August 1986

UILU-ENG-86-2232
CSG-56

COORDINATED SCIENCE LABORATORY
College of Engineering

PARALLEL TESTING FOR PATTERN SENSITIVE FAULTS IN SEMICONDUCTOR RANDOM ACCESS MEMORY

Pinaki Mazumder
Janak H. Patel

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION NA	1b. RESTRICTIVE MARKINGS NA
2a. SECURITY CLASSIFICATION AUTHORITY NA	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE NA	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-86-2232 (CSG-56)	5. MONITORING ORGANIZATION REPORT NUMBER(S) NA
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Laboratory University of Illinois, Urbana 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Semiconductor Research Corporation
6c. ADDRESS (City, State and ZIP Code) 1101 W. Springfield Urbana, IL 61801	7b. ADDRESS (City, State and ZIP Code) P. O. Box 12053 Research Triangle Park, NC 27709
8a. NAME OF FUNDING/SPONSORING ORGANIZATION see 7a 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER SRC RSCH 84-06-049-4
8c. ADDRESS (City, State and ZIP Code) see 7b	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: N/A; PROJECT NO.: N/A; TASK NO.: N/A; WORK UNIT NO.: N/A
11. TITLE (Include Security Classification) Parallel Testing for Pattern Sensitive Faults in Semiconductor Random Access Memory	
12. PERSONAL AUTHOR(S) Pinaki Mazumder, Janak H. Patel	
13a. TYPE OF REPORT Technical 13b. TIME COVERED FROM ___ TO ___	14. DATE OF REPORT (Yr., Mo., Day) June, 1986 15. PAGE COUNT 26
16. SUPPLEMENTARY NOTATION NA	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) RAM, Passive Neighborhood Pattern Sensitive Fault, Active Neighborhood Pattern Sensitive Fault, Bit Lines, Word Lines, Sense Amplifiers

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

This technical report proposes a new test algorithm for pattern sensitive faults detection in large size RAM with high circuit density. The algorithm tests an n-bit RAM organized in $k \geq 1$ subarrays in $97.5\sqrt{\frac{n}{ke}}$ time where e is the eccentricity and is defined as the ratio of number of bit lines to number of word lines. The test algorithms detect both the active and passive neighborhoods pattern sensitive faults in RAM. The existing RAM architecture has been modified very little so that tthe proposed technique can be implemented very easily. The test algorithms are not generated within the chip and the additional hardware is minimal. If b is the number of bit lines in a subarray, then the additional overhead is $3bk+13k$ transistors (which is about $3\sqrt{n}$ transistors if $k=e=1$) including the error latches. This is the lowest overhead known to date in any RAM circuit with parallel testing capability. The low overhead allows high reliability and the additional circuit for each bit line can fit within the 3λ to 6λ pitch width in high density single transistor d-RAM. Eventhough the proposed test algorithm is designed to detect the pattern sensitive faults, the modified architecture can be readily used to speed up other conventional algorithms of complexity $O(n^k)$ to $O(n^{k/2})$.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code) 22c. OFFICE SYMBOL none

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

unclassified
SECURITY CLASSIFICATION OF THIS PAGE

Parallel Testing for Pattern Sensitive Faults in Semiconductor Random Access Memory

Pinaki Mazumder & Janak Patel
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801.
June, 1986

ABSTRACT

This technical report proposes a new test algorithm for pattern sensitive faults detection in large size RAM with high circuit density. The algorithm tests an n-bit RAM organized in $k \geq 1$ subarrays in $97.5\sqrt{\frac{n}{ke}}$ time where e is the eccentricity and is defined as the ratio of number of bit lines to number of word lines. The test algorithms detect both the active and passive neighborhoods pattern sensitive faults in RAM. The existing RAM architecture has been modified very little so that tthe proposed technique can be implemented very easily. The test algorithms are not generated within the chip and the additional hardware is minimal. If b is the number of bit lines in a subarray, then the additional overhead is $3bk+13k$ transistors (which is about $3\sqrt{n}$ transistors if $k=e=1$) including the error latches. This is the lowest overhead known to date in any RAM circuit with parallel testing capability. The low overhead allows high reliability and the additional circuit for each bit line can fit within the 3λ to 6λ pitch width in high density single transistor d-RAM. Eventhough the proposed test algorithm is designed to detect the pattern sensitive faults, the modified architecture can be readily used to speed up other conventional algorithms of complexity $O(n^k)$ to $O(n^{k/2})$.

Keywords: RAM, Passive Neighborhood Pattern Sensitive Fault, Active Neighborhood Pattern Sensitive Fault, Bit Lines, Word Lines, Sense Amplifiers

1. INTRODUCTION

Over the past one decade, random access memory (RAM) size is being quadrupled in about every two to four years. This has been made possible by scaling down the feature width by a factor of 6, increasing the overall chip dimension by six fold and finally reducing the individual cell area by a ratio of 90:1 (as shown in Table 1). Lewyn and Meindl [1] have enumerated the physical limits of VLSI dynamic RAMs and have shown that 64M bit dRAM (having an access time of 50 nsec) can be realized using 0.35μ technology and the conventional VLSI packaging. A further extension to 256M bit dRAM can be achieved by increasing the die size to 1.2 cm^2 and cooling the chip by liquid nitrogen. This enormous prospect of RAM design has posed a number of challenges to the RAM test engineers. Conventional test algorithms like marching test and GALPAT [2] can not economically test these large size RAM chips. GALPAT having a test complexity of $4n^2+4n$ for an n bit RAM will need about 800 days to test a 4M bit RAM chip with 100 nsec access time. Moreover, these tests have been designed to detect only stuck-at faults and restricted

Table 1: Cell and Array Geometric Parameters in RAM

Memory Size (bits)	Feature Width (μ)	Storage Area (cm^2)	Cell Area (μ^2)	Pitch Width (μ)	Year
4K	7	0.07	1764	6	1973
16K	5	0.1	800	6	1976
64K	3	0.15	216	5	1978
256K	2	0.3	96	5	1982
1M	1.3	0.2	20	4	1984
16M	0.5	0.3	1.5	3	?
64M	0.35	0.6	0.7	3	?

number of 2-coupling faults. But in large size memory, with high chip density the faults are largely due to the patterns of 0's and 1's stored in the neighboring cells. Hayes [3], Suk & Reddy [4], have proposed near optimal algorithms to detect pattern sensitive faults in the neighborhood of 5 cells. Eventhough these algorithms have linear complexity, they have an $O(k\,2^k)$ constant multiplier, where k is the size of neighborhood. Fuentes, et al. , [5] have proposed random testing as an alternate to deterministic algorithms for pattern sensitive fault testing and have analyzed the Markov chain model to find out that for 99.9% fault coverage test size in random testing is much larger than the deterministic algorithms in [3, 4]. To test a 4M bit RAM with 500 nsec access time, these algorithms will need over an hour. Eventhough this high overhead can be reduced by allowing many RAM chips are being tested in parallel and are being compared with a reference fault free chip [6], it is necessary to evolve novel techniques to test RAM chip such that many storage cells are tested in parallel. In conventional sequential testing techniques the Memory Address Register (MAR) and the Memory Data Register (MDR) form the bottleneck in speeding up the test algorithms. In this paper, a new parallel testing technique has been proposed and an $O(\sqrt{n})$ algorithm has been designed to detect the pattern sensitive faults. In the test mode, the decoder makes multiple selection of bit lines allowing to write the same data simultaneously at many storage locations on the same word line. In the read mode, a multi-bit comparator concurrently detects the error, if any. Conceptually, it has extended the system level parallel testing at the on chip level. But unlike in system level parallel testing, where a fault free reference chip is needed to locate the faulty chip, here a fault is detected by concurrent comparison to test whether all the input to the comparator are identical. The additional hardware is minimal and has been designed such that they can be fitted within the limits of 3λ to 6λ (where λ is the technological feature width) inter cell pitch width.

