Scan More with Memory Scan Test

Said Hamdioui Zaid Al-Ars
Delft University of Technology
Computer Engineering Laboratory
Mekelweg 4, 2628 CD Delft, The Netherlands
{S.Hamdioui, Z.Alars}@tudelft.nl

Abstract: *Almost all manufacturing memory test programs use the time-efficient Scan test to screen the defective chips in an early stage. Usually, Scan is used to screen out the easy-to-detect hard faults like stuck-at-faults. In this paper we will show how Scan can be modified to increase the fault coverage and detect unique faults. It will be shown that many additional faults are detectable using Scan if appropriate read-write sequence, an appropriate data-background and appropriate addressing method are used. Such additional faults consist not only of static/traditional faults, but also of dynamic and time-related faults which are of increasing importance with the technology scaling. Examples of such faults are dynamic faults in the peripheral circuits (e.g., sense amplifiers, pre-recharge circuits, etc) and in the address decoders. Industrial results are presented to validate the proposed approach.*

Keywords: *memory testing, static faults, dynamic faults, Scan test, fault coverage*

1 Introduction

Random Access Memories, which are an integral part of any ULSI chip (e.g., a microprocessor), are widely considered to be one of the most critical components. Not only, because the memory share of the chip area increases and is expected to be about 94% in 2014 [1], but also because of technology shrinking, which makes memories more sensitive to defects.

In order to realize a high fault coverage and therefore a high overall product quality, usually a *set* of memory tests, rather than a *single* test, are used in the manufacturing memory test program to screen out the defective memory part [2, 3]. Each test is used to target some specific faults; e.g., memory array faults, address decoder faults, data retention faults, etc. It is interesting to note that the ad-hoc test *Scan* [4], developed begin eighties, belongs almost always to the memory test program. The main purpose of Scan is to screen out the easy-to-detect defective memory parts; e.g., memory parts suffering from stuck-at-faults.

This paper analyzes Scan test and shows how stress combinations and the freedom of march tests can be adapted in such way that additional faults can be detected, including new fault models like dynamic and time-related faults in the peripheral circuits and address decoders. It also introduces new versions of Scan along with silicon results to validate the proposed approach.

The paper is organized as follows. Section 2 gives an overview of traditional/static faults and advanced dynamic faults. Section 3 introduces the notion of memory algorithms and stresses. Section 4 give the detection conditions for the targeted dynamic faults. Section 5 analyzes Scan and gives new versions to cover the targeted dynamic faults. Section 6 discusses some silicon results. Last, Section 7 ends with the conclusions.

2 Memory fault models

Traditional memory fault models referred to as *static faults* are generally speaking divided into three types:

1. Static Memory Cell Array Faults: these consist of static faults occurring in the memory array; they can be either single-cell faults (e.g., State Fault, Transition Fault, Read Destructive Fault, etc.) or coupling faults (CFs) (e.g., State CF, Transition CF, Disturb CF, etc.) [5]-[8].
2. Static Address Decoder Faults: these are faults occurring in the address decoders. They consist of four known fault models [5]: No-Access fault, Multiple-Cell fault, Multiple-Address fault and Other-Cells fault.
3. Static Peripheral Circuit Faults: these are faults occurring in the rest of the memory circuit (e.g., sense amplifier, pre-charge circuits, etc.). These faults are modeled as stuck-at-faults and bridging faults and considered to be covered with any test detecting static memory cell array faults and therefore they are mapped into memory cell array faults [5].

To detect the above three types of static faults, many test algorithms have been introduced with different degrees of success.

With the scaling of technology, new defect mechanisms take place in addition to the known traditional ones. The new defect mechanisms cause faulty behaviors that are different from the static faults. They cause mainly *time-related* faults also known as *dynamic faults*. Dynamic faults can also be divided into three types:

