A New Method for March Test Algorithm Generation and Its Application for Fault Detection in RAMs

Gurgen Harutyunyan, Samvel Shoukourian, Valery Vardanian, and Yervant Zorian, *Fellow, IEEE*

***Abstract*—In this paper, all linked and unlinked static and two-operation dynamic faults are considered. A classification for their description is introduced. To generate a test algorithm for detection of all the considered faults, it was shown that it is not an easy problem. For this purpose, a new structure-oriented method is developed. Based on the proposed method, an efficient test algorithm March LSD of complexity 75N is generated for the detection of the considered linked static and dynamic faults.**

***Index Terms*—Linked faults, March test, random access memory, static and dynamic faults, test generation.**

I. Introduction

MANY FUNCTIONAL fault models (FFMs) for static and dynamic random access memories (RAMs) have been introduced in the past. Specifically, static and dynamic FFMs were introduced in [1] and [2]. Further, a concept of linked functional fault models was developed [3]. Based on inductive fault analysis (IFA), their existence and importance in current memories was validated experimentally [4]–[12].

Earlier, an efficient test algorithm March SS of complexity 22N [1] and the minimal test algorithm March MSS of complexity 18N [13] were proposed for detection of all unlinked static faults. For detection of all linked and unlinked static faults test algorithm March SL of complexity 41N [3] was introduced. In [3], the class of all linked static faults was attempted for the first time to classify. However, they assumed erroneously that the number of all static linked faults was 480. In addition to the described linked faults, in [14] extra new 120 linked faults were described. Thus, it was assumed that the number of all static linked faults was essentially higher—600. This was done based on introduction of the notion of "2-composite faults," thus extending the notion of linked static faults and covering all linked and unlinked static faults. Correspondingly, in [14], a minimal test algorithm March MSL of complexity 23N was proposed for detection of all "2-composite" (linked and unlinked) static faults.

Manuscript received June 15, 2011; revised October 6, 2011; accepted January 4, 2012. Date of current version May 18, 2012. This paper was recommended by Associate Editor C.-W. Wu.

G. Harutyunyan and V. Vardanian are with Synopsys, Inc., Yerevan 0026, Armenia (e-mail: gharutyu@synopsys.com; vvardani@synopsys.com).

S. Shoukourian is with the IT Educational and Research Center, Yerevan State University, Yerevan 0025, Armenia, and also with Synopsys, Inc., Yerevan 0026, Armenia (e-mail: samshouk@synopsys.com).

Y. Zorian is with Synopsys, Inc., Mountain View, CA 94043 USA (e-mail: zorian@synopsys.com).

Digital Object Identifier 10.1109/TCAD.2012.2184107

In [4], the authors showed the importance of dynamic faults for new static random access memory technologies. It has been shown [5], [6], [8] that dynamic faults can manifest themselves in many practical applications [15], [16] and development of test algorithms for their detection is important. In [10], the authors investigated thoroughly, both theoretically and practically, the nature and the root cause of dynamic faults based mainly on resistive open defects in memory circuits. Many dynamic faults were validated and the realistic ones were shown.

The considered functional fault models are validated by us based on electrical circuit level analysis, customer feedbacks, and experiments on test chips. Different types of faults, including static and dynamic faults, address decoder faults, interport and intraport faults, process variation faults, and also linked static and dynamic faults were also considered.

During test algorithm creation we pass through the following three major levels.

1) Layout to electrical circuit extraction: the extraction is done using memory structural information.
2) Electrical circuit to fault modeling extraction: the extraction is done using circuit level simulation. A comprehensive set of faults are injected on electrical circuits (memory array, address decoder, sense amplifier, write driver) and are validated by the experiments.
3) Fault modeling to test algorithm extraction: the extraction is performed using test algorithm generation tools.

In [15], a minimal test algorithm March MD2 of complexity 70N was proposed for detection of all dynamic two-operation single-cell faults, as well as the subclass of two-operation two-cell dynamic faults where both of the sensitizing operations were applied either to the aggressor cell or to the victim cell. In [15] and [16], only these subclasses of dynamic faults were considered. The other two subclasses with two sensitizing operations to be applied sequentially, one to the aggressor (respectively, victim) cell and the other one to the victim (respectively, aggressor) cell, were not considered due to high complexity of the corresponding cases. If the memory scrambling is not available (this is the typical case) then the test algorithm has to consider all possible pairs of memory cells for completeness of testing, and the algorithm will have complexity quadratic with respect to the number of memory cells (words). Such algorithms are not feasible for testing large memories.

In [15], there was shown that March MD2 additionally detected all linked and unlinked static faults. Since in this paper we consider only two-operation dynamic faults, in the sequel we will call them just "dynamic faults."

In this paper, based on the notion of "2-composite" faults introduced in [14], we define the whole space of linked and unlinked static and dynamic faults and classify it. All possible links between static and dynamic faults are considered, i.e., "static*static," "static*dynamic," and "dynamic*dynamic" where "X*Y" means all the links between all units of X and Y. Thus, the whole universe of known functional fault models (static and two-operation dynamic faults and their links) is considered.

We proposed an efficient test algorithm March LSD of complexity 75N that detects all the 2-composite static and dynamic faults, including all linked and unlinked static and dynamic faults. "LSD" stands for "linked static and dynamic." The test algorithm March LSD is considered as "efficient" since its complexity is very close to the complexity of March MD2 [15], while March LSD additionally detects all the faults from the subclasses of linked faults such as static*dynamic and dynamic*dynamic faults.

The test algorithm March LSD is generated by a new method which is sufficiently general and efficient to generate symmetric March test algorithms for different combinations (links) of static and dynamic faults. The method is based on the observation that almost all known minimal or efficient March test algorithms are symmetric. In Section V, several examples of symmetric March test algorithms generated by the proposed method are adduced. Some of them were known previously. Generation of only symmetric March test algorithms reduces drastically the set of all candidate March test algorithms to be considered.

For the sake of completeness with respect to the space of all linked and unlinked static and dynamic faults, it seems justified to develop test algorithms for the whole space of faults both from theoretical and practical points of view. A nonrealistic for the current technology fault may appear to be realistic in a future technology. Thus, excluding some faults from consideration does not seem justified. Of course, it would be ideal to have information on all realistic static and dynamic faults for the designs of certain companies and certain technologies, but in real life usually such detailed information is missing. In such cases, development of universal test algorithms for detection of all linked and unlinked static and dynamic faults would be very useful.

This paper is organized as follows. Section II gives the main notations and definitions. In Section III, 2-composite static and dynamic faults are defined. An efficient March test algorithm for detection of all 2-composite static and dynamic faults is described in Section IV. In Section V, a new method for generation of symmetric March test algorithms is introduced, and finally Section VI ends with conclusions.