The problem of parallel testing has been addressed by other researchers. You & Hayes [7] partitioned the memory array in two or more subarrays and concurrently test them in $O(\sqrt{n})$ time. But their test only detects a limited types of pattern sensitive faults where a

write operation becomes faulty in presence of a few specific patterns in its adjacent cells. It does not detect the faults caused by transitions in the neighborhood. Moreover, it relies on comparing only two cells in memory arrays having two partitions. Thus if two cells are identically faulty, it fails to detect. As opposed to their technique, a multi-bit comparator has been used in this paper which concurrently detects simultaneous 0's and 1's. Thus reliability of fault detection is very high. You & Hayes reconfigure the memory subarray of size s bit into an s bit cyclic shift register where the data recirculates whenever a read operation is made. The reconfiguration has been done by introducing pass transistors on the bit lines which deteriorates both the sensitivity of the sense amplifiers by V_T (threshold voltage in MOS devices) and the access time of RAM in normal mode of operation. Also, since the test procedure is generated inside the chip, high extra overhead is involved in their design (including self testing decoder). To ensure the reliability of the test circuits inside the chip, extra pins are needed for the controllability and observability of the test circuit. Kinoshito & Saluja [8] analyzed the extra overhead involved in micro-coded ROM based built-in test generator for testing the pattern sensitive faults but their algorithms have linear time complexity. Sridhar[9] proposed an alternate scheme which does not require built-in test generator and used parallel signature analyzer (PSA) to access many bit lines concurrently. The PSA operates in three distinct mode: in the scan mode it is loaded with a specific pattern from outside in bit sequentially; in the write mode it writes its stored value to many bit lines in parallel; finally, in the signature mode it reads the content of the memory cells written earlier and generates an error quotient bit if there is any error. There are three potential limitations of this technique. At first, the scan mode reduces the parallelism in testing. Unless the test data generates a periodic signature which can be used to write subsequently, maximum speed up achievable in their technique is 3.5 for marching test of [2] having test complexity of $14n$. For complex test procedure necessary to test pattern sensitive faults, it may be necessary to scan in data several times. Secondly, a b-bit PSA can be implemented by approximately $15b$ transistors (requiring an area of $25\lambda \times 30\lambda$

for each bit) and does not fit within the allocated maximum limits of 6λ pitch width between bit lines. Thus, it becomes difficult to adjust the PSA on the periphery of the memory array. Finally, the PSA itself consists of dynamic shift registers which are highly vulnerable to soft errors. Thus the reliability of the testing circuit is very much questionable.

2. NOTATIONS AND FAULT MODEL

An $n \times 1$ word RAM, denoted by M_n, consists of p identical two dimensional submatrices each of size $b \times w$ $(= r$, say) such that $n = kbw$. A submatrix, M_r, consists of b bit lines and w word lines. Since in two layer VLSI technology either the bit line or the word line consists of diffusion/polysilicon wire which has quadratic signal propagation delay as opposed to the linear delay of the metal wire, it is mandatory to partition the M_n into p submatrices so that the access time does not deteriorate for large value of n. Also in order to drive long bit lines larger sense amplifiers are required to maintain their sensitivity. The partitioning has the intrinsic advantage since all the submatrices can be tested concurrently and thereby a testing speed up of p is easily achieved by incorporating built-in testing circuits. But p partitions are achieved at the expense of proliferating the sense amplifiers by a factor of $\sqrt{p}$ and also physically redistributing the decoder logic which introduces additional routing complexity and delay. For practical large size RAM having size 64K bit and more usually the value of p is chosen between 2 and 16. The ratio $e = b/w$ is called the *eccentricity* of M_r and is chosen such that the access time is minimized. $e \geqslant 1$ if the word line is made of metal and $e \leqslant 1$ if the bit line is made of metal.

Let $B = \{0,1,\cdots,b-1\}$ denote the set of bit lines in M_r and $W = \{0,1,\cdots,w-1\}$ denote its set of word lines. The ordered pair (i,j) described by the cartesian product $B \times W$ denotes the address of the storage cell which occurs at the crosspoint of i-th bit line and j-th word line. Such a cell is denoted by C_{ij}. The state of the cell C_{ij} is denoted by s_{ij}. An operation, ψ, on cell C_{ij} is denoted by $\psi(C_{ij})$ and the new state by s^{+}_{ij}. Valid

operations are to write a 1 or a 0 on a cell and to read the content of the cell. These operations are denoted by W_1,W_0 and R, respectively. Further to above notations, the operation of writing an either 0 or 1 (i.e., don't care, x) will be denoted by W_x and the operation of writing $y \in \{0, 1\}$ will be denoted by W_y. Depending on the previous state of the cell $s_{ij} \in \{0,1\}$, the effect of application of $\psi \in \{W_1, W_0, R\}$ on C_{ij} can be further classified by the ordered pair in cartesian product $s_{ij} \times \psi$.

(1) Ordered pair $(0,W)$ is to write 1 erasing a previous 0 and is denoted by ↑.
(2) Ordered pair $(0,W_0)$ is to write 0 over a previous 0 and is called *non-transition write 0*.
(3) Ordered pair $(1,W_1)$ is to write 1 over a previous 1 and is called *non-transition write 1*.
(4) Ordered pair $(1,W_0)$ is to write 0 over a previous 1 and is denoted by ↓.
(5) Ordered pairs $(0,R)$ and $(1,R)$ denote the operations of reading a cell whose previous contents are 0 and 1 respectively.

In memories with non-destructive read, the operations $(0,R)$ and $(1,R)$ are such that $s^+_{ij} = s_{ij}$. But in memories with destructive read, s^+_{ij} may be different from s_{ij}.

2.1. Memory Cell Faults:

In this paper, all read operations will be assumed to be fault free and thereby applicable to memories with non-destructive read. The test algorithms discussed in this paper has $O(\sqrt{n/(pe)})$ complexity and can detect the restricted pattern sensitive fault (PSF) [10]. An unrestricted PSF has been shown to have [11] $(3n^2+2n)2^n$ test complexity and is impractical in large size RAM. In restricted PSF, an operation $\psi(C_{ij})$ is faulty due to the presence of a specific pattern in a set of physically adjacent cells (called its neighbors) or an operation on its neighbor changes s_{ij}. Figure 1 shows the physically adjacent cells of C_{ij} and their neighbors are defined as follows.

Definition 1 A base cell C_{ij} with its four rookwise adjacent neighboring cells $C_{i\pm1,j}$ and $C_{i,j\pm1}$ are called **von Neumann neighborhood** (or type 1 neighborhood) of C_{ij}. If the base cell is deleted, the set of remaining cells in the neighborhood are denoted by N_1. Cells

$C_{i\pm1,j}$ that share the same word line with the base cell are denoted by N_w, while the cells $C_{i,j\pm1}$ that share the same bit line with the base cell are denoted by N_b.

Definition 2 A base cell C_{ij} with its eight adjacent neighboring cells $C_{i\pm1,j\pm1}$ are called **Moore neighborhood** (or type 2 neighborhood) of C_{ij} If the type 1 neighborhood is deleted from the type 2 neighborhood, the set of remaining cells in the neighborhood are denoted by N_2.

Definition 3 A pattern sensitive fault is called static (SPSF) if an operation $\psi(C_{ij})$ is faulty in presence of a fixed pattern in $N_1 \bigcup N_2$. This fault is denoted by $\psi(C_{ij})/s(N_b)s(N_w)s(N_2)$, where $s(N_b)$, $s(N_w)$ and $s(N_2)$ represent the states of all cells in N_b, N_w and N_2, respectively.

