Mathematical Problems of Computer Science 30, 18–24, 2008.

Efficient March-Like Algorithm for Detection of All Two-Operation Dynamic Faults from Subclass S_{av}

H. Avetisyan[†], G. Harutunyan[‡], V.A. Vardanian[‡]

[†] Russian-Armenian State University
e-mail: hamazasp_avetisyan@yahoo.com
[‡]Virage Logic
e-mail: {gurgen.harutyunyan, valery.vardanian}@viragelogic.com

Abstract

This paper introduces an efficient March-like algorithm for detection of the well known class S_{av} of dynamic faults. S_{av} is the subclass of all two-operation dynamic functional fault models that are sensitized by means of applying two consecutive operations, one applied on the aggressor cell and the second operation applied on the victim cell. Earlier, only subclasses S_{aa} and S_{vv} were considered by a few authors when both sensitizing operations were applied either on the aggressor or victim cell, and March algorithms were developed by them. Subclasses S_{av} and S_{va} were not considered due to their complexity. A larger class of March-like algorithms has to be considered for detection of those subclasses since March algorithms cannot detect them. It is shown that $392N$ operations are sufficient for detection of faults from S_{av}.

1 Introduction

New memory technologies and processes introduce new, previously unknown, defects that significantly impact on the defect-per-million (DPM) level and yield. Conventional memory test algorithms detect memory failures to determine whether a chip is defective or not. To optimize a fault detection algorithm for a given memory, there are two ways for enhancement: either leveraging memory design information at the layout level to perform Inductive Fault Analysis (IFA) and generate a corresponding fault detection algorithm, or using real chips to leverage the failure history of a given cell design and process technology. This information helps generate a dedicated test algorithm. March test algorithms (see [1]) are widely used for fault detection in memories due to their effectiveness, simplicity and linear length with respect to the memory size. An important requirement for reduction of the testing time is to use minimal test algorithms. Predefined test algorithms are not always sufficient to detect all memory defects because subtle process variations cannot be predicted and accounted for ahead of time. The quality of a test algorithm is characterized by its fault coverage with respect to the functional fault models (FFM) [1] defined on the basis of fault primitives (FP) (see [2], [3]). In addition to the previously known static FFMs [1], in [4]-[7], new dynamic FPs and FFMs were introduced based on IFA and their existence in current memory chips was validated experimentally. Dynamic faults were shown to be realistic in the absence of the

static faults. The class of dynamic faults has been recently shown to be an important class of faults for the new technologies of Random Access Memories (RAM) with significant impact on DPM levels. Very little research has been done in the design of memory test algorithms targeting dynamic faults. In [4], two March test algorithms RAW1 of length $13N$ and RAW of length $26N$, N is the number of memory cells, for classes of single-cell and two-cell dynamic faults, respectively, were proposed. In [8] two March test algorithms, March AB1 of length $11N$ and March AB of length $22N$, for classes of single-cell and two-cell dynamic faults, respectively, were proposed improving the length of those proposed in [4]. In [9], a March test algorithm of length $100N$ was proposed for detection of two-cell dynamic faults with two fault-sensitizing operations both applied on the victim or aggressor cells. Note that the proposed test appeared to be redundant, i.e. an operation can be removed from the test without any impact on the fault coverage. In [10], [11] the authors considered the class of two-operation all single-cell dynamic faults, as well as the subclass of two-operation two-cell dynamic faults where both of the sensitizing operations are applied either on the aggressor cell or the victim cell. For the first time, they have proposed minimal March test algorithms for detection of all two-operation single-cell dynamic faults in Random Access Memories, as well as a minimal March test algorithm of length $70N$ for detection of two-cell dynamic faults with two fault-sensitizing operations both applied on the victim or aggressor cells. The March test algorithm of length $100N$ proposed in [9] was reduced by $30N$ operations. Additionally, it has been shown that the $70N$ test detects also all realistic unlinked and linked static faults. In this paper, for the first time in the literature, we considered the subclass of two-operation dynamic faults when the first sensitizing operation is applied on the aggressor cell and the second sensitizing operation - on the victim cell. This subclass of dynamic faults cannot be detected by March tests. Thus, we enlarged the latter by considering the class of March-like algorithms. For the case of fixed positions (physical addresses) of the aggressor and victim cells, we proposed a minimal March-like algorithm of complexity $98N$ to detect all faults from the subclass S_{av}. Thus, for all possibilities of the position of the aggressor and victim cells, when only the realistic cases are considered, i.e. the victim cell is physically adjacent to the aggressor cell, the complexity of the algorithm becomes $4 \times 98N = 392N$. Thus, if we consider Type 2 neighborhood, the complexity of the proposed algorithm will become $8 \times 98N = 784N$. Although the coefficient of N seems to be very large for practical tests, however they are the first results for the mentioned subclass S_{av} and can be reduced further if a more efficient way of running the aggressor-victim pair of cells over the memory could be found. We hope to do it in the future. The results obtained for the subclass S_{av} can be easily modified with slight changes to be valid also for the subclass S_{va}.

2 Notations and Definitions

In Table I, you can see subclass S_{av} [4] of dynamic faults, where (O stands for R or W). This subclass assumes that operation on the victim cell is performed after applying the first sensitizing operation on the aggressor cell. It should be noted that we can not use March tests for this class of functional fault models (FFMs) because we need to do an operation on the victim cell just after the operation applied on the aggressor cell that is not always possible because of the possibility of memory scrambling (March tests assume logical representation of memory).

TABLE I
FAULT CLASS S_{av}

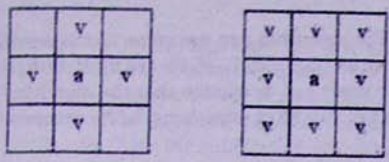
FFM	Fault Primitive
dCFrd	$< xOy, 0r0/1/1 >, < xOy, 1r1/0/0 >$
dCFdrd	$< xOy, 0r0/1/0 >, < xOy, 1r1/0/1 >$
dCFir	$< xOy, 0r0/0/1 >, < xOy, 1r1/1/0 >$
dCFtr	$< xOy, 0w1/0/- >, < xOy, 1w0/1/- >$
dCFwd	$< xOy, 0w0/1/- >, < xOy, 1w1/0/- >$

Here we need physical representation of the memory and the actual physical addresses of the aggressor and victim cells should be taken into consideration that assume the knowledge of at least the address scrambling (see e.g. [12]) of the memory. Thus, after the fault sensitization we have to apply another Read operation on the victim cell in order to detect the FFM. So we have to enlarge the notation of the March test notation, thus introducing March-like tests (see e.g. [13], [14]) to be able to detect FFMs from subclasses S_{av} and S_{va}. In this paper, we restrict ourselves by considering only the subclass S_{av}. However, it is easy to consider, in a similar way, the subclass S_{va} as well. For a March-like algorithm [13], [14], $W_v x$ (respectively, $W_a x$) is the operation of writing $x \in \{0, 1\}$ only to the victim (respectively, aggressor) cell, $R_v x$ (respectively, $R_a x$) is the read operation performed only to the victim (respectively, aggressor) cell with expected value x. In the sequel, we denote R_v as R_{vb} in expressions if R_v is the first operation in the corresponding March element, otherwise we use R_{va}.

3 Minimal March-Like Algorithm For Class S_{av}: Fixed Agressor-Victim Case

In Table II, a minimal March-like algorithm March DAV for detection of all faults from class S_{av} is given with the fixed positions (addresses) of the aggressor and victim cells. Further on, the aggressor cell has to be considered as the base-cell and run over all N words (or cells) of the memory. For each fixed aggressor cell, we assume that the realistic ones are those words (or cells) that are physically adjacent to the aggressor cell. In the literature, only two types of neighborhoods are mainly considered: Type 1 and Type 2 (see Fig. 1). Type 1 neighborhood is the set of words (or cells) located physically at the North, South, West and East of the aggressor word (or cell). Type 2 neighborhood includes, in addition to the North, South, West and East words (or cells) of the Type 1 neighborhood, the words (or cells) that are located diagonally with respect to the base (aggressor) cell at the left and right sides of the North and South cells.

For construction of the minimal test March DAV (Dynamic test for S_{av} with fixed Aggressor-Victim pair) we construct the corresponding graph-theoretical model consisting of four vertices {00, 01, 10, 11} corresponding to each of the four possible states of the pair of aggressor-victim cells with edges corresponding to all passes from a state to another. Then, two cells (i. e. two states) are connected with an arrow if there is a transition from one state $v1$ to another $v2$, i.e. we can pass from one state to another by applying the corresponding sensitizing operations.



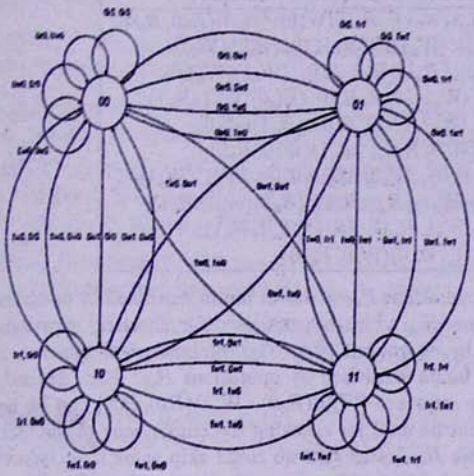


Figure 2: The Eulerian graph for March DAV

In Figure 2, we can see the Eulerian graph for March DAV where, for example, we can pass from state "10" to "01" by applying operations $1w0$, $0w1$. In order to construct a full detection test for the subclass S_{av} we have to find an Eulerian path in the corresponding Eulerian graph (See Fig.2) that passes through each edge of the graph once and only once. The Eulerian path, thus, sensitizes all dynamic faults from subclass S_{av} once and only once and performs the corresponding Read operation on the victim cell in order to detect the sensitized dynamic fault corresponding to the label of the edge in the Eulerian path. Thus, the test should be minimal since it sensitizes each fault once and only once and applies a Read operation after sensitization of the corresponding fault.

Theorem: The minimal algorithm for the class S_{av} with fixed positions (addresses) of the aggressor and victim words (or cells) should have at least 98 operations. Δ

Proof: Note that after sensitization of a fault, i.e. after applying the first sensitizing operation on the aggressor cell, and the second one - on the victim cell, we have to Read the value of the victim cell in order to be able to detect the sensitized FFM. We will name the process of reading the value of the victim cell after its sensitization, as "checking the consistency of the victim cell". Let us show that the minimal algorithm for the class S_{av} should have at least $12R_a$, $25W_a$, $25W_v$ and $36R_v$ operations. First of all, we must initialize the aggressor and victim cells (at least two Write operations). It is easy to check by observing Table II that to sensitize each fault in S_{av} we must perform at least $12R_a$, $24W_a$, $24W_v$ and $12R_{av}$ operations. Then the minimal algorithm must check consistency of the victim

cell. To do this it must use R_v operations. The algorithm can use some of the sensitizing operations R_{vb} for this purpose. E.g., instead of $(R_a0)(W_v1, R_v1)$, $(R_a0)(R_v1, R_v1)$ the sequence $(R_a0)(W_v1)$, $(R_a0)(R_v1, R_v1)$ can be used. Let us assume that the algorithm uses p operations R_{vb} to check the value of the victim cell after sensitizing faults by operations W_v.

TABLE II
MINIMAL ALGORITHM MARCH DAV FOR FAULT CLASS S_{av} WITH FIXED AGGRESSOR AND VICTIM CELLS

$(W_a0)(W_v0)$; $(R_a0)(W_v1)$; $(R_a0)(R_v1, R_v1)$; $(R_a0)(W_v0)$; $(R_a0)(R_v0, R_v0)$; $(W_a0)(W_v1)$; $(W_a0)(R_v1, R_v1)$; $(W_a1)(W_v0, R_v0)$; $(R_a1)(W_v1)$; $(R_a1)(R_v1, R_v1)$; $(R_a1)(W_v0)$; $(R_a1)(R_v0, R_v0)$; $(W_a1)(W_v1)$; $(W_a1)(R_v1, R_v1)$; $(W_v1)(W_v0)$; $(W_a1)(R_v0, R_v0)$; $(W_a0)(W_v1, R_v1)$; $(W_a0)(W_v0)$; $(W_a0)(R_v0)$; $(W_a1)(R_v0, R_v0)$; $(R_a1)(W_v0)$; $(W_a0)(R_v0, R_v0)$; $(R_a0)(W_v0, R_v0)$; $(W_a1)(W_v0, R_v0)$; $(W_a1), (W_v0, R_v0)$; $(W_a0)(W_v0, R_v0)$; $(W_a0)(W_v0, R_v0)$; $(W_a1)(W_v1)$; $(W_a0)(R_v1, R_v1)$; $(R_a0)(W_v1)$; $(W_a1)(R_v1, R_v1)$; $(R_a1)(W_v1, R_v1)$; $(W_a0)(W_v1, R_v1)$; $(W_a0)(W_v1, R_v1)$; $(W_a1)(W_v1, R_v1)$; $(W_a1)(W_v1, R_v1)$; $(W_a0)(W_v0, R_v0)$;

In this case we need at least $24-p$ operations R_{ve} to detect faults sensitized by operations W_v. Besides, we got $12-p$ operations R_{vb} which are not used for checking consistency of the victim after sensitizing faults by operations W_v. The minimal algorithm can use operations R_{vb} for detection of some faults sensitized by operations R_{vb}. E.g., instead of $(W_a0)(R_v0, R_v0)$, $(W_a1)(R_v0, R_v0)$ the sequence $(W_a0)(R_v0)$, $(W_a1)(R_v0, R_v0)$ can be used as well. Note that the same R_{vb} can not be used for checking the consistency of the victim after sensitizing operation W_v and after R_{vb} (otherwise we could skip some fault behavior or got redundant operations), so there should be $12-p$ remained operations R_{vb} that are not used for checking consistency of the victim cell after sensitizing by operations W_v, but are used for checking the consistency of victims after sensitizing by operations R_{vb}. So p operations R_{vb} are not used for checking consistency of the victim cell after sensitization by operations R_{vb}. Here we need additional p operations R_{ve} to detect the faults sensitized by operations R_{vb}. Eventually, we need $(24-p)+p=24$ operations R_{ve}. Taking into account the initialization operations W_v and W_a, we can conclude that any algorithm detecting all faults from S_{av} should do at least $12R_a$, $25W_a$, $25W_v$, $12R_{vb}$ and $24R_{ve}$ operations, i.e., at least 98 operations. Taking into account this theorem we can conclude that March DAV is a minimal March-like algorithm for detection of all faults from S_{av} in the fixed aggressor-victim case. In order to check faults from class S_{av} in the memory, we must apply March DAV considering each cell as an aggressor cell and consider its 4 neighboring cells from the Type 1 neighborhood as victim cells. So we can check faults from class S_{av} using March DAV applying it $4N$ times (4 times for each cell), where N is the number of memory bits. The total complexity of our proposed algorithm becomes $392N$.

4 Consclusions

In this paper, we proposed an algorithm that for the fixed aggressor-victim pair applies a minimal March-like algorithm for detection of all two-operation dynamic functional fault models from subclass S_{av}, i.e., two-operation dynamic faults with the first sensitizing op-

eration applied on the aggressor cell, and the second sensitizing operation applied on the victim cell. Still remains open the case when the aggressor and victim cells are not fixed. We have to develop a minimal or an efficient test for detection of S_{av} by finding an efficient way of running the aggressor cell over the memory and considering all possible positions of the victim cell. Also the subclass of dynamic FFMs S_{va} is planned to be considered in the future work.

References

[1] A. J. van de Goor, "Testing semiconductor memories: Theory and Practice", *John Wiley and Sons*, 1991.

[2] S. Hamdioui, A.J. van de Goor, M. Rodgers, "March SS: a test for all static simple faults", *Records of IEEE Int. Workshop MTDT*, pp. 95-100, 2002.

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

[4] S. Hamdioui, Z. Al-Ars and A.J. van de Goor, "Testing static and dynamic faults in random access memories", *In Proc. of IEEE VLSI Test Symposium*, pp. 395-400, 2002.

[5] R. D. Adams, E. S. Cooley, "Analysis of deceptive read destructive memory fault model and recommended testing", *Proc. of IEEE North Atlantic Test Workshop*, pp. 27-32, 1996.

[6] S. Hamdioui, R. Wadsworth, J. D. Reyes, A.J. van de Goor, "Importance of dynamic faults for new SRAM technologies" *In Proc. of IEEE European Test Workshop*, pp. 29-34, 2003.

[7] S. Hamdioui, G.N. Gaydadjiev, and A.J. van de Goor, "A fault primitive based analysis of dynamic memory faults", *IEEE 14th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, the Netherlands, pp. 84-89, 2003.

[8] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "March AB, March AB1: New March tests for unlinked dynamic memory faults", *ITC*, 2005.

[9] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "Automatic march test generation for static and dynamic faults, in SRAMs", *Proc. ETS 2005, Tallinn*, pp. 122-127, 2005 .

[10] G. Harutunyan, V. A. Vardanian, Y. Zorian, "Minimal march tests for dynamic faults in random access memories", *Journal of Electronic Testing: Theory and Applications*, vol. 23, Number 1, pp. 55-74, 2007.

[11] G. Harutunyan, V.A. Vardanian, Y. Zorian, "Minimal march tests for dynamic faults in random access memories", *In Proc. of IEEE European Test Symposium*, pp. 43-48, 2006.

[12] A.J. van de Goor, I. Schanstra, "Address and data scrambling: Causes and impact on memory tests", *Proc. IEEE Workshop DELTA*, pp. 128-136, 2002.

[13] J.-F. Li, K.-L. Cheng, C.-T. Huang, and C.-W. Wu, "March based RAM diagnostic algorithms for stuck-at and coupling faults", *Proc. IEEE ITC*, pp. 758-767, 2001.

[14] V. A. Vardanian, Y. Zorian, "A march-based fault location algorithm for static random access memories", *Proc. IEEE Int. Workshop MTDT*, pp. 62-67, 2002.

Արդյունավետ մարշատիպ ալգորիթմ դինամիկ անսարքությունների ենթադասի բոլոր անսարքությունների հայտնաբերման համար

Հ. Ավետիսյան, Գ. Հարությունյան, Վ. Վարդանյան

Ամփոփում

Այս հոդվածում ներկայացվում է արդյունավետ մարշատիպ ալգորիթմ, որը կարողանում է հայտնաբերել դասի բոլոր դինամիկ անսարքությունները. դասը երկու գործողությամբ զգայունացվող դինամիկ անսարքությունների ենթադաս է, այսինքն անսարքություններ, որոնք զգայունացվում են հիշողության բջջի նկատմամբ հաջորդական երկու գործողություն կատարելիս, առաջին գործողությունը ագրեսոր է, իսկ երկրորդը զոհ է բջջի նկատմամբ: Նախկինում դիտարկված են եղել և ենթադասերը, որոնց համար հեղինակները ներկայացրել էին մարշ ալգորիթմեր: Եվ դասերը ուսումնասիրված չեն եղել, քանի որ նրանց համար հնարավոր չէ կառուցել մարշ անգորիթմ: Սակայն այս դասերի համար հնարավոր է կառուցել մարշատիպ անգորիթմներ: Հոդվածում ներկայացված է մարշատիպ ալգորիթմ, որը կատարելով գործողություն հայտնաբերում է դասի բոլոր անսարքությունները: