Fault Detection with Optimum March Test Algorithm

Conference Paper · February 2012

DOI: 10.1109/ISMS.2012.88

Nor azura Zakaria
Malaysian Institute of Microelectronic Systems
7 PUBLICATIONS **5** CITATIONS
SEE PROFILE

W. Z.W. Hassan
Universiti Putra Malaysia
82 PUBLICATIONS **133** CITATIONS
SEE PROFILE

Izhal Abdul Halin
Universiti Putra Malaysia
44 PUBLICATIONS **186** CITATIONS
SEE PROFILE

Xiaoqing Wen
Kyushu Institute of Technology
160 PUBLICATIONS **1,607** CITATIONS
SEE PROFILE

Fault Detection with Optimum March Test Algorithm

Nor Azura Zakaria,

MIMOS Berhad,
Microelectronic Department,
Technology Park Malaysia,
Bukit Jalil,
57000 Kuala Lumpur,
Malaysia
norazura@mimos.my

W.Z.W. Hasan, I.A. Halin, R.M Sidek

Department of Electrical and Electronic Engineering,
Faculty Engineering, University Putra Malaysia,
Serdang, Selangor, Malaysia
wan@eng.upm.edu.my,
izhal@eng.upm.edu.my,
roslina@eng.upm.edu.my

Xiaoqing Wen

Department of Creative Informatics
Graduate School of Computer Science & Systems Engineering
Kyushu Institute of Technology
Iizuka, Japan
wen@cse.kyutech.ac.jp

Abstract— **Integrating a large number of embedded memories in System-on-Chips (SoC's) occupies up to more than 70% of the die size, thus requiring Built-In Self-Test (BIST) with the smallest possible area overhead. This paper analyzes MATS++(6N), March C-(10N), March SR(14N), and March CL(12N) test algorithms and shows that they cannot detect either Write Disturb Faults (WDFs) or Deceptive Read Destructive Faults (DRDFs) or both. Therefore to improve fault detection, an automation program is developed based on sequence operation (SQ) generation rules. However after solving the undetected fault, the outcome in term of its detection result of Static Double Cell Faults using the specified test algorithm especially Transition Coupling Faults (CFtrs), Write Destructive Coupling Faults (CFwds), Read Destructive Coupling Faults (CFrds) and Deceptive Read Destructive Faults (CFdrds) are observed.**

Keywords – Deceptive Read Destructive Faults; Write Disturb Faults; March Test Algorithm

I. INTRODUCTION

Nowadays Static Random Access Memory (SRAM) has become an indispensable component of digital systems, which can be used as a standalone product or embedded memories in a System on Chip (SoC) product. The number of SRAM cores in an SoC is increasing dramatically because of the design requirement of facilitating multiple applications. This leads to ever-higher density and ever-larger die sizes. On the other hand, technology scaling leads to smaller feature sizes while enabling a huge number of transistors to be fabricated into a single chip. However, such technology scaling also leads to higher risk of unknown defects that randomly occur in such a huge number of memory cells. Therefore, fault diagnosis and debugging of SRAM are complicated and require efficient test algorithms.

In the industry, test algorithms with 10N to 14N test operations, such as March C- (10N), March2 (14N) [17], March SR (14N), and March CL (12N), are usually used for testing memories in an SoC. It has been shown that March SS with 22N [1] operations can detect all Static Single Cell Faults (SSCFs) and Static Double Cell Faults (SDCFs) and March MSS with 18N operations [6] can detect all unlinked faults of SSCFs and SDCFs. However, the number of test operations of these algorithms is too large for an SoC since memories occupy around 70% to 90% of the chip area and thus the area available for Built-in-Self-Test (BIST) is limited. For testing more than 30 memories with various sizes and port numbers, the area overhead associated with a runtime programmable controller (Soft-Programmable controller) is higher than a controller that uses only hard-coded algorithms (Hard-Programmable controller) [18]. If the implementation is using Hard-Programmable controller, the physical routing will become more complicated and hard to manage. Hence, our BIST hardware is limit to memory test algorithms of up to 14N test operations.