Definition 4 A pattern sensitive fault is called dynamic (DPSF) if s_{ij} changes because of an operation $\psi(C_{ij})$ on one or more cells in $N_1 \bigcup N_2$. A DPSF is denoted by $s_{ij}/\psi(C_{i\pm1,j}),s(N_w),s(N_2)$, $s_{ij}/s(N_b),\psi(C_{i,j\pm1}),s(N_2)$, or $s_{ij}/s(N_b),s(N_w),\psi(N_2)$, depending on whether the operation made on bit lines of N_1, word lines of N_1 or N_2

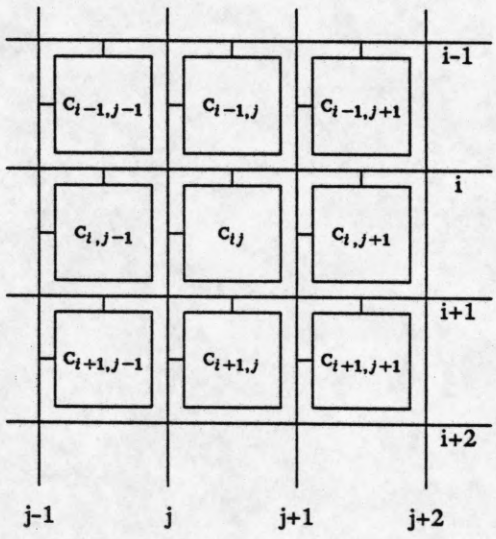


Figure 1: Base Cell, C_{ij} and its neighborhoods

changes s_{ij}.

Let the cells in the neighborhood be assigned k distinct positive number in the set $\{0,1,\cdots k-1\}$ such that the number i is assigned to the base cell. The content of the neighborhood can be denoted by a a state vector $<s_0 s_1 \cdots s_i \cdots s_{k-1}>$ where s_j is the state of the cell which has been assigned the number $j \in \{0,1,\cdots,k-1\}$. Clearly, the state vector space of a neighborhood of size k describes a k dimensional boolean hypercube in which the nodes represent all different possible patterns which can be stored in the neighborhood. A node in the state vector space graph is numbered m, where $m = \sum_{j=0}^{k-1} s_j 2^j$ and evidently $0 \leqslant m \leqslant 2^k - 1$. An edge corresponds to the state transition due to a write operation, W_x, on a cell in the neighborhood. Let $\psi(j)$ denote an operation of $\psi \in \{W_1, W_0, W_x, R\}$ on the cell numbered j. Thus $W_x(i)$ changes the state vector from $<s_0 s_1 \cdots s_i \cdots s_{k-1}>$ to $<s_0 s_1 \cdots \overline{s_i} \cdots s_{k-1}>$. All such $W_x(i)$ operations on 2^k nodes sensitize the SPSF and hence a test length of 2^k is required to detect SPSF in a neighborhood of k cells. All other operations $W_x(j \neq i)$ sensitize the DPSF and the total test length is equal to $(k-1)2^k$ (note that each undirected edge in hypercube corresponds to two transitions). In Figure 2, a three dimensional boolean hypercube represents the state vector space of a neighborhood of size $k=3$. Assuming that $i=2$ is the base cell, all the diagonal edges between the outer and inner

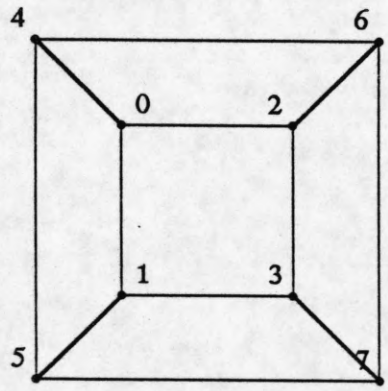


Figure 2: State space graph for neighborhood with $k=3$.

squares sensitize the SPSF while the other edges sensitize the DPSF. It is interesting to note that the operation $W_x(i)$ corresponds to C_i routing in k-cube SIMD network [12]. A path on state vector space graph from any arbitrary state to another state represents a write sequence on memory cells in the neighborhood. The number of transitions in a write sequence indicates the path length. If the path terminates at the originating state, it is called a cycle and its diameter is equal to half of the cycle length.

Definition 5 A path is called a k-hamiltonian cycle if the path describes a cycle having diameter k and it does not visit any state (except the originating state) twice.

It may be observed that all cells stuck-at-faults are detected by SPSF. Stuck-at-faults in memory cells usually occur because of oxide pin holes in storage transistors and thus cause irreversible permanent faults. If the operation $W_1(i)R(i)$ is done for all possible 2^{k-1} patterns in $k-1$ adjacent cells, it will sensitize the base cell stuck-at-0 fault, denoted by $C(i)/0$. Similarly, the remaining 2^{k-1} $W_0(i)R(i)$ operations in SPSF will sensitize the base cell stuck-at-1 fault, denoted by $C(i)/1$. In RAM design where the memory cell consists of access transistors, the pin hole defects on access transistor gate oxides may cause cell stuck-at-word-line ($C(i)/L_w$), cell stuck-at-bit-line ($C(i)/L_b$) and bit-word-line-crosstalk (L_w/L_b). In switched capacitance single transistor dynamic RAM memory cell design which does not use the access transistor, these faults are largely due to leakage current between the cell, the bulk substrate, the word line and the bit line. Chatterjee, et al. [13], have made intensive study of failure modes in RAM due to leakage current and noted that these leakage current vary widely depending on the patterns stored in adjacent cells. Like the stuck-at-faults, the SPSF sensitizes all these faults based on device modeling. It may also be noted that each $W_x(i)R(i)$ operation automatically sensitizes the transition faults in base cell.

Operations $W_x(j)R(i)$ for all $j \neq i$ check the 2-coupling faults between the base cell and other cells in its neighborhoods. Two cells i and j are called 2-coupled if a

transitional write operation $\psi_1(j)$ simultaneously coerce a transitional write operation $\psi_2(i)$ and the fault is denoted by $\psi_1(j)=>\psi_2(i)$. Evidently ψ_1 and ψ_2 can be $\uparrow$ and $\downarrow$. If $\psi_1=\psi_2$, then the two cells are said to be symmetrically coupled. If either the fault $\uparrow(j)=>\uparrow(i)$ or the fault $\downarrow(j)=>\downarrow(i)$ occur but not the both, then the two cells are said to be 1-way symmetrically coupled and it is denoted by $\uparrow(j)=>\uparrow(i)$ || $\downarrow(j)=>\downarrow(i)$. If both the faults occur, then i and j are said to be 2-way symmetrically coupled and it is denoted by $\uparrow(j)=>\uparrow(i)$ && $\downarrow(j)=>\downarrow(i)$. On the contrary, if ψ_1 and ψ_2 are not necessarily identical, then the cells are said to be asymmetrically coupled. 1-way asymmetric coupling is denoted by $\uparrow(j)=>\uparrow(i)$ || $\uparrow(j)=>\downarrow(i)$ || $\downarrow(j)=>\uparrow(i)$ || $\downarrow(j)=>\downarrow(i)$. 2-way asymmetric coupling is denoted by $(\uparrow(j)=>\uparrow(i)$ || $\uparrow(j)=>\downarrow(i))$ && $(\downarrow(j)=>\uparrow(i)$ || $\downarrow(j)=>\downarrow(i))$. It may be noted that all the $W_x(j)R(i)$ which sensitize the DPSF also sensitize the above four types of 2-coupling faults between the base cell and its neighboring cells.

