ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ
LOGICAL DESIGN

УДК 004.33.054
https://doi.org/10.37661/1816-0301-2023-20-4-7-23

Оригинальная статья
Original Paper

Формальная модель описания и условия обнаружения связных неисправностей взаимного влияния запоминающих устройств

В. Н. Ярмолик✉, Д. В. Деменковец, В. В. Петровская, А. А. Иванюк

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

Аннотация

Цели. Целью работы являются разработка и анализ формальной модели описания сложных связных неисправностей взаимного влияния запоминающих устройств и формулировка необходимых и достаточных условий их обнаружения. Актуальность данных исследований заключается в том, что современные запоминающие устройства, характеризующиеся большим объемом хранимых данных и изготовленные по новейшим технологическим нормам, отличаются проявлением в них сложных разновидностей неисправностей.

Методы. Результаты исследования основаны на классической теории и практике однократных маршевых тестов (*March tests*) запоминающих устройств. В частности, в работе используются формальные математические модели описания неисправностей памяти и показывается их ограниченность для представления связных неисправностей взаимного влияния. Главная идея предлагаемого авторами подхода заключается в применении нового формального описания подобных неисправностей, ключевым элементом которого является использование ролей, выполняемых ячейками запоминающего устройства, участвующими в неисправности.

Результаты. Определены три основные роли, которые выполняют ячейки связной неисправности взаимного влияния, а именно: роль агрессора (A), роль жертвы (V), а также роль, включающая роли жертвы и агрессора (B), выполняемые двумя ячейками одновременно по отношению друг к другу. Показано, что сценарий реализации ролей ячеек неисправности памяти определяется применяемым маршевым тестом и в первую очередь используемой им адресной последовательностью обращения к ячейкам. Приведена процедура построения формальной модели связной неисправности, основу которой составляют роли, выполняемые ячейками, входящими в неисправность, и сценарий, задаваемый тестом. На базе нового формального описания связных неисправностей взаимного влияния сформулировано утверждение, определяющее необходимые и достаточные условия обнаружения подобных неисправностей. Показывается наличие необнаруживаемых неисправностей взаимного влияния и определяется возможность их обнаружения в рамках многократных маршевых тестов. Проведенные экспериментальные исследования подтвердили справедливость сформулированных положений статьи. На базе классического примера связной неисправности взаимного влияния показано выполнение необходимых и достаточных условий ее обнаружения однократным маршевым тестом.

Заключение. Результаты исследований подтверждают, что предложенная формальная математическая модель описания связных неисправностей взаимного влияния позволяет идентифицировать их покрытие маршевыми тестами. В рамках предложенной модели определяются необходимые и достаточные условия обнаружения связных неисправностей взаимного влияния маршевыми тестами, покрывающими одиночные связные неисправности.

Ключевые слова: тестирование вычислительных систем, неисправности взаимного влияния, связные неисправности, маршевые тесты запоминающих устройств, адресная последовательность

Для цитирования. Формальная модель описания и условия обнаружения связных неисправностей взаимного влияния запоминающих устройств / В. Н. Ярмолик [и др.] // Информатика. – 2023. – Т. 20, № 4. – С. 7–23. https://doi.org/10.37661/1816-0301-2023-20-4-7-23

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 05.08.2023
Подписана в печать | Accepted 27.09.2023
Опубликована | Published 29.12.2023

Formal description model and conditions for detecting linked coupling faults of the memory devices

Vyacheslav N. Yarmolik✉, Denis V. Demenkovets, Vita V. Petrovskaya, Alexander A. Ivaniuk

Belarusian State University of Informatics and Radioelectronics,
st. P. Brovki, 6, Minsk, 220013, Belarus
✉*E-mail: yarmolik10ru@yahoo.com*

Abstract

Objectives. The aim of the work is to develop and analyze a formal model for describing complex linked coupling faults of memory devices and to formulate the necessary and sufficient conditions for their detection. The relevance of these studies lies in the fact that modern memory devices, characterized by a large amount of stored data and manufactured according to the latest technological standards, are distinguished by the manifestation of complex types of faults in them.

Methods. The presented results are based on the classical theory and practice of march tests (*March tests*) of memory devices. In particular, the paper uses formal mathematical models for describing memory faults and shows their limitations for representing complex linked coupling faults. The main idea of the approach proposed by the authors is based on the use of a new formal description of such faults, the key element of which is the introduction of roles performed by the cells involved in the fault.

Results. Three main roles are defined that cells of the complex linked coupling faults perform, namely the role of the aggressor (*A*), the role of the victim (*V*), as well as the role of both the victim and the aggressor (*B*), performed by two cells simultaneously in relation to each other. It is shown that the scenario for the implementation of the roles of memory failure cells is determined by the marching test used, and, first of all, by the address sequence used to access the cells. The procedure for setting a formal model of a linked fault is given, the basis of which is the roles performed by the cells included in the fault and the scenario specified by the test. A statement is given that determines, on the basis of a new formal description of linked coupling faults, the necessary and sufficient conditions for the detection of such faults. The presence of undetectable linked coupling faults is shown, and the conditions for their detection are formulated using multiple March tests. The conducted experimental studies have confirmed the validity of the formulated provisions of the article. On the basis of the classical example of a linked coupling fault, the fulfillment of necessary and sufficient conditions for its detection by a single march test is shown.

Conclusion. The results of the research confirm that the proposed formal mathematical model for describing linked coupling faults makes it possible to determine their detection by marching tests. Within the framework of the proposed model, the necessary and sufficient conditions for detecting linked coupling faults by marching tests that detect single coupled faults are determined.

Keywords: testing of computing systems, coupling faults, linked faults, march tests of memory devices, address sequence

For citation. Yarmolik V. N. *Formal description model and conditions for detecting linked coupling faults of the memory devices*. Informatika [*Informatics*], 2023, vol. 20, no. 4, pp. 7–23 (In Russ.).
https://doi.org/10.37661/1816-0301-2023-20-4-7-23

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

Введение. Запоминающие устройства занимают доминирующее место среди аппаратной составляющей современных вычислительных систем. Спрос на высокоскоростную память с высокой степенью интеграции и низким энергопотреблением растет беспрецедентными темпами. Это связано с тем, что облачные вычисления, искусственный интеллект и телекоммуникационные технологии пятого поколения (5G) среди прочего позиционируются как основные направления четвертой промышленной революции. Для обеспечения характеристик современной памяти вычислительных систем, отвечающей требованиям новых технологических достижений, неизбежно возрастают необходимость и значимость тестирования запоминающих устройств [1]. Главной задачей тестирования памяти является обнаружение ее неисправных состояний, которые описываются различными моделями неисправностей [2–4].

Среди большого множества неисправностей запоминающих устройств выделяют неисправности взаимного влияния (*coupling faults*, *CF*), в которых участвуют две запоминающие ячейки. К разновидностям таких неисправностей относятся [2–8]:

1. Инверсные неисправности взаимного влияния (*inversion coupling faults*, *CFin*). В неисправности $CFin(a_i, a_j)$ участвуют ячейки a_i и a_j с адресами $i \neq j$. Ячейка a_i оказывает воздействие (влияние) на ячейку a_j и называется агрессором (*aggressor*, *A*), а ячейка a_j, на которую оказывается влияние, называется жертвой (*victim*, *V*). При наличии данной неисправности логический переход из 1 в 0 ($\downarrow$) или из 0 в 1 ($\uparrow$) в агрессоре a_i приводит к инверсии ($\updownarrow$) логического значения в ячейке-жертве a_j, что описывается двумя моделями инверсных неисправностей $CFin(a_i, a_j)$, а именно: $\langle\uparrow, \updownarrow\rangle$ и $\langle\downarrow, \updownarrow\rangle$ [2, 3]. Для представления соотношения адреса i агрессора и j жертвы в адресном пространстве памяти используют символы $\wedge$ и $\vee$. Это позволяет определить четыре различные инверсные неисправности $CFin(a_i, a_j) = \{\wedge\langle\uparrow, \updownarrow\rangle, \wedge\langle\downarrow, \updownarrow\rangle, \vee\langle\uparrow, \updownarrow\rangle, \vee\langle\downarrow, \updownarrow\rangle\}$. Согласно классическим определениям символ $\wedge$ означает, что адрес агрессора меньше адреса жертвы ($i < j$), а символ $\vee$, наоборот, больше ($i > j$). Известно, что однократные маршевые тесты реализуют обращение к ячейкам памяти путем последовательного перебора всех их адресов. Поэтому соотношение $i < j$ сопоставляет не только значения адресов ячеек a_i и a_j, но и время обращения к ним. При реализации элемента маршевого теста с возрастающей последовательностью адресов ($\Uparrow$) и соотношением адресов $i < j$ первоначально реализуется обращение к ячейке a_i и только затем, через некоторое время, к ячейке a_j.