Referring to the discussion in Section II.B and based on our analysis towards Functional Fault Primitive (FFP) and studies on SSCFs detection by well-known algorithms such as MATS++(6N), March C-(10N), March SR (14N) and March CL(12N), the following were concluded: March C- and MATS++ algorithms cannot detect DRDFs and WDFs, both March SR and March CL cannot detect WDFs and March CL algorithm can only detect DRDF0. This occurrence are due to the test sequence of the test algorithm inability to fulfill their Functional Fault Primitives (FFPs) for both or one of them, referring to FFP for WDF = ***(<0w0/1/-> , <1w1/0/->)*** which represent WDF0 and WDF1 respectively and FFPs for DRDF= ***(<r0/↑/1>, <r1/↓/0>***), which represent DRDF0 and DRDF1 respectively [4][5]. Previous works only addressed the testing of DCFs with shorter time, but did not discuss how to improve the testing of DRDFs and WDFs [4-8].

This work will focus on the latest result obtained after solving the undetected fault of SSCFs and to observe the outcome in term of its detection result of SDCFs using the specified test algorithm especially Transition Coupling Faults (CFtrs), Write Destructive Coupling Faults (CFwds), Read Destructive Coupling Faults (CFrds) and Deceptive Read Destructive Faults (CFdrds). For improving fault detection, an automation program is developed to generate a modified March-Test algorithm based on the sequence operation (SQ) generation rule scheme.

The rest of the paper is organized as follows: Section II shows the FFP list for all faults and its notation. This section also summarizes the steps in generating a new sequence of transition and non-transition operation to test for undetected SSCFs. In Section III, the development of new March Test algorithm from the test program based on SQ generation rule scheme is discussed in detail. This section also includes detection results and hardware evaluation results. Section IV discusses the performance of new test algorithm compared with previous works. Section V concludes this paper.

II. Background

The FFPs for SSCFs and SDCFs are listed in the Table I. SRAM testing can be classified into two categories, namely the testing of Single Cell Faults (SCFs) and the testing of Double Cell Faults (DCFs), which can be further sub-divided into dynamic faults and static faults, respectively. There are six types of static SCFs (SSCFs), which are Transition Faults (TFs), State Faults (SFs), Write Disturb Faults (WDFs), Read Disturb Faults (RDFs), Incorrect Read Faults (IRFs), and Deceptive Read Disturb Faults (DRDFs) [1-5]. For static DCFs (SDCFs), our focus is on Write Destructive Coupling Faults (CFwds), Deceptive Read Destructive Faults (CFdrds) and Transition Coupling Faults (CFtrs).

A. Fault Notation

In order to describe these faults, a compact notation referred as *Fault Primitive (FP)* will be used, as shown in Table I. All the notations and fault behaviours are discussed in [4]. The notation of FP is explained as below:

1) < S/F/R > or < S/F/R $>_v$) denotes a FP involving a *single cell;* the cell c_v (victim cell) used for sensitizing a fault is the same as where the fault appears. *S* describes the *sensitizing* operation or state; *S* {0,1, w0, w1, w, w r0, r1} whereby 0 denotes a 0 value, 1 denotes a 1 value, w0 (w1) denotes write 0 (1) operation, w ↑ (w↓) denotes an up (down) transition write operation, *r0* (r1) denotes a read 0 (1) operation.

2) <S_a;S_v,/F/R> (or <S_a;S_v,/F/R$>_{a,v}$): denotes a FP involving *two cells;* S_a denotes the sensitizing operation or state of the *aggressor cell (a-cell);* while S_v denotes the sensitizing operation or state of the *victim cell (v-cell).* The a-cell *(c_a)* is the cell sensitizing a fault in another cell called the v-cell (c_v). The set of S_i is defined as: S_i Є{ *0,1, X, w0, wl, w*↑, *w*↓*, r0, rl*}(*i* Є *{a,v)),* whereby X is the don't care value *X* Є {0,1}.

3) In both notations, *F* denotes the value of the *faulty* cell (v-cell); *F* Є{0,l,↑,↓,?}, whereby ↑(↓) denotes an up (down) transition and "?" denotes an undefined logical value. R denotes the logical value which appears at the output of the SRAM if the sensitizing operation applied to the v-cell is a *read* operation; *R* Є {0,l,?,-}, whereby "?" denotes an undefined or random logical value. An undefined logical value can occur if the voltage difference between the bit lines (used by the sense amplifier) is very small. A '-' in *R* means that the output data is not applicable in that case; e.g., if *S* = *w0,* then no data will appear at the memory output, and therefore *R* is replaced by a '-'.

TABLE I. Functional Fault Primitives

Single Cell Faults	
Functional Fault Model	**Fault Primitives**
SF	**<1/0/->, <0/1/->**
TF	**<0w1/0/->, <1w0/1/->**
WDF	**<0w0/↑/->, <1w1/↓/->**
RDF	**<r0/↑/1>, r1/↓/0>**
DRDF	**<r0/↑/0>, r1/↓/1>**
IRF	**<r0/0/1>, r1/1/0>**

Double Cell Faults	
Functional Fault Model	**Fault Primitives**
CFwd	**<0;0w0/↑/->, <1;0w0/↑/->, <0;1w1/↓/->, <1;1w1/↓/->**
CFdrd	**<0;r0/↑/0>, <1;r0/↑/0>, <0;r1/↓/1>,1; r1/↓/1>**
CFtr	**<0;0w1/0/->, <1;0w1/0/->, <0;1w0/1/->, <1;1w0/1/->**

B. Undetectable Fault Analysis on WDFs and DRDFs

As mentioned in Section I, the fault analysis will focus on March Algorithm with maximum of 14N test operations. Table II shows four types of well known March Test Algorithms in this case study, namely MATS++, March C-, March SR and March CL.

TABLE II. WELL KNOWN MARCH TEST ALGORITHM

March Algorithm	Operation Sequence
MATS++ :6N [9,10,15]	*{⇕ (w(0)); ⇑ r(0),w(1)); ⇓ (r(1) , w(0), r(0)) }.*
March C- :10N [8,9,10,14]	*{ (w0); (r0,w1); (r1,w0); (r0,w1); (r1,w0); (r0) }*
March SR: 14N [3]	*{ (w0) ; (r0,w1,r1,w0) (r0,r0,); (w1); (r1,w0,r0, w1); (r1,r1) }*
March CL 12N [2,7,10]	*{ (w0); (r0,w1); (r1 , r1, w0); (r,w1,r1,); (r1,w0); (r0) }*

TABLE III. TEST OPERATION SEQUENCE OF WDF AND DRDF DETECTION

Fault Type	FFP	Test Operation Sequence
Write Disturb Fault (WDF)	WDF0 = < 0w0/↑/-> WDF1 = < 1w1/↓/->	⇕ (wx,wx,**rx**) or (..wx) ;⇕ (wx**rx**) or ⇕ (..wx..wx) ;⇕ (**rx**..) or ⇕ (wx, rx;) ⇕ (wx,**rx**) *⇕ - arbitrary addressing order, it can be opposite addressing or same addressing order,* **rx** *,the bold font denotes the detection, if the fault occur,* **x** *is read inverted value.* *If x is 0, WDF0 is detected and if x is 1, WDF1 is detected*

Deceptive Read Destructive Fault (DRDF)	DRDF0= < r0/ ↑/0>
	
	DRDF1 = < r1/ ↓/1>
		⇕ (rx,**rx**) or ⇕ (..rx) ;⇕ (**rx..**) or ⇕ (rx) ⇕ (**rx**) or ⇕ (..rx) ⇕ (**rx**..)
		
		⇕ - arbitrary addressing order, it can be opposite addressing or same addressing order, **rx** *,the bold font denotes the detection, if the fault occur,* ***x*** *is read inverted value.*
		
		If x is 0, DRDF0 is detected and if x is 1, DRDF1 is detected

TABLE IV. MARCH ALGORITHM AND ITS DETECTION ANALYSIS

March Algorithm	Fault detection towards FFMs and its test sequence operation			
	WDF0	WDF1	DRDF0	DRDF1
MATS++	X	X	X	X
MARCH C-	X	X	X	X
MARCH SR	X	X	√	√
MARCH CL	X	X	√	X

Table III shows the functional fault primitives and the test sequence operation to detect WDFs and DRDFs. As shown in the third column of the Table III, if any of the test sequence operation is matched with the test operation listed in the elements of the March Algorithm, the specified fault can be detected. Based on the analysis on selected March Algorithm, list fault that can be detected and undetected is shown from Table IV. It shows that the MATS++ and March C- cannot detect WDFs and DRDFs. But March SR can detect DRDFs. Both March SR and March CL cannot detect WDFs however March CL can only detect DRDF0. Therefore, this is a motivation to detect the undetected faults by modifying the sequence of the transition values but maintaining the addressing order and its element.

C. Proposed Solution

To improve the fault detection of SRAM testing, some rules have been set to generate a new sequence of transition and non-transition values. This is to ensure the undetected faults such as Deceptive Read Destructive Faults (DRDFs) and Write Disturb Faults (WDFs) can be recovered. The March-test sequences should also consider the detection for Static Double Cell Faults (SDCFs) as well. The following steps are taken:

1) Allowing non-transition and transition operations and two consecutive double read operations in order to sensitize and desensitize faults such WDFs, TFs and DRDFs. There will be a customized March Test by modifying well-known algorithm with the automation program based on the sequence operation (SQ) generation scheme. This new March-test is developed with solid DBs, 0s and 1s.

2) The analysis of the fault detection of all SSCFs and SDCFs based on its fault descriptions and predefined condition stated in [5] will be conducted.

3) In the specified March algorithm, there is a need to have a double read operation whereby one of this operation is listed in the specified March Algorithm, which follows one of the test operation sequence specified in Table III. The operation rule must operate both condition values. If the double read operation only allows FFP ***(<r1/ ↓/0>)*** *of the DRDF1 to be detected, the March-test sequences should also facilitate to satisfy FFP* ***(<r0/ ↑/1)*** *of the DRDF0 detection and vice versa.*

4) If the test sequence operation does not consist of operations to detect WDFs specified in Table III, this sequence need to be added in the defined test Algorithm and if the sequence fulfilled only one condition, the operation rule of the non- transition values need to be added.

5) If the test sequence operation does not consists of test operations to detect TF↓ by given "w1w0r0" operations and to detect TF↑ by given "w0w1r1", this sequence operation needs to be added in the defined test Algorithm.

6) The March-test sequences must also consider the detection for Static Double Cell Faults (SDCFs) as well.

III. MODIFIED MARCH ALGORITHM AND ITS DETECTION

A. Developmentof Automation Program based on SQ Generation Rule Scheme

Firstly, one automation program is developed to generate the new March-test Algorithm. The solution is to include a new sequence of transition and non-transition values in the specified algorithm to ensure that the undetected fault such as Deceptive Read Destructive Faults (DRDFs) and Write Disturb Faults (WDFs) can be detected. In developing the new sequence of writing operations referred to as SQ, one rule was set and this is referred to as SQ generation rule. The rules are as follows:

a) Count Number of Write Operation in the test algorithm, given W_N where $\{W\}_N = W1, W2, W3, W4,\ldots WN$.

b) Writing 0 is a must for initialization. So it always starts with writing a 0, and removes all possible sequences that start with writing 1. With this condition, the total of possible sequence, total SQ list = $2^{(N-1)}$.

c) Writing a non-transition operation, *w0→w0* and *w1→w1,* must occur as the next sequence where this is specified as a static scheme operation. Generate any possible combination of non-transition operation sequences, where $W_{NT} = \{W_N-1\ \} => 4$. Collect all the combination of possible sequences and append the generated sequences with "w0" as the first operation.

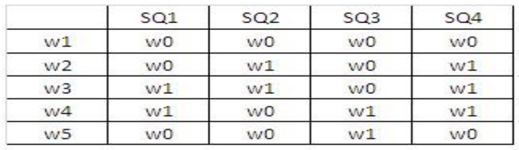
d) Writing transition operation, *w0→w1* and *w1→w0* must occur, otherwise, the sequence operation is omitted. This scheme is denoted as a dynamic scheme.

e) Double read operation position needs to be identified by ensuring both *r0,r0* and *r1,r1* occurs between it. If not, the generated sequence operation (SQ) is omitted. For example, the test algorithm has the double read operation after W3 = {w0} and W5={w0} where only DRDF0 can be detected but not DRDF1,which violated rule (d) and the generated SQ will then be omitted from the specified test.

f) The program will search for other possible sequence which obeys rule (a) to rule (e).

g) Finally a customized algorithm will be produced by replacing SQ list with all write operations and re-union with other operations that follows its March Element in the original March Test algorithm.

TABLE V. EXAMPLE OF THE POSSIBLE WRITING SEQUENCE IN SPECIFIED MARCH ALGORITHM



	SQ1	SQ2	SQ3	SQ4
w1	w0	w0	w0	w0
w2	w0	w1	w0	w1
w3	w1	w1	w0	w1
w4	w1	w0	w1	w1
w5	w0	w0	w1	w0

An example of possible sequence operation for a specified test algorithm consisting of five write operations, $W_N = 5$, is illustrated in Table V. The generated SQs follows rule (a) and rule (b). For rule (c), SQ4 is omitted as there is no w0→w0. For rule (d), SQ3 is omitted as there is no transition w1→w0 thus TF1 cannot be detected. For rule (e), if the read operation happened after w2 and w4, SQ4 is omitted because DRDF0 cannot be detected. Since SQ3 and SQ4 are omitted, SQ1 and SQ2 satisfy all rule requirements, making it the final sequence operation that can detect SSCFs.

B. Development of New March Algorithm and Its Detection

The test algorithm with the new SQ list operation will produce a new test algorithm called new-March-test Algorithm. All possible SQs will be inserted into the March Algorithm until the optimum number of SQs is fulfilled to obtain the highest coverage.

1) Modification of MATS++ and March C-

MATS++ consists of 3 write operations that is not sufficient to meet SQ rule (c) and SQ rule (d). For March C- the implementation only can be done up to SQ rule (d) and since the test algorithm cannot comply with SQ rule (e), the implementation work for Mod March C- is not discussed. Note that with SQ lists that are generated according to SQ rule (d), the detection improvement only covers WDFs. Therefore the development of Mod March C- and Mod MATS++ can be ignored.

2) Modification of March SR(14N)

The Modified March SR Algorithm has two sets of March-test sequences for the detection of all SSCFs and some SDCFs such CFwd, CFdrd and CFtr. The new March-tests are as follows:

1) New March SR-1 :{⇕ (w(0)) ⇑ (r(0),w (1), r(1), w(1)); ⇑ (r(1),r(1)); ⇑ (w(1); ⇓ (r(1),w(0),r(0), w(0)); ⇓(r(0),r(0))}
2) New March SR-2 :{⇕ (w(0)); ⇑ (r(0),w (1), r(1), w(1));⇑ (r(1),r(1)); ⇑ (w(0); ⇓ (r(0),w(0),r(0), w(0)); ⇓r(0),r(0))}

The generated March Algorithm shows less number of transitions whereby March SR-1 has 3 times of transition write operations and March SR-2 has 2 times of transition write operations compare to the original March SR has 4 times transition write operations. Thus in term of test power performance, power consumption for generated Algorithm is reduced.

3) Modification of March CL (12N)

The Modified March Algorithm for March CL has 2 types of March-test sequences for the detection of SSCFs. The modified March-tests are:

1) New March-test-CL(1):{⇕(w(0)); ⇑(r(0),w(0)); ⇕(r(0)); ⇑(r(0)w(1)); ⇓(r(1),w(1)); ⇕(r(1)); ⇓ (r(1),w(0)); ⇕(r(0)) }
2) New March-test-CL(2):{⇕(w(0)); ⇑ (r(0),w(1)); ⇕ (r(1)); ⇑ (r(1),w(1)); ⇓ (r(1),w(0)); ⇕ (r(0)); ⇓ (r(0),w(0)); ⇕ (r(0)) }

Both generated test Algorithm only has two times transition write operations compare to the original test Algorithm has 3 times transition write operations, thus reduce test power consumption.

IV. RESULTS AND DISCUSSION

Referring to Section III.B.2 and Section III.B.3, the summary result of the fault detection and the total fault coverage of the original March Algorithm compare to its new March Algorithm is shown in Table VI and Table VII. The evaluation data collected in Table VI shows the fault detected number compare to the total of the expectation faults to be detected. The total detection of each type CFs denotes 8 detections refer to the condition of the position of the aggressor cell, where one is the address of the aggressor cell is higher than the victim cell ($c_a > c_v$) and another one is the address of the victim cell is higher than the aggressor cell ($c_v > c_a$). If the FP condition is 4, the total condition detection will be 8.

TABLE VI. FAULT DETECTION

	March SR	March SR-1	March SR-2	March CL	March CL-1	March CL-2
CFwd	0/8	6/8	6/8	0/8	4/8	4/8
CFdrd	6/8	4/8	6/8	4/8	4/8	4/8
CFtr	8/8	2/8	2/8	6/8	4/8	2/8
WDF	0/2	2/2	2/2	0/2	2/2	2/2
DRDF	0/2	2/2	2/2	1/2	2/2	2/2

TABLE VII. FAULT COVERAGE

	March SR	March SR-1	March SR-2	March CL	March CL-1	March CL-2
CFwd	0.00%	75.00%	75.00%	0.00%	50.00%	50.00%
CFdrd	75.00%	50.00%	75.00%	50.00%	50.00%	50.00%
CFtr	100.00 %	25.00%	25.00%	75.00%	50.00%	25.00%
SSCFs	91.67%	100.00%	100.00%	75.00%	100.00%	100.00%
Total	66.67%	62.50%	68.75%	50.00%	62.50%	56.25%

By using the SQ generation rule scheme, the generated test algorithms achieve 100% fault coverage for SSCFs. Detection performance especially improves the detection of CFwd per each generated Algorithm. However for CFtr detection, March Cl-1 gives the moderate detection by achieving 50% fault coverage. For CFdrd detection, March SR-2 still maintains the detection performance compare to the original Algorithm. As described in Section III.B, the generated test algorithm still leads the test power performance even the performance of CFs are not achieving up to 100% fault coverage. The detection of CFs in this work is limited because the program maintains the element of the operation in the algorithm and only focuses to change the values data to achieve 100% of SSCFs compare to the original Algorithm. The undetected fault, WDFs and DRDFs is definitely improved by the proposed solution.

V. Conclusion

In this paper, we have shown that the initially undetectable SSCFs faults can be detected by generating new March-test algorithms from well-known March algorithms with a maximum of 14N operations. The customized March-test algorithm is programmed based on the SQ generation rule. The SQ generation rule dominates rules in writing and reading sequences to fulfill FP of SSCFs. The generated test algorithm also considers the SDCF detection. The proposed work shows by achieving SSCF detection up to 100% fault coverage, it effects the detection of CFwd, CFdrd and CFtr. This is because the generated algorithm is maintaining the element of the operation without having an additional operation to improve the detection result. Ultimately, a March SR-2 gives the highest coverage with 68.75% with 2% improvement and less power consumption. For the future work, some features in the proposed sequence operation to improve SDCFs detection will be enhanced later.

Acknowledgment

We wish to express our great gratitude to Edward V. Bautista Jr from MIMOS Berhad, Malaysia and Nor Zaidi Haron from TUDELFT, Netherland for the valuable advice.

References

[1] S. Hamdioui, A. J. van de Goor, and M. Rodgers, “March SS: A Test for All Static Simple RAM faults,” Proceeding of IEEE International Workshop Memory Technology, Design and Testing, Bendor, France, pp. 95–100, July 2002

[2] J. V.A Vardanian, Y. Zorian, “A March-based fault location algorithm for static random access memories” Proceedings of the Eighth IEEE International in On-Line Testing Workshop, pp. 256 - 261, 8-10 July 2002

[3] S. Hamidoui, A.J. van de Goor, “An Experiment Analysis of Spot Defects in SRAMs: Realistic Fault Models and Tests” Proceedings of the Ninth Asian Test Symposium, 2000 , pp. 131 – 138, 4-6 Dec. 200

[4] S. Hamdioui, ‘Testing Static Random Access Memories: Defects, Fault Models and Test Patterns’, Kluwer Academic Publishers, Boston, MA; ISBN 1-4020-7752-1, 2004

[5] A.J. van de Goor and Z. Al-Ars, “Functional Memory Faults: A Formal Notation and a Taxonomy", Proceeding of 18th VLSI Test Symposium 2000, pp. 281-289, 6 August 2002

[6] G. Harutunvan, V.A Vardanian, Y. Zorian, “Minimal March Tests for Unlinked Static Faults in Random Access Memories” Proceeding of 23rd VLSI Test Symposium, 2005, pp. 53 – 59, 2005

[7] C.-F. Wu et al., ‘‘Fault Simulation and Test Algorithm Generation for Random Access Memories,’’ IEEE Trans.Computer-Aided Design of Integrated Circuits and Systems, vol. 21, no. 4, 2002, pp. 480-490.

[8] Wei-Lun Wang, Kuen-Jong Lee "A programmable data background generator for march based memory testing" Proceeding of IEEE Asia-Pacific Conference on Digital 2002, APASIC 2002, pp 347-350, 2002

[9] Jin-Fu Li and Cheng-Wen Wu, “Memory Fault Diagnosis by Syndrome Compression”, Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001, pp. 97–101, 13-16 March 2001

[10] Wan Hassan, W.Z, Othman, M., Suparjo, B.S., “A Realistic March-12N test And Diagnosis Algorithm For SRAM Memories” IEEE International Conference on Semiconductor Electronics, pp. 919 – 923, 2 July 2007

[11] Bosio, A. Di Carlo, S. Di Natale, G. Prinetto, P. “March AB, a state-of-the-art march test for realistic static linked fault and dynamic faults in SRAMs” Computers & Digital Techniques, IET , vol.1 No. 3, pp. 237-245, 21 May 2007

[12] L.-T. Wang, C.-W. Wu, and X. Wen, (Editors), Chapter 8, VLSI Test Principles and Architectures: Design for Testability, San Francisco: Elsevier, 2006.

[13] S. Hamdioui, J.E.Q.D. Reyes "New data-background sequences and their industrial evaluation for word-oriented random-access memories" IEEE Transaction On Computer Aided Design Of Integrated Circuits and System, Vol. 24, No. 6, pp. 892-904, 2005

[14] A.J. van de Goor, I.B.S. Tlili, and S. Hamdioui, “Converting March Tests for Bit-Oriented Memories into Tests for Word-Oriented Memories” In Proc. of IEEE Int. Workshop on Memory Technology, Design

[15] A.J. van de Goor, Testing semiconductor memories: Theory and Practice, John Wiley & Sons, Chichester, England, 1991

[16] R. Dean Adams, High Performance Memory Testing: Design Principles, Fault Modeling and Self Test, Secaucus, NJ, USA, Kluwer Academic Publishers. pp. 138, 2002

[17] Sultan M. Al-Harbi, Fadel Noor, Fadi M. Al Turjman, “March DSS: A New diagnostic March Test For All Memory Simple Static Fault” IEEE Transaction On Computer Aided Design Of Integrated Circuits and System, Vol. 26, No. 9, September 2007

[18] Mentor’s Product ,Tessent™ MemoryBIST Usage Guide and Reference, November 2010

[19] International Technology Roadmap For Semiconductors 2010 Update Overview