applied sciences

MDPI

Article

Linked Coupling Faults Detection by Multirun March Tests

Ireneusz Mrozek [1,*] and Vyacheslav N. Yarmolik [2,*]

[1] Faculty of Computer Science, Bialystok University of Technology, 15-351 Białystok, Poland
[2] Faculty of Computer Science, Belarusian State University of Informatics and Radioelectronics, 220013 Minsk, Belarus
[*] Correspondence: i.mrozek@pb.edu.pl (I.M.); yarmolik10ru@yahoo.com (V.N.Y.)

Abstract: This paper addresses the problem of describing the complex linked coupling faults of memory devices and formulating the necessary and sufficient conditions for their detection. The main contribution of the proposed approach is based on using a new formal model of such faults, the critical element of which is the introduction of roles and scenarios performed by the cells involved in the fault. Three roles are defined such that the cells of the complex linked coupling faults perform, namely, the roles of the aggressor (A), the victim (V), and both (B), performed by two cells simultaneously in relation to each other. The memory march test and applied address sequence and background determine the scenario for implementing the roles of memory faulty cells. The necessary and sufficient conditions for detecting linked coupling faults are given based on a new formal model. Formally, the undetectable linked coupling faults are defined, and the conditions for their detection are formulated using multirun memory march tests. The experimental investigation confirmed the validity of the proposed formulated statements. Based on the example of a linked coupling fault, this study demonstrates the fulfillment of the necessary and sufficient conditions for its detection using a single march test. As demonstrated in this article, employing the approach proposed by the authors, a two-pass march C test, for instance, enables the attainment of 55.42% fault coverage for linked coupling faults, inclusive of undetectable faults identified by the single-pass march test.

Keywords: memory tests; march tests; coupling faults; linked faults

check for updates

Citation: Mrozek, I.; Yarmolik, V.N. Linked Coupling Faults Detection by Multirun March Tests. *Appl. Sci.* **2024**, *14*, 2501. https://doi.org/10.3390/app14062501

Academic Editor: Juan A. Gómez-Pulido

Received: 3 February 2024
Revised: 28 February 2024
Accepted: 13 March 2024
Published: 15 March 2024

1. Introduction

Memory devices occupy a dominant place among the hardware components of modern computer systems. Demand for high-speed, highly integrated, low-power memory is growing unprecedentedly because cloud computing, artificial intelligence, and fifth-generation (5G) communications, among others, are positioned as the main directions of the fourth Industrial Revolution. To ensure the characteristics of modern memory of computer systems that meet the requirements of new technological advances, the necessity and importance of testing memory devices have increased sufficiently [1,2]. The main task of memory testing is to detect faulty states described by various models of their failures [3–5]. Among the large variety of memory faults, coupling faults, *CF*s, involve two memory cells, a_i and a_j. The types of such faults are discussed below [3,6–9].

First, in an inverse coupling fault , *CFin*, two cells, a_i and a_j, with addresses $i \neq j$ participate in the model $CFin(a_i, a_j)$ in the fault; here, one cell a_i influences the second a_j and is called the aggressor (A), and the second a_j, which is affected, is the victim (V). In this fault, a logical transition from 1 to 0 (↓) or from 0 to 1 (↑) in the aggressor a_i leads to inverting (↕) the logical value in the victim cell a_j, which is described by two models of inverse faults $CFin(a_i, a_j)$: $<\uparrow, \updownarrow>$ and $<\downarrow, \updownarrow>$ [3]. To represent the relation of address i of the aggressor and j of the victim, in the memory address space, the symbols $\wedge$ and $\vee$ are used, which define four inverse faults $CFin(a_i, a_j) = \{\wedge<\uparrow,\updownarrow>, \wedge<\downarrow,\updownarrow>, \vee<\uparrow,\updownarrow>, \vee<\downarrow,\updownarrow>\}$. According to the classical definitions, the symbol $\wedge$ indicates that the address of the aggressor is less than the address of the victim $i < j$. In contrast, for the symbol $\vee$, it is greater, $i > j$. One-run march tests implement sequential access to memory cells by enumerating all addresses in an increasing or decreasing sequence. Therefore, the relation

$i < j$ compares the values of the cells a_i and a_j addresses and the time of accessing them. When implementing an element of a march test with an increasing sequence of addresses ($\Uparrow$) and the relation of addresses is $i < j$, the address for cell a_i is generated initially, and only after some time is the address of cell a_j generated.

Second, in an idempotent coupling fault, *CFid*, during a logical transition from 1 to 0 or from 0 to 1 in the aggressor cell a_i, a certain logical value 0 or 1 is forced to be set in the victim cell a_j [3]. Eight faults of direct action are recognized: $CFid(a_i, a_j)$ = {∧<↑,0>, ∧<↑,1>, ∨<↑,0>, ∨<↑,1>, ∧<↓,0>, ∧<↓,1>, ∨<↓,0>, ∨<↓,1>} [3].

Third, in a static fault, *CFst*, the transition of a victim cell to any state $a_j \in$ {0,1} occurs at a certain value $a_i \in$ {0,1} of the aggressor cell. Eight faults are possible: $CFst(a_i, a_j)$ = {∧<0,0>, ∧<0,1>, ∨<0,0>, ∨<0,1>, ∧<1,0>, ∧<1,1>, ∨<1,0>, ∨<1,1>} [3].

There are single coupling faults, which can be any of the above fault models, and multiple coupling faults, consisting of a subset of single coupling faults [3]. In the set of multiple faults, unlinked multiple coupling faults are distinguished. Thus, a particular memory cell is involved in only one coupling fault included in the considered multiple faults. In general, n unlinked coupling faults involve an even number $r = 2n$ of memory cells, half of which are aggressors and half are victims. Thus, each cell of a single *CF* has only one role: the aggressor (A) or victim (V).

Due to the tremendous capacity of modern memory devices, tests are currently widely used. They are still being developed with a time complexity that depends linearly on the memory capacity. Such tests are called march tests [1,3,5–10] and are used for bit-oriented memory, where each cell stores one bit. A march test consists of a finite number of march elements [3]. Each march element contains a symbol that determines the order of formation of the address sequence of the cells of the memory, determining the order in which the cells are accessed. The symbol $\Uparrow$ indicates a sequential enumeration of memory addresses in increasing order (forward), and the symbol $\Downarrow$ indicates decreasing order (in reverse order). March elements contain a sequence of read (r) and write (w) operations, enclosed in parentheses and separated by semicolons. The transition to the next cell, according to the address sequence, is conducted only after performing all the operations with the current cell according to the march element [1,3,6,7]. For example, the simplest march element used in many tests has the form $\{\Uparrow(r0, w1)\}$. In accordance with this element and the address sequence, 0 is read from each cell, and 1 is written. The read operation compares the read value with the expected value (0). A flowchart of this march element is presented in Figure 1.

The detection of faults in memory, as a result of the march test, is based on receiving incorrect values read from its cells that are different from the reference ones. Thus, a faulty memory state is fixed, and a diagnostic procedure is performed if necessary [3,8,9]. Detecting single and multiple unlinked coupling faults within the march tests framework is considered solved [3]. Effective tests have proven to detect them in practice [3,6,7,11]. March tests that detect coupling faults assume a one-run memory test with a standard initial state of storage cells, usually zero, and a standard counter address sequence [3]. The problem of detecting linked coupling faults, *LCF*s, using one-run march tests remains practically open, requiring further research due to the variety and complexity of the manifestation mechanisms of such faults [6–9].

This article proposes a formal model of linked coupling faults, describing the roles of all cells involved in the fault and the scenario of their manifestation regarding the march test and its address sequence. The necessary and sufficient conditions for detecting such faults within the framework of one-run tests are determined, and the efficiency of their detection using multiple tests is evaluated.

In the era of rapidly advancing computing technologies and the ever-increasing importance of data integrity, our research plays an important role in ensuring the reliability of memory devices. The formal model and conditions we propose for detecting complex linked coupling faults have significant implications in various computing domains, including big data analytics, where memory errors can have cascading effects on data processing and reliability. Moreover, the ability to detect and address such faults is of paramount importance in networked systems, cloud computing infrastructure, and data storage, where data integrity and availability are critical. Our results not only contribute to the field of memory testing but also contribute to the overall robustness and reliability of computing systems.

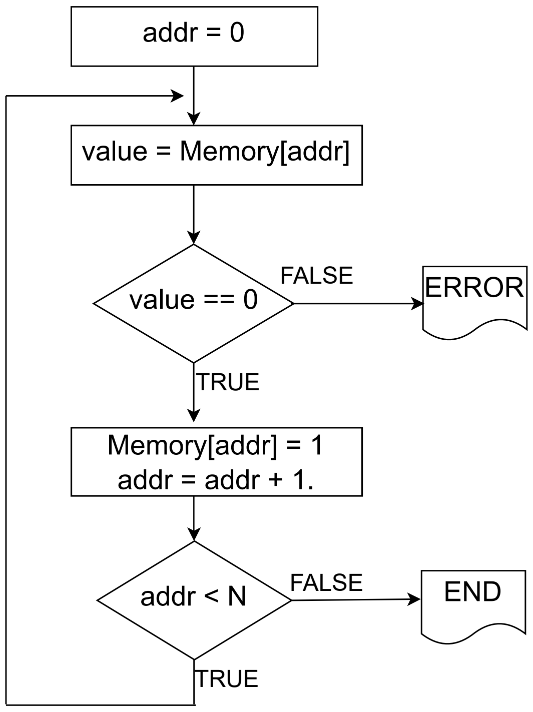


Figure 1. Flowchart of simple march element.

One well-known march test is the `MATS+` written as:

$$\{\underset{M0}{\Updownarrow(w0);}\quad \underset{M1}{\Uparrow(r0,w1);}\quad \underset{M2}{\Downarrow(r1,w0)}\}. \tag{1}$$

It has three march elements $M0 :\Updownarrow(w0)$, $M1 :\Uparrow(r0,w1)$, and $M2 :\Downarrow(r1,w0)$. Algorithm 1 presents the interpretation of this notation in pseudocode.

Algorithm 1 MATS+ march test algorithm

```
for $cell = 0$ to $N - 1$ do
    Memory[cell] = 0;
end for
for $cell = 0$ to $N - 1$ do
    value = Memory[cell];
    if $value <> 0$ then
        goto error;
    end if
    Memory[cell] = 1;
end for
for $cell = N - 1$ downto 0 do
    value = Memory[cell];
    if $value <> 1$ then
        goto error;
    end if
    Memory[cell] = 0;
end for
```

However, these studies are limited to the detection of linked coupling memory faults. The proposed research aims to establish the necessary and sufficient conditions for detecting such faults in modern memory devices. Currently, there are no one-run march test approaches for detecting undetectable linked coupling memory faults. Most of the recent research in this area focuses on developing and enhancing march tests to detect new faults introduced by Very Deep SubMicron (VDSM) memory devices, such as CF_{ir}, CF_{drd}, CF_{wd}, and CF_{ir}. The problem of detecting linked coupling memory faults has been tackled for neighborhood pattern-sensitive faults (*NPSF*s). Several test algorithms have been developed to detect *NPSF*s, but they come with very high test complexities. As mentioned in [12], a $68N$ arch test algorithm can be used for *ANPSF*s and *PNPSF*s, while it requires a $96N$ march test to detect all *ANPSF*s, *PNPSF*s, and *SNPSF*s. Detecting very complex *NPSF*s is not only time-consuming but also does not guarantee the detection of undetectable faults, including *CF*s. The approach proposed in this paper enables the identification of previously undetectable linked coupling faults. Additionally, the proposed mathematical model for describing linked coupling memory faults may be applicable to other memory faults involving more than two cells. Nonetheless, this necessitates further rigorous and time-consuming research.

2. Linked Coupling Faults

In the structure of classical models of memory faults, linked faults occupy one of the most prominent places due to the difficulty of their detection using existing march memory tests [3,6,7]. Linked faults are understood as faults of various types, including typical cells involved in several single faults. In the case of linked coupling faults consisting of n single *CF*s, $r < 2n$ memory cells are involved. An example of an *LCF*1 consisting of two $(n = 2)$ single coupling faults is illustrated in Figure 2 [3]. In this figure, memory cells with addresses $i < j < k$ are presented sequentially from left to right in the time dependence of their access during the execution of the direct phase of the march test. The designations *LCF*1 and *LCF*2 refer to the same *LCF* where the manifestation mechanism (scenario) differs and depends on the formation time of the address of the participating cells.

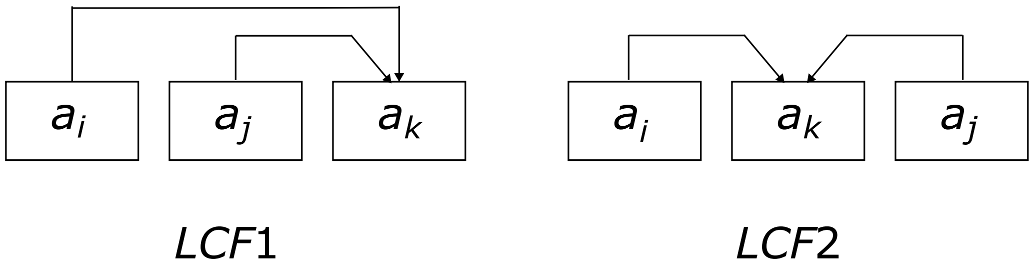


Figure 2. Linked coupling faults for $n = 2$.

As depicted in Figure 2, *LCF*1 and *LCF*2 involve $r = 3$ cells and include two single *CF*s, each consisting of two cells. The above example of *LCF*1 is presented in numerous literary sources and is positioned as an example of an undetectable fault in the framework of classical march tests [3]. If we assume that the addresses of the cells a_i, a_j, and a_k are formed in the sequence i, j, and k; that the cells a_i and a_k participate in the fault $\wedge$<↑,↕>; and that the cells a_j and a_k form a similar fault $\wedge$<↑,↕>, we have an undetectable fault *LCF*1. The fact that this fault was not detected refers to one-run march tests using an address sequence satisfying the condition of generating address i first, then j, and finally k. Then, the successive manifestation of two single faults included in the linked fault *LCF*1 (Figure 2) leads to a sequential double inversion of the state of the cell a_k. Thus, the state of cell a_k, when accessed, is always correct, and the *LCF*1 is undetectable. The considered example of the *LCF*1 linked fault is given as an argument regarding the nondetection of linked faults in general. However, in particular, the linked fault in Figure 2, consisting of two faults $\wedge$<↑,0> or $\wedge$<↑,1>, is detectable. Moreover, when using a one-run march test in which the addresses are generated in the time sequence i, then k, and finally j, the *LCF*1 designated as *LCF*2 is detectable regardless of which two CFs form *LCF*2.

Thus, the linked coupling fault, *LCF*, depends on the number, n, of single linked faults and their varieties and the number, r, of cells participating in the *LCF*. For the general case, the number of roles and their types, A and V, for each *LCF* cell is also crucial. For example, for the *LCF*1 where cells a_i and a_j play the aggressor role A, this is the only role for both cells, whereas cell a_k has two victim roles, V. The concept of a role defines the interaction between two cells, so the cell a_k has two roles, V, resulting from the interaction with a_i and a_j within the single *CF*s that comprise the *LCF*1. Abstracting from realistic and unrealistic *LCF*s, the sets of which are primarily determined by the design and technological features of the memory, we formulate several statements for the general case of an *LCF*. First, assuming that r cells participate in the *LCF*, we estimate the minimum, min(n), and maximum, max(n), number, n, of single coupling faults organized in the *LCF*. Thus, the *LCF* consists of single *CF*s, and each of the r cells is included in at least one *CF*. Statement 1 presents the critical factor for obtaining the boundary values of min(n) and max(n).

Statement 1. *If a $CF(a_i, a_j)$ is included in an* LCF *consisting of* r *cells, then the number of roles for one of the cells a_i or a_j forming the* CF *must be at least two, and the total number for both cells of the* CF *lies in the range from 3 to $4r - 6$.*

The validity of this statement follows from the fact that if each of the cells a_i and a_j forming a single *CF* has only one role, then this *CF* is not included in the *LCF*. Such a fault is not associated with other *CF*s because the cells included in this fault do not have roles that connect it with other memory cells or other *CF*s. Thus, the presence of at least one of the cells a_i and a_j of the *CF* of more than one role provides a connection between other single *CF*s, including in the *LCF*. Thus, the minimum number of roles for a *CF* in an *LCF* is three. The upper bound of the number of roles for the fault cells of a *CF* participating in the *LCF* (equal to $4r - 6$) is achieved when each of the two fault cells plays the roles of both the aggressor and victim in relation to the other $r - 2$ cells. Two more roles are possible between the *CF* cells a_i and a_j. Then, the maximum number of roles of *CF* cells included in the *LCF* is defined as $2(r-2) + 2(r-2) + 2 = 4r - 6$.

The minimum number (min(n)) of single *CF*s that form an *LCF* is $r - 1$. This result follows from Statement 1, corresponding to the minimum number of roles for each *CF*, two describing the *CF* itself, and the third defining the relationship with another *CF*. An example would be when, in each subsequent *CF*, the aggressor cell is also the victim, for which the victim cell of the current fault acts as the aggressor. Thus, min(n)= $r - 1$, which is also achievable for the second extreme case, where one of the r *LCF* cells is the aggressor for the remaining $r - 1$ cells. We also have $r - 1$ single *CF*s, where one of the cells performs more than two roles, in this case, $r - 1$ roles. Examples of such faults *LCF*3 and *LCF*4 for $r = 3$ are depicted in Figure 2. The maximum number (max(n)) of single *CF*s generating an *LCF* is achieved when any possible pair of cells from r cells forms two *CF*s, respectively, and the first cell plays the role of A, the second V, and vice versa. Thus, $\max(n) = 2 \cdot \binom{r}{2} = r(r-1)$. An example of an *LCF* consisting of the maximum number of single *CF*s for $r = 3$ is presented in Figure 3 as the *LCF*5.

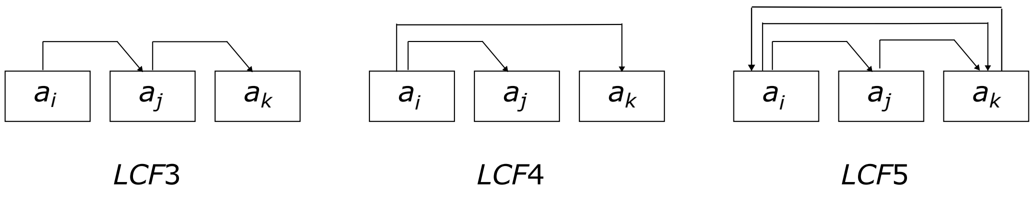


Figure 3. Linked coupling faults for $r = 3$.

Thus, for an *LCF* consisting of r cells, the number n of single *CF*s forming this fault is in the following range:

$$min(n) = r - 1 \leq n \leq max(n) = r(r-1). \tag{2}$$

The resulting limit values for the number of *CF*s included in the *LCF* correspond to similar values for the number of edges in connected labeled directed graphs consisting of r vertices. Using graph theory, the number of types of *LCF*s, consisting of r memory cells, can be estimated as the total number of simple labeled directed graphs with r vertices. This number $2r(r-1)$ grows exponentially with the number of cells r and can be used as an upper bound for the number of *LCF* varieties.

The min(n) and max(n) values of the number of single *CF*s that form an *LCF* (2), including r memory cells, were obtained without considering the type of CFs, their location in the memory, and the specifics of their mutual influence on each other. Therefore, the above estimates of the number of *LCF* configurations only tentatively indicate numerous *LCF*s. We explain this by the example of the minimum number $(r = 3)$ of memory cells participating in the *LCF* and the minimum number $n = r - 1 = 2$ of single *CF*s describing *LCF*s. Considering the specific type of *CF*, whether it is a *CFin*, *CFid*, or *CFst*, and the number is 20 (see the previous section), we can conclude that two *CF*s involving three cells form 400 different *LCF*s. The estimation of 400 is given only for one of the various *LCF* failures that can occur in the case of $r = 3$ specific physical memory cells, considering their relative position. As indicated in the above example of the number of *LCF*s involving only three cells, the number of such faults is exceptionally high. Even in this case (a small number of cells participating in the *LCF*), the number of *LCF*s and their diversity do not allow for analyzing each of these faults for detection or nondetection, which was possible for single *CF*s [3].

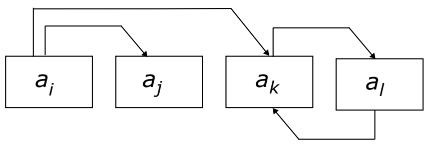
3. Necessary and Sufficient Conditions for *LCF* Detection

The problem of detecting single and multiple *CF*s has long been the primary stimulus for developing new march tests [3]. The widely known `March C` and `March C-` tests, developed to detect all kinds of single *CFid* faults, are very effective in detecting faulty memory states and their low time complexity [3,7]. Within the framework of one-run march tests, the `March M` test was developed, in which the conditions for detecting coupling faults were implemented for the first time, including double coupling faults of various configurations [13]. More efficient tests were proposed by ascertaining the presence of undetectable *LCF*s using one-run march tests, such as the `March LR` and `March LA` [13,14]. These tests provided conditions for detecting realistic *LCF*s within the considered models of linked faults and determining the technological and design features of the memory under testing [13,14]. Various modifications of the `March LR` and `March LA` tests are effectively used to detect dynamic unlinked faults [15], test dynamic memory (DRAM) [16], and implement memory built-in self-tests (BISTs) [17,18]. Specific march tests, such as `March SS`, `March BLC-Opt`, `March SR`, `March MSS`, and `March m-MSS`, are applied to cover bit-line coupling faults [19–21] and their diagnoses by applying `March Cd` [22].

The required sets of march elements and their sequence in the testing were determined as conditions for detecting various types of *LCF*s, ensuring the detection of all faults of the considered type [3]. For example, the condition for detecting all single *CFid*s and all linked double *LCFid*s is the presence in the test of the following marching elements $\Uparrow\{ra, w\bar{a}, \ldots, wa, \ldots\}; \Uparrow\{r\bar{a}, wa, \ldots, w\bar{a}, \ldots\}; \Downarrow\{ra, w\bar{a}, \ldots, wa, \ldots\}; \Downarrow\{r\bar{a}, wa, \ldots, w\bar{a}, \ldots\}$. In turn, the conditions for the presence of the necessary march elements in the test were formulated based on the conditions for activating a particular fault and the conditions for its detection. Similar conditions apply to the memory cells directly involved in the fault and describe the sequence of actions necessary to detect the fault. For the simplest *CF* of the form $\wedge<\uparrow,0>$, where cell a_i is the aggressor ($\uparrow$) and cell a_j is the victim (0), the following conditions must be fulfilled sequentially. First, the initial setting of both cells to the initial state is necessary: cell a_i to State 0 and cell a_j to State 1. Then, the fault activation condition is executed to perform a 0 to 1 state change in cell a_i. Finally, the condition for detecting the fault results from reading the content (0) of cell a_j and comparing it with the reference (previously set in cell a_j) value equal to 1.

To obtain and formulate the conditions for detecting complex, numerous, and diverse configurations of linked coupling faults *LCF*$(a_i, a_j, a_k, \ldots, a_z)$ including r cells $a_i, a_j, a_k, \ldots, a_z$ with an ordered relation of addresses $i < j < k < \ldots < z$, we introduce their formal description. The basis of such a description is the role of each cell of the considered *LCF*s concerning all other cells included in the fault description. Similarly, as in the simplest case of a *CF*, two cells, a_i and a_j, where $i \neq j$, can influence each other. One can be a victim (V) and the other an aggressor (A). In the examples of fault models *LCF*1, *LCF*2, *LCF*3, *LCF*4, and *LCF*5, the roles of each cell are indicated by links and arrows. For the general case, within the framework of linked faults that combine two single *CF*s that include the same two cells, a_i and a_j, we introduce the third role (both, *B*) of cell a_i, which means two roles, *A* and *V*, for cell a_j. Accordingly, cell a_j in relation to a_i also has two roles, *V* and *A*, denoted by the same symbol *B*. The absence of a connection (roles) between cells a_i and a_j is denoted by the symbol (-). Thus, in the description of the *LCF*, all participating cells and all their respective roles are presented.

In the formal description of *LCF*($a_i, a_j, a_k, \ldots, a_z$), the information is sequentially presented in the form of a set of symbols ($-$, *A*, *V*, and *B*) for all cells $a_i, a_j, a_k, \ldots, a_z$. When describing cells, their sequence $a_i, a_j, a_k, \ldots, a_z$ is also preserved (i.e., the first position of all the descriptions contains information about cell a_i, then a_j, and so on). This sequence is determined by the cell access time, corresponding to the sequence of their addresses for single march tests. As an example, we consider *LCF*6 presented in graphical form and the form of a new formal model (Figure 4).



*LCF*6

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

Figure 4. Linked coupling fault *LCF*6 with description.

In the above example, *LCF*6 cells a_k and a_l, included in this fault, form two simple *CF*s and act in relation to each other in both roles, indicated by the symbol *B* in the description of *LCF*6 (Figure 4). As explained, *B* indicates that the cell has two roles, and which one it performs is determined by the scenario, set by the sequence of accessing the cells within the framework of the march test. Depending on the direct $\Uparrow$ addressing, when the addresses are related as follows $i < j < k < \ldots < z$, or are reversed $\Downarrow$, when $i > j > k > \ldots > z$, the *LCF*6 description is presented as follows:

$$\begin{aligned} LCF6(a_i, a_j, a_k, a_l) &= \{\langle a_i, A, A, -\rangle; \langle V, a_j, -, -\rangle; \langle V, -, a_k, B\rangle; \langle -, -, B, a_l\rangle\} = \\ &\Uparrow\{\langle a_i, A, A, -\rangle; \langle V, a_j, -, -\rangle; \langle V, -, a_k, A\rangle; \langle -, -, V, a_l\rangle\}, \\ &\Downarrow\{\langle a_i, A, A, -\rangle; \langle V, a_j, -, -\rangle; \langle V, -, a_k, A\rangle; \langle -, -, V, a_l\rangle\}. \end{aligned} \tag{3}$$

As noted, the ordered ratio of cell addresses involved in the *LCF* failure corresponds to the time sequence of access to them within the framework of one-run march tests with a fixed address sequence. The relation of the values of the physical cell addresses in the *LCF* and the relation of the access times to these cells can be different. An example of such a situation is the failure presented in Figure 2. Physically, this is the same fault. However, its manifestation depends on the address sequence, which is displayed in the description of this fault for two cases of address sequences:

$$\begin{aligned} LCF1 &= LCF(a_i, a_j, a_k) = \{\langle a_i, -, A\rangle; \langle -, a_j, A\rangle; \langle V, V, a_k\rangle\} \\ LCF2 &= LCF(a_i, a_k, a_j) = \{\langle a_i, A, -\rangle; \langle V, a_k, V\rangle; \langle -, A, a_j\rangle\} \end{aligned} \tag{4}$$

Similarly, any of the possible *LCF*s are described. Examples of such descriptions for the previously given *LCF*s are provided below:

$$\begin{aligned} LCF3 &= LCF(a_i, a_j, a_k) = \{\langle a_i, A, -\rangle; \langle V, a_j, A\rangle; \langle -, V, a_k\rangle\}; \\ LCF4 &= LCF(a_i, a_j, a_k) = \{\langle a_i, A, A\rangle; \langle V, a_j, -\rangle; \langle V, -, a_k\rangle\}; \\ LCF5 &= LCF(a_i, a_j, a_k) = \{\langle a_i, B, B\rangle; \langle B, a_j, B\rangle; \langle B, B, a_k\rangle\}. \end{aligned}$$

For *LCF*6 (3), the *LCF*5 failure is represented as $\Uparrow\{\langle a_i, A, A\rangle;\ \langle V, a_j, A\rangle;\ \langle V, V, a_k\rangle\}$, $\Downarrow\{\langle a_i, V, V\rangle;\ \langle A, a_j, V\rangle;\ \langle A, A, a_k\rangle\}$.

The main advantage of the formal model for describing the configurations of linked faults is its compact notation containing complete information about all single *CF*s that form the *LCF*. For example, from the description of *LCF*1, it follows that this linked fault is formed from two single *CF*s: $CF(a_i, a_k) = \langle a_i, A \rangle$, $\langle V, a_k \rangle$ and $CF(a_j, a_k) = \langle a_j, A \rangle$,$\langle V, a_k \rangle$.

The ordered access to the cells $a_i, a_j, a_k, \ldots, a_z$ of the fault *LCF*$(a_i, a_j, a_k, \ldots, a_z)$, when implementing the march test element, is determined by the time sequence of generating their addresses $i, j, k, \ldots, z$. This sequence is a crucial element of the formal description of the *LCF*, which, for example, can be observed from the descriptions (4) of the *LCF* in Figure 2. Indeed, for the sequence of addresses i, j, and k, we have the *LCF*1 description of the fault, and for i, k, and j, another description is used: *LCF*2 (4). The position of a particular cell (e.g., cell a_k in its description) and the description itself in the formal *LCF* model correspond to the temporal order of accessing this cell when implementing the march test element. For example, in the formal model *LCF*1 = $\{\langle a_i, -, A \rangle;\langle -, a_j, A \rangle;\langle V, V, a_k \rangle\}$, the description of cell a_k is in the third position because the test generates the address k after forming addresses i and j. In the description of $\langle V, V, a_k \rangle$ of cell a_k, its designation is also given in the third position.

Thus, the basis of the new formal *LCF* model is the three roles that the cells of the linked coupling fault perform, aggressor (*A*), victim (*V*), and both (*B*), performed by two cells simultaneously in relation to each other. In reality, the proposed model describes the scenario for implementing the roles of memory fault cells, determined by the march test and the address sequence in this test to access the cells. For tests with a maximum fault coverage *CF*, the scenario is specified only by the address sequence, displayed in the formal *LCF* model, as demonstrated in the examples of *LCF*1 and *LCF*2.

For each cell $a_i, a_j, a_k, \ldots, a_z$, the *LCF*s distinguish between the left and right in the description. Moreover, those fault cells activated before activating the current cell are on the left side and are on the right side after activation. Activation refers to accessing a memory cell under the march element of the test and performing the corresponding write and read operations. The formal *LCF* model allows us to formulate the conditions for detecting such faults using one-run march tests.

Statement 2. *The necessary and sufficient conditions for detecting a linked coupling fault* LCF$(a_i, a_j, a_k, \ldots, a_z)$ *are listed below.*

1. *The application of a march test with an arbitrary address sequence, which detects all kinds of single coupling faults.*
2. *The presence in the formal model* LCF$(a_i, a_j, a_k, \ldots, a_z)$ *of a fault that corresponds to the address sequence of the applied test, at least one cell* a_l, $l \in \{i, j, k, \ldots, z\}$, *where only one role* V *is present in the right, or on the left side of the description, or in both parts.*

As demonstrated, an arbitrary fault *LCF*$(a_i, a_j, a_k, \ldots, a_z)$ is a composition of single *CF*s, where the entire set is detected by march tests, for example, the `March LA` and `March LR` [13,14]. All possible single *CF*s are guaranteed to be detected by the indicated tests only if they are not included in the *LCF* fault. When single *CF*s are included in the *LCF*, these tests detect them either by the wrong state of the aggressor cell or are guaranteed to provide a condition for their activation by changing the state of the victim cell and detecting the wrong state. When the aggressor cell cannot fulfill the activation condition, the presence of a faulty memory state (an *LCF*) is detected by the wrong state of this cell because tests that detect single *CF*s provide the conditions for activating all such faults by setting the required initial state of the aggressor cell $(0, 1)$ or performing a transition $(\uparrow, \downarrow)$ in it (see *CFin*, *CFid*, and *CFst*) [3]. Thus, all *CF* aggressor cells included in the *LCF* are either the cause of their detection, becoming victims within other *CF*s, or exercise their influence on the victim cells.

In most cases, these tests detect changes in the victim cells relative to the original reference values [3], which follows from the properties of tests that detect the complete set of single coupling faults [3]. The exception is cases of masking that occur when several aggressors influence one cell of the victim. For example, one aggressor changes the reference value of the victim to the opposite, and the second aggressor returns the original value of the victim cell. An example of masking is illustrated in Figure 2 and is described by the formal model *LCF*1 (4). As observed in this case, cell a_k has two victim roles $\langle V, V, a_k \rangle$ (4), and both are on the same (right) side in the description. When the victim cell has only one victim role (i.e., only one aggressor) before accessing this cell, masking is impossible. If the cell has two victim roles simultaneously, but in both parts of the formal description, the cell content is only read after a single change in its state due to the manifestation of a *CF*. Accordingly, an *LCF* is detected. A similar case is illustrated by an example of an *LCF*2 with the formal description of two roles of the victim *LCF*3 of cell a_k. However, in this case, one role, V, is on the right, and the second, V, is on the left in the formal description.

The masking mechanism is very diverse and is typical for memory devices. However, within the framework, this article considers the case of a memory *LCF* and the possible masking of the *CF*s included in them [23].

The guaranteed detection of any fault in the mutual influence of the *CF*, regardless of whether it is included in the linked fault of the *LCF*, is provided by analyzing the state of the victim cell by reading its state after realizing the influence of only one aggressor on it (see Statement 2). An example of the implementation of such a scenario is described by the formal model *LCF*3, where the state of the victim cell a_j ($\langle V, a_j, A\rangle$) is analyzed after only the aggressor cell a_i has been influenced. The presence of at least one cell a_l, $l \in \{i, j, k, \ldots, z\}$ of the formal model $LCF(a_i, a_j, a_k, \ldots, a_z)$ in the description, where there is only one victim role (on the right or left or both simultaneously), ensures the detection of $LCF(a_i, a_j, a_k, \ldots, a_z)$, which is a necessary and sufficient condition for *LCF* detection using a test covering single *CF*s.

As demonstrated, the cell role B in the *LCF* consists of two roles, each realized in one of the two phases of the test, direct or reverse, which is equivalent to having one role of the victim on the right or left side of the *LCF* description. The presence of two roles (V and B) on the right or left of the description of at least one cell in the *LCF* also guarantees the detection of this fault because, in one of the phases of the test, the role of B is equivalent to the role of A. Accordingly, this case is reduced to the case of the presence of one role of the victim. Thus, in the formal model $LCF(a_i, a_j, a_k, \ldots, a_z)$, corresponding to the scenario of the address sequence of the test, a necessary and sufficient condition for detecting this fault is at least one cell a_l that has only one of the roles of the victim V or B, or both simultaneously on the right or left.

Returning to the fault in Figure 2 for a sequence of addresses i, j, and k of a test that detects single communication faults and its description $\{\langle a_i, -, A\rangle;\langle -, a_j, A\rangle;\langle V, V, a_k\rangle\}$, this fault was not detected because the conditions of Statement 2 were not met. Using other address sequences (other scenarios) can ensure its detection, as demonstrated by various descriptions of this fault, depending on the time sequence of access to the cells in this fault:

$$\begin{aligned}
LCF(a_i, a_j, a_k) &= \{\langle a_i, -, A\rangle;\langle -, a_j, A\rangle;\langle V, V, a_k\rangle\};\\
LCF(a_i, a_k, a_j) &= \{\langle a_i, A, -\rangle;\langle V, a_k, V\rangle;\langle -, A, a_j\rangle\};\\
LCF(a_j, a_i, a_k) &= \{\langle a_j, -, A\rangle;\langle -, a_i, A\rangle;\langle V, V, a_k\rangle\};\\
LCF(a_j, a_k, a_i) &= \{\langle a_j, A, -\rangle;\langle V, a_k, V\rangle;\langle -, A, a_i\rangle\};\\
LCF(a_k, a_i, a_j) &= \{\langle a_k, V, V\rangle;\langle A, a_i, -\rangle;\langle A, -, a_j\rangle\};\\
LCF(a_k, a_j, a_i) &= \{\langle a_k, V, V\rangle;\langle A, a_j, -\rangle;\langle A, -, a_i\rangle\}.
\end{aligned} \tag{5}$$

In two ($LCF(a_i, a_k, a_j)$ and $LCF(a_j, a_k, a_i)$) of the six cases in (5), the conditions of Statement 2 are satisfied, providing the necessary and sufficient conditions for their detection.

4. Multirun March Tests for LCF Detection

Multi-cell faults such as pattern-sensitive faults (*PSF*s) or linked coupling faults (*LCF*s) are triggered when specific bit patterns occur in the cells affected by these faults or when a certain value change occurs in one of the cells. Due to the limitations of march tests, a single run of such a test can only generate a limited number of patterns in the selected memory cells. Increasing the number of generated patterns in memory cells cannot be achieved by simply increasing the test's complexity. One effective solution for achieving high fault coverage for multi-cell faults, as demonstrated in [24–30], is multirun testing. This testing concept involves executing the same test procedure multiple times, each time with different initial conditions (different initial memory backgrounds and/or address sequences). Executing the same march test under varying initial conditions generates different bit patterns in memory cells, thereby enabling the detection of a wider range of potential multi-cell faults, including various types of linked coupling faults.

The low coverage of *LCF*s by classic one-run march tests is explained by the sequential procedure for accessing memory cells and the unchanged test implementation scenario [23]. Analyzing the example of six formal descriptions (5) of the *LCF*1 in Figure 2 reveals that only two of the six scenarios guarantee the detection of this fault. Accordingly, the fault can be detected only by changing the scenarios and repeatedly applying the test. In this case, the scenario is the address sequence of the march test that detects the entire set of single coupling faults. Multiple (multirun) memory testing was considered within the exhaustive, pseudo-exhaustive, and transparent memory testing frameworks [3,23]. In all cases, it was necessary to repeat the march test for various scenarios determined by the degrees of freedom inherent in march tests [20]. March memory tests detect complex faults by applying them repeatedly for various address sequences or initial memory cell states [31]. An essential result of multirun memory testing is detecting all possible faulty states. However, achieving 100% coverage may require a sufficient number (multiplicity) of test applications.

For the variable address sequence that defines the scenario for $LCF(a_i, a_j, a_k, \ldots, a_z)$, a time sequence always exists for accessing cells $a_i, a_j, a_k, \ldots, a_z$, satisfying the conditions of Statement 2. An example is *LCF*1 (5), where two of the six sequences of cell addresses a_i, a_j, and a_k are detectable for this fault. In the general case, for any faulty memory state, a set of address sequences exists for which this memory state is detected by a single test. Accordingly, in a random address sequence selection procedure, the ratio p, representing the number of address sequences for which a fault is detectable, to the total number of sequences, expressed as a percentage, determines the fault coverage, FC, when detecting this fault using a one-run memory test with randomly selected address sequences. The value of $p = FC/100\%$ can be understood as the probability of fault detection, while the probability of this fault remaining undetectable is $1 - p$. For an f-run memory test with a random address sequence selection procedure, the probability of this fault being undetectable is $(1 - p)^f$ and the detection probability is estimated as $1 - (1 - p)^f$. Assuming that a single application of the march test allows for detecting the *LCF* failure with FC_{Test}, its f-time use with arbitrary (random) address sequences allows for reaching the value $FC_{Test}(f)$ determined as follows (6) [23]:

$$FC_{Test}(f) = \left(1 - \left(1 - \frac{FC_{Test}}{100\%}\right)^f\right) \times 100\%;\ f = 1, 2, 3, \ldots \tag{6}$$

As f increases, the value of the fault coverage $FC_{Test}(f)$ tends to 100%. For the *LCF*1 failure, considering (5), $FC_{Test} = (2/6) \times 100\% = 33.33\%$.

5. Experimental Investigations

The first experiment was provided according to the following conditions:

1. Memory size: 1024 one-bit cells;
2. *LCF*1 consisting of two single *CFin*s: $CFin(a_i, a_j) = \langle \uparrow, \updownarrow \rangle$ and $CFin(a_j, a_k) = \langle \uparrow, \updownarrow \rangle$ in cells a_i, a_j, and a_k with addresses $i < j < k$;
3. Address sequence: random;
4. Number of one-run march test $(f = 1)$ executions for the average value of $FC_{Test}(f)$ calculation: 100,000.

The results for the one-run memory testing in Table 1 satisfy the theoretical estimation (6) for march tests. The results in Table 1 originate from computer simulations conducted using custom tools developed by the authors. These simulations involved generating *LCF*1s in random memory cells and then executing the relevant test to detect the faults. For each test, 100,000 simulations were conducted, and the result of the test was recorded each time. Finally, the average fault coverage (FC) was calculated based on these simulations. The number of simulations was limited to 100,000 because it was observed that further increasing the number of simulations did not significantly affect the obtained average result.

Table 1. Fault coverage $FC_{Test}(f)$ of $LCF1$ for one-run march tests.

MATS++	March C-	March LA	March Y
33.21%	33.45%	33.15%	32.92%

The next experiment confirms the validity of the relation (6) for multirun memory testing $(f > 1)$ by applying the `March C-` test. The results of this experiment for the same $LCF1$ are provided in Figure 5. The results demonstrate the high efficiency of detecting $LCF1$s using multirun memory tests.

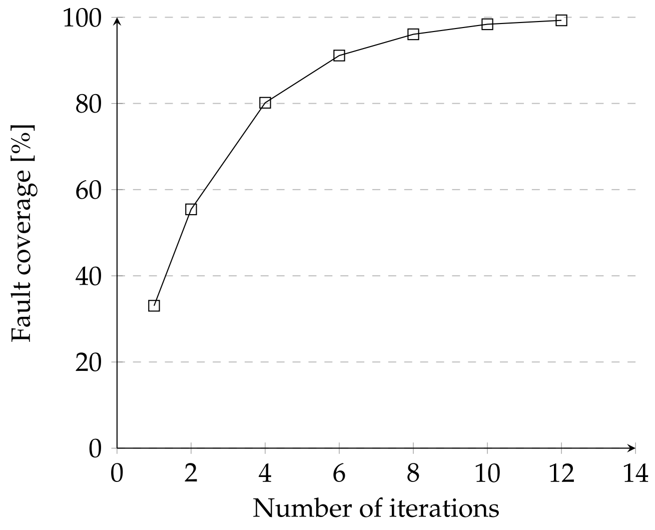


Figure 5. Fault coverage $FC_{Test}(f)$ of $LCF1$ for multirun `March C-` test (address sequences: random, and background: zeros).

The efficiency of multirun memory testing for detecting linked coupling faults was validated for various LCF and memory tests. The experiment conditions are similar to the previous ones, except for the LCF fault configurations. The critical element of the multirun testing for this experiment is a random address sequence applied at each test iteration.

It should be noted that the proposed approach is applicable for detecting previously undetectable memory linked coupling faults using march tests. March tests exhibit computational complexity that linearly depends on the memory size N in bits. The approach shares comparable computational complexity with classical march tests and also linearly depends on the memory size. For instance, the `March LR` test has a complexity of $22N$, but it does not enable the detection of undetectable linked coupling faults. Conversely, based on the proposed approach, the `MATS+` application ($5N$), in a multirun scenario (four runs), facilitates the detection of 89.57% of the undetectable linked coupling faults (see Table 2). The complexity of this procedure is estimated as $4 \times 5N = 20N$.

Tables 2 and 3 list the high efficiency of detecting complex faults with multirun tests. With an increase in the number of iterations, the test efficiency levels off. As listed in Tables 2 and 3, after four iterations, test `MATS+` has the same effectiveness as the more complex test `March C-` designed as a CF detection test.

A similar dependence is also observed for the case of multirun tests based on a variable initial memory state (random backgrounds) of the tested memory. The same faults were analyzed as in the previous experiment, and the `March C-` and simplest one-phase tests $\{\Uparrow(ra, w\bar{a})\}$ were used. The results for both cases are presented in Tables 4 and 5.

As presented in Tables 4 and 5, multirun testing with a random background technique results in the same $FC_{Test}(f)$ for both simple tests like “one march element” ($\{\Uparrow(ra, w\bar{a})\}$) and complex tests, such as `March C-`.

Table 2. Fault coverage $FC_{Test}(f)$ of the *LCF* for the multirun `MATS+` test (address sequences: random, and background: zeros).

$\boldsymbol{CFin(a_i, a_j)}$	$\boldsymbol{CFin(a_j, a_k)}$	$\boldsymbol{FC_{Test}(f)[\%]}$					
		$\boldsymbol{f = 1}$	$\boldsymbol{f = 2}$	$\boldsymbol{f = 4}$	$\boldsymbol{f = 8}$	$\boldsymbol{f = 10}$	$\boldsymbol{f = 12}$
$\wedge<\uparrow,\updownarrow>$	$\wedge<\uparrow,\updownarrow>$	33.53	55.93	80.51	96.25	98.53	99.31
$\wedge<\uparrow,\updownarrow>$	$\wedge<\downarrow,\updownarrow>$	66.35	88.59	98.64	99.99	100.00	100.00
$\wedge<\downarrow,\updownarrow>$	$\wedge<\uparrow,\updownarrow>$	66.47	88.83	98.62	99.98	100.00	100.00
$\wedge<\downarrow,\updownarrow>$	$\wedge<\downarrow,\updownarrow>$	33.52	56.06	80.49	96.23	98.53	99.34
TOTAL $FC_{Test}(f)$		49.97	72.35	89.57	98.12	99.27	99.66

Table 3. Fault coverage $\mathrm{FC}_{Test}(f)$ of the *LCF* for the multirun `March C-` test (address sequences: random, and background: zeros).

$\boldsymbol{CFin(a_i, a_j)}$	$\boldsymbol{CFin(a_j, a_k)}$	$\boldsymbol{FC_{Test}(f)[\%]}$					
		$\boldsymbol{f = 1}$	$\boldsymbol{f = 2}$	$\boldsymbol{f = 4}$	$\boldsymbol{f = 8}$	$\boldsymbol{f = 10}$	$\boldsymbol{f = 12}$
$\wedge<\uparrow,\updownarrow>$	$\wedge<\uparrow,\updownarrow>$	33.65	55.91	80.41	96.24	98.52	99.33
$\wedge<\uparrow,\updownarrow>$	$\wedge<\downarrow,\updownarrow>$	100.00	100.00	100.00	100.00	100.00	100.00
$\wedge<\downarrow,\updownarrow>$	$\wedge<\uparrow,\updownarrow>$	100.00	100.00	100.00	100.00	100.00	100.00
$\wedge<\downarrow,\updownarrow>$	$\wedge<\downarrow,\updownarrow>$	33.24	56.06	80.40	96.24	98.53	99.30
TOTAL $FC_{Test}(f)$		66.72	77.99	90.20	98.12	99.26	99.65

Table 4. Fault coverage $\mathrm{FC}_{Test}(f)$ of the *LCF* for the multirun `March C-` test (address sequences: counting sequences, and background: random).

$\boldsymbol{CFin(a_i, a_j)}$	$\boldsymbol{CFin(a_j, a_k)}$	$\boldsymbol{FC_{Test}(f)[\%]}$					
		$\boldsymbol{f = 1}$	$\boldsymbol{f = 2}$	$\boldsymbol{f = 4}$	$\boldsymbol{f = 8}$	$\boldsymbol{f = 10}$	$\boldsymbol{f = 12}$
$\wedge<\uparrow,\updownarrow>$	$\wedge<\uparrow,\updownarrow>$	49.97	75.26	93.70	100.00	100.00	100.00
$\wedge<\uparrow,\updownarrow>$	$\wedge<\downarrow,\updownarrow>$	50.04	74.25	94.65	99.51	100.00	100.00
$\wedge<\downarrow,\updownarrow>$	$\wedge<\uparrow,\updownarrow>$	49.99	75.03	94.11	100.00	100.00	100.00
$\wedge<\downarrow,\updownarrow>$	$\wedge<\downarrow,\updownarrow>$	50.15	75.24	95.27	99.71	100.00	100.00
TOTAL $FC_{Test}(f)$		50.04	74.95	94.43	99.80	100.00	100.00

Table 5. Fault coverage $\mathrm{FC}_{Test}(f)$ of the *LCF* for the multirun $\{\Uparrow(ra, w\bar{a})\}$ test (address sequences: counting sequences, and background: random).

$\boldsymbol{CFin(a_i, a_j)}$	$\boldsymbol{CFin(a_j, a_k)}$	$\boldsymbol{FC_{Test}(f)[\%]}$					
		$\boldsymbol{f = 1}$	$\boldsymbol{f = 2}$	$\boldsymbol{f = 4}$	$\boldsymbol{f = 8}$	$\boldsymbol{f = 10}$	$\boldsymbol{f = 12}$
$\wedge<\uparrow,\updownarrow>$	$\wedge<\uparrow,\updownarrow>$	49.98	74.42	93.57	100.00	100.00	100.00
$\wedge<\uparrow,\updownarrow>$	$\wedge<\downarrow,\updownarrow>$	49.93	74.09	94.69	100.00	100.00	100.00
$\wedge<\downarrow,\updownarrow>$	$\wedge<\uparrow,\updownarrow>$	50.05	75.27	94.46	100.00	100.00	100.00
$\wedge<\downarrow,\updownarrow>$	$\wedge<\downarrow,\updownarrow>$	50.15	75.07	93.29	100.00	100.00	100.00
TOTAL $FC_{Test}(f)$		50.02	74.71	94.00	100.00	100.00	100.00

6. Conclusions

A formal mathematical model is proposed to describe linked coupling faults, based on the behavior of cells involved in the fault. The cell behavior scenario is determined by the address sequence of a one-run march test. The necessary and sufficient conditions for detecting linked coupling faults are provided. Theoretically and experimentally, undetectable *LCF*s are effectively detected by multirun tests using random address sequences and variable initial memory states. The experiments demonstrate the high efficiency of detecting *LCF*1s using multirun memory tests. For instance, an eight-run test with a random background technique achieves 100% coverage of *LCF*1s, even for simple tests, like the "one march element" ($\{\Uparrow(r0, w1)\}$). The proposed mathematical model for linked coupling faults appears promising for their identification and diagnostic localization. It is crucial in further research to utilize the mathematical model proposed by the authors to describe other complex memory faults, including other types of undetectable faults, especially neighborhood pattern-sensitive faults (*NPSF*s). This approach, according to the authors, will enable the formulation of necessary and sufficient conditions for detecting other types of complex memory faults.

Author Contributions: Conceptualization, V.N.Y.; Methodology, V.N.Y.; Validation, V.N.Y. and I.M.; Formal analysis, V.N.Y. and I.M.; Investigation, V.N.Y. and I.M.; Writing—review & editing, V.N.Y. and I.M. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by grant WZ/WI-IIT/3/2023 from the Faculty of Computer Science at Bialystok University of Technology, Ministry of Science and Higer Education, Poland.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lee, K.; Kim, J.; Baeg, S. Fault Coverage Re-Evaluation of Memory Test Algorithms With Physical Memory Characteristics. *IEEE Access* **2021**, *9*, 124632–124639. [CrossRef]
2. Wojciechowski, A.A.; Marcinek, K.; Pleskacz, W.A. Configurable MBIST Processor for Embedded Memories Testing. In Proceedings of the 2019 MIXDES—26th International Conference "Mixed Design of Integrated Circuits and Systems", Rzeszow, Poland, 27–29 June 2019; pp. 341–344. [CrossRef]
3. Goor, A.J.v.d. *Testing Semiconductor Memories: Theory and Practice*; John Wiley & Sons: Chichester, UK, 1991.
4. Hamdioui, S.; Gaydadjiev, G.; Van de Goor, A.J. The state-of-art and future trends in testing embedded memories. In Records of the 2004 International Workshop on Memory Technology, Design and Testing, San Jose, CA, USA, 10 August 2004; IEEE: Piscataway, NJ, USA, 2004; pp. 54–59. [CrossRef]
5. Koshy, T.; Arun, C.S. Diagnostic data detection of faults in RAM using different march algorithms with BIST scheme. In Proceedings of the 2016 International Conference on Emerging Technological Trends (ICETT), Kollam, India, 21–22 October 2016; pp. 1–6. [CrossRef]
6. Caşcaval, P.; Bennett, S. Efficient march test for 3-coupling faults in random access memories. *Microprocess. Microsystems* **2001**, *24*, 501–509. [CrossRef]
7. Caşcaval, P.; Caşcaval, D. March test algorithm for unlinked static reduced three-cell coupling faults in random-access memories. *Microelectron. J.* **2019**, *93*, 104619. [CrossRef]
8. Nor, A.; Zakaria, N.; Wan Hasan, W.; Abdul Halin, I.; Sidek, R.; Wen, X. Fault Detection with Optimum March Test Algorithm. In Proceedings of the 3rd International Conference on Intelligent Systems Modelling and Simulation, ISMS 2012, Kota Kinabalu, Malaysia, 8–10 February 2012; Volume 47. [CrossRef]
9. Cockburn, B.F. Deterministic Tests for Detecting Single V–Coupling Faults In RAMs. *J. Electron. Test.* **1994**, *5*, 91–113. [CrossRef]
10. Harutyunyan, G.; Shoukourian, S.; Vardanian, V.; Zorian, Y. A New Method for March Test Algorithm Generation and Its Application for Fault Detection in RAMs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2012**, *31*, 941–949. [CrossRef]
11. Jidin, A.Z.; Hussin, R.; Mispan, M.S.; Fook, L.W. Novel March Test Algorithm Optimization Strategy for Improving Unlinked Faults Detection. In Proceedings of the 2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS), Penang, Malaysia, 22–26 November 2021; pp. 117–120. [CrossRef]
12. Jidin, A.Z.; Hussin, R.; Fook, L.W.; Mispan, M.S. A review paper on memory fault models and test algorithms. *Bull. Electr. Eng. Inform.* **2021**, *10*, 3083–3093. [CrossRef]
13. van de Goor, A.; Gaydadjiev, G.; Mikitjuk, V.; Yarmolik, V. March LR: A test for realistic linked faults. In Proceedings of the 14th VLSI Test Symposium, Princeton, NJ, USA, 28 April–1 May 1996; pp. 272–280. [CrossRef]
14. van de Goor, A.; Gaydadjiev, G.; Yarmolik, V.; Mikitjuk, V. March LA: A test for linked memory faults. In Proceedings of the European Design and Test Conference. ED and TC 97, Paris, France, 17–20 March 1997; p. 627. [CrossRef]
15. Ying, L.W.; Hussin, R.; Ahmad, N.; Fook, L.W.; Jidin, A.Z. Modified March MSS for Unlinked Dynamic Faults Detection. In Proceedings of the 2022 IEEE 20th Student Conference on Research and Development (SCOReD), Bangi, Malaysia, 8–9 November 2022; pp. 68–72. [CrossRef]

16. Chou, C.W.; Chen, Y.X.; Li, J.F. Testing Inter-Word Coupling Faults of Wide I/O DRAMs. In Proceedings of the 2015 IEEE 24th Asian Test Symposium (ATS), Mumbai, India, 22–25 November 2015; pp. 67–72. [CrossRef]
17. Manasa, R.; Verma, R.; Koppad, D. Implementation of BIST Technology using March-LR Algorithm. In Proceedings of the 2019 4th International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT), Bangalore, India, 17–18 May 2019; pp. 1208–1212. [CrossRef]
18. 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. *J. Eng. Technol. (JET)* **2023**, *13*, 67–80.
19. Irobi, S.; Al-Ars, Z.; Hamdioui, S. Detecting memory faults in the presence of bit line coupling in SRAM devices. In Proceedings of the 2010 IEEE International Test Conference, Austin, TX, USA, 2–4 November 2010; pp. 1–10. [CrossRef]
20. Zordan, L.B.; Bosio, A.; Dilillo, L.; Girard, P.; Pravossoudovitch, S.; Virazel, A.; Badereddine, N. Optimized march test flow for detecting memory faults in SRAM devices under bit line coupling. In Proceedings of the 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, Cottbus, Germany, 13–15 April 2011; pp. 353–358. [CrossRef]
21. Azevedo, J.; Virazel, A.; Bosio, A.; Dilillo, L.; Girard, P.; Todri, A.; Prenat, G.; Alvarez-Herault, J.; Mackay, K. Coupling-based resistive-open defects in TAS-MRAM architectures. In Proceedings of the 2012 17th IEEE European Test Symposium (ETS), Annecy, France, 28–31 May 2012; p. 1. [CrossRef]
22. Hasan, W.Z.W.; Halin, I.; Shafie, S.; Othman, M. An efficient diagnosis march-based algorithm for coupling faults in SRAM. In Proceedings of the 2011 IEEE Regional Symposium on Micro and Nano Electronics, Kota Kinabalu, Malaysia, 28–30 September 2011; pp. 198–201. [CrossRef]
23. Yarmolik, V. *Testing and Diagnoses of Computer Systems*; Bestprint: Minsk, Belarus, 2019. (In Russian)
24. Cockburn, B.F. Deterministic tests for detecting scrambled pattern-sensitive faults in RAMs. In Proceedings of the MTDT '95: Proceedings of the 1995 IEEE International Workshop on Memory Technology, Design and Testing, San Jose, CA, USA, 7–8 August 1995; pp. 117–122.
25. Yarmolik, V.N.; Klimets, Y.V.; Demidenko, S.N. March PS(23N) Test for DRAM Pattern-Sensitive Faults. In Proceedings of the Asian Test Symposium, Singapore, 2–4 December 1998; p. 354.
26. Malaiya, Y.K. Antirandom Testing: Getting The Most Out Of Black-Box Testing. In Proceedings of the 6th IEEE International Symposium on Software Reliability Engineering, ISSRE '95, Toulouse, France, 24–27 October 1995; pp. 86–95.
27. Mrozek, I. Analysis of multibackground memory testing techniques. *Int. J. Appl. Math. Comput. Sci.* **2010**, *20*, 191–205. [CrossRef]
28. Mrozek, I.; Yarmolik, V. Optimal Backgrounds Selection for Multi Run Memory Testing. In Proceedings of the 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2008, Bratislava, Slovakia, 16–18 April 2008; pp. 332–338. [CrossRef]
29. Das, D.; Karpovsky, M. Exhaustive and Near-Exhaustive Memory Testing Techniques and their BIST Implementations. *J. Electron. Test.* **1997**, *10*, 215–229. [CrossRef]
30. Huzum, C.; Cascaval, P. A Multibackground March Test for Static Neighborhood Pattern-Sensitive Faults in Random-Access Memories. *Electron. Electr. Eng.* **2012**, *119*, 81–86.
31. Mrozek, I.; Yarmolik, V.N. Transparent Memory Tests Based on the Double Address Sequences. *Entropy* **2021**, *23*, 894. [CrossRef] [PubMed]