2. Неисправности прямого действия (*idempotent coupling faults*, *CFid*). При наличии данной неисправности во время логического перехода из 1 в 0 или из 0 в 1 во влияющей ячейке a_i происходит принудительная установка определенного логического значения 0 или 1 в ячейке-жертве a_j [2, 3]. Различают восемь неисправностей прямого действия $CFid(a_i, a_j) = \{\wedge\langle\uparrow,0\rangle, \wedge\langle\uparrow,1\rangle, \wedge\langle\downarrow,0\rangle, \wedge\langle\downarrow,1\rangle, \vee\langle\uparrow,0\rangle, \vee\langle\uparrow,1\rangle, \vee\langle\downarrow,0\rangle, \vee\langle\downarrow,1\rangle\}$ [2, 3].

3. Статические неисправности взаимного влияния (*state coupling faults*, *CFst*). Переход зависимой ячейки в какое-либо состояние $a_j \in \{0,1\}$ происходит при определенном значении $a_i \in \{0,1\}$ влияющей ячейки. Возможны восемь неисправностей $CFst(a_i, a_j) = \{\wedge\langle 0,0\rangle, \wedge\langle 0,1\rangle, \wedge\langle 1,0\rangle, \wedge\langle 1,1\rangle, \vee\langle 0,0\rangle, \vee\langle 0,1\rangle, \vee\langle 1,0\rangle, \vee\langle 1,1\rangle\}$ [2, 3].

Различают одиночные (однократные) неисправности взаимного влияния, в качестве которых может быть любая из приведенных выше моделей неисправностей, и кратные (многократные), включающие некоторое подмножество одиночных неисправностей взаимного влияния [2]. В множестве кратных неисправностей выделяют несвязные кратные (*unlinked multiple*) неисправности взаимного влияния. Для таких неисправностей конкретная ячейка памяти участвует только в одной неисправности взаимного влияния, входящей в рассматриваемую кратную неисправность. В общем случае несвязная n-кратная неисправность взаимного влияния включает четное количество $r = 2n$ ячеек памяти, половина из которых являются агрессорами, а половина – жертвами. Таким образом, каждая ячейка однократной неисправности *CF*, входящей в n-кратную, имеет только одну роль, а именно агрессора (*A*) или жертвы (*V*).

В силу чрезвычайно большой емкости запоминающих устройств в настоящее время широко применяются и по-прежнему разрабатываются тесты с временной сложностью, линейно зависящей от емкости памяти. Подобные тесты имеют общее название «маршевые тесты» (*March tests*) [1–8] и применяются для случая бит-ориентированной памяти, каждая ячейка которой

хранит один бит. В общем случае маршевый тест состоит из конечного числа маршевых элементов (*march elements*) [2, 3]. В свою очередь каждый маршевый элемент содержит символ, определяющий порядок формирования адресной последовательности (*address sequence*) ячеек запоминающего устройства, задающей порядок обращения к ячейкам. Символ $\Uparrow$ указывает на последовательный перебор адресов памяти по возрастанию (в прямом порядке), а символ $\Downarrow$ – по убыванию (в обратном порядке). Маршевый элемент содержит последовательность операций чтения (*read*, *r*) и записи (*write*, *w*), заключенных в круглые скобки и разделяемых точкой с запятой. Переход к следующей ячейке согласно адресной последовательности осуществляется только после выполнения всех операций с текущей ячейкой в соответствии с маршевым элементом [1–6]. Например, простейший маршевый элемент, применяемый во многих тестах, имеет вид $\Uparrow(r0,w1)$. В соответствии с этим элементом согласно адресной последовательности из каждой ячейки читается 0 и записывается 1. Каждая операция чтения сопряжена со сравнением прочитанного значения с ожидаемым (0).

Обнаружение неисправностей запоминающих устройств как результат применения маршевого теста основано на получении прочитанных из его ячеек неверных значений, отличных от эталонных (ожидаемых). Таким образом, фиксируется неисправное состояние памяти и при необходимости проводится диагностическая процедура [2–4, 7, 8].

В рамках маршевых тестов проблема обнаружения как одиночных, так и кратных несвязных неисправностей взаимного влияния считается решенной [2–4]. Для этого существуют эффективные тесты, хорошо зарекомендовавшие себя на практике [2–6]. Маршевые тесты, обнаруживающие неисправности взаимного влияния, предполагают однократное их применение для тестирования запоминающих устройств со стандартным начальным состоянием запоминающих ячеек, как правило нулевым, и стандартной счетчиковой адресной последовательностью [2, 3].

Проблема обнаружения связных неисправностей взаимного влияния (*linked coupling faults*, *LCF*) однократными маршевыми тестами в силу многообразия и сложных механизмов проявления таких неисправностей остается практически открытой и требует дальнейших исследований [3–8].

В настоящей статье предлагается формальная модель связных неисправностей взаимного влияния, описывающая роли всех ячеек, участвующих в неисправности, и сценарий их проявления в соответствии с маршевым тестом и его адресной последовательностью. Определяются необходимые и достаточные условия обнаружения таких неисправностей в рамках однократных тестов, и оценивается эффективность их обнаружения многократными тестами.

Связные неисправности взаимного влияния. В структуре классических моделей неисправностей памяти связные неисправности занимают одно из самых видных мест в силу сложности их обнаружения существующими маршевыми тестами памяти [2–6]. Под связными неисправностями понимают совокупность одиночных неисправностей различных типов, включающие в себя общие ячейки, которые участвуют в нескольких неисправностях одновременно. Отметим, что связная неисправность может включать одиночные неисправности одного и того же типа.

Рассматривая случай связной неисправности взаимного влияния, состоящей из n однократных *CF*, отметим, что в подобной неисправности участвуют $r < 2n$ ячеек памяти. Пример такой неисправности *LCF*1, состоящей из двух ($n = 2$) однократных неисправностей взаимного влияния, приведен на рис. 1 [2–4]. На этом рисунке ячейки памяти a_i, a_j и a_k с адресами $i < j < k$ представлены последовательно слева направо во временной зависимости обращения к ним в процессе выполнения прямой фазы маршевого теста. Дуги означают отношение между ячейками в рамках однократной неисправности взаимного влияния и проводятся от агрессора к жертве. Обозначения *LCF*1 и *LCF*2 относятся к одной и той же *LCF*-неисправности, механизмы проявления которой различны и зависят от временной последовательности формирования адресов участвующих в ней ячеек.

Рис. 1. Связная неисправность взаимного влияния для $n = 2$

Fig. 1. Linked coupling fault for n = 2

Из рис. 1 видно, что в *LCF*1 и *LCF*2 участвуют $r = 3$ ячейки. Они включают по две однократные неисправности *CF*, каждая из которых состоит из двух ячеек. Приведенный пример неисправности *LCF*1 представлен в большом числе литературных источников и позиционируется как пример необнаруживаемых неисправностей в рамках классических маршевых тестов [2–4]. Действительно, если предположить, что адреса ячеек a_i, a_j и a_k формируются в последовательности i, j, k, ячейки a_i и a_k участвуют в неисправности $\wedge\langle\uparrow, \updownarrow\rangle$ и, кроме того, ячейки a_j и a_k образуют аналогичную неисправность $\wedge\langle\uparrow, \updownarrow\rangle$, то будем иметь случай необнаруживаемой неисправности *LCF*1. Подчеркнем, что факт необнаружения этой неисправности относится к однократным маршевым тестам, использующим адресную последовательность, для которой выполняется условие генерирования сначала адреса i, затем j и последним k. Тогда очевидно, что последовательное проявление двух одиночных неисправностей, входящих в связную неисправность *LCF*1 (рис. 1), приведет к последовательному двукратному инвертированию состояния ячейки a_k. В результате состояние ячейки a_k при обращении к ней будет всегда верным и, соответственно, неисправность *LCF*1 является необнаруживаемой. Рассмотренный пример связной неисправности *LCF*1 приводится в качестве аргумента о необнаружении связных неисправностей вообще, хотя, в частности, связная неисправность, представленная на рис. 1 и состоящая, например, из двух неисправностей $\wedge\langle\uparrow,0\rangle$ или $\wedge\langle\uparrow,1\rangle$, будет обнаруживаемой. Более того, при использовании однократного маршевого теста, в котором адреса формируются во временной последовательности i, затем k и, наконец, j, неисправность *LCF*1, обозначенная как *LCF*2, является обнаруживаемой независимо от того, какие две неисправности *CF* образуют *LCF*2.

Таким образом, можно заключить, что связная неисправность взаимного влияния *LCF* зависит от количества n одиночных неисправностей взаимного влияния и их разновидностей, а также от числа r ячеек, участвующих в *LCF*. Для общего случая важным является также количество ролей (A, V) и их вид для каждой ячейки *LCF*. Поясним это на примере неисправности *LCF*1, для которой ячейки a_i и a_j играют роль A (агрессора), и для обеих ячеек эта роль одна. В то же время ячейка a_k имеет две роли V (жертвы). В общем случае понятие «роль» определяет взаимодействие между двумя ячейками, поэтому у ячейки a_k две роли V как результат взаимодействия с a_i и a_j в рамках одиночных неисправностей *CF*, составляющих *LCF*1.

Абстрагируясь от реалистичных и нереалистичных связных неисправностей *LCF*, множества которых в основном определяются конструктивными и технологическими особенностями памяти, сформулируем для общего случая *LCF* ряд положений. Во-первых, предположив, что в *LCF* участвуют r ячеек, оценим минимальное $\min(n)$ и максимальное $\max(n)$ количество n одиночных неисправностей взаимного влияния, образующих *LCF*. Отметим, что *LCF* состоит из одиночных *CF* и каждая из r ячеек входит как минимум в одну *CF*. Важным фактором для получения граничных значений $\min(n)$ и $\max(n)$ является следующее утверждение.

