Impact of Stresses on the Fault Coverage of Memory Tests

Said Hamdioui Zaid Al-Ars Ad J. van de Goor
Delft University of Technology
Computer Engineering Laboratory
Mekelweg 4, 2628 CD Delft, The Netherlands
{S.Hamdioui, Z.e.al-ars, A.J.vandeGoor}@ewi.tudelft.nl

Rob Wadsworth
ST Microlelctronics
1310 Electronics Drive
Carrollton, TX 75006, USA
Rob.Wadsworth@st.com

Abstract

Memory tests are applied in the industry using different algorithmic stresses (e.g., data-backgrounds) and non-algorithmic stresses (e.g., supply voltage). This paper presents an industrial analysis of the impact of stresses on the fault coverage (FC) of the memory tests. The experimental results show that stresses have an important impact on the FC, that the variation of the FC due to non-algorithmic stresses is higher than that of algorithm stresses, and that the non-algorithmic stresses achieve a better FC than algorithm stresses. The paper also discusses the causes behind this variation in the FC and concludes that the variation can be barely explained with the current fault models, and that this increasing variation is potentially due to partially/not modeled/understood defect mechanism in the scaled memory technologies (e.g., increase in voltage drop, in cross talk and in leakage; reduction in noise margin, etc).

1 Introduction

Much has been published on fault models and test design for memories [1, 2, 5, 7, 10, 11, 15]. The key questions are (a) how effective are the test algorithms, and (b) how does this affectiveness change with the new memory technologies? Some researchers have addressed these questions by measuring the *fault coverage (FC)* of the test algorithms. In [6] the results of testing 1024 128K$*$8 SRAM chips have been reported, using a small set of test algorithms, combined with a few stresses. The results indicated that the FC depends heavily on the used stress, such as the load on the output pins and/or the power supply voltage. In [17], test results of 1896 1M$*$4 DRAM chips have been reported; these also indicated that stresses have a profound influence on the efficiency of the test algorithms. In addition, they showed that many functional tests detect faults that can not be explained using the known fault models, which indicates the existence of other fault models. Similar conclusions have been published in [14], based on testing 3876 256K SRAM chips, as well as in [16], based on applying a large number of tests with different stresses to DRAMs.

The next questions are: what is causing this variation in the FC, due to stresses, and why it is becoming more significant? Generally speaking, the design of a test to target a certain set of faults is done independent of the to be used stresses with; e.g., MATS+ [11] was designed to detect *Stuck-at-faults* and March C- [10, 15] was designed to detect *Transition Faults* and *Idempotent Coupling Faults*, irrespective of the to be used stress combinations with the tests. The variation in the FC is due to the fact that the tests also detect some faults that are stress dependent. So far, no theoretical base exists to model defects that are stress dependent, and therefore to predict the FC for a given test.

This paper presents an industrial evaluation of the impact of the commonly used stress combinations on the FC of memory tests, and discusses the causes behind the variation in the FC due to the stresses. It is organized as follows: Section 2 defines and classifies the used test stresses; Section 3 presents memory tests and the results of their industrial evaluation; Section 4 analyzes the impact of the stresses on the FC; Section 5 discusses the causes behind the variation in the FC; Section 6 ends with the conclusions.

2 Stresses

When testing, each test is applied using several *stresses*. Such stresses can be divided into *algorithm stresses* and *non-algorithm stresses*.

2.1 Algorithm stresses

An algorithm stress specifies the way the algorithm is performed, and therefore it influences the sequence and/or the type of the memory operations. The most known algorithmic stresses are the addresses and the data-backgrounds.

Addressing stresses can be divided into the categories: (a) *Address Order*, and (b) *Address Direction*.

Proceedings of the 2005 IEEE International Workshop on Memory Technology, Design, and Testing (MTDT'05)
1087-4852/05 $20.00 © 2005 **IEEE**

The *Address Order (AO)* specifies the order in which the addressing sequence has to be performed, in either *incrementing (⇑)* or *decrementing (⇓)* If the AO is irrelevant, then it is denoted as '⇕'. Therefore AO$\in \{\Uparrow, \Downarrow, \Updownarrow\}$. Note also that AO is one dimensional; it is defined by the test algorithm, and therefore cannot be considered as an addressing stress, as it is well the case for addressing direction described next.

The *Address Direction (AD)* is the extension of the one-dimensional AO to the two dimensional space of the memory cell array. A real memory consists of a number of rows and columns (and thus also of a number of diagonals). The AD specifies the direction (i.e., rows, columns, or diagonals) that the address sequence has to be performed. The commonly used ADs in the industry consist of three types:

1. **Fast X (FX)**: Fast X addressing increments or decrements the address in such a way that each step goes to the next row.
2. **Fast Y (FY)**: Fast Y addressing increments or decrements the address in such a way that each step goes to the next column.
3. **Fast D (FD)**: Fast D addressing increments or decrements the address in such a way that each step goes to the next diagonal. Fast D is less frequently used.

A *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 (i.e., 0000.../0000...) or all 1s
Checkerboard (bDB): 0101.../1010.../0101.../1010...
Column Stripes (cDB): 0101.../0101.../0101.../0101...
Row Stripes (rDB): 0000.../1111.../0000.../1111...

2.2 Non-algorithm stresses

A non-algorithm stress does not impact the sequence and/or the type of the memory operations. However, it may have a great impact on the fault coverage. It consists of the environmental conditions, externally applied to the design under test. The most well known non-algorithmic stresses are the voltage, the timing (the clock frequency) and the temperature at which the test is performed.

3 Industrial evaluation of memory tests

This sections gives an industrial evaluation of some memory test algorithms, while considering algorithm and non-algorithmic stresses.

3.1 List of used tests and stresses

The used tests in the experiment are:

1. Scan [1]: $\{\Uparrow (w0); \Uparrow (r0); \Uparrow (w1); \Uparrow (r1)\}$
2. MATS+ [11]: $\{\Uparrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0)\}$
3. MATS++ [4]: $\{\Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0, r0)\}$
4. March C- [10, 15]: $\{\Updownarrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0); \Downarrow (r0, w1); \Downarrow (r1, w0); \Updownarrow (r0)\}$
5. March RAW[8]: $\{\Updownarrow (w0); \Uparrow (r0, w0, r0, r0, w1, r1); \Uparrow (r1, w1, r1, r1, w0, r0); \Downarrow (r0, w0, r0, r0, w1, r1); \Downarrow (r1, w1, r1, r1, w0, r0); \Updownarrow (r0)\}$
6. March SR [7]: $\{\Downarrow (w0); \Uparrow (r0, w1, r1, w0); \Uparrow (r0, r0); \Uparrow (w1); \Downarrow (r1, w0, r0, w1); \Downarrow (r1, r1)\}$
7. March SS [7]: $\{\Updownarrow (w0); \Uparrow (r0, r0, w0, r0, w1); \Uparrow (r1, r1, w1, r1, w0); \Downarrow (r0, r0, w0, r0, w1); \Downarrow (r1, r1, w1, r1, w0); \Updownarrow (r0)\}$
8. PMOVI [5]: $\{\Downarrow (w0); \Uparrow (r0, w1, r1); \Uparrow (r1, w0, r0); \Downarrow (r0, w1, r1); \Downarrow (r1, w0, r0)\}$

The above algorithms (i.e., *base tests (BTs)*) were applied using the following stresses:

- Voltages: High Voltage 'HV' ($Vdd = 1.32V$) and Low Voltage 'LV' ($Vdd = 1.08V$); i.e., $\sim \pm 10\%$ of the nominal voltage.
- Speed: High Speed 'HS' ($20ns$) and Low Speed 'LS' ($40ns$); i.e., $\sim \pm 33\%$ of the nominal speed.
- Addressings: Fast X (FX) and Fast Y (FY).
- Data-backgrounds: Solid (sDB), Checkerboard (bDB), Column Stripe (cDB) and Row Stripe (rDB).

3.2 Experimental results

A total of 256 tests [i.e., 8(the number of BTs) $\times$2(voltages) $\times$2(speeds) $\times$2(Addressings) $\times$ 4(DBs)] were implemented and applied to 0.13 micron 512 Kbits ST SRAM chips at 30^oC. Since the data base of the test results is very large, it has to be simplified for analysis purposes. Therefore we will first consider the FC of each BT with all its algorithmic stresses (i.e., ADs and DBs). The FC of a BT is the *union* of the fault coverages of its corresponding algorithm stresses. A die belongs to the union (i.e., considered detected), if at least one non-algorithm stress (i.e., HS, LS, HV or LV) of that BT found the die to be faulty. For example, Scan is implemented at HS&HV, HS&LV, LS&HV, LS&LV; each with 8 algorithmic stresses (i.e., FX-sDB, FX-bDB, FX-cDB, FX-rDB, FY-sDB, FY-bDB, FY-cDB and FY-rDB).

Table 1 shows the *unions* and the *intersections* of the 8 BTs for HS&HV, HS&LV, LS&HV and LS&LV. A die belongs to the union of two BTs if at least one of the two BTs found the die to be faulty, and belongs to the intersection of two BTs if *both* BTs found the die to be faulty. The table consists of four subtables. The first column in each subtable gives the BT number; the second column the name of

Table 1. The intersections and the unions of the base tests

High Speed (HS) Testing																					
High Voltage (HV); FC=418												Low Voltage (LV); FC=412									
#	Base Test	FC	UF	1	2	3	4	5	6	7	8	#	FC	UF	1	2	3	4	5	6	7	8
1	Scan	404	2	**404**	400	398	399	399	399	400	400	1	396	2	**396**	390	389	390	390	390	389	390
2	MATS+	407	0	411	**407**	404	403	404	400	405	405	2	399	1	405	**399**	394	394	393	392	395	393
3	MATS++	405	0	411	408	**405**	404	404	400	404	404	3	395	0	402	400	**395**	394	392	390	393	392
4	March C-	408	0	413	412	409	**408**	406	404	408	406	4	398	0	404	403	399	**398**	396	394	396	396
5	March RAW	408	0	413	411	409	410	**408**	404	408	407	5	398	0	404	404	401	400	**398**	394	396	396
6	March SR	409	2	414	416	414	413	413	**409**	406	403	6	400	3	406	407	405	404	404	**400**	393	394
7	March SS	411	0	415	413	412	411	411	414	**411**	407	7	398	0	405	402	400	400	400	405	**398**	395
8	PMOVI	408	0	412	410	409	410	409	414	412	**408**	8	398	1	404	404	401	400	400	404	401	**398**
Low Speed (LS) Testing																					
High Voltage (HV); FC=408												Low Voltage (LV); FC=410									
#	Base Test	FC	UF	1	2	3	4	5	6	7	8	#	FC	UF	1	2	3	4	5	6	7	8
1	Scan	401	1	**401**	398	399	400	399	400	399	399	1	390	1	**390**	385	384	386	386	386	386	384
2	MATS+	402	0	405	**402**	402	402	402	399	402	402	2	395	0	400	**395**	391	392	391	389	393	392
3	MATS++	403	0	405	403	**403**	403	403	400	403	403	3	391	0	397	395	**391**	391	390	386	390	390
4	March C-	404	0	405	404	404	**404**	403	401	403	403	4	398	1	402	401	398	**398**	394	391	395	395
5	March RAW	406	1	408	406	406	407	**406**	401	404	404	5	397	0	401	401	398	401	**397**	392	394	394
6	March SR	402	0	403	405	405	405	407	**402**	401	400	6	398	3	402	404	403	405	403	**398**	391	390
7	March SS	404	0	406	404	404	405	406	405	**404**	403	7	399	1	403	401	400	402	402	406	**399**	396
8	PMOVI	404	0	406	404	404	405	406	406	405	**404**	8	397	0	403	400	398	400	400	405	400	**397**

the BT. The column 'FC' lists the FC of the corresponding BT; the column 'UF' lists the *unique faults (UFs)* each BT detects. Unique faults are faults that are only detected once with a single test; e.g., at HS&HV, March SR detects 2 UFs that are not detected by any other test at HS&HV. The numbers on the diagonal (printed in bold font) give the fault coverage (FC) of the BTs, which are also listed in the column 'FC'; e.g., March SS has a FC=411 at HS&HV. The part above the main diagonal shows the intersection for each BT pair, while the part under the diagonal lists the union of each BT pair; for example, at LS&HV the union of March SS and March C- is 405 and their intersection is 403.

4 Impact of stress on the FC

This section analyzes the impact of non-algorithm and algorithmic stresses on the FC.

4.1 Impact of non-algorithm stresses

Table 2 shows the impact of non-algorithmic stresses on the FC. E.g., the second column in the table gives the FC of each of the BTs at HS&HV testing. The table clearly shows that testing at HS&HV is the most effective. It further shows that the STD= $\sqrt{variance}$ of the FCs for each BT, given in column STD, varies between 4.15 and 5.72. (Note: variance[X]=$E[X^2] - (E[X])^2$, E denotes the *mean* and X the measured FC).

Table 2. Impact of non-algorithm stresses

BT	HS		LS		STD
	HV	LV	HV	LV	
Scan	404	396	401	390	5.31
MATS+	407	399	402	395	4.38
MATS++	405	395	403	391	5.72
March C-	408	398	404	398	4.24
March RAW	408	398	406	397	4.82
March SR	409	400	402	398	4.15
March SS	411	398	404	399	5.15
PMOVI	408	398	404	397	4.49

4.2 Impact of algorithm stresses

Table 3 shows the impact of algorithmic stresses on the FC. E.g., the FC of Scan using sDB with FX and all non-algorithmic stresses (i.e., HS&HV, HS&LV, LS&HV and LS&LV) is 393. Interesting enough, the table shows that the STD's are about 50% less than those found with the non-algorithm stresses, except for Scan and March SR, which are the only march tests with *non-inverting* march elements (i.e, the state of the cell before and after each march element is the same).

Based on the data of Table 1, Table 2 and Table 3, one can conclude the following:

- For a given BT with *inverting* march elements, the STD in the FC due to non-algorithmic stresses is about two times larger than its STD due to the algorithmic stresses (Note: an inverting march element has the property that before the application of the march element the cell is in state x, and upon the completion of the march element the cell is in a complementary state $\overline{x}$). However, this STD is almost the same for

Table 3. Impact of algorithm stresses

BT	FX				FY				STD
	sDB	bDB	cDB	rDB	sDB	bDB	cDB	rDB	
Scan	393	403	396	403	394	402	391	404	4.94
MATS+	409	406	406	405	412	410	412	409	2.55
MATS++	410	406	408	411	412	407	411	409	1.98
March C-	411	408	410	414	414	412	414	417	2.65
March RAW	413	413	410	413	418	417	416	416	2.50
March SR	407	412	405	413	407	416	408	418	4.41
March SS	411	410	410	413	415	417	417	415	2.74
PMOVI	411	409	410	414	415	410	414	413	2.12

BTs with *non-inverting* march elements (i.e., Scan and March SR). The latter are the tests detecting the largest number of unique faults; see Table 1.

- Stresses have an important impact on the FC of the BTs. This is because the tests not only detect the faults they are supposed to, but also some of complex (not yet modeled) faults, which are stress dependent.
- The FC for a given BT depends on the used stresses; hence the determination of the most effective stresses is very important, especially for tests that detect unique faults.
- Testing at high speed is more effective than testing at low speed, and testing at high voltage is more effective than testing at low voltage. However, both speeds and voltages are necessary in order to detect all faults since each stress detects some unique faults.
- Fast Y addressing with sDB or rDB seems to be more effective in general.
- Using a test with various non-algorithmic stresses achieves a higher FC than using it with different algorithm stresses. That means that the non-algorithmic stresses have the capability to sensitize more faults than the algorithm stresses.

5 Discussion

Faults in memory systems are divided into memory cell array faults, address decoder faults and peripheral circuit faults (e.g., write drivers, precharge circuits, etc.). Most published work on memory testing focussed on faults in the memory cell array because the memory array occupies most of the area of a memory die [1, 2, 5, 7, 10, 11, 15]. Limited work is published on faults in address decoders [9, 12, 13, 15]. However, almost nothing is published on faults in the peripheral circuits [3, 18].

Memory cell array faults can be divided into single-cell faults and two-cell (i.e., coupling) faults. Single-cell faults are faults involving a single cell; i.e., the cell used to sensitize a fault is the same as where the fault appears. *Functional Fault Models (FFMs)* like Transition Fault and Read Destructive Fault belong to this class of faults. *Coupling Faults (CFs)* are faults involving two cells. A CF is then sensitized when considering the effect two different cells have on each others. The FFMs like State CF, Transition CF and Disturb CF belongs to that class.

The question is now: Do the stresses impact the FC (of a memory test) of the above mentioned memory cell array faults? For the non-algorithm stresses, no theoretical base exists to model such stresses and to predict the FC for a given test. For algorithm stresses, the answer is given in the following three lemmas.

Lemma 1: The algorithmic stresses (i.e., ADs and DBs) do not impact the FC of *single-cell* faults; i.e., the single-cell FC of a given BT is independent of the used algorithm stresses.

Proof: The detection of single-cell faults requires the application of the sensitizing/detection operations with the data values 0 and 1. The sequence in which the faulty cell is addressed is therefore irrelevant for such faults. Since the DBs are applied with their true and complement values, each DB guarantees that the cell will be accessed with the data values 0 and 1.

Lemma 2: The AD does not impact the FC of two-cell (i.e., coupling) faults; i.e., the two-cell FC for a certain BT is independent of the AD (i.e., FX, FY and FD).

Proof: The detection of coupling faults requires the application of the sensitizing operation (or state) to the *aggressor-cell*, and thereafter the application of the detection (or sensitization and detection) operation to the *victim-cell*. The address order (FX, FY or FD) is irrelevant as long as the aggressor and the victim cell are accessed in the correct sequence that detects the fault.

Lemma 3: the DBs do impact the FC of coupling faults.

Proof: The sensitization of a CF, for instance between two adjacent cells (say between cell C_a and cell C_v), requires the generation of the four different states of the two cells, and the application of the sensitizing operation (e.g., read or write) to the cell C_a and the detection operation to cell C_v. Using different DBs generates different states and therefore impacts the FC, as will be shown in the following example.

Consider Scan [1]: $\{\Uparrow\ (w0); \Uparrow\ (r0); \Uparrow\ (w1); \Uparrow\ (r1)\}$ and the *Read Destructive Coupling Fault (CFrd)* [7]. Two cells are said to have a CFrd when a read operation applied to the *victim cell (v-cell)* flips the data in that cell and returns an incorrect value, while the *aggressor cell (a-cell)* is in a ceratin state. The CFrd consist of four *fault primitives (FPs)* [7]: $< 1; 0r0/1/1 >_{a,v}$ (i.e., the state of the a-cell is 1, and the read 0 operation applied to the v-cell with content 0 will cause that cell to flip to 1 and the returned value by the read operation is 1), $< 0; 0r0/1/1 >$ $< 1; 1r1/0/0 >$ and $< 0; 1r1/0/0 >$.

Table 4 shows how the coverage of Scan for the four FPs of CFrd varies with the used different DBs. The CFrd is classified as a function of the relative location of the a-cell and the v-cell into:

- *Column neighbors (cn)* faults: these are CFrd between cells which are physically adjacent, and belong to the same column.
- *Row neighbors (rn)* faults: these are CFrd between cells which are physically adjacent, and belong to the same row.

In the table, a '+' denotes that Scan does detect the FP by the corresponding DB, while a '-' denotes that Scan does not; e.g., the first FP occurring between adjacent cells in the same column (cn) is detected with Scan only if the test is used with sDB or cbDB.

Table 4. Impact of DBs on the FC for Scan

Fault primitives of CFrd	sDB		bDB		cDB		rDB	
	cn	rn	cn	rn	cn	rn	cn	rn
$< 0; 0r0/1/1 >$	+	+	-	-	+	-	-	+
$< 1; 1r1/0/0 >$	+	+	-	-	+	-	-	+
$< 0; 1r1/0/0 >$	-	-	+	+	-	+	+	-
$< 1; 0r0/1/1 >$	-	-	+	+	-	+	+	-

Based on the above example we conclude the following:

- The FC for CFs strongly depends on the used DB.
- A CF that may be not detected with a certain DB can be covered with an other DB, and vice versa. E.g., for Scan, the CFrd $< 1; 0r0/1/1 >$ is not detected with sDB; however the same FP is detected with bDB.
- A test may cover all FPs of a certain CF when used with different DBs. E.g., all FPs of CFrd are covered with Scan if at least two DBs are used.

Generally speaking, when a BT is designed to target a certain CF, the sDB is considered to be used with that test. The DBs do have impact on the FC of certain CF only if the BT (designed to be used with sDB by default) does not cover *all* FPs of that CF. For example, March SS [7], designed to target all coupling faults, detects all CFs irrespective of the used DB. Therefore the variation in the FC due to DBs cannot be explained completely by the variation in the FC for CFs; i.e., other faults, which are strongly depend on the used DBs, must exist.

Based on the previous discussion, one can conclude that the algorithmic stresses barely impact the FC of single-cell and two-cell (i.e., CFs) memory cell array faults. However the algorithmic stresses have a great impact on faults like:

- *Peripheral circuit* faults: they are faults in the peripheral memory circuits (e.g., precharge circuits, write drivers, sense amplifiers).
- Address decoder *delay* faults.
- Faults caused by defect mechanism introduced with the continued scaling of the technology; these are mainly related to signal integrity issues (e.g., noise margin, signal interference, supply bounce, voltage drop, etc).

In the rest of this section, we will show how the faults in the peripheral circuits and the address decoder faults are strongly dependent on the used algorithm stresses.

5.1 Peripheral circuit faults versus stresses

Let's consider for instance the write driver. 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.

To detect a *Slow Write Driver Fault (SWDF)*[18], the worst stress scenario should be generated. The bit lines should be then forced into a certain state using a write x operation (wx, $x \in \{0, 1\}$), and then *immediately* followed with a '$w\bar{x}$' operation, applied to the next cell in the *same column* (i.e., to the same bit lines) to force the opposite state. That means that the FX addressing *must* be used either with the bDB or the rDB; otherwise the SWDF may be not detected. Note that for SWDF not only the kind of addressing and the DBs are critical, but also the type of read-write sequence used by the test; i.e., the SWDF requires *a write after a write operation* (back-to-back).

Inspecting the test algorithms industrially evaluated in this paper (see Section 3.1) reveals that only Scan satisfies the requirements for detecting SWDF's. This may explain the fact the Scan usually detects *unique* faults as it is also the case in the work presented here; see Table 1.

5.2 Address decoder delay faults versus stresses

Opens are the major cause of delays in the address decoder paths; they can cause *Address decoder Delay Faults 'ADFs'*. Figure 1(a) shows a part of CMOS address decoder with an *inter-gate* open defect [9, 13]. The address bits A_2, A_1 and A_0 are used to select the appropriate *word line 'WL'* (i.e., row). The decoding of the word lines is done using 3-input CMOS NAND gates and 2-input NOR gates, together with a buffer circuit. The signal 'Timing' is used with the NOR gates to generate the word lines with correct timing. The inter-gate open defect causes a *delay fault* in WL_1. Figure 1(b) depicts a sequence of memory accesses, sequentially accessing memory locations with a *good Word Line* '$WL0$' and a potentially *faulty Word Line* '$WL1$'. In case of an ADF, the activation and/or the deactivation of $WL1$ will be delayed, causing an *Activation Delay 'ActD'* fault and/or a *Deactivation Delay 'DeactD'* fault.

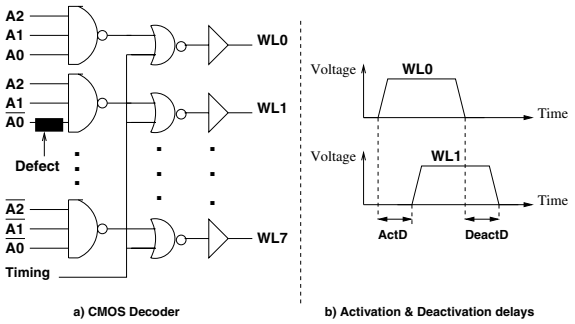

Figure 1. Address decoder delay faults

To detect the ActD an appropriate address sequence should be generated. The ActD is caused by a $0 \rightarrow 1$ transition of $\overline{A_0}$. This can be represented by the address transition for example from $WL0 \rightarrow WL1$. Due to the ActD the memory cycle involving $WL1$ may only be performed partially, which may lead to an incorrect operation. Note that the detection of ActD will then require the use of FX address direction.

In addition to the required address transition and direction, an appropriate operation sequence should be generated. The operations (i.e., write 0 to $WL0$ and write 1 to $WL1$) have then to be applied to the two addresses back-to back (i.e, immediately after each others), with *complementary data values*; that means that bDB or rDB is required.

6 Conclusions

In this paper the impact of the algorithm and non-algorithmic stresses on the FC of the memory tests has been investigated. The experimental results show that using the memory tests with non-algorithmic stresses achieves higher FC than using them with algorithm stresses; and that, for a given memory test with *inverting march elements*, the STD in the FC due to non-algorithmic stresses is about two times larger than that due to the algorithm stresses.

The variation in the FC due to the stresses (which are becoming larger with the scaling technologies) can be barely explained with the current memory fault models. The variation is potentially due to capability of the used tests to detect some not yet modeled/understood faults when they are used with specific stresses. Two examples have been discussed to show how the faults in the peripheral circuits and the address decoders are strongly dependent on the used stresses. The continue decrease of the feature sizes in deep-submicron technology will further be the source of new defects and faults that are strongly dependent on the stresses in their detection; issues like process variation causing threshold voltage deviation, increasing influence parasitics, cross talk, propagation delays, increase in supply noise and reduction in the noise margin are couple of examples.

References

[1] M. Abadir and J.K. Reghbati. Functional Testing of Semiconductor Random Access Memories. *ACM Computer Surveys*, 15(3):175–198, 1983.

[2] D. Adams. *High Performance Memory Testing*. Kluwer Academic Publishers, MA, USA, 2003.

[3] R. Adams and E.S. Cooley. False Write Through and Un-Restored Write Electrical Level Faults Models for SRAMs. *Proc. of IEEE Int. Workshop on Memory Technology, Design and Testing*, pages 27–32, 1997.

[4] M. Breuer and A. D Friedman. *Diagnosis and Reliable Design of Digital Systems*. Computer Science Press, 1976.

[5] J. De Jonge and A.J. Smeulders. Moving Inversions Test Pattern is Thorough, Yet Speedy. *In Comp. Design*, pages 169–173, 1976.

[6] S. Goto and K. Iwasaki. Experiment fault analysis of 2Mb SRAM chips. *Proc. of European Design and Test Conference*, pages 623–630, 1999.

[7] S. Hamdioui. *Testing Static Random Access Memories:Defects, Fault Models and Test Patterns*. Kluwer Academic Publishers, Boston Hardbound, ISBN 1-4020-7752-1, 2004.

[8] 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*, pages 395–400, 2002.

[9] M. Klaus and A.J. van de Goor. Tests for Resistive and Capacitive Defects in Address Decoders. *Proc. of IEEE Asian Test Symposium*, pages 31–36, 2001.

[10] M. Marinescu. Simple and Efficient Algorithms for Functional RAM Testing. *Proc. of the IEEE Int. Test Conference*, pages 236–239, 1982.

[11] R. Nair. Efficient Test Algorithms for Testing Semiconductor Random Access Memories. *IEEE Transactions on Computers*, C-28(3):572–567, 1978.

[12] S. Nakahara, et.al. Built-In Self-Test for GHz Embedded SRAMs Using Flexible Patterns Generator and New Repair Algorithm. *Proc. of the IEEE Int. Test Conference*, pages 301–307, 1999.

[13] K. Sachdev. Open Defects in CMOS RAM Address Decoders. *IEEE Design & Test of Comp.*, 14(2):26–33, 1997.

[14] I. Schanstra and A.J. van de Goor. Industrial evaluation of Stress Combinations for March Tests Applied to SRAMs. *Proc. of the IEEE Int. Test Conference*, pages 983–992, 1999.

[15] A. van de Goor. *Testing Semiconductor Memories, Theory and Practice*. ComTex Publishing, Gouda, The Netherlands, 1998.

[16] A. van de Goor and A. Paalvast. Industrial Evaluation of DRAMs SIMM Tests. *Proc. of the IEEE Int. Test Conference*, pages 426–435, 2000.

[17] A. van de Goor and J. de Neef. Industrial Evaluation of DRAMs Tests. *In Proc. of Design Automation and Test in Europe*, pages 623–630, March 1999.

[18] A. van de Goor S. Hamdioui and R. Wadsworth. Detecting Faults in the Pheripheral Circuits and an Evaluation of SRAMs. *Proc. of the IEEE Int. Test Conference*, pages 114–123, 2004.