LEVEL II

RADC-TR-79-269
Final Technical Report
November 1979

TEST PROCEDURES FOR SEMICONDUCTOR RANDOM ACCESS MEMORIES

University of Iowa

Sudhakar M. Reddy
D. S. Suk

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-79-269 has been reviewed and is approved for publication.

APPROVED: Allen P. Converse

ALLEN P. CONVERSE
Project Engineer

APPROVED: David C. Luke

DAVID C. LUKE, Lt Col, USAF
Chief, Reliability & Compatibility Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBRM), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-79-269	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER,	
4. TITLE (and Subtitle): TEST PROCEDURES FOR SEMICONDUCTOR RANDOM ACCESS MEMORIES	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report, July 78 — Apr 79
6. PERFORMING ORG. REPORT NUMBER: N/A	
7. AUTHOR(s): Sudhakar M. Reddy, D. S. Suk	8. CONTRACT OR GRANT NUMBER(s): F30602-78-C-0083
9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of Iowa, Division of Information Engineering, Iowa City IA 52242	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 23380191
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (RBRM), Griffiss AFB NY 13441	12. REPORT DATE: November 1979
	13. NUMBER OF PAGES: 105
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Allen P. Converse (RBRM)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

random access memory
test procedures
functional faults
stuck-at faults
critical timing
pattern sensitivity
cell coupling
test algorithms

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Currently available memory testing algorithms were reviewed and evaluated to assess their inadequacies in testing large scale integrated circuit random access memories (RAMs). Categories of functional faults were proposed to include several types of coupling faults. A neighborhood for pattern sensitive faults was defined. A fault model for stuck-at failures in dynamic RAMs was derived, as were requirements to detect abnormal timing parameters. Procedures to detect the following classes of faults were then developed: (1) Functional faults, (2) neighborhood pattern sensitive faults, (3) stuck-at faults in —

DD FORM 1 JAN 73 1473

UNCLASSIFIED (Cont'd)

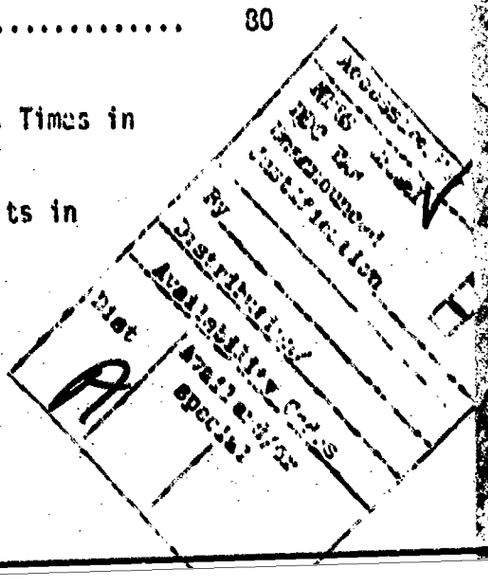
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

406737

Item 20 (Cont'd)

dynamic RAMs, and (4) abnormal access times.

TABLE OF CONTENTS



LIST OF TABLES

LIST OF FIGURES

EVALUATION

The objective of this program was to review currently available memory testing algorithms and to assess their inadequacies in testing large scale integrated circuit random access memories (RAMs). This was accomplished by first dividing memory faults into three typical classes: functional faults, pattern sensitive faults, and DC parametric faults. In general, the following results were determined.

A test algorithm to detect functional faults was developed using "march type" operations, and it was then shown to be a minimal test for a category of coupling faults. The use of Eulerian cycles in directed graphs produced a near minimal test algorithm for the detection and location of neighborhood pattern sensitive faults. Test procedures were developed to detect abnormal access times in RAMs and stuck-at-failures in dynamic RAMs.

The above procedures should reduce the complexity of testing requirements for high density memories. The results of this effort will be incorporated into detail specifications for semiconductor RAMs in support of MIL-M-38510, General Specification for Microcircuits.

ALLEN P. CONVERSE
Project Engineer

CHAPTER I

INTRODUCTION

Rapid developments in semiconductor technology have made larger and denser semiconductor memories on a single chip a reality [1, 4, 10, 20, 28, 31, 32]. Semiconductor memories can be divided into two types. One is a read-and-write type and the other is a read-only type. It is common to call a memory of read-and-write type as a RAM (random-access-memory) if the memory can be accessed at any cell location in the memory independent of the cell location of the previous operation. A read-and-write type of memory which can be accessed in a serial fashion only is not said to be a RAM and we will not consider this type of memory. A memory of read-only type is said to be a ROM (read-only-memory) and a memory of this type cannot be written with new data while the memory is under a normal operational mode. There are some ROMs which can be erased and rewritten with new data by way of some special processes which are depending on the technology employed to each specific ROM and the erasing and rewriting processes are normally done while the memory is not under the normal operation mode. Random access memories are customarily available in sizes of N words with m bits per word, where N is commonly a number which equals a positive integer power of two. Throughout this thesis, unless otherwise specified, we

assume that m=1. It is also a normal practice to give the number of bits in a RAM as Mk, where k=1024 bits. RAMs with 16k bits are in commercial use [32] and RAMs with 64k bits and 128k bits are being developed.

As more and more memory cells are packed into a single RAM chip, not only does the number of failures associated with RAMs increase but the nature of failure modes becomes much more complex. It turns out to be very difficult to detect or locate memory faults, mainly because of the large number of cells in a single RAM chip and also because of the varieties of the technology being used in manufacturing of those densely packed RAM chips. With these increasing difficulties in testing large sized RAMs, most of the RAM manufacturing industries are still relying upon a broken reef which was used to be an acceptable test for small sized RAMs and totally giving up comprehensive tests because such comprehensive tests may eventually lead those manufacturers to the situation that they cannot compete with others in the market due to the high cost of memory testing. This is why we need some new practical test algorithms for the densely packed RAMs with a reasonably good fault coverage.

But it is easy to see that it is almost impossible to perform a perfect test to prove that a given RAM operates correctly in all combinations of data patterns, accessing orders, DC parameters, timing parameters, etc. For example, to exercise a memory of 1k bits in all data combinations, at least $2x2^{1024}$ READ and WRITE operations are required. That means it will take more than 10^{293} years with a

100 nanosecond memory cycle time to test a memory of 1k bits only for all the possible data combinations. For this reason the normal strategy in testing RAMs with large number of memory cells, commonly 1k bits or more, is to identify their failure modes and use this information to design test procedures to detect the faults caused by these failure modes. There are quite a few test algorithms designed to detect one or more of known failure modes [5, 9, 11, 12, 19, 21, 24, 26, 29, 30]. Lists of such procedures can be found in references [5, 12, 16] together with the failure modes they cover. However, as we will show in the next chapter, all of them are inadequate to be used to test large scale integrated circuit RAMs for one reason or another. In designing more effective and practically usable test algorithms we will take two basic ideas into consideration, one is that of all known test algorithms we will choose some methods which are most effective in detecting faults at the same time being simple and proportional to the number of cells in the memory and to be easily implemented. The other idea is that we will try to get the maximum fault coverage from each chosen method while making use of the common design concepts of RAMs. Our test algorithms will be made applicable to all RAMS.

The organization of this report is as follows. In chapter II the general problem area of RAM testing is discussed and some deficiencies and impracticalities in known test algorithms are pointed out. Chapter III discusses functional fault model, march algorithm and fault coverage of the minimal march algorithm we found. In

Chapter IV practical neighborhood for pattern sensitive faults together with its fault model is given and minimal test algorithms for detection and location of the neighborhood pattern sensitive faults are found. Chapter V contains conclusions.

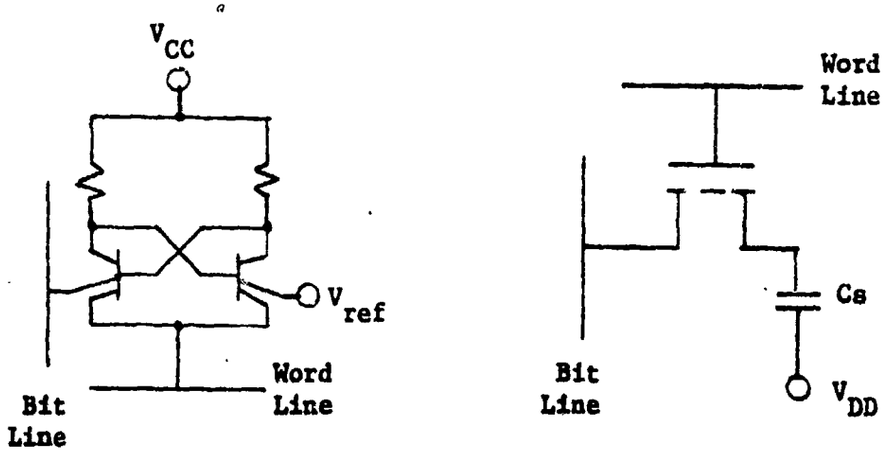
Appendix A includes a description of the development of a procedure to detect abnormal access times in semiconductor RAMs. Appendix B describes a fault model for stuck-at failures in dynamic RAMs, and it indicates a test procedure for their detection.

CHAPTER II

RAM TESTING AND INADEQUACIES IN KNOWN RESULTS

2.1 RAM Testing

Semiconductor RAMs are generally of two types [17], viz. (i) static and (ii) dynamic. Static RAMs consist of flip-flops and generally each flip-flop is one memory cell which is able to store a bit of information [31, 32]. Static RAMs employ various technologies such as bipolar transistor, MOSFET, SOS, Schottky diode, etc. Each technology has its own advantages and disadvantages, so users may choose proper ones which fit their requirements most. Dynamic RAMs keep the data in the form of charge on a capacitor [1, 4, 10, 28] and one typical value of such storage capacitance Cs with an inversion layer electrode under a silicon gate is 55fF (1 femtofarad (fF) = 10^{-15} farads) [28]. Most of the dynamic RAMs are using MOS technology with a few variations [1, 4, 10, 28]. A dynamic RAM requires periodic refreshing cycle, typically 2 milliseconds, since the stored data in the storage capacitors will be eventually leaked through several different paths depending on the actual technologies used for the design of the memory [9, 10, 11]. Several typical memory cell designs are shown in Figure 1 [10, 18, 20, 28, 31, 32].



(a) Bipolar Cell (b) Dynamic MOS Cell

Figure 1. Memory Cell Configurations

Most of commercially available RAMs have two dimensional memory cell array, row and column address decoder, write driver, sense amplifier, I/O port and control unit. A block diagram of semiconductor RAM is given in Figure 2 [30]. A memory cell array is divided into m rows and n columns. A particular cell in the memory cell array is accessed by addressing the row and the column corresponding to the cell and activating proper operation mode either READ or WRITE. Throughout this thesis we assume that the memory cells are numbered 0 through N-1 or equivalently the addresses of the cells are numbered 0 through N-1.

By testing of RAMs we mean the application of a selected test algorithm to the RAMs to detect or locate faults. The test algorithm normally comprises of a sequence of WRITE and READ operations on the RAM. In general each test algorithm is applied many times with

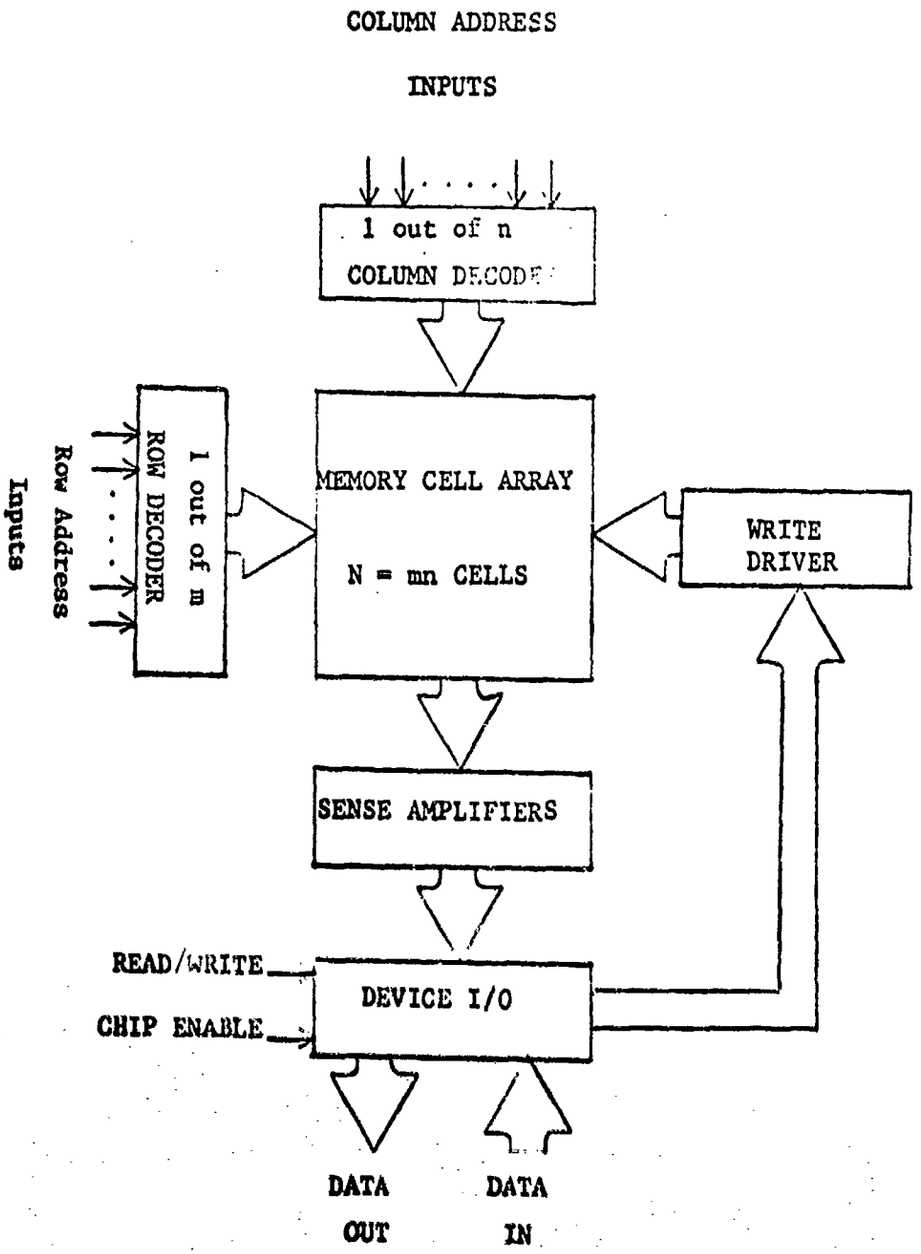


Figure 2. Block Diagram of a Semiconductor RAM

different supply voltages, and timing conditions under several different temperature settings. With large number of applications of test algorithm, the manufacturers determine the worst case conditions. But to keep the cost of testing within economic limits, production level testings are done against the worst case conditions within a few seconds, for example, 1.5 seconds for a 4k RAM [11]. For this reason it is quite obvious that test algorithms should be optimized with respect to the number of operations they require.

Test algorithms for RAMs can be conceptually divided into three parts [29, 30] (no standard terminology exists in this area and we are adopting the one in reference [30] as it appears to be most commonly used):

1. Functional testing: the test must detect physical failures which cause the RAM to function incorrectly; e.g., faults in memory cells, address logic, sense amplifiers, write drivers, noise coupling between cells, etc.

2. Pattern sensitivity testing: even though a RAM has no physical failure, there could be device anomalies and parasitic effects which could make its dynamic behavior sensitive to data and/or patterns. The test must detect these conditions; e.g., the effect of dynamic behavior of certain memory cells on the contents of another cell, sensitivity of various timing parameters [6, 7, 14], etc.

3. D.C. parameter testing: the D.C. parameters like power dissipation, fan out capabilities, noise margins and leakage currents must be checked.

For dynamic RAMs an extra test procedure to detect faults in refresh circuitry must be included.

Since D.C. parameter testing is usually not a major problem area in the testing of RAMs [11, 30], this thesis will only deal with the areas of functional testing and pattern sensitivity testing. In the following two sections, we will discuss the inadequacies of the currently known test algorithms for the functional testing and pattern sensitivity testing.

2.2 Functional Testing

By considering various failure modes in the components of a RAM (the memory cell array, the address decoder, the READ/WRITE logic) it was earlier shown that the following fault model is appropriate for deriving functional test algorithms for RAM [29, 30].

Fault Model A:

1. One or more cells are stuck-at-zero or stuck-at-one (these faults are called cell stuck-at faults). It should be emphasized that when a cell is stuck-at-x, then it will remain in x state, independent of reads and writes in any cell of the memory.
2. One or more cells fail to undergo a 0 to 1 and/or 1 to 0 transition, when the complement of the contents of the memory cell is written into the cell (these faults are called transition faults).

3. There exist two or more cells which are coupled. By this it is meant that a 0 to 1 or 1 to 0 transition in a cell changes the state of another cell in the memory, independent of the contents of other cells. This does not imply that if a transition in the state of cell i changes the state of cell j, a transition in the state of cell j changes the state of cell i (these faults are called coupling faults).

Definition 1: The faults given in the Fault Model A are called functional faults.

