Error Catch and Analysis for Semiconductor Memories Using March Tests

Chi-Feng Wu, Chih-Tsun Huang, Chih-Wea Wang,
Kuo-Liang Cheng, and Cheng-Wen Wu
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 30013

Abstract

We present an error catch and analysis *(ECA) system for semiconductor memories. The system consists of a test algorithm generator called TAGS, a fault simulator called RAMSES, and an error analyzer (ERA). We use TAGS to generate a set of test algorithms of different lengths and diagnostic resolutions for the memory under test, and use RAMSES to generate the* March dictionary *for each test algorithm. With the March dictionaries, ERA is able to support March algorithms for easy diagnosis of faulty RAMs. Legacy test algorithms also can be reused. When integrated with a RAM tester, our ECA system can generate RAM bitmaps that are similar to the RAM layout. The bitmaps provide detail information about the error locations and faults causing the errors. Based on the information, diagnosis of the RAM chips for yield and reliability improvement can be done more easily.*

1. Introduction

RAMs are continuing to play an important role in the semiconductor industry. The booming markets of computer, communications, and consumer electronics are intensifying the need for bigger and faster semiconductor memories to handle the rapidly increasing volume of audio/video data. High capacity and high density, however, brings challenges to the memory designers as well as manufacturers. Yield is the primary concern—it drops due to higher failure probability caused by increased capacity and density. Traditionally, dedicated memory testers have been used to test the chips, locate the errors, and perform repair analysis. The process and equipments are mainly designed for back-end volume production, so the entire test flow provides only very limited information to the interests of the memory designers or process engineers, who care about design flaws, reliability, and yield. Bitmaps generated by the tester normally provides only the locations of the faulty cells. The engineers have to figure out possible causes of the errors by manual analysis.

The diagnostic test algorithms can provide more information, i.e., in addition to error location, fault type can be identified. These test algorithms are usually derived for a certain set of fault models, either classic fault models [1, 2] or realistic fault models [3]. Although some good diagnostic test algorithms have been derived in the past [1–3], a systematic approach to generating the test algorithms and bitmaps and integrating them into the test flow for easy diagnosis remains to be seen.

This paper describes an error catch and analysis (ECA) system. It is more powerful and flexible than a traditional diagnostic test. Our purpose is to categorize the errors from a given test rather than to only locate the faults. With the ECA system, the test requirement specification is flexible, and trade-offs can be made between test lengths and diagnostic resolution. Error analysis is done off-line, so it can easily be integrated into the existing testers. The ECA system consists of a test algorithm generator called TAGS [4], a memory fault simulator called RAMSES [5], and an error analyzer called ERA. Given the test requirements specified by the user, TAGS generates a set of test algorithms with different lengths and diagnostic resolutions, and RAMSES reports the fault coverage figures and generates a *March dictionary* for each test algorithm. After applying a March test, the tester will report the error data log, which is forwarded to ERA for producing the bitmaps. The ECA system has been integrated with a commercial tester and has generated useful bitmaps that helped memory designers to identify design flaws of their products.

2. Fault Models and Definitions

Several popular RAM fault models are used to illustrate our methodology, including stuck-at fault (SAF), address decoder fault (AF), transition fault (TF), inversion coupling fault (CFin), idempotent coupling fault (CFid), and state coupling fault (CFst) [6].

Each of the fault models can be expressed in detail by its explicit sub-types if exist. When error catch and analysis is desirable, faults should be defined as detail as possible. For example, a SAF can be expressed explicitly by whether it is a stuck-at-0 (SA0) or a stuck-at-1 (SA1). A coupling fault can be specified explicitly by the state of the coupling cell (aggressor), the state of the coupled cell (victim), and the faulty value. For example, $< 0; 0/1 >$ is a state coupling fault with aggressor cell being 0 and victim cell being forced from 0 to 1. For the ease of discussion in this paper, we give names to sub-types as listed in Table 2. Agr is the state of aggressor. Vtm is the state of the victim in the form of fault-free/faulty. Addr is the address relation of the aggressor and the victim, e.g., A $<$ V denotes the address of aggressor is less than that of victim.

The most widely used test algorithm for memories is the March test. Fig. 1 shows the March C– as an example, which consists of six March elements, denoted by $M_0 \cdots M_5$. Each march element contains one or more memory operations with the given address orders. $E_0 \cdots E_9$ are defined for error analysis and are explained later in this section. A March test algorithm is designed for detecting a set of target fault models. For example, March C– detects all of the SAF, AF, TF, CFin, CFid, and CFst. During the test procedure, an error is detected whenever the result of a memory operation is different from the fault free value. An error is recorded by its address, failing operation, and data syndrome (the bit positions and failing values).

Name	Agr	Vtm	Addr
SAF_0	-	1/0	-
SAF_1	-	0/1	-
TF_0	-	$\downarrow$/1	-
TF_1	-	$\uparrow$/0	-
$CFin_0$	$\downarrow$	$\updownarrow$	$A < V$
$CFin_1$	$\downarrow$	$\updownarrow$	$A > V$
$CFin_2$	$\uparrow$	$\updownarrow$	$A < V$
$CFin_3$	$\uparrow$	$\updownarrow$	$A > V$
$CFst_0$	0	1/0	$A < V$
$CFst_1$	0	1/0	$A > V$
$CFst_2$	0	0/1	$A < V$
$CFst_3$	0	0/1	$A > V$
$CFst_4$	1	1/0	$A < V$
$CFst_5$	1	1/0	$A > V$

Name	Agr	Vtm	Addr
$CFst_6$	1	0/1	$A < V$
$CFst_7$	1	0/1	$A > V$
$CFid_0$	$\downarrow$	1/0	$A < V$
$CFid_1$	$\downarrow$	1/0	$A > V$
$CFid_2$	$\downarrow$	0/1	$A < V$
$CFid_3$	$\downarrow$	0/1	$A > V$
$CFid_4$	$\uparrow$	1/0	$A < V$
$CFid_5$	$\uparrow$	1/0	$A > V$
$CFid_6$	$\uparrow$	0/1	$A < V$
$CFid_7$	$\uparrow$	0/1	$A > V$
AF_0	-	-	$A < V$
AF_1	-	-	$A > V$
SOF	-	-	-

Table 1: Fault names and its meaning.

$$\begin{array}{cccccc} M_0 & M_1 & M_2 & M_3 & M_4 & M_5 \\ \Updownarrow(w0) & \Uparrow(r0,w1) & \Uparrow(r1,w0) & \Downarrow(r0,w1) & \Downarrow(r1,w0) & \Updownarrow(r0) \\ E_0 & E_1\ E_2 & E_3\ E_4 & E_5\ E_6 & E_7\ E_8 & E_9 \end{array}$$

Figure 1: The March C– algorithm.

An error bitmap stores the locations of errors for the memory unit under test. After applying all test patterns, all faulty cells are recorded in an error bitmap, which is then passed to the laser repair stage if the memory is repairable. Process designers and memory designers also use the error bitmaps to find out possible flaws to improve the yield and reliability of their products.

The error bitmaps can also be generated in a section of the test pattern. In Fig. 1, $E_0 \cdots E_9$ represent the error bitmaps generated with respect to the the specific memory operations. For example, E_3 is the error bitmap for the read operation in M_2. The detection capability for the write operation depends on the memory architecture. For a single port memory, E_0, E_2, E_4, E_6, and E_8 in Fig. 1 will always be empty. For a two port memory with a *write-through* mode, these error bitmaps record write-through errors.

The error bitmap for all faulty cells is defined as

$$E_{all} = (E_0 \cup E_1 \cup \ldots \cup E_{N-1}), \tag{1}$$

where N is the number of read/write operations in the March test. The complement of an error map, E_n, is defined as

$$\overline{E_n} = (E_{all} - E_n). \tag{2}$$

Fault dictionary is a data base constructed for logic-level diagnosis [7]. Fault diagnosis based on fault dictionaries is also called the *cause-effect* analysis. Here we propose a similar analysis data base for diagnosing memory faults called the March dictionary.

