая операция чтения является необходимой.

При классической реализации любого из рассмотренных выше тестов, обнаруживающих максимальное количество неисправностей *PNPSFk*, численные значения $FC_{\max}$(*PNPSFk*) при однократном их применении (табл. 5) принимают невысокие значения, особенно с ростом величины k.

Таблица 5

Полнота покрытия, %

PNPSFk	*PNPSF*3	*PNPSF*4	*PNPSF*5	*PNPSF*6	*PNPSF*7	*PNPSF*8	*PNPSF*9
$FC_{\max}$(*PNPSFk*)	66,66	37,50	20,0	10,41	5,35	2,73	1,38
$FC_{March\ C-}$(*PNPSFk*)	50	25	12,5	6,25	3,12	1,56	0,78
$FC_{\min}$(*PNPSFk*)	12,50	6,25	3,12	1,56	0,78	0,39	0,19

Максимальная полнота покрытия, равная 66,66 % для $k = 3$, достигается ранее рассмотренными маршевыми тестами, покрывающими все восемь видов неисправностей *PNPSFk*. Минимальной полнотой покрытия характеризуются тесты, обнаруживающие только один вид неисправностей *PNPSFk*, например *MATS*, а достаточно высокую покрывающую способность показывает тест *March C−*.

Радикальным увеличением полноты покрытия сложных кодочувствительных неисправностей памяти является применение многократных маршевых тестов, хорошо изученных для случая детерминированных начальных состояний памяти с известной физической топологией [3, 6, 11–14, 26] и для случая, когда топология неизвестна [3, 24, 27]. Большой цикл работ по неразрушающему тестированию (*transparent testing*), по сути, представляющий собой многократное тестирование, а также псевдоисчерпывающее (*pseudoexhaustive*) тестирование запоминающих

устройств, основан на применении маршевых тестов с изменяющимся содержимым памяти [3, 16, 17, 27–31]. Модификация адресных последовательностей для улучшения качества тестов памяти первоначально была предложена и исследована в работе [33] и получила свое развитие в работах [5, 17, 34, 35], однако в этих работах рассматривались многократные тесты небольшой кратности, использующие стандартные детерминированные адресные последовательности [5, 17, 34, 35].

Оценка эффективности обнаружения неисправностей *PNPSFk* многократными маршевыми тестами с изменяемыми адресными последовательностями. Развитие методов многократного применения маршевых тестов привело к появлению псевдоисчерпывающих тестов памяти [5, 16]. Сущность подобных тестов заключается в формировании в произвольных k из N ячейках памяти всевозможных 2^k двоичных комбинаций. Как показано в [16], основой высокой эффективности таких тестов является формирование восьми видов орбит, представляющих собой набор двоичных комбинаций в произвольных k из N ячейках памяти. Вид орбит и их количество, формируемое маршевым тестом, используются для определения покрывающей способности теста для различных типов неисправностей памяти. Применяя понятие орбиты, по аналогии с псевдоисчерпывающими тестами введем восемь видов орбит, представляющих условия активизации восьми видов неисправностей *PNPSFk*, приведенных в табл. 2. В табл. 6 представлены орбиты O_0, O_1, O_2 и O_3, характерные для менее сложных маршевых тестов типа *March C–*, а в табл. 7 – орбиты Q_0, Q_1, Q_2 и Q_3, присущие более сложным их разновидностям.

Таблица 6

Орбиты O_0, O_1, O_2 и O_3, формируемые тестами типа *March C–*

O_0					O_1					O_2					O_3				
i_4	i_3	i_2	i_1	i_0	i_4	i_3	i_2	i_1	i_0	i_4	i_3	i_2	i_1	i_0	i_4	i_3	i_2	i_1	i_0
0	0	0	0	↑	↑	0	0	0	0	1	1	1	1	↓	↓	1	1	1	1
0	0	0	↑	1	1	↑	0	0	0	1	1	1	↓	0	0	↓	1	1	1
0	0	↑	1	1	1	1	↑	0	0	1	1	↓	0	0	0	0	↓	1	1
0	↑	1	1	1	1	1	1	↑	0	1	↓	0	0	0	0	0	0	↓	1
↑	1	1	1	1	1	1	1	1	↑	↓	0	0	0	0	0	0	0	0	↓

Таблица 7

Дополнительные орбиты Q_0, Q_1, Q_2 и Q_3, формируемые тестами типа *March PS*

Q_0					Q_1					Q_2					Q_3				
i_4	i_3	i_2	i_1	i_0	i_4	i_3	i_2	i_1	i_0	i_4	i_3	i_2	i_1	i_0	i_4	i_3	i_2	i_1	i_0
0	0	0	0	↑	↑	0	0	0	0	1	1	1	1	↓	↓	1	1	1	1
0	0	0	↑	0	0	↑	0	0	0	1	1	1	↓	1	1	↓	1	1	1
0	0	↑	0	0	0	0	↑	0	0	1	1	↓	1	1	1	1	↓	1	1
0	↑	0	0	0	0	0	0	↑	0	1	↓	1	1	1	1	1	1	↓	1
↑	0	0	0	0	0	0	0	0	↑	↓	1	1	1	1	1	1	1	1	↓

Каждая из представленных орбит описывает условие активизации одного из видов обнаруживаемых неисправностей *PNPSFk* и соответствует фазам маршевого теста, представляющим примитив для активизации и обнаружения данного вида *PNPSFk* (см. табл. 2). Нетрудно заметить, что только одну орбиту O_0 формирует классический тест *MATS*, две орбиты O_0 и O_1 – тест *MATS++*, четыре орбиты O_0, O_1, O_2 и O_3 – тесты типа *March C–* и, наконец, максимальное количество орбит, дополнительно включающих орбиты Q_0, Q_1, Q_2 и Q_3, генерируют тесты типа *March PS*. Отметим, что произвольный маршевый тест генерирует произвольное сочетание орбит, что и определяет его эффективность.

При однократной реализации маршевого теста с изменяемыми начальными условиями, в том числе и с измененной адресной последовательностью, полнота покрытия кодочувстви-

тельных неисправностей *PNPSFk*, так же как и любых других неисправностей, количественно остается неизменной [6, 33]. Отличием являются только конкретные конфигурации неисправностей *PNPSFk*, которые обнаруживаются либо не обнаруживаются при реализации теста с заданной последовательностью адресов [33]. В качестве иллюстрации последнего утверждения приведем пример реализации теста *MATS* для двух отличающихся адресных последовательностей, в результате применения которых формируются две различные орбиты O_{01} и O_{02}. В первом случае для произвольных $k = 5$ из N ячеек с адресами $i_0 < i_1 < i_2 < i_3 < i_4$ тестом *MATS* будут обнаружены пять неисправностей *PNPSFk* ⟨0,0,0,0,↑⟩, ⟨0,0,0,↑,1⟩, ⟨0,0,↑,1,1⟩, ⟨0,↑,1,1,1⟩ и ⟨↑,1,1,1,1⟩, образующих орбиту O_{01} (табл. 8).

Таблица 8

Орбиты O_{01} и O_{02}

O_{01}					O_{02}				
i_4	i_3	i_2	i_1	i_0	i_4	i_3	i_2	i_1	i_0
0	0	0	0	↑	0	0	↑	0	0
0	0	0	↑	1	0	0	1	0	↑
0	0	↑	1	1	↑	0	1	0	1
0	↑	1	1	1	1	↑	1	0	1
↑	1	1	1	1	1	1	1	↑	1

При изменении порядка адресов в новой адресной последовательности теста *MATS*, для которой, например, окажется, что $i_2 < i_0 < i_4 < i_3 < i_1$, этот тест обнаружит такое же количество неисправностей *PNPSFk* в тех же $k = 5$ из N ячейках памяти, но уже других, а именно ⟨0,0,1,0,↑⟩, ⟨1,1,1,↑,1⟩, ⟨0,0,↑,0,0⟩, ⟨1,↑,1,0,1⟩ и ⟨↑,0,1,0,1⟩.

Для последующих итераций многократного тестирования памяти будем применять нулевые начальные состояния ячеек памяти и изменяемые адресные последовательности, которые представляют собой случайные последовательности адресов из множества $N!$ возможных последовательностей. Несмотря на ряд ограничений, таких как количество адресов N и их формирование без повторений, представляется возможным генерирование подобных последовательностей, максимально близких по своим свойствам к случайным последовательностям [5, 16].

В отличие от многократных маршевых тестов, основанных на изменении начального состояния памяти, при многократном тестировании с изменяемыми адресными последовательностями маршевый тест должен удовлетворять необходимому условию обнаружения неисправностей *PNPSFk*, для которых базовая ячейка выполняет оба перехода, а именно переходы ↑ и ↓. Это необходимо для достижения полноты покрытия, близкой к 100 % при большом числе l повторений теста. В частности, тест *MATS* при многократном его применении позволяет достичь только 50 % полноты покрытия, так как обнаруживает лишь те неисправности, в которых базовая ячейка не может выполнить только один переход (↑).