In this paper, test algorithms will be discussed which sensitize the SPSFs and DPSFs between each cell in the memory and its **Moore Neighborhood.** Since 4608 transition writes are required to sensitize all the SPSFs and DPSFs between the base cell and its neighbors, certain constraints are invoked here. These constraints are on the basis of practical observations and allow to collapse the number of transition writes to 64 to test the SPSFs and DPSFs. Under the constraints, physically adjacent nine cells in Moore neighborhood constitute logically adjacent 5 cells. It will be assumed that the cells in N_b, N_w and N_2 are mutually consistent in the sense that if an operation $\uparrow$ or $\downarrow$ is applied to one cell or simultaneously to more than one cell belonging to a neighborhood N (where N can be N_b, N_w or N_2) then the next write operation in the test will be to apply the previous transition write to the remaining cells in N and the state transition in the base cell occurs only once. The consistency criterion changes the definition of DPSF in fault model slightly and is stated below.

Definition 5 If the base cell, i is in state $s_i \in \{0, 1\}$ and a transition write ψ is made over one or two cells simultaneously belonging to N (where N can be N_b, N_w or N_2) then a DPSF is said to be consistent if the state of cell i does not change back from $\overline{s_i}$ to s_i after the successive ψ operation on the remaining cells in N.

Under the consistency criterion, the state of a neighborhood, $s(N)$ will be denoted by 1 or 0 if all the cells in the neighborhood have state 1 or 0, respectively. A transition write $\psi(N)$ will be denoted by $\uparrow$ or $\downarrow$ if the transition write ψ is made over all the cells in N before another transition write is made on any other cell not included in N. All the SPSFs and DPSFs under these new notations are shown in Table 2.

In addition to the faults in memory cell array, faults may occur in bit line and word line decoders, memory address buffers and read/write logic in RAM. Before the test algorithms are defined, these faults are needed to be modeled and the test algorithm is designed to cover these faults.

Table 2: All possible SPSFs and DPSFs

fault type	fault notation
SPSF	↑/000, ↑/100, ↑/010, ↑/110, ↑/001, ↑/101, ↑/011, ↑/111. ↓/000, ↓/100, ↓/010, ↓/110, ↓/001, ↓/101, ↓/011, ↓/111.
DPSF	0/↑00, 0/↑10, 0/↑01, 0/↑11, 1/↑00, 1/↑10, 1/↑01, 1/↑11, 0/↓00, 0/↓10, 0/↓01, 0/↓11, 1/↓00, 1/↓10, 1/↓01, 1/↓11, 0/0↑0, 0/1↑0, 0/0↑1, 0/1↑1, 1/0↑0, 1/1↑0, 1/0↑1, 1/1↑1, 0/0↓0, 0/1↓0, 0/0↓1, 0/1↓1, 1/0↓0, 1/1↓0, 1/0↓1, 1/1↓1, 0/00↑, 0/10↑, 0/01↑, 0/11↑, 1/00↑, 1/10↑, 1/01↑, 1/11↑, 0/00↓, 0/10↓, 0/01↓, 0/11↓, 1/00↓, 1/10↓, 1/01↓, 1/11↓.

2.2. Decoder Faults:

In RAM decoders are divided into bit line decoders and word line decoders. In this section the faults of bit line decoders are discussed and they are also applicable to word line decoders. A storage cell in an r bit subarray of a RAM is addressed by $\log_2 b + \log_2 w$ bit. $\log_2 b$ bit address to bit line decoder describes an input address space A of cardinality b. A decoder, D maps an element $\alpha \in A$ onto a bit line $j \in B$ such that $D(\alpha)=j$. A decoder is fault free if D is a bijective mapping. A decoder is said to be faulty if D is not a bijective mapping. A mapping, D^*_1 is said to be type 1 faulty if there exists an $\alpha \in A$ such that $D^*_1(\alpha) \notin B$. A mapping D^*_2 is said to be type 2 faulty if there exist $\alpha \in A$ and $\beta \in A$ such that for all $j \in C$ and $D^*_2(\alpha)=j$, where $B \subseteq C$ and there exists at least one element $k \in C$ so that $D^*_2(\alpha)=D^*_2(\beta)=k$. A mapping D^*_3 is said to be type 3 faulty if there exist $\alpha \in A$ and $\beta \in C$, where $C \supset A$ such that for all $\beta \in C$, $D^*_3(\beta)=D^*_3(\alpha)$.

All possible decoder faults like line open, line shorted, bridging, crosspoint stuck at 0 and 1, multiple access, etc., are covered by the composite mapping $D^*_1 \circ D^*_2 \circ D^*_3$.

It may be noted that the stuck-at-faults in memory buffers will result into one of the three types of fault and need not be considered separately. Stuck-at-faults in read/write logic will be modeled as SPSF in memory cells. Capacitive coupling effects in input/output lines will be detected by DPSFs.

3. TEST METHODS AND TEST ALGORITHMS:

The basic strategy adopted for parallel testing can be conceived as fault syndrome based. From the previous fault model, it may be noted that in the presence of a multiple access (type 2) fault in decoder, a write operation results in parallel writing the content of data-in buffer to all the locations selected by the decoder. A read operation results in storing either the boolean OR or logical AND of the contents in multiple accessed cells into the data-out buffer. In this paper, it will be assumed that it is unknown whether the content of data-out buffer results due to OR or AND function. The strategy adopted in this paper

is to divide the b bit lines in a memory subarray into g groups such that the bit line i belongs to group j, where $j = i \ (mod \ g)$. In the test mode, all the bit lines in a group j are selected in parallel to execute a read or write operation. A write operation allows to write simultaneously on $q = \lfloor b/g \rfloor$ cells as selected by word line. If due to SPSF, a write operation on a cell fails, then the error is detected in read operation when all these cells are read in parallel. It is not possible to determine the error from the content of data-out buffer. Hence a parallel comparator of q input is included to concurrently detect whether all the q bits are identical. If any discrepancy occurs, the comparator triggers the error latch to indicate the occurrence of a fault. The organization of RAM with modified decoder and comparator is shown in Figure 3.

In order to detect all the SPSFs and DPSFs discussed in the fault model, the value of g is chosen to be 2. Thus the decoder allows parallel access of all even bit lines at a time and all odd bit lines at a time. The word line is accessed sequentially and to detect the pattern sensitive faults, they are accessed in interlace scanning mode. The interlace scanning can select the even and odd word lines in either ascending or descending order. Thus the testing technique can be defined as cross product of concurrent interlaced bit line and sequential interlaced word line scanning. The test algorithm is given in Algorithm 1. The RAM cells are divided into four classes of cells. The set of bit lines $i \in B$ are divided into B_{even} and B_{odd} depending on whether i is even or odd, respectively and the set of word lines $j \in W$ are divided into W_{even} and W_{odd} depending on whether j is even or odd, respectively. All cells whose addresses are given by the cartesian product $B_{odd} \times W_{odd}$ go through a transitional write when the procedure ProcA is called. Similarly, all cells whose addresses are given by $B_{odd} \times W_{even}$, $B_{even} \times W_{even}$ and $B_{even} \times W_{odd}$ go through a transitional write when procedures ProcB, ProcC and ProcD are called, respectively. Procedures described here are similar to marching algorithms. Unlike the marching elements of earlier marching test algorithms [14] where the marching elements move in ascending or descending sequence of

memory address, here a set of operations, called a *marching macro element*, move in ascending sequence of W_{even} or W_{odd} word lines number. If $\psi_1, \psi_2, \ldots, \psi_t$ are a set of operations applied successively to a memory cell, C_{ij}, and following to those operations ψ_{t+1} is applied to memory cells, $C_{i+x_1,j+y_1}, \ldots, C_{i+x_k,j+y_k}$, then the marching macro element is denoted by $\psi_1, \cdots \psi_t(C_{ij})\psi_{t+1}(C_{i+x_1,j+y_1}) \cdots \psi_{t+1}(C_{i+x_k,j+y_k})$. The use of marching macro element facilitates hardware generation of the algorithm. Thus the test procedure described here is well suited for built-in self testable (BIST) random access memory design.