The March dictionary is generated by memory fault simulation. With the dictionary recording capability added into the simulation procedure of RAMSES [5], it can generate the March dictionary for a March test.

Table 2 shows a March dictionary of a 11N March test:

$$\Updownarrow(w0)\Uparrow(r0,w1)\Updownarrow(r1)\Uparrow(r1,w0)\Downarrow(r0,w1)\Downarrow(r1,w0)\Updownarrow(r0), \tag{3}$$

which is generated for SAF, CFin, and CFst by the methodology proposed later in Section 4. For each fault model, the corresponding *March signature* indicates the response of the March test in each error bitmap. In a March signature, there is a 1 in the column position if this fault is detected in the corresponding error bitmap; otherwise there is a 0. For example, stuck-at-1 is detected in E_1, E_6, and E_{10}, so its March signature is $\langle 01000010001 \rangle$. We can also use the fault dictionary by column. For each error bitmap, there is a 1 if the corresponding fault can be detected; otherwise there is a 0. For example, error bitmap E_1 contains errors caused by SAF_1, $CFin_2$, $CFst_3$, and $CFst_6$.

Fault/Error Bitmap	$E_0E_1E_2E_3E_4E_5E_6E_7E_8E_9E_{10}$
SAF_0	0 0 0 1 1 0 0 0 1 0 0
SAF_1	0 1 0 0 0 0 1 0 0 0 1
$CFin_0$	0 0 0 0 1 0 0 0 0 0 1
$CFin_1$	0 0 0 0 0 0 1 0 1 0 0
$CFin_2$	0 1 0 0 0 0 0 0 1 0 0
$CFin_3$	0 0 0 1 1 0 1 0 0 0 0
$CFst_0$	0 0 0 0 1 0 0 0 0 0 0
$CFst_1$	0 0 0 0 0 0 0 0 1 0 0
$CFst_2$	0 0 0 0 0 0 1 0 0 0 1
$CFst_3$	0 1 0 0 0 0 0 0 0 0 1
$CFst_4$	0 0 0 1 0 0 0 0 1 0 0
$CFst_5$	0 0 0 1 1 0 0 0 0 0 0
$CFst_6$	0 1 0 0 0 0 0 0 0 0 0
$CFst_7$	0 0 0 0 0 0 1 0 0 0 0

Table 2: March dictionary example.

For word-oriented memories, the fault types should be further classified by explicitly specifying the syndrome (bit position). For a 4-bit word-oriented memory, SAF_0 is extended to $SAF_{0<0001>}$, $SAF_{0<0010>}$, $SAF_{0<0100>}$, and $SAF_{0<1000>}$.

Diagnostic resolution is defined as the ratio of distinguishable faults and all detectable faults. In general, two faults are distinguishable if they have different March signatures.

3. Error Catch and Analysis

Keeping the data log of memory testers, the error bitmaps $E_0 \cdots E_N$ can be obtained by parsing the data log. Error analysis is a procedure which takes error bitmaps and the March dictionary as inputs and generates fault bitmaps which contain the fault locations and the corresponding fault types.

The error catch and analysis (ECA) system is shown in Fig. 2. The main components are 1) RAMSES—memory fault simulator, 2) TAGS—test algorithm generator, and 3) ERA—error analyzer. For a unit under test (UUT), we have a user-defined test requirements including target fault models, fault coverage, diagnostic resolution, and test length. RAMSES evaluates the fault coverage, diagnostic resolution, and constructs the March dictionary for a March test. TAGS generates a March test based on RAMSES results to meet the test requirements.

After applying the March test, the data log of error detections are forwarded to ERA. ERA converts the data log to form the error bitmaps $E_0 \cdots E_N$, then generates fault bitmaps according to the error bitmaps and the March dictionary.