Earlier several algorithms were given to detect some or all of the functional faults [2, 5, 9, 11, 17, 19, 29, 30]. But it was shown in reference 21 that with the exception of test algorithms called GALPAT and GALWREC all the test procedures proposed to date are unable to detect all of the functional faults. But unfortunately, since GALPAT and GALWREC required $O(N^2)$ operations they are not applicable to large RAMs [9, 11, 16, 30]. For example GALPAT requires over eight minutes for one application to a 16K RAM with 500 n second READ/WRITE cycle time. It should also be mentioned that several test procedures that require $O(N^{3/2})$ operations (e.g., Galloping Diagonal [5, 11, 16], GALTCOL [2, 5, 11]) have been proposed and used. The fault coverage of these test procedures is not as comprehensive as GALPAT [2, 5, 11, 16]. Furthermore, these procedures may not be practical for memories with 16K or more cells. For example, with a

500 n second READ/WRITE cycle time GALTCOL would require over six seconds for one application. There are some test algorithms whose test length is O(N) [24, 26]. But none of them claim to have minimality against any class of functional faults and some necessary conditions in one algorithm [24] turn out to be not necessary as will be shown in the next chapter. Of those test algorithms two of them [24, 26] have reasonable test length, 31N and 30N, but by utilizing the fault detection capabilities of the march type algorithm we can reduce the test length further and end up with having minimal test algorithm among a class of algorithms for certain category of functional faults, as will be shown in the next chapter.

2.3 Pattern Sensitive Faults

Unrestricted pattern sensitive faults are much more difficult to detect than the functional faults [5, 11, 14]. The worst case test length of the unrestricted pattern sensitive faults was established by Hayes, where the pattern sensitive fault model is formalized by defining an incompletely specified sequential machine with 2^N states and 3N inputs [14]. The resulting test procedure requires $(3N^2 + 2N)2^N$ READ/WRITE operations. It is easy to see that this test procedure cannot be used in practice. For example, we need about 10^{314} READ/WRITE operations for a 1K RAM. For this reason it is necessary to find some other proper methods which can reduce the length of test procedures for pattern sensitive faults that are most likely to occur.

There are two different approaches known. One is finding test procedures which can detect pattern sensitive faults due to some identifiable anomalies such as dynamic timing parameters under their worst case condition. In fact several of such anomalous behaviors of most RAMs and their worst case conditions have been identified [9, 29, 30] and a few procedures to detect failures due to these problems have been given in the references [9, 29]. But as was pointed out in the reference [26] some of those procedures exhibit their own deficiencies. Moreover, the pattern sensitive faults can be caused not only by the identifiable anomalies but by any combinations of known and/or unknown anomalies. So the first approach can only cover the pattern sensitive faults which happen to fit the identifiable anomalies. This is why we need another approach which can detect pattern sensitive faults without considering actual individual causes of the faults, but with taking into consideration of the most likely faults in the memory.

The other approach is checking all possible dynamic patterns of a "neighborhood" for each memory cell in the RAM under test. We found that the following fault model is appropriate for this approach.

Fault Model B:

1. The content of a memory cell, say C_i, changes as a result of certain patterns of zeros, ones, zero to one transitions and/or one to zero transitions in the memory cells in the

"neighborhood" of C_i, where C_i is not supposed to change its contents.

2. A memory cell, say C_i, cannot perform a zero to one transition and/or a one to zero transition due to certain patterns of zeros and ones in the "neighborhood" of C_i.

Note: In Chapter IV we will define a neighborhood that appears to be appropriate for the semiconductor random access memories being produced.

Definition 2: The faults given in the fault model B are called pattern sensitive faults (PSFs).

But even with this definition of PSF it turns out to be impractical to test PSF if the "neighborhood" contains large number of memory cells. So it is necessary to find further reasonable restrictions on the size of the neighborhood. There are couple of known test procedures which belong to this approach [14, 27]. One is the test algorithm for single (local) pattern sensitive faults by Hayes which can detect certain pattern sensitive faults. This procedure requires 2720N READ/WRITE operations for the neighborhood to be defined in Chapter IV since the procedure requires $(3q^2 + 2q)2^q$ READ/WRITE operations for each neighborhood of size q. The other procedure is the API test by Srini [27] for a five-cell-neighborhood with 64N READ/WRITE operations. But it can be easily seen that this test doesn't cover all the PSFs of the fault model B for the five-cell-neighborhood.

2.4 Conclusion of Survey

So far every known fault testing algorithm for solid state RAMs has fallen into either of the following two cases or both: (1) when the test length is reasonably short, its fault coverage is not broad enough to be used alone. Furthermore, there are no known good methods of combining some of those test algorithms to achieve a desirable fault coverage; (2) when a test algorithm has comprehensive coverage of faults, the test length becomes impractically long to be used for a larger sized RAM. Also it can be noticed that most of the test algorithms were designed by using more intuition than systematic analysis.

Hence in this report we will do the following: (1) define a fault model for functional faults and choose a neighborhood for a neighborhood pattern sensitive fault model, (2) find a lower bound on the number of operations required for each fault model and (3) develop test algorithms which meet or closely approach the lower bounds and cover the faults in the fault models.

CHAPTER III

MINIMAL TEST ALGORITHM FOR FUNCTIONAL FAULTS

3.1 Fault Model

The coupling faults are the ones which are not completely covered by currently known test algorithms with the exception of the GALPAT and GALWREC. So we will refine the definition of coupling faults depending on the expected nature of the coupling faults then we will define several new categories of coupling faults by re-grouping these types of coupling faults.

Definition 3: A memory cell, say ith cell, is said to be (0→1)-asymmetrically coupled ((1→0)-asymmetrically coupled) to the jth cell, $i \neq j$, if and only if a 0 to 1 (1 to 0) transition in the contents of jth cell changes, independent of the contents of other cells, the contents of the ith cell from 0 to 1 and from 1 to 0.

Definition 4: A memory cell, say ith cell, is said to be (0→1;0)-asymmetrically coupled ((1→0;0)-asymmetrically coupled) to the jth cell, $i \neq j$, if and only if a 0 to 1 (1 to 0) transition in the contents of the jth cell changes, independent of the contents of other cells, the contents of the ith cell only when it is zero.

Definition 5: A memory cell, say ith cell, is said to be $(0\rightarrow 1;1)$-asymmetrically coupled ($(1\rightarrow 0;1)$-asymmetrically coupled) to the jth cell, $i \neq j$, if and only if a 0 to 1 (1 to 0) transition in the contents of the jth cell changes, independent of the contents of other cells, the contents of the ith cell only when it is one.

The cells, C_i and C_j in all three of above definitions will be called coupled cell and coupling cell, respectively.

Note that if a cell, say C_i, is $(x\rightarrow\bar{x};y)$-asymmetrically coupled to cell C_j then after an x to $\bar{x}$ transition in the contents of C_j is made, the contents of C_i will be $\bar{y}$ regardless of its original contents. Coupling faults can arise due to capacitive coupling between cells or due to leakage current from one cell to another [12, 16].

By using these refined definitions of coupling faults we can have fifteen different combinations of coupling faults between any two storage cells in the memory as listed below in Table 1. Next we define several categories of functional faults and then derive test algorithms for what will be called Category I functional faults.

Defintion 6: A RAM is said to contain one of the following three categories of functional faults depending on the nature of coupling faults in the RAM under test:

1. Category I coupling faults in a RAM are characterized as the faults from 1 through 8 of Table 1 (i.e., all coupling faults are asymmetric coupling faults).

Table 1

List of Functional Couplings

1. (0→1;0)-asymmetrically
2. (1→0;0)-asymmetrically
3. (0→1;1)-asymmetrically
4. (1→0;1)-asymmetrically
5. (0→1;0)-asymmetrically and (1→0;0)-asymmetrically
6. (0→1;0)-asymmetrically and (1→0;1)-asymmetrically
7. (0→1;1)-asymmetrically and (1→0;0)-asymmetrically
8. (0→1;1)-asymmetrically and (1→0;1)-asymmetrically
9. (0→1)-symmetrically
10. (1→0)-symmetrically
11. (0→1)-symmetrically and (1→0;0)-asymmetrically
12. (0→1)-symmetrically and (1→0;1)-asymmetrically
13. (1→0)-symmetrically and (0→1;0)-asymmetrically
14. (1→0)-symmetrically and (0→1;1)-asymmetrically
15. (0→1)-symmetrically and (1→0)-symmetrically.

2. Category II coupling faults in a RAM are characterized as the faults 1, 2, 3, 4, 9 and 10 of Table 1 (i.e., a memory cell C_i is coupled to a memory cell C_j in exactly one way).
3. Category III coupling faults include all faults in Table 1.

Note that the categories of functional faults defined above allow any number of couplings as long as the conditions on the individual coupling faults are satisfied.

To derive test algorithms following definitions are required.

Definition 7: Every memory cell in a RAM has three states.

1. Internal state is the actual content of a memory cell.
2. Apparent state is the result of read operation of a memory cell.
3. Expected state is the expected content of a cell after one or more write operations.

Definition 8: Faults are said to be detected if and only if there exist one or more differences between the expected states and the apparent states of the cells under test.

In the development of test algorithms we assume the following.

Assumption 1: Read operation is fault free.

Assumption 2: If a 0(1) is written into a memory cell when its internal state is 0(1) then the next internal state is 0(1) (and hence one need not test for correct operation of these writes). Furthermore, this write operation does not cause any coupling faults.

These assumptions are implicit in all the work on memory testing and appear to be valid for the current memory technologies. In testing memory for Category I coupling faults, the memory under test is assumed to have four or more memory cells.

There are no known bounds on the number of operations required for any of these categories of faults. In this chapter we will determine a lower bound for Category I coupling fault detection by assuming that only march type algorithm is used. There are some recently developed test procedures which can detect all of the Category I coupling faults with O(N) READ/WRITE operations [24, 26], but they are not minimal in test length. We will find a minimal test algorithm for the Category I coupling faults and investigate its coverage of fault other than the Category I coupling faults.

3.2 March Algorithm

All the currently known memory testing algorithms with O(N) READ/WRITE operations for functional fault detection are using march type operations. Major advantages of the march type operations among many techniques currently being used for the memory testing algorithms are based on their simplicity. Because of the simplicity, march type test algorithm is easy to analyze and handy to implement in actual memory testing devices. There are only two different orders to be considered in the march type test algorithms. The memory operations required to be applied to each memory cell during the application of one march element have to be same for every

cell in the memory. In this section we formally define march element and march algorithm and dig out several important properties of march algorithms.

Definition 9: A RAM under test is assumed to be able to perform both READ and WRITE operations. We use the following notations:

- R = READ operation on a cell,
- W = WRITE operation on a cell,
- W_0 = An operation of writing 0 into a cell,
- W_1 = An operation of writing 1 into a cell,
- W_c = An operation of writing the complement of the previous apparent or expected state of a cell,
- ↑ = An operation of writing 1 into a cell when the previous apparent or expected state of the cell was 0,
- ↓ = An operation of writing 0 into a cell when the previous apparent or expected state of the cell was 0.

Definition 10: A march element is a finite sequence of operations applied to every cell in the memory in either one of two orders, increasing address order from address zero or decreasing address order from address (N-1), where operations applied to each cell have to be the same for every cell. Let $\overline{O_1 \ldots O_n}$ denote the operations O_1 through O_n being applied to each cell from the lowest addressed cell to the highest addressed cell, and let $\underline{O_1 \ldots O_n}$

denote the operations O_1 through O_m being applied to each cell from the highest addressed cell to the lowest addressed cell. Also let $\widetilde{O_1 \ldots O_n}$ denote either $\overline{O_1 \ldots O_n}$ or $\underline{O_1 \ldots O_n}$.

Definition 11: A march algorithm is a finite sequence of march elements. It can be denoted as $(M_1, M_2, \ldots, M_{m-1}, M_m)$, where M_i is a march element, $1 \leq i \leq m$.

In a march algorithm, $(\overline{O}_1, \overline{O}_2)$ and $(\overline{O_1 O_2})$ are different from each other since $(\overline{O}_1, \overline{O}_2)$ implies that the operation O_1 is applied to each cell in the increasing address order then the operation O_2 is applied to each cell in the increasing address order while $(\overline{O_1 O_2})$ implies that the operation O_1 is applied to the lowest addressed cell first and the operation O_2 is applied to the same cell next and then the same operations in the same order are performed on the next higher addressed cell and so on until all the operations are finished at the highest addressed cell. In memory operations, we are assuming that reading a memory cell does not effect the contents of any memory cells. That is to say that we are assuming that internal state of cell is same as its apparent state.

In the remainder of this chapter we will derive a test procedure to detect functional faults that include Category I coupling faults. We have found it convenient to first derive a test algorithm to detect Category I coupling faults and then augment it to cover the stuck-at and transition faults. For this reason in this section and the next section we assume that only Category I

coupling faults are potentially present in the memory under test.

Definition 12: A march algorithm is said to be a complete march test for Category I coupling faults if and only if we can conclude that there do not exist any Category I coupling faults in the memory whenever all the read operations in the algorithm show no differences between expected states and corresponding apparent states.

Definition 13: A march algorithm is said to be an incomplete march test for Category I coupling faults if we cannot conclude nonexistance of any Category I coupling faults in the memory with no differences between expected states and corresponding apparent states from all the read operations in the algorithm.

Definition 14: A march algorithm is said to be an irredundant complete march test if and only if deletion of any one or more operations from the algorithm results in an incomplete march test.

Definition 15: A march algorithm is said to be a minimal complete march test if and only if there is no other complete march test which requires fewer number of operations.

Note that a minimal complete march test is an irredundant complete march test.

Before we start constructing minimal complete march tests, it is necessary to investigate some more properties of march elements which can be part of minimal complete march tests.

Lemma 1: Let A be an irredundant complete march test for Category I coupling faults. If a march element of this algorithm has read operations then the march element has one and only one read operation. Furthermore, the read operation has to be the first operation in the march element.

Proof: Let $\overbrace{0_1 \ldots 0_n}$ be a march element with read operations.

1. If n=1 then the lemma is clearly true.
2. If $n \geq 2$ then let 0_k be a read operation and $2 \leq k \leq n$.
   a. If 0_{k-1} is also a read operation then 0_k reads the same apparent state for each cell as 0_{k-1} has read. So 0_k is not necessary.
   b. If 0_{k-1} is a write operation then 0_k essentially reads what was written into a cell in the immediately preceding write operation. But such a read operation is unnecessary since in the absence of transition faults and cell stuck-at faults, the apparent, expected and internal state of the memory cell are all identical at that time.

Because of a and b no operation can precede a read operation in a march element of an irredundant complete march algorithm. Therefore, only place a read operation can appear in a march element is the first position.

Q.E.D.

Lemma 2: If a march element of an irredundant complete march test has more than one write operation then any two consecutive write

operations must write into each cell two different values, i.e., if the first write operation of the two is an operation of writing a zero then the second one should be an operation of writing a one, and vice versa.

Proof: In the absence of transition and stuck-at faults the first of the two write operations is guaranteed to put memory cell in the desired state and hence the second operation is redundant.

Q.E.D.

As a result of Lemma 1 and Lemma 2, a march element with one or more write operations can be denoted by specifying at least one write operation of the march element at the beginning or immediately after a read operation and the number of the write operations following the specified ones. For example, a march element $\overbrace{0_1 \ldots 0_k}(m)$ $(\overbrace{0_1 \ldots 0_k}(2m))$, $k \geq 1$ and $m > 0$, implies that there are m(2m) write operations of which each write operation complements the previous content of the memory cell immediately following the first k operations.

Theorem 1: Let MR be a memory with four or more cells. Assume that MR has no transition faults and stuck-at faults. Let the memory have either zeros in all locations or ones in all locations when it is powered up. To detect coupling fault of each row in Table 2 with an irredundant complete march test, the march element in the corresponding row of Table 2 has to be included in the irredundant complete march test.

Table 2

Typical Coupling Faults and Detecting

March Element in MR

Typical Coupling Fault(s) in MR	Detecting March Elements
1. C_3 is (0→1;0) and (1→0;1) coupled to C_0, C_3 is (0→1;1) coupled to C_1 and C_3 is (1→0;0) coupled to C_2	$\overline{R\uparrow\downarrow(m)}$ *
2. C_3 is (0→1;0) and (1→0;1) coupled to C_0, C_3 is (1→0;0) coupled to C_1 and C_3 is (0→1;1) coupled to C_2	$\overline{R\downarrow\uparrow(m)}$
3. C_2 is (0→1;0) and (1→0;1) coupled to C_0 and C_1	$\overline{R\uparrow(2m)}$ or $\overline{R\downarrow(2m)}$
4. C_0 is (0→1;0) and (1→0;1) coupled to C_3, C_0 is (0→1;1) coupled to C_2 and C_0 is (1→0;0) coupled to C_1	$\underline{R\uparrow\downarrow(m)}$
5. C_0 is (0→1;0) and (1→0;1) coupled to C_3, C_0 is (1→0;0) coupled to C_2 and C_0 is (0→1;1) coupled to C_1	$\underline{R\downarrow\uparrow(m)}$
6. C_1 is (0→1;0) and (1→0;1) coupled to C_3 and C_2	$\underline{R\uparrow(2m)}$ or $\underline{R\downarrow(2m)}$

* Note: m is a nonnegative integer.

Proof: We will prove this theorem by assuming absence of each required march element of Table 2 and showing nondetectability of the corresponding coupling fault. This will give us contradiction to the completeness of the irredundant complete march test.

