Fault coverage analysis of RAM test algorithms

Conference Paper · January 1995

DOI: 10.1109/VTEST.1995.512642 · Source: IEEE Xplore

2 authors:

Marc Riedel
University of Minnesota Twin Cities
105 PUBLICATIONS **1,974** CITATIONS
SEE PROFILE

Janusz Rajski
Mentor, A Siemens Business
338 PUBLICATIONS **9,184** CITATIONS
SEE PROFILE

Fault Coverage Analysis of RAM Test Algorithms

Marc Riedel†
MACS Laboratory, McGill University
Montreal, Quebec, Canada H3A 2A7
riedel@vlsi.ee.mcgill.ca

Janusz Rajski
Mentor Graphics Corporation
Wilsonville, OR 97070
rajski@wv.mentorg.com

Abstract

A methodology for evaluating the fault coverage of RAM test algorithms is proposed and the architecture of a flexible software analysis program is described. The analysis, performed for arbitrary test sequences, provides a comprehensive set of coverage statistics for functional cell-array faults. An overview of the analysis capabilities of the program is given, the fault state-transition conditions for several representative fault classes are specified, and coverage analysis results for a variety of test algorithms are presented.

1 Introduction

A considerable amount of research has been devoted to developing adequate fault models and efficient, yet complete, functional tests for semiconductor memories. Numerous test algorithms of varying complexity have been developed by both academics and memory test practitioners in recent years. An extensive survey of the work in this area is given by van de Goor in [2].

The aim of the present work is to describe computational techniques for evaluating the fault coverage of memory test algorithms. A flexible software program called RAMFLT has been developed to analyze the coverage of arbitrary test sequences. The program can accommodate the definition of a broad range of functional fault behaviors, developed under a variety of modeling assumptions. All of the cell-array fault models described in [2] have been implemented, including neighborhood pattern-sensitive faults and generalized k-coupling faults. Fault models are not hard-coded into the program; rather, they are specified as inputs in the form of a list of the required sensitization and desensitization conditions. Thus, new models can easily be added and experimented with.

Coverage statistics have been compiled for virtually all test algorithms proposed in the literature. The analysis results for 7 common tests are given in Table 3; statistics for other tests could not be included here due to space constraints, but are available from the authors.

Previous work in the area includes a structural fault simulation tool developed at Siemens in which defects are built into a cell-array model at representative locations [8]. Also, a methodology was recently proposed for automatically verifying so-called March tests [3] (tests which consist of sequences of operations applied linearly over the entire address range).

The present work addresses the problem of systematically characterizing the coverage of test algorithms through simulation and analysis. With the RAMFLT program, test algorithms can be evaluated and ranked based on a comprehensive and unified set of coverage statistics. Furthermore, the program is a valuable aid for the development and refinement of new testing schemes.

2 Methodology

A widely-used functional model for RAM devices was proposed by Nair, Thatte and Abraham [7]. The memory device is divided into three blocks: the cell array, the address decoder circuitry, and the read/write circuitry, including the sense amplifiers. A hierarchy of functional fault classes based on this RAM model has been proposed in the literature [2]. Only faults in the cell array are considered in the present work. Specifically, coverage statistics are presented for the following fault classes:

single-cell faults: stuck-at (SAF), transition (TF), and stuck-open (SOF).

coupling faults: idempotent (CFid), inversion (CFin), state (SCF), and dynamic (CFdyn).

bridging faults: AND-type/conjunctive (ABF) and OR-type/disjunctive (OBF).

neighborhood pattern-sensitive faults: active (ANPSF), passive (PNPSF), and static (SNPSF).

†This work was supported by Micronet, a Canadian federal network of centers of excellence in microelectronics.

The notational conventions and terminology adopted in [2] will be adhered to. For simplicity, the analysis performed by the RAMFLT program assumes the memory to be bit-addressable.

The physical failure mechanisms underlying faults can be complex [1]. However, from a functional standpoint, a fault manifests itself by causing a cell to present the wrong logical value, i.e., a value 1 when the last value written to the cell was a 0, or vice-versa. The affected cell is termed the *base* cell.

A fault is said to be *sensitized* when a faulty value appears in the base cell. This can occur if:

1. the base cell fails to make a transition when written to (*passive* sensitization),
2. the base cell is forced to a faulty value by a transition in some other cell (*active* sensitization).

In principle, a fault could also be sensitized by a neutral write operation (i.e., a write-0 to a cell with value 0 or a write-1 to a cell with value 1) or by a read operation.

A fault is *desensitized* when the base cell reverts to its correct value. This can occur if:

1. the faulty value in the base cell is over-written with the correct value (*passive* desensitization),
2. a write operation to some other cell resets the base cell to the correct value (*active* desensitization).

Note that if a fault is desensitized prior to being read, then the fault will not be detected by the read operation; this is referred to as *fault masking*.

The complexity of functional fault models ranges from simple stuck-at behavior to elaborate neighborhood pattern sensitivities. For the purposes of coverage analysis, a list of sensitization and desensitization conditions must be specified explicitly for each fault type.

For example, consider a 2-cell idempotent coupling fault, the CFid $(\uparrow; 1)$. With this fault, a base cell x is set to a value 1 if a $0 \rightarrow 1$ transition occurs in some other cell y. Hence, such a fault is sensitized by a write-1 operation to cell y when cells x and y both have value 0. It is desensitized by a write operation of either polarity to cell x. The sensitized state is denoted here as $\overline{s}$, and the unsensitized state as $\underline{s}$. Sensitization and desensitization are referred to as fault state transitions (FSTs).

A general fault state transition table, with a list of FST conditions for each fault type, takes the form of Table 1. An FST is triggered by an operation $o(y)$ consisting of either a read, a write-0 or a write-1 to a cell with address y (either the base cell or some other cell). Further, each transition is contingent upon the presence of specific values $v_1, v_2, \ldots, v_k$ in a number of cells, termed the neighborhood. The size and configuration of the neighborhood varies according to the fault class: for single-cell faults, the neighborhood consists of the base cell only; for neighborhood pattern-sensitive faults, it consists of the base cell and physically adjacent cells; and for generalized k-coupling faults, it consists of the base cell and some $k-1$ cells situated anywhere in the cell array.

It is assumed that an instance of each fault type may occur at every location in the memory, except possibly at the boundaries of the cell array. Thus, in Table 1, the locations of the addressed cell y and the neighborhood cells forming the pattern p are specified relative to the base cell.

Fault	Trans.	Operation	Pattern
f_1	$\underline{s} \rightarrow \overline{s}$	$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$\vdots$	$\vdots$
	$\overline{s} \rightarrow \underline{s}$	$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$\vdots$	$\vdots$
f_2	$\underline{s} \rightarrow \overline{s}$	$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$\vdots$	$\vdots$
	$\overline{s} \rightarrow \underline{s}$	$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
f_m	$\underline{s} \rightarrow \overline{s}$	$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$\vdots$	$\vdots$
	$\overline{s} \rightarrow \underline{s}$	$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$o(y)$	$p = (v_1 v_2 \cdots v_k)$
		$\vdots$	$\vdots$

Table 1: FST Table, Indexed by Fault Type.

A given operation may trigger state transitions in several base cells. In order to facilitate the task of determining which FSTs occur for a given entry in the test sequence, it is expedient to reorganize the information about state transitions in the format shown in Table 2. The indexing scheme of this table is as follows: first, according to the type of operation: read, write-0 or write-1; next, according to the offset y of the base cell from the addressed cell; last, according to the pattern p in the base cell's neighborhood.

Op.	Offset	Pattern	Fault State Transitions
r	y_1	p_1	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		p_2	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		$\vdots$	$\vdots$
	y_2	p_1	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		p_2	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		$\vdots$	$\vdots$
	$\vdots$	$\vdots$	$\vdots$
w_0	y_1	p_1	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		p_2	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		$\vdots$	$\vdots$
	y_2	p_1	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		p_2	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		$\vdots$	$\vdots$
	$\vdots$	$\vdots$	$\vdots$
w_1	y_1	p_1	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		p_2	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		$\vdots$	$\vdots$
	y_2	p_1	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		p_2	$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$
		$\vdots$	$\vdots$
	$\vdots$	$\vdots$	$\vdots$

Table 2: FST Table, Indexed by Operation.