Утверждение 1. *Если неисправность $CF(a_i, a_j)$ входит в связную неисправность LCF, состоящую из r ячеек, то количество ролей для одной из ячеек a_i или a_j, образующих CF, должно быть не менее двух, а их общее количество для обеих ячеек неисправности CF лежит в диапазоне от 3 до $4r - 6$.*

Справедливость данного утверждения следует из того, что если каждая из ячеек a_i и a_j, образующих одиночную *CF*, имеет только одну роль, то данная *CF* не входит в *LCF*. Такая неисправность не связана с другими неисправностями *CF*, так как у ячеек, входящих в данную неисправность, отсутствуют роли, связывающие ее с другими ячейками памяти, и, соответст-

венно, с другими неисправностями *CF*. Таким образом, наличие хотя бы у одной из ячеек a_i и a_j неисправности $CF(a_i, a_j)$ больше чем одной роли обеспечивает связь между другими одиночными *CF*, образующими *LCF*. Следовательно, минимальное количество ролей для неисправности *CF*, входящей в *LCF*, равняется трем. Верхняя оценка количества ролей для ячеек участвующей в *LCF* неисправности $CF(a_i, a_j)$, равная $4r - 6$, достигается в том случае, когда каждая из двух ячеек неисправности выполняет роли и агрессора, и жертвы по отношению к остальным $r - 2$ ячейкам. Еще две роли возможны между самими ячейками a_i и a_j неисправности *CF*. Тогда максимальное число ролей ячеек неисправности *CF*, входящей в *LCF*, определяется как $2(r - 2) + 2(r - 2) + 2 = 4r - 6$.

Минимальное количество $\min(n)$ одиночных *CF*, образующих *LCF*, равняется $r - 1$. Это следует из утверждения 1, что соответствует минимальному количеству ролей у каждой *CF*, две из которых описывают саму *CF*, а третья определяет связь с другой *CF*. Примером может быть случай, когда в каждой последующей неисправности *CF* ячейка-агрессор одновременно является и жертвой, агрессором для которой выступает ячейка-жертва текущей неисправности. Таким образом, значение $\min(n) = r - 1$ также достижимо для второго крайнего случая, когда одна из r ячеек *LCF* является агрессором для остальных $r - 1$ ячеек. Здесь также имеем $r - 1$ одиночных неисправностей *CF*, одна из ячеек каждой из которых выполняет более двух ролей, в данном случае $r - 1$ ролей. Примеры подобных неисправностей *LCF*3 и *LCF*4 для $r = 4$ приведены на рис. 2.

Максимальное количество $\max(n)$ одиночных неисправностей *CF*, образующих *LCF*, достигается для случая, когда любая возможная пара ячеек из r ячеек образует две *CF*. Соответственно, первая ячейка выполняет роль *A*, вторая *V*, и наоборот. Таким образом, $\max(n) = 2 \cdot \binom{r}{2} = r(r-1)$.

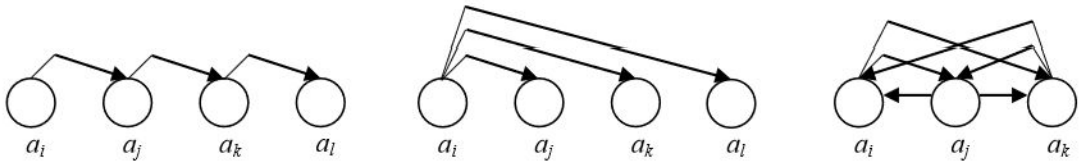


Рис. 2. Связные неисправности взаимного влияния для $r = 4$ и $r = 3$
Fig. 2. Linked coupling faults for r = 4 and r = 3

Пример *LCF*, состоящей из максимального количества одиночных *CF* для $r = 3$, приведен на рис. 2 как неисправность *LCF*5.

Таким образом, можно заключить, что для неисправности *LCF*, состоящей из r ячеек, количество n одиночных неисправностей *CF*, образующих эту неисправность, принадлежит диапазону

$$\min(n) = r - 1 \le n \le \max(n) = r(r-1). \quad (1)$$

Полученные предельные значения для количества n неисправностей *CF*, входящих в *LCF*, соответствуют аналогичным значениям для количества ребер в связных помеченных ориентированных графах, состоящих из r вершин. Пользуясь теорией графов, можно оценить число разновидностей неисправностей *LCF*, состоящих из r запоминающих ячеек, как общее число простых помеченных ориентированных графов, имеющих r вершин. Это число $2^{r(r-1)}$ растет экспоненциально от количества r ячеек и может быть использовано в качестве верхней оценки числа разновидностей *LCF*.

Отметим, что значения min(n) и max(n) количества n одиночных *CF*, образующих *LCF* (1), включающую r ячеек памяти, были получены без учета вида неисправностей *CF*, их расположения в памяти и специфики взаимного влияния друг на друга. Поэтому приведенные оценки количества конфигураций *LCF* лишь ориентировочно указывают на огромное число *LCF*. Поясним это на примере минимального числа (r = 3) ячеек памяти, участвующих в *LCF*, и минимального количества $n = r - 1 = 2$ одиночных *CF*, описывающих неисправности *LCF*. Учитывая конкретный вид *CF* (а именно, это неисправность *CFin*, *CFid* либо *CFst*) и их количество, равное 20 (см. предыдущий раздел), можно заключить, что две неисправности *CF*, в которых участвуют три ячейки, образуют 400 различных *LCF*. Количественная оценка 400 приведена только для одной из всевозможных неисправностей *LCF*, которые могут быть в случае конкретных r = 3 физических ячеек памяти с учетом их взаимного расположения.

Из рассмотренного примера о количестве неисправностей *LCF*, в которых участвуют только три ячейки, видно, что число таких неисправностей чрезвычайно велико. Даже в этом случае (небольшого количества ячеек, участвующих в *LCF*) число самих *LCF* и их многообразие не позволяют проводить анализ каждой из таких неисправностей на предмет ее обнаружения или необнаружения, что было возможным для одиночных *CF* [2, 4].

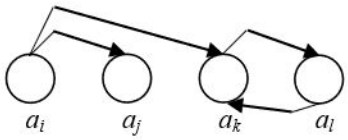
Необходимые и достаточные условия обнаружения *LCF*. Проблема обнаружения неисправностей *CF*, а также кратных неисправностей *CF* достаточно долго являлась основным стимулом разработки новых маршевых тестов [2]. Широко известные тесты *March C* и *March C–*, которые разрабатывались для обнаружения всевозможных одиночных неисправностей *CFid*, оказались весьма эффективными как по обнаруживающей способности неисправных состояний памяти в целом, так и по их невысокой временной сложности [2–6]. В рамках однократных маршевых тестов был разработан тест *March M*, в котором впервые реализовались условия обнаружения связных неисправностей и в первую очередь – двукратных связных неисправностей с различными конфигурациями [9]. Констатируя наличие необнаруживаемых неисправностей *LCF* однократными маршевыми тестами, были предложены более эффективные тесты, такие как *March LR* и *March LA* [10, 11]. В данных тестах обеспечивались условия обнаружения реально возможных неисправностей *LCF* в рамках рассматриваемых моделей связных неисправностей, а также технологических и конструктивных особенностей тестируемой памяти [10, 11]. Различные модификации тестов *March LR* и *March LA* эффективно используются для обнаружения динамических несвязных неисправностей [12], тестирования динамических запоминающих устройств [13] и реализации встроенного самотестирования памяти [14, 15].

В качестве условий обнаружения различных разновидностей *LCF* определялись необходимые множества маршевых элементов и их последовательность в тесте, которые обеспечивали обнаружение всех неисправностей рассматриваемого вида [2, 4]. Например, условием обнаружения всех одиночных *CFid* и всех связных двукратных *LCFid* является наличие в тесте следующих маршевых элементов: $\Uparrow\{ra, w\bar{a}, \ldots, wa, \ldots\}$, $\Uparrow\{r\bar{a}, wa, \ldots, w\bar{a}, \ldots\}$, $\Downarrow\{ra, w\bar{a}, \ldots, wa, \ldots\}$, $\Downarrow\{r\bar{a}, wa, \ldots, w\bar{a}, \ldots\}$ [4]. В свою очередь, условия наличия необходимых маршевых элементов в тесте формулировались на основании условий активизации конкретной неисправности и условий ее обнаружения. Подобные условия касаются ячеек памяти, непосредственно входящих в неисправность, и описывают последовательность действий, необходимых для обнаружения неисправности. В случае простейшей *CF* вида $\wedge\langle\uparrow,0\rangle$, где ячейка a_i является агрессором ($\uparrow$), а ячейка a_j – жертвой (0), необходимо последовательное во времени выполнение следующих условий. Во-первых, необходима первоначальная установка обеих ячеек в начальное состояние, а именно ячейки a_i в нулевое состояние, а ячейки a_j – в единичное. Затем выполняется условие активизации неисправности, которое заключается в выполнении изменения состояния из 0 в 1 в ячейке a_i, и, наконец, условие обнаружения неисправности как результата чтения нулевого содержимого ячейки a_j и сравнения его с эталонным (предварительно установленным в ячейке a_j) значением, равным 1.