Если предположить, что однократное применение маршевого теста позволяет обнаруживать неисправности *PNPSFk* с полнотой покрытия $FC_{Test}(PNPSFk)$, то l-кратное его использование при произвольных (случайных) адресных последовательностях позволяет достичь полноты покрытия, вычисляемой согласно выражению

$$FC_{Test}(PNPSFk, l) = \left(1 - \left(1 - \frac{FC_{Test}(PNPSFk)}{100\ \%}\right)^{l}\right) \cdot 100\ \%. \qquad (7)$$

Для трех разновидностей тестов, а именно тестов типа *MATS++*, *March C−* и тестов семейства *March PS*, соответствующие выражения для полноты покрытия имеют вид

$$FC_{MATS++}(PNPSFk,l)=\left(1-\left(1-\frac{1}{2^{k-1}}\right)^{l}\right)\cdot 100\ \%,$$

$$FC_{March\ C-}(PNPSFk,l)=\left(1-\left(1-\frac{1}{2^{k-2}}\right)^{l}\right)\cdot 100\ \%, \qquad (8)$$

$$FC_{March\ PS}(PNPSFk,l)=\left(1-\left(1-\frac{k-1}{k\cdot 2^{k-3}}\right)^{l}\right)\cdot 100\ \%.$$

Соответствие аналитических метрик (8) и полноты покрытия неисправностей *PNPSFk* реальным значениям было оценено путем экспериментальных исследований для небольших объемов памяти и трех значений величины *k*. Исходное состояние памяти для каждой итерации многократного теста принималось неизменным, а сами реализации тестов отличались только адресной последовательностью, применяемой в каждой из последующих итераций. В трех последовательных реализациях тестов *MATS++*, *March C–* и *March PS* адреса ячеек памяти формировались в псевдослучайной последовательности с использованием генераторов *M*-последовательностей [5, 17]. В результате суммарная полнота покрытия при двукратном и трехкратном применении маршевых тестов заметно возрастает и в целом соответствует аналитическим оценкам, что подтверждают данные табл. 9 и 10.

Таблица 9

Экспериментальные значения полноты покрытия *FC* для случайных последовательностей адресов, %

PNPSFk	*PNPSF*3			*PNPSF*4			*PNPSF*5		
l	1	2	3	1	2	3	1	2	3
FC_{MATS++}(*PNPSFk*)	24,90	42,74	55,83	12,48	22,61	30,95	6,31	11,87	16,91
$FC_{March\ C-}$ (*PNPSFk*)	49,87	72,74	83,04	24,93	41,42	53,23	12,46	22,14	30,02
$FC_{March\ PS}$ (*PNPSFk*)	66,31	85,13	92,54	37,39	58,07	71,11	20,01	34,93	45,28

Таблица 10

Аналитические оценки полноты покрытия *FC* для случайных последовательностей адресов, %

PNPSFk	*PNPSF*3			*PNPSF*4			*PNPSF*5		
l	1	2	3	1	2	3	1	2	3
FC_{MATS++}(*PNPSFk*)	25	43,75	57,81	12,5	23,43	33,01	6,25	12,10	17,60
$FC_{March\ C-}$ (*PNPSFk*)	50	75	87,5	25	43,75	57,81	12,5	23,43	33,01
$FC_{March\ PS}$ (*PNPSFk*)	66,66	88,88	96,29	37,5	60,93	75,58	20	36	48,80

Анализ приведенных результатов позволяет сделать вывод о достаточно хорошей эффективности многократных тестов с изменяемыми случайными адресными последовательностями для обнаружения неисправностей *PNPSFk*. Видно, что использование трехкратного теста типа *March PS* позволяет достичь полноты покрытия $FC_{March\ PS}$ (*PNPSF*5), практически близкой к 50 %, а максимальная полнота покрытия, близкая к 100 %, достижима только для достаточно больших значений *l*. В то же время следует отметить, что выражение (7) позволяет оценить кратность теста *l* для требуемой полноты покрытия FC_{Test}(*PNPSFk, l*). Эта величина определяется из полученного на основании (7) выражения

$$l=\left\lceil\frac{\log_{10}\left(1-\frac{FC_{Test}(PNPSFk,l)}{100\ \%}\right)}{\log_{10}\left(1-\frac{FC_{Test}(PNPSFk)}{100\ \%}\right)}\right\rceil. \qquad (9)$$