1. If $\overline{R\uparrow\downarrow(m)}$ is not included in the test then the test cannot detect the coupling fault 1 of Table 2, since as shown below no march element can sensitize this fault.
   (a) $\underline{0_1 \ldots 0_n}$ cannot sensitize the fault since (1) it does not have any operations on C_0, C_1 and C_2 prior to the operations on C_3. (2) at the end of $\underline{0_1 \ldots 0_n}$ C_3 will have internal state which is same as its expected state because C_3 is $(0\rightarrow1;0)$ and $(1\rightarrow0;1)$ coupled to C_0.
   (b) $\overline{\uparrow(m)}$ or $\overline{\downarrow(m)}$ cannot sensitize the fault at the end of the march element since C_3 has no transition or stuck-at faults and hence C_3 will be written correctly and no further change can be made on C_3.
   (c) $\overline{R\downarrow(m)}$ cannot sensitize the fault before C_3 is read in the march element since C_3 is $(1\rightarrow0;0)$ coupled to C_2.
   (d) $\overline{R\uparrow}$ cannot sensitize the fault before C_3 is read in the march element since C_3 is $(0\rightarrow1;1)$ coupled to C_1 and the coupling between C_3 and C_2 cannot occur.

Now there are no more march elements to be considered other than $\overline{R\uparrow\downarrow(m)}$.

2. If $\overline{R\downarrow\uparrow(m)}$ is not included in the test then the test cannot detect the coupling fault 2 of Table 2, since as shown below no march

element can sensitize this fault.

(a) $\underline{0_1 \ldots 0_n}$ cannot sensitize the fault since (1) it does not have any operations on C_0, C_1 and C_2 prior to the operations on C_3, (2) at the end of $\underline{0_1 \ldots 0_n}$ C_3 will have internal state which is same as its expected state because C_3 is $(0\rightarrow1;0)$ and $(1\rightarrow0;1)$ coupled to C_0.

(b) $\overline{\uparrow(m)}$ or $\overline{\downarrow(m)}$ cannot sensitize the fault at the end of the march element since C_3 has no transition or stuck-at faults so C_3 will be written correctly and no further change can be made on C_3.

(c) $\overline{R\uparrow(m)}$ cannot sensitize the fault before C_3 is read in the march element since C_3 is $(0\rightarrow1;1)$ coupled to C_2.

(d) $\overline{R\downarrow}$ cannot sensitize the fault before C_3 is read in the march element since C_3 is $(1\rightarrow0;0)$ coupled to C_1 and the coupling between C_3 and C_2 cannot occur.

Now there are no more march elements to be considered other than $\overline{R\downarrow\uparrow(m)}$.

3. If none of $\overline{R\uparrow(2m)}$ and $\overline{R\downarrow(2m)}$ are included in the test then the test cannot detect the coupling fault 3 of Table 2, since as shown below no other march element can sensitize this fault.

   (a) $\underline{0_1 \ldots 0_n}$ cannot sensitize the fault since (1) C_j, $j\geq3$, is not a coupling cell for C_2, (2) at the end of $\underline{0_1 \ldots 0_n}$ C_2 will have internal state which is same as its expected state because C_2 is $(0\rightarrow1;0)$ and $(1\rightarrow0;1)$ coupled to C_0.

   (b) $\overline{\uparrow(m)}$ or $\overline{\downarrow(m)}$ cannot sensitize the fault at the end of the

march element since C_2 has no transition faults or stuck-at faults so C_2 will be written correctly and no further change can be made on C_2.

(c) $\overline{R\uparrow\downarrow(2m)}$ or $\overline{R\downarrow\uparrow(2m)}$ cannot sensitize the fault before C_2 is read in the march element since C_2 is (0→1;0) and (1→0;1) coupled to C_1.

Now there are no more march elements to be considered other than $\overline{R\uparrow(2m)}$ and $\overline{R\downarrow(2m)}$.

4. 5. 6. For the march elements in 4. 5. and 6. of Table 2, same proofs as 1. 2. and 3. can be made by simply changing the order of operations and the order of the addresses of those cells into opposite order.

Now if an irredundant complete march test does not include any one of the six different types of march element then the test is not complete any longer.

Q.E.D.

Theorem 2: A lower bound on the number of memory operations in complete march test for Category I coupling faults of a memory with N cells, N≥4, under the assumption of absence of transition and stuck-at faults is 14N.

Proof: Since at power up all the cells of a memory under test could contain one or zero, an irredundant complete march test must perform $\overline{R\uparrow\downarrow(m_1)}$, $\overline{R\downarrow\uparrow(m_2)}$, $R\uparrow\downarrow(m_4)$ and $R\downarrow\uparrow(m_5)$. With $m_1 = m_2 = 0$, the coupling fault 3 of Table 2 will not be sensitized. So one or more

operations have to be added to detect the coupling faults 3 of Table 2. This can be minimally done by letting either m_1 or m_2 to be 1. In a similar manner we can conclude that either m_4 and m_5 should be 1. This implies that fourteen memory operations have to be included in any irredundant complete march test algorithm.

Q.E.D.

3.3 Minimal Complete March Test

The lower bound of Theorem 2 was proved by assuming that the memory under test initially contained identical value in all cells at power on. However, the algorithm given below will be shown to detect all Category I coupling faults and still require only 14N operations, even when the memory under test is not initialized to an all zero or an all one state.

Definition 16: Algorithm A is a sequence of march elements, $\overline{RW_cW_cW_c}$, $\overline{RW_cW_c}$, $RW_cW_cW_c$, RW_cW_c, where each march element is denoted as M_1, M_2, M_3, M_4, respectively.

Theorem 3: Algorithm A is a minimal complete march test for Category I coupling faults in a memory with four or more cells under the assumption of absence of transition and stuck-at faults.

Proof: Let the apparent states of the cells, C_0, C_1,...,C_{N-2}, C_{N-1} in the memory under test when read at R of M_1 be X_0, X_1,...,X_{N-2}, X_{N-1}, respectively.

1. Let C_i be asymmetrically coupled to some cells with addresses lower than i, and let C_j be the highest addressed coupling cell of C_i, with $j < i$.

(a) If C_i is $(X_j \rightarrow \overline{X}_j; a)$ coupled to C_j, $a = X_i$ or $\overline{X}_i$, and C_i may or may not be $(\overline{X}_j \rightarrow X_j; b)$ coupled to C_j, $b = X_i$ or $\overline{X}_i$. Then since the last write operation on C_j changes the content of C_j from X_j to $\overline{X}_j$ in M_1 as well as in M_2, C_i will be read with $\overline{a}$ in both the march elements.

(b) If C_i is only $(\overline{X}_j \rightarrow X_j; b)$ coupled to C_j, $b = X_i$ or $\overline{X}_i$, then we will read C_i with $\overline{b}$ in M_1 as well as in M_2 since each of M_1 and M_2 has a write operation which changes the content of C_j from $\overline{X}_j$ to X_j.

But the expected state of C_i in M_1 is different from the expected state of C_i in M_2. Therefore, the fault will be detected by Algorithm A.

2. Let C_i be asymmetrically coupled to some cells with addresses higher than i, and let C_j be the lowest addressed coupling cell of C_i, $j > i$. We can have similar proof as in 1 by using M_3 and M_4 instead of M_1 and M_2.

Q.E.D.

3.4 Modification and Comment

We next consider stuck-at and transition faults.

Lemma 3: All stuck-at faults will be detected by Algorithm A under the assumption of the absence of transition faults.

Proof: For every cell in the memory under test, expected state of the cell in M_1 is different from the expected state of the cell in M_2.

But existance of stuck-at fault in a certain cell will result in two identical apparent states in M_1 and M_2. Therefore, the fault will be detected.

Q.E.D.

To detect transition faults in the memory, two additional read operations are required in Algorithm A. The resulting modified algorithm is $\overline{RW_cRW_cRW_c}$, $\overline{RW_cW_c}$, $\underline{RW_cW_cW_c}$, $\underline{RW_cW_c}$, which requires 16N memory operations. We will call this algorithm Algorithm AT.

Theorem 4: Algorithm AT detects all stuck-at faults, transition faults and Category I coupling faults even if a RAM has any combinations of such faults.

Proof: Whenever a RAM has stuck-at faults, Algorithm AT will detect them regardless of the presence of transition faults and Category I coupling faults since stuck-at faults will not be changed further by other faults and in the algorithm each cell is read with each of two different expected states. If the RAM contains transition faults and may or may not contain Category I coupling faults then the transition faults will be detected in M_1 of Algorithm AT since contents of each cell is read immediately after each of the first two transitions in M_1. Because of Theorem 3, all Category I coupling faults will be detected by Algorithm AT since the two extra read operations in M_1 of Algorithm AT do not reduce the fault detecting capability of Algorithm A.

Q.E.D.

If there are reasons to believe that an operation of writing zero (one) into a cell when its content prior to the operation is also zero (one) may not be done correctly then to detect such fault two more read operations and two extra write operations are required to be added to Algorithm AT. To present this modified algorithm in the same sequential form of march elements we have been using in this chapter we need a new notation for such write operation. Let W_s denote a write operation which writes a cell with the same content as it has prior to the operation. The algorithm which can detect faults due to faulty W_s in addition to stuck-at faults, transition faults and Category I coupling faults is $\overline{RW_cRW_sRW_cRW_sRW_c}$, $\overline{RW_cW_c}$, $\underline{RW_cW_cW_c}$, $\underline{RW_cW_c}$. This algorithm requires 20N memory operations.

In one recently published paper [24], necessary conditions for test algorithm which can detect stuck-at faults, transition faults and Category I coupling faults were claimed. It can be easily seen that our Algorithm AT clearly violates the condition 2 of the paper while detecting all stuck-at faults, transition faults and Category I coupling faults. In the following paragraph the condition 2 of the paper and an example which does not satisfy the condition 2 are shown.

"Condition 2 [24]. For every pair of cells (i,j), cell i (i.e., the cell with address i) must be read after cell j makes a forced transition (i.e., a transition which is initiated by the testing algorithm by writing into the cell) and before cells i and j make any further forced transitions for the following states of cell i and transitions in cell j:

(a) cell i in state 0, cell j making a 0 → 1 transition,

(b) cell i in state 1, cell j making a 0 → 1 transition,

(c) cell i in state 0, cell j making a 1 → 0 transition, and

(d) cell i in state 1, cell j making a 1 → 0 transition."

To make our discussion easy to understand, let the memory have zeros in all the cells when the memory is powered up. If $j<i$ then in Algorithm AT the pair of memory cells C_i and C_j do not go through case (b) of condition 2 and if $i<j$ then in Algorithm AT the pair of memory cells do not perform the case (d) of condition 2. Therefore Altorithm AT clearly violates the condition 2 of the paper.

In this chapter we have so far concentrated on Category I coupling faults. The algorithms given can be shown to detect many Category II coupling faults. Several different algorithms (some requiring fewer operations than Algorithms A and AT) can be derived to detect large subclasses of Categories II and III coupling faults. But at this time no new algorithm to detect all Categories II and III coupling faults has been found. The difficulty in deriving such algorithms lies in the difficulty of detecting all symmetric coupling faults.

CHAPTER IV

TEST ALGORITHMS FOR PATTERN SENSITIVE FAULTS

4.1 Neighborhood Model

As was pointed out in Chapter II the choice of neighborhood is the most critical factor in developing practical test algorithm for pattern sensitive faults. To determine a size of neighborhood which is small enough to be employed in developing reasonably short test algorithms as well as good enough to cover most probable pattern sensitive faults in the memory we will take advantage of the knowledge of device geometry and operational principle of most of currently available RAMs [7, 16].

Basically most of the densely packed RAMs have two-dimensional memory cell array [1, 4, 10, 18, 28, 31]. One typical memory cell array of single-transistor cells with sense amplifier is shown in Figure 3 [10, 28]. Even though there are many technical variations in operational principle of accessing memory cells to store data in the memory cells and to read data from the memory cells, one basic principle of accessing memory cells is common to all the densely packed RAMs available, which is that whenever a memory cell is to be accessed, the column and the row which the memory cell belongs to are chosen to be activated [1, 10, 18, 28]. The cell

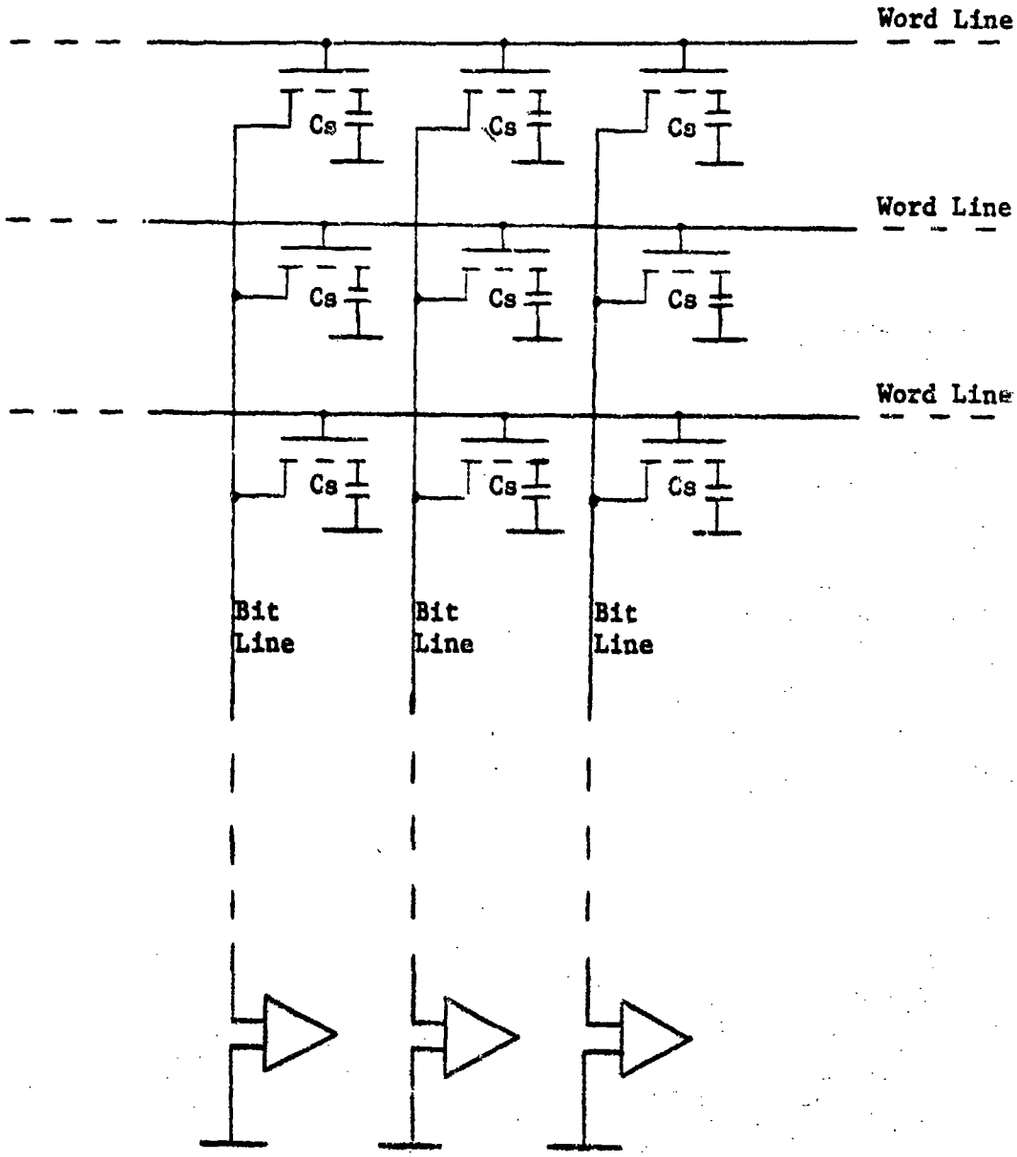


Figure 3. Memory Cell Array with Sense Amplifiers

in the crossing of the row and the column is the one fully activated by this selecting process. Since the purpose in this chapter is finding proper test algorithms for pattern sensitive faults we will not consider the operational principle of peripheral circuitry in a RAM chip, which is peculiar to each chip depending on the design technique employed in manufacturing process.

The most probable causes of pattern sensitive faults in RAMs are the charge leakage of memory cells and the effect of parasitic capacitance coupling on memory cells [1, 18, 22, 28], where the charge leakage could be caused by subthreshold current in the thick oxide separating one storage cell from another, by low field threshhold voltage, or by sneak paths created by an incorrect clock sequence [27].

It has earlier been suggested that a practical and effective choice of a neighborhood of a memory cell for pattern sensitive faults in a RAM is the four cells which are the ones in the top, bottom, left and right of the memory cell as elements of neighborhood [20, 22, 23, 27]. Following definition of neighborhood will be used throughout this chapter.

Definition 17: Let a memory have two-dimensional array organization with dimension n by m, where $n,m \geq 3$. Let a cell in the memory be fixed and call it a base cell. Then the set of five memory cells which include the base cell and its four adjacent neighboring cells which are located to the top, bottom, left and right of the base

cell is said to be the neighborhood of the base cell. A set of memory cells which excludes the base cell from its neighborhood is said to be the deleted neighborhood of the base cell.

For the completeness of this chapter we will define pattern sensitive fault with above definition of neighborhood, deleted neighborhood and base cell. In Definition 17 it must be noticed that the cells in the top row, bottom row, leftmost column and rightmost column do not have four adjacent neighboring cells and such neighborhood will have less than five cells but it will always have three or more cells as its elements.

Definition 18: Let a memory have two-dimensional array organization with dimension n by m, where $n,m \geq 3$.

1. If a memory cell, say C_i, changes its contents as a result of certain patterns of zeros, ones, changing (by writing) the contents from zero to one and/or from one to zero in the memory cells of the deleted neighborhood of C_i, then the memory is said to have active neighborhood pattern sensitive fault (ANPSF).
2. If the contents of a memory cell, say C_i, cannot be changed (by writing) from a zero to one and/or from a one to zero due to certain patterns of zeros and ones in the deleted neighborhood of C_i then the memory is said to have passive neighborhood pattern sensitive fault (PNPSF).
3. If any of the above faults exist in the memory then the memory is said to have neighborhood pattern sensitive faults (NPSFs).

In the development of test algorithms for NPSFs we assume that (1) read operation is fault-free, and (2) any propagational coupling faults which propagate from outside of a neighborhood to the cells in the neighborhood do not exist.

Let C_E be a base cell and C_A, C_B, C_C and C_D be its four deleted neighborhood cells located at the top, bottom, left and right of C_E, respectively. Each cell in the memory can have either static states or dynamic states during each memory cycle. It is convenient to define several symbols for those states.

Definition 19: There are four states for each memory cell during a memory cycle as following:

1. Static state zero, denoted by "0," is a state of a memory cell when the memory cell keeps the content zero from the beginning till the end of the memory cycle.
2. Static state one, denoted by "1," is a state of a memory cell when the memory cell keeps the content one from the beginning till the end of the memory cycle.
3. Dynamic state zero to one, denoted by "$\uparrow$," is a state of a memory cell when the memory cell changes its content from zero to one during a memory cycle by write operation.
4. Dynamic state one to zero, denoted by "$\downarrow$," is a state of a memory cell when the memory cell changes its content from one to zero during a memory cycle by write operation.

During each memory cycle a memory cell can have one of the four states defined above.

Definition 20: Pattern of a neighborhood of a base cell C_E is defined to be an ordered quintuple (u, v, w, x, y), where u, v, w, x and y are states of C_A, C_B, C_C, C_D and C_E, respectively.

To test a memory for all NPSFs, neighborhood of each base cell must be sensitized for all faults. By the term "sensitizing" we mean generating a pattern of states in the neighborhood of base cell which could exhibit faulty situation for a specific NPSF.

Definition 21: A pattern of a neighborhood of a base cell which can sensitize ANPSF (PNPSF) is called an active neighborhood pattern (ANP) (a passive neighborhood pattern (PNP)).

For example, a quintuple (0 ↑ 0 1 0) is an ANP since it sensitizes an ANPSF which may change the content of base cell from zero to one while C_A, C_C and C_D have contents 0, 0 and 1 and the content of C_B is changing from zero to one. A quintuple (1 0 1 1 ↓) is an example of PNP since it sensitizes a PNPSF due to the contents of 1, 0, 1 and 1 in C_A, C_B, C_C and C_D, respectively, which cause the content of the base cell C_E not to change from one to zero, when zero is written in the base cell with it initially containing 1.

It can be easily seen that there are 128 distinct ANPs and 32 PNPs for the neighborhood of a base cell as shown in Tables 3 and 4, respectively.

Table 3

ANPs of Neighborhood of Five Cells

A	↑	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓	↓	↓	↓
B	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
C	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
D	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

A	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
B	↑	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
C	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
D	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

A	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
C	↑	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
D	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

A	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
C	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
D	↑	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓	↓	↑	↑	↑	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Note: 1. A = top cell, B = Bottom cell, C = left cell, D = right cell, E = base cell.
2. ↑ = 0 to 1 transition, ↓ = 1 to 0 transition.
3. Each column with five entries denotes one ANP of a base cell.

Table 4

PNPs of Neighborhood

with Five Cells

Top	Bottom	Left	Right	Base	Top	Bottom	Left	Right	Base
0	0	0	0	↑	0	0	0	0	↓
0	0	0	1	↑	0	0	0	1	↓
0	0	1	0	↑	0	0	1	0	↓
0	0	1	1	↑	0	0	1	1	↓
0	1	0	0	↑	0	1	0	0	↓
0	1	0	1	↑	0	1	0	1	↓
0	1	1	0	↑	0	1	1	0	↓
0	1	1	1	↑	0	1	1	1	↓
1	0	0	0	↑	1	0	0	0	↓
1	0	0	1	↑	1	0	0	1	↓
1	0	1	0	↑	1	0	1	0	↓
1	0	1	1	↑	1	0	1	1	↓
1	1	0	0	↑	1	1	0	0	↓
1	1	0	1	↑	1	1	0	1	↓
1	1	1	0	↑	1	1	1	0	↓
1	1	1	1	↑	1	1	1	1	↓

Note: 1. ↑ = 0 to 1 transition, ↓ = 1 to 0 transition.
2. Each row with five entries denotes one PNP of a base cell.

4.2 Lower Bounds

For the manufacturer of RAMs it may be important to find a location of the fault since it may exhibit some defects in their original layout of the RAM or in their manufacturing process [7, 9, 11]. So locating pattern sensitive faults in RAMs will be useful to manufacturers. Before developing test algorithms which can detect and locate NPSFs, we will find lower bounds on the number of required READ and WRITE operations in such test algorithms. By detection and location of an NPSF we mean that the test algorithm identifies the type and the location of the NPSF; that is, it identifies the base cell effected and the nature of the affliction. The fact that the cells at the corners and on the edges of the rectangular array memory have fewer than 4 cells in their deleted neighborhoods (e.g., the cell in the top left corner has only two cells in its deleted neighborhood) makes the exact derivation of these bounds unnecessarily cumbersome. For this reason we will assume that all cells in the memory have 4 cells in their neighborhood by assuming that the cells on the top edge of the memory are adjacent to the cells on the bottom edge and the cells on the left edge of the memory are adjacent to the cells on the right edge.

A convenient technique to derive these lower bounds is to look at the contents of a base cell to be the state of a two state sequential machine, with the application of an ANP to its neighborhood

corresponding to an input to the sequential machine while the current state is at the value specified in the ANP and reading of the contents of the base cell corresponding to observing the state of the two state machine. The following two lemmas will be used in the proofs of Theorems 5 and 6.

Lemma 3: 2^{2r} different sequential machines are realizable with two states and r distinct inputs.

Proof: For each input the next state can either be the same state as the present state or be the different state from the present state. So 2^{2r} different flow tables can be constructed and each flow table represents one unique sequential machine.

Q.E.D.

Lemma 4: Let a sequential machine have two states and r distinct inputs. Assume that the state of the sequential machine can be observed at the end of the application of each input to the machine. To identify this machine from 2^{2r} different sequential machines 2r observations are minimally required.

Proof: If we make only t observations, $t<2r$, then we can distinguish at most one flow table out of 2^t different flow tables. So $\log_2(2^{2r})$ observations are minimally required to distinguish one out of 2^{2r} machines.

Q.E.D.

Definition 22: If we neglect the entries in position E of Table 3 then there are 64 distinct columns with four entries. We call each of these columns a modified ANP (MANP). Also if we neglect the entries in the base position of Table 4 then there are 16 distinct rows with four entries. We call each of these rows a modified PNP (MPNP).

Lemma 5: To detect and locate ANPSFs in the neighborhood of a base cell, 128 read operations are minimally required.

Proof: Consider the 64 MANPs as inputs to a sequential machine with the content of the base cell corresponding to the state of the sequential machine. Then in the presence of ANPSFs the contents of a base cell behave like faulty sequential machine. Hence from Lemmas 3 and 4 we need to apply 128 inputs and make 128 observations of the contents (i.e., 128 read operations) of the base cell.

Q.E.D.

Lemma 6: To detect and locate NPSFs in a neighborhood of a base cell, 160 read operations of the base cell are minimally required.

Proof: The proof is similar to that of Theorem 5, except that now we have to consider 128 ANPs and 32 PNPs.

Q.E.D.

Lemma 7: To generate all ANPs, 32 write operations per cell are minimally required.

Proof: Since the application of an ANP requires one write operation, 128 write operations on the cells in the deleted neighborhood of a base cell would require 32 write operations per cell in the neighborhood. As a matter of fact again from the sequential machine analogy 32 write operations are minimally required for each cell in the deleted neighborhood. Since every cell in a memory is in the deleted neighborhood of some other to sensitize all ANPs for the memory, if we assume that every cell in the memory has a deleted neighborhood of four cells.

Q.E.D.

Summarizing Lemmas 5, 6, and 7, we see that 128 read and 32 write operations on each cell of the memory are required to detect and locate ANPSFs and extra 32 read operations on each cell are required to detect and locate all NPSFs. This is formally stated in the next theorem.

Theorem 5: To detect and locate ANPSFs in a RAM at least 128 read and 32 write operations must be performed on each cell in the RAM and to detect and locate NPSFs at least 160 read and 32 write operations must be performed on each cell of the RAM.

In testing RAMs, the ability to just detect the faults is also important since most of the users do not have facilities to fix the located faults in the RAMs.

To do this it will be of advantage to know good lower bounds on the number of operations required to detect ANPSFs and PNPSFs. Next theprem gives a lower bound on the number of operations required to detect ANPSFs.

Theorem 6: To detect all ANPSFs 32 write and 65 read operations per cell are necessary.

Proof: According to Lemma 7, 32 write operations per cell are required to generate all ANPSFs. To prove the necessity of 65 read operations per cell, let the 64 distinct MANPs be ordered from 1 to 64. To prove the validity of our assertion on the number of read operations, we will consider ANPSFs of following type only: if an MANP of an ANP changes the content of the base cell from zero to one then the MANP also changes the content of the base cell from one to zero. Since there are 64 distinct MANPs and each of them could be either faulty or fault free, there are $2^{64}-1$ faulty patterns of 64 MANPs and one fault free pattern of 64 MANPs, that affect the base cell in the manner described above. Let each of these 2^{64} patterns of 64 MANPs be represented by a 64 bit binary vector, F_i, such that $F_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,64})$, where $f_{i,j} = 0$ if the j^{th} MANP is fault free in F_i and $f_{i,j} = 1$ if the j^{th} MANP is faulty in F_i. Given a test algorithm with k read operations it is possible to divide the sequence into k sections by disconnecting the sequence after each read operation. Since at each read operation in the sequence we are comparing the expected state and

the apparent state of the base cell, if the apparent state of the base cell is different from the expected state of the base cell then faults are detected, otherwise faults are not detected. By assuming the existence of the type of ANPSFs previously mentioned only, if even number of MANPs, that were causing the base cell contents to change, appear between two consecutive read operations of the sequence then the faulty MANPs will not be detected since the read operation which follows the faulty MANPs will result in an apparent state which is the same as expected state provided that previous read operation did not detect any faults. Therefore given a test algorithm and a fault pattern of MANPs, between at least one pair of consecutive read operations of the test algorithm an odd number of MANPs that are noted faulty in the given fault pattern must be present. Let us convert each section of write operations which are sandwiched between two consecutive read operations into a 64 bit binary vector such that $V_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,64})$, $1 \leq i \leq k$, where $v_{i,j} = 0$ if the j^{th} MANP occurs even number of times in the i^{th} section and $v_{i,j} = 1$ if the j^{th} MANP occurs odd number of times in the i^{th} section. The faults are detected by the test sequence if and only if for every given F_i, there exists a V_j such that bit by bit logical AND operations between F_i and V_j result in odd number of ones, which is equivalent to say that the scalar product of F_i and V_j with mod2 addition assumed is one. This implies that given $V_1, V_2, \ldots, V_k$, there does not exist a nonzero vector F_i that is orthogonal to each one

of V_1, V_2, ..., V_k. But there will not exist a nonzero vector F_i that is orthogonal to each one of V_1, V_2, ..., V_k if and only if the matrix

$$\begin{bmatrix} V_1 \\ V_2 \\ \cdot \\ \cdot \\ \cdot \\ V_{k-1} \\ V_k \end{bmatrix}$$

has rank 64. Therefore k has to be equal to or greater than 64, and hence 65 read operations are now required to detect ANPSFs.

Q.E.D.

There are $4^{64}-1$ distinct patterns of faulty MANPs. We will call a faulty MANP a unidirectional faulty MANP if the MANP causes a fault in the base cell in one way only, i.e., a fault occurs because of the MANP only when the content of the base cell is zero (one). This restriction rules out only $2^{64}-1$ of $4^{64}-1$ faults associated with a base cell. The next theorem gives a bound on the number of operations to detect ANPSFs which include at least one unidirectional faulty MANP.

Theorem 7: To detect all ANPSFs that include at least one unidirectional faulty MANP for some base cell, 32 write and 2 read operations are required per each cell.

Proof: 32 write operations are necessary to generate all ANPs

as was shown in Lemma 7. If a base cell is read only once and the apparent state of the base cell is the same as the expected state, say zero (one), then a possible fault of zero to one (one to zero) transition in the base cell when a unidirectional faulty MANF is applied to its deleted neighborhood will not be detected. So two read operations per each cell with the expected states being zero and one are required.

Q.E.D.

The lower bound on the number of operations required to detect PNPSFs is the same as the lower bound on the number of operations required to detect and locate PNPSFs. Because each PNP requires one write operation on the base cell and there are 32 PNPs per each base cell where every cell in the memory has to be a base cell for all 32 PNPs, we need 32 write operations per each cell. Furthermore, if we do not read the base cell after each write operation then the following write operation on the base cell will complement the content of the base cell and then the possible PNPSF generated previously will be erased. Therefore, 32 read operations per each cell are also required. These discussions are formally stated in the next theorem.

Theorem 8: To detect all PNPSFs, 32 write and 32 read operations per cell are required.

Because of Theorems 7 and 8, we know that the number of operations for detection of ANPSFs could be reduced quite a bit compared with the required number of operations for detection and location of ANPSFs. But there are no advantages in finding test algorithms which can only detect all PNPSFs.

In this section we have found several lower bounds which could be used as goals to reach in our development of test algorithms. These lower bounds are summarized as follows: the lower bounds on the numbers of operations required to detect and locate ANPSFs and NPSFs are 160N and 192N, respectively, and the lower bounds on the number of operations required to detect ANPSFs and PNPSFs are 97N and 64N, respectively.

To obtain a minimal or a near minimal test algorithm, which satisfies the lower bounds found in this section, first difficulty which has to be overcome is a problem of overlapping ighborhood. In the following section we will find a method which results in nonoverlapping neighborhoods. The following two sections are the basis of our near minimal algorithms to be found later.

4.3 Cell Assignment

Since most of the densely packed RAMs have m by n memory cell array and m and n are usually integer powers of two, we will assume that $m=2^p$ and $n=2^q$, where $p \geq 3$ and $q \geq 3$. Note that this assumption does not necessarily mean that the algorithms to be developed

are applicable to memories with more than 64 cells only. For smaller memories the algorithms can still be applied.

Each memory cell will be addressed (i, j) when the cell belongs to ith column and jth row. We call i the column address and j the row address. In doing the above assignment of two dimensional addressing we assume that the memory under test has lowest addressed cell at the upper left corner of the memory cell array and that the row and column addresses of the cells range from 0 through 2^p-1 and 0 through 2^q-1, respectively.

Definition 23: Let S_{even} (S_{odd}) be the set of memory cells whose sum of row and column addresses is even (odd).

Definition 24: Let A, B, C and D be four symbols to be assigned to each cell in the memory. Assignment of these symbols is done according to the following rule: Let (i, j) be the (column address, row address) of a cell and $(i', j') = (i_{\text{mod } 8}, j_{\text{mod } 8})$.

1. For the cells in S_{even}
   a. A is assigned to a cell whose $i' = (3j')_{\text{mod } 8}$
   b. B is assigned to a cell whose $i' = (2+3j')_{\text{mod } 8}$
   c. C is assigned to a cell whose $i' = (4+3j')_{\text{mod } 8}$
   d. D is assigned to a cell whose $i' = (6+3j')_{\text{mod } 8}$
2. For the cells in S_{odd}
   a. A is assigned to a cell whose $i' = (1+3j')_{\text{mod } 8}$
   b. B is assigned to a cell whose $i' = (3+3j')_{\text{mod } 8}$

c. C is assigned to a cell whose $i' = (5+3j')_{\text{mod } 8}$

d. D is assigned to a cell whose $i' = (7+3j')_{\text{mod } 8}$.

Resulting memory arrays of size 8x8, after the assignment of the cells in S_{even} and S_{odd} are shown in Figure 4. Following observations are made to use this special cell assignment in developing our test algorithms for NPSFs.

Observation 1: Every cell in S_{even} (S_{odd}) has a deleted neighborhood consisting of four cells in S_{odd} (S_{even}). Furthermore, this deleted neighborhood contains every symbol exactly once. All possible patterns of symbol appearances with respect to a base cell are given in Figure 5.

Observation 2: Any pair of deleted neighborhoods of two cells with the same symbol in S_{even} (S_{odd}) are disjoint to each other.

Observation 3: Union of deleted neighborhoods of cells with the same symbol in S_{even} (S_{odd}) is S_{odd} (S_{even}).