Для установления условий обнаружения сложных, многочисленных и разнообразных конфигураций связных неисправностей взаимного влияния $LCF(a_i, a_j, a_k, \ldots, a_z)$, включающих r ячеек $a_i, a_j, a_k, \ldots, a_z$ с упорядоченным соотношением адресов $i < j < k < \ldots < z$, введем их фор-

мальное описание. Основой такого описания являются роли каждой из ячеек рассматриваемой *LCF* по отношению ко всем остальным ячейкам, входящим в описание неисправности. Аналогично, как и в простейшем случае *CF*, две ячейки a_i и a_j, $i \neq j$, могут влиять друг на друга, т. е. одна быть жертвой (*V*), а вторая – агрессором (*A*). В примерах моделей неисправностей *LCF*1, *LCF*2, *LCF*3, *LCF*4 и *LCF*5 роли каждой из ячеек указываются связями между ними. Ячейка с исходящей дугой – агрессор по отношению к ячейке, к которой направлена дуга. Для общего случая в рамках связных неисправностей, объединяющих две одиночные *CF*, в которые входят те же две ячейки a_i и a_j, введем третью роль (*both*, *B*) ячейки a_i, означающую две роли *A* и *V* по отношению к ячейке a_j. Соответственно, ячейка a_j по отношению к a_i также будет иметь две роли *V* и *A*, которые для нее будут обозначаться тем же символом *B*. Отсутствие связи (ролей) между ячейками a_i и a_j будем обозначать символом (-). Таким образом, в описании *LCF* будут представлены все ячейки, участвующие в ней, и все роли каждой из них.

В формальном описании $LCF(a_i, a_j, a_k, \ldots, a_z)$ последовательно представляется информация в виде набора символов -, *A*, *V* и *B* о всех ячейках $a_i, a_j, a_k, \ldots, a_z$. При описании ячеек также сохраняется их последовательность $a_i, a_j, a_k, \ldots, a_z$, т. е. на первой позиции всех описаний приводится информация о ячейке a_i, затем a_j и т. д. Эта последовательность определяется временем обращения к ячейкам, которое для однократных маршевых тестов соответствует последовательности их адресов. В качестве примера рассмотрим неисправность *LCF*6, представленную как в графическом виде, так и в виде нового формального описания (рис. 3).



$$LCF6(a_i, a_j, a_k, a_l) = \{\langle a_i, A, A, \text{-}\rangle; \langle V, a_j, \text{-}, \text{-}\rangle; \langle V, \text{-}, a_k, B\rangle; \langle \text{-}, \text{-}, B, a_l\rangle\}$$

Рис. 3. Связная неисправность взаимного влияния *LCF*6 и ее описание
Fig. 3. Linked coupling fault LCF6 with description

В приведенном примере *LCF*6 ячейки a_k и a_l, входящие в эту неисправность, образуют две простейшие *CF* и выступают по отношению друг к другу в обеих ролях, что обозначено символом *B* в описании *LCF*6 (рис. 3). Как уже пояснялось, символ *B* означает, что у ячейки есть две роли, а какую именно она реализует, определяется сценарием, который в данном случае задается последовательностью обращения к ячейкам в рамках маршевого теста. В зависимости от прямой $\Uparrow$ адресации, когда адреса соотносятся как $i < j < k < \ldots < z$, или обратной $\Downarrow$, когда $i > j > k > \ldots > z$, описание *LCF*6 представляется следующим образом:

$$\begin{aligned} &\Uparrow\{\langle a_i, A, A, \text{-}\rangle; \langle V, a_j, \text{-}, \text{-}\rangle; \langle V, \text{-}, a_k, A\rangle; \langle \text{-}, \text{-}, V, a_l\rangle\}, \\ &\Downarrow\{\langle a_i, A, A, \text{-}\rangle; \langle V, a_j, \text{-}, \text{-}\rangle; \langle V, \text{-}, a_k, V\rangle; \langle \text{-}, \text{-}, A, a_l\rangle\}. \end{aligned} \quad (2)$$

Как отмечалось ранее, упорядоченное соотношение адресов ячеек, участвующих в неисправности *LCF*, соответствует временной последовательности обращения к ним в рамках однократных маршевых тестов с фиксированной адресной последовательностью. Очевидно, что соотношение значений физических адресов ячеек, участвующих в *LCF*, и соотношение времен обращения к этим ячейкам могут быть различными. Примером подобной ситуации является неисправность, показанная на рис. 1. Аналогично неисправности *LCF*1 и *LCF*2, приведенные на рис. 1, могут быть описаны во введенной нотации следующим образом:

$$\begin{aligned} &LCF1 = LCF(a_i, a_j, a_k) = \{\langle a_i, \text{-}, A\rangle; \langle \text{-}, a_j, A\rangle; \langle V, V, a_k\rangle\}; \\ &LCF2 = LCF(a_i, a_k, a_j) = \{\langle a_i, A, \text{-}\rangle; \langle V, a_k, V\rangle; \langle \text{-}, A, a_j\rangle\}. \end{aligned} \quad (3)$$

Таким же образом описываются любые из возможных неисправностей *LCF*. Рассмотрим примеры данных описаний для ранее приведенных *LCF*:

$$LCF3(a_i, a_j, a_k, a_l) = \{\langle a_i, A, -, -\rangle; \langle V, a_j, A, -\rangle; \langle -, V, a_k, A\rangle; \langle -, -, V, a_l\rangle\};$$
$$LCF4(a_i, a_j, a_k, a_l) = \{\langle a_i, A, A, A\rangle; \langle V, a_j, -, -\rangle; \langle V, -, a_k, -\rangle; \langle V, -, -, a_l\rangle\};$$
$$LCF5(a_i, a_j, a_k) = \{\langle a_i, B, B\rangle; \langle B, a_j, B\rangle; \langle B, B, a_k\rangle\}.$$

Как и для *LCF*6 (2), неисправность *LCF*5 представляется в виде $\Uparrow\{\langle a_i, A, A\rangle; \langle V, a_j, A\rangle; \langle V, V, a_k\rangle\}$, $\Downarrow\{\langle a_i, V, V\rangle; \langle A, a_j, V\rangle; \langle A, A, a_k\rangle\}$.

Основным достоинством формальной модели описания конфигураций связных неисправностей является ее компактная запись, содержащая полную информацию о всех одиночных *CF*, образующих *LCF*. Например, из описания *LCF*1 следует, что данная связная неисправность образована из двух одиночных *CF*, а именно $CF(a_i, a_k) = \{\langle a_i, A\rangle; \langle V, a_k\rangle\}$ и $CF(a_j, a_k) = \{\langle a_j, A\rangle; \langle V, a_k\rangle\}$.

Упорядоченное обращение к ячейкам $a_i, a_j, a_k, \ldots, a_z$ неисправности $LCF(a_i, a_j, a_k, \ldots, a_z)$ при реализации маршевого элемента теста определяется временной последовательностью генерирования их адресов $i, j, k, \ldots, z$. Эта последовательность является важным элементом формального описания *LCF*, что, например, видно из описаний *LCF*1 и *LCF*2 (3) (см. рис. 1). Действительно, для последовательности адресов i, j, k используем описание неисправности *LCF*1, а для i, k, j – описание *LCF*2 (3). Позиция конкретной ячейки, предположим ячейки a_k, в ее описании так же, как и само описание в формальной модели *LCF*, соответствует временному порядку обращения к этой ячейке при реализации маршевого элемента теста. Например, в формальной модели $LCF1=\{\langle a_i, -, A\rangle; \langle -, a_j, A\rangle; \langle V, V, a_k\rangle\}$ описание ячейки a_k находится на третьей позиции, так как адрес k генерируется тестом после формирования адресов i и j. В самом описании $\langle V, V, a_k\rangle$ ячейки a_k ее обозначение также приведено на третьей позиции.

Таким образом, в основе новой формальной модели *LCF* лежат три роли, которые выполняют ячейки связной неисправности взаимного влияния, а именно роль агрессора (A), роль жертвы (V), а также роли жертвы и агрессора (B), выполняемые двумя ячейками одновременно по отношению друг к другу. В реальности предложенная модель описывает сценарий реализации ролей ячеек неисправности памяти, который определяется применяемым маршевым тестом и в первую очередь используемой в этом тесте адресной последовательностью обращения к ячейкам. Отметим, что для тестов с максимальной покрывающей неисправности *CF* способностью сценарий будет задаваться только адресной последовательностью, что и отображается в формальной модели *LCF*, как это видно на примере *LCF*1 и *LCF*2.

Для каждой ячейки $a_i, a_j, a_k, \ldots, a_z$ неисправности *LCF* в ее описании различают левую и правую часть, причем в левой части представлены те ячейки неисправности, которые активизируются до активизации текущей ячейки, а в правой части – после ее активизации. Под активизацией понимается обращение к ячейке памяти в соответствии с маршевым элементом теста и выполнение соответствующих операций записи и чтения.