1. Dynamic Memory Cell Array Faults: these are dynamic faults that occur in the memory array. They require more than one operation to be applied to the victim-cell and/or to the aggressor cell in order to sensitize the fault in the victim cell. Examples of such faults are: Dynamic Read Destructive Fault, Dynamic Transition Fault, Dynamic Write Disturb Fault, Dynamic Coupling Fault, etc. [9]-[13].
2. Dynamic Address Decoder Faults: these consist of delay faults that can occur in the address decoders. They are mainly caused by partial opens. Such faults are modeled as [14]-[17]:
   (a) Activation delay fault (ActD)
   (b) De-activation delay fault (DeactD)
3. Dynamic Peripheral Circuit Faults: these are time-related faults caused by defects occurring in the peripheral circuits of the memory system like partial opens, partially filled vias and leakages. Dynamic peripheral circuit faults consists mainly of [18, 20]:
   (a) Slow Write Driver Fault (SWDF)
   (b) Slow Sense Amplifier Fault (SSAF)
   (c) Slow PRecharge Circuit Fault (SPRF)

3. Memory test algorithms and stresses

Because most memory tests have the form of a march test, below the notation of march tests will be given, followed by a description of the stresses of interest to this paper.

Notation of march tests: A *march test* is a sequence of *March Elements 'MEs'*. A ME consists of a sequence of operations applied to every cell (n is the number of cells in the memory), in either one of two *Address Orders 'AOs'*: an *increasing AO* '$\Uparrow$', (e.g., from cell 0 to cell $n-1$), or a *decreasing AO* '$\Downarrow$' (e.g., from cell $n-1$ to cell 0). When the AO is irrelevant the symbol '$\Updownarrow$' is used.

Example: $\{\Uparrow(w0); \Uparrow(r0); \Uparrow(w1); \Uparrow(r1)\}$ is the Scan test [4]. It consists of four MEs. E.g., the notation '$\Uparrow(r0)$' means 'for $i = 0$ to $n-1$ do {read $A[i]$ with expected value 0'.

Algorithmic stresses: When performing memory testing, each test is applied using several stresses. The stresses can be divided into algorithmic stresses and non-algorithmic stresses.

A *non-algorithmic stress*, also referred to as an *environmental* stress, specifies the environmental values, such as the supply voltage, the temperature, the timing (the clock frequency), etc.; they are effective during the application of the test. For example, the test algorithm can be Scan and the non-algorithmic stresses can be $V_{DD} = 1.8V, Temp = 70^0C$, clock frequency = 1MHz, etc.

An *algorithmic stress* specifies the way the test is performed, and therefore it influences the sequence and/or the type of the memory operations. The most well known algorithmic stresses are the address methods, the address direction and the data-background stresses.

Address Method 'AM' describes the method used for generating the sequence of addresses. A well-known AM is the *Binary AM 'Bin'*; for $N = 3$ (where N is the number of address lines), it consists of the address sequences $\Uparrow^{Bin}$= $\{0,1,2,3,4,5,6,7\}$ and $\Downarrow^{Bin}$= $\{7,6,5,4,3,2,1,0\}$.

Address Direction 'AD' specifies the direction (i.e., along rows, columns, or diagonals) in which the address sequence has to be performed. The commonly used ADs in the industry are: Fast-X, Fast-Y and Fast D. *Fast X 'fX'* addressing is incrementing or decrementing the address in such a way that each step goes to the next row. *Fast Y 'fY'* addressing is incrementing or decrementing the address in such a way that each step goes to the next column. *Fast D 'fD'* is incrementing or decrementing the address in such a way that each step goes to the next diagonal.

Data Background 'DB' is the pattern of ones and zeros as seen in an array of memory cells. The most common types of DBs are:

Solid (sDB): all 0s and all 1s;
Checkerboard (bDB):0101.../1010...;
Column Stripe (cDB): 0101.../0101...; and
Row Stripe (rDB): 0000.../1111....

4 Detection conditions for dynamic faults

We will deal with dynamic faults in the peripheral circuits and address decoders; dynamic faults in the memory cell array are addressed in e.g., [9]-[13]. A way to express the capability of a test to detect dynamic faults in the peripheral circuits and the address decoder is in terms of required *Read-Write Sequences 'RWSs'* [17, 20]. A RWS is a property of a march element and is defined by the last operation applied to an address and the first operation applied to the next address. E.g., the march element '$\Uparrow(r0, w1, r1, w1)$' performs a *Read-after-Write 'RaW'* RWS, because the last

operation to an address is the '$w1$' operation, and the first operation applied to the next address is the '$r0$' operation. The set of RWSs consists of:

RaR: Read-after-Read,
WaW: Write-after-Write,
WaR: Write-after-Read, and
RaW: Read-after-Write.

Dynamic Peripheral Circuits Faults: Below, a summary will be given of the required detection conditions (e.g., RWSs and DBs) for detecting *Dynamic Peripheral circuit Faults 'DPFs'* [20]. DPFs consist mainly of faults due to: (a) a *Slow Write Driver*, (b) a *Slow Sense Amplifier*, and (c) a *Slow PRecharge circuit*; see Section 2.

- **SWDF**: The write driver may be too slow due to a defect in the driver circuit and/or due to resistive defects (such as partial open vias) in its path to the to-be-written cells. The result will be that the differential voltage on the bit lines during the write operation is reduced. This may cause the cell not to be written [20, 19]. *The WaW RWS is the most stressful RWS, using alternating data values of '0' and '1' (i.e., using the bDB or the rDB) and fX Addressing Direction (AD).*
- **SSAF**: The sense amplifier may be too slow, or is asymmetric (because of some offset voltage) due to a defect in its circuitry and/or due to resistive defects in the path from the cell to its circuitry. This may cause read operations to produce incorrect results. *The RaR RWS is the most stressful RWS, using the bDB or rDB with fX AD.*
- **SPRF**: The prechage circuit may be too slow, or it may not precharge both bit lines to the same voltage level, due to defects in its circuitry and/or due to resistive defects in the bit lines [18]. The result may be that especially read operations will produce incorrect results, because they are most sensitive to bit line voltage offset errors. Any RWS using alternating data values may detect the fault. *The RaW RWS is the most stressful RWS, using the bDB or rDB with fX AD.*

Tests for above peripheral circuit faults should be based on special RWS, use bDB or the rDB, and use fX address direction (because they all deal with faults in a column of the cell array). This shows the importance of choosing appropriate RWS, DBs and ADs to detect some special faults.

Dynamic Address Decoder Faults: Dynamic address decoder faults consist of (a) Activation Delay (ActD) and (b) Deactivation Delay (DeactD). These faults are caused by (partial) resistive opens in the address decoder paths. When a defect causes a delay in a rising edge of word line/ column select, then we are talking about an ActD; when a defect causes a delay in a falling edge of word line/ column select, then we are talking about DeactD. Depending on the nature of the decoder circuit, only one or both types of delay faults can occur [17].

Table 1. Summary detection conditions

Fault	RWS	AM	AD	DBs
SWDF	WaW	Any	fX	bDB or rDB
SSAF	RaW	Any	fX	bDB or rDB
SPRF	RaW	Any	fX	bDB or rDB
ActD & DeActD for RD	WaW	H1	fX	bDB or rDB
ActD & DeActD for CD	WaW	H1	fY	bDB or cDB
Note: RD= Row Decoders, and CD= Column Decoders				

The detection of the ActD and DeactD requires special Addressing Methods (AMs) to generate the required *sensitizing address transitions*. As shown in [17], in order to guarantee the detection of both ActD and DeactD, e.g., H1 addressing is required; it guarantees the generation of all required sensitizing address transitions. H1 addressing is briefly explained next.

H1 stands for *Hamming Addressing* where the hamming distance between two successive addresses is 1. For example for N=2, the generated H1 addresses are: {00, 01, 00, 10, 00, **11, 01, 11, 10, 11**}. One method to generate H1 is to use *constant weight codes*, also referred to as *m-out-of-n codes* [22, 17]. They have the property that each n-bit *Code-Word 'CW'* contains exactly m 1's (i.e., has a *weight* of m). The total number of addresses generated using H1 addressing (guarantying all required sensitizing address transitions) is: $N_{H1}(N)=(2N+1)*2^{N-1}$, where N is the number of address lines [17]; e.g., for N=2, this will be $N_{H1}(N=2)=10$.

In addition to appropriate AMs, special RWS are needed for detecting dynamic address decoder faults. The WaW RWS is the most stressful RWS, using alternating data values of '0' and '1. In conclusion, to test dynamic address decoder faults (i.e., ActD and DeacD) in row decoders, the test should use WaW RWS, fX AD and bDB or the rDB; while to test for the same faults in the column decoders, the test should use WaW RWS, fY AD and bDB or cDB should be used. For example, if a word line WLx has a deactivation delay, it may be selected at the same time when word line WLy is selected. Preforming WaW using (a) alternating values 1 and 0 and (b) fX will cause the written cell with 1 to be overwritten with 0.

Table 1 summarizes the detection conditions and the most stressful RWS required for the detection of dynamic faults in peripheral circuits and address decoders. Each row in the table consists of a fault (e.g., SWDF), the required RWS, and the required algorithmic stresses (AM, AD and DBs). In the table, "'any"' denotes any possible AM (e.g. binary, H1, etc). It is clear from the table that in order to achieve a better fault coverage, when Scan is used with fX, then it is

recommended to use bDB and/or rDB; and when it is used with fY, then it is recommended to use bDB and/or cDB.

5 Scan test and its new variants

In this section, first conventional Scan is briefly addressed. Thereafter, new versions of Scan with appropriate stresses will be proposed to improve the overall fault coverage; this will be done based on the detection conditions analyzed in the previous section.

5.1 Conventional Scan

The traditional Scan test is a very simple test. Its description is: $\{\Uparrow(w0); \Uparrow(r0); \Uparrow(w1); \Uparrow(r1)\}$. The test is usually used with a *binary AM*, and therefore its test length is $4n$, where n is the size of the memory (i.e., the total number of addresses). Despite its simplicity, Scan sometimes detects *unique faults* that cannot be detected with any other test [2, 21]. That may indicate that Scan can detect additional faults that are beyond the conventional/static faults. In fact, an evaluation of Scan reveals that its traditional/static fault coverage is low as compared with other known march test like March C-[7]; Scan can detect stuck-at faults, partially detect transition faults, detect read disturb faults, partially detect coupling faults, etc.

Scan test is special because it is the *only* march test which has a WaW and a RaR RWS, without other operations in between, and therefore it has the capability to detect e.g., the SWDF. For example, the WaW RWS, using the bDB, performs the following sequence of write operations: w0, w1, w0, w1, etc. When performed using the fX AD, the write drivers, as well as the row decoder, are stressed the most: each write operation has to write the inverse data, while at the same time it has to apply the operation to the next row address!

5.2 New versions of Scan

If now, Scan is used with H1 addressing method, fX with checkerboard or row-stripe data-background, and fY with checkerboard or column-stripe data-background, then Scan will satisfy the *most stressful* detection conditions for SWDF, SSAF and dynamic address decoder faults (ActD and DeactD); see Table1. This new version of Scan is given in the first row of Table 2 and referred to as H1-Scan. Note that 2^i can also be used instead of H1; however H1 is used in this case since it produces a test with shorter test length. The test uses the following notation for the operations '$(Op1/Op2)$', where $Op1, Op2 \in \{wx, rx\}$; $x \in \{0, 1\}$. '/' denotes that the two operations '$Op1$ and $Op2$' are applied to two successive generated addresses. For example for the number of address lines N=2, H1 addresses are {00, 01, 00, 10, 00, **11, 01, 11, 10, 11**}. So, the first march element will apply w0 to address 00, then w1 to 01, then w0 to 00 etc. till w0 to 10 and w1 to 11. The next march element $\Downarrow^{H1}(r1/r0)$ will apply first r1 to address 11, then r0 to 10, etc.

Note that the addressing orders AOs '$\Uparrow$' and '$\Downarrow$' with H1 AM are specified. The AOs are important for the detection of dynamic address decoders fault. For example, assume that a word line WLx with address Ax has DeactD. Assume that sensitization address transition requires the access of addresses (Ax,Ay) where $Ax<Ay$; Ay is the address of word line WLy. When performing $\Uparrow^{H1}(w0/w1)$ with bDB or rDB, due to DeactD in WLx, both WLy and WLx will be high for a while therefore the accessed cell via WLx, say C_x, will be overwritten by the opposite data 1 used to write the cell accessed via WLy, say C_y. When now performing $\Downarrow^{H1}(r1/r0)$, 1 will be first read from C_y and thereafter the incorrect value 1 from C_x; the fault is detected. The total test length H1-Scan based on H1 AM is $4 * N_{H1}(N)=(2N+1) * 2^{N-1} = \frac{1}{2}n(2log_2(n)+1)$, where N is the number of address lines and $n = 2^N$. Hence the time complexity of H1-Scan becomes $\Theta(n * log(n))$.

Scan can also be modified in order to detect SPRF fault (see Table 1). For such fault, RaW is the most stressful RWS. Conventional Scan does not apply RaW RWS. The new version of Scan based on RaW is shown in the second row of Table 2. The test uses the same notation as that used for H1-Scan, but now $Op1, Op2 \in \{-, wx, rx\}$ where '$-$' denotes no-operation. Note that for '$\Uparrow\ (-/w0)$' only 50% of the address space is addrssed (e.g., odd addresses). Note that RaW-Scan can also detect dynamic address decoder faults (in row decoders if fX is used and in column decoders if fY is used). However, RaW RWS is not the most stressful RWS. For RaW-Scan, any AM can be used (e.g., binary, H1). However, binary AM is preferable since it results in the shortest test length. In that case the total test length of RaW-Scan will be $5n$.

In conclusion replacing of conventional Scan with H1-Scan and/or RaW-Scan will detect, in addition to traditional faults, also advanced faults in deep-submicron memory technology like dynamic faults in the address decoder and peripheral circuits. Moreover, the new versions of Scan can detect two-operation dynamic coupling faults [9, 12] in the memory cell array (e.g., faults sensitized by WaW, RaR or RaW). Note that a special addressing method H1 is required for H1-Scan. In case the memory tests are implemented using a test program stored in ATE, it will be easy to generate a software code that provides such addressing. However, if the tests are programed in a memory BIST, special attention should be then given to the address generation hardware. A possible hardware implementation of H1 ad-

Table 2. New versions of Scan

Test	description	AM	AD	DBs	Fault coverage
H1-Scan	$\{\Uparrow^{H1}(w0/w1); \Downarrow^{H1}(r1/r0); \Downarrow^{H1}(w0/w1); \Uparrow^{H1}(r1/r0)\}$	H1	fX and fY	bDB or rDB and cDB	SWDF, SSAF, ActD and DeactD for RD and CD
RaW-Scan	$\{\Uparrow(-/w0); \Uparrow(w1/r0); \Downarrow(r0/w1); \Uparrow(-/w1); \Uparrow(w0/r1); \Downarrow(r1/w0)\}$	Any	fX	bDB or rDB	SPRF

dressing is given in [17].

The main advantages of using the H1-Scan is the test time reduction and increase in the fault coverage. H1-Scan test time is $\frac{1}{2}n(2log_2(n)+1)$, and it is able to detect not only easy-to-detect faults, but also dynamic faults in the address decoders and peripheral circuits. If separate tests will be used to target the mentioned faults, then the total test length will be much higher. Merging the test algorithms while keeping the targeted fault coverage results always in lower test time. It worth to stress that none of the conventional march test algorithms (e.g., MATS+, March C-, March G, etc) detects dynamic address decoder faults as these algorithms use binary addressing (or pseudo-random addressing); as already mentioned, dynamic address decoders faults requires specific addressing methods.

6 Silicon results

Currently we are performing a memory test experiment at Altera using the different addressing methods (e.g., binary, H1) and different data-backgrounds. The results will be available during the preparation of the final version of the paper.

However, and in order to validate the proposed approach and theory, the analysis of some test results of a test experiment done in a previous work with STMicroelectrnics will be done. The purpose of the experiment was to analyze the test coverage of traditional tests (e.g., Scan, MATS+, March C-, etc.) with some advanced tests used to cover more advanced faults in deep-submicron memory technology (e.g., March RAW, Tests for intra-word faults, etc). The test experiment was performed using four stresses; two non-algorithmic stresses and two algorithmic stresses:

- Voltages: High Voltage HV (Vdd=1.32V) and Low Voltage LV (Vdd=1.08V); i.e., 10% of the nominal voltage.
- Speed: High Speed HS (20ns) and Low Speed LS (40ns); i.e., 33% of the nominal speed.
- Addressing Directions: fast X (fX) and fast Y (fY)
- Data-backgrounds: Solid (sDB), Checkerboard (bDB), Column Stripe (cDB) and Row Stripe (rDB).

In the rest of the section, we will only focus on the analysis of test results of Scan as it is the subject of this paper. The test experiment was performed on a certain volume of embedded SRAMs (0.13 micron 512 Kbits) at 30^{o}C. The coverage results of Scan are shown in Table 3 for the four combinations of non-algorithmic stresses; e.g., HSHV means High Speed and High Voltage. The fault coverage FC in the table indicates the number of detected faulty chips, while UFs gives the number of *unique faults* each test detects. A unique fault is a fault which is only detected by a single test; none of other tests used in the experiment nor any other stress combination detect the fault.

Table 3. Coverage results of Scan

Stress	FC	UFs
HSHV	113	0
HSLV	96	2
LSHV	97	0
LSLV	85	0

Figure 1 and Figure 2 give the FC of Scan using different DBs with fX and fY respectively. In the figures, Xs/Ys denotes fX/fY with sDB, Xb/Yb denotes fX/fY with bDB, Xr/Yr denotes fX/fY with rDB and Xc/Yc denotes fX/fY with cDB. Based on the two figures one can conclude the following:

- At fast X, the highest FC is systematically obtained with bDB and/or cDB for all non-algorithmic stresses. The bDB is most stressful when fast X used; see also Table 1. Note that using fX with cDB scores better than rDB (which was not expected; see Table 1); this can be explained by the fact the memory under test is more sensitive for faults in the column decoder than in the row decoders.
- At fast X, the lowest FC is obtained with sDB at all non-algorithmic stresses.
- At fast Y, the highest FC is systematically obtained with bDB and/or cDB for all non-algorithmic stresses except for HSLV. These two DBs are the most stressful when fast Y is used; see also Table 1.
- At fast Y, the lowest FC is obtained with sDB and/or rDB at all non-algorithmic stresses except for HSLV.
- For both fast X and Fast Y, the best FC is obtained at HSHV using bDB.

In conclusion, to produce the highest FC for Scan:

- Apply a sequence of write and read operations along *columns* using an alternating 0101... pattern.
- Apply a sequence of write and read operations along *rows* using an alternating 0101... pattern.

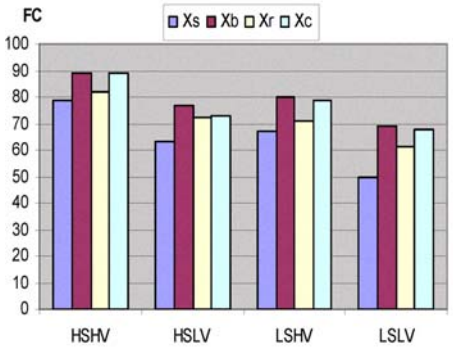

Figure 1. Effect of DBs and fX on FC of Scan

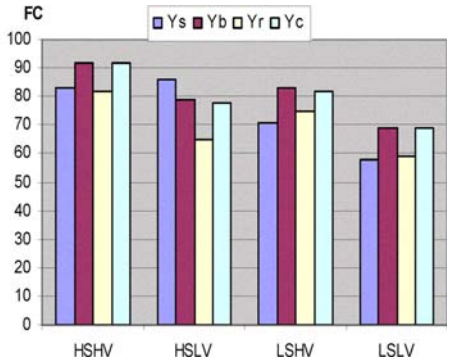

Figure 2. Effect of DBs and fY on FC of Scan

- Use HSHV to realize the highest FC and HSLV to detect more unique faults; see Table 3.

These are the most stressful patterns for the write drivers, the precharge circuits, the sense amplifiers, row decoders and column decoders, as explained in Section 4. Note that the FC can be further increased by using e.g., H1 addressing method instead of binary addressing since it will guarantee the generation of *all* sensitization address pairs required for 100% FC of address decoders delays/dynamic faults. Binary addressing does not guadrantee the generation of all required sensitization address pairs for the detection of address decoder dynamic faults [17].

7 Conclusions

In this paper, the importance of Scan test is analyzed. It has been shown that its fault coverage can be improved by targeting advanced faults in deep-submicron memory technologies (like dynamic faults in the peripheral circuits and address decoders). The improvement in the FC can be achieved by choosing appropriate algorithmic stresses such as addressing method (e.g., H1 addressing), addressing direction (e.g., Fast X) and data-backgrounds (e.g., checkerboard). Using appropriate stresses will allow not only the increase in the fault coverage, but also the detection of unique faults that cannot be detected with other march tests. Silicon results have been presented to validate the proposed stress combinations.

Acknowledgment

We thank R. Wadsworth from ST for the implementation of the test experiment and providing us with the silicon results.

References

[1] A. Allan, *et al*, "International Technology Roadmap for Semiconductors (ITRS), *Computers*, pp. 42-53, Jan. 2002.

[2] S. Hamdioui, A.J. van de Goor, J.D Reyes, M. Rodgers, "Memory Test Experiment: Industrial Results and Data", *IEE Proceedings of Computers and Digital Techniques*, pp. 1-8, V.153, Issue 1, January 2006,

[3] D.M. Wu *et al.*, "'An Optimized DFT and Test Pattern Generation Strategy for Intel High Performance Microprocessors", *In Proc. of IEEE International Test Conference*, pp. 38-47, 2004.

[4] M.S. Abadir and J.K. Reghbati, "Functional Testing of Semiconductor Random Access Memories", *ACM Computer Surveys*, **15**(3), pp. 175-198, 1983.

[5] A.J. van de Goor, "Testing Semiconductor Memories, Theory and Practice", ComTex Publishing, Gouda, The Netherlands, (second edition), 1998.

[6] R.D. Adams and E.S. Cooley, "Analysis of Deceptive Destructive Read Memory Fault Model and Recommended Testing", Records North Atlantic Test Workshop, pp. 27-32, May 1996.

[7] M. Marinescu, "Simple and Efficient Algorithms for Functional RAM Testing", In Proc. of International Test Conference, pp. 236-239, 1982.

[8] R. Dekker et. al., "A Realistic Fault Models and Test Algorithms for Static random Access Memories", IEEE Trans. on Computers,C9(6), pp. 567-572, 1990.

[9] Z. Al-Ars, A. J. van de Goor, Static and Dynamic Behavior of Memory Cell Array Spot Defects in Embedded DRAMs, IEEE Transactions on Computers, pp. 293-309, March 2003

[10] G. Harutunyan, V.A. Vardanian and Y. Zorian, "Minimal March Tests for Dynamic Faults in Random Access Memories", In Proc of ETS, pp. 43 - 48, 2006

[11] M. Azimane, *et. al*, "A new algorithm for dynamic faults detection in RAMs", *in Proc of VLSI Test Symposium*, pp. 177-182, 2005.

[12] S. Hamdioui, Z. Al-Ars, and A.J. van de Goor, "Testing Static and Dynamic Faults in Random Access Memories," Proc. Of IEEE VLSI Test Symposium, 2002, pp. 395-400.

[13] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri, M Hage-Hassan, "Dynamic read destructive fault in embedded-SRAMs: analysis and march test solution", In proc. Of IEEE European Test Symposium, pp. 140 - 145, 2004.

[14] M. Sachdev, "Open Defects in CMOS RAM Address Decoders", IEEE Design and Test of Computers, pp. 26-33, Apr.-June 1997.

[15] J. Otterstedt et al., "Detection of CMOS Address Decoder Open Faults with March and Pseudo Random Memory Tests", Proc. IEEE Int'l Test Conf., pp. 53-62, 1998

[16] L. Dilillo et al., "Comparison of Open and Resistive-Open Defect Test Conditions in SRAM Address Decoders," IEEE Asian Test Symp., pp. 250-255, 2003.

[17] S. Hamdioui, Z. Al-ars, A.J. van de Goor, "Opens and Delay Faults in CMOS RAM Address Decoders", IEEE Trans. Computers, Vol. 55, No. 12, pp. 1630-1639, Nov., 2006.

[18] R.D. Adams and E.S. Cooley, "False Write Through and Un-Restored Write Electrical Level Faults Models for SRAMs", In Proc. Of lEEE Inr. Workshop on Memory Technology, Design and Test, pp. 27-32, 1997

[19] A. Ney, *et al.*, "Slow write driver faults in 65nm SRAM technology: analysis and March test solution", *Proc. of Design Automation and Test Conference*, pp. 1-6 , 2007.

[20] A.J. van de Goor, S. Hamdioui, and R. Wadsworth, "Detecting Faults in the Peripheral Circuits and Evaluation on SRAM Tests", Proc. IEEE Int'l Test Conf., pp. 114-123, 2004.

[21] A.J. van de Goor, S. Hamdioui and Z. Alars, "The effectiveness of the scan test and its new variants"', van de Goor, A.J.; Hamdioui, S.; Al-Ars, Z.; *Proc. IEEE Int. Workshop on Memory Technology, Design and Testing*, pp. 26-31, 2004.

[22] T.R.N. Rao and E. Fujiwara, 'Error-Control Coding for Computer Systems', Prentice-Hall International Editions, Englewoods Cliffs, NJ, 1989.