FAME: A Fault-Pattern Based Memory Failure Analysis Framework

Kuo-Liang Cheng, Chih-Wea Wang, Jih-Nung Lee,
Yung-Fa Chou, Chih-Tsun Huang, and Cheng-Wen Wu
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 30013, ROC

Abstract

A memory failure analysis framework is developed—the Failure Analyzer for MEmories (FAME). The FAME integrates the Memory Error Catch and Analysis (MECA) system and the Memory Defect Diagnostics (MDD) system. The fault-type based diagnostics approach used by MECA can improve the efficiency of the test and diagnostic algorithms. The fault-pattern based diagnostics approach used by MDD further improves the defect identification capability. The FAME also comes with a powerful viewer for inspecting the failure patterns and fault patterns. It provides an easy way to narrow down the potential cause of failures and identify possible defects more accurately during the memory product development and yield ramp-up stage. An experiment has been done on an industrial case, demonstrating very accurate results in a much shorter time as compared with the conventional way.

1 Introduction

Memories of heterogeneous types, such as SRAM, DRAM, flash memory, etc., have become the major components in a typical system-on-chip (SOC). A large number of memory cores not only increase the design and integration complexity, but also dominate the chip area. In addition, memories have been widely used as the technology driver, i.e., they are often designed with a density that is at the extremes of the process technology. Therefore, the SOC yield is largely determined by the yield of the embedded memories. The demand in more efficient product development methodologies to provide a better yield learning curve is becoming more and more urgent, so far as reaching a profitable yield level within a short time-to-volume is concerned [1]. An effective memory failure analysis (FA) methodology thus is one of the key factors in the success of SOC products.

Conventional failure analysis approaches rely on FA engineers to identify the possible defect locations based on their experiences and the statistical information of physical defects. Usually the defect information is presented as a failure bitmap that shows the failed cells in a memory array, or as a wafer map that provides global process flaws. The failure patterns are some specific shapes formed by the failed cells. Their distributions are used to narrow down the potential defects that cause the failures [2]. Commercial memory testers and their yield analysis tools also support the automatic analysis and location of the failure patterns in the memory bitmap or wafer map [3]. In addition to failure pattern analysis, inductive fault analysis (IFA) has also been used to link the defects to functional fault models for certain semiconductor memories. Given IFA results, effective test and diagnostic algorithms can be developed [4, 5]. Automatic test generator apparently is an important tool for new memory designs and technologies that are emerging rapidly. In addition to test algorithms, diagnostic algorithms for fault type identification have also been developed [6, 7].

Neither the failure pattern based nor fault type based approach is satisfactory for failure analysis. Failure pattern information usually leads to a large number of suspects, because different faulty behaviors from different defects may lead to identical failure pattern. For each different technology, the mapping between failure patterns and defects has to be constructed from scratch, which is a time-consuming and inaccurate process. On the other hand, the fault-type based approach relies on precise fault modeling for each technology and memory. Diagnostics using fault types, although efficient, lacks topological and physical information. Critical defect information can be missing simply from fault type identification.

Previously we have presented the Memory Error Catch and Analysis (MECA) system [6] and the Memory Defect Diagnostics (MDD) system [8] to perform defect diagnostics and FA automatically. The MECA system provides test/diagnosis information based on fault types. In addition, the MDD system runs the memory FA automatically and creates the mapping table between the fault patterns and suspect defects, called the *defect dictionary*. The fault pattern approach combines the fault types of the failed cells and the failure patterns to effectively reduce the search space of suspect defects.

In this paper, we propose a memory failure analysis framework—the Failure Analyzer for MEmories (FAME)—that integrates the MECA and the MDD systems, as shown in Fig. 1. The error catch and test algorithm generation schemes of the MECA are improved to better cooperate with the ATE. The fault pattern and defect dictionary in the MDD are modeled efficiently for automatic fault pattern and defect analysis. Moreover, a GUI-based viewer is developed to facilitate the failure and fault pattern inspection on the bitmap. Combining the MECA and the MDD, the FAME further speeds up the defect diagnostics systematically for yield improvement, especially for new process technologies. Finally, the analysis of an industrial case justifies the effectiveness of the FAME.

2 Improved Error Catch Scheme and Test Algorithm Generation

In this section, an error catch scheme is presented for the effectively cooperation of the ATE and the MECA. The test algorithm generator is also improved accordingly. The MECA system is a fault-type based diagnosis system that requires March syndrome or error bitmaps of each read operation in the test [6]. For example, a March 17N test generated by the TAGS is listed as follows, which can distinguish most of the traditional functional faults, including stuck-at fault (SAF), transition fault (TF), stuck-open fault (SOF), read disturb fault

ICCAD'03, November 11-13, 2003, San Jose, California, USA.
Copyright 2003 ACM 1-58113-762-1/03/0011 ...$5.00.

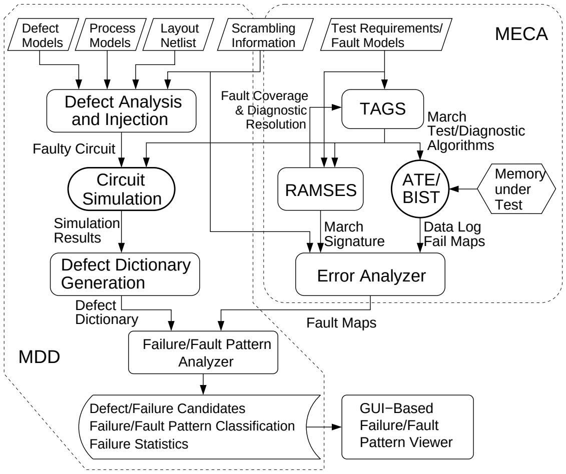

Defect Models
Process Models
Layout Netlist
Scrambling Information
Test Requirements/ Fault Models
MECA
Defect Analysis and Injection
Fault Coverage & Diagnostic Resolution
TAGS
March Test/Diagnostic Algorithms
Faulty Circuit
Circuit Simulation
RAMSES
ATE/ BIST
Memory under Test
Simulation Results
March Signature
Data Log Fail Maps
Defect Dictionary Generation
Error Analyzer
Defect Dictionary
Fault Maps
MDD
Failure/Fault Pattern Analyzer
Defect/Failure Candidates
Failure/Fault Pattern Classification
Failure Statistics
GUI−Based Failure/Fault Pattern Viewer

Figure 1. The memory failure analysis framework.

(RDF), coupling faults and address decoder fault (as defined in [9]):

$$\text{Test 1:}\quad \begin{array}{llll} \underset{-}{\Uparrow(w0)}; & \underset{E_0\ -\ E_1}{\Uparrow(r0,w1,r1)}; & \underset{E_2\ -\ E_3}{\Uparrow(r1,w0,r0)}; & \underset{E_4\ -}{\Uparrow(r0,w1)}; \\ \underset{E_5\ -\ E_6}{\Downarrow(r1,w0,r0)}; & \underset{E_7}{\Downarrow(r0)}; & \underset{E_8\ -\ E_9}{\Downarrow(r0,w1,r1)}; & \underset{E_{10}}{\Downarrow(r1)}; \end{array}$$

There are 11 read operations, resulting in a total of 11 error bitmaps ($E_0E_1\ldots E_{10}$). With appropriate manipulation of these error bitmaps, the MECA system can distinguish the fault type of each failed cell. Conventional test scheme using external ATE only records the union set of all the error bitmaps (i.e., the overall failure bitmap). We used the Credence Kalos-XP tester to demonstrate the capability of external ATE for our MECA system. In the Kalos-XP tester, two embedded Error Catch RAMs (ECR), each of 72Mbit, are used for logging the failure bitmap. They can be merged as a single 144Mbit ECR. After the test is complete, a failure bitmap of at most 144Mbit will be stored in ECR with row (X) address and column (Y) address properly specified. In the MECA system, we modified the test program, introducing an additional Z address to represent the index of read operations that are applied to memory cells. Figure 2 shows the address mapping of each error bitmap, where n is the address space of the memory under test, m is the number of the read operations, and C indicates the size of ECR. In this scheme the size limitation of memory under test is that $n \leq C/m$.

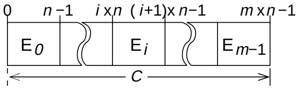


Figure 2. Mapping of error bitmaps in ECR.

To extend the capability, the compromise is to merge all the bitmaps in each March element, e.g., $E_0' = E_0 \cup E_1$ and $E_1' = E_2 \cup E_3$ for the first two March elements with reads, $\Uparrow(r0,w1,r1)$ and $\Uparrow(r1,w0,r0)$, respectively. Thus the ATE can diagnose a C-bit memory, the same capability as conventional failure-bitmap based approach. Diagnostic resolution will decrease with this modification. With improved March signature scheme, the TAGS and RAMSES can generate a new 21N test as follows:

$$\begin{array}{llllll} \underset{-}{\Uparrow(w0)}; & \underset{E_0}{\Uparrow(r0,w1,r1)}; & \underset{E_1}{\Uparrow(r1)}; & \underset{E_2}{\Uparrow(r1,w0,r0)}; & \underset{E_7}{\Uparrow(r0)}; & \\ \underset{E_4}{\Downarrow(r0,w1,r1)}; & \underset{E_5}{\Downarrow(r1)}; & \underset{E_6}{\Downarrow(r1,w0,r0)}; & \underset{E_7}{\Uparrow(r0,w1)}; & \underset{E_8}{\Uparrow(r1,w0)}; & \underset{E_9}{\Uparrow(r1)}. \end{array}$$

Note that the error bitmap represents the failed cells detected by all the read operations in a March element. Using the RAMSES and TAGS, the regenerated test has identical diagnostic resolution with previous 17N one. And the ATE can store and transfer the bitmap after each March element, keeping its maximum capacity to diagnose the memory under test. In the MECA system, effective test can be generated systematically under different test requirements, target fault models and ATE/BIST limitation.

3 Fault Pattern Modeling for The MDD

In our previous work [8], the MDD system is proposed to create the *fault patterns*, which is defined as failure patterns with fault type of each failed cell, and the defect dictionary. The automation makes the fault pattern based diagnostics adaptable for different memory designs and technologies. Here we show the modeling technique for the fault pattern and defect dictionary.

Figure 3 shows an experimental result of fault patterns using a commercial 0.25μm embedded SRAM design, with the particle defect size ranging from 250nm to 400nm, and resistances from 100$k\Omega$ to 1$G\Omega$, by using the methodology in [8]. A total of 22 fault patterns have been derived, with 18 fault patterns examples shown in the figure. The last 4 patterns (FPs 19 to 22) are similar to FPs 15 to 18 except they are row-wise ones. Their defect dictionary was derived in our previous work [8].

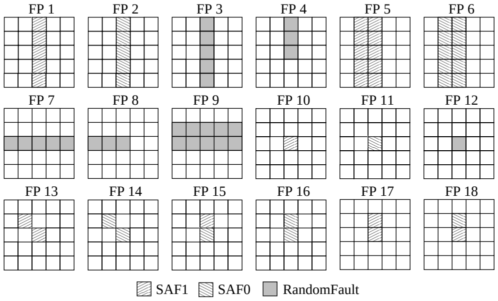


Figure 3. Examples of fault patterns.

Table 1 defines the configurations of some fault patterns in Fig. 3. The *type* can be Row, Column, DoubleCell, SingleCell, etc. The *fault_type* indicates the fault types of the failed cells. The faults along a row or a column is represented as $(F_1 : R_1, F_2 : R_2, \ldots, F_N : R_N)$, where F_k is the k^{th} fault type in the pattern for $1 \leq k \leq N$ and R_k is the number of the repetition (see Fig. 4). R_k can be omitted when it is 1. For example, a failed row consists of interleaved stuck-at-0 and stuck-at-1 faults can be described as (*SAF0, SAF1*). For fault patterns with multiple columns, rows or cells (e.g., FP5, FP6, or FP9), cascaded fault types can be used. In Table 1, FP5 is defined as {(*SAF1*), (*SAF1*)} with row width of 2. There are several optional parameters: *base* denotes the fault type of the base cell where the defect is injected. For some particular patterns, the fault type of the base cell is different from that of other faulty cells, e.g., a SAF1 row with the base cell of SAF0. In addition, we also define *row_fault_type* and *column_fault_type* to describe Cross-like fault pattern. Irregular or user-defined pattern is also supported by the FAME with geometry expandability. Unrecognized fault can be represented as March syndrome directly. Our failure analysis framework can process fault patterns even with unrecognized fault types because the defect dictionary is generated automatically, regardless the fault type is recognized or not. When a fault pattern is found, *defect* parameter denotes its defect candidates with probabilistic weighting. Currently we support three possible defect types: short, open and missing contact, which are described by "+", "–" and "!", respectively.

Table 1. The configurations of fault patterns.

# FP1	# FP5

type = Column
row_width = 1
tolerance = 0%
fault_type = (SAF1)
base = SAF1
defect = {GND+BLb:0.11, VDD+BL:0.09}
	type = Column
	row_width = 2
	tolerance = 0%
	fault_type = {(SAF1), (SAF1)}
	base = SAF1
	defect = {WL+WL:0.13}

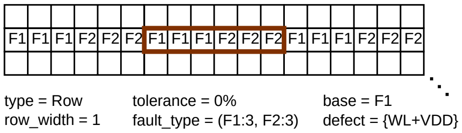

F1 F1 F1 F2 F2 F2 F1 F1 F1 F2 F2 F2 F1 F1 F1 F2 F2 F2
type = Row
row_width = 1
tolerance = 0%
fault_type = (F1:3, F2:3)
base = F1
defect = {WL+VDD}

Figure 4. A fault pattern example.

In addition to probability of defect candidates (see Table 1), fault patterns can have priority during the analysis. For example, an FP1 pattern can also be recognized as *m* FP10 along a faulty column of *m* cells. In the FAME framework, the fault pattern with more failed cells has higher priority due to higher probability by default. Extreme cases such as the selection of an FP1 or *m* FP10s are performed automatically. But ambiguous situations (e.g., a pattern of an FP5 can also be the combination of two FP1s) require user-defined priorities or manual inspection. Note that even the resultant fault patterns have unrecognized fault types (signatures) from conventional functional faults, defect-level diagnostics is still effective using our methodology due to the fault pattern and defect dictionary modeling.

4 Fault/Failure Pattern Viewer

Although there is a high degree of automation, experienced engineer is always the key in the memory failure analysis framework. We developed a GUI-based viewer to facilitate the manual debugging process (see Fig. 5). Physical geometry of the memory array can be displayed with memory architecture and scrambling information. Failed cells with their fault types are plotted and categorized using different colors. In addition, both failure and fault patterns can be visually highlighted with their suspect defect information. With the failure/fault pattern viewer, the topological defect information, such as boundary weakness, etc., can be effectively analyzed by memory designers and FA engineers for further yield improvement.

5 Experimental Results

We have done an experiment using industrial single-port SRAM chips of size 64K×12. The memory array consists of four banks, arranged as two by two in physical layout. Each bank has 512 rows and 384 columns.

To examine the effectiveness of the FAME, we applied 2 different 17*N* March algorithms with the same fault diagnostic resolution. The two algorithms are labeled as Test 1 (as discussed in Sec. 2) and Test 2 [6] which is listed as follows: $\Uparrow(w0);\Uparrow(r0,w1,r1);\Uparrow(r1);\Uparrow(r1,w0,r0);\Uparrow(r0);\Downarrow(r0,w1,r1);\Downarrow(r1);\Downarrow(r1,w0,r0);\Uparrow(r1)$. Table 2 shows the statistical results of one memory chip. In the first iteration, a total of 40 fault types are detected both for Test 1 and Test 2 (see Table 2(a)). However, the numbers of failed cells are inconsistent. By inspecting the failed cells with identical fault types, the fourth column of Table 2(a) shows that 9 faults are coincident in the two tests. They are SAF0, SAF1, RDF0, SOF and 5 sub-type idempotent coupling faults (CFid) (including CFid(↓;0)$_s$, CFid(↓;0)$_l$, CFid(↓;1)$_s$, CFid(↑;0)$_s$, and CFid(↑;1)$_l$, where the subscript 's' or 'l' represents that the address of aggressor cell is smaller or larger than that of victim cell [6]), as shown in Table 2(c).

On average about 62.5% of the failed cells are recognized as one of these 9 faults. There are also failed cells with one particular fault type in Test 1 but recognized as another fault in Test 2. The unmatched faults are about 12.8% (see the fifth column in Table 2(a)) on average. After all, failed cells with unrecognized fault types are about 25.9% and 23.5%, respectively. Since the inversion coupling fault (CFin) and state coupling fault (CFst) are not found in these memory chips, they can be removed from the target fault set. The MECA can regenerate a new 15*N* test ($\Uparrow(wa);\Uparrow(ra,wb,rb);\Uparrow(rb);\Uparrow(rb,wa);\Uparrow(ra);\Downarrow(ra,wb);\Downarrow(rb);\Downarrow(rb,wa,ra);\Downarrow(ra);$) with the same diagnostic resolution, reducing 11.8% test time. Therefore, test/diagnostic algorithms can be optimized for test efficiency.

Table 2. Statistics of the fault type analysis.

(a) With original target fault set.

		Total	Matched	Unmatched	Unrecognized
Test 1	# Types	40	9	15	24
	# Cells	21848 100%	13419 61.4%	2776 12.7%	5653 25.9%
Test 2	# Types	40	9	14	25
	# Cells	21106 100%	13419 63.6%	2734 12.9%	4953 23.5%

(b) With additional linked CFid.

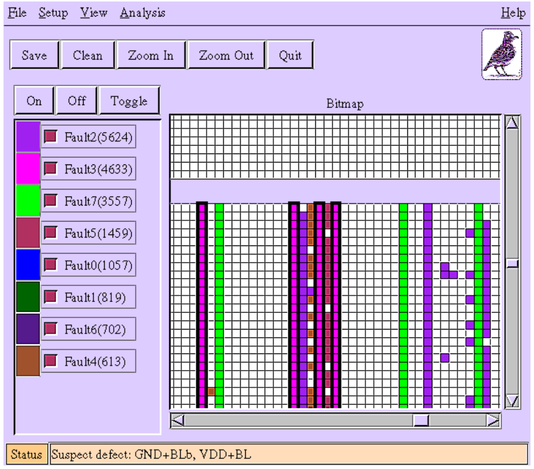
		Total	Matched	Unmatched	Unrecognized
Test 1	# Types	40	13	22	17
	# Cells	21848 100%	17333 79.3%	1392 6.4 %	3123 14.3%
Test 2	# Types	40	13	23	16
	# Cells	21106 100%	17333 82.1%	694 3.3%	3079 14.6%

(c) List of matched faults with failed cell count.

SAF0	SAF1	RDF0	SOF	CFid (5)	CFid-CFid (4)	Total
5182	4698	206	522	2811	3914	17333
29.9%	27.1%	1.2%	3.0%	16.2%	22.6%	100%

Unmatched faulty behavior among different tests can come from the aliasing of March signatures for the tests of insufficient diagnostic resolution or impractical fault modeling. For example, a failed cell with SAF1 behavior in a MSCAN algorithm ($\Uparrow(w0);\Uparrow(r0);\Uparrow(w1);\Uparrow(r1)$) may be actually a complex coupling fault. But these two faults are indistinguishable because they have identical March signature in this short test. To further inspect the faulty behavior of these unmatched failed cells, we added linked faults of CFin and CFid to our target fault set. The MECA then reported that 4 previously unmatched fault types are identified as CFid-CFid linked faults in both tests. Thus a total of 3914 failed cells are recognized (see Table 2(c)). Finally, about 14.3% (in Test 1) and 14.6% (in Test 2) are unrecognized faults in this experiment. Approximately 80% of failed cells have matched faulty behavior (see Table 2(b)) in both tests. Using the MECA, the confident level of fault identification can increase efficiently with multiple tests. Note that the summation of matched, unmatched and unrecognized fault number may be not coincident with the total fault number, because for a specific fault type, part of the failed cells may have unmatched faulty behavior for different tests. Therefore, that fault type will be counted in both matched and unmatched groups. In addition, Test 1 detects more failed cells than Test 2, because of its $\Uparrow(r0,w1)$ element. There are 731 CFids detected by exclusively this March element. The MECA helps to identify the importance of $\Uparrow(r0,w1)$ element to detect the weakness of this specific memory design. With the help of the failure/fault viewer, the distribution of each fault type can be observed and analyzed.

In addition, Table 3 lists the statistics of the fault patterns. The five chips fall into two categories. Chips 3, 4 and 5 have only one fault pattern, i.e., FP7, which is a failed row with random faults. The faulty behavior is that the failed cells have incorrect read-out for almost all the read operations, which matches the behavior of the random fault. In addition, some failed cells behaved irregularly. After the analysis of their physical location by the viewer, we found out that these cells are all located at local boundary of the physical memory array.



Suspect defect: GND+BLb, VDD+BL

Figure 5. Snapshot of the fault/failure pattern viewer.

Table 3. Fault pattern analysis results.

	Suspect defects	Chip 1		Chip 2		Chips 3–5	
		#FP	# Cell	#FP	# Cell	#FP	# Cell
FP1	2	6	3072	10	5120	—	—
FP2	2	8	4096	16	8192	—	—
FP3	1	—	—	45	23040	—	—
FP4	2	25	12019	15	7434	—	—
FP5	1	—	—	5	5120	—	—
FP6	1	—	—	2	2048	—	—
FP7	2	—	—	—	—	1	384
FP10	6	2	2	18	18	—	—
FP11	6	3	3	13	13	—	—
FP12	3	175	175	896	896	—	—
FP13	1	90	180	—	—	—	—
FP14	1	114	228	—	—	—	—
FP15	1	7	14	4	8	—	—
FP16	1	86	172	23	46	—	—
FP17	1	68	136	30	60	—	—
FP18	1	7	14	4	8	—	—
FP19	1	14	28	2	4	—	—
FP20	1	17	34	1	2	—	—
FP21	1	83	166	—	—	—	—
FP22	1	73	146	—	—	—	—
Unknown	—	—	1352	—	4706	—	—
Total	—	—	21848	—	56715	—	384

On the other hand, Chips 1 and 2 had a large amount of failed columns. With the MDD, the failed columns can be further classified into FPs 1 to 6. For example, an FP1 has only 2 suspect defects, instead of 9 (candidates of FP1∪FP2∪FP3∪FP4∪FP5∪FP6) for column failure patterns. In addition, the single-cell failure pattern has 15 defect candidates. However, our framework can distinguish them as FP10, FP11 and FP12, reducing the searching space to 6, 6, and 4 candidates, respectively. The failed cells that do not belong to any of the fault patterns are labeled as unknown. About 6.19% and 8.29% of the failed cells in Chip 1 and Chip 2 are unknown. Multiple defects in a single cell, particle out of the predefined distribution may be the cause of unknown fault patterns. With the help of failure viewer, FA engineers can further analyze these unknown fault patterns. The result can be fed back to refine the defect modeling, improving the framework.

6 Conclusions

Combining the MECA, MDD, and viewer, we have proposed an integrated framework—the Failure Analyzer for MEmories (FAME). With proposed error catch scheme for commercial ATE, an experiment has been done on an industrial case to demonstrate accurate results for the diagnostics and debugging of memories. Fault types not only can be identified, but also can be justified by our approach systematically. The target fault set can be further improved effectively for specific memory design and technology. Test/diagnostic algorithms can thus be optimized by the FAME accordingly. In addition, fault pattern based diagnosis facilitates the categorization of the failed chips, simplifying the defect diagnostics efficiently. Our fault pattern modeling technique is flexible and extensible. The sophisticated failure/fault pattern viewer facilitates the defect inspection. Using the information of fault type and fault pattern together, cost-effective defect identification and yield improvement can be achieved with the FAME.

Our future work includes the improvement of the pattern recognition for overlapped and inexact patterns, and the improvement of the methodology to create more accurate and realistic defect dictionary for higher test/diagnostics quality.

Acknowledgment

The authors would like to thank Frank Huang and Hong-Tzer Yang of Spirox Co., Hsinchu, as well as Chuang Cheng and Kevin Chiu of Faraday Technologies, Hsinchu, for their assistance in our experiments.

References

[1] Y. Zorian, "Embedded infrastructure IP for SOC yield improvement", in *Proc. IEEE/ACM Design Automation Conf. (DAC)*, New Orleans, June 2002, pp. 709–712.

[2] J. Segal, A. Jee, D. Lepejian, and B. Chu, "Using electrical bitmap results from embedded memory to enhance yield", *IEEE Design & Test of Computers*, vol. 15, no. 3, pp. 28–39, May 2001.

[3] M. A. Merino, S. Cruceta, A. Garcia, and M. Recio, "SmartBit^{TM}: bitmap to defect correlation software for yield improvement", in *Advanced Semiconductor Manufacturing Conference and Workshop, IEEE/SEMI*, Boston, Sept. 2000, pp. 194–198.

[4] A. J. van de Goor and B. Smit, "Generating march tests automatically", in *Proc. Int. Test Conf. (ITC)*, 1994, pp. 870–878.

[5] C.-F. Wu, C.-T. Huang, K.-L. Cheng, and C.-W. Wu, "Simulation-based test algorithm generation for random access memories", in *Proc. IEEE VLSI Test Symp. (VTS)*, Montreal, Apr. 2000, pp. 291–296.

[6] C.-F. Wu, C.-T. Huang, C.-W. Wang, K.-L. Cheng, and C.-W. Wu, "Error catch and analysis for semiconductor memories using March tests", in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, Nov. 2000, pp. 468–471.

[7] D. Niggemeyer and E. Rudnick, "Automatic generation of diagnostic March tests", in *Proc. IEEE VLSI Test Symp. (VTS)*, Marina Del Rey, California, Apr. 2001, pp. 299–304.

[8] C.-W. Wang, K.-L. Cheng, J.-N. Lee, Y.-F. Chou, C.-T. Huang, C.-W. Wu, F. Huang, and H.-T. Yang, "Fault pattern oriented defect diagnosis for memories", in *Proc. Int. Test Conf. (ITC)*, Charlotte, Sept. 2003 (to appear).

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