As we can see from the Tables 3 and 4, each deleted neighborhood of a base cell has to be sensitized with 64 distinct patterns for ANPs with the state of the base cell zero and with the state of the base cell one, also it has to be sensitized with 16 distinct patterns for PNPs with the state of the base cell $\uparrow$ and $\downarrow$. Because of our cell assignment, it is possible to sensitize a neighborhood with a test sequence for all 64 ANPs with the state of the base cell

A		B		C		D	
	D		A		B		C
B		C		D		A	
	A		B		C		D
C		D		A		B	
	B		C		D		A
D		A		B		C	
	C		D		A		B

(a) S_{even} Assignment

	A		B		C		D
C		D		A		B	
	B		C		D		A
D		A		B		C	
	C		D		A		B
A		B		C		D	
	D		A		B		C
B		C		D		A	

(b) S_{odd} Assignment

Figure 4. Assignment of 8 x 8 Memory Cell Array

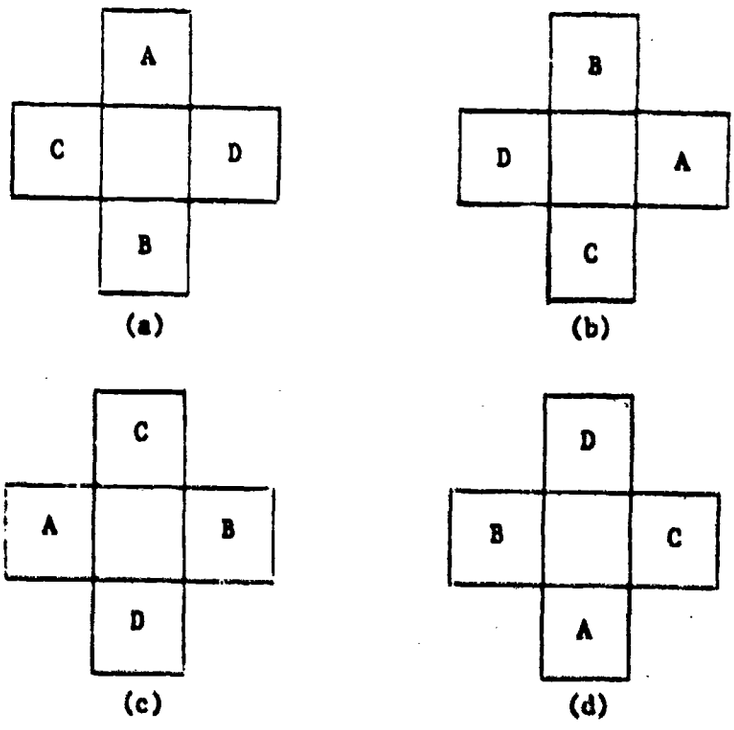


Figure 5. Assigned Neighborhood Patterns

zero and with the state of the base cell one. The cells of the whole memory can be sensitized by applying the same test sequence to all the cells with the same assigned symbol since if we generate all 64 distinct patterns of ANPs in a deleted neighborhood of Figure 5(a) then the other three deleted neighborhoods of Figure 5 will also be exercised with all 64 distinct patterns of ANPs by performing the same memory operations on the cell with the same symbol as was done on the cell in the deleted neighborhood of Figure 5(a). Same thing is true for the PNP case. Above statement can be justified by our previous observations.

4.4 Digraph and Minimal Sequence

To find a sequence of memory operations which can generate all ANPs and PNPs within minimal number of memory operations we will use several digraphs to be constructed, which have special properties. There are no standard terminologies in graph theory so we will adopt the terminology used in the "Structural Models: An Introduction to the Theory of Directed Graphs" by F. Harary [13]. We refer to nodes by the notation $v_1, v_2, \ldots, v_p$. We write v_1v_2 for an arc from v_1 to v_2.

Definition 25: A symmetric digraph is an irreflexive symmetric relation. Thus, for every arc v_iv_j in a symmetric digraph, there is also an arc v_jv_i.

Definition 26: The outdegree of node v_i, written $od(v_i)$, is the number of arcs from v_i. The indegree of node v_i, written $id(v_i)$ is the number of arcs to v_i.

Definition 27: A digraph is called an isograph if for every node v_i, $id(v_i) = od(v_i)$.

Clearly, every symmetric digraph is an isograph.

Definition 28: A node-arc sequence is an alternating sequence of nodes and arcs which begins and ends with a node and has the property that each arc is preceded by its first node and followed by its second node. A node-arc sequence is written in the form: $v_1v_2\ldots v_n$. The node v_1 is the initial node of this sequence and v_n is the terminal node.

Definition 29: If the initial node and the terminal node of a node-arc sequence are the same node, the sequence is said to be a cycle.

Definition 30: A collection of cycles is said to be arc-disjoint if no two of the cycles have an arc in common.

Two arc-disjoint cycles may have nodes in common.

Definition 31: A digraph is said to be Eulerian if it is possible to start at any node v_i, travel along each arc exactly once, and return to v_i and each such cycle is called an Eulerian cycle.

We shall find, following Euler that every isograph is Eulerian [13].

Definition 32: A digraph is said to be decomposed into m subgraphs when the set X of arcs is partitioned into m mutually disjoint subsets $Y_1, Y_2, \ldots, Y_m$ and each Y_i together with the set of nodes forms a new digraph, $1 \leq i \leq m$.

Let us consider ABCD as a four digit binary number, where A, B, C and D correspond to the contents of the four cells assigned with symbols A, B, C and D in the previous section. The number ABCD can have sixteen different binary numbers. By considering these sixteen binary numbers as sixteen different nodes and by connecting any pair of nodes at Hamming distance 1 from each other with two arcs in both directions, we have a digraph as shown in Figure 6. For example, two nodes, (0110) and (1100) are at Hamming distance two away from each other, hence no arcs will be drawn between these two nodes but two nodes, (1010) and (1110) are at Hamming distance one away from each other, hence two arcs in both directions, one from the node (1010) to the node (1110) and the other from the node (1110) to the node (1010) can be drawn. Notice that there are 64 distinct arcs in the graph. We will denote each node with (abcd) and each arc with one of (abcx), (abxd), (axcd) and (xbcd) where a, b, c, d $\in \{0,1\}$ and x $\in \{\uparrow, \downarrow\}$.

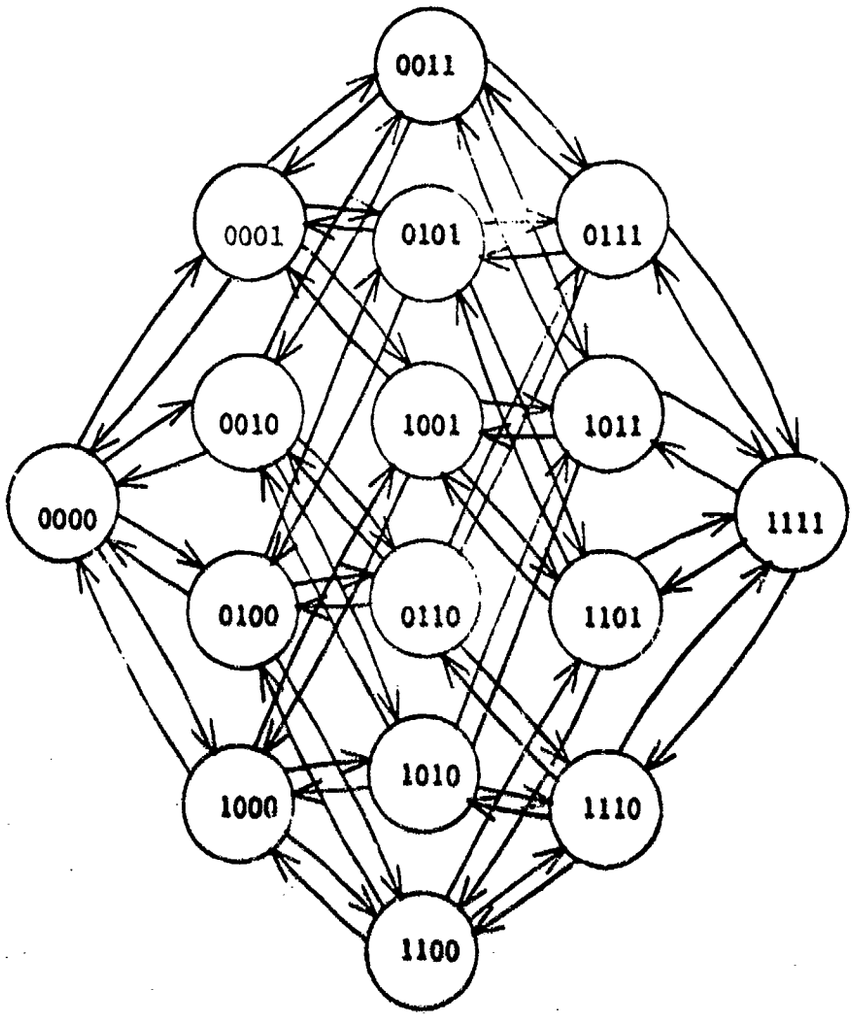
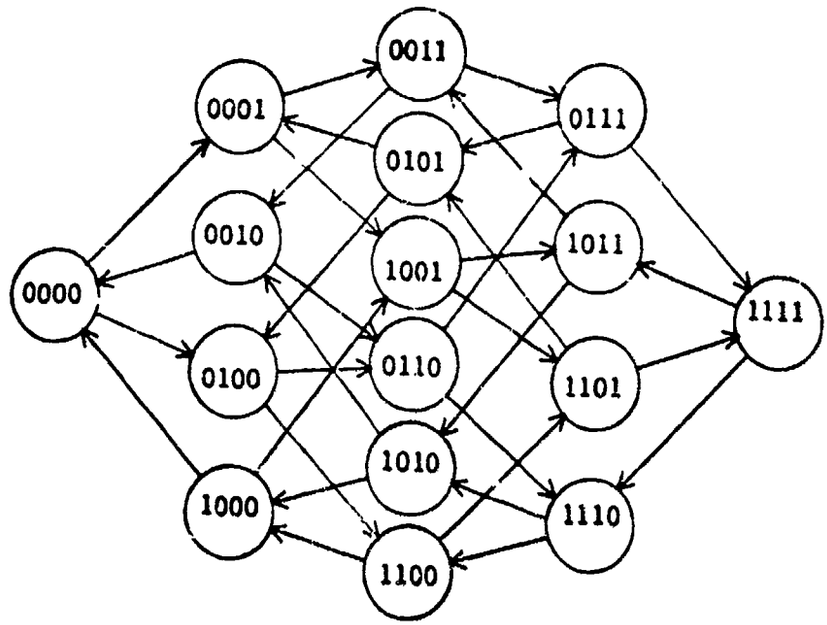


Figure 6. Digraph of Neighborhood Patterns

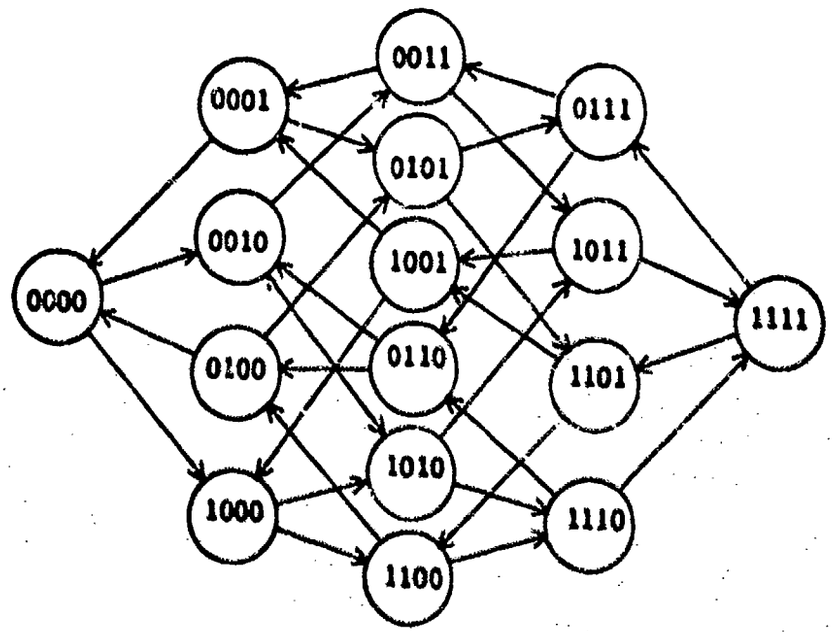
By the analogy between the digraph and the patterns of the states of four cells in a deleted neighborhood, we have the following relationships:

1. Each node of the digraph represents one MPNP.
2. Each arc of the digraph represents one MANP.

If there are Eulerian cycles in the digraph then MANPs and MPNPs will be generated within minimal number of memory operations because of this analogy. Since every node of the digraph has indegree and outdegree four, the digraph is an isograph. Now we know there are Eulerian cycles in the digraph, but to use this property more effectively, we will find a subgraph with the smallest possible positive indegree and outdegree in each node. We can decompose the digraph into two subgraphs with $id(V_i) = od(V_i) = 2$ for every V_i by properly choosing one arc from each symmetric arc pair. An example of two subgraphs after such decomposition is shown in Figure 7. If there is an arc between any two nodes in one subgraph then there is an arc with opposite direction between the same two nodes in the other subgraph. Because both of them are isographs there are Eulerian cycles in each of the subgraphs. Furthermore, for each Eulerian cycle in one subgraph there is a corresponding Eulerian cycle in the other subgraph which travels each node and arc in exactly the opposite direction. Notice that each of these subgraphs has 32 arcs and 16 nodes. From now on we will discuss one of the subgraphs, since the effect on the other subgraph can immediately be understood. We will choose the one in Figure 7(a).



(a) A Subgraph

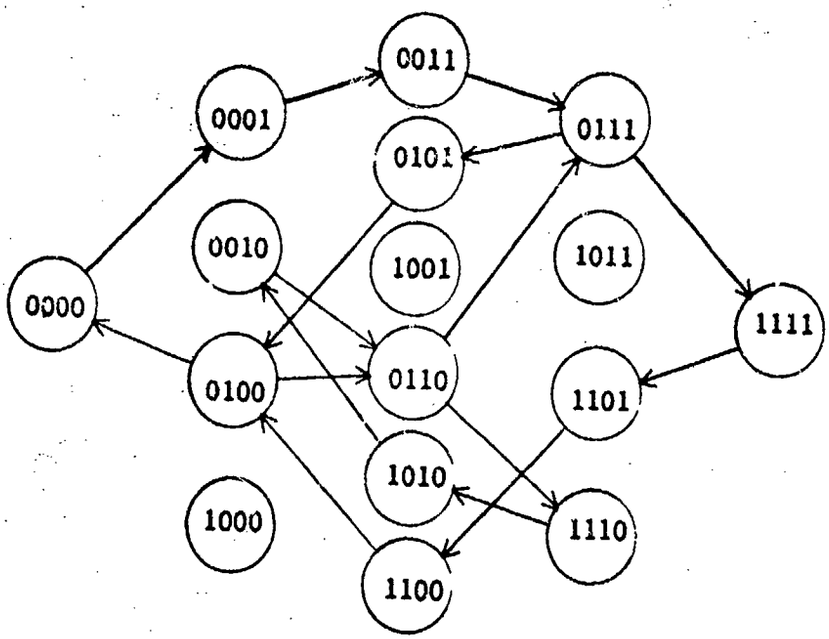


(b) Opposite Directional Graph of (a)

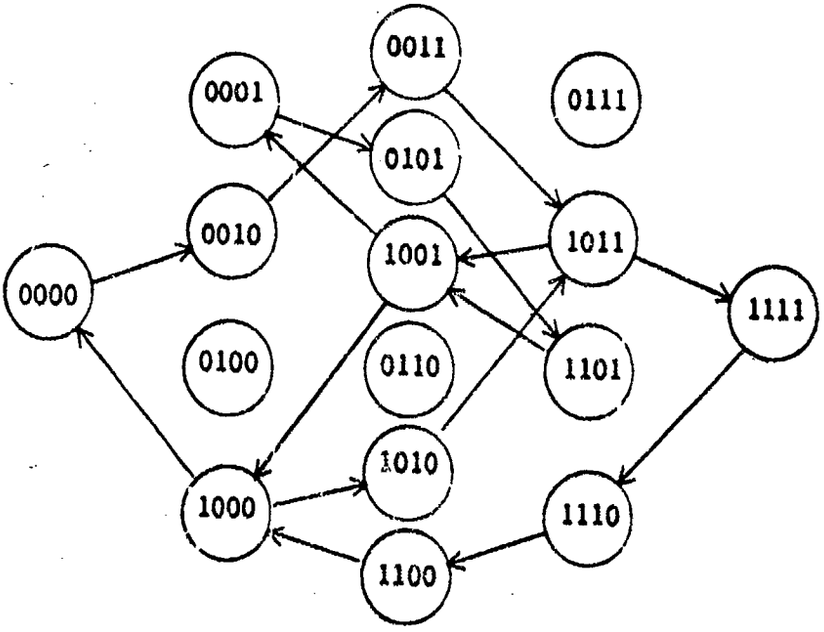
Figure 7. Subgraphs of Figure 6

It is possible to decompose the subgraph into two subgraphs with 16 arcs per each subgraph while maintaining the property of isograph in the resulting subgraphs, because the subgraph in Figure 7(a) has indegree and outdegree two for every node. There are many ways of decomposing the subgraph of Figure 7(a), but we will choose a way which satisfies the following requirement between the two subgraphs of 16 arcs: If there is an arc from (to) a node (a b c d) in one subgraph then the other subgraph must have an arc from (to) a node ($\bar{a}$ $\bar{b}$ $\bar{c}$ $\bar{d}$). Two subgraphs of the subgraph in Figure 7(a) satisfying the above requirements are given in Figure 8. The two subgraphs for the subgraph in Figure 7(b) can be obtained by simply changing the direction of each arc of the subgraphs in Figure 8.

Because each of these four graphs with 16 arcs is an isograph, there are Eulerian cycles. If we choose an Eulerian cycle, say Cycle zero, which starts from the node (0 0 0 0) in the subgraph Figure 8(a) then we can have an Eulerian cycle, say Cycle One, which starts from the node (1 1 1 1) and follows each node which is a complement of the corresponding node of Cycle Zero. With these Cycle Zero and Cycle One and corresponding cycles which have directions of all the arcs reversed from the Cycle Zero and Cycle One we can construct a sequence of memory operations which can minimally generate all 64 MANPs. But as one may notice from the property of an isograph, a minimal sequence which can generate all 64 MANPs can be directly constructed from the digraph of Figure 6. The usefulness of our four subgraphs of 16 arcs will be demonstrated in the next



(a) Cycle Zero



(b) Cycle One

Figure 8. Subgraphs of Figure 7(a)

section when we actually construct a test algorithm for NPSFs. We conclude this section by tabulating the subgraphs in the form of MANPs and MPNPs. They are shown in Tables 5 and 6. In Table 5 (Table 6) the row 0 (row $\bar{0}$) shows the initial pattern of A, B, C and D which corresponds to the initial node of Cycle Zero (One). Row 1 (row $\bar{1}$) through row 16 (row $\bar{16}$) correspond to the 16 arcs and their ending nodes of the Cycle Zero (Cycle One) in the order of its traveling in the subgraph. Row 17 (row $\bar{17}$) through row 32 (row $\bar{32}$) correspond to the 16 arcs and their ending nodes of the opposite directional cycle of Cycle Zero (Cycle One) in the order of its traveling in the subgraph. Note that each MPNP of Table 5 (Table 6) is an immediate consequence of the write operation of the MANP in the same row.

4.5 Detection and Location Algorithms

In this section we will present a test algorithm for the detection and location of NPSFs by using Table 5 and 6. Then we will show it actually detects and locates NPSFs. Each row of Table 5 and 6 contains one write operation on cells labeled with a specific symbol from A, B, C and D. We refer to row i as operation i where $1 \leq i \leq 32$ for Table 5 and we refer to row $\bar{i}$ as operation $\bar{i}$ where $\bar{1} \leq \bar{i} \leq \bar{32}$ for Table 6.

Algorithm NPSF

Step 1. Write zeros into every cell in S_{even} and write ones into every cell in S_{odd} and then read zeros from every cell in

Table 5

Sequence of Cycle Zero and its Opposite Directional Cycle

Operation	MANP				MPNP			
	A	B	C	D	A	B	C	D
0					0	0	0	0
1	0	0	0	↑	0	0	0	1
2	0	0	↑	1	0	0	1	1
3	0	↑	1	1	0	1	1	1
4	0	1	↓	1	0	1	0	1
5	0	1	0	↓	0	1	0	0
6	0	1	↑	0	0	1	1	0
7	↑	1	1	0	1	1	1	0
8	1	↓	1	0	1	0	1	0
9	↓	0	1	0	0	0	1	0
10	0	↑	1	0	0	1	1	0
11	0	1	1	↑	0	1	1	1
12	↑	1	1	1	1	1	1	1
13	1	1	↓	1	1	1	0	1
14	1	1	0	↓	1	1	0	0
15	↓	1	0	0	0	1	0	0
16	0	↓	0	0	0	0	0	0
17	0	↑	0	0	0	1	0	0
18	↑	1	0	0	1	1	0	0
19	1	1	0	↑	1	1	0	1
20	1	1	↑	1	1	1	1	1
21	↓	1	1	1	0	1	1	1
22	0	1	1	↓	0	1	1	0
23	0	↓	1	0	0	0	1	0
24	↑	0	1	0	1	0	1	0
25	1	↑	1	0	1	1	1	0
26	↓	1	1	0	0	1	1	0
27	0	1	↓	0	0	1	0	0
28	0	1	0	↑	0	1	0	1
29	0	1	↑	1	0	1	1	1
30	0	↓	1	1	0	0	1	1
31	0	0	↓	1	0	0	0	1
32	0	0	0	↓	0	0	0	0

Table 6

Sequence of Cycle One and its Opposite Directional Cycle

Operation	MANP				MPNP			
	A	B	C	D	A	B	C	D
0					1	1	1	1
1	1	1	1	↓	1	1	1	0
2	1	1	↓	0	1	1	0	0
3	1	↓	0	0	1	0	0	0
4	1	0	↑	0	1	0	1	0
5	1	0	1	↑	1	0	1	1
6	1	0	↓	1	1	0	0	1
7	↓	0	0	1	0	0	0	1
8	0	↑	0	1	0	1	0	1
9	↑	1	0	1	1	1	0	1
10	1	↓	0	1	1	0	0	1
11	1	0	0	↓	1	0	0	0
12	↓	0	0	0	0	0	0	0
13	0	0	↑	0	0	0	1	0
14	0	0	1	↑	0	0	1	1
15	↑	0	1	1	1	0	1	1
16	1	↑	1	1	1	1	1	1
17	1	↓	1	1	1	0	1	1
18	↓	0	1	1	0	0	1	1
19	0	0	1	↓	0	0	1	0
20	0	0	↓	0	0	0	0	0
21	↑	0	0	0	1	0	0	0
22	1	0	0	↑	1	0	0	1
23	1	↑	0	1	1	1	0	1
24	↓	1	0	1	0	1	0	1
25	0	↓	0	1	0	0	0	1
26	↑	0	0	1	1	0	0	1
27	1	0	↑	1	1	0	1	1
28	1	0	1	↓	1	0	1	0
29	1	0	↓	0	1	0	0	0
30	1	↑	0	0	1	1	0	0
31	1	1	↑	0	1	1	1	0
32	1	1	1	↑	1	1	1	1

S_{even} and read ones from every cell in S_{odd}.

Step 2. Do the following:

(a) Set i=1.

(b) Do the oepration i on S_{even}.

(c) Read all the cells written in (b).

(d) Read all the cells in S_{odd}.

(e) Do the operation $\bar{i}$ on S_{odd}.

(f) Read all the cells written in (e).

(g) Read all the cells in S_{even}.

(h) Increment i by one.

(i) If i=33 then go to (j), otherwise go back to (b).

(j) Reset i=13.

(k) Do the operation $\tilde{i}$ on S_{even}.

(l) Read all the cells written in (k).

(m) Read all the cells in S_{odd}.

(n) Do the operation i on S_{odd}.

(o) Read all the cells written in (n).

(p) Read all the cells in S_{even}.

(q) Increment i by one.

(r) If i=33 then go to (s), otherwise go to (t).

(s) Reset i=1 and go to (k).

(t) If i=13 then go to next step, otherwise go to (k).

Step 3. Write zeros into every cell in S_{odd} then read zeros from every cell in S_{odd} and S_{even}.

Step 4. Do the following:

(a) Set i=1.

(b) Do the operation i on S_{even}.

(c) Read all the cells written in (b).

(d) Read all the cells in S_{odd}.

(e) Do the operation i on S_{odd}.

(f) Read all the cells written in (e).

(g) Read all the cells in S_{even}.

(h) Increment i by one.

(i) If i=33 then go to (j), otherwise go back to (b).

(j) Reset i=13.

(k) Do the operation $\bar{i}$ on S_{even}.

(l) Read all the cells written in (k).

(m) Read all the cells in S_{odd}.

(n) Do the operation $\bar{i}$ on S_{odd}.

(o) Read all the cells written in (n).

(p) Read all the cells in S_{even}.

(q) Increment i by one.

(r) If i=33 then go to (s), otherwise go to (t).

(s) Reset i=1 then go to (k).

(t) If i=13 then the test is done, otherwise go to (k).

Theorem 9: The algorithm NPSF is a sufficient test to detect and locate all NPSFs in the memory.

Proof: We will prove the theorem for all ANPSFs and then the case for all PNPSFs will be proved. According to the observations in Section 4.3, for each base cell in S_{even} (S_{odd}) all the cells in its deleted neighborhood are in S_{odd} (S_{even}). Furthermore, union of the deleted neighborhoods of cells with the same symbol in S_{even} (S_{odd}) is S_{odd} (S_{even}). So each cell with the same symbol in S_{even} (S_{odd}) can be regarded as a cell in a deleted neighborhood of four different base cells in S_{odd} (S_{even}) as well as a base cell whose deleted neighborhood has all four cells in the S_{odd} (S_{even}).

1. ANPSFs detection and location: Because of the analogy between the MANP and the arcs of Figure 6, all 64 MANPs can be generated by applying the sequences of operations of Tables 5 and 6. In Algorithm NPSF, we are applying the sequences of operations of Tables 5 and 6 twice for a deleted neighborhood of each cell in the memory: Once in step 2 and another in step 4. Now it is clear that all MANPs are generated twice in the algorithm so only thing remained to be shown is that for each MANP generated by the algorithm the contents of its base cell contained both zero and one.

   Claim: If an MANP of interest for a base cell in S_{even} is generated in step 2 (step 4) and the content of the base cell at the moment the MANP is generated, is X, $X \in \{0,1\}$ then the same MANP will be generated in step 4 (step 2) too with the content of the base cell at the moment the MANP being generated $\overline{X}$.

Furthermore, all 128 ANPs are generated for each such cell and any ANPSF affecting the base cell is detected and identified.

Proof: Let the MANP be generated at (e) of kth iteration in step 2 then the same MANP has to be generated by the operation $\bar{k}$ of (n) in step 4, too. But (a) through (i) of step 2 and (j) through (t) of step 4 perform two different sequences of operations on S_{even}: one is the sequence of Table 5 and the other is the sequence of Table 6. Since Table 5 and Table 6 have states of each cell exactly complement to each other, the read operations on the base cell in (g) of step 2 following operations $\bar{k}$ on S_{odd} and in (p) of step 4 following operations $\bar{k}$ on S_{odd}, will show two complementary contents of the cell, otherwise the fault is detected and located. Therefore, every base cell in S_{even} is in two complementary states in each of step 2 and step 4 for the same MANP. The fact that all 128 ANPs are executed follows from the arguments above and Tables 5 and 6 and from the fact that each cell that was written into at each step and the cells that are potentially affected by such writes due to ANPSFs are read. We conclude that every ANPSF affecting a base cell in S_{even} is detected and identified.

By a similar claim and proof it can be shown that for every base cell in S_{odd} all the 128 ANPs are executed and all ANPSFs are detected and located.

2. PNPSFs detection and location: It is sufficient to show that a cell with each of four symbols has all 16 MPNPs when it performs

zero to one (one to zero) transition. Because of the symmetry among the four symbols A, B, C and D, the Observations 1, 2 and 3 in Section 4.3 and the symmetry of the cycles represented by Tables 5 and 6, we will only prove the theorem explicitly for the case of a cell with symbol A and a zero to one transition on this cell. The remaining cases follow due to the symmetries mentioned above.

Let an MPNP be denoted by an ordered quadruple (uvwx), where u, v, w and x are the contents of the cells with symbols A, B, C and D in the deleted neighborhood, respectively.

Each of the 16 MPNPs is applied to the deleted neighborhood of a cell in S_{even} with symbol A which is performing a zero to one transition as following:

(1) (0000) at the operation $\overline{21}$ of (k) in Step 4.

(2) (0001) at the operation $\overline{26}$ of (k) in Step 4.

(3) (0010) at the operation 24 of (b) in Step 4.

(4) (0011) at the operation $\overline{15}$ of (k) in Step 4.

(5) (0100) at the operation 18 of (b) in Step 4.

(6) (0101) at the operation $\overline{9}$ of (k) in Step 4.

(7) (0110) at the operation 7 of (b) in Step 4.

(8) (0111) at the operation 12 of (b) in Step 4.

(9) (1000) at the operation 12 of (b) in Step 2.

(10) (1001) at the operation 7 of (b) in Step 2.

(11) (1010) at the operation $\overline{9}$ of (k) in Step 2.

(12) (1011) at the operation 18 of (b) in Step 2.

(13) (1100) at the operation $\overline{15}$ of (k) in Step 2.

(14) (1101) at the operation 24 of (b) in Step 2.

(15) (1110) at the operation $\overline{26}$ of (k) in Step 2.

(16) (1111) at the operation $\overline{21}$ of (k) in Step 2.

Each of the 16 MPNPs are applied to the deleted neighborhood of a cell in S_{odd} with symbol A which is performing a zero to one transition as following:

(1) (0000) at the operation 12 of (n) in Step 2.

(2) (0001) at the operation 7 of (n) in Step 2.

(3) (0010) at the operation $\overline{9}$ of (e) in Step 2.

(4) (0011) at the operation 18 of (n) in Step 2.

(5) (0100) at the operation $\overline{15}$ of (e) in Step 2.

(6) (0101) at the operation 24 of (n) in Step 2.

(7) (0110) at the operation $\overline{26}$ of (e) in Step 2.

(8) (0111) at the operation $\overline{21}$ of (e) in Step 2.

(9) (1000) at the operation $\overline{21}$ of (e) in Step 4.

(10) (1001) at the operation $\overline{26}$ of (e) in Step 4.

(11) (1010) at the operation 24 of (e) in Step 4.

(12) (1011) at the operation $\overline{15}$ of (e) in Step 4.

(13) (1100) at the operation 18 of (e) in Step 4.

(14) (1101) at the operation $\overline{9}$ of (n) in Step 4.

(15) (1110) at the operation 7 of (e) in Step 4.

(16) (1111) at the operation 12 of (e) in Step 4.

Furthermore, if a cell has ANPSFs and PNPSFs at the same time then we can still identify each of these faults and locate the faults since each of ANPs and PNPs is individually generated for each cell and each of the potentially affected cell is read immediately after the generation of each pattern. Therefore, all NPSFs are detected and located by Algorithm NPSF.

Q.E.D.

The number of operations required for the application of Algorithm NPSF is 195.5N of which 3.5N operations are for the purpose of initialization and reset. Since 192N memory operations are minimially required to detect and locate all NPSFs, Algorithm NPSF is a near-minimal algorithm.

In the remainder of this section we will present test algorithms which can detect and locate each of ANPSFs and PNPSFs as two separate algorithms.

Algorithm PNPSF

Do the same procedures as Algorithm NPSF except (d), (g), (m) and (p) of Step 2, and (d), (g), (m) and (p) of Step 4.

Theorem 10: Algorithm PNPSF can detect and locate all PNPSFs.

Proof: Proof is similar to the proof for PNPSFs in the proof of Theorem 9.

Q.E.D.

Algorithm PNPSF requires 33.5N write operations and 34N read operations including initialization and reset.

Before presenting an algorithm for detection and location of ANPSFs, we will reorder the operations in Table 6 as following: (1) if $1 \leq i \leq 12$ then the operation $\overline{i}$ of Table 6 becomes operation i+52, and (2) if $13 \leq i < 32$ then the operation $\overline{i}$ of Table 6 becomes operation i+20.

Algorithm ANPSF

Step 1. Write zeros into every cell in S_{even} and then write zeros into every cell in S_{odd}. Next read zeros from every cell in S_{even}.

Step 2. Do the following.

(a) Set i=1.

(b) Do the operation i on S_{even} and then read all the cells in S_{odd}.

(c) Increment i by one.

(d) If i=13 then go to (e), otherwise go to (b).

(e) Reset i=1.

(f) Do the operation i on S_{odd} and then read all the cells in S_{even}.

(g) Increment i by one.

(h) If i=65 then go to (i), otherwise go to (f).

(i) Reset i=13.

(j) Do the operation i on S_{even} and then read all the cells in S_{odd}.

(k) Increment i by one.

(l) If i=65 then go to next step, otherwise go to (j).

Step 3. Do the following.

(a) Set i=1.

(b) Do the operation i on S_{odd}.

(c) Increment i by one.

(d) If i=13 then go to (e), otherwise go to (b).

(e) Read zero from each cell in S_{even} and reset i=1.

(f) Do the operation i on S_{even}.

(g) Increment i by one.

(h) If i=65 then go to (i), otherwise go to (f).

(i) Read one from each cell in S_{odd} and reset i=13.

(j) Do the operation i on S_{odd}.

(k) Increment i by one.

(l) If i=65 then go to (m), otherwise go to (j).

(m) Read zero from each cell in S_{even}.

Theorem 11: The algorithm ANPSF is a sufficient test to detect and locate all ANPSFs in the memory.

Proof: The proof is similar to that of Theorem 9, except that now we only consider ANPSFs.

Q.E.D.

The algorithm ANPSF requires 161.5N memory operations compared with minimal requirement of 160N memory operations so it is also close to the minimum.

4.6 Detection Algorithms

It is common to most users of RAMs that they test RAMs to detect the faults not to locate the faults, so finding algorithms which can detect ANPSF and NPSF with shorter test lengths is worthwhile. In the algorithm given below the renaming of the rows of Table 6 as given on page 72 is assumed.

Algorithm DANPSF

Step 1. Write zeros into every cell of S_{even} and then write zeros in every cell of S_{odd}. Next read zeros from every cell in S_{even}.

Step 2. Do the following.

(a) Set i=1.

(b) Do the operation i on S_{even} and then read all the cells in S_{odd}.

(c) Increment i by one.

(d) If i=65 then go to (e), otherwise go to (b).

(e) Reset i=1.

(f) Do the operation i on S_{odd} and then read all the cells in S_{even}.

(g) Increment i by one.

(h) If i=65 then go to next step, otherwise go to (f).

Step 3. Write ones into every cell of S_{even}.

Step 4. Do the following.

(a) Set i=1.

(b) Do the operation i on S_{odd}.

(c) Increment i by one.

(d) If i=13 then go to (e), otherwise go to (b).

(e) Read a one from every cell in S_{even}.

(f) Do the operation i on S_{even}.

(g) Increment i by 1.

(h) If i=65, then let i=1 and go to (f) and if f=13, go to (j), otherwise go to (f).

(j) Read a one from every cell in S_{odd}.

(k) Do the operation i on every cell in S_{odd}.

(l) Increment i by 1.

(m) If i=65, then go to (n), otherwise go to (k).

(n) Read a one from every cell in S_{even}.

(o) End.

Theorem 12: The algorithm DANPSF can detect all ANPSFs in the memory.

Proof: If an MANP with base cell containing zero results in a fault then it will be detected at step 2 regardless of the existance of the fault caused by an MANP with base cell containing one. If no faults are detected in step 2 and an MANP with base cell containing one results in a fault then it will remain and be detected at step 4 since no ANPSFs with base cell containing zero exist.

Q.E.D.