Рассмотренная формальная модель *LCF* позволяет сформулировать условия обнаружения подобных неисправностей однократными маршевыми тестами.

Утверждение 2. *Необходимыми и достаточными условиями обнаружения связной неисправности взаимного влияния* $LCF(a_i, a_j, a_k, \ldots, a_z)$ *являются:*

1. Применение маршевого теста с произвольной адресной последовательностью, обнаруживающего всевозможные одиночные неисправности взаимного влияния CF.

2. Наличие в формальной модели неисправности $LCF(a_i, a_j, a_k, \ldots, a_z)$*, которая соответствует адресной последовательности теста, хотя бы одной ячейки* a_l, $l \in \{i, j, k, \ldots, z\}$*, в правой, либо левой, либо в обеих одновременно частях описания которой присутствует только одна роль V.*

Как показывалось ранее, произвольная неисправность $LCF(a_i, a_j, a_k, \ldots, a_z)$ представляет собой композицию одиночных неисправностей *CF*, все множество которых обнаруживается маршевыми тестами, например, такими, как *March LA* и *March LR* [10, 11]. Отметим, что всевоз-

можные одиночные неисправности *CF* гарантированно обнаруживаются указанными тестами только в случае, когда они не входят в связную неисправность *LCF*. В случае когда одиночные неисправности *CF* входят в *LCF*, указанные тесты либо обнаруживают эти неисправности по неверному состоянию ячейки агрессора, которая является жертвой в другой неисправности, либо гарантированно обеспечивают условие их активизации, изменив состояние ячейки-жертвы, с последующим обнаружением неверного состояния. Действительно, в случаях когда ячейка-агрессор не может выполнить условие активизации, наличие неисправного состояния памяти (наличие *LCF*) будет обнаружено по неверному состоянию такой ячейки. Это объясняется тем, что тесты, обнаруживающие одиночные *CF*, обеспечивают условия активизации всех подобных неисправностей путем задания необходимого начального состояния ячейки-агрессора (0, 1) или выполнения в ней одного из переходов (↑, ↓) (см. неисправности *CFin*, *CFid* и *CFst*) [2]. Таким образом, все ячейки-агрессоры неисправностей *CF*, входящие в *LCF*, либо будут являться причиной их обнаружения, став жертвами в рамках других *CF*, либо реализуют влияние на свои ячейки-жертвы. Изменения в ячейках-жертвах по отношению к первоначальным эталонным значениям в большинстве случаев будут обнаружены данными тестами [2, 4]. Это следует из свойств тестов, обнаруживающих все множество одиночных неисправностей взаимного влияния [2]. Исключением являются случаи маскирования, которые возникают при влиянии нескольких агрессоров на одну ячейку-жертву, когда, например, один агрессор изменяет эталонное значение жертвы на противоположное, а второй агрессор возвращает первоначальное значение ячейки-жертвы. Пример маскирования приведен на рис. 1 и описан формальной моделью *LCF*1 (3). Как видно, в данном случае у ячейки a_k две роли жертвы ($\langle V, V, a_k \rangle$) (3) и обе находятся в одной (левой) части ее описания. В случае когда ячейка-жертва выполняет только одну роль жертвы, т. е. у нее есть только один агрессор, перед обращением к этой ячейке маскирование невозможно. Более того, наличие двух ролей жертвы у ячейки одновременно, но в обеих частях формального описания означает, что содержимое данной ячейки будет прочитано только после однократного изменения ее состояния как результата проявления *CF*. Соответственно, *LCF* будет обнаружена. Подобный случай иллюстрируется примером неисправности *LCF*2, в формальном описании которой присутствуют две роли жертвы ($\langle V, a_k, V \rangle$) ячейки a_k, в данном случае одна роль V в правой, а вторая роль V в левой части формального описания.

Следует отметить, что механизм маскирования является весьма многообразным и характерен не только для запоминающих устройств, однако в рамках данной статьи рассматривается случай неисправностей *LCF* запоминающих устройств и возможное маскирование неисправностей *CF*, входящих в них [16].

Гарантированное обнаружение любой неисправности взаимного влияния *CF* безотносительно того, входит она либо не входит в связную неисправность *LCF*, обеспечивается анализом состояния ячейки-жертвы путем выполнения операции чтения ее состояния после реализации влияния на нее только одного агрессора (см. утверждение 2). Пример реализации подобного сценария описан формальной моделью *LCF*3, где анализ состояния ячейки-жертвы a_j ($\langle V, a_j, A, - \rangle$) выполняется после оказания влияния на нее только ячейки-агрессора a_i. Наличие хотя бы одной ячейки a_l, $l \in \{i, j, k, \ldots, z\}$, формальной модели $LCF(a_i, a_j, a_k, \ldots, a_z)$, в описании которой в правой, либо левой части, либо в обеих частях одновременно присутствует только одна роль жертвы, обеспечивает обнаружение данной $LCF(a_i, a_j, a_k, \ldots, a_z)$, что является необходимым и достаточным условием обнаружения *LCF* тестом, покрывающим одиночные *CF*.

Более сложное поведение запоминающего устройства происходит в случае наличия в нем неисправностей *LCF*, включающих *CF*, ячейки которых выполняют роль *B*, а также если проявление самих неисправностей *LCF* носит лавинообразный характер.

Сложные неисправности *LCF*. Под сложными неисправностями *LCF* будем понимать неисправности, включающие *CF*, ячейки которых выполняют роль *B*, или неисправности, проявление которых носит лавинообразный характер. Содержание сложных неисправностей *LCF* поясним на примерах *LCF*7, *LCF*8 и *LCF*9 (рис. 4).

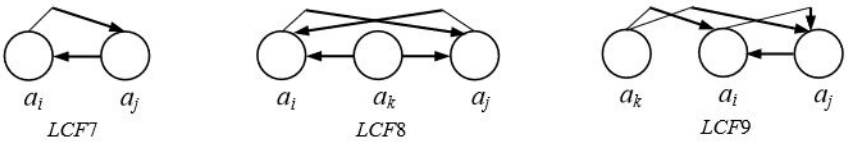


Рис. 4. Сложные связные неисправности взаимного влияния
Fig. 4. Complex linked coupling faults

Простейшим примером сложной *LCF* является *LCF*7, в которую входят две *CF*, образованные двумя ячейками a_i и a_j. Формальное описание неисправности $LCF7(a_i, a_j) = \{\langle a_i, B,\rangle; \langle B, a_j\rangle\}$ в зависимости от адресации $\Uparrow\{\langle a_i, A\rangle; \langle V, a_j\rangle\}$, $\Downarrow\{\langle a_i, V\rangle; \langle A, a_j\rangle\}$ показывает его соответствие условиям утверждения 2, что свидетельствует об ее обнаружении. Причем эта неисправность будет обнаруживаться в случае проявления одной из двух неисправностей $CF(a_i, a_j)$ или $CF(a_j, a_i)$ при стандартной реализации маршевого элемента теста и отсутствии лавинообразного характера проявления неисправностей.

Следует отметить, что чаще всего сложные неисправности *LCF* проявляются (активизируются) за счет активизации только одного из агрессоров *CF*, входящих в эту неисправность. Подобные ситуации объясняются физическими особенностями памяти. Как отмечалось в ряде литературных источников [9, 10, 12, 13], реалистичные неисправности *CF* образуются из ячеек, связанных общими шинами, на которые подаются электрические сигналы. Соответственно, при наличии сигналов, инициирующих изменение в ячейке-агрессоре, и происходит влияние на его жертву, которая, будучи потенциальным агрессором, не оказывает влияния на свою жертву.

Лавинообразный характер *LCF* возникает тогда, когда активизация одной *CF*, входящей в *LCF*, вызывает активизацию другой либо других *CF*, принадлежащих этой же *LCF*. Данный эффект заключается в том, что в активизированной ячейке-жертве одной неисправности *CF*, которая выполняет роль агрессора в другой *CF*, происходит изменение, активизирующее эту неисправность.

Структура сложных неисправностей *LCF*, а также сценарий их обнаружения, определяемый маршевым тестом, может приводить к различным эффектам их лавинообразного проявления. Специфика лавинообразного характера неисправностей во многом определяется и физическими характеристиками памяти. Возможна различная глубина лавинообразного характера проявления неисправности *LCF*, определяемая числом последовательно во времени активизированных *CF*. Эффект гонок сопряжен с процессом активизации нескольких *CF*, которые приводят к взаимоисключающим действиям на ячейку-жертву. Этот же эффект влияет на последовательность активизации других *CF*, входящих в *LCF*. Режим автогенерации, как правило затухающей, также является возможным эффектом, когда ячейка, входящая в *LCF*, многократно изменяет свое состояние на противоположное.