An example is used to illustrate the ECA procedure. Given the target faults, SAF, CFin, and CFst, and an unlimited test length, TAGS generates an 11N March test with 100% diagnostic resolution as shown in Eq. 3. Assume the UUT is a 1-bit single port RAM with a 10×10 cell array, and after parsing the tester data log, the error maps are generated as in Fig. 3. Error maps for write operations, E_0, E_2, E_5, E_7, and E_9 are always empty.

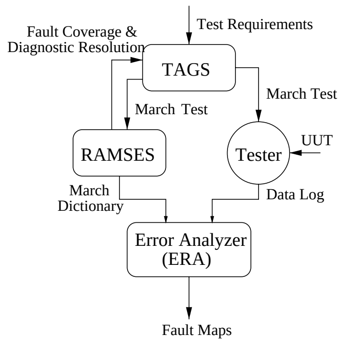

Figure 2: Error catch and analysis system.

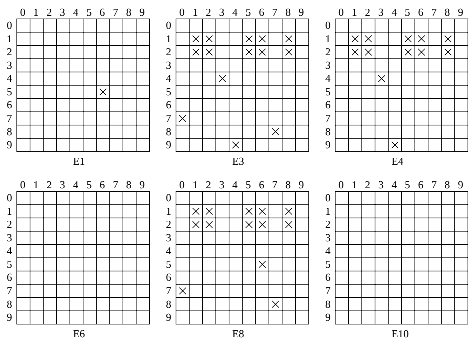

Figure 3: A brief example: error bitmaps.

The March dictionary of this case is shown in Table 2. For each fault model, the fault bitmap is generated by processing the error bitmaps with intersect operations. According to the March dictionary, when there is a 1 for the corresponding E_n, the bitmap is used; otherwise the complemented bitmap, i.e., $\overline{E_n}$, is used. For example, the fault bitmap of SAF$_0$ can be generated by

$$\begin{aligned} F_{SAF_0} &= \overline{E_0} \cap \overline{E_1} \cap \overline{E_2} \cap E_3 \cap E_4 \cap \\ & \quad \overline{E_5} \cap \overline{E_6} \cap \overline{E_7} \cap E_8 \cap \overline{E_9} \cap \overline{E_{10}}. \end{aligned} \tag{4}$$

When an error bitmap is empty, i.e., $E_n = \emptyset$, then $\overline{E_n} = E_{all}$ according to Eq. 2. From the definition of E_{all} in Eq. 1, the intersection of E_{all} with any E_n equals to E_n. Therefore, empty error bitmaps are redundant and can be removed. For example, Eq. 4 can be reduced to

$$F_{SAF_0} = \overline{E_1} \cap E_3 \cap E_4 \cap E_8. \tag{5}$$

Other target fault bitmaps, SAF$_1$, CFin$_0$ $\cdots$ CFin$_3$, CFst$_0$ $\cdots$ CFst$_7$, can be generated in a similar way by their specific equations according to the March dictionary. The resulting fault bitmaps are shown in Fig. 4 except empty bitmaps.

Like E_{all}, we can stack fault bitmaps to generate a F_{all} bitmap. As shown in Fig. 5, F_{all} provides detail fault models for each error, and at the same time provides fault statistics.

The limitation of the March test is that it can locate the coupled cell but not the coupling cell of a coupling fault. Therefore, when the location of both the coupling cell and coupled cell is desired, the fault bitmaps of coupling faults can be fed back to the tester to do non-March test, e.g., GALPAT [6], for further diagnosis.

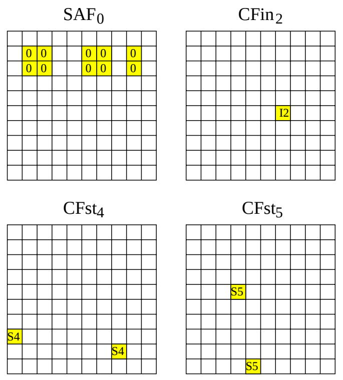

Figure 4: A brief example: fault bitmaps.