Например, для получения требуемой полноты покрытия $FC_{March\ PS}(PNPSF5,\ l)$ = 30 % неисправностей *PNPSF*5 с помощью многократного теста типа *March PS* величина кратности, определенная согласно (9), равняется $l = \lceil (-0{,}155)/(-0{,}097) \rceil = 2$. Отметим, что в соответствии с (5) $FC_{March\ PS}$ (*PNPSF*5) = 20 %, а полученный результат согласуется с экспериментальными данными, приведенными в табл. 9. В то же время для желаемых величин полноты покрытия, близких к 100 %, величина l может рассматриваться как оценка кратности многократного теста. Для полноты покрытия $FC_{MarchPS}(PNPSF5,\ l)$ = 95 % значение этой оценки равняется 14.

При определении эффективности тестирования запоминающих устройств часто возникает вопрос о средней величине кратности многократного теста, покрывающего все заданные неисправности, т. е. обеспечивающего 100%-ю полноту покрытия. Подобная задача решалась в рамках псевдоисчерпывающего тестирования [5, 16, 30] и сводилась к классической задаче собирателя купонов [36]. Теоретические выводы и оценки, полученные для данной задачи, хорошо согласуются с экспериментальными результатами, приведенными для формирования псевдоисчерпывающих тестовых процедур на базе тестов *MATS*++ и *March C*– [16, 30].

Для неисправностей *PNPSFk* и многократных маршевых тестов с изменяемыми случайными адресными последовательностями примем те же допущения и ограничения, которые принимались для генерирования многократных тестов, реализующих псевдоисчерпывающие тесты [16]. Тогда согласно задаче собирателя купонов оценка среднего значения $l_{Test}(PNPSFk)$ кратности l для тестов типа *MATS*++, *March C*– и *March PS* вычисляется на основании следующих выражений:

$$
\begin{aligned}
l_{MATS++}(PNPSFk) &= \frac{k\cdot 2^k}{2k}\cdot\sum_i^{k\cdot 2^k}\frac{1}{i} = 2^{k-1}\cdot\sum_i^{k\cdot 2^k}\frac{1}{i};\\
l_{MarchC-}(PNPSFk) &= \frac{k\cdot 2^k}{4k}\cdot\sum_i^{k\cdot 2^k}\frac{1}{i} = 2^{k-2}\cdot\sum_i^{k\cdot 2^k}\frac{1}{i};\\
l_{MarchPS}(PNPSFk) &= \frac{k\cdot 2^k}{8\cdot(k-1)}\cdot\sum_i^{k\cdot 2^k}\frac{1}{i} = \frac{k\cdot 2^{k-3}}{k-1}\cdot\sum_i^{k\cdot 2^k}\frac{1}{i}.
\end{aligned}
\qquad (10)
$$

Соотношения (10) позволяют определить среднее значение кратности l соответствующего теста для обнаружения всех $k2^k$ возможных неисправностей *PNPSFk*. Численные значения данной характеристики приведены в табл. 11.

Таблица 11

Среднее значение кратности теста, необходимое для обнаружения всех неисправностей *PNPSFk*

k	2	3	4	5	6	7	8	9
$l_{MATS++}(PNPSFk)$	5,434	15,104	37,952	85,904	208,93	472,00	1049,9	2307,3
$l_{March\ C-}(PNPSFk)$	2,717	7,556	18,976	42,952	104,46	236,00	524,93	1153,7
$l_{March\ PS}(PNPSFk)$	1	4,164	12,651	26,845	62,678	137,66	299,95	648,93

Данные табл. 11 позволяют сделать вывод, что маршевые тесты, специально ориентированные на обнаружение неисправностей *PNPSFk* (тесты типа *March PS*), при многократном их применении имеют несомненное преимущество для случая обнаружения подобных неисправностей. Однако среднее значение кратности принимает существенные значения уже для величин $k > 5$.

Заключение. В работе проведен анализ моделей неисправностей запоминающих устройств и методов их обнаружения. Получены оценки полноты покрытия сложных кодочувствительных неисправностей *PNPSFk* маршевых тестов. Введено понятия примитива, описывающего условия активизации и обнаружения неисправностей, и приведены примеры построения тестов, основанных на реализации требуемого множества подобных примитивов. Показана невысокая эффективность однократного применения тестов для обнаружения неисправностей *PNPSFk*,

которая уменьшается с увеличением количества *k* ячеек памяти, участвующих в неисправности. Обосновано применение многократных маршевых тестов с изменяемыми адресными последовательностями. Получены аналитические оценки кратности многократных маршевых тестов для обнаружения неисправностей *PNPSFk*, позволяющие определить как полноту покрытия тестом неисправностей *PNPSFk*, так и требуемое значение кратности многократного теста.