В случае неисправности *LCF*7 возможен лавинообразный характер ее проявления, когда при реализации элемента маршевого теста активизируется $CF(a_i, a_j)$, т. е. в ячейке-агрессоре a_i проводится операция записи, в результате которой изменяется состояние в ячейке-жертве a_j. В зависимости от вида другой $CF(a_j, a_i)$ как результат изменения состояния ячейки a_j, в данном случае агрессора, может произойти очередное изменение состояния a_i. При определенных условиях, например, если в *LCF*7 входят неисправности $CF(a_i, a_j) = \langle\uparrow, \updownarrow\rangle$ и $CF(a_j, a_i) = \langle\uparrow, \updownarrow\rangle$ и присутствует лавинообразный эффект, при начальных значениях $a_i = a_j = 0$ состояние ячейки a_i останется неизменным. Таким образом, при лавинообразном проявлении неисправности *LCF*7 отличными от ожидаемых будут состояния обеих ячеек, что обеспечивает обнаружение этой неисправности.

Лавинообразный характер проявления одиночных *CF*, входящих в *LCF*, может приводить к маскированию состояния ячеек запоминающего устройства. В конечном счете маскирование состояния ячейки может привести к маскированию неисправности *LCF*.

В табл. 1 приведены последовательные состояния ячеек неисправностей *LCF*, включающих только $CFin = \langle\uparrow, \updownarrow\rangle$, как результат применения маршевого элемента $\Uparrow(r0,w1)$. Каждая из неисправностей описывается как $LCF7 = LCF(a_i, a_j) = \{\langle a_i, B\rangle; \langle B\ a_j\rangle\}$, $LCF8 = LCF(a_i, a_k, a_j) = \{\langle a_i, V, B\rangle; \langle A, a_k, A\rangle; \langle B, V, a_j\rangle\}$, $LCF9 = LCF(a_k, a_i, a_j) = \{\langle\ a_k, A, A\rangle; \langle V, a_i, B\rangle; \langle V, B, a_j\rangle\}$.

Таблица 1
Диаграмма состояний ячеек, входящих в *LCF*7, *LCF*8 и *LCF*9

Table 1
*Cells state diagram included in LCF*7, *LCF*8 *and LCF*9

Неисправность *Malfunction*	*LCF*7	*LCF*7*	*LCF*8	*LCF*8*	*LCF*9	*LCF*9*
	a_i, a_j	a_i, a_j	a_i, a_k, a_j	a_i, a_k, a_j	a_k, a_i, a_j	a_k, a_i, a_j
Начальное состояние	0, 0	0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
$\Uparrow(r0,w1)$	0/1, 0/**1**	0/1/0, 0/**1**	0/1, 0, 0/1 1/0, 0/1, 1/0 0/1, 1, 0/1	0/1/0, 0, 0/1 0/1, 0/1, 1/0/**1**	0/1, 0/1, 0/1 1, **1**, 1	0/1, 0/1/0, 0/1 1, 0/1, 1/0 1, 1/**0**, 0/1
Эталонное состояние	1, 1	1, 1	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1

Для нулевого начального состояния ячеек *LCF* в табл. 1 приведены последовательности изменений их состояний как результат проявления неисправностей *LCF*7, *LCF*8 и *LCF*9 в случае стандартного процесса их активизации и обнаружения. Лавинообразный характер проявления указанных неисправностей приведен в столбцах табл. 1, помеченных как *LCF*7*, *LCF*8* и *LCF*9*. Последовательность адресов, генерируемых тестом, соответствует последовательности индексов обозначений ячеек. Например, для неисправности *LCF*8* согласно маршевому элементу $\Uparrow(r0,w1)$ первоначально выполняется обращение к ячейке a_i, затем к ячейке a_k и, наконец, к ячейке a_j. В содержательной части таблицы приводится последовательность изменений состояния ячеек как результат действий маршевого элемента и проявления рассматриваемой неисправности. Продолжая пример неисправности *LCF*8*, первое обращение будет выполнено к ячейке a_i, из которой будет прочитано значение 0 и записана 1 (0/1). Такое изменение ($\uparrow$) состояния ячейки a_i активизирует неисправность $CF(a_i, a_j) = \langle\uparrow, \updownarrow\rangle$, что приведет к инвертированию состояния ячейки a_j (0/1). Выполнение перехода из 0 в 1 ($\uparrow$) в ячейке a_j, в свою очередь, активизирует неисправность $CF(a_j, a_i) = \langle\uparrow, \updownarrow\rangle$, приводящую к лавинообразному инвертированию состояния ячейки a_i (1/0). Окончательно после обращения к ячейке a_i состояние ячеек примет вид a_i, a_k, a_j = 0/1/0, 0, 0/1 = 0, 0, 1. Символы **0** и **1**, приведенные в табл. 1, свидетельствуют об отличии состояний ячеек от ожидаемых, что означает обнаружение соответствующей *LCF* при выполнении операции чтения из этих ячеек и прекращение выполнения теста.

Наличие символа **1** для неисправностей *LCF*7 и *LCF*7* свидетельствует об их обнаружении, а отсутствие символов **0** и **1** в случае неисправности *LCF*8 говорит об ее необнаружении, что также следует из ее формального описания, в котором не выполняются условия утверждения 2. В то же время неисправность *LCF*9 является обнаруживаемой, хотя физически это та же неисправность *LCF*8 с измененным сценарием обращения к ячейкам, входящим в эту неисправность. Лавинообразный характер проявления неисправности *LCF*9* после обращения к ячейке a_k сказывается на состоянии ячеек, участвующих в неисправности, и имеет вид a_k, a_i, a_j = 0/1, 0/1/0, 0/1 = 1, 0, 1 (см. табл. 1). В данном случае возможно и другое поведение ячеек памяти, а именно a_k, a_i, a_j = 0/1, 0/1, 0/1/0 = 1, 1, 0, входящих в неисправности *LCF*9*. Это поведение определяется эффектом гонок.

Как показывалось ранее, роль ячейки *B*, входящей в *LCF*, включает две роли, каждая из которых реализуется в одной из двух фаз теста, а именно прямой либо обратной, что эквивалентно наличию одной роли жертвы в правой либо левой части описания *LCF*. Наличие двух ролей, а именно *V* и *B*, в правой либо левой части описания хотя бы одной ячейки, входящей в *LCF*, также гарантирует обнаружение этой неисправности, так как в одной из фаз теста роль *B* эквивалентна роли *A*. Соответственно, этот случай сводится к случаю наличия одной роли жертвы. Таким образом, можно заключить, что наличие в формальной модели $LCF(a_i, a_j, a_k, \ldots, a_z)$, которая соответствует сценарию адресной последовательности теста, хотя бы одной ячейки a_l, $l \in \{i, j, k, \ldots, z\}$, в описании которой в правой или левой части присутствует только одна из ролей жертвы *V* или *B* либо обе одновременно, является необходимым и достаточным условием обнаружения данной неисправности.

Возвращаясь к неисправности, приведенной на рис. 1 для случая последовательности адресов i, j, k теста, обнаруживающего одиночные связные неисправности, и ее описанию $\{\langle a_i, -, A\rangle; \langle -, a_j, A\rangle; \langle V, V, a_k\rangle\}$, можно констатировать необнаружение этой неисправности, так как не выполняются условия утверждения 2. В то же время применение других адресных последовательностей (других сценариев) может обеспечить ее обнаружение, как это видно из всевозможных описаний данной неисправности в зависимости от временной последовательности обращения к ячейкам, входящим в данную неисправность:

$$\begin{aligned}
LCF(a_i, a_j, a_k) &= \{\langle a_i, -, A\rangle; \langle -, a_j, A\rangle; \langle V, V, a_k\rangle\};\\
LCF(a_i, a_k, a_j) &= \{\langle a_i, A, -\rangle; \langle V, a_k, V\rangle; \langle -, A, a_j\rangle\};\\
LCF(a_j, a_i, a_k) &= \{\langle a_j, -, A\rangle; \langle -, a_i, A\rangle; \langle V, V, a_k\rangle\};\\
LCF(a_j, a_k, a_i) &= \{\langle a_j, A, -\rangle; \langle V, a_k, V\rangle; \langle -, A, a_i\rangle\};\\
LCF(a_k, a_i, a_j) &= \{\langle a_k, V, V\rangle; \langle A, a_i, -\rangle; \langle A, -, a_j\rangle\};\\
LCF(a_k, a_j, a_i) &= \{\langle a_k, V, V\rangle; \langle A, a_j, -\rangle; \langle A, -, a_i\rangle\}.
\end{aligned} \quad (4)$$

Очевидно, что для двух случаев из шести в выражениях (4) выполняются положения утверждения 2, а именно для $LCF(a_i, a_k, a_j)$ и $LCF(a_j, a_k, a_i)$, что обеспечивает необходимые и достаточные условия их обнаружения.

Многократные маршевые тесты для обнаружения *LCF*. Невысокая покрывающая способность связных неисправностей *LCF* классическими однократными маршевыми тестами объясняется последовательной процедурой доступа к ячейкам памяти и неизменным сценарием реализации теста [16]. Анализируя пример шести формальных описаний (4) неисправности *LCF*1, показанной на рис. 1, можно отметить, что только два из шести сценариев позволят обнаружить эту неисправность. Соответственно, неисправность будет обнаружена только путем изменения сценария и многократного применения теста. Сценарием в данном случае будет адресная последовательность маршевого теста, обнаруживающего все множество одиночных неисправностей взаимного влияния.