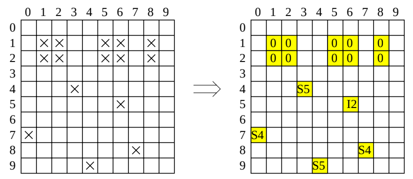

Figure 5: A brief example: E_{all} and F_{all}.

The computation time of ERA is proportional to the number of errors in the unit under test (UUT). Assume the number of errors that occurs in the UUT is N, the number of target fault models is m, and the length of the March test is p. The worse-case time complexity of ERA is $O(pmN)$. The values of p and m are constant for a given March test to detect a given set of fault models. Therefore, the complexity of the ERA is linear, i.e., $O(N)$.

4. Test Algorithm Generation for the ECA

One of the advantage of our ECA procedure is that it does not require a specific test/diagnosis algorithm. Existing test procedures and test programs can be re-used. However, production test algorithms have usually been optimized for test only, the diagnostics resolution may not be high enough to meet the ECA requirements. We propose an automatic test algorithm generation methodology for the requirement of a higher diagnostic resolution.

The test algorithm generation is based on TAGS (test algorithm generation by simulation) in our previous work [4]. After the complete test is generated, we continue the TAGS algorithm but only insert read operations and apply filter options. A user-specific test can also be used for read insertion.

We illustrate the generation by several popular fault models. The target fault models are SAF, TF, AF, SOF, CFin, CFid, and CFst. For these target faults and unlimited test length, TAGS generates an 11N test that detects 100% of the above faults. Beginning with the 11N test, i.e., $\Uparrow(w0)$ $\Uparrow(r0,w1)$ $\Uparrow(r1,w0)$ $\Downarrow(r0,w1)$ $\Downarrow(r1,w0,r0)$ $\Uparrow(r0)$, the test generation procedure for the ECA ends with a 17N algorithm, i.e., $\Uparrow(w0)$ $\Uparrow(r0,w1,r1)$ $\Uparrow(r1)$ $\Uparrow(r1,w0,r0)$ $\Uparrow(r0)$ $\Downarrow(r0,w1,r1)$ $\Uparrow(r1)$ $\Downarrow(r1,w0,r0)$ $\Uparrow(r0)$ The diagnostic resolution is 0.996. It is not 100% due to the behavior of SAF and TF. SAF$_0$ and TF$_1$ are indistinguishable if the initial background is 0; SAF$_1$ and TF$_0$ are indistinguishable if the initial background is 1.

We also use two popular March tests as the user specified tests, March X (6N), and IFA9N [8].

These algorithm can be used in test algorithm generation and result in some points of diagnostic resolutions. The comparison is shown in Fig. 6, which shows the trade-off on test length and the diagnostic resolution. When the diagnostic resolution requirement is not high, e.g., only certain fault bitmaps are of interest, a shorter and cost-effective test algorithm is preferred.

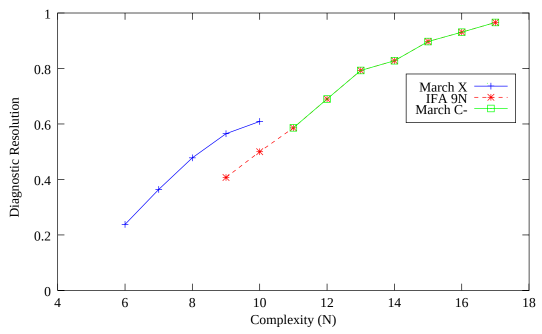

Figure 6: Diagnostic resolution for test algorithms.

The test generation algorithm proposed here is only a local optimal solution for a given test, either generated by TAGS or specified by the user. An optimal (shortest) test algorithm for a given test requirements can be approached by iteratively running with all test algorithms in the TAGS test library, which is generated for a given set of fault models during test generation procedure.

5. Experimental Results

We have applied our error catch and analysis methodology on a 16Kx8 embedded SRAM (FS80A020) test chip which is tested by a commercial tester (Credence SC212). Through the system illustrated in Fig. 2, fault bitmaps have been generated.