The following data structures are required for coverage analysis:

1. A record of the correct (i.e., fault-free) values of the memory cells,

   $$m(x) \in \{0,1\} \text{ for } 0 \leq x \leq n-1,$$

   where n is the memory size.
2. A record of the fault state for each fault type at every cell,

   $$f_i(x) \in \{\underline{s}, \overline{s}\} \text{ for } 0 \leq x \leq n-1 \text{ and } 1 \leq i \leq m,$$

   where m is the number of fault types being considered.
3. A record of the fault coverage (a true/false value for each fault type at every cell),

   $$d_i(x) \in \{\mathrm{T}, \mathrm{F}\} \text{ for } 0 \leq x \leq n-1 \text{ and } 1 \leq i \leq m.$$

With the data structures thus configured, coverage analysis proceeds as follows. As the test sequence is scanned sequentially, one keeps track of which state transitions occur. With the FST conditions listed in the format of Table 2, this simply requires performing a table-lookup operation for each entry in the test sequence. According to the type of operation (a read, write-0 or write-1) one obtains a list of offsets indicating the positions of cells which may be affected by the operation. For each offset y_i, the location of the corresponding base cell b_i is

$$b_i = x + y_i.$$

A pattern p_i consisting of the values of the cells in the base cell's neighborhood is constructed. If the neighborhood consists of k cells with offsets $z_1, z_2, \ldots, z_k$ from the base cell b_i, then

$$p_i = \langle m(b_i + z_1), m(b_i + z_2), \ldots, m(b_i + z_k) \rangle .$$

Given the operation (r, w_0 or w_1), the offset y_i and the pattern p_i, one obtains from Table 2 a list of fault state transitions of the form,

$$\langle f_1 \vert \underline{s} \rightarrow \overline{s} \rangle, \langle f_2 \vert \overline{s} \rightarrow \underline{s} \rangle, \ldots$$

These faults at the base cell b_i are updated to the appropriate state.

Ascertaining which faults are covered on a read operation is straightforward: all faults at the addressed cell which are in a sensitized state are deemed to be covered.

The complexity of the analysis is dependent upon the number of faults which undergo state transitions as a result of each operation in the test sequence. With coupling faults and neighborhood pattern-sensitive faults, this number can be quite large. For instance, there are 2^{12} distinct types of active NPSFs with type II neighborhoods which may occur at any given base cell. Accordingly, up to 2^{12} ANPSFs may be sensitized or desensitized as a result of a single write operation. Some of the more elaborate algorithms have $O(n^2)$ operations, where n is the memory size. Fault coverage analysis of such algorithms is an inherently lengthy process.

3 Fault Models

What follows is a summary of the functional specifications for the fault models implemented in the RAMFLT program. For a detailed description and justification of these models, the reader is referred to [2].

In a **stuck-at** fault, the value of a cell is stuck at a certain logical value (either 0 or 1), irrespective of

what value has been written to the cell. In a **transition** fault, a cell fails to undergo either a $0 \rightarrow 1$ or a $1 \rightarrow 0$ transition. In a **stuck-open** fault, the contents of a cell cannot be accessed, most commonly as a result of an open word line. If the operation of the sense amplifier is non-transparent, a read to a stuck-open cell will return the result of the previous read operation to some other cell [2].

With k-coupling faults, the value of the base cell is influenced by either the values of some $k-1$ other cells in the memory, or by changes in the values of these cells:

state coupling fault: the presence of certain values $v_1, v_2, \ldots, v_{k-1}$ in some $k-1$ cells forces the base cell to a specific value v_k.

idempotent coupling fault: a $v_1 \rightarrow \overline{v}_1$ transition in some cell, while some other $k-2$ cells have certain values $v_2, v_3, \ldots, v_{k-1}$, forces the base cell to a specific value v_k.

inversion coupling fault: a $v_1 \rightarrow \overline{v}_1$ transition in some cell, while some other $k-2$ cells have certain values $v_2, v_3, \ldots, v_{k-1}$, inverts the value of the base cell.

With **dynamic coupling** faults, a faulty value in the base cell may be induced by either a read or a write to some other cell. Four types of 2-cell dynamic coupling faults are reported in [6]: $< r0|w0; 0 >$, $< r0|w0; 1 >$, $< r1|w1; 0 >$ and $< r1|w1; 1 >$.

Bridging faults are a subset of state coupling faults which arise from galvanic shorts between cells [2]. Two types of behaviors for such faults are considered:

AND-type bridging fault: the presence of a value 0 in any one of the cells forces a value 0 in the remaining cells.

OR-type bridging fault: the presence of a value 1 in any one of the cells forces a value 1 in the remaining cells.

Two-cell and three-cell versions of these coupling and bridging fault types have been implemented in the RAMFLT program.

Neighborhood pattern-sensitive faults (NPSFs) are k-coupling faults in which the $k-1$ cells are restricted to a fixed neighborhood in physical proximity of the base cell.

active NPSF: the base cell is forced to a certain value v as a result of a specific transition in the neighborhood pattern. (Note that this is equivalent to a restricted k-cell idempotent coupling fault).

passive NPSF: the base cell fails to undergo a $v \rightarrow \bar{v}$ transition if its neighborhood contains a specific pattern.

static NPSF: the presence of a specific neighborhood pattern forces the base cell to a certain value v. (Note that this is equivalent to a restricted k-cell state coupling fault.)

The number of possible fault instances grows exponentially with the size of the neighborhood; for all but small neighborhood sizes, the combinatorics become unmanageable. Four types of neighborhoods have been implemented in the RAMFLT program, the largest of which consists of 9 cells.

1. Type-I: the base cell and the four cells directly adjacent to it.
2. Type-II: the base cell and the eight cells directly and diagonally adjacent to it.
3. 5×1: a row of 5 cells centered on the base cell.
4. 1×5: a column of 5 cells centered on the base cell.

4 Fault State Transitions

A fault model is specified to the RAMFLT program by providing a list of the fault's state transition conditions, of the form described in Section 2. The FST conditions for two fault classes, the state coupling fault and the passive NPSF, are described here as representative examples. The required FST conditions for the other fault models follow similarly from the definitions presented in the previous section.

State Coupling Fault

Consider a 2-cell state coupling fault in which the presence of a value v_y in some cell y forces the base cell x to a value v_x. This fault is denoted as SCF($v_y \hookrightarrow v_x$). The fault is sensitized by:

1. An attempt to change the value of cell x to $\overline{v}_x$ while cell y has value v_y.
2. A $\overline{v}_y \rightarrow v_y$ transition in cell y while cell x has value $\overline{v}_x$.

The fault is desensitized by:

1. A write-v_x operation to cell x, since this brings the cell to the value it was forced to by the fault.
2. A write-$\overline{v}_x$ operation to cell x once the value of cell y has changed to $\overline{v}_y$; in this case, the write operation succeeds since cell x is no longer forced to value v_x.

The FST conditions for the state coupling fault are summarized in Figure 1 ($m(i)$ denotes the value of cell i and $w(v,i)$ denotes a write-v to cell i).

for each cell x, $0 \leq x \leq n-1$ **do**
  for each $v_x \in \{0,1\}$ **do**
    for each cell y, $0 \leq y \leq n-1$, $y \neq x$ **do**
      for each $v_y \in \{0,1\}$ **do**

SCF ($v_y \hookrightarrow v_x$)

FST	Condition
$\underline{s} \rightarrow \overline{s}$	$w(\overline{v}_x, x)$ while $m(x) = v_x$, $m(y) = v_y$
	$w(v_y, y)$ while $m(x) = \overline{v}_x$, $m(y) = \overline{v}_y$
$\overline{s} \rightarrow \underline{s}$	$w(v_x, x)$ while $m(x) = \overline{v}_x$
	$w(\overline{v}_x, x)$ while $m(x) = \overline{v}_x$, $m(y) = \overline{v}_y$

      end-each
    end-each
  end-each
end-each

Figure 1: FST conditions for the 2-cell SCF.

Passive NPSF

Consider a passive NPSF in which the base cell fails to undergo the transition $v \rightarrow \overline{v}$ if its neighborhood contains a specified pattern p. The fault is sensitized if a write-$\overline{v}$ operation to the base cell is attempted while the neighborhood contains the pattern p. If subsequently the pattern changes and then the write-$\overline{v}$ operation is re-attempted, it succeeds and the fault is desensitized. A write-v operation desensitizes the fault regardless of the pattern present. The FST conditions for the passive NPSF are summarized in Figure 2 (N stands for the neighborhood).

for each cell x, $0 \leq x \leq n-1$ **do**
  for each $v \in \{0,1\}$ **do**
    for each $p \in \{0,1\}^k$ **do**

PNPSF (p; $v \rightarrow \overline{v}$)

FST	Condition
$\underline{s} \rightarrow \overline{s}$	$w(\overline{v}, x)$ while $N = p$
$\overline{s} \rightarrow \underline{s}$	$w(\overline{v}, x)$ while $N \neq p$
	$w(v, x)$

      end-each
    end-each
end-each

Figure 2: FST conditions for the Passive NPSF.

5 Analysis of Test Algorithms

What follows is a summary of the coverage analysis results for 7 test algorithms of varying complexity. Trials were conducted for a memory size of 256 bits (16 rows $\times$ 16 columns). The coverage statistics in Table 3 are broken down into the following categories: single-cell faults , global 2-cell coupling and bridging faults, local 2-cell coupling and bridging faults, local 3-cell coupling and bridging faults, and neighborhood pattern-sensitive faults

The global statistics for the 2-cell coupling and bridging fault classes include fault instances in which the coupling cell is located at any distance from the base cell, whereas the local statistics include only fault instances in which the coupling cell is located within a 3×3 neighborhood centered on the base cell. Note that each cell in the pair or trio of cells affected by a bridging fault is considered a separate fault instance. The global statistics are further broken down into subclasses according to the polarity of the base and coupling cells. Global statistics for 3-cell faults were beyond the computational resources available – the number of such 3-cell faults for a 256-bit memory is on the order of 2^{27}. Statistics for the NPSF classes are presented for the four types of neighborhoods discussed in Section 4.

The **MATS++** test [2], an acronym for Modified Algorithmic Test Sequence, is a primitive test of length $6n$ (where n is the memory size) guaranteed to detect only stuck-at and transition faults.

$$\{\Updownarrow (w_0); \Uparrow (r, w_1); \Downarrow (r, w_0, r)\}$$

MATS++ Test.

The RAMFLT analysis results show that the MATS++ test covers three-quarters of the 2-cell inversion coupling faults, 37.5% of the 2-cell idempotent coupling faults, 37.5% of the local 3-cell inversion coupling faults and only 19% of the local 3-cell idempotent faults.

The **March X test** is a $6n$ sequence which targets inversion 2-cell coupling faults [2].

$$\{\Updownarrow (w_0); \Uparrow (r, w_1); \Downarrow (r, w_0); \Updownarrow (r)\}$$

March X Test.

The RAMFLT analysis results confirm that all 2-cell inversion coupling faults are indeed covered. In general, this test offers appreciably better coverage than MATS++, a test of equal length. The March X test covers 50% of the 2-cell idempotent coupling faults,

50% of the local 3-cell inversion coupling faults and 25% of the local 3-cell idempotent faults.

The **March C-** test is a $10n$ sequence which targets idempotent 2-cell coupling faults [2].

$$\{\Updownarrow (w_0); \Uparrow (r, w_1); \Uparrow (r, w_0); \Downarrow (r, w_1); \Downarrow (r, w_0) \Updownarrow (r)\}$$

March C- Test.

The RAMFLT analysis results show that the March C- test covers all of the 2-cell fault classes, as proved theoretically in [2]. With a length of only $10n$, this is an impressive feat. Note that this test covers all local 3-cell inversion coupling faults, state coupling faults and bridging faults as well. However, it only covers 50% of the local 3-cell idempotent coupling faults.

The **Rajsuman** $7n$ algorithm, described in [10], has the following innovative feature: instead of a single write sequence spanning the entire range of addresses, two separate write sequences are run simultaneously on the upper and lower halves of the address space. The RAMFLT analysis results show that the Rajsuman $7n$ test provides worse coverage of 2-cell idempotent coupling faults than the $6n$ March X test (25% versus 50%). Furthermore, it covers only three quarters of the 2-cell state coupling faults.

The **GALPAT** algorithm [2] (also known as Galloping 1/0) is a common $O(n^2)$ test. At the outset, the memory is filled with a background value of 0's. A value 1 is written to the first cell and then each cell in the memory is read. Then the background is restored by rewriting a value 0 to the first cell. This is repeated for every cell in the memory. Then the whole process is repeated with a background value of 1's.

The RAMFLT analysis results show that the GALPAT test covers all 2-cell faults except for 0.3% of the idempotent coupling faults. Among the local 3-cell faults, it covers all bridging faults, 94% of the inversion coupling faults, 80% of the state coupling faults and only 48% of the idempotent coupling faults. GALPAT's coverage of the neighborhood pattern-sensitive fault classes is far from complete: 12% for the active 5-cell NPSFs, 16% for the passive 5-cell NPSFs and 41% for the 5-cell static NPSFs. Coverage of the 9-cell NPSFs is negligible.

The **Papachristou** algorithm for 3-cell coupling faults, described in [9], is based on a so-called moving inversion test pattern (MOVI) scheme. The test consists of a linear $36n$ segment followed by a $24n \log n$ segment. As expected, the test covers all 2-cell and 3-cell faults. It averages 51% coverage for the 5-cell ANPSFs, 65% for the 5-cell PNPSFs and 85% for the 5-cell SNPSFs.

The **TLANPSF1G** algorithm (for "Test to Locate Active NPSFs with Type I Neighborhoods using the 2-Group Method" – abbreviated as TNPSF in Table 3) belongs to a family of elaborate and lengthy tests for NPSFs. The so-called 2-Group method, described in [11], is used to generate Eulerian paths through all possible patterns in the 5-cell, type I neighborhood of each base cell. The resulting test has length $163n$. The RAMFLT analysis results confirm that the algorithm does indeed cover all active, type I NPSFs. It also covers all static, type-I NPSFs, but it misses 0.4% of the passive, type-I NPSFs. The algorithm covers all local 2-cell coupling faults and nearly all of the local 3-cell coupling faults. However, it does not fare as well for the global 2-cell coupling faults, where it achieves 91% coverage.

The **TLAPNPSF2T** algorithm targets active and passive NPSFs with type-II neighborhoods. The so-called tiling method, described in [2], is used to generate Eulerian paths through all possible patterns in the 9-cell, type II neighborhood of each base cell. The resulting test has length $5122n$. With a memory size of 256 bits, the test sequence consists of $5122 \times 256 = 128,050$ operations. Coverage statistics for this algorithm are presented in graphical form in Figure 3. The percent coverage for six fault classes is plotted versus the test sequence entry as the simulation progresses. The test targets type II ANPSFs directly. Accordingly, coverage of this fault class proceeds linearly from 0 to 100%. All stuck-at and transition faults are covered within the first 15,000 operations. The 2-cell idempotent coupling faults, 3-cell idempotent coupling faults and type I ANPSFs are not covered completely until the 110,000th operation.

6 Conclusions

An efficient, yet general, methodology for analyzing the fault coverage of functional test algorithms has been proposed. A flexible software program called RAMFLT has been developed to analyze the coverage properties of arbitrary test sequences. Detailed coverage statistics have been compiled for most of the test algorithms found in the literature.

The RAMFLT program is a powerful aid for the development and refinement of new testing schemes. The statistics computed by the program obviate the need for formal, theoretical proofs of coverage properties of the type given in [2]. Of particular value are the statistics of which fault instances test algorithms fail to detect; given this information, it is straightforward, in some cases, to extend the algorithms to

provide complete coverage.

One of the strengths of the RAMFLT program is its flexibility to accommodate new fault models. To date, most of the fault models proposed in the literature have been of a general nature. However, work currently underway on inductive fault analysis and fault diagnosis [5] may lead to increasingly complex fault models, tailored to specific memory architectures and layouts. The RAMFLT program would be of particular value in developing highly-customized test algorithms for such specialized faults.

The RAMFLT program does not make any *a priori* assumptions about the test sequence. Thus, it is well-suited for the analysis of pseudo-random test sequences. Also, the program could be of value in assessing the effectiveness of various BIST algorithms, and it could serve to validate testing schemes developed in the nascent work on automatic test pattern generation for memories described in [4].

References

[1] E. A. Amerasekera and D. S. Campbell, *Failure Mechanisms in Semiconductor Devices,* John Wiley & Sons, Chichester, UK, 1987.

[2] A. J. van de Goor, *Testing Semiconductor Memories,* John Wiley & Sons, Chichester, UK, 1991.

[3] A. J. van de Goor, "Automatic Verification of March Tests," *Rec. IEEE Int. Workshop on Memory Testing,* pp. 131-137, 1993.

[4] A. J. van de Goor, "The Automatic Generation of March Tests," *Rec. IEEE Int. Workshop on Memory Technology, Design and Testing,* pp. 86-91, 1994.

[5] S. Naik, F. Agricola and W. Maly, "Failure Analysis of High-Density CMOS SRAMs," *IEEE Design & Test of Computers,* Vol. 10, No. 2, pp. 13-23, March 1993.

[6] B. Nadeau-Dostie, A. Silburt and V. K. Agarwal, "Serial Interfacing for Embedded-Memory Testing," *IEEE Design & Test of Computers,* Vol. 7, No. 2, pp. 52-63, April 1990.

[7] R. Nair, S. M. Thatte and J. A. Abraham, "Efficient Algorithms for Testing Semiconductor Random Access Memories," *IEEE Trans. Computers,* Vol. C-27, No. 6, pp. 572-576, 1978.

[8] H.-D. Oberle and P. Muhmenthaler, "Test Pattern Development and Evaluation for DRAMs," *Proc. IEEE Int. Test Conf.,* pp. 548-555, 1991.

[9] C. A. Papachristou and N. B. Sahgal, "An Improved Method for Detecting Functional Faults," *IEEE Trans. Computers,* Vol. C-34, No. 2, pp. 110-116, 1985.

[10] R. Rajsuman, "New Algorithm for Testing Random Access Memories," *Electronics Letters,* Vol. 27, No. 7, pp. 574-575, 1991.

[11] D. S. Suk and S. M. Reddy, "Test Procedures for a Class of Pattern-Sensitive Faults," *IEEE Trans. on Computers,* Vol. C-29, No. 6, pp. 419-429, 1980.