Идея многократного тестирования памяти рассматривалась в рамках исчерпывающего, псевдоисчерпывающего и неразрушающего тестирования памяти [2, 4, 16]. Во всех случаях необходимым являлось повторение маршевого теста для различных сценариев, определяемых степенями свободы, которые присущи маршевым тестам [17, 18]. Показано, что маршевые тесты памяти обнаруживают сложные неисправности путем их многократного применения для различных адресных последовательностей и (или) начального состояния запоминающих ячеек [3, 20]. Важным результатом многократного тестирования запоминающих устройств является обнаружение всевозможных их неисправных состояний. Однако достижение 100%-й полноты покрытия может потребовать большого числа (кратности) применения теста.

Для случая изменяемой адресной последовательности, определяющей сценарий для $LCF(a_i, a_j, a_k, \ldots, a_z)$, всегда существует временная последовательность обращения к ячейкам $a_i, a_j, a_k, \ldots, a_z$, для которой выполняются условия утверждения 2. Примером этому может быть неисправность *LCF*1 (4). Для нее существуют две из шести последовательностей адресов ячеек a_i, a_j и a_k, для которых эта неисправность будет обнаруживаемой. В общем случае для любого неисправного состояния памяти существует множество адресных последовательностей, для которого это состояние памяти будет обнаружено однократным тестом [3, 4, 16]. Соответственно, при случайной процедуре выбора адресной последовательности отношение p количества адресных последовательностей, для которых неисправность является обнаруживаемой, к общему числу последовательностей будет определять вероятность обнаружения этой неисправности однократным тестом.

Если предположить, что однократное применение маршевого теста дает возможность обнаруживать неисправности *LCF* с вероятностью $p_1 = p$, то f-кратное его использование при произвольных (случайных) адресных последовательностях позволяет достичь значения вероятности обнаружения p_f с помощью выражения [3, 21]

$$p_f = 1 - (1 - p_1)^f, \; f = 1, 2, 3, \ldots. \quad (5)$$

Очевидно, что с ростом f значение вероятности p_f стремится к единице. Соответственно, полнота покрытия таких неисправностей стремится к 100 %.

Для неисправности $LCF1$ с учетом (4) $p_1 = 2/6 = 0{,}333\ldots$.

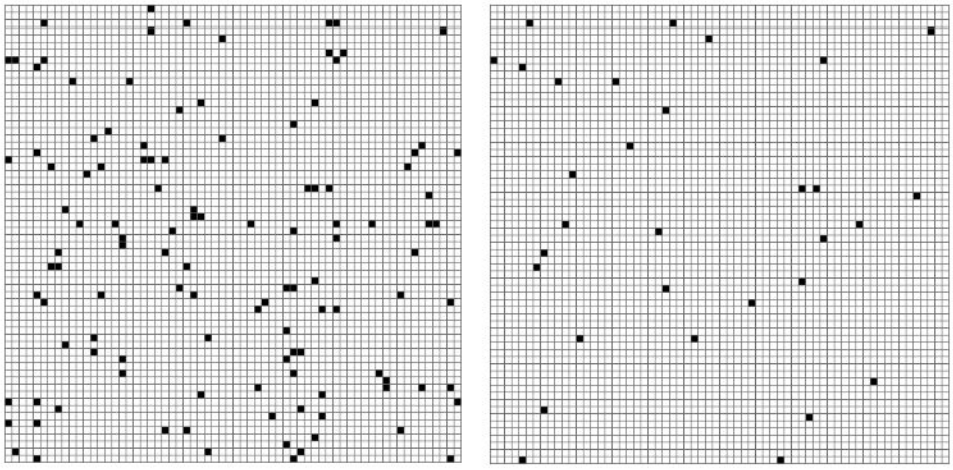
Экспериментальные исследования. Для оценки эффективности обнаружения неисправностей *LCF* и подтверждения полученных результатов было разработано программное средство, которое моделирует процесс тестирования памяти с использованием маршевых тестов и осуществляет визуализацию процесса тестирования, а также анализ обнаруживаемых тестом неисправностей запоминающих элементов памяти.

Входными данными программного средства являются:
– модель памяти, задаваемая количеством ячеек;
– модели неисправностей памяти и их количество;
– маршевый тест;
– кратность f маршевого теста;
– адресная последовательность для каждой итерации теста.

Выходными данными программного средства являются:
– графическое отображение алгоритма выбранного маршевого теста;
– визуальное отображение результатов тестирования;
– отчеты об обнаруженных неисправностях.

Программное средство разработано на языке C#. В качестве основы логики работы моделей неисправностей *LCF* был использован поведенческий паттерн «наблюдатель» (*observer*).

На рис. 5, *a* показан пример массива ячеек, выполняющих роль жертв в исходных неисправностях $LCF1$, включающих три ячейки, а на рис. 5, *b* – результат работы программного средства после двух итераций маршевого теста *March LR*. На первой итерации была использована адресная последовательность, представляющая собой отраженный код Грея, а на второй – последовательность АнтиГрея.



a) *b)*

Рис. 5. Пример визуализации процесса тестирования с применением разработанного программного средства: *a)* массив ячеек, выполняющих роль жертв; *b)* количество ячеек жертв после двух итераций теста

Fig. 5. An example of visualization of testing process using the developed software tool: a) an array of cells acting as victims; b) number of victim cells after two test iterations

Как видно из рис. 5, *b*, количество ячеек жертв необнаруженных *LCF* и, соответственно, самих неисправностей после реализации двукратного теста *March LR* существенно уменьшилось.

Для оценки вероятности обнаружения неисправности $LCF1$ многократными маршевыми тестами был проведен следующий эксперимент. В программном средстве была смоделирована память размером 8 бит и установлена связная неисправность взаимного влияния *LCF*, в которой адреса ячеек памяти расположены в соотношении $i < j < k$. В рассматриваемую *LCF* входят неисправности $CF(a_i, a_k) = \langle\uparrow, \updownarrow\rangle$ и $CF(a_j, a_k) = \langle\uparrow, \updownarrow\rangle$.

Для обнаружения неисправности *LCF* многократно выполнялся маршевый тест *March LA*. В каждой новой итерации теста использовалась случайно выбранная адресная последовательность из всех возможных ранее сгенерированных 40 320 последовательностей для памяти размером 8 бит. Было выполнено 10 000 экспериментов, каждый из которых заключался в обнаружении смоделированной ранее неисправности *f*-кратным маршевым тестом. В табл. 2 представлены значения вероятности обнаружения неисправности *LCF*1 в зависимости от кратности *f* маршевого теста в виде ее экспериментальной оценки (эксп.) и значения, вычисленные согласно выражению (5) (теор.).

Таблица 2
Вероятность p_f обнаружения неисправности *LCF*1 *f*-кратным тестом *March LR*

Table 2
*Probability p_f of LCF*1 *fault detection by f-run test March LR*

f		1	2	3	4	5	6	7	8	9	10	11	12
p_f	Теор.	0,3333	0,5555	0,7037	0,8024	0,8683	0,9122	0,9414	0,9609	0,9739	0,9826	0,9884	0,9922
	Эксп.	0,3300	0,5521	0,6985	0,7982	0,8669	0,9104	0,9402	0,9592	0,9745	0,9829	0,9886	0,9919

Результат эксперимента показал, что вероятность обнаружения неисправности *LCF*1 однократным маршевым тестом *March LR* составляет 0,33. С увеличением кратности теста *f* вероятность обнаружения неисправности стремится к 1, что соответствует выражению (5) и подтверждает правильность выбора авторами данной математической модели.

Заключение. Предложенная авторами формальная математическая модель описания связных неисправностей взаимного влияния позволила сформулировать необходимые и достаточные условия обнаружения таких неисправностей. Обобщенный характер формальной модели исключает необходимость анализа каждой конкретной конфигурации связной неисправности. Несомненным достоинством новой формальной модели является ее модифицируемость за счет изменения адресной последовательности маршевого теста. Сценарий поведения ячеек при тестировании памяти определяется адресной последовательностью однократного маршевого теста. Перспективным представляется применение предложенной модели связных неисправностей взаимного влияния для их идентификации и диагностической локализации. Очевидна применимость данной модели и для других видов связных неисправностей запоминающих устройств, а также для широкого класса кодочувствительных неисправностей.

Вклад авторов. *В. Н. Ярмолик* предложил формальную модель описания связных неисправностей взаимного влияния и сформулировал условия их обнаружения, *Д. В. Деменковец* и *В. В. Петровская* приняли участие в экспериментальных исследованиях и анализе полученных результатов, *А. А. Иванюк* обобщил и проанализировал полученные результаты.

Список использованных источников

1. Lee, K. Coverage re-evaluation of memory test algorithms with physical memory characteristics / K. Lee, J. Kim, S. Baeg // IEEE Access. – 2021. – Vol. 9. – P. 124632–124639.

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

3. Ярмолик, С. В Маршевые тесты для тестирования ОЗУ / С. В. Ярмолик, А. П. Занкович, А. А. Иванюк. – Saarbrücken, Germany : LAP Lambert Academic Publishing, 2012. – 302 с.

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

5. 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.

6. Caşcaval, P. March test algorithm for unlinked static reduced three-cell coupling faults in random-access memories / P. Caşcaval, D. Caşcaval // Microelectronics J. – 2019. – Vol. 93, iss. C. – Art. 104619.