The address/data scrambling topology has been provided by the memory designer. Through address remapping, we have been able to generate the fault bitmaps with the floorplan and physical locations of memory cells, including the block boundaries and gaps. Fault bitmap examples for SAF_0 and $CFin_2$ are shown in Fig. 7 and Fig. 8, respectively. These fault bitmaps significantly help memory designer visually in order to investigate the possible cause of errors.

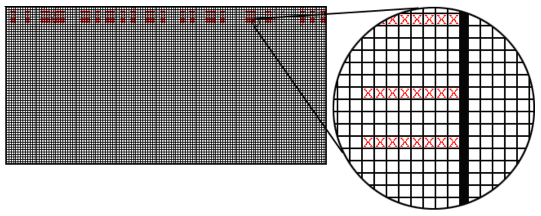

Figure 7: SAF_0 fault bitmap of an 16Kx8 embedded SRAM (FS80A020) test chip.

After the ECA procedure, there are still some errors that are not included in any fault bitmap but apears in error maps, i.e., these errors are caused by the faults that are not included in the target fault list or even have not been defined. These faults are called *unmodeled faults*. By investigating the error bitmaps, we can create a March signature for a specific type of error. A new fault model can be defined and added for a specific error behavior that appears frequently.

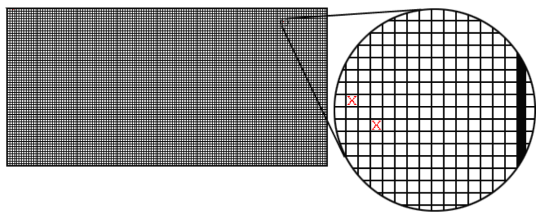

Figure 8: $CFin_2$ fault bitmap of an 16Kx8 embedded SRAM (FS80A020) test chip.

6. Conclusions

Bitmaps reported by commercial testers are not sufficient for memory designers and process designers, more diagnostic information should be available for them to improve the yield and reliability. This paper presents an error catch and analysis (ECA) system, which consists of a test algorithm generator (TAGS), a fault simulator (RAMSES), and an error analyzer (ERA), for generating more useful information such as fault bitmaps and fault statistics. The ECA system is implemented for the off-line analysis, and can easily be integrated into the existing testing flow. With the ECA system, useful bitmaps can be generated to help memory designers for identifying design flaws of their products. We are working on the integration of this system with design for testability (DFT) circuits such as built-in self-test (BIST) or built-in self-diagnosis (BISD) of semiconductor memories.

Acknowledgement

The authors would like to thank Kevin Chiu, Jia-Bin Huang, and Tony Teng of Faraday Technology Co. for providing the SRAM data and tester support.

References

[1] V. N. Yarmolik, Y. V. Klimets, A. J. van de Goor, and S. N. Demidenko, “RAM diagnostic tests”, in *Proc. IEEE Int. Workshop on Memory Technology, Design and Testing (MTDT)*, 1996, pp. 100–102.

[2] T. J. Bergfeld, D. Niggemeyer, and E. M. Rudnick, “Diagnostic testing of embedded memories using BIST”, in *Proc. Design, Automation and Test in Europe (DATE)*, Paris, Mar. 2000, pp. 305–309.

[3] L. Shen and B. F. Cockburn, “An optimal march test for locating faults in DRAMs”, in *Proc. IEEE Int. Workshop on Memory Testing*, 1993, pp. 61–66.

[4] 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.

[5] C.-F. Wu, C.-T. Huang, and C.-W. Wu, “RAMSES: a fast memory fault simulator”, in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Albuquerque, Nov. 1999, pp. 165–173.

[6] A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*, John Wiley & Sons, Chichester, England, 1991.

[7] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, New York, 1990.

[8] R. Dekker, F. Beenker, and L. Thijssen, “Fault modeling and test algorithm development for static random access memories”, in *Proc. Int. Test Conf. (ITC)*, 1988, pp. 343–352.