The algorithm DANPSF requires total of 99.5 memory operations. If we compare this number with 97N for minimal required operations, the length of the algorithm is close to the minimum.

As we can find in most of the recent literature [3,6,9,11,16] it is more probable in RAMs that a fault may occur in one direction, either zero to one or one to zero. Therefore unidirectional ANPSFs are of interest.

Tha algorithm given below detects the faults covered by Theorem 7.

Algorithm UANPSF

Step 1. Write zeros into every cell of S_{even} and then write zeros into every cell of S_{odd}. Read zeros from every cell in S_{even}.

Step 2. Do the following.

(a) Set i=1.

(b) Do the operation i on S_{even}.

(c) Increment i by one.

(d) If i=65 then go to (e), otherwise go to (b).

(e) Read zeros from all the cells in S_{odd}.

(f) Reset i=1.

(g) Do the operation i on S_{odd}.

(h) Increment i by one.

(i) If i=65 then go to (j), otherwise go to (g).

(j) Read zeros from all the cells in S_{even}.

Step 3. Write ones into every cell in S_{even} and read zeros from every cell in S_{odd}.

Step 4. Do the following.

(a) Set i=1

(b) Do the operation i on S_{odd}

(c) Increment i by one

(d) If $i \leq 64$ go to b

(e) Read one from every cell in S_{odd}

Step 5. Write zeroes into every cell in S_{even} and then write ones into every cell in S_{odd}.

Step 6. Do the following.

(a) Set i=1

(b) Do the operation i on S_{even}

(c) Increment i by one

(d) If $i \leq 64$ go to b

(e) Read one from every cell in S_{odd}

Theorem 13: The algorithm UANPSF is a sufficient test for detection of ANPSFs in a RAM if whenever it is faulty then there exists at least one unidirectional faulty MANP for some base cell.

The algorithm requires 34.5 write operations and 3 read operations per each cell and 2.5 of the write and two of the read operations are for initialization and reset. From Theorem 7 the algorithm is also found to be near minimal.

The near minimality of the algorithms presented in this chapter is based on the assumption that every cell in the memory has 4 cells in its deleted neighborhood. If we take the cells at the corners and on the edges of the rectangular array memory into account then the actual lower bound on the number of required operations for each fault coverage will be less than the one given in Section 4.2 by $O(N^{1/2})$ operations.

CHAPTER V

CONCLUSIONS

5.1 Summary

In the areas of testing large scale integrated circuit random access memories, inadequacies in most of currently available test algorithms which can efficiently test large scale integrated circuit random access memories, memory faults were divided into functional faults, pattern sensitive faults and DC parametric faults.

Coupling faults turned out to be a type of functional faults which required impractically long test sequence. Functional coupling faults are classified into symmetric coupling faults and asymmetric coupling faults. By use of these two types of coupling faults we devided the coupling faults into three categories. March type test has a few advantages over other types of tests in testing functional faults. By assuming that only the march type operations are used in test algorithms to test coupling faults, we found a lower bound on the number of operations which can detect Category I coupling faults. By finding some crucial necessary conditions in testing a memory of four cells we determined sets of required march elements. From the sets, a minimal test sequence, Algorithm A, was found and shown to be sufficient. Algorithm A is a minimal march test which can detect all Category I coupling faults. Several modifications of Algorithm A for broader fault models were developed.

Overlapping neighborhood problem together with determination of proper neighborhood type was the main difficulty in finding a practical test algorithm for pattern sensitive faults. A neighborhood of five cells, which includes a base cell and its four adjacent cells which are located to the top, bottom, left and right of the base cell were suggested earlier as a reasonable choice for pattern sensitive faults [27]. Two categories of pattern sensitive faults for this neighborhood were defined and near minimal algorithms to detect and locate these faults were found.

REFERENCES

1. Abbott, Robert A., Regitz, William M., and Karp, Joel A., "A 4K MOS Dynamic Random-Access Memory," IEEE J. Solid-State Circuits, Vol. SC-8, No. 5, pp. 292-298, Oct. 1973.

2. Barraclough, W., Chiang, A. C. L., and Sohl, W., "Techniques for Testing Microcomputer Family," Proc. IEEE, Vol. 64, pp. 943-950, June 1976.

3. Barrett, C. R., and Smith, R. C., "Failure Modes and Reliability in Dynamic RAMs," Digest of Papers Compcon Spring 77, February 28 - March 3, IEEE Comp. Soc., pp. 179-182.

4. Boonstra, Loek, Lambrechtse, Cees W., and Salters, Roelof H. W., "A 4096-b One-Transistor Per Bit Random-Access Memory with Internal Timing and Low Dissipation," IEEE J. Solid-State Circuits, Vol. SC-8, No. 5, pp. 305-310, Oct. 1973.

5. Breuer, M. A., and Friedman, A. D., "Diagnosis and Reliable Design of Digital Systems," Computer Science Press, Inc., 1976, pp. 139-146 and 156-170.

6. Brown, J. R., Jr., "1103 Semiconductor Memory Device Sensitivity and Error Modes," Digest of Semiconductor Test Symp., IEEE Comp., October 1973, pp. 63-76.

7. Brown, J. R., Jr., "Pattern Sensitivity in MOS Memories," Dig. of Symp. Testing to Integrate Semiconductor Memories into Computer Mainframes, October 1972, pp. 33-46.

8. Brown, J. R., Jr., "Timing Peculiarities of Multiplexed RAMs," Computer Design, July 1977, pp. 85-92.

9. Cocking, J., "RAM Test Patterns and Test Strategy," Digest of Papers, 1975 Symposium on Semiconductor Memory Testing, IEEE Comp. Soc., October 1975, pp. 1-8.

10. Cohen, L., Green, R., Smith, K.,and Seely, J. L., "Single-transistor Cell Makes Room for more Memory on an MOS Chip," Electronics, August 2, 1971, pp. 69-75.

11. Fee, W.G., "Memory Testing," LSI Testing Tutorial, Compcon Spring 77, February 28 - March 3, 1977, San Francisco, California, IEEE Comp. Soc., pp. 51-58.

12. Fisher, J. E., "Test Problems and Solutions for 4K RAMs," Digest of Semiconductor Test Symp., IEEE Comp. Soc., November 1974, pp. 53-71.

13. Harary, F.,Norman, R. Z., and Cartwright, D., "Structrual Models: An Introduction to the Theory of Directed Graphs," John Wiley & Sons, New York, 1965.

14. Hayes, J. P., "Detection of Pattern-Sensitive Faults in Random Access Memories," IEEE Trans. Comp., Vol. C-24, Feb. 1975, pp. 150-157.

15. Hnatek, E. R., "r-Kilobit Memories Present a Challenge to Testing," Computer Design, May 1975, pp. 117-125.

16. Hnatek, E. R., and Schmitt, R. G., "A 2048 bit RAM Characterization Program," Digest of Papers. 1976 Symposium on Semiconductor Memory Testing, IEEE Comp. Soc., October 1976, also in LSI Testing Tutorial, Compcon Spring 77, Feb. 28 - Mar. 3, San Francisco, IEEE Comp. Soc., pp. 77-83.

17. Hodges, D. A., "Semiconductor Memories," IEEE Press, New York, 1972.

18. Hoffman, William K., and Kalter, Howard L., "An 8K b Random Access Memory Chip Using the One-Device FET Cell," IEEE J. Solid-State Circuits, Vol. SC-8, No.5, pp. 298-305, Oct. 1973.

19. The INTEL Memory Design Handbook, August 1973.

20. de Jonge, J. H., and Smulders, A. J., "Moving Inversions Test Pattern is Thorough, Yet Speedy," Computer Design, May 1976 and also in LSI Testing Tutorial, Compcon Spring 77, Feb. 28 - Mar. 3, San Francisco, IEEE Comp. Soc., pp. 119-126.

21. Knaizuk, J., Jr., and Hartmann, C. R. P., "An Optimal Algorithm for Testing Stuck-at Faults in Random Access Memories," IEEE Trans. Comp., Vol. C-26, Nov. 1977, pp. 1141-1144.

22. Kuo, C., Kitagawn, N., Ward, E., and Drayer, P., "Sense Amplifier Design is Key to 1-Transistor Cell in 4K bit RAM," Electronics, September 13, 1973, pp. 116-121.

23. Luecke, G., Mize, J. P., and Carr, W. C., Semiconductor Memory Design and Application, McGraw Hill, New York, 1973.

APPENDIX A:

A Test Procedure to Detect Abnormal Access Times in Semiconductor Random Access Memories

ABSTRACT

Fundamental requirements on test procedures to detect abnormal timing parameters in semiconductor random access memories are derived. Optimal test procedures to detect abnormal access times are then given.

I. Introduction

Rapid developments in semiconductor technology have made larger and denser semiconductor memories on a single chip a reality. As more and more memory cells are packed into a single chip the number of failure modes increases and the need for efficient algorithms to detect faults in them becomes more critical. One of the more difficult fault diagnosis problems is to detect what are known as pattern sensitive faults [1-4]. In this paper we consider pattern sensitive faults affecting dynamic timing parameters of semiconductor read/write random access memories (henceforth called RAMs). We assume that RAMs under test have $N = 2^K$, $K > 0$, works.

II. Problem Formulation

For the purposes of this paper it is convenient to visualize a RAM as shown in the block diagram given in Figure 1 [4]. In some memories the row and column addresses are time multiplexed onto the same pins of a chip. The problem considered in this paper is the generation of test patterns to verify timing parameters (such as access times, write recovery time, etc.) of RAM under test [4,5]. The timing parameters tend to depend on (a) data in the memory cell being accessed and (b) address transitions or changes (due to noise on address lines,

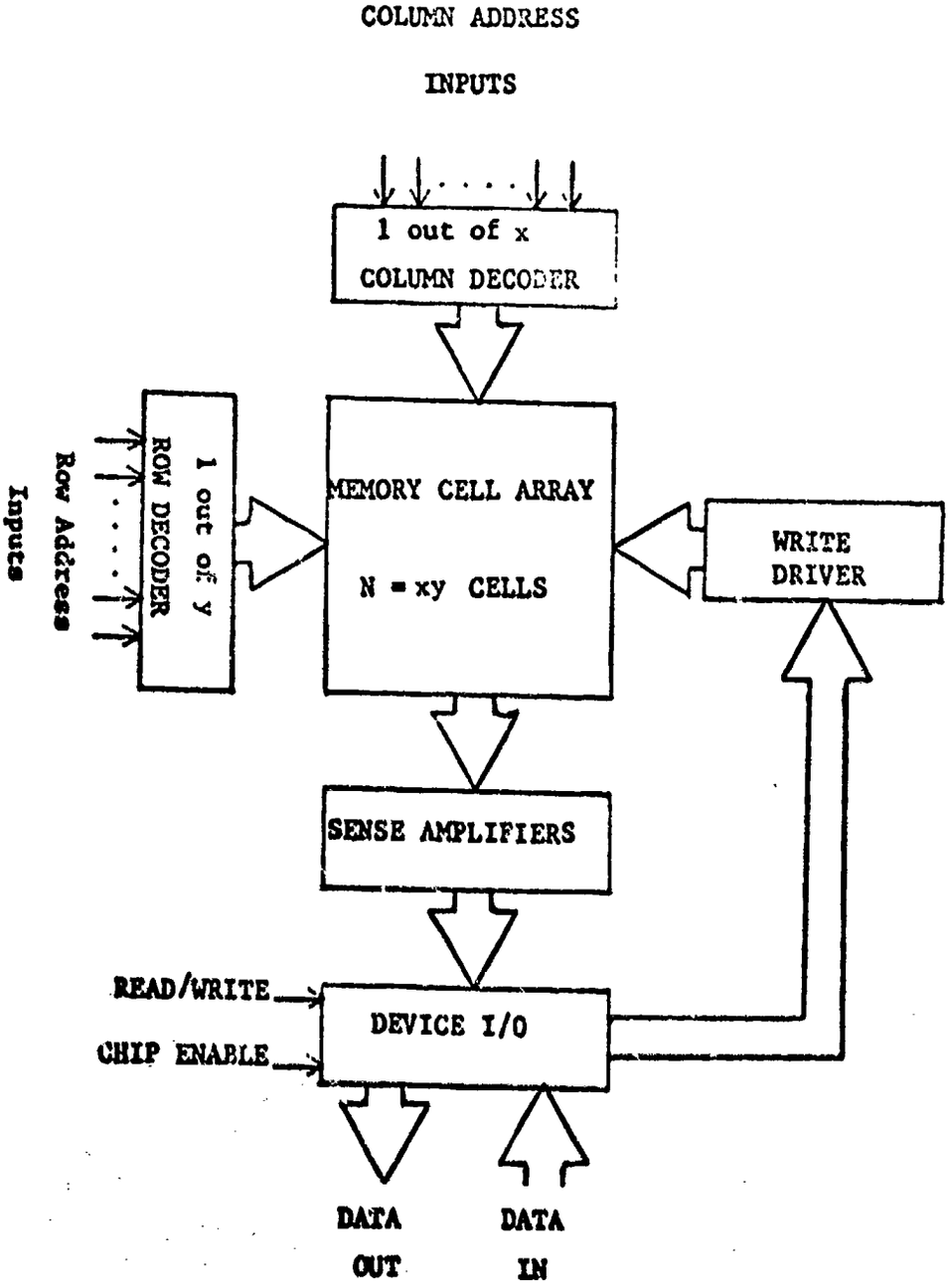


Figure 1. Block Diagram of a Semiconductor RAM

pattern dependent propagation delays in address decoders, ecc.) and hence are pattern sensitive [4,5]. Earlier several algorithms to detect abnormal timing parameters were given [4]. We will exhibit deficiencies in these algorithms and then give new algorithms. For the sake of simplicity and brevity, the test algorithm for verifying access time of a RAM only will be discussed.

Definition 1 [4]. The length of time from the appearance of a valid address on the pins to the appearance of valid data on the data output pins in a READ operation is called access time, t_{acc}.

The maximum value of t_{acc}, $t_{acc_{max}}$, specifies how long a wait is required before valid data is available after the address was provided. For RAMs with time multiplexed row and column addresses the valid address is assumed to be given after second half of the address was provided.

Due to variation in propagation delays in the decoder logic (which potentially cause hazards) and noise generated in the decoders when the logic levels on the address lines change, the following faults could occur [4]:

(a) neither the addressed memory cell nor any other cell is read, or

(b) the addressed cell is not read, but some other cell or cells are read, or

(c) the addressed cell and some other cell or cells are read at the end of t_{acc} specified for the RAM under test.

Depending on the details of realization of a RAM, when more than one cell is read (due to errors) either the logical OR or the AND of the

contents of the cells read appears at the output. Furthermore when no cells are read (again due to errors) the output could be zero or one, again depending on the details of the realization of the RAM under test. In the next section we give test procedures to detect excessive t_{acc}.

III. Test Procedure

The worst case condition for t_{acc} has been established earlier [4,5]. The worst case address changes are the ones that cause all address lines to change (e.g., address changing from 00...0 to 11...1). An algorithm, to detect excessive t_{acc}, that required 5N operations (an operation is a READ or WRITE) on a RAM under test was given earlier [4]. It can be shown that this algorithm does not cover the faults modeled. The basic reason for this shortcoming is that the algorithm does not satisfy a fundamental requirement we establish next.

Note that the fault model given earlier implies that when a cell is being read there is a possibility of reading more than one cell and when this occurs the output is the logical OR or AND of the contents of all the cells read. For a given RAM however it is either logical OR or AND and it can be assumed known. If we assume that logical OR of the contents of the cells is what occurs then any test procedure for detecting excessive t_{acc} must admit the following fundamental requirements.

Requirement 1: If a test for a RAM requires reading a particular location in RAM and there is a potential to unintentionally READ other location in the RAM, then the desired location should contain a zero and the "potentially readable" locations must contain 1, prior to the initiation of READ operation for the desired location; only then the

multiple accesses can be detected. Since the worst case situation for t_{acc} requires the changing of all address lines "the other potentially readable" locations include all the locations in the memory.

A similar requirement (say Requirement 2) can be obtained for the case when AND of the contents of the multiple cells read is the output. We obtain this by interchanging 1 with 0 and vice versa in Requirement 1.

Test Procedures for RAMs for Which Address Bits Are Not Time Multiplexed

Procedure A: (for RAMs for which OR of the contents of multiple cells read is the output).

1. WRITE ones into all locations and let i = 0.
2. WRITE a zero into location i.
3. READ a one from location N-1-i (note that the address bits of location i and N-1-i are complements of each other hence all address bits change from their previous values thus creating the worst case situation for t_{acc}).
4. READ a zero from location i (same comments as for step 3, and the note that this is the step that is required by Requirement 1).
5. WRITE a one into location i.
6. Increment i modulo N and if i is not zero go to 2, otherwise done.

Procedure A requires 5N operations. For RAMs for which multiple cell reads lead to AND of their contents Procedure B given below is required.

Procedure B:

Same as Procedure A except interchange 0 with 1 and 1 with 0.

Test Procedures for RAMs with time multiplexed address bits

Let K (note $N = 2^K$) be an even number (this is the current trend in the development of RAMs). Let the left most K/2 bits of the address correspond to the row address and the right most K/2 bits correspond to the column address. Further assume that the row address is given first and then the column address. Further assume that the row address is given first and then the column address. To create "worst case" conditions for t_{acc} we must now make the column address be the 1's complement of the row address and vice versa. Let the address of a cell be represented by an ordered pair (i,j) where i is its column address and j its row address (note that i and j are binary K/2-tuples) and let $\bar{j}$ be the bit by bit complement of j.