7. 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, 21–25 Oct. 1995. – Washington, DC, USA, 1995. – P. 23–32.

8. Cockburn, B. E. Deterministic tests for detecting single V-coupling faults in RAMs / B. E. Cockburn // J. of Electronic Testing: Theory and Applications. – 1994. – Vol. 5, no. 1. – P. 91–113.

9. Mikitjuk, V. G. RAM testing algorithm for detection multiple linked faults / V. G. Mikitujk, V. N. Yarmolik // Proc. of the 1996 European Design and Test Conf. (ED&TC'96), Paris, France, 11–14 Mar. 1996. – Paris, France, 1996. – P. 435–440.

10. March LR: a test for realistic linked faults / A. J. Goor [et al.] // Proc. of the 14th VLSI Test Symp., Princeton, NJ, USA, 28 Apr. – 01 May 1996. – Princeton, NJ, USA, 1996. – P. 272–280.

11. March LA: a test for linked memory faults / A. J. Goor [et al.] // Proc. of the 1997 European Design and Test Conf. (ED&TC'97), Paris, France, 17–20 Mar. 1997. – Paris, France, 1997. – P. 627.

12. Modified March MSS for unlinked dynamic faults detection / L. W. Ying [et al.] // Proc. of the IEEE 20th Student Conf. on Research and Development (SCOReD), Bangi, Malaysia, 08–09 Nov. 2022. – Bangi, Malaysia, 2022. – P. 68–72.

13. Chou, C.-W. Testing inter-word coupling faults of wide I/O DRAMs / C.-W. Chou, Y.-X. Chen, J.-F. Li // Proc. of the 2015 IEEE 24th Asian Test Symp., Mumbai, India, 22–25 Nov. 2015. – Mumbai, India, 2015. – P. 22–25.

14. Manasa, R. Implementation of BIST technology using March-LR algorithm / R. Manasa, R. Verma, D. Koppad // Proc. of the 2019 4th Intern. Conf. on Recent Trends on Electronics Information Communication & Technology (RTEICT), Bangalore, India, 17–18 May 2019. – Bangalore, India, 2019. – P. 1208–1212.

15. Implementation of minimized March SR algorithm in a memory BIST controller / A. Z. Jidin [et al.] // J. of Engineering and Technology. – 2022. – Vol. 13, no. 2. – P. 1–14.

16. Ярмолик, В. Н. Контроль и диагностика цифровых устройств ЭВМ / В. Н. Ярмолик. – Минск : Наука и техника, 1988. – 240 с.

17. Sokol, B. Address sequence for march tests to detect pattern sensitive faults / B. Sokol, S. V. Yarmolik // Proc. of 3rd IEEE Intern. Workshop on Electronic Design Test and Applications (DELTA'06), Kuala Lumpur, Malaysia, 17–19 Jan. 2006. – Kuala Lumpur, Malaysia, 2006. – P. 354–357.

18. Sokol, B. Impact of the address changing on the detection of pattern sensitive faults / B. Sokol, I. Mrozek, V. N. Yarmolik // Information Processing and Security Systems. – London : Springer Science + Business Media, Inc., 2005. – P. 217–226.

19. Yarmolik, S. V. Address sequences and backgrounds with different Hamming distance for multiple run March tests / S. V. Yarmolik // IEEE Intern. J. of Applied Mathematics and Computer Science. – 2008. – Vol. 18, no. 3. – P. 329–339.

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

21. Построение и применение маршевых тестов для обнаружения кодочувствительных неисправностей запоминающих устройств / В. Н. Ярмолик [и др.] // Информатика. – 2021. – № 1(18). – С. 25–42.

References

1. Lee K., Kim J., Baeg S. Coverage re-evaluation of memory test algorithms with physical memory characteristics. *IEEE Access*, 2021, vol. 9, pp. 124632–124639.

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

3. Yarmolik S. V., Zankovich A. P., Ivaniuk A. A. Marshevue testu dlya testirovaniya OZU. *March Tests for Memory Testing*. Saarbrücken, Germany, LAP Lambert Academic Publishing, 2012, 302 p. (In Russ.).

4. Yarmolik V. N., Murashko I. A., Kummert A., Ivaniuk A. A. Nerazrushayuschee testirovanie zapominayuschih ustroistv. *Non-Destructive Storage Testing*. Minsk, Bestprint, 2005, 230 p. (In Russ.).

5. 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.

6. Caşcaval P., Caşcaval D. March test algorithm for unlinked static reduced three-cell coupling faults in random-access memories. *Microelectronics Journal*, 2019, vol. 93, iss. C, art. 104619.

7. 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, 21–25 October 1995.* Washington, DC, USA, 1995, pp. 23–32.

8. Cockburn B. E. Deterministic tests for detecting single V-coupling faults in RAMs. *Journal of Electronic Testing: Theory and Applications*, 1994, vol. 5, no. 1, pp. 91–113.

9. Mikitjuk V. G., Yarmolik V. N. RAM testing algorithm for detection multiple linked faults. *Proceedings of the 1996 European Design and Test Conference (ED&TC'96), Paris, France, 11–14 March 1996.* Paris, France, 1996, pp. 435–440.

10. Goor A. J., Gaydadjiev G. N., Yarmolik V. N., Mikitujk V. G. March LR: a test for realistic linked faults. *Proceedings of the 14th VLSI Test Symposium, Princeton, NJ, USA, 28 April – 01 May 1996.* Princeton, NJ, USA, 1996, pp. 272–280.

11. Goor A. J., Gaydadjiev G. N., Yarmolik V. N., Mikitujk V. G. March LA: a test for linked memory faults. *Proceedings of the 1997 European Design and Test Conference (ED&TC'97), Paris, France, 17–20 March 1997.* Paris, France, 1997, p. 627.

12. Ying L. W., Hussin R., Ahmad N., Fook L. W., Jidin A. Z. Modified March MSS for unlinked dynamic faults detection. *Proceedings of the IEEE 20th Student Conference on Research and Development (SCOReD), Bangi, Malaysia, 08–09 November 2022*. Bangi, Malaysia, 2022, pp. 68–72.

13. Chou C.-W., Chen Y.-X., Li J.-F. Testing inter-word coupling faults of wide I/O DRAMs. *Proceedings of the 2015 IEEE 24th Asian Test Symposium, Mumbai, India, 22–25 November 2015.* Mumbai, India, 2015, pp. 22–25.

14. Manasa R., Verma R., Koppad D. Implementation of BIST technology using March-LR algorithm. *Proceedings of the 2019 4th International Conference on Recent Trends on Electronics Information Communication & Technology (RTEICT), Bangalore, India, 17–18 May 2019.* Bangalore, India, 2019, pp. 1208–1212.

15. Jidin A. Z., Hussin R., Mispan M. S., Lee W. F., Zakaria N. A. Implementation of minimized March SR algorithm in a memory BIST controller. *Journal of Engineering and Technology*, 2022, vol. 13, no. 2, pp. 1−14.

16. Yarmolik V. N. Kontrol' i diagnostika cifrovyh ustrojstv jelektronno-vychislitel'nyh mashin. *Monitoring and Diagnostics of Digital Devices of Electronic Computers*. Minsk, Nauka i tehnika, 1988, 240 p. (In Russ.).

17. Sokol B., Yarmolik S. V. Address sequence for march tests to detect pattern sensitive faults. *Proceedings of 3rd IEEE International Workshop on Electronic Design Test and Applications (DELTA'06), Kuala Lumpur, Malaysia, 17–19 January 2006*. Kuala Lumpur, Malaysia, 2006, pp. 354–357.

18. Sokol B., Mrozek I., Yarmolik V. N. Impact of the address changing on the detection of pattern sensitive faults. *Information Processing and Security Systems*. London, Springer Science + Business Media, Inc., 2005, pp. 217–226.

19. Yarmolik S. V. Address sequences and backgrounds with different Hamming distance for multiple run March tests. *IEEE International Journal of Applied Mathematics and Computer Science*, 2008, vol. 18, no. 3, pp. 329−339.

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

21. Yarmolik V. N., Levantsevich V. A., Demenkovets D. V., Mrozek I. *Construction and application of march tests for pattern sensitive memory faults detection.* Informatika [*Informatics*], 2021, vol. 18, no. 1, pp. 25–42 (In Russ.).

Информация об авторах

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

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

Петровская Вита Владленовна, магистр технических наук, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: vita.petrovskaya@gmail.com

Иванюк Александр Александрович, доктор технических наук, доцент, профессор кафедры информатика, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: ivaniuk@bsuir.by

Information about the authors

Vyacheslav N. Yarmolik, D. Sc. (Eng.), Prof., Belarusian State University of Informatics and Radioelectronics.
E-mail: yarmolik10ru@yahoo.com

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

Vita V. Petrovskaya, M. Sc. (Eng.), Belarusian State University of Informatics and Radioelectronics.
E-mail: vita.petrovskaya@gmail.com

Alexander A. Ivaniuk, D. Sc. (Eng.), Assoc. Prof., Prof. of Computer Science Department, Belarusian State University of Informatics and Radioelectronics.
E-mail: ivaniuk@bsuir.by