Last June 96

Transparent Random Access Memory Testing for Pattern Sensitive Faults*

M. G. KARPOVSKY, *Fellow, IEEE* and V.N.YARMOLIK
Research Laboratory of Design and Testing of Computer Hardware
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, MA 02215

Abstract: This paper presents a new methodology for RAM testing based on the PS(n,k) fault model (the k out of n pattern sensitive fault model). According to this model the contents of any memory cell which belongs to an n-bit memory block, or the ability to change the contents, is influenced by the contents of any $k-1$ cells from this block. The proposed methodology is a transparent BIST technique, which can be efficiently combined with on-line error detection. This approach preserves the initial contents of the memory after the test and provides for a high fault coverage for traditional fault and error models, as well as for pattern sensitive faults. This paper includes the investigation of testing approaches based on transparent pseudoexhaustive testing and its approximations by deterministic and pseudorandom circular tests. The proposed methodology can be used for periodic and manufacturing testing and require lower hardware and time overheads than the standard approaches.

Key words: random access memory, memory testing, transparent memory testing, built-in self-test, pseudoexhaustive memory testing, pattern sensitive faults, signature analysis.

1 Introduction

The increasing sizes of memory chips resulted in dramatically increasing test costs for RAMs [1, 2]. There are several approaches to decrease a test cost while maintaining the required reliability. One of the most attractive solutions is based on memory testing methods which are available both for a manufacturer (manufacturing testing) and for a user (periodic field testing) [1]. Periodic testing is useful for critical applications such as nuclear plants, aerospace equipment, military electronics hardware, etc. In these cases it is important to recover the memory contents at the end of a test session. Transparent off-line testing avoids these drawbacks. It does not destroy the memory contents during a test session and provides high fault coverage [3].

*This work was supported by the NSF under Grant MIP9208487 and NATO under Grant 910411.

Cost reduction for manufacturing and periodic testing can be achieved by built-in self-test (BIST) [4]. There are several advantages of BIST. These include possibilities of at-speed testing and reduction of expensive external test equipment [5, 6]. For periodic testing, standard BIST cannot preserve initial contents of the RAM.

As the first step in development of a test for off-line RAM testing, a fault model representing physical defects is constructed [7, 8]. Typically, the test procedure is intended to detect all faults for the selected model (target faults).

With increasing densities, certain types of faults, such as neighborhood pattern sensitive faults (NPSFs), which are harder to detect, are becoming more important [9, 10, 11]. Active NPSFs (ANPSFs), Passive NPSFs (PNPSFs) and Static NPSFs (SNPSFs) have been considered in [10]. The actual choice of a neighborhood depends largely on the technical realization of the memory chip.

On-line testing is the main tool that ensures high reliability during the operational life time of the memory by employing on-chip error-detecting/correcting circuits. At the first step of developing an error-correction and/or error-detection built-in circuit, an error model representing failures that occur during normal operation is selected. Built-in hardware is required to detect and/or correct these target errors.

It is obvious, that target faults for off-line testing have the corresponding description as target errors used for on-line testing and vice versa [12]. But nevertheless, traditionally BIST for target faults (off-line testing) and error-correction circuits (on-line testing) have not been combined to minimize hardware and time overheads. On-line and off-line error detection techniques have been treated in isolation in the context of a system design.

We present in this paper a methodology for RAM testing which combines off-line and on-line approaches and can be used for periodic and manufacturing testing. For the case of manufacturing testing, our methodology allows us to generate tests with complexities from $O(N)$ up to $O(N^2)$. For periodic testing, our approach ensures the preservation of the initial contents of the memory after test sessions and provides high fault coverage for target faults. For different initial states of the block-under-test at every test session, we have different backgrounds that provide high fault coverage for non-target faults.

We propose in this paper a unified approach for transparent memory BIST based on the following general fault model. According to this model the contents of any cell which belongs to a n-bit memory block, or ability to change the contents, is influenced by the contents of any $k-1$ cells of the block. These contents consist of a pattern of 0s and 1s or changes in the patterns. We denote these faults as PS(n,k) faults.

One of the goals of this paper is to develop a unified approach for detecting PS(n,k) faults. We will show that the proposed testing techniques provide high fault coverage for existing memory fault models.

Specifically, in this paper the following problems will be investigated: (a) analysis of the efficiency of the selected PS(n,k) fault model, (b) design of pseudoexhaustive transparent tests for detection of PS(n,k) faults, (c) design of deterministic and pseudorandom transparent circular tests for detection of these faults, (d) BIST RAM implementations of the proposed tests, (e) analysis of performance (hardware and time

overheads, error/fault detection probabilities, fault coverage) of the proposed periodic and manufacturing memory tests, and (f) off-line utilization of error detection/correction memory circuits.

2 Fault Model

A good survey of the commonly used fault models for RAMs can be found in [10]. Here, we review only the background required for understanding the proposed general fault model.

By a simple fault we mean any *single fault*, which is not a *pattern sensitive fault*. The following simple faults are frequently considered [10]:

- *Stuck-at faults (SAF):* A permanent stuck-at 0 or stuck-at 1 fault that may occur in any memory cell. A cell i of the memory may be stuck-at 0 or stuck-at 1.
- *Transition faults (TF):* A memory cell i in the state $a_i \in \{0, 1\}$ fails to undergo an a_i to $\overline{a}_i$ transition when $\overline{a}_i$ is to be written in the cell; however, both states are possible for the cell, for instance, at power-on.
- *Coupling idempotent faults (CF_{id}):* A coupling idempotent fault is present from a cell i to a cell j if, when the cells contain a particular pair of binary values a_i and a_j, and $\overline{a}_i$ is written into cell i, then cell j, as well as cell i, changes state. These faults are denoted as $\langle\uparrow, 0\rangle$, $\langle\uparrow, 1\rangle$, $\langle\downarrow, 0\rangle$ and $\langle\downarrow, 1\rangle$ where $\uparrow$ and $\downarrow$ denote 0 to 1 and 1 to 0 transitions correspondingly.
- *Coupling inversion faults (CF_{in}):* A single coupling inversion fault is present from a cell i to a second cell j if, when cells i, j has values a_i and a_j, and $\overline{a}_i$ is written into cell i, then the state of cell j is complemented or "toggled" with respect to its previous value. These faults are denoted as $\langle\uparrow, \updownarrow\rangle$, $\langle\downarrow, \updownarrow\rangle$ where $\updownarrow$ denotes 0 to 1 or 1 to 0 transitions.

The above coupling faults, which involves two cells (*2-couplings*), are a special case of more general *k-coupling faults*. A *k-coupling fault* uses the same two cells as the 2-coupling fault and, in addition, only allows the fault to occur when another $k-2$ cells are in a certain state [13].

A *pattern sensitive fault* (PSF) is the most general memory fault model defined as follows: the contents of cells, or the ability to change the contents, are influenced by the contents of all other cells in the memory. These contents consists of a pattern of 0s and 1s, or changes in the patterns [10]. PSFs can be considered as the general case of *k-coupling faults*, for the case when $k = N$ (N is the size of the memory).

The common restriction on the class of PSFs is the *size n of the neighborhood*, and the number k of cells involved in the fault [10]. For $k = 2$ and $n = N$, the cell can be influenced by any other cell. This is a case of the coupling faults [13, 14, 15]. For a restricted k-coupling not linked (i.e. disjoint) fault, with $k = 3$ the test complexity is $N + 32 \cdot N \cdot \log_2 N$ [13] or $36 \cdot N + 24 \cdot N \cdot \log_2 N$ [19] operations. A family of efficient

deterministic tests for detecting *k-coupling faults* for $k \leq 5$, were recently described by Cockburn [16, 17]. Pseudorandom tests for k-coupling faults were proposed by Savir in [29]. These tests are less efficient for detecting simple faults (such as $SAFs$), and for faults for which deterministic tests are rather easy to construct.

In summary, current memory BIST approaches have the following major drawbacks: a low efficiency for detection of non-target faults, large hardware and time overheads, a destructive (not transparent) test procedure, and a fixed (not flexible) testing procedure, which usually generate only one or two test algorithms. These approaches are not tailored for the most efficient combined utilization of on-line and off-line test circuitry.

As a realistic and general fault model we propose the k out of n pattern sensitive fault (PS(n,k)), where $n \leq N$ is a block (region, row or column of memory array) for memory with the size N. According to this model, the contents of any memory cell which belongs to an n-bit memory block, or ability to change the contents, is influenced by the contents of any $k-1$ cells in the same block. These contents consists of a pattern of 0s and 1s or changes in the contents.

This PS(n,k) fault model does not depend on the realization of the memory chip and can be considered as an extension of the existing fault models. Depending on the values of k and n, PS(n,k) generalizes the following cases considered in the literature:

1. For $k = n = N$, PS(n,k) represents PSF [10], when the contents of a cell, or the ability to change the contents, is influenced by the contents of all other cells in the memory.

2. For $k < n$ and $n = N$ we have the case of the general k-coupling faults, when the base cell is influenced by a group of $k-1$ cells which can be placed anywhere in the memory.

3. For $k = n < N$ PS(n,k) is reduced to NPSFs, where the base cell is influenced by the $k-1$ cells from the neighbourhood. In this case PS(n,k) includes ANPSFs, PNPSFs and SNPSFs.

4. The case when $k < n$ and $n < N$ or $n \ll N$ is the more common and realistic one, when the contents of any memory cell which belongs to an n-bit memory block, or ability to change the contents, is influenced by the contents of any $k-1$ cells of the same block.

The major goal of this paper is to develop a unified approach for detection of PS(n,k) faults.

In the following sections we propose a unified methodology for memory testing based on the PS(n,k) model and a pseudoexhaustive transparent memory test (PXT) for detection of PS(n,k) faults. This PXT approach provides 100% fault coverage for PSFs with the complexity $C \cdot N$ where C depends on n and k only.

To reduce the overhead, two approximations to PXT are proposed. These approximations are based on deterministic and pseudorandom circular test sequences, which can be used for transparent periodic and manufacturing testing and require lower overhead.

3 Pseudoexhaustive transparent memory testing

Pseudoexhaustive tests (PXT) [20, 21, 22] for combinational devices have several attractive features. In this case, the testing procedure and its fault coverage are basically dependent neither on the fault model assumed nor on the specific circuit under test. This approach guarantees 100% coverage for all combinational faults. The major disadvantage of PXT is related to the fact that this approach is efficient only for combinational devices with outputs depending on a small number of inputs.

Denote by $E(n,s)$ a set of n-bit binary vectors such that all 2^s vectors appear at any s positions in $E(n,s)$. For example, $E(3,2) = \{000, 011, 101, 110\}$. Techniques for construction of $E(n,s)$ and estimations on minimal numbers $f(n,s) =\mid E(n,s) \mid$ of pseudoexhaustive test patterns can be found in [20, 21, 22].

Some examples of pseudoexhaustive $E(n,s)$ test patterns are shown in Table 1.

Table 1: Pseudoexhaustive test patterns $E(n,s)$

$E(3,2)$	$E(4,2)$	$E(4,3)$	$E(5,2)$	$E(5,3)$	$E(6,2)$	$E(6,3)$
000	0000	0000	11111	10000	000000	011111
011	0111	0011	10000	01000	000011	101111
101	1011	0110	01000	00100	011100	110111
110	1101	0101	00100	00010	101101	111011
	1110	1100	00010	00001	110110	111101
		1111	00001	01111	111011	111110
		1010		10111		100000
		1001		11011		010000
				11101		001000
				11110		000100
						000010
						000001

We will show in this section that PXT combined with standard March tests may be very efficient for transparent memory testing for detection of PS(n,k) faults.

The problem of pseudoexhaustive transparent memory testing can be formulated as construction of a test procedure for memory testing such that [10]:

Property 3.1 Each memory cell within k out of n cells in the block must be read in state 0 and in state 1, for all possible changes in remaining $k-1$ cells.

Property 3.2 Each memory cell within k out of n cells must be written and read in state 0 and in state 1, for all 2^{k-1} contents of the remaining in $k-1$ cells.

These properties should be satisfied for any k cells in any block of n cells to ensure P(n,k) detectability.

To simplify the test procedure we use the standard transparent march test MATS+ [10], which ensures 1 out of n exhaustive tests $E(n,1)$ such that in every 1 memory cell all 2^1 possible binary values 0 and 1 appear at least once. We apply MATS+ for different backgrounds $A(n,k-1)$ which represent modulo two sums of $f(n,k-1) = |E(n,k-1)|$ patterns of a pseudoexhaustive test, $E(n,k-1)$, with an initial n-bit memory block content A. Transparent memory test algorithm MATS+ (one-bit version) can be represented as [3]

$$\{\Downarrow [r(a_i), w(\overline{a}_i)]; \Uparrow [r(a_i), w(\overline{a}_i)]\}. \tag{1}$$

The proposed test procedure PXT consists of the following main stages:

1. A new background $A_{j+1}(n,k-1)$ is generated as a modulo two sum of the previous value of $A_j(n,k-1) = A \ominus E_j(n,k-1)$ and $\Delta E_{j+1} = E_j(n,k-1) \ominus E_{j-1}(n,k-1)$. As the result we will get a modulo two sum of the next pattern $E_{j+1}(n,k-1)$ with an initial n-bit memory block content ($A_{j+1}(n,k-1) = A \oplus E_{j+1}(n,k-1)$).

2. The transparent MATS+ memory test algorithm is applied.

3. The items 1 and 2 are repeated for all $f(n,k-1)$ backgrounds.

For example, if $E(3,2) = \{000, 011, 101, 110\}$, then $A(3,2) = \{a_0a_1a_2,\ a_0\overline{a}_1\overline{a}_2,\ \overline{a}_0a_1\overline{a}_2,\ \overline{a}_0\overline{a}_1a_2\}$ and for $E(5,2) = \{11111, 10000, 01000, 00100, 00010, 00001\}$ we will get $A(5,2) = \{\overline{a}_0\overline{a}_1\overline{a}_2\overline{a}_3\overline{a}_4,\ \overline{a}_0a_1a_2a_3a_4,\ a_0\overline{a}_1a_2a_3a_4,\ a_0a_1\overline{a}_2a_3a_4,\ a_0a_1a_2\overline{a}_3a_4\ ,\ a_0a_1a_2a_3\overline{a}_4\}$.

As in [16], the PXT procedure based on pseudoexhaustive $E(n,k-1)$ test patterns generate all 2^k k-bit test patterns in each k cells for any block of n cells and all cells within a k bits implement $\uparrow$ (0 to 1) and $\downarrow$ (1 to 0) transitions for all combinations of the remaining $k-1$ cells. This satisfies *Properties* 3.1 and 3.2 which allows detection of all PS(n,k) faults. For different initial states A of the block-under-test at every test session we have different backgrounds $A_j(n,k-1) = A \ominus E_j(n,k-1)$ which provides high fault coverage.

Two memory arrays are required for BIST implementation of the pseudoexhaustive transparent memory testing: array, S, for storing an initial content of the n-bit memory block-under-test, and array, E, for storing $f(n,k-1)$ by n-bit pseudoexhaustive test patterns. (Arrays S and E, may be a part of the original memory-under-test).

The following simple procedure can be used for BIST, based on pseudoexhaustive transparent memory testing.

Procedure PXT

Input:	Memory block size n;	
	Starting address w;	
Loop1:	For $i = 0$ to $n-1$ Do	} Store initial contents
Write:	$[a_{w+i}]$ to s_i;	
End Loop1:		
Loop2:	For $j = 0$ to $f(n,k-1)-1$ Do	
Loop3:	For $i = 0$ to $n-1$ Do	} Load next background
Read:	$[a_{w+i}]$ to D_0 and Compare with $[s_i]$	
Write:	$[D_0] \ominus [e_i^j]$ to a_{w+i};	
End Loop3:		

Loop4:	For $i = 0$ to $n-1$ Do	MATS+
Read:	$[a_{w+n-1-i}]$ to D_0 and Compare with $[s_{n-1-i}] \ominus [e^j_{n-1-i}]$;	
Write:	$[D_0] \ominus 1$ to $a_{w+n-1-i}$;	
End Loop4:		
Loop5:	For $i = 0$ to $n-1$ Do	
Read:	$[a_{w+i}]$ to D_0 and Compare with $[s_i] \ominus [e^j_i] \ominus 1$;	
Write:	$[D_0] \ominus 1$ to a_{w+i} and to s_i;	
End Loop5:		
End Loop2:		
Loop6:	For $i = 0$ to $n-1$ Do	Restoring initial contents
Read:	$[a_{w+i}]$ to D_0, and Compare with $[s_i]$	
Write:	$[D_0] \ominus [e_i^{f(n,k-1)-1}(n, k-1)]$ to a_{w+i}	
End Loop6:		

Here D_0 is an additional D-flip-flop; $[a_i]$ means the content of the ith memory cell; $\Delta E_j(n, k-1) = e^j_0 e^j_1 e^j_2 \ldots e^j_{n-1}$, $j \in \{0, 1, 2, \ldots, f(n, k-1)-1\}$ is determined by the patterns $E_j(n, k-1)$ and $E_{j+1}(n, k-1)$ of the pseudoexhaustive test $E(n, k-1)$, where $\Delta E_0(n, k-1) = E_0(n, k-1)$, and $\Delta E_c(n, k-1) = E_c(n, k-1) \ominus E_{c-1}(n, k-1)$, $c \in \{1, 2, 3, \ldots, f(n, k-1)-1\}$.

Pseudoexhaustive transparent memory testing procedure for the case of PS(5,3), where $k = 3$, $n = 5$, $A = 00000$ and backgrounds generated by the test $E(5, 2)$ (see Table 1) is shown in Table 2. (The bit we Read or Write at a given step of the procedure is printed in bold in Table 2.)

In this case, first we write in the block of five 1-bit cells pseudoexhaustive pattern $11111 \in E(5, 2)$, next read rightmost bit a_4, and write 0 in this cell, then read 1 in the next bit a_3 and write 0 in this cell. After zeros are written in all five cells we read the leftmost bit a_0 and write 1 in this cell. We repeat the procedure for all five cells of the block, and, after this, we repeat the procedure for the next pseudoexhaustive pattern $10000 \in E(5, 2)$. This procedure ensures detectability of PS(5,3) faults. The complexity of the above test algorithm depends on the complexity of MATS+ algorithm and the value of $f(n, k-1)$. For the total amount of *Write* and *Read* operations we have:

$$T[PXT(n, k)] = 6\, n \cdot f(n, k-1) + 3n, \qquad (2)$$

where $f(n, k-1)$ depends on k and n. Table 3 presents upper bounds for $f(n, 3)$ for $n = 2^c \leq 2^{20}$ [23]. Table 4 presents some known exact values of $f(n, k)$ for $n \leq 14$ and $k \leq 8$ [23, 26, 27, 28].

A hardware implementation of BIST based on PXT requires n memory cells for S array storing an initial data in the block-under-test, $nf(n, k-1)$ cells for E-array storing pseudoexhaustive backgrounds $E(n, k-1)$, two counters with $\lceil \log_2 n \rceil$ and $\lceil \log_2 f(n, k-1) \rceil$ flip-flops, a $\lceil \log_2 N \rceil$-bit register, a $\lceil \log_2 N \rceil$-bit adder, $\lceil \log_2 N \rceil$ (2×1) MUXs and a finite state machine with 5 flip-flops. Thus for the hardware overhead (in transistors) we have

$$L[PXT(n, k)] = O(\log_2 N + \log_2 n + nf(n, k-1)). \qquad (3)$$

Table 2: Pseudoexhaustive memory testing

March element		MATS+ for Different Backgrounds					
		11111	10000	01000	00100	00010	00001
$\Downarrow [r(a_i), w(\overline{a}_i)]$	$r(a_4)$	11111	10000	01000	00100	00010	00001
	$w(\overline{a}_4)$	11110	10001	01001	00101	00011	00000
	$r(a_3)$	11110	10001	01001	00101	00011	00000
	$w(\overline{a}_3)$	11100	10011	01011	00111	00001	00010
	$r(a_2)$	11100	10011	01011	00111	00001	00010
	$w(\overline{a}_2)$	11000	10111	01111	00011	00101	00110
	$r(a_1)$	11000	10111	01111	00011	00101	00110
	$w(\overline{a}_1)$	10000	11111	00111	01011	01101	01110
	$r(a_0)$	10000	11111	00111	01011	01101	01110
	$w(\overline{a}_0)$	00000	01111	10111	11011	11101	11110
$\Uparrow [r(a_i), w(\overline{a}_i)]$	$r(a_0)$	00000	01111	10111	11011	11101	11110
	$w(\overline{a}_0)$	10000	11111	00111	01011	01101	01110
	$r(a_1)$	10000	11111	00111	01011	01101	01110
	$w(\overline{a}_1)$	11000	10111	01111	00011	00101	00110
	$r(a_2)$	11000	10111	01111	00011	00101	00110
	$w(\overline{a}_2)$	11100	10011	01011	00111	00001	00010
	$r(a_3)$	11100	10011	01011	00111	00001	00010
	$w(\overline{a}_3)$	11110	10001	01001	00101	00011	00000
	$r(a_4)$	11110	10001	01001	00101	00011	00000
	$w(\overline{a}_4)$	11111	10000	01000	00100	00010	00001

Table 3: The values of $f(n, 3)$

n	2^2	2^4	2^6	2^8	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}
$f(n,3)$	8	17	32	51	60	84	101	108	129	132

Table 4: The values of $f(n,k)$

n	k						
	2	3	4	5	6	7	8
4	5	8	16				
5	6	10	16	32			
6	6	12	21	32	64		
7	6	12	24	42	64	128	
8	6	12	24	56	85	128	256
9	6	12	24	72	120	170	256
10	6	12	24	90	165	240	341
11	7	12	24	101	213	330	496
12	7	16	24	101	261	440	715
13	7	16	38	104	309	572	1001
14	7	16	52	118	357	728	1365

(This overhead may be further decreased if the part of the original memory is used for storing S and E-arrays.) Hardware overhead in terms of numbers of transistors required for BISTed CMOS SRAMs are shown in Table 5

Table 5: Hardware overheads (in percentages) for BISTed CMOS SRAMs

N	n=16			n=32			n=64		
	$k=4$	$k=5$	$k=6$	$k=4$	$k=5$	$k=6$	$k=4$	$k=5$	$k=6$
2^{18}	0.274	0.729	1.709	0.400	1.269	2.932	0.681	2.332	5.481
2^{20}	0.071	0.185	0.432	0.103	0.321	0.740	0.173	0.589	1.382
2^{22}	0.018	0.047	0.109	0.026	0.081	0.186	0.051	0.148	0.347

Since $f(n,3) \leq 7.5 \log_2 n$ [23] (see also Table 3), we have

$$T[PXT(n,4)] \leq 6\ n \cdot f(n,3) + 3n \leq 45\ n \cdot \log_2 n + 3n. \tag{4}$$

As we can see from (4) for the case $n = N$ and $k = 4$ our approach has about the same complexity as the known algorithms [16, 17] and detects all PS(n,k) faults without destructing initial data in the memory. The required time (see Table 6) to perform PXT for a 2^{20}-bit RAM and memory cycle time $50ns$ is about $47.3sec$.

Pseudoexhaustive tests based on $E(n,k-1)$ for $n = N$ (PXT(N,k)) detect all PS(N,k). The complexity $T[PXT(N,k)]$ is $O(N \cdot \log_2 N)$ since

$$2^{k-2} \log_2 N \leq f(N,k-1) \leq \cdot 2^{k-1}(k-1)(\log_2 e)^{-1} \log_2 N, \tag{5}$$

Table 6: The complexity $T[PXT(n,4)]$ in seconds

Size n of a block in bits	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}	2^{22}	2^{24}
$T[PXT(n,4)](sec.)$	0.01	0.09	0.49	2.12	10.2	42.0	208.1	907.9

for k fixed and N large enough [25, 26, 27].

We note that upper bound in (5) is not constructive. Construction for $E(n,k-1)$ with a complexity close to this bound for small k can be found in [23, 26, 27, 28] (see also Table 3 and 4).

Comparing pseudoexhaustive tests (PXTs) with pseudorandom tests proposed in [29], we note that PXTs guarantee that the escape probability (probability of missing an error) is equal to zero for PS(n,k) faults whereas for pseudorandom tests $\epsilon > 0$ and complexities of pseudorandom tests are higher than complexities of PXTs. For example, for pseudorandom tests [29, 10] complexity of detection of *2-couplings* is $90N$ for unbiased tests and $74N$ for best weighted tests with $\epsilon = 0.001$ [10] and for PXT with $\epsilon = 0$ we have by (2) $T(PXT(N,2)) = 15N$ only, since $f(N,2) = 2$ for any N. For *3-couplings* unbiased and weighted pseudorandom tests require $404N$ and $362N$ operations for $\epsilon = 0.001$ and for pseudoexhaustive tests with $\epsilon = 0$ we have by (2) for $N = 10^6 \quad T(PXT(10^6,3)) = 147N$.

We note that complexity $T(PXT(N,k))$ (see (2, (5)) can be considerably reduced if we replace pseudoexhaustive $(N,k-1)$ arrays $E(N,k-1)$ by "almost pseudoexhaustive" $(N,k-1)$ arrays [24]. For these arrays we require that a fraction of $k-1$ tuples of positions where all 2^{k-1} appear will be at least $1-\epsilon$ for some small $\epsilon > 0$. For example, using the techniques developed in [24] one can construct an almost pseudoexhaustive array for $N = 2^{20}$, $k-1 = 5$ and $\epsilon = 0.01$ with only 2^{12} test patterns. If we use this array as $E(2^{20},5)$ we will be able to detect at least 99% of all *6-couplings* for one test session which will take 1,238 sec. for a RAM with $N = 2^{20}$ and access time $50ns$. To increase fault coverage one can repeat these test sessions for different initial states of the memory-under-test.

Thus, PXT based on $E(N,k-1)$ provides for detection of all k-coupling faults, as well as all memory faults covered by the PS(N,k) fault model and its complexity is only $O(N \cdot \log_2 N)$. For $n = N$ the length of pseudoexhaustive test patterns equals N, which make this approach a viable alternative only for external testing.

In the following section we describe a modification of the PXT approach which results in a drastic reduction of length for pseudoexhaustive test patterns and the test complexity but still preserves fault coverage for PS(n,k) faults very close to 100%.

4 Test Session Organization

To simplify the test generation procedure for PXT we partition the original memory into $M = M(N,q,n)$ blocks with size $q \geq n$. In this case, every test session consists of

$M(N, q, n)$ elementary PXT sessions, where q represents the size of the memory block for one test session and n is the number of overlapped memory bits in two consecutive test sessions (see Fig. 1) to ensure pseudoexhaustive testing within any n-bit block of the memory array.

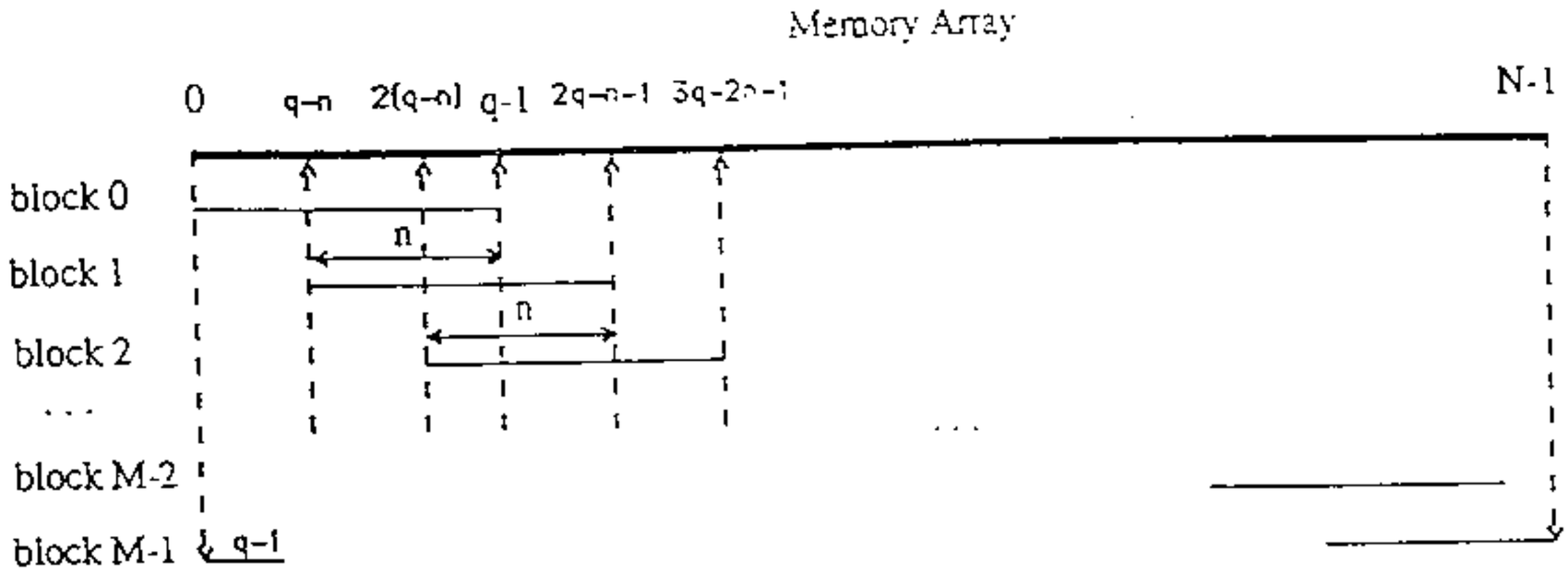

Figure 1: Timing diagram for the modified test procedure PXT for a 1-dim. memory array

For the case of a 2-dim. memory, the number of sessions is determined as

$$M(N, q, n_x, n_y) = \lceil \frac{N}{q_x q_y} \rceil = \frac{N}{q_x q_y}, \tag{6}$$

where $q = (q_x + n_x)(q_y + n_y)$ and $n = n_x n_y$. For this case, four consecutive PXT sessions are shown in Fig. 2.

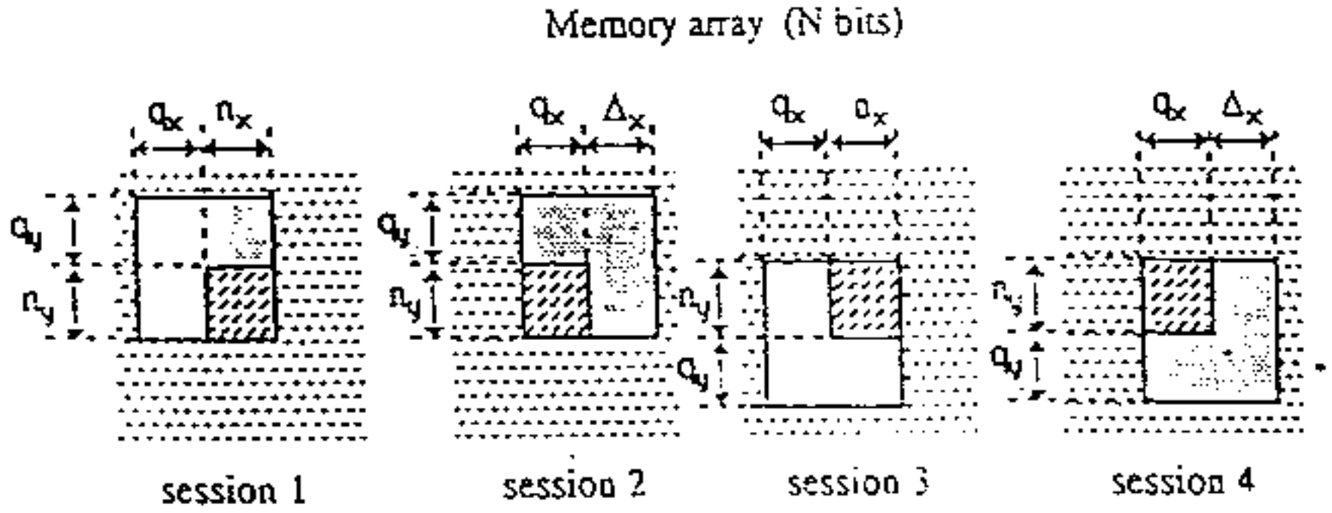

Figure 2: Timing diagram for a 2-dim. memory array

Then, by (2) we have for complexity $T = T[PXT(N, q, k)]$ of the modified PXT

$$T[PXT(N, q, k)] = M(N, q, n_x, n_y)\ T[PXT((q_x + n_x)(q_y + n_y), k)] =$$

$$= \frac{N}{q_x q_y}[\ 6\ (q_x + n_x)(q_y + n_y)\ f[(q_x + n_x)(q_y + n_y), k - 1] + 3(q_x + n_x)(q_y + n_y)]. \tag{7}$$

The hardware overhead for storage of $f[(q_x+n_x)(q_y+n_y), k-1]$ pseudoexhaustive patterns of length $q=(q_x+n_x)(q_y+n_y)$ and initial block contents can be estimated as

$$L=(q_x+n_x)(q_y+n_y)\ [f((q_x+n_x)(q_y+n_y), k-1)+1]. \tag{8}$$

Equations (7) and (8) illustrate the tradeoff between testing time and overhead for detection of PS(n,k) faults. For a given $n=n_xn_y$, an increase in the block size $q=(q_x+n_x)(q_y+n_y)$ results in a decrease of the test time T (determined by (7)) and in an increase in the overhead L (determined by (8)). It is easy to show that to minimize TL, optimal values for n_x and n_y are q_x and q_y.

In this case for linear 1-dim. arrays, $n_x=n$ and $q=q_x+n_x=2n$, $M(N,2n,n)=N/n$. Then

$$T[PXT(N,q,k)]=\frac{N}{n}\cdot T[PXT(2n,k)]=(12\cdot f(2n,k-1)+6)\cdot N. \tag{9}$$

For the case of a $\sqrt{N}\times\sqrt{N}$ 1-bit RAM with crosstalk within rows or columns only, one can take as blocks rows and columns. Since $q=\sqrt{N}$, n is 0 or 1, since any lines (rows or columns) either do not intersect or intersect in one cell only, and $M(N,\sqrt{N},n)=2\sqrt{N}$ we have $L=\sqrt{N}f(n,k-1)+\sqrt{N}$ and

$$T[PXT(N,\sqrt{N},k)]=2\sqrt{N}\ T[PXT(\sqrt{N},k)]=12N\cdot f(\sqrt{N},k-1)+6N. \tag{10}$$

For example, if $k=4$, then by (4) $f(\sqrt{N},3)\le 7.5\log_2\sqrt{N}=3.75\log_2 N$ and

$$L\le 3.75\sqrt{N}\log_2 N+\sqrt{N}, \qquad T[PXT(N,\sqrt{N},4)]\le 45N\log_2 N+6N. \tag{11}$$

For $N=2^{22}$ and access time $50\,ns$ we have $L\le 171,008$ bits and $T\le 208.8\,sec$ and this test detects all PS(n,4) faults.

To detect all PS(n,k) faults within any 2-dim. block with the size $n=n_xn_y$ we have $q_x=q_y=n_x=n_y=\sqrt{n}$, $q=4n$, and

$$M(N,4n,\sqrt{n},\sqrt{n})=\frac{N}{n}.$$

Test complexity $T[PXT(N,q,k)]$ for a 2-dim. memory is

$$T[PXT(N,q,k)]=\frac{N}{n}\cdot T[PXT(4n,k)]=(24\cdot f(4n,k-1)+12)\cdot N. \tag{12}$$

For example, for $n=16$, $k=4$ for detection of PS(16,4) faults, since $f(64,3)=32$ (see Table 3), we have by (12) $T[PXT(N,16,4)]=(24\cdot f(64,3)+12)\cdot N=780\cdot N$. The required time to perform PXT for a $N=2^{22}$ bit RAM and cycle time $50ns$ in this case is $163.6sec$.

The hardware overhead for the above mentioned example of BISTed $4Mb$ RAM consists of an extra (32×64) memory E for storing $f(64,3)=32$ by 64 bit pseudoexhaustive patterns and 64-bit memory S, which is about 0.016% of the original array.

5 Transparent Circular Testing

A considerable reduction in overhead can be achieved by replacing PXT by a test with a short test cycle, which does not depend on the initial memory contents.

The method consists of simulation of a deterministic circular test pattern generator on a block of the memory-under-test and it has a simple hardware implementation.

Let us assume that a memory-under-test contains N bits. We partition the original memory into $M = M(N, q, n)$ blocks with size $q \geq n$, where n represents the number of overlapped memory bits in two consecutive blocks. For every block we will organize q ($q \ll N$) neighboring cells in the testing mode as a circular test pattern generator.

The test session for this approach consists of M subsessions as described in Section 4. The implementation of a subsession requires the following three main stages.

1. Compute the signature for the block-under-test initial contents.

2. The block-under-test with the size q operates as the deterministic circular test pattern generator.

3. Compute the signature of the new contents of the memory block after stage 2 and compare it with the signature computed at stage 1.

The faults manifesting themselves during intermediate Read and Write operations would be detectable by the observation of distortions in final memory contents for the case when only one coupling may ocur in the block. Since the length of the cycle does not depend on an initial content of a block-under-test coupling, we only need to verify the resulting memory contents.

For RAM with a built-in error-detection/correction circuit, stages 1 and 3 can be avoided. In this case, a nonzero output (syndrome) of the error-detection/correction circuitry indicates the presence of a fault.

5.1 Deterministic Circular Test Patterns

In this section, we will use a twisted ring counter which creates well-defined periodic patterns [30] as a circular test pattern generator. A twisted ring counter is a circular shift register with the complemented output of the last flip-flop connected to the input of the first flip-flop.

It is easy to show that the twisted ring counter with q stages has cycles with the constant length $2q$ for any initial state iff $q = 2^c,\ c = 0, 1, 2, \ldots$ [31].

The following simple procedure can be used to simulate a twisted ring counter over a q-bit block of the memory-under-test. Here D_0 and D_1 are the first and second stages of an additional two stage shift register, SR; $q = 2^c$ is the memory-under-test block size; $[a_i]$ means the initial contents of the ith memory cell.

Procedure TRC

Input:	Memory block size q;
	Starting address w;
Read:	$[a_{w+q-1}]$ to D_0;
Shift:	SR;
Loop1:	For $j = 1$ to $2q$ Do
Read:	$[a_w]$ to D_0;
Write:	$[D_1] \oplus 1$ to a_w;
Shift:	SR;
Loop2:	For $i = 1$ to $q - 2$ Do
Read:	$[a_{w+i}]$ to D_0;
Write:	$[D_1]$ to a_{w+i};
Shift:	SR;
End:	Loop2;
Write:	$[D_1]$ to a_{w+q-1};
End:	Loop1;

In the above procedure, the *Write* and *Shift* operations in Loop1 and Loop2 can be accomplished simultaneously and do not need more than one control signal. It should be mentioned that we can start this procedure from any address $w \in \{0, 1, 2, \ldots, N-1\}$. There are direct and inverse TRC procedures with opposite directions of simulated shifts of the twisted ring counter. Complexity $T = T[TRC]$ of TRC for a block with q cells is equal to

$$T(TRC) = 4q^2 - 2q + 2. \tag{13}$$

Procedure TRC can be easily extended to a case of the word-oriented memory with the size $N = W \times m$, with the same test complexity determined by (13).

For any initial state of a q-bit block-under-test and for any $k \leq q = 2^c, (c = 0, 1, 2, \ldots)$ the twisted ring counter algorithm provides for at least $2k$ different binary codes at any k positions [31].

When we use the direct and inverse TRC procedures for the same memory block, all simple faults are detectable [31]. For this case, the TRC transparent procedure consists of the following steps.

1. Compute the signature of the memory contents $A = (a_0 a_1 a_2 \ldots a_{q-1})$ $(S \leftarrow A)$.
2. Simulation of $2q$ shifts for the direct TRC procedure.
3. Compute the signature S^* of resulting memory contents and compare it with the signature S of initial contents $(S = S^*?)$. If $S = S^*$, then repeat the steps 2 and 3 for inverse TRC procedure.

Every subsession of TRC transparent testing consists of the twisted ring counter simulation (direct and inverse) and three stages of signature calculation. The total complexity $T = T[TRC(q)]$ of TRC transparent testing is

$$T[TRC(q)] = 2T[TRC] + 3q = 8q^2 - q + 4. \quad (14)$$

This approach provides high fault coverages for pattern-sensitive faults due to a high probability of generating a large portion of all possible test patterns of length $k \leq q$ within any k memory cells [31].

Thus, the deterministic circular test has good fault coverage for pattern sensitive faults PS(n,k) for a small size k of the neighborhood and a nonzero but decreasing coverage for growing k.

The fault coverage for simple faults is 100%. All TFs, CFs and single SFs are detectable [31]. Every single stuck-at fault cannot be detected during one session of TRC procedure (direct or invers) for only one initial state of q cells. For every initial state there is a set of undetectable multiple stuck-at faults.

If an initial state is equal, for example, to 0000 and we use direct TRC procedure there are one single ($a_3 \equiv 0$), three double (($a_0 \equiv 0, a_3 \equiv 0$), ($a_1 \equiv 0, a_3 \equiv 0$) and ($a_2 \equiv 0, a_3 \equiv 0$)), three triple (($a_0 \equiv 0, a_1 \equiv 0, a_3 \equiv 0$), ($a_0 \equiv 0, a_2 \equiv 0, a_3 \equiv 0$) and ($a_1 \equiv 0, a_2 \equiv 0, a_3 \equiv 0$)) and one fault with multiplicity four ($a_0 \equiv 0, a_1 \equiv 0, a_2 \equiv 0, a_3 \equiv 0$) which are undetectable. At the same time, for 0101 the set of undetectable faults contains only two faults i.e. ($a_1 \equiv 1, a_2 \equiv 0, a_3 \equiv 1$) and ($a_0 \equiv 0, a_1 \equiv 1, a_2 \equiv 0, a_3 \equiv 1$).

It should be mentioned that if we use both direct and inverse TRC procedures, all stuck-at faults with the multiplicity less than q are detectable. For example, for $(a_0a_1a_2a_3) = 0000$ the only stuck-at fault which is not detected by both direct and inverse TRC procedures is the fault ($a_0 \equiv 0, a_1 \equiv 0, a_2 \equiv 0, a_3 \equiv 0$) with multiplicity four.

Probabilities $P(l)$ of detection of multiple SFs for different multiplicities l, are shown in Fig. 3. (All experimental results in Fig. 3 and 4 have been obtained for 10^5 randomly chosen initial states A of the block under test.)

It follows from Fig. 3 that already for $q = 32$ we have all 2^k test vectors at any k bits within any block-under-test for $k \leq 5$, which allows detection of *6-coupling faults* and in addition to this all multiple stuck-at faults with a multiplicity at most 24 are detected. In this case for a bit-oriented 2^{20}-bit RAM with a cycle time $50nsec.$ and $q = 2n = 128$ the required testing time is 107.3sec. (see formula (18) below) for 1-dim. and 429.4sec. for 2-dim. memory (see formula (19)).

5.2 Pseudorandom Circular Test Patterns

Transparent testing of RAMs based on simulation of twisted ring counters on blocks of q cells (section 6.1) has a simple implementation and a low test complexity, but the number, $f(k)$, of different combinations which are generated at any k bits in a block for this approach depends on the initial state of the cells in the block. In the worst case for one test session this number is equal to $2k$, which may be not sufficient for detection of PS(n,k) faults. To overcome this difficulty we will outline in this section another approach which can provide for any $f(k)$, ($f(k) \leq 2^k$), by increasing test complexity. This approach will be based on simulation of LFSRs [32, 33] generating pseudorandom (PSR) test patterns on a block of size q.

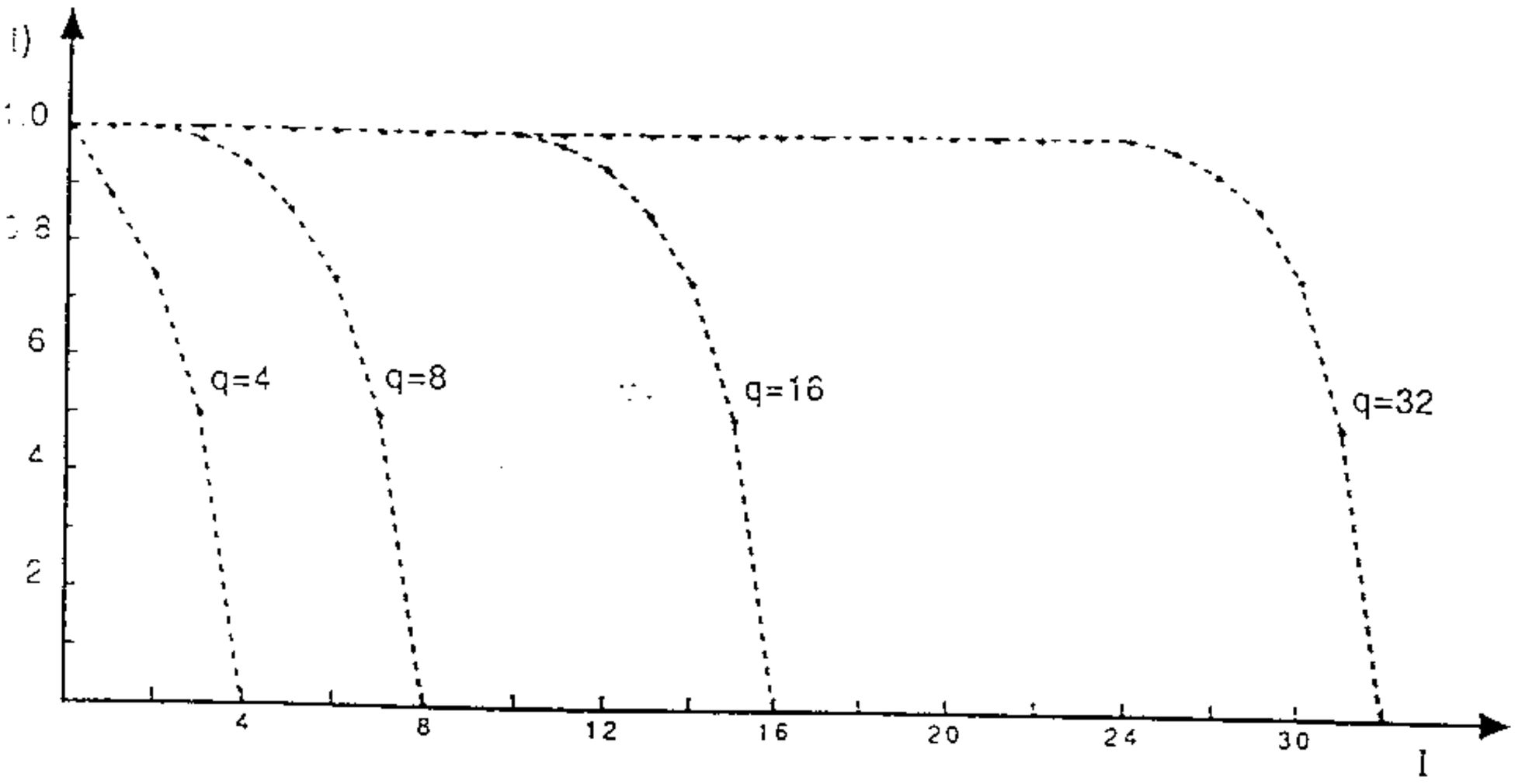

Figure 3: Probability $P(l)$ of detection of multiple SFs for deterministic tests based on simulation of q-bit twisted ring counter on q-bit blocks-under-tes

A major advantage of the pseudorandom testing as an approximation of pseudoexhaustive testing is in the simplicity of its implementation. LFSR properties are determined by the characteristic polynomial $\varphi(x) = 1 \oplus \beta_1 x^1 \oplus \beta_2 x^2 \oplus \ldots \oplus \beta_q x^q$, where $\beta_i \in \{0, 1\}$ for $i = 1, 2, \ldots, q$. LFSRs have been widely used for generation of PSR test patterns and for compression of test responses [33, 34].

We will describe in this section a procedure PSR for transparent testing of RAMs based on simulation of Q shifts of two LFSRs with generating polynomials $\varphi(x) = 1 \oplus x^1 \oplus x^q$ and $\psi(x) = 1 \oplus x^{q-1} \oplus x^q$ on a memory block with q cells. (Polynomials $\varphi(x)$ and $\psi(x)$ may be replaced by any pair of reciprocal polynomials. Selecting $\varphi(x) = 1 \oplus x^1 \oplus x^q$ results in only one two-input XOR gate required for simulation, minimizes switching of addresses and allows usage of a standard counter for address generation. Polynomials $\varphi(x)$ and $\psi(x)$ are primitive for $q = 2, 3, 4, 6, 7, 15, 22, 60, 63, 127, \ldots$).

If we use the polynomial and its reciprocal sequentially in one algorithm we will have the possibility to return back to the initial state at the end of a test session.

The pseudorandom transparent memory testing procedure consists of the following main steps.

1. Compute the signature of an initial memory contents $A = (a_0 a_1 a_2 \ldots a_{q-1})$ $(S \leftarrow A)$.

2. Simulation of Q shifts for the LFSR generated by $\varphi(x) = 1 \oplus x^1 \oplus x^q$ which generates the following sequence of internal states of the block $A, \alpha A, \alpha^2 A, \ldots, \alpha^Q A$, where $1 \oplus \alpha \oplus \alpha^q = 0$.

3. Simulation of Q shifts of the LFSR generated by $\psi(x) = 1 \oplus x^{q-1} \oplus x^q$ which generates a sequence $\alpha^Q A, \alpha^{Q-1} A, \ldots, \alpha A, A$.

4. Compute the signature S^* of the resulting memory contents and compare it with the signature S of the initial contents. If $S = S^*$ then go to the next step.

5. Negate data in the block ($A \leftarrow \overline{A}$).

6. Repeat steps 2 and 3.

7. Negate data in the block ($A \leftarrow \overline{A}$).

8. Compute the signature S^* of the resulting memory contents and compare it with the signature S of initial contents.

We note, that with increase in Q, the number of simulated shifts, the number $f(k)$ of different k-bit patterns appearing at any k cells of the block grows monotonically. This results in the monotonic increase in the probability of detection of PS(n,k) faults.

The following simple procedure PSR can be used to simulate the LFSR, described by $\varphi(x) = 1 \oplus x^1 \oplus x^q$ over q-bit memory block. We use the same notation as for procedure TRC (see section 5.1).

Procedure PSR

Input:	Memory block size q;
	Starting address w;
	Number of shifts Q;
Read:	$[a_{w+q-1}]$ to D_0;
Shift:	SR;
Loop1:	For $j = 1$ to Q Do
Read:	$[a_w]$ to D_0;
Write:	$[D_1] \oplus [D_0]$ to a_w;
Shift:	SR;
Loop2:	For $i = 1$ to $q - 2$ Do
Read:	$[a_{w+i}]$ to D_0;
Write:	$[D_1]$ to a_{w+i};
Shift:	SR;
End:	Loop2;
Write:	$[D_1]$ to a_{w+q-1};
End:	Loop1;

The *Write* and *Shift* operations in Loop1 and Loop2 can be accomplished simultaneously and the complexity, $T = T[PSR]$, of the simulation is

$$T[PSR] = 2qQ - Q + 2. \tag{15}$$

The complexity $T = T[PSR(q,Q)]$ of pseudorandom transparent testing which includes all stages of simulation and signature calculation for a memory block with the size q is

$$T[PSR(q,Q)] = 8qQ - 4Q + 8 + 7q. \quad (16)$$

Test session organizations with overlapping blocks of q cells and performance analysis for pseudorandom tests based on PSR(q,Q) can be performed the same way as for deterministic patterns generated by simulation of twisted ring counters.

For example, for a $\sqrt{N} \times \sqrt{N}$ 1-bit RAM with crosstalk limited to rows and columns only, we can use a BIST realization of the pseudorandom test for $q = \sqrt{N}$. In this case blocks are rows and columns. Then, taking into account that a number of elementary sessions $M = 2\sqrt{N}$ we have for the test complexity

$$T[PSR(N,Q)] = 2\sqrt{N}\, T[PSR(\sqrt{N},Q)] = 16NQ + 14N - 8\sqrt{N}Q + 16\sqrt{N}, \quad (17)$$

For example, for $N = 2^{22}$ and $Q = 32$ and cycle time $50ns$ we have $T = 109.1sec.$

The efficiency of the pseudorandom transparent memory testing depends on the value of Q. For the case of $Q = q$ it is easy to show that any single stuck-at fault within a q-bit memory block is masked only for one out of 2^q possible initial states [32].

The number of different combinations, $f(k)$, which are generated at any k bits within a memory block does not depend on initial states of the cells. We can get a good approximation to pseudoexhaustive testing by increasing the value of, Q, which provides for any $f(k), k \leq q$.

As in the case of deterministic tests (see Section 5.1), faults manifesting themselves during intermediate Read and Write operations are detected by observation of distortions in final states of the block-under-test.

Thus, pseudorandom circular tests have good fault coverage for pattern sensitive faults for a small size neighborhood, k, and a nonzero but decreasing coverage for growing k.

Aliasing probabilities P_{al} (probabilities of missing a fault) for CF_{id} $\langle\uparrow,0\rangle$, $\langle\uparrow,1\rangle$, $\langle\downarrow,0\rangle$ and $\langle\downarrow,1\rangle$ (see Section 2) are shown in Fig. 4. As we can see all CFs are detectable with probability very close to 1 for $Q \geq 8$.

Fault coverage for single and multiple stuck-at faults for pseudorandom tests is the same as for deterministic tests described in the previous subsection [32].

As it was shown in [32] by selecting $q = 32$ and $Q = 16$ we have all 2^k test vectors for any k bits ($k \leq 4$) within the block-under-test and, in addition to this, the fault coverage for *2-coupling faults* is very close to 100%. For a bit-oriented 2^{20}-bit RAM with $q = 2n = 2^{10}$, $Q = 32$ and cycle time $= 50\,nsec.$ the required testing time is 107.2sec. for a 1–dim. array (see formula (20) below) and 214.8 sec. for a 2-dim. array (see formula (21)).

In general, TRC and PSR procedures have simple hardware implementations, low test complexities and high fault coverages for simple and pattern sensitive faults. The TRC procedure is very efficient for a BIST implementation with a constant testing time,

which is equal to $4q^2 - 2q + 2$. The PSR procedure may be used to obtain high fault coverage by increasing test complexity (testing time).