Fault Class		**Neighb.**	MATS++	March X	March C	Rajsum.	GALPAT	Papach.	TNPSF
Single Cell									
SAF		1 cell	100%	100%	100%	100%	100%	100%	100%
TF		1 cell	100%	100%	100%	100%	100%	100%	100%
SOF		1 cell	3.12%	3.12%	3.52%	7.03%	4.69%	30.1%	3.12%
Global 2-cell Coupling									
CFid	$<\uparrow;0>$	R×C	50.0%	50.0%	100%	50.0%	99.6%	100%	93.9%
	$<\downarrow;0>$	R×C	50.0%	50.0%	100%	0%	100%	100%	87.8%
	$<\uparrow;1>$	R×C	50.0%	50.0%	100%	0%	100%	100%	87.8%
	$<\downarrow;1>$	R×C	0.00%	50.0%	100%	50.0%	99.2%	100%	93.9%
	$\forall$ 4	R×C	37.5%	50.0%	100%	25.0%	99.7%	100%	90.9%
CFin	$<\uparrow;\updownarrow>$	R×C	100%	100%	100%	50.0%	100%	100%	93.9%
	$<\downarrow;\updownarrow>$	R×C	50.0%	100%	100%	50.0%	100%	100%	93.9%
	$\forall$ 2	R×C	75.0%	100%	100%	50.0%	100%	100%	93.9%
SCF	$<0\hookrightarrow0>$	R×C	50.0%	50.0%	100%	50.0%	100%	100%	93.9%
	$<1\hookrightarrow0>$	R×C	50.0%	100%	100%	100%	100%	100%	100%
	$<0\hookrightarrow1>$	R×C	100%	100%	100%	100%	100%	100%	100%
	$<1\hookrightarrow1>$	R×C	50.0%	50.0%	100%	50.0%	100%	100%	93.9%
	$\forall$ 4	R×C	62.5%	75.0%	100%	75.0%	100%	100%	97.0%
OBF		R×C	50.0%	50.0%	100%	50.0%	100%	100%	87.8%
ABF		R×C	50.0%	50.0%	100%	50.0%	100%	100%	87.8%
CFdyn	$<r0\|w0;0>$	R×C	50.0%	50.0%	100%	0%	100%	100%	87.8%
	$<r0\|w0;1>$	R×C	50.0%	50.0%	100%	75.1%	100%	100%	100%
	$<r1\|w1;0>$	R×C	50.0%	50.0%	100%	75.1%	100%	100%	100%
	$<r1\|w1;1>$	R×C	50.0%	50.0%	100%	0.00%	100%	100%	87.8%
	$(\forall$ 4)	R×C	50.0%	50.0%	100%	37.5%	100%	100%	93.9%
Local 2-cell Coupling									
CFid	$\forall$ 4	3×3	37.5%	50.0%	100%	25.0%	93.5%	100%	100%
CFin	$\forall$ 2	3×3	75.0%	100%	100%	50.0%	100%	100%	100%
SCF	$\forall$ 4	3×3	62.5%	75.0%	100%	75.0%	100%	100%	100%
OBF		3×3	50.0%	50.0%	100%	50.0%	100%	100%	100%
ABF		3×3	50.0%	50.0%	100%	50.0%	100%	100%	100%
CFdyn	$\forall$ 4	3×3	50.0%	50.0%	100%	37.5%	100%	100%	100%
Local 3-cell Coupling									
CFid	$\forall$ 8	3×3	18.8%	25.0%	50.0%	12.5%	48.2%	100%	96.8%
CFin	$\forall$ 4	3×3	37.5%	50.0%	100%	25.0%	93.6%	100%	96.8%
SCF	$\forall$ 8	3×3	75.0%	100%	100%	50.0%	79.9%	100%	98.6%
OBF		3×3	78.2%	78.2%	100%	78.2%	100%	100%	100%
ABF		3×3	50.0%	50.0%	100%	50.0%	100%	100%	100%
Neighborhood Pattern Sensitive									
ANPSF		Type I	4.69%	6.25%	12.5%	3.12%	11.7%	50.0%	100%
		Type II	0.29%	0.39%	0.78%	0.20%	0.81%	5.27%	6.28%
		5×1	4.69%	6.25%	12.5%	3.12%	11.7%	51.6%	25.4%
		1×5	4.69%	6.25%	12.5%	3.12%	13.3%	51.6%	50.0%
PNPSF		Type I	6.25%	6.25%	12.5%	6.25%	15.6%	68.8%	99.6%
		Type II	0.39%	0.39%	0.78%	0.39%	0.98%	5.08%	6.30%
		5×1	6.25%	6.25%	12.5%	6.25%	15.6%	62.5%	26.6%
		1×5	6.25%	6.25%	12.5%	6.25%	15.6%	62.5%	51.6%
SNPSF		Type I	12.5%	15.6%	31.2%	12.5%	40.6%	81.2%	100%
		Type II	1.56%	1.76%	3.52%	1.18%	4.10%	18.8%	32.0%
		5×1	12.5%	15.6%	31.2%	12.5%	40.6%	87.5%	75.0%
		1×5	12.5%	15.6%	31.2%	12.6%	40.6%	87.5%	100%

Table 3: RAMFLT Coverage Analysis Results.