Список использованных источников

1. The International Technology Roadmap for Semiconductors: 2003 Edition. – San Jose, CA, USA, Semiconductor Industry Association, 2003. – 65 p.

2. Sharma, A. K. Advanced Semiconductor Memories: Architectures, Designs, and Applications / A. K. Sharma. – London : John Wiley & Sons, 2003. – 652 p.

3. Wang, L.-T. VLSI Test Principles and Architectures: Design for Testability / L.-T. Wang, C.-W. Wu, X. Wen. – Amsterdam : Elsevier, 2006. – 808 p.

4. Bushnell, M. L. Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits / M. L. Bushnell, V. D. Agrawal. – N. Y., USA : Kluwer Academic Publishers, 2000. – 690 p.

5. Ярмолик, В. Н. Контроль и диагностика вычислительных систем / В. Н. Ярмолик. – Минск : Бестпринт, 2019. – 387 с.

6. Goor, A. J. Testing Semiconductor Memories: Theory and Practice / A. J. Goor. – Chichester, UK : John Wiley & Sons, 1991. – 536 p.

7. Hayes, J. P. Detection of pattern-sensitive faults in random access memories / J. P. Hayes // IEEE Transactions on Computer. – 1975. – Vol. 24, no. 2. – P. 150–157.

8. Anderson, K. Device manufacturers test problems / K. Anderson // Proc. of IEEE Semiconductor Test Symp. – Cherry Hill, NJ, USA, 1972. – P. 17–26.

9. Suk, D. S. Test procedures for a class of pattern-sensitive faults in semiconductor random-access memories / D. S. Suk, S. M. Reddy // IEEE Transactions on Computer. – 1980. – Vol. 29, no. 6. – P. 419–429.

10. Hayes, J. P. Testing memories for single-cell pattern-sensitive fault / J. P. Hayes // IEEE Transactions on Computer. – 1980. – Vol. 29, no. 3. – P. 249–254.

11. Cheng, K.-L. Efficient neighborhood pattern-sensitive fault test algorithms for semiconductor memories / K.-L. Cheng, M.-F. Tsai, C. T. Wu // Proc. of 19th IEEE VLSI Test Symp. – Marina Del Rey, CA, USA, 2001. – P. 225–237.

12. Cheng, K.-L. Neighborhood pattern-sensitive fault testing and diagnostics for random-access memories / K.-L. Cheng, M.-F. Tsai, C. T. Wu // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2002. – Vol. 21, no. 11. – P. 284–267.

13. Cascaval, P. Efficient march test for 3-coupling faults in random access memories / P. Cascaval, S. Bennett // Microprocessors and Microsystems. – 2001. – Vol. 24, no. 10. – P. 501–509.

14. Kang, D.-C. An efficient built-in self-test algorithm for neighborhood pattern sensitive faults in high-density memories / D.-C. Kang, S.-B. Cho // Proc. of 4th Korea-Russia Intern. Symp. (KORUS 2000). – Ulsan, South Korea, 2000. – Vol. 2. – P. 218–223.

15. Cockburn, B. F. Deterministic tests for detecting scrambled pattern-sensitive faults in RAMs / B. F. Cockburn // Proc. IEEE Intern. Workshop Memory Technology Design and Testing (MTDT). – San Jose, CA, USA, 1995. – P. 117–122.

16. Ярмолик, В. Н. Псевдоисчерпывающее тестирование запоминающих устройств на базе маршевых тестов типа March A / В. Н. Ярмолик, И. Мрозек, С. В. Ярмолик // Информатика. – 2020. – № 2(17). – С. 54–70.

17. Ярмолик, С. В. Маршевые тесты для самотестирования ОЗУ / С. В. Ярмолик, А. П. Занкович, А. А. Иванюк. – Минск : Бестпринт, 2009. – 270 с.

18. Franklin, M. A built in self-test algorithm for row/column pattern sensitive faults in RAMs / M. Franklin, K. Saluja, K. Kinoshita // IEEE J. of Solid-State Circuits. – 1990. – Vol. 25, no. 2. – P. 514–524.