Theorem 1: *Algorithm 1 detects all Static Pattern Sensitive Faults and all Dynamic Pattern Sensitive Faults in a memory array.*

Proof Sketch: Each "for" loop in lines 2, 3, 4, 5, 7, 8 , 9 and 10 describes a 4-hamiltonian cycle for every cells in the memory. The number of nodes in the cycle is 8 and the number of directed edges in the cycle are also 8. Thus each memory cell makes eight 4-hamiltonian tour such that in no tour the same directed edge is traversed twice. Since 8 tours on 4 dimensional hypercube cover 64 transition writes, all the SPSFs and DPSFs are covered if after each transition write all the cells in the neighborhood are read. For details of tour, see Table 3 and Figure 4. The reading operation describes a tessellation by tetrominoes. □

Theorem 2: *Algorithm 1 has* $97.5\sqrt{n/(pe)}$ *test complexity, where* n *is the size of RAM in bits,* p *is the number of partitions in RAM and* e *is the eccentricity of each partition.*

Proof: Each "for" loop (in Algorithm 1) which describes 4-hamiltonian cycle needs $2\times6w = 12w$ operations. Since there are 8 4-hamiltonian cycles, total number of operations is $96w$. Line 1 and 6 are used for initialization of memory and need $1.5w$ operations. Since there are p partitions, $n = pbw = pew^2$ and therefore, $w = \sqrt{n/(pe)}$. □

The algorithm is near optimal and needs $0.5w$ extra write operations. By describing an eulerian tour over the hypercube optimal algorithm can be constructed. In order to gen-

Algorithm 1: Parallel Pattern Sensitive Fault Test

Notation: $B_{even} = \{i \mid (i \in B) \& (i \,(mod\ 2)=0)\}$
$B_{odd} = \{i \mid (i \in B) \& (i \,(mod\ 2)=1)\}$
$B = B_{even} \bigcup B_{odd}$
$i' = (i+1)\ (mod\ b)$
$j' = (j-1)\ (mod\ w)$

```
{
  /* begin */
(1)   for (j=0;j<w;j++)
        pardo ( for all C_ij  ∈ B×j )
          W0(C_ij );
(2)   for (x=1;x≥0;x—) {
        ProcA(x); ProcC(x); ProcD(x); ProcB(x);}
(3)   for (x=1;x≥0;x—) {
        ProcB(x); ProcD(x); ProcC(x); ProcA(x);}
(4)   for (x=1;x≥0;x—) {
        ProcC(x); ProcA(x); ProcB(x); ProcD(x);}
(5)   for (x=1;x≥0;x—) {
        ProcD(x); ProcB(x); ProcA(x); ProcC(x);}

(6)   for (j=0;j<w;j++)
        pardo ( for all C_ij  ∈ B_even×j )
          W1(C_ij );
(7)   for (x=1;x≥0;x—) {
        ProcC(1-x); ProcA(x); ProcB(x); ProcD(1-x);}
(8)   for (x=1;x≥0;x—) {
        ProcD(1-x); ProcB(x); ProcA(x); ProcC(1-x);}
(9)   for (x=1;x≥0;x—) {
        ProcA(x); ProcC(1-x); ProcD(1-x); ProcB(x);}
(10)  for (x=1;x≥0;x—) {
        ProcB(x); ProcD(1-x); ProcC(1-x); ProcA(x);}
  /* end */
  }

(11) ProcA(y)
  {     for (j=0;j<w;j++)
          if ( j (mod 2) = 1 )
            pardo ( for all C_ij  ∈ B_odd×j )
              RW_y R(C_ij )R(C_ij·)R(C_i'j )R(C_i'j·);
  }
```

```
(12) ProcB(y)
     {       for (j=0;j <w ;j ++)
               if ( j(mod 2) = 0 )
                  pardo ( for all C_ij  ∈ B_odd × j )
                    RW_y R(C_ij )R(C_ij·)R(C_i'j )R(C_i'j·);
     }
(13) ProcC(y)
     {       for (j=0;j <w ;j ++)
               if ( j(mod 2) = 0 )
                  pardo ( for all C_ij  ∈ B_even × j )
                    RW_y R(C_ij )R(C_ij·)R(C_i'j )R(C_i'j·);
     }
(14) ProcD(y)
     {       for (j=0;j <w ;j ++)
               if ( j(mod 2) = 1 )
                  pardo ( for all C_ij  ∈ B_even × j )
                    RW_y R(C_ij )R(C_ij·)R(C_i'j )R(C_i'j·);
     }
```

Table 3: Transition Write Sequence in Algorithm 1

"for" Loop in	Cell Type	Hamiltonian Cycle
line #2	$C_{ij} \in B_{odd} \times W_{odd}$	<0,1,9,13,15,14,6,2,0>
	$C_{ij} \in B_{odd} \times W_{even}$	<0,2,6,14,15,13,9,1,0>
	$C_{ij} \in B_{even} \times W_{even}$	<0,8,9,11,15,7,6,4,0>
	$C_{ij} \in B_{even} \times W_{odd}$	<0,4,6,7,15,11,9,8,0>
line #3	$C_{ij} \in B_{odd} \times W_{odd}$	<0,2,6,14,15,13,9,1,0>
	$C_{ij} \in B_{odd} \times W_{even}$	<0,1,9,13,15,14,6,2,0>
	$C_{ij} \in B_{even} \times W_{even}$	<0,4,6,7,15,11,9,8,0>
	$C_{ij} \in B_{even} \times W_{odd}$	<0,8,9,11,15,7,6,4,0>
line #4	$C_{ij} \in B_{even} \times W_{odd}$	<0,8,9,11,15,7,6,4,0>
	$C_{ij} \in B_{odd} \times W_{even}$	<0,4,6,7,15,11,9,8,0>
	$C_{ij} \in B_{even} \times W_{even}$	<0,1,9,13,15,14,6,2,0>
	$C_{ij} \in B_{even} \times W_{odd}$	<0,2,6,14,15,13,9,1,0>
line #5	$C_{ij} \in B_{odd} \times W_{odd}$	<0,4,6,7,15,11,9,8,0>
	$C_{ij} \in B_{odd} \times W_{even}$	<0,8,9,11,15,7,6,4,0>
	$C_{ij} \in B_{even} \times W_{even}$	<0,2,6,14,15,13,9,1,0>
	$C_{ij} \in B_{even} \times W_{odd}$	<0,1,9,13,15,14,6,2,0>
line #7	$C_{ij} \in B_{odd} \times W_{odd}$	<12,4,5,7,3,11,10,8,12>
	$C_{ij} \in B_{odd} \times W_{even}$	<12,8,10,11,3,7,5,4,12>
	$C_{ij} \in B_{even} \times W_{even}$	<3,2,10,14,12,13,5,1,3>
	$C_{ij} \in B_{even} \times W_{odd}$	<3,1,5,13,12,14,10,2,3>
line #8	$C_{ij} \in B_{odd} \times W_{odd}$	<12,8,10,11,3,7,5,4,12>
	$C_{ij} \in B_{odd} \times W_{even}$	<12,4,5,7,3,11,10,8,12>
	$C_{ij} \in B_{even} \times W_{even}$	<3,1,5,13,12,14,10,2,3>
	$C_{ij} \in B_{even} \times W_{odd}$	<3,2,10,14,12,13,5,1,3>
line #9	$C_{ij} \in B_{odd} \times W_{odd}$	<12,13,5,1,3,2,10,14,12>
	$C_{ij} \in B_{odd} \times W_{even}$	<12,14,10,2,3,1,5,13,12>
	$C_{ij} \in B_{even} \times W_{even}$	<3,11,10,8,12,4,5,7,3>
	$C_{ij} \in B_{even} \times W_{odd}$	<3,7,5,4,12,8,10,11,3>
line #10	$C_{ij} \in B_{odd} \times W_{odd}$	<12,14,10,2,3,1,5,13,12>
	$C_{ij} \in B_{odd} \times W_{even}$	<12,13,5,1,3,2,10,14,12>
	$C_{ij} \in B_{even} \times W_{even}$	<3,7,5,4,12,8,10,11,3>
	$C_{ij} \in B_{even} \times W_{odd}$	<3,11,10,8,12,4,5,7,3>

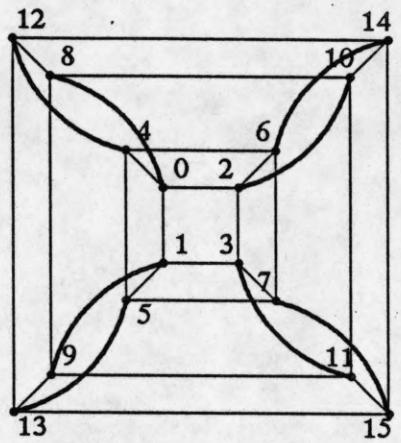


Figure 4: State space graph for neighborhood with k =4.

erate the eulerian tour, it is necessary to construct 4 bit reflected grey code which has a hamming distance of 1. Write sequence can be derived from these codes to describe eulerian path. Thus it can be seen that the eulerian tour is far more difficult to generate by hardware for embedded applications. On the contrary, algorithm 1 described here is very simple to generate by using a $\log_2 w$ bit counter.

After the parallel testing is completed, the bit lines are tested for multiple access faults in decoder by Algorithm 2.

Theorem 3: *Algorithm 1 and 2 detect all decoder faults described by composite mapping D^*.*

Proof: Algorithm 1 detects all the word line decoder type 1, 2 and 3 faults. By assumption the output of data out buffer is either the logical AND or OR of the content of the storage cells at the cross point of the selected word line and the bit lines accessed in parallel. Thus, if no fault occurs, the data output will be the same irrespective whether it is a logical OR or a logical AND implementation. Thus by reading the data output, the content of the cells can be read without ambiguity if there is no fault. Thus type 1 word line decoder fault can be easily detected by reading the data output. Type 2 and type 3 faults in word line decoder can be easily detected by the parallel comparator while executing Algorithm 1. Algorithm 1

Algorithm 2: Multiple Access in Decoder

/* assume the memory is initialized to zero */

/* multiple access where decoder selects the correct bit line i along with one or more bit lines $j > i$ */

```
j = 0;
y = 1;
for ( i=0; i <b ; i++)
    RW_y (C_ij );
```

/* multiple access where decoder selects the correct bit line i along with one or more bit lines $j < i$ */

```
y = ȳ;
for ( i=b−1; i ≥0; i−−)
    RW_y (C_ij );
```

---

also detects type 2 and 3 faults in bit line decoder except multiple accesses between odd bit lines and even bit lines. Algorithm 2 detects all bit line decoder type 1 and multiple accesses faults. □

4. IMPLEMENTATION:

4.1. Modified Decoder Circuit:

The technique described in this paper can be used for any fanout non-reconvergent decoder topology by simply adding a two input OR gate at each bit line decoder output and connecting it with one of the input of OR gate. In normal mode of operation, the other input of OR gate is grounded and the decoder output is passed by the OR gate. In testing mode, all decoder outputs are grounded and the bit lines are selected by the other input of OR gates. To illustrate the technique, it is assumed that the bit line decoders are made of NOR plane of two level PLA logic. The bit line decoder has been modified as shown in Figure 5.

Lines L1 and L2 are added to select the odd and even lines in parallel. Transistors Q_0 through Q_{b-1} are used to select the bit lines by L_1 and L_2. For normal operation mode TEST=0 and both L_1=L_2=1. Thus the bit lines are selected by the decoder depending on the content of address lines $(a_0, a_1, \cdots, a_k)$ where $k = \log_2 b - 1$. In test mode, all the address input is set to zero (i.e., a_i=0 for all $0 \leqslant i \leqslant k$.) Transistor Q_b pulls down the lower most product line to zero while all other product lines are pulled down to zero by one or more cross-point decoding transistors. If the line L_1=0 and L_2=1, all the bit lines connected to line L_1 (say, the set of bit lines B_{even}) will be selected in parallel. If the line L_1=1 and L_2=0, all the odd bit lines in set B_{odd} connected to line L_2 will be selected in parallel.

4.2. Parallel Comparator and Error Detector:

The parallel comparator and error detector senses the output of sense amplifiers connected to bit lines which are selected in parallel. It detects the concurrent occurrence of either 0's or 1's. In case, any selected bit line is different from other it triggers the error latch indicating the occurrence of a fault. Figure 6 shows the parallel comparator and error detector. Transistors $Q_0, \cdots, Q_{b-1}$ are connected to the sense amplifiers and they are selected by L_2. If L_2=0, all the p-type pass transistors conduct and connect the set of bit lines in B_{odd} to the parallel comparator while B_{even} lines are disconnected by the n-type transistors. If L_2=1 all the n-type transistors connect set of bit lines in B_{even} to the parallel comparator. Transistors $T_1, \cdots, T_{l-1}$ are connected in series and detect concurrent occurrence of 1 in bit lines set. Transistors $P_1, \cdots, P_{l-1}$ are connected in parallel and detect concurrent occurrence of 0 in bit lines set. Transistors T_0 and P_0 are the precharge transistors which are turned on by precharge clock phase $\phi 1$. Transistors T_l and P_l are the discharge transistors which remain cut off during precharge phase and turns on during discharge clock phase $\phi 2$. Transistors S_0, S_1 and S_2 form a coincidence detector. If all the selected bit lines are 0 or 1, then both S_1 and S_2 remain in cut off and the output of the detector is 1. The output of the detector is connected to the error latch consisting of transis-

tors $V_0, \cdots, V_3$. The error latch output is ERROR=0, when the selected bit lines are identical. If the bit lines are not identical, then S_1 conducts and S_2 remains cutoff and the detector output is 0. This triggers the error latch setting latch output to ERROR=1. During write phase and normal mode of operation, the error latch is clamped to zero by V_4. The error detector is inhibited by the discharge transistors Q_l and T_l during the start of read phase when the sense amplifiers output are not identical because of sluggish changes in some of the sense amplifiers.

5. CONCLUSION

This paper has discussed an efficient technique to speed up the RAM test algorithms. Specifically, it has proposed a novel test algorithm to test the pattern sensitive faults in RAM. In Table 4 the performance of the proposed algorithm is compared with the test algorithms of [3] and [4] for different size of RAMs. All these test algorithms detect the static and dynamic pattern sensitive faults, eventhough there are some subtle differences in their faults model. The three noticeable aspects of the comparison are that - i) the proposed algorithm is more than thousand times faster than the other two algorithms, ii) as the RAM size increases by a factor of 64, the test complexity in proposed algorithm increases by a factor of 4 (unlike the other two algorithms in Table 4 for which test complexity increases by a factor of 64), and iii) the proposed algorithm utilizes the memory partitions to speed up the test eventhough its implementation and reliability are independent of RAM organization. Another key aspect of the implementation is that it can be used to speed up any existing algorithm. Hayes algorithm on pattern sensitive faults can be speeded up by a factor of $0.2\sqrt{pn}$ for an n bit RAM. Similarly, Suk and Reddy's PSF algorithms can be speeded up by a factor of $0.125\sqrt{pn}$. But the proposed algorithm is much faster than these modified speeded up algorithms.

The implementation scheme uses minimal extra hardware. The parallel comparator consists of $2b+12$ transistors and the extra hardware in modified decoder is $b+1$

Table 4: Comparison Of Different PSF Test Algorithms

RAM Size	256k	1M	4M	16M
# Of Partitions (p)	4	8	8	16
Hayes Test $(3k+2)2^k n$	28.5 sec	114.1 sec	456.3 sec	1825.3 sec
Suk & Reddy's Test $(k+5)2^{k-1}n$	8.4 sec	33.6 sec	134.2 sec	536.9 sec
Proposed Test $(97.5\sqrt{n}/(pe)$	4.6 msec	6.4 msec	12.8 msec	18.2 msec

Assumptions: memory cycle time = 200 nsec, e = 1.2

transistors. Thus the overall extra hardware is only $3b+13$ transistors and the overall chip area expansion is only 0.4% for 256k RAM. In contrast to that the parallel signature analyzer needs about 2.5% of extra chip area for 256k RAM. Moreover, parallel signature analyzer can not be laid out within the maximum interceller pitch width of 6λ, where λ is the technological feature width. The proposed technique needs only one transistor to fit within the pitch width and easily fits even for the future type of vertically integrated single cell RAM design having intercell pitch width of 3λ.

Bibliography

[1] L. L. Lewyn and J. D. Meindl, "Physical Limits of VLSI dRAM's," *IEEE Journal of Solid-State Circuits*, vol. SC-20, pp. 231-241, February 1985.

[2] M. A. Breuer and A. D. Friedman, *Diagnosis and Reliable Design of Digital Systems*. CA: Woodland Hills, 1976.

[3] J. P. Hayes, "Testing Memories for Single-Cell Pattern-Sensitive Faults," *IEEE Transaction on Computers*, vol. C-29, pp. 249-254, March 1980.

[4] D. S. Suk and S. M. Reddy, "Test Procedures for a Class of Pattern Sensitive Faults in Semiconductor Random Access Memories," *IEEE Transaction on Computers*, vol. C-29, pp. 419-429, June 1980.

[5] A. Fuentes, R. David and B. Courtois, "Random Testing versus Deterministic Testing of RAMs," *Proceedings of FTCS*, July 1986.

[6] M. A. Rich and D. E. Gentry, "The Economics of Parallel Testing," *Proc. 1983 International Test Conference*, pp. 656-661, 1983.

[7] Y. You and J. P. Hayes, "A Self-Testing Dynamic RAM Chip," *IEEE Journal of Solid-State Circuits*, vol. SC-20, pp. 428-435, February 1985.

[8] K. Kinoshita and K. K. Saluja, "Built-in Testing of Memory using On-chip Compact Testing Scheme," *Proc. 1984 International Test Conference*, pp. 271-281, 1984.

[9] T. Sridhar, "A New Parallel Test Approach for Large Memories," *Proc. 1985 International Test Conference*, pp. 462-470, 1985.

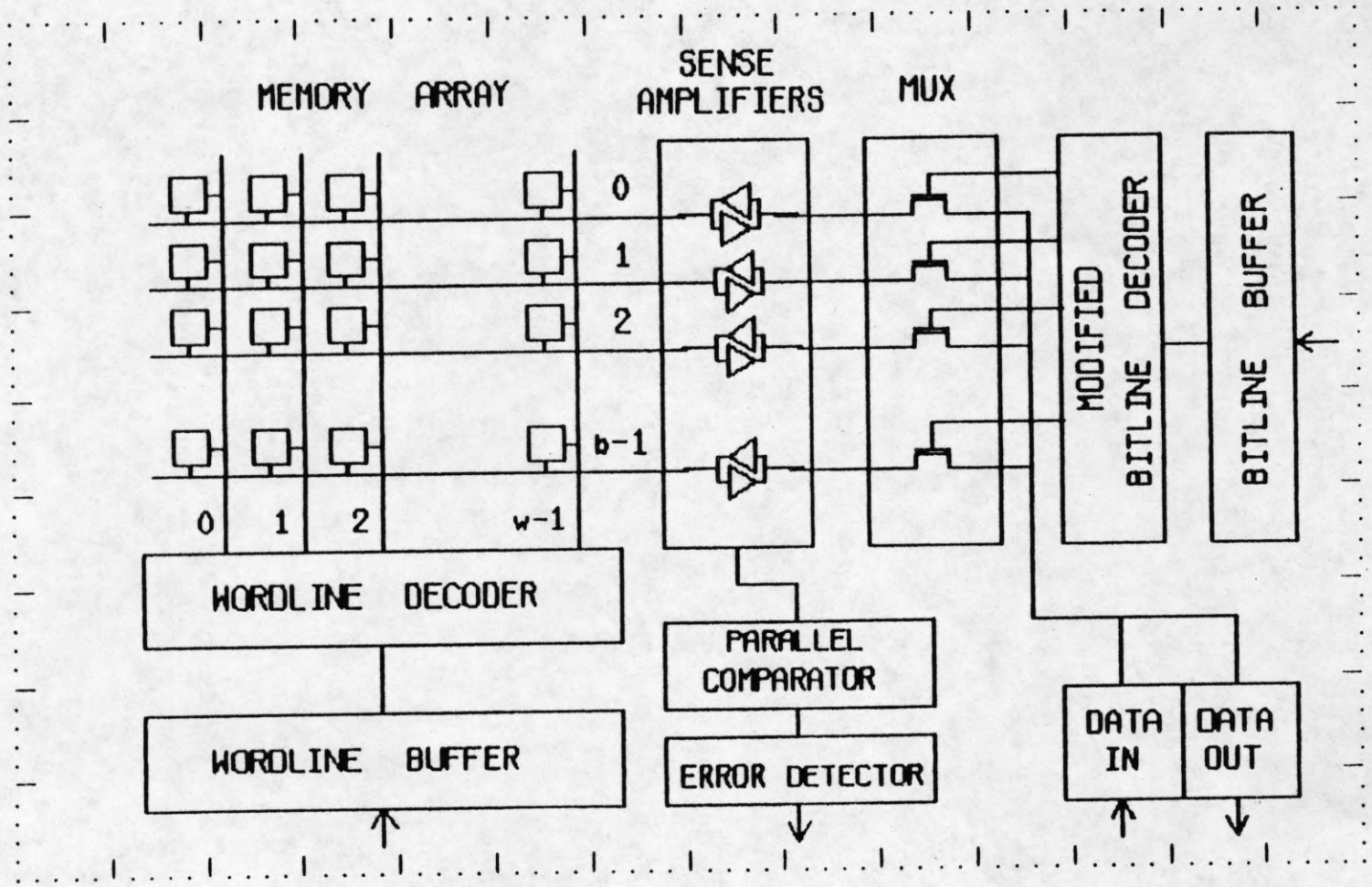
[10] J. R. Brown, "Pattern Sensitivity in MOS Memories," *Dig. Symp. Testing to Integrated Semiconductor Memories into Computer Mainframes*, pp. 33-46, October 1972.

[11] J. P. Hayes, "Detection of Pattern-Sensitive Faults in Random-Access Memories," *IEEE Transaction on Computers*, vol. C-24, pp. 150-157, February 1975.

[12] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*. New York: McGraw Hill Book Company, 1984.

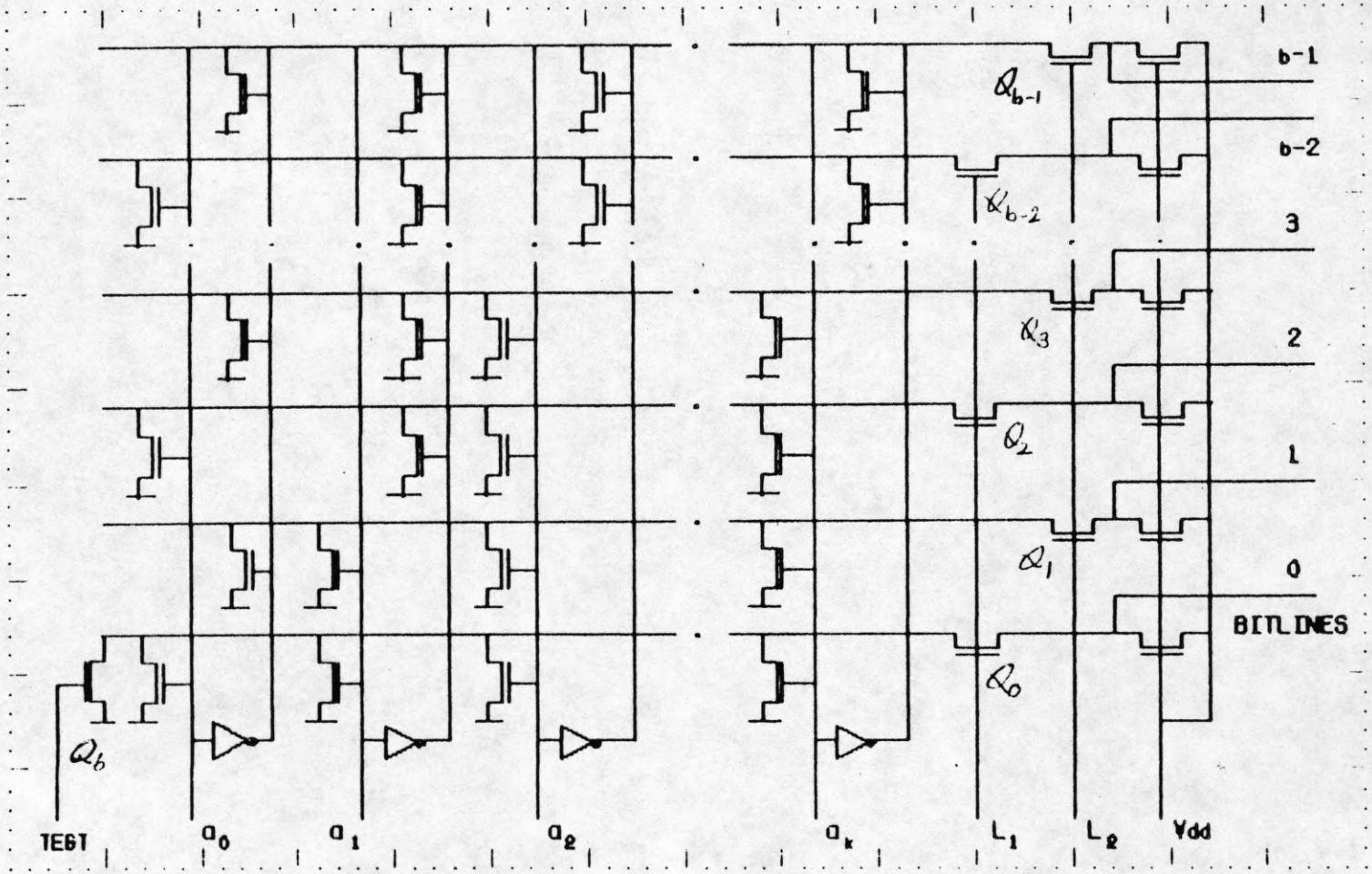
[13] P. K. Chatterjee, G. W. Tayor, A. F. Tasch, H. S. Fu, "High-Density Dynamic MOS Memory Devices," *IEEE Journal of Solid-State Circuits*, vol. SC-14, pp. 486-497, April 1979.

[14] D. S. Suk and S. M. Reddy, "A March Test for Functional Faults in Semiconductor Random Access Memories," *IEEE Transaction on Computers*, vol. C-30, pp. 982-985, December 1981.



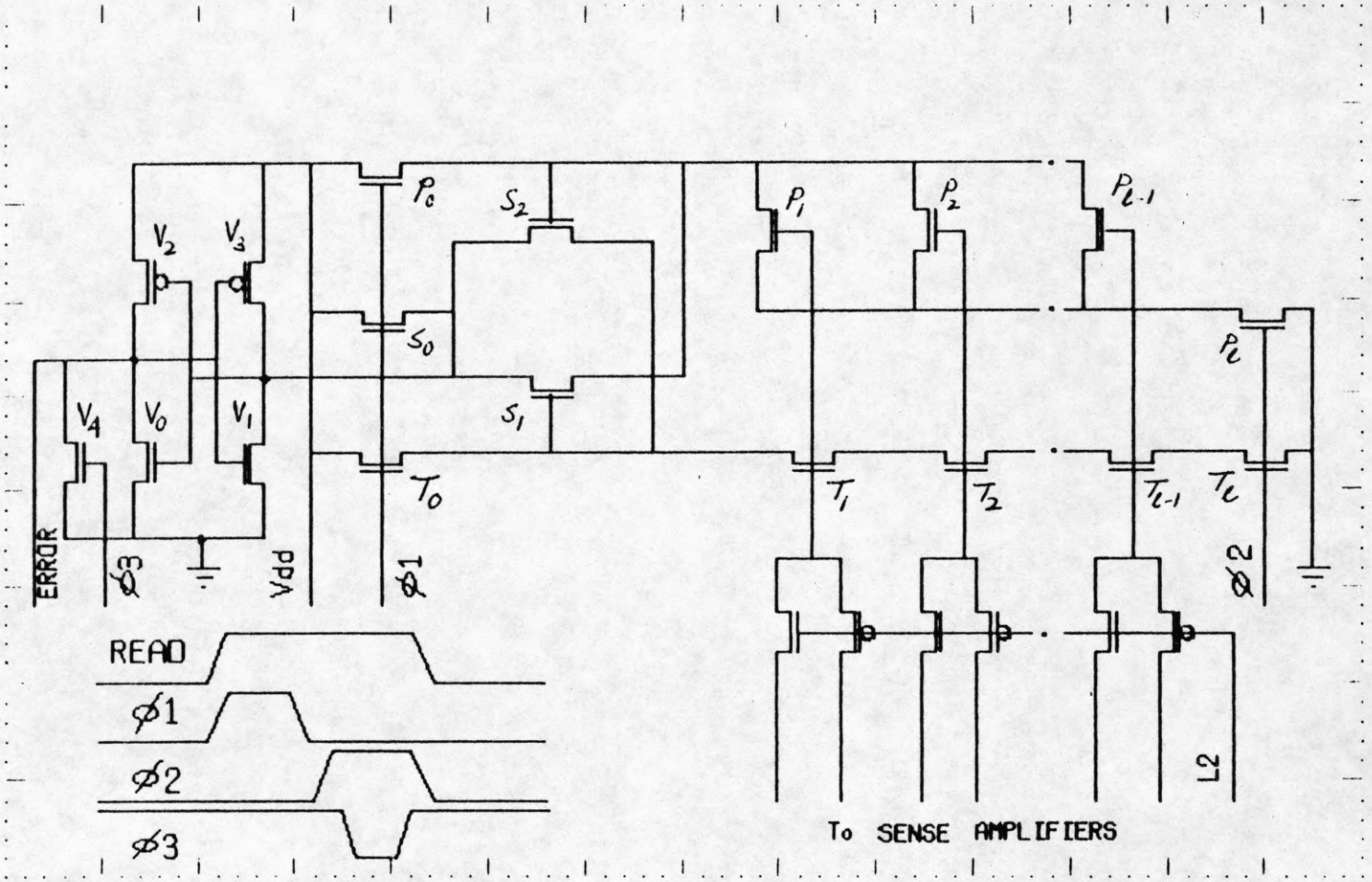
: TESTABLE RAM ORGANIZATION

Figure 3



: PLA IMPLEMENTATION OF RAM DECODER

Figure 5



: PARALLEL COMPARATOR WITH ERROR DETECTOR

Figure 6