Procedure C: (for RAMs with time multiplexed address bits and for which OR of the contents of multiple accessed cells is the output).

1. WRITE 1's into every cell and set i = 0.
2. WRITE 0 into the cell with address $(i,\bar{i})$.
3. READ one from the cell with address $(\bar{i},i)$.
4. READ zero from the cell with the address $(i,\bar{i})$.
5. WRITE one into the cell with address $(i,\bar{i})$.
6. Increment i by one modulo $2^{K/2} = \sqrt{N}$ and if $i \neq 0$, go to 2.
7. Set m = 0.
8. WRITE 0 into the cell with address m (note that now m is a binary K-tuple).
9. READ one from the cell with address N-1-m.
10. READ zero from the cell with address m.
11. Increment m by one modulo 2^K and if $i \neq 0$ go to 8, otherwise done.

Note that Procedure C requires $4N + 4\sqrt{N}$ operations on the RAM under test. For RAMs for which AND of the contents of multiple assessed cells is the output, Procedure D given below is to be applied.

Procedure D:

Same as Procedure C, except interchange 0 with 1 and 1 with 0.

Similar test procedures for other timing parameters (write enable time, data set-up and release time, write recovery time, etc.) can be given. All these procedures will be derived to admit Requirements 1 and 2 given earlier. Furthermore it can be shown that the test procedures given are optimal, in the sense that they require minimum number of operations.

IV. Conclusions

Fundamental requirements on test procedures to detect abnormal timing parameters of semiconductor RAMs have been established and optimal algorithms to detect abnormal timing parameters have been given.

REFERENCES

1. Brown, J.R., Jr., "Pattern Sensitivity in MOS Memories", Dig. of Symp. Testing to Integrate Semiconductor Memories into Computer Mainframes, October 1972, pp. 33-46.
2. Hayes, J.P., "Detection of Pattern-Sensitive Faults in Random Access Memories", IEEE Trans. Comp., Vol. C-24, Feb. 1975, pp. 150-157.
3. Suk, D.S., "Functional and Pattern Sensitive Fault Testing Algorithms for the Semiconductor Random Access Memories", Ph.D. Thesis, Electrical Engineering, University of Iowa, Iowa City, Iowa, July 1978.

4. Thatte, S.M. and Abraham, J.A., "Testing of Semiconductor Random Access Memories", Proceedings of the 7th Annual Int. Conf. on Fault-Tolerant Computing, June 1977, pp. 81-87.

5. Springer, J., "Making Sense Out of Delay Specs in Semiconductor Memories", Electronics, pp. 82-88, October 25, 1971.

APPENDIX B

A PROCEDURE TO DETECT STUCK-AT-FAULTS IN DYNAMIC RANDOM ACCESS MEMORIES*

S. M. Reddy
Division of Information Engineering
University of Iowa
Iowa City, IA 52242

Abstract

A fault model for failure modes due to stuck-at-faults in semiconductor dynamic random access memories is given. Two test procedures to detect modeled faults are presented and their fault coverage is analyzed.

I. INTRODUCTION

Several researchers have proposed tests to detect stuck-at-faults in random access memories (RAMs) [1-3]. One of the basic assumptions made in deriving these tests is that when multiple memory cells are accessed due to, say faults in the address decoders, then the output of the RAM under test is the logical AND or logical OR of the contents of the accessed cells [1-3]. It can be readily verified that this assumption is valid only for single faults in the address decoders for the currently made single device per cell dynamic RAMs. Even then both logical AND and logical OR effect could occur, for different faults, in the same RAM under test. Multiple stuck-at faults in a RAM leading to an address accessing multiple cells, could lead an output which is a unate function of the contents of the accessed cells. We will first model the effect of stuck-at faults in RAMs and then give test procedures to detect modeled faults.

II. FAULT MODEL

A block diagram appropriate for currently manufactured one device per cell semiconductor dynamic RAMs is given in Figure 1. In this block diagram details regarding address multiplexing and input/output are not shown. One of the characteristics of these RAMs that is important, to develop the fault model, is that the sense amplifiers divide the memory array into two halves and that typically in one-half of the array a logic 1 is stored as a high voltage across a storage capacitor, while in the other half a 1 is stored as zero or low voltage across the storage capacitor. The details of READ operation in these RAMs can be explained by referring to Figure 2. Note that the cells on the same digit line but on the opposite sides of a sense amplifier/latch are labeled digit and digit lines. The information is stored as charge in a storage capacitor C_S. Each cell consists of a storage capacitor, C_S, and the transistor connected to it. The dummy or reference cells on either side of the sense amplifier have a storage capacitor C_R, whose value is half that of C_S. The cells on the digit side store logic one as high voltage and a logic zero as a low voltage, whereas the voltages corresponding to stored values on the digit side are reversed. When reading a cell on the digit side, appropriate (depending on the address decoder outputs) digit/digit line is precharged high and the storage capacitor of the dummy cell on the digit side is set to zero and then the selected cell and the dummy cell on the digit side are connected to the digit/digit line by turning on the associated transistors, selected by the word line (which in turn is selected by the address decoders). The digit line is pulled down due to charging of the capacitor in the dummy cell and digit line is pulled down to a lower voltage if the selected storage capacitor was storing a zero, otherwise digit line stays high. Next the latch in the sense amplifier is set and takes a state determined by the relative voltage values of the digit and digit line. Similar actions occur when a cell on the digit side is chosen, except now the dummy cells on the digit side are involved. If the output is derived from the same side of the latch the inversion introduced by the stored voltage on the digit side is automatically corrected.

Let us assume now that due to faults, for example in the address decoders, two cells C_1 and C_2 on the same digit/digit line are selected. Two of the possible cases are shown in Figure 3. In deriving the tables in Figure 3, it is assumed that the output is derived from the digit side of the sense amplifier latch. Note that the two faults depicted in Figure 3 could be caused by different single faults in the address decoders. In case (a) the output is the AND of the contents of cells C_1 and C_2, whereas the output in case (b) is $(C_1) + (C_2)$. However, if we remember that the cells in digit side store complement data, the output in case (b) can be considered as $(C_1) \cdot (C_2)$ if we now assume the contents of cells to be the expected value at the output under fault-free conditions. Now if we assume that multiple faults can occur, it can be readily argued that the output when accessing at an address can be a positive unate function of the contents of the several cells. It appears that if stuck-at faults do not lead to faulty behavior that exhibits memory, then the output due to multiple stuck-at faults will be a positive unate function of cells accessed. If we assume such is the case then the Fault Model S given below will

* The research reported was supported by Air Force Office of Scientific Research under Grant No. AFOSR-78-3582 and by Rome Air Development Center Contract F30602-78-C-0041.

cover all failure modes in a semiconductor RAM afflicted with stuck-at faults.

Definition 1: By logical content of a memory cell C_i we mean the expected output when that cell alone is read (this definition allows us to neglect the inversion in value stored in cells on the digit side).

Fault Model S

When an address, say A_i, is applied to a RAM no cell is accessed or more than one cell is accessed or a constant 1 or 0 occurs at the output whenever a READ operation is performed at this address. Furthermore, when more than one cell is accessed then the output (when read) is a positive unate function of the logical contents of the cells accessed.

III. TEST PROCEDURES

Recently a test procedure requiring 5N-2 READ/WRITE operations on the RAM under test was proposed [3]. It was proved that this test procedure detects all multiple stuck-at faults in the RAM if the output when multiple cells are accessed is the OR or AND of the logical contents of the accessed memory cells. The test procedure is given in Figure 4 and we will call it Test Procedure 1. In Figure 4 R_0, R_1, W_0 and W_1 stand for reading a zero, reading a 1, writing a zero and writing a 1, respectively, at the address indicated.

The following new result can be proved with regard to Test Procedure 1 when applied to RAMs that exhibit faults modeled by Fault Model S.

Theorem 1: If the faults in a RAM under test are those modeled by Fault Model S, then Test Procedure 1 detects all faults if there are no undetectable faults in the RAM under test.

Proof: Let S_i be the set of cells accessed by address A_i (note that under fault-free condition $S_i \cap S_j = \emptyset$ and $|S_i| = 1$ for all i and j, $i \neq j$).

Case 1: Some S_i is empty. In this case when the memory is read at the corresponding address, the output will be a constant and hence the fault is detected since a 1 and 0 are read at each address in Test Procedure 1.

Case 2: No S_i is empty and $S_i \cap S_j = \emptyset$ for $i \neq j$. The assumption implies that $|S_i| = 1, \forall i$, $0 \leq i \leq N-1$, and hence the only faults possible in this case are those that correspond to a memory cell content stuck-at-1 or stuck-at-0. Again these faults when present are detected by test procedure 1.

Case 3: No S_i is empty and $S_i \cap S_j \neq \emptyset$ for some $i \neq j$. Let A_k be the highest address such that cells accessed by A_k are not accessed by any address higher than A_k. Let S_k be the union of disjoint subsets S_{ks}, S_{kc} and S_{ka}, where S_{ks} is the subset of cells whose contents are stuck-at a constant, S_{kc} is the subset of cells accessed by A_k and addresses lower than A_k and S_{ka} is the subset of cells accessed by A_k alone. At the time a zero is attempted to be read at A_k, when Test Procedure 1 is applied, the logical contents of all the cells in S_{kc} will contain 1, cells in S_{ks} will contain some constants and cells in S_{ka} will contain 0. If a zero was correctly read then since the output is a positive unate function of the logical contents of cells in S_k, we can conclude that the output will be a zero independent of the contents of S_{kc} (these cells contained 1 at test time and even if the contents of some or all of the cells in S_{kc} were to change to 0, the truth value of the positive unate function will remain at zero). Hence if S_{ka} is empty then the output will always be zero when A_k is read and hence the fault will be detected when a 1 is expected to be read from A_k. If S_{ka} is nonempty then it implies that once a zero is written into the RAM at address A_k, then a zero will be correctly read independent of other write operations on the RAM and before a 1 is written at A_k. Similar arguments related to reading of 1 at A_k in Test Procedure 1 will lead to the conclusion that either the faults that caused A_k to access multiple cells are detected or else once a 1 or a zero is written at A_k, it is correctly preserved and reproduced. In the latter case we have exhibited an undetectable fault.

Q.E.D.

The Test Procedure 2 given below detects all modeled faults.

Test Procedure 2:

1. Set i = 0
2. Write zero at A_i and then write 1 at all other addresses
3. Read zero at A_i
4. Write one at A_i and then write 0 at all other addresses
5. Read one at A_i
6. Increment i and if i < N, go to 2
7. END

Test Procedure 2 requires $2N^2$ write and 2N read operations.

Theorem 2: When Test Procedure 2 is applied to a RAM under test, all faults modeled by Fault Model S are detected.

Proof: Adopting the notation used in the proof of Theorem 1, we conclude that the proof for cases 1 and 2 are identical even when Test Procedure 2 is used. Therefore, let us assume that no S_i is empty and that $S_i \cap S_j \neq \emptyset$ for some $i \neq j$. Since the number of cells in the memory is equal to the number of addresses, we conclude that there exists an address A_k such that every

cell in S_k is also accessed by some other address. Again adopting the notation used in the proof of Theorem 1, let $S_k = S_{ks} \cup S_{kc}$. At the time when a zero is read at A_k, in the Test Procedure 2, the logical contents of all cells in S_{kc} contain a 1 and if a zero is correctly read then a zero will always be read from A_k (again because the output is assumed to be a positive unate function of the logical contents of cells in S_{ks} and S_{kc} and hence even if the contents of some or all cells in S_{kc} change to zero the truth value of the positive unate function will remain 0). Therefore, the fault is detected when a 1 is attempted to be read at A_k.

Q.E.D.

IV. CONCLUSIONS

A fault model for failure modes due to stuck-at faults in currently made one device per cell semiconductor dynamic RAMs was given. Two procedures to detect modeled faults were analyzed. The test procedure which detects all modeled faults requires $2(N^2 + N)$ read/write operations. It appears that to derive test procedures needing fewer operations requires a more detailed analysis of the effect of stuck-at faults in these RAMs.

V. REFERENCES

1. J. Knaizuk, Jr., and C.R.P. Hartmann, "An algorithm for testing random access memories," IEEE Trans. Comput., Vol. C-26, pp. 414-416, April 1977.

2. J. Knaizuk, Jr. and C.R.P. Hartmann, "An optimal algorithm for testing stuck-at faults in random access memories," IEEE Trans. Comput., Vol. C-26, pp. 1141-1144, Nov. 1977.

3. R. Nair, Comments on "An optimal algorithm for testing stuck-at faults in random acces memories," IEEE Trans. Comput., Vol. C-28, pp. 258-261, March 1979.

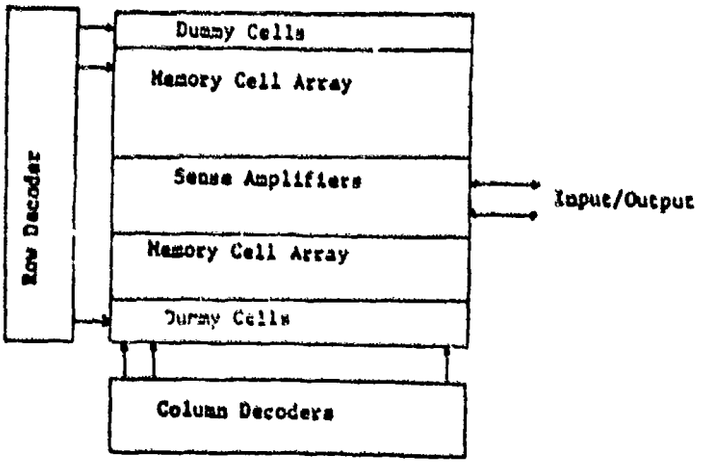


Figure 1. Block diagram of a P'M.

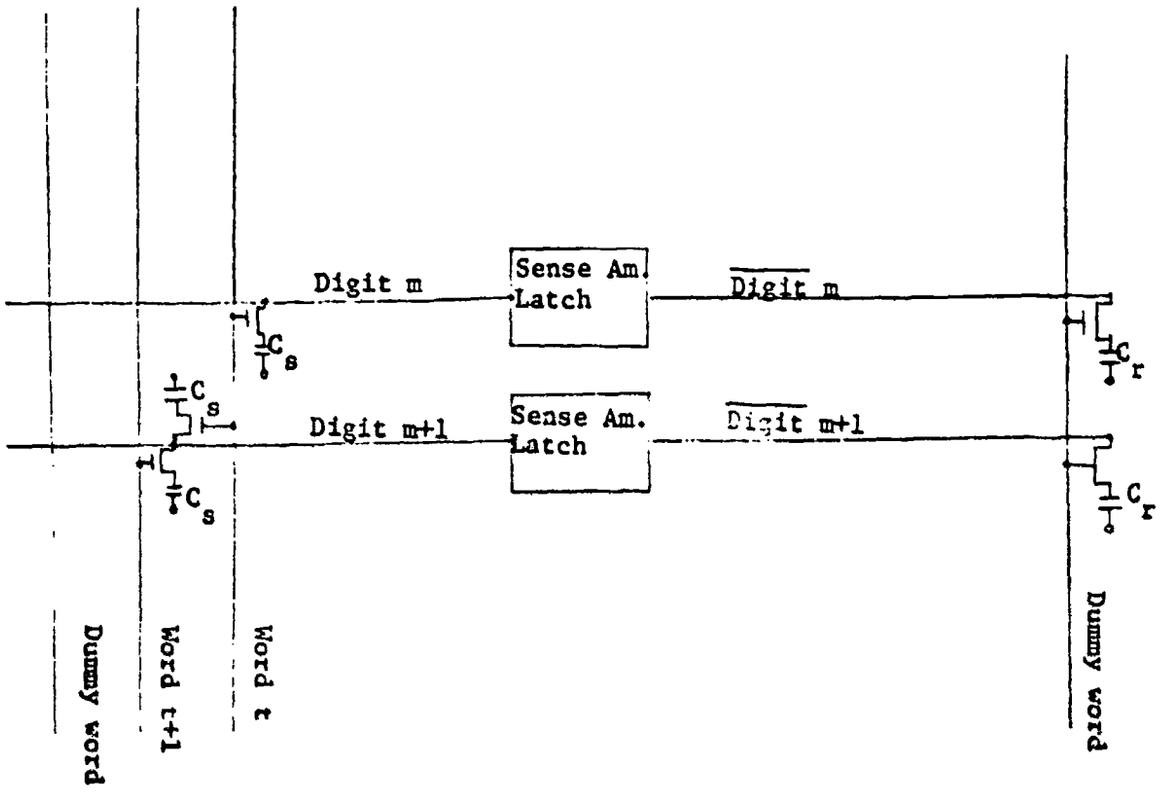


Figure 2: Illustrating reading.

(C_1)*	(C_2)	Output
0	0	0
0	1	0
1	0	0
1	1	1

(a) C_1 and C_2 both on the Digit side.

(C_1)	(C_2)	Output
0	0	1
0	1	0
1	0	1
1	1	1

(b) C_1 on the Digit side and C_2 on the $\overline{\text{Digit}}$ side.

* (C_i) = Contents of C_i.

Figure 3: Faulty behaviour of RAM

Step	1	2	3 •••	N-1	N	N+1	N+2	3N-5	3N-4	3N-3	3N-2	3N-1	3N	3N+1	3N+2	5N-5	5N-4	5N-3	5N-2
Add.																			
A_0					W_1							R_1	W_0						R_0
A_1	W_0					R_0	W_1							R_1	W_0				
⋮																			
A_{N-2}			W_0					R_0	W_1							R_1	W_0		
A_{N-1}				W_0						R_0	W_1							R_1	

Figure 4. Test Procedure 1.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.