19. Sfikas, Y. Physical design oriented DRAM neighborhood pattern sensitive fault testing / Y. Sfikas, Y. Tsiatouhas // Proc. of 12th Intern. Symp. on Design and Diagnostics of Electronic Circuits & Systems (DDECS). – Liberec, Czech Republic, 2009. – P. 108–113.

20. Parallel testing of multi-port static random access memories / F. Karimi [et al.] // Microelectronics J. – 2003. – Vol. 34, no. 1. – P. 3–21.

21. Min, D.-S. Multiple twisted data line techniques for coupling noise reduction in embedded DRAMS / D-S. Min, D. Langer // IEEE Custom Integrated Circuits Conf. – San Diego, CA, USA, 1999. – P. 231–234.

22. Kang, D.-C. An efficient built-in self-test algorithm for neighborhood pattern and bit-line-sensitive faults in high density memories / D.-C. Kang, S. M. Park, S.-B. Cho // ETRI J. – 2004. – Vol. 26, no. 6. – P. 520–534.

23. Goor, A. J. van de. Disturb neighborhood pattern sensitive fault / A. J. van de Goor, I. B. S. Tlili // IEEE Intern. VLSI Test Symp. – San Diego, CA, USA, 1997. – P. 37–45.

24. Yarmolik, V. N. Transparent memory testing for pattern-sensitive faults / V. N. Yarmolik, M. G. Karpovsky // Proc. of Intern. Test Conf. – Washington DC, USA, 1994. – P. 860–869.

25. Cockburn, B. E. Synthesized transparent BIST for detecting scrambled pattern-sensitive faults in RAMs / B. E. Cockburn, Y. F. Sat // Proc. of the IEEE Intern. Test Conf. – Washington DC, USA, 1995. – P. 23–32.

26. Yarmolik, V. March PS(23N) test for DRAM pattern-sensitive faults / V. Yarmolik, Y. Klimets, S. Demidenko // Proc. Seventh IEEE Asian Test Symp. (ATS). – Singapore, 1998. – P. 354–357.

27. Mrozek, I. Multi-run Memory Tests for Pattern Sensitive Faults / I. Mrozek. – Cham : Springer International Publishing AG, 2019. – 135 p.

28. Nicolaidis, M. Transparent BIST for RAMs / M. Nicolaidis // Proc. IEEE Intern. Test Conf. – Washington DC, USA, 1992. – P. 598–607.

29. Неразрушающее тестирование запоминающих устройств / В. Н. Ярмолик [и др.]. – Минск : Бестпринт, 2005. – 230 с.

30. Ярмолик, В. Н. Псевдоисчерпывающее тестирование запоминающих устройств на базе многократных маршевых тестов / В. Н. Ярмолик, И. Мрозек, В. А. Леванцевич // Информатика. – 2018. – № 1(15). – С. 110–121.

31. Das, D. Exhaustive and near-exhaustive memory testing techniques and their BIST implementations / D. Das, M. G. Karpovsky // J. of Electronic Testing. – 1997. – Vol. 10. – P. 215–229.

32. Ярмолик, С. В. Итеративные почти псевдоисчерпывающие вероятностные тесты / С. В. Ярмолик, В. Н. Ярмолик // Информатика. – 2010. – № 2(26). – С. 66–75.

33. Niggemeyer, D. Integration of non-classical faults in standard march tests / D. Niggemeyer, M. Redeker, J. Otterstedt // Proc. IEEE Intern. Workshop on Memory Technology, Design and Testing. – San Jose, USA, 1998. – P. 91–96.

34. Ярмолик, В. Н. Адресные последовательности для многократного тестирования ОЗУ / В. Н. Ярмолик, С. В. Ярмолик // Информатика. – 2014. – № 2(42). – С. 124–136.

35. Ярмолик, С. В. Многократные неразрушающие маршевые тесты с изменяемыми адресными последовательностями / С. В. Ярмолик, В. Н. Ярмолик // Автоматика и телемеханика. – 2007. – № 2. – С. 21–30.

36. Flajolet, P. Birthday paradox, coupon collectors, caching algorithms and self-organizing search / P. Flajolet, D. Gardy, L. Thimonier // Discrete Applied Mathematics. – Vol. 39, no. 3 – P. 207–229.

References

1. *The International Technology Roadmap for Semiconductors: 2003 Edition*. San Jose, CA, USA, Semiconductor Industry Association, 2003, 65 p.

2. Sharma A. K. *Advanced Semiconductor Memories: Architectures, Designs, and Applications.* London, John Wiley & Sons, 2003, 652 p.

3. Wang L.-T., Wu C.-W., Wen X. *VLSI Test Principles and Architectures: Design for Testability*. Amsterdam, Elsevier, 2006, 808 p.

4. Bushnell M. L., Agrawal V. D. *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*. New York, USA, Kluwer Academic Publishers, 2000, 690 p.

5. Yarmolik V. N. Kontrol' i diagnostika vuchislitel'nuh system. *Monitoring and Diagnostics of Computer Systems*. Minsk, Bestprint, 2019, 387 p. (in Russian).

6. Goor A. J. *Testing Semiconductor Memories: Theory and Practice*. Chichester, UK, John Wiley & Sons, 1991, 536 p.

7. Hayes J. P. Detection of pattern-sensitive faults in random access memories. *IEEE Transactions on Computer*, 1975, vol. 24, no. 2, pp. 150–157.

8. Anderson K. Device manufacturers test problems. *Proceedings of IEEE Semiconductor Test Symposium*. Cherry Hill, NJ, USA, 1972, pp. 17–26.

9. Suk D. S., Reddy S. M. Test procedures for a class of pattern-sensitive faults in semiconductor random-access memories. *IEEE Transactions on Computer*, 1980, vol. 29, no. 6, pp. 419–429.

10. Hayes J. P. Testing memories for single-cell pattern-sensitive fault. *IEEE Transactions on Computer*, 1980, vol. 29, no. 3, pp. 249–254.

11. Cheng K.-L., Tsai M.-F., Wu C. T. Efficient neighborhood pattern-sensitive fault test algorithms for semiconductor memories. *Proceedings of 19th IEEE VLSI Test Symposium.* Marina Del Rey, CA, USA, 2001, pp. 225–237.

12. Cheng K.-L., Tsai M.-F., Wu C. T. Neighborhood pattern-sensitive fault testing and diagnostics for random-access memories. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, vol. 21, no. 11, pp. 284–267.

13. Cascaval P., Bennett S. Efficient march test for 3-coupling faults in random access memories. *Microprocessors and Microsystems*, 2001, vol. 24, no. 10, pp. 501–509.

14. Kang D.-C., Cho S.-B. An efficient built-in self-test algorithm for neighborhood pattern sensitive faults in high-density memories. *Proceedings of 4th Korea-Russia International Symposium (KORUS 2000).* Ulsan, South Korea, 2000, vol. 2, pp. 218–223.

15. Cockburn B. F. Deterministic tests for detecting scrambled pattern-sensitive faults in RAMs. *Proceedings IEEE International Workshop Memory Technology Design and Testing (MTDT*). San Jose, CA, USA, 1995, pp. 117–122.

16. Yarmolik V. N., Mrozek I., Yarmolik S. V. Pseudoischerpuvayuchie testirovanie zapominayuschih ustroistv na baze marchevuh testov tipa March A [Pseudo-exhaustive memory testing based on March A tests]. Informatika [*Informatics*], 2020, no. 2(17), pp. 54–70 (in Russian).

17. Yarmolik V. N., Zankovich A. P., Ivanuok A. A. Marshevue testu dlya samotestirovaniya OZU. *RAM Self-Test March Tests*. Minsk, Bestprint, 2009, 270 p. (in Russian).

18. Franklin M., Saluja K., Kinoshita K. A built in self-test algorithm for row/column pattern sensitive faults in RAMs. *IEEE Journal of Solid-State Circuits*, 1990, vol. 25, no. 2, pp. 514–524.

19. Sfikas Y., Tsiatouhas Y. Physical design oriented DRAM neighborhood pattern sensitive fault testing. *Proceedings of 12th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS).* Liberec, Czech Republic, 2009, pp. 108–113.

20. Karimi F., Irrinki S., Crosbuy T., Lombardi F. Parallel testing of multi-port static random access memories. *Microelectronics Journal*, 2003, vol. 34, no. 1, pp. 3–21.

21. Min D.-S., Langer D. Multiple twisted data line techniques for coupling noise reduction in embedded DRAMS. *IEEE Custom Integrated Circuits Conference.* San Diego, CA, USA, 1999, pp. 231–234.

22. Kang D.-C., Park S. M., Cho S.-B. An efficient built-in self-test algorithm for neighborhood pattern and bit-line-sensitive faults in high density memories. *ETRI Journal*, 2004, vol. 26, no. 6, pp. 520–534.

23. Goor A. J. van de, Tlili I. B. S. Disturb neighborhood pattern sensitive fault. *IEEE International VLSI Test Symposium.* San Diego, CA, USA, 1997, pp. 37–45.

24. Yarmolik V. N., Karpovsky M. G. Transparent memory testing for pattern-sensitive faults. *Proceedings of International Test Conference.* Washington DC, USA, 1994, pp. 860–869.

25. Cockburn B. E., Sat Y. F. Synthesized transparent BIST for detecting scrambled pattern-sensitive faults in RAMs. *Proceedings of the IEEE International Test Conference.* Washington DC, USA, 1995, pp. 23–32.

26. Yarmolik V., Klimets Y., Demidenko S. March PS(23N) test for DRAM pattern-sensitive faults. *Proceedings Seventh IEEE Asian Test Symposium (ATS).* Singapore, 1998, pp. 354–357.

27. Mrozek I. *Multi-run Memory Tests for Pattern Sensitive Faults*. Cham, Springer International Publishing AG, 2019, 135 p.

28. Nicolaidis M. Transparent BIST for RAMs. *Proceedings IEEE International Test Conference.* Washington DC, USA, 1992, pp. 598–607.

29. Yarmolik V. N., Murashko I. A., Kummert A., Ivaniuk A. A. Nerazrushayuschee testirovanie zapominayuschih ustroistv. *Transparent Memory Testing.* Minsk, Bestprint, 2005, 230 p. (in Russian).

30. Yarmolik V. N., Mrozek I., Levantsevich V. A. Pseudoischerpuvayuchie testirovanie zapominayuschih ustroistv na baze mnogokratnuch marchevuh testov [Pseudo-exhaustive memory testing based on multiple march tests]. Informatika [*Informatics*], 2018, no. 1(15), pp. 110–121 (in Russian).

31. Das D., Karpovsky M. G. Exhaustive and near-exhaustive memory testing techniques and their BIST implementations. *Journal of Electronic Testing*, 1997, vol. 10, pp. 215–229.

32. Yarmolik S. V., Yarmolik V. N. Iterativnue pochti pseudoischerpuvayuchie veroyatnostnue testu [Iterative near pseudo-exhaustive tests]. Informatika [*Informatics*], 2010, no. 2(26), pp. 66–75 (in Russian).

33. Niggemeyer D., Redeker M., Otterstedt J. Integration of non-classical faults in standard march tests. *Proceedings IEEE International Workshop on Memory Technology, Design and Testing.* San Jose, USA, 1998, pp. 91–96.

34. Yarmolik V. N., Yarmolik S. V. Adresnue posledovatel'nosti dlya mnogokratnogo testirovaniya OZU [Address sequences for multi run RAM testing]. Informatika [*Informatics*], 2014, no. 2(42), pp. 124–136 (in Russian).

35. Yarmolik S. V., Yarmolik V. N. Mnogokratnue nerazrushayuschie marshevue testu s izmenyaemumi adresnumi posledovatel'nostymi [Multiple non-destructive marching tests with variable address sequences]. Avtomatika i telemehanika [*Automation and Remote*], 2007, no. 2, pp. 21–30 (in Russian).

36. Flajolet P., Gardy D., Thimonier L. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229.

Информация об авторах

Ярмолик Вячеслав Николаевич, доктор технических наук, профессор, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: yarmolik10ru@yahoo.com

Леванцевич Владимир Александрович, магистр технических наук, старший преподаватель, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: lvn@bsuir.by

Деменковец Денис Викторович, магистр технических наук, старший преподаватель, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: demenkovets@bsuir.by

Мрозек Иренеуш, доктор, адъюнкт, Белостокский технический университет.
E-mail: i.mrozek@pb.edu.pl

Information about the authors

Vyacheslav N. Yarmolik, Dr. Sci. (Eng.), Professor, Belarusian State University of Informatics and Radioelectronics.
E-mail: yarmolik10ru@yahoo.com

Vladimer A. Levantsevich, M. Sci. (Eng.), Senior Lecture, Belarusian State University of Informatics and Radioelectronics.
E-mail: lvn@bsuir.by

Denis V. Demenkovets, M. Sci. (Eng.), Senior Lecture, Belarusian State University of Informatics and Radioelectronics.
E-mail: demenkovets@bsuir.by

Ireneusz Mrozek, Dr. Sci., Lecture, Bialystok University of Technology.
E-mail: i.mrozek@pb.edu.pl