II. Notations and Definitions

A March test algorithm M is a test algorithm with a finite number of March elements $M = M_1; M_2; \ldots; M_k$ [17], where each March element M_i consists of an addressing order A_i and a finite number of Read/Write operations

$M_i = A_i(O_1D_1, \ldots, O_mD_m)$.

1) $A_i \in \{\Uparrow, \Downarrow, \Updownarrow\}$—addressing order: $\Uparrow$—ascending, $\Downarrow$—descending, $\Updownarrow$—arbitrary.
2) $O_j \in \{R, W\}$—operations: R—Read, W—Write.
3) D_j—background pattern.

The definition of the fault primitive (FP) concept, used to define faults, can be found in [1] and [2]. Single-cell FPs are described by <S/F/R> and two-cell (coupling) FPs are described by $<S_a; S_v/F/R>$. In notations of FPs, S, S_a, and S_v are the sequences of operations required for fault sensitization (S is applied to the faulty cell, S_a—to the aggressor cell, and S_v—to the victim cell), $F \in \{0, 1\}$ is the observed memory behavior that deviates from the expected one. $R \in \{0, 1, -\}$ is the result of a Read operation applied to the faulty cell, in case if the last operation of S is a Read operation." - " is used when the last operation of S is not a Read operation. Note that if S_a is a sequence of operations then S_v should be a state; if S_a is a state then S_v can be a state or a sequence of operations. For example, if a cell has the fault <0W0/1/->, it means that if it contains value 0, then applying operation W0 on it will flip the cell value from 0 to 1. Or if two cells contain the fault <1; 0W1R1/0/1>, it means the following: if the aggressor cell has value 1, the victim cell has value 0 then applying two sequential operations {W1, R1} will fail. Though the read operation will return the correct value 1, the victim cell value will remain 0.

A FFM is defined as a nonempty set of FPs. The difference between static and dynamic faults is determined by the number of operations required in S, S_a, or S_v. Static faults are the faults sensitized by performing at most one operation. Dynamic faults are the faults that are sensitized by performing more than one operation.

We consider all unlinked static faults and all dynamic two-operation single-cell faults, as well as the subclass of two-operation two-cell unlinked dynamic faults where both of the sensitizing operations are applied either to the aggressor cell or to the victim cell. The faults of this subclass are considered as the most important dynamic faults in RAMs [2], [4], [15]. Furthermore, we will call them just unlinked dynamic faults.

The unlinked static and dynamic faults are listed in Table I (see [1], [2]). The symbol "$\sim$" used in the table denotes logical negation, and x, y, z, $t \in \{0, 1\}$. All these faults are presented by <S/F/R> and $<S_a; S_v/F/R>$. For example, TF includes FPs <0W1/0/-> and <1W0/1/->, while CFdrd includes FPs <0, 0R0/1/0>, <1, 0R0/1/0>, <0, 1R1/0/1>, and <1, 1R1/0/1>.

In [3], a definition of static linked faults (LFs) is introduced. In [14], the LF definition is generalized by removing some constraints given in [3] and extending the class of static linked faults. In this paper, we use the same definition to define dynamic linked faults, as well as the links between static and dynamic faults.

LF definition [3]: Let $FP_1 = <S_1/F_1/R_1>$ and $FP_2 = <S_2/F_2/R_2>$ be any static or dynamic fault primitives. If FP_1 and FP_2 share the same victim cell, then it is said that there is a linked fault (denoted as $FP_1 \rightarrow FP_2$) if the following three

TABLE I
UNLINKED STATIC AND TWO-OPERATION DYNAMIC FAULTS

Subclass	Functional Fault Models	Fault Primitives
Unlinked static single-cell faults	State fault (SF)	<x/∼x/->
	Transition fault (TF)	<xW(∼x)/x/->
	Write destructive fault (WDF)	<xWx/∼x/->
	Read destructive fault (RDF)	<Rx/∼x/∼x>
	Deceptive read destructive fault (DRDF)	<Rx/∼x/x>
	Incorrect read fault (IRF)	<Rx/x/∼x>
Unlinked static two-cell faults	State coupling fault (CFst)	<x; y/∼y/->
	Transition coupling fault (CFtr)	<x; yW(∼y)/y/->
	Write destructive coupling fault (CFwd)	<x; yWy/∼y/->
	Read destructive coupling fault (CFrd)	<x; Ry/∼y/∼y>
	Deceptive read destructive coupling fault (CFdrd)	<x; Ry/∼y/y>
	Incorrect read coupling fault (CFir)	<x; Ry/y/∼y>
	Disturb coupling fault (CFds)	<Rx; y/∼y/->, <xWy; z/∼z/->
Unlinked two-operation dynamic single-cell faults	Dynamic read destructive fault (dRDF)	<xWyRy/∼y/∼y>,<xRxRx/∼x/∼x>
	Dynamic deceptive read destructive fault (dDRDF)	<xWyRy/∼y/y>, <xRxRx/∼x/x>
	Dynamic incorrect read fault (dIRF)	<xWyRy/y/∼y>, <xRxRx/x/∼x>
	Dynamic transition fault (dTF)	<xWyW(∼y)/y/->, <xRxW(∼x)/x/->
	Dynamic write destructive fault (dWDF)	<xWyWy/∼y/->, <xRxWx/∼x/->
Unlinked two-operation dynamic two-cell faults	Dynamic read destructive coupling fault (dCFrd)	<x; yWzRz/∼z/∼z>, <x; zRzRz/∼z/∼z>
	Dynamic deceptive read destructive coupling fault (dCFdrd)	<x; yWzRz/∼z/z>, <x; zRzRz/∼z/z>
	Dynamic incorrect read coupling fault (dCFir)	<x; yWzRz/z/∼z>, <x; zRzRz/z/∼z>
	Dynamic transition coupling fault (dCFtr)	<x; yWzW(∼z)/z/->, <x; zRzW(∼z)/z/->
	Dynamic write destructive coupling fault (dCFwd)	<x; yWzWz/∼z/->, <x; zRzWz/∼z/->
	Dynamic disturb coupling fault (dCFds)	<xWyWt; z/∼z/->, <xWyRy; z/∼z/->, <xRxWy; z/∼z/->, <xRxRx; z/∼z/->

conditions are satisfied.

1) C_1: read operations of FP_1 and FP_2 do not detect a fault.
2) C_2: FP_2 masks FP_1, i.e., $F_2 = \sim F_1$.
3) C_3: FP_2 is compatible with FP_1, which means that if S_2 is applied immediately after S_1, then the final state of the aggressor or the victim cell after performing S_1 should be the same as the initial state required by S_2.

In [3], the LFs are divided into five groups as it is shown in Fig. 1.

1) LF1: combination of two single-cell unlinked faults. Both faults have the same faulty (victim) cell.
2) $LF2_{av}$: combination of one two-cell FP_1 and one single-cell FP_2 unlinked faults, where FP_1 is sensitized first. The victim cell of FP_1 coincides with the faulty cell of FP_2.
3) $LF2_{va}$: combination of one single-cell FP_1 and one two-cell FP_2 unlinked faults, where FP_1 is sensitized first. The faulty cell of FP_1 coincides with the victim cell of FP_2.
4) $LF2_{aa}$: combination of two two-cell unlinked faults FP_1 and FP_2. FP_1 and FP_2 have the same aggressor, as well as the same victim cells.
5) LF3: combination of two two-cell unlinked faults FP_1 and FP_2. FP_1 and FP_2 have the same victim cells, but different aggressor cells.

III. 2-COMPOSITE STATIC AND DYNAMIC FAULTS

In this section, we will introduce all 2-composite static and dynamic faults. In [14], the notion of 2-composite faults was introduced. This allows defining all the possible links between the considered faults. Let $FP_1=<S_1/F_1/R_1>$ and $FP_2=<S_2/F_2/R_2>$ be any fault primitives. "2-composite fault," denoted as FP_1*FP_2, is the combination of fault primitives FP_1 and FP_2 having the same victim cell (if both FPs are two-cell faults, then their aggressor cells can be either the same or different) with the only restriction that these FPs cannot be sensitized simultaneously. This limitation is to prevent nonrealistic cases. For example, in <1; 0R0/1/0>*<1; 0R0/0/1> the two FPs are sensitized simultaneously by the R0 operation applied to the victim cell if the aggressor cells of both FPs are in state 1. First FP tends to flip the victim cell to state 1 while the second tends to keep the state of the cell in 0. As a result, the victim cell may accept a random value. Similarly, the output may also tend to a random value. We will exclude from our consideration such faults that may lead to random outputs. Such faults are considered as nonrealistic (see [3], [17], [24]).

In 2-composite faults there is no special order for fault sensitization, i.e., $FP_1*FP_2 = FP_2*FP_1$. The fault sensitization depends on the structure of a March test algorithm, for example, one March test algorithm can first sensitize FP_1, the second one—FP_2. All three conditions (restrictions) defined for linked faults in [3] are not taken into account in the definition of 2-composite faults. Thus, 2-composite fault class is essentially wider than the class of "linked" faults defined in [3].

2-composite faults include unlinked and linked faults, as well as two faults having the same victim cell but not masking each other. For example <0R0; 0/1/->*<0R0; 1/0/-> and <0R0; 0/1/->*<0R0; 0/1/-> both are 2-composite faults. The

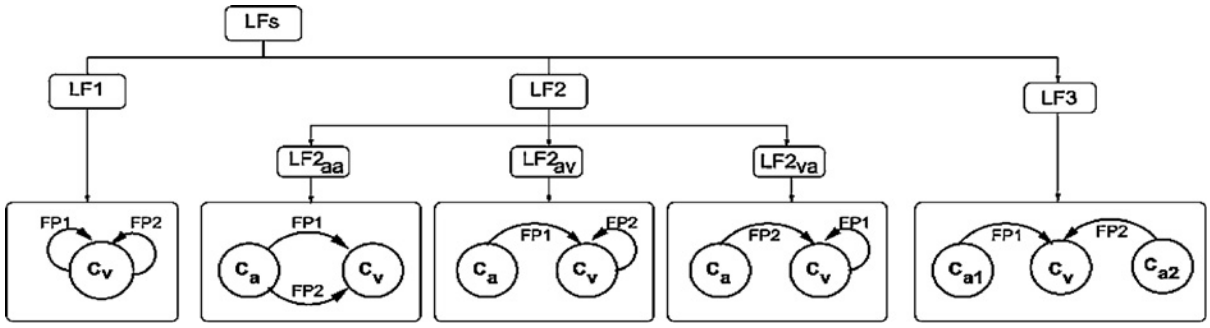

Fig. 1. Classification of LFs.

first fault is considered as a linked fault since it satisfies all three conditions C_1, C_2, and C_3 described in Section II. The second fault is not considered as a linked fault since condition C_2 is violated. However, the second fault also can be realistic in RAMs. Thus, it is expedient to consider all the possible cases of two static and dynamic faults having the same victim cell (none of the conditions C_1, C_2, and C_3 is taken into account).

In FP_1*FP_2, in general, FP_1 and FP_2 can be any types of faults. In this paper, as FP_1 and FP_2 we will consider only static and dynamic faults (see Table I). It is allowed that $FP_1 = \emptyset$ or $FP_2 = \emptyset$ ($\emptyset$ is empty FP) but not simultaneously $FP_1 = \emptyset$ and $FP_2 = \emptyset$. The following equalities are true, $FP*\emptyset = \emptyset*FP = FP$.

The whole space of 2-composite static and dynamic faults can be divided into five classes:

1) unlinked static faults: FP*$\emptyset$, where FP is an unlinked static fault;
2) unlinked dynamic faults: FP*$\emptyset$, where FP is an unlinked dynamic fault;
3) linked static*static faults: FP_1*FP_2, where FP_1 and FP_2 are both unlinked static faults;
4) linked static*dynamic faults: FP_1*FP_2, where FP_1 is an unlinked static fault and FP_2 is an unlinked dynamic fault;
5) linked dynamic*dynamic faults: FP_1*FP_2, where FP_1 and FP_2 are both unlinked dynamic faults.

Based on the classifications of LFs and 2-composite faults, we can classify the universe of static and dynamic faults (see Fig. 2). In the figure, the class $LF2_{va}$ is not depicted since it is the same class as $LF2_{av}$ with respect to 2-composite faults.

The addressing orders of the aggressor cell "a" (or aggressor cells "a_1" and "a_2") and the victim cell "v" of a fault are essential since a test algorithm can detect the same fault if a<v and not detect if a>v, or vice versa. For a single-cell FP we will consider that the aggressor and victim cells coincide, i.e., a = v.

For a 2-composite fault FP_1*FP_2, we denote "a_1" as the aggressor cell of FP_1, "a_2" as the aggressor cell of FP_2, and "v" as the victim cell. All the possible orders of the aggressor-victim cells for 2-composite faults are listed below.

1) Unlinked 1-cell fault: a=v.
2) Unlinked 2-cell fault: a>v and a<v.
3) LF1: $a_1=a_2=v$.

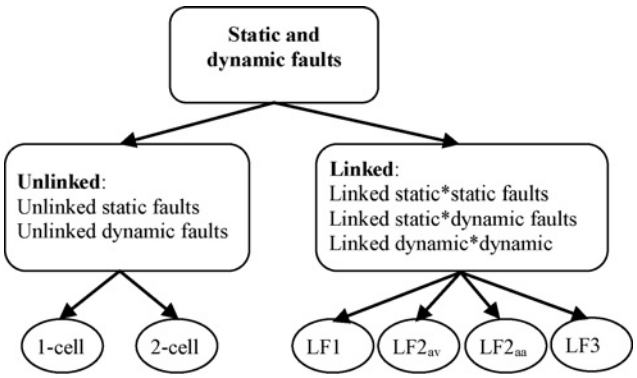

Fig. 2. Classification of static and dynamic faults.

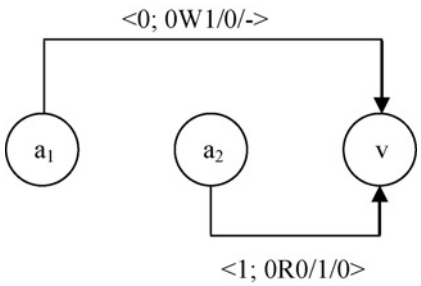

Fig. 3. Example of a linked static*static LF3 fault.

4) $LF2_{av}$: $a_1=a_2=a$, a>v, a<v.
5) $LF2_{aa}$: $a_1=a_2=a$, a>v, a<v.
6) LF3: $a_1>a_2>v$, $a_2>a_1>v$, $a_1>v>a_2$, $a_2>v>a_1$, $v>a_1>a_2$, $v>a_2>a_1$.

Fig. 3 shows an example of a linked static*static LF3 fault <0; 0W1/0/->*<1; 0R0/1/0>. This fault is composed by two unlinked static coupling faults having different aggressor cells. The fault is sensitized in two different conditions.

1) C_1: If aggressor cell "a_1" has value 0, victim cell "v" has value 0, then operation W1 applied to the victim cell will fail and the victim cell value will remain 0.
2) C_2: If aggressor cell "a_2" has value 1, victim cell "v" has value 0 then operation R0 applied to the victim cell will flip the victim cell value returning the correct result 0.

Fig. 4 shows an example of a linked static*dynamic $LF2_{av}$ fault <0W1R1; 1/0/->*<1W0/1/->. This fault is composed by an unlinked dynamic coupling fault and an unlinked static

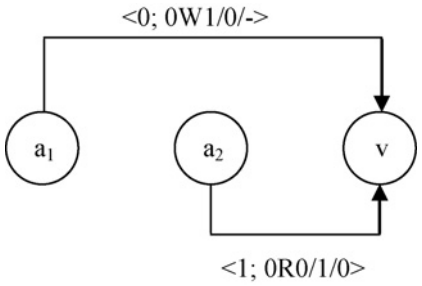

Fig. 4. Example of a linked static*dynamic $LF2_{av}$ fault.

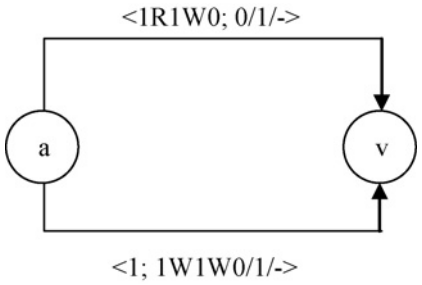

Fig. 5. Example of a linked dynamic*dynamic $LF2_{aa}$ fault.

single-cell fault. This fault is sensitized in two different conditions.

1) C_1: If aggressor cell "a" has value 0, victim cell "v" has value 1, then sequence of operations {W1, R1} applied to the aggressor cell will change the victim cell value from 1 to 0.
2) C_2: If victim cell "v" has value 1, then operation W0 applied to the victim cell will fail and the victim cell value will remain 1.

Fig. 5 shows an example of a linked dynamic*dynamic $LF2_{aa}$ fault <1R1W0; 0/1/->*<1; 1W1W0/1/->. This fault is composed by two unlinked dynamic coupling faults having the same aggressor cell. This fault is sensitized in two different conditions.

1) C_1: If aggressor cell "a" has value 1, victim cell "v" has value 0, then sequence of operations {R1, W0} applied to the aggressor cell will change the victim cell value from 0 to 1.
2) C_2: If aggressor cell "a" has value 1, victim cell "v" has value 1, then sequence of operations {W1, W0} applied to the victim cell will fail and the victim cell value will remain 1.

In [14], it was shown that the number of all linked static*static FPs is 600. The classes of linked static*dynamic and linked dynamic*dynamic faults contain a huge number of FPs compared with the class of linked static*static faults. For example, the class of linked dynamic*dynamic faults contains 12 459 FPs.

1) LF1 = FP_1*FP_2, FP_1 and FP_2 are single cell dynamic FPs. The number of single cell dynamic FPs is 30 (see [15]). Since FP_1*FP_2 = FP_2*FP_1, and there are 18 nonrealistic dynamic*dynamic LF1 faults (see Table II), then the number of FPs in LF1 is $30(30+1)/2-18 = 447$.

TABLE II
NONREALISTIC DYNAMIC*DYNAMIC FAULTS

Subclass	Nonrealistic Faults
LF1	<xWyRy/~y/~y>*<xWyRy/y/~y> <xWyRy/~y/~y>*<xWyRy/~y/y> <xWyRy/y/~y>*<xWyRy/~y/y> <xRxRx/~x/~x>*<xRxRx/x/~x> <xRxRx/~x/~x>*<xRxRx/~x/x> <xRxRx/x/~x>*<xRxRx/~x/x>
$LF2_{av}$	<x; yWzRz/~z/~z>*<yWzRz/z/~z> <x; yWzRz/~z/~z>*<yWzRz/~z/z> <x; yWzRz/z/~z>*<yWzRz/~z/~z> <x; yWzRz/z/~z>*<yWzRz/~z/z> <x; yWzRz/~z/z>*<yWzRz/~z/~z> <x; yWzRz/~z/z>*<yWzRz/z/~z> <x; zRzRz/~z/~z>*<zRzRz/z/~z> <x; zRzRz/~z/~z>*<zRzRz/~z/z> <x; zRzRz/z/~z>*<zRzRz/~z/~z> <x; zRzRz/z/~z>*<zRzRz/~z/z> <x; zRzRz/~z/z>*<zRzRz/~z/~z> <x; zRzRz/~z/z>*<zRzRz/z/~z>
$LF2_{aa}$	<x; yWzRz/~z/~z>*<x; yWzRz/z/~z> <x; yWzRz/~z/~z>*<x; yWzRz/~z/z> <x; yWzRz/z/~z>*<x; yWzRz/~z/z> <x; zRzRz/~z/~z>*<x; zRzRz/z/~z> <x; zRzRz/~z/~z>*<x; zRzRz/~z/z> <x; zRzRz/z/~z>*<x; zRzRz/~z/z>
LF3	<x; yWzRz/~z/~z>*<t; yWzRz/z/~z> <x; yWzRz/~z/~z>*<t; yWzRz/~z/z> <x; yWzRz/z/~z>*<t; yWzRz/~z/z> <x; zRzRz/~z/~z>*<t; zRzRz/z/~z> <x; zRzRz/~z/~z>*<t; zRzRz/~z/z> <x; zRzRz/z/~z>*<t; zRzRz/~z/z>

2) $LF2_{av}$ = FP_1*FP_2, FP_1 is a single cell and FP_2 is a two-cell dynamic FPs. The number of two-cell dynamic FPs is 96 (see [15]). Since there are 72 nonrealistic dynamic*dynamic $LF2_{av}$ faults (see Table II), then the number of FPs in $LF2_{av}$ is $30 \times 96 - 72 = 2808$.
3) $LF2_{aa}$ = FP_1*FP_2, FP_1 and FP_2 are two-cell dynamic FPs. Since FP_1 * FP_2 = FP2*FP_1, and there are 36 nonrealistic dynamic*dynamic $LF2_{aa}$ faults (see Table II), then the numbers of FPs in $LF2_{aa}$ is $96(96+1)/2-36 = 4620$.
4) LF3 = FP_1*FP_2, FP_1 and FP_2 are two-cell dynamic FPs. Since FP_1*FP_2 = FP_2*FP_1, and there are 72 nonrealistic dynamic*dynamic LF3 (see Table II), then the numbers of FPs in LF3 is $96(96+1)/2-72 = 4584$.
5) The class of linked dynamic*dynamic faults contains $447 + 2808 + 4620 + 4584 = 12\,459$ FPs.

A total of 198 dynamic*dynamic nonrealistic faults are described in Table II. Note that the symbol "~" used in Table II denotes logical negation, and x, y, z, t $\in \{0, 1\}$.

The same way, it is easy to calculate also the number of realistic FPs from the class of linked static*dynamic faults.

IV. EFFICIENT TEST ALGORITHM MARCH LSD

In this section, we propose an efficient test algorithm March LSD for detection of all 2-composite static and dynamic faults. Table III describes the structure of March LSD. It is a symmetric March test algorithm of complexity 75N, where N is the number of memory cells (words).

Fault coverage of this test algorithm was obtained by injecting all the considered faults into a memory model and running the test algorithm. The simulation results showed that all the faults were detected by March LSD. Since the number of the considered faults is very large only some of 2-composite faults and the corresponding operations for sensitizing and detecting them are shown in Table IV. If a fault is detected more than once, then the first detection scenario is presented in the table. If the sensitized FP is a static fault, then the column "Sensitization" contains one operation and if the FP is a dynamic fault, then accordingly there are two operations. In Table IV, $M_i(j)$ denotes jth operation of ith March element, $i \geq 0$, $j \geq 0$.

Let us consider fault LF3: <0W1R1; 1/0/->*<1R1R1; 0/1/-> ($a_1 < a_2 < v$) (see Table IV, #8). This fault is a LF3 fault, where "a_1" is the aggressor cell of FP <0W1R1; 1/0/->, "a_2" is the aggressor cell of FP <1R1R1; 0/1/-> and "v" is the victim cell for both FPs. The fault is sensitized by two sequential operations {R1, R1} which correspond to operations $\{M_1(16), M_1(17)\}$ in March LSD. The fault is detected by the first operation (operation R0) of March element M_1, i.e., $M_1(0)$.

Table IV shows also the sensitization and detection scenario for the same fault but with the given order of the aggressor and victim cells $a_2 < a_1 < v$ (see #9). We can see that these faults are sensitized and detected by different operations. This example shows that the order of the aggressor-victim cells of a fault is essential and all the possible orders of the aggressor-victim cells should be considered.

In Table V, we have compared test algorithm March LSD with other well-known March test algorithms. As we can see from the table, only March LSD covers 100% of all the considered faults. March LSD is an efficient March test algorithm since its complexity is more than the complexity of March MD2 only by 5N, while additionally March LSD detects all linked static*dynamic and dynamic*dynamic faults. Let us remember from the previous section that the number of only the linked dynamic*dynamic FPs is 12 459.

V. Structure-Oriented Method for Generation of March Test Algorithms

Different methods for March test algorithm generation were proposed in the past [19]–[22]. Some of them proposed to perform an exhaustive search of March test algorithms starting from a shortest March test algorithm and then increasing its length. The advantage of such flow is that the found March test algorithm is minimal while the main disadvantage of it is the huge amount of runtime (days, months).

The other approach is based on heuristic algorithms, i.e., searching March test algorithms nonexhaustively taking into account some conditions extracted from the fault types. The advantage of such flow is the speed, i.e., March test algorithms can be found in seconds or milliseconds. Unfortunately, such a flow has a disadvantage that is the found March test algorithm can be "inefficient," i.e., far with its complexity from the complexity of the minimal March test algorithm.

The mentioned above methods are not applicable for generating a March test algorithm for detection of all 2-composite static and dynamic faults. The reason is that the exhaustive search to detect a minimal March test algorithm is time-consuming, while the existing heuristic methods either cannot generate the required March test algorithm (since they support only static faults or only unlinked static and dynamic faults, and so on) or generate inefficient March test algorithms.

TABLE III
TEST ALGORITHM MARCH LSD (75N)

March Element	Address Direction	Operations
M_0	⇳	W0
M_1	⇑	R0, W1, R1, W1, W1, R1, W1, W0, R0, W1, W1, R1, W0, W1, R1, W1, R1, R1
M_2	⇑	R1, W1, W1, R1, W1, W0, R0, W1, W1, R1, W0, W1, R1, W1, R1, R1, R1, W0
M_3	⇑	R0
M_4	⇓	R0, W0, W0, R0, W0, W1, R1, W0, W0, R0, W1, W0, R0, W0, R0, R0, R0, W1
M_5	⇓	R1, W0, R0, W0, W0, R0, W0, W1, R1, W0, W0, R0, W1, W0, R0, W0, R0, R0
M_6	⇓	R0

The structure-oriented method proposed in this paper is based on searching of symmetric March test algorithms, where the same March element is usually repeated with opposite addressing orders or with opposite background patterns. For example, in March C-: {M_0: ⇳(W0); M_1: ⇑(R0, W1); M_2: ⇑(R1, W0); M_3: ⇓(R0, W1); M_4: ⇓(R1, W0); M_5: ⇳(R0)}, March elements M_1 and M_3 have the same sequence of operations (with the same background patterns) but opposite addressing orders, while March elements M_1 and M_2 have the same addressing order, the same sequence of operations but opposite background patterns.

The idea of the proposed method is the following: since usually for detection of a special set of faults most of the well-known March test algorithms have a symmetric structure (note that all March test algorithms listed in Table V are symmetric), then we will search only a specific class of symmetric March test algorithms. In this case instead of searching the whole March test algorithm, it will be enough to search only the unique instances of March elements.

Let us assume that $S(x, y) = O_1, O_2, \ldots, O_k$, where $O_j \in \{R0, R1, W0, W1\}$, $1 \leq j \leq k$. The sequence S(x, y) satisfies the following conditions.

1) If O_j = W0 or R0, then $O_{j+1} \neq R1$.
2) If O_j = W1 or R1, then $O_{j+1} \neq R0$.
3) O_1 = Rx.
4) O_k = Wy or Ry.

Let us consider the following two structures of March test algorithms:

MT_1=⇳(W0);⇑(S_1);⇑(~S_1);[⇳(R0);]⇓(S_2);⇓(~S_2);⇳(R0)
MT_2=⇳(W0);⇑(S_1);⇑(~S_2);[⇳(R0);]⇓(S_2);⇓(~S_1);⇳(R0)
where:

1) S_1, $S_2 \in S(0, 1)$;
2) [⇳ (R0);]—means that the March test can contain this March element optionally.

TABLE IV
SOME OF 2-COMPOSITE FAULTS DETECTED BY TEST ALGORITHM MARCH LSD

#	Fault	Sensitization	Detection
1	Unlinked static fault: <0W1; 0/1/-> (a<v)	$M_1(1)$	$M_1(0)$
2	Unlinked dynamic fault: <1: 0W0R0/1/0> (v<a)	$M_4(13)$, $M_4(14)$	$M_4(15)$
3	LF1: <1W1/0/->*<0W1/0/->	$M_1(1)$	$M_1(2)$
4	LF2av: <0W1R1; 0/1/->*<0W1W0/1/-> (a<v)	$M_1(1)$, $M_1(2)$	$M_1(0)$
5	LF2av: <0W1R1; 0/1/->*<0W1W0/1/-> (v<a)	$M_2(11)$, $M_2(12)$	$M_3(0)$
6	LF2aa: <0R0R0; 1/0/->*<1: 1R1R1/0/1> (a<v)	$M_1(16)$, $M_1(17)$	$M_2(0)$
7	LF2aa: <0R0R0; 1/0/->*<1: 1R1R1/0/1> (v<a)	$M_2(14)$, $M_2(15)$	$M_2(16)$
8	LF3: <0W1R1; 1/0/->*<1R1R1; 0/1/-> ($a_1<a_2<v$)	$M_1(16)$, $M_1(17)$	$M_1(0)$
9	LF3: <0W1R1; 1/0/->*<1R1R1; 0/1/-> ($a_2<a_1<v$)	$M_2(11)$, $M_2(12)$	$M_2(0)$
10	LF3: <1W1; 0/1/->*<0: 0W1W1/0/-> ($a_1<v<a_2$)	$M_1(3)$	$M_1(0)$
11	LF3: <1W1; 0/1/->*<0: 0W1W1/0/-> ($a_2<v<a_1$)	$M_2(7)$, $M_2(8)$	$M_2(9)$
12	LF3: <0R0R0; 0/1/->*<0R0R0; 1/0/-> ($v<a_1<a_2$)	$M_4(14)$, $M_4(15)$	$M_4(0)$
13	LF3: <0R0R0; 0/1/->*<0R0R0; 1/0/-> ($v<a_2<a_1$)	$M_5(16)$, $M_5(17)$	$M_5(0)$

TABLE V
COMPARISON OF TEST ALGORITHM MARCH LSD WITH OTHER MARCH TEST ALGORITHMS

March Test Algorithm	Complexity	Unlinked Static Faults	Linked Static*Static Faults	Unlinked Dynamic Faults	Linked Static*Dynamic Faults	Linked Dynamic*Dynamic Faults
March C- [17]	10N	–	–	–	–	–
March MSS [13]	18N	+	–	–	–	–
March SS [1]	22N	+	–	–	–	–
March AB [18]	22N	+	–	–	–	–
March MSL [14]	23N	+	+	–	–	–
March SL [3]	41N	+	+	–	–	–
March MD2 [15]	70N	+	+	+	–	–
March LSD	75N	+	+	+	+	+

For example, March MSS can be obtained from MT_1 or MT_2, if $S_1 = S_2 = \{R0, R0, W1, W1\}$ and the optional March element $\Updownarrow(R0)$ is absent.

Fig. 6 describes the proposed method for March test algorithm generation. The flow starts to construct sequences S_1 and S_2 (of lengths L_1 and L_2) from class S(0, 1). Note that $\{R0, W1\}$ is the shortest element of class S(0, 1). Each time next S_1 or S_2 is constructed from class S(0, 1). If there is no any other sequence of length L_1 (or L_2) to generate, then the L_1 (respectively, L_2) is increased by 1 and a new sequence of length L_1 (respectively, L_2) is constructed. L_{MAX} is a user specified value, which limits the complexity of the searched March test by number $2*L_{MAX}+3$ (see structures MT_1 and MT_2). If the flow responds "No solution," then the reason is the small value of L_{MAX}. This means that L_{MAX} should be increased and run the flow again.

In [13]–[15], several necessary and/or sufficient conditions are described for static and dynamic fault detection. Using these conditions, one can conclude that the given March test algorithm does not detect the given set of faults without running the March test algorithm on the memory model with injected faults (e.g., if a March test does not contain two sequential R0 operations then it does not detect DRDF fault, see Table I). This speeds up the March test algorithm generation flow essentially. Thus, a March test algorithm is generated only if S_1 and S_2 satisfy the given conditions (see Fig. 6). The generated March test algorithm is run on the memory model with injected faults to find out whether it detects all the considered faults or not. We have taken into account the whole space of 2-composite static and dynamic faults according to the classification in Fig. 2. If the current March test algorithm detects all the considered faults, then we consider that the March test algorithm is found, otherwise the next March test algorithm is generated. If structure MT_1 is fully used, then structure MT_2 can be used for March test algorithm generation. We consider only the structures MT_1 and MT_2 since they give a satisfactory solution for the problem considered in this paper. Besides MT_1 and MT_2, other structures of March test algorithms also can be used in the proposed method.

Table VI shows the March test algorithms that we have obtained applying the proposed method. Test algorithms March MSS and March MD2 are minimal correspondingly for detection of all unlinked static and all unlinked dynamic faults.

The minimal test algorithm for detection of all linked and unlinked static faults is March MSL of complexity 23N (see Table V), while we have obtained a new test algorithm March SL24 of complexity 24N. Finally, for detection of all linked and unlinked static and dynamic faults, we have obtained test algorithm March LSD. From Table VI, it can be noted

TABLE VI
MARCH TEST ALGORITHMS GENERATED BY THE PROPOSED METHOD

Faults	March Test Algorithm	Complexity	Minimal	Structure	Time
All unlinked static faults	March MSS [13]: ⇕(W0); ⇑(R0, R0, W1, W1); ⇑(R1, R1, W0, W0); ⇓(R0, R0, W1, W1); ⇓(R1, R1, W0, W0); ⇕(R0)	18N	Yes	MT1	0.12 s
All linked and unlinked static faults	March SL24: ⇑(W0); ⇑(R0, W1, W0, W0, R0, W1, R1); ⇑(R1, W0, W0, R0); ⇓(R0, W1, W1, R1); ⇓(R1, W0, W1, W1, R1, W0, R0); ⇓(R0)	24N	No	MT2	0.74 s
All unlinked dynamic faults	March MD2 [15]: ⇕(W0); ⇑(R0, W1, W1, R1, W1, W1, R1, W0, W0, R0, W0, W0, R0, W0, W1, W0, W1); ⇑(R1, W0, W0, R0, W0, W0, R0, W1, W1, R1, W1, W1, R1, W1, W0, W1, W0); ⇓(R0, W1, R1, W1, R1, R1, R1, W0, R0, W0, R0, R0, R0, W0, W1, W0, W1); ⇓(R1, W0, R0, W0, R0, R0, R0, W1, R1, W1, R1, R1, R1, W1, W0, W1, W0); ⇕(R0)	70N	Yes	MT1	15 s
All linked and unlinked static and dynamic faults	March LSD (see Table III)	75N	Unknown	MT2	198 s

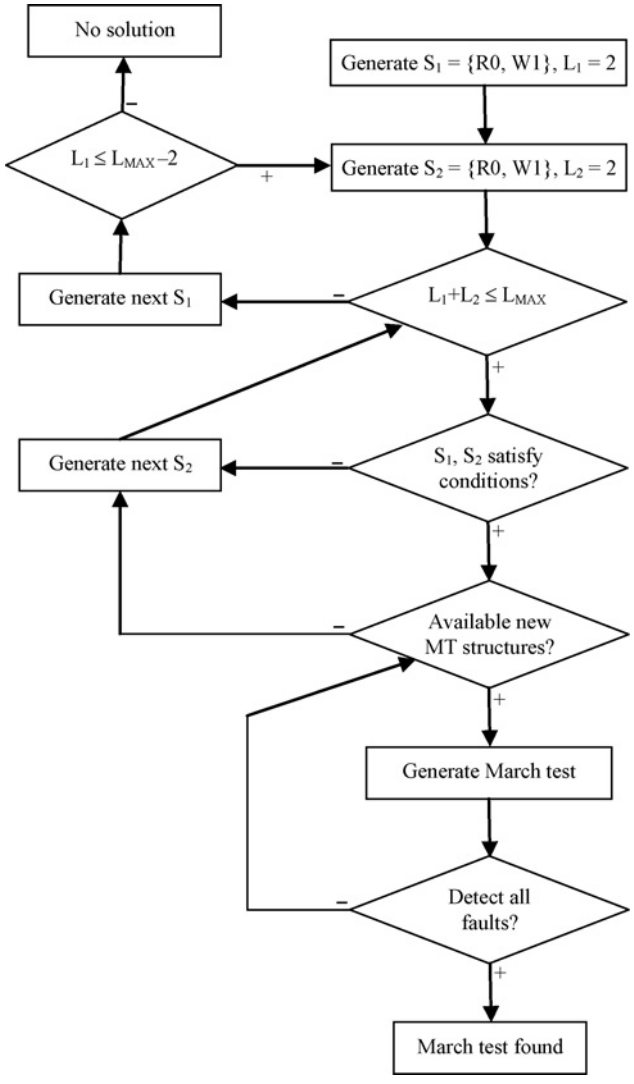

Fig. 6. March test algorithm generation method.

that it took only 198 s to generate March LSD. Currently it is unknown whether March LSD is a minimal March test algorithm or not with respect to the considered faults.

The experiments show that the proposed method is an efficient method since the generated March test algorithms are either minimal or very close to the minimal one, as well as the generation times are acceptable (e.g., generation time for March LSD is 198 s). Since the generation of a test algorithm is a one-time task, then we assume that 198 s is an acceptable time (even one day, or one week can be acceptable). Thus, we can conclude that the method allows easily generating of complex March test algorithms with symmetric structures.

Complete comparison with other methods is impossible since none of them has 100% fault coverage with respect to the considered faults. A per our knowledge, only the proposed method can generate March LSD.

The test algorithm library of Synopsys STAR Memory System (see [23]) contains all test algorithms listed in Table V, including March LSD. These algorithms are widely used in Synopsys built-in self-test (BIST) circuits, as well as in debug and diagnosis tools. The BIST runtimes for these test algorithms are in the acceptable range. We have done several experiments on test chips and the results show that March LSD has the highest fault coverage among test algorithms described in Tables V and VI.

VI. CONCLUSION

Based on the notion of 2-composite faults, a classification for linked and unlinked static and two-operation dynamic faults is introduced. An efficient test algorithm March LSD of complexity 75N is proposed for detection of the considered faults. A new structure-oriented method for generation of efficient symmetric March test algorithms detecting different combinations (links) of static and dynamic faults is described.

In the future, we will explore new effective March test structures to enlarge the space of the considered faults (e.g.,

three-operation dynamic faults). Also, we are planning either to prove that test algorithm March LSD is minimal or to improve the method to obtain a minimal March test algorithm for detection of all linked and unlinked static and dynamic faults.

REFERENCES

[1] S. Hamdioui, A. J. van de Goor, and M. Rodgers, "March SS: A test for all static simple RAM faults," in *Proc. IEEE Workshop MTDT*, Jul. 2002, pp. 95–100.

[2] S. Hamdioui, G. N. Gaydadjiev, and A. J. van de Goor, "A fault primitive based analysis of dynamic memory faults," in *Proc. IEEE Workshop Circuits Syst. Signal Process.*, Nov. 2003, pp. 84–89.

[3] S. Hamdioui, A. J. van de Goor, and M. Rodgers, "Linked faults in random access memories: Concept, fault models, test algorithms, and industrial results," *IEEE Trans. Comput.-Aided Des.*, vol. 23, no. 5, pp. 737–756, May 2004.

[4] S. Hamdioui, R. Wadsworth, J. D. Reyes, and A. J. van de Goor, "Importance of dynamic faults for new SRAM technologies," in *Proc. IEEE Eur. Test Workshop*, May 2003, pp. 29–34.

[5] L. Dilillo, P. Girard, S. Pravossoudovitch, and A. Virazel, "Dynamic read destructive fault in embedded-SRAMs: Analysis and March test solution," in *Proc. IEEE Eur. Test Symp.*, May 2004, pp. 140–145.

[6] M. Azimane, A. K. Majhi, G. Gronthoud, M. Lousberg, S. Eichenberger, and A. L. Ruiz, "A new algorithm for dynamic faults detection in RAMs," in *Proc. IEEE VLSI Test Symp.*, May 2005, pp. 177–182.

[7] A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, and M. Bastian, "Slow write driver faults in 65 nm SRAM technology: Analysis and March test solution," in *Proc. IEEE DATE*, Apr. 2007, pp. 528–533.

[8] A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, and M. Bastian, "Dynamic two-cell incorrect read fault due to resistive-open defects in the sense amplifiers of SRAMs," in *Proc. IEEE Eur. Test Symp.*, May 2007, pp. 97–104.

[9] Z. Al-Ars, S. Hamdioui, and G. Gaydadjiev, "Manifestation of precharge faults in high speed DRAM devices," in *Proc. IEEE DDECS*, Apr. 2007, pp. 179–184.

[10] A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, and A. Virazel, *Advanced Test Methods for SRAMs: Effective Solutions for Dynamic Fault Detection in Nanoscaled Technologies*. Berlin, Germany: Springer, 2009.

[11] A. J. van de Goor, S. Hamdioui, G. Gaydadjiev, and Z. Al-Ars, "New algorithms for address decoder delay faults and bit line imbalance faults," in *Proc. IEEE Asian Test Symp.*, Nov. 2009, pp. 391–396.

[12] R. A. Fonseca, L. Dilillo, A. Bosio, P. Girard, S. Pravossoudovitch, A. Virazel, and N. Badereddine, "Impact of resistive-bridging defects in SRAM core-cell," in *Proc. IEEE Int. Symp. Electron. Des. Test Applicat.*, Jan. 2010, pp. 265–269.

[13] G. Harutunyan, V. A. Vardanian, and Y. Zorian, "Minimal March tests for unlinked static faults in random access memories," in *Proc. IEEE VLSI Test Symp.*, May 2005, pp. 53–59.

[14] G. Harutunyan, V. A. Vardanian, and Y. Zorian, "Minimal March test algorithm for detection of linked static faults in random access memories," in *Proc. IEEE VLSI Test Symp.*, Apr.–May 2006, pp. 120–125.

[15] G. Harutunyan, V. A. Vardanian, and Y. Zorian, "Minimal March tests for detection of dynamic faults in random access memories," *J. Electron. Testing: Theory Applicat.*, vol. 23, no. 1, pp. 55–74, Feb. 2007.

[16] S. Hamdioui, Z. Al-Ars, and A. J. van de Goor, "Testing static and dynamic faults in random access memories," in *Proc. IEEE VLSI Test Symp.*, Apr.–May 2002, pp. 395–400.

[17] A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*. New York: Wiley, 1991.

[18] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, and P. Prinetto, "March AB, March AB1: New March tests for unlinked dynamic memory faults," in *Proc. IEEE Int. Test Conf.*, Nov. 2005, pp. 834–841.

[19] S. M. Al-Harbi and S. K. Gupta, "An efficient methodology for generating optimal and uniform March tests," in *Proc. IEEE VLSI Test Symp.*, Apr.–May 2001, pp. 231–237.

[20] C.-F. Wu, C.-T. Huang, K.-L. Cheng, and C.-W. Wu, "Fault simulation and test algorithm generation for random access memories," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 4, pp. 480–490, Apr. 2002.

[21] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, and P. Prinetto, "Automatic March tests generation for static and dynamic faults in SRAMs," in *Proc. IEEE Eur. Test Symp.*, May 2005, pp. 122–127.

[22] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, and P. Prinetto, "March test generation revealed," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1704–1713, Dec. 2008.

[23] K. Darbinyan, G. Harutyunyan, S. Shoukourian, V. Vardanian, and Y. Zorian, "A robust solution for embedded memory test and repair," in *Proc. IEEE Asian Test Symp.*, Nov. 2011, pp. 461–462.

[24] A. J. van de Goor, G. N. Gaydadjiev, V. G. Mikitjuk, and V. N. Yarmolik, "March LR: A test for realistic linked faults," in *Proc. IEEE VLSI Test Symp.*, Apr.–May 1996, pp. 272–280.

Gurgen Harutyunyan received the Bachelors, Masters, and Ph.D. degrees, all from Yerevan State University, Yerevan, Armenia, in 2003, 2005, and 2008, respectively.

Since 2010, he has been with Synopsys, Inc., Yerevan, as a Senior Research and Development Engineer with the Department of Embedded Test and Repair. He has published more than 20 papers in the area of memory testing. His current research interests include fault detection, diagnosis, and location of faults in memory devices.

Samvel Shoukourian received the D.Sc. degree in physics and mathematics.

He is currently the Scientific Leader with the IT Educational and Research Center, Yerevan State University (YSU), Yerevan, Armenia. In 1996, he was a Full Member of the National Academy of Sciences of Armenia, Yerevan. Since 1994, simultaneously with the work at YSU, he has been a Chief Scientific Advisor and the Development Director with different international organizations/companies and since 2010, he has been a Senior Manager with the Department of Embedded Test and Repair, Synopsys, Inc., Yerevan. He has authored more than 80 papers, and holds Russian and U.S. patents. The main topics of his current research interests include testing of electronic devices and systems, formal models of distributed systems, architecture of computer systems, architectures for distance learning, and multimedia-based eLearning.

Valery Vardanian received the M.Sc. degree (hons.) from Yerevan State University (YSU), Yerevan, Armenia, and the Ph.D. degree from Moscow State University, Moscow, Russia.

He was with the National Academy of Sciences of Armenia, Yerevan, as a Senior Research Associate. Currently, he is with Synopsys, Inc., Yerevan, as a Manager with the Embedded Test and Repair Methodology Group. He also teaches at YSU as an Adjunct Professor (out of campus). His current research interests include algorithmic issues of memory test, diagnosis and repair, design for testability, and inductive fault analysis. He has published more than 70 papers and 3 U.S. patents on testing, diagnosis, and repair of memories, as well as on complexity issues of testing digital circuits.

Yervant Zorian (M'83–F'99) received the M.Sc. degree from the University of Southern California, Los Angeles, and the Ph.D. degree from McGill University, Montréal, QC, Canada.

He is currently a Chief Architect with Synopsys, Inc., Synopsys, Inc., Mountain View, CA, and an Adjunct Professor with the University of British Columbia, Vancouver, BC, Canada. Previously, he was the Vice President and Chief Scientist with Virage Logic, Yerevan, a Distinguished Member of the Technical Staff with AT&T Bell Laboratories, Murray Hill, NJ, and the Chief Technology Advisor with Logic Vision, San Jose, CA. He has authored over 300 papers and four books, and holds 29 U.S. patents.

Dr. Zorian has served as the IEEE Computer Society Vice President for conferences and tutorials, the Vice President for technical activities, the Chair of the IEEE Test Technology Technical Council, and the Editor-in-Chief of the IEEE DESIGN AND TEST OF COMPUTERS. He received numerous Best Paper Awards and the Bell Laboratories Research and Development Achievement Award. He was selected by *EE Times* among the top 13 influencers on the semiconductor industry. He was the recipient of the prestigious IEEE Industrial Pioneer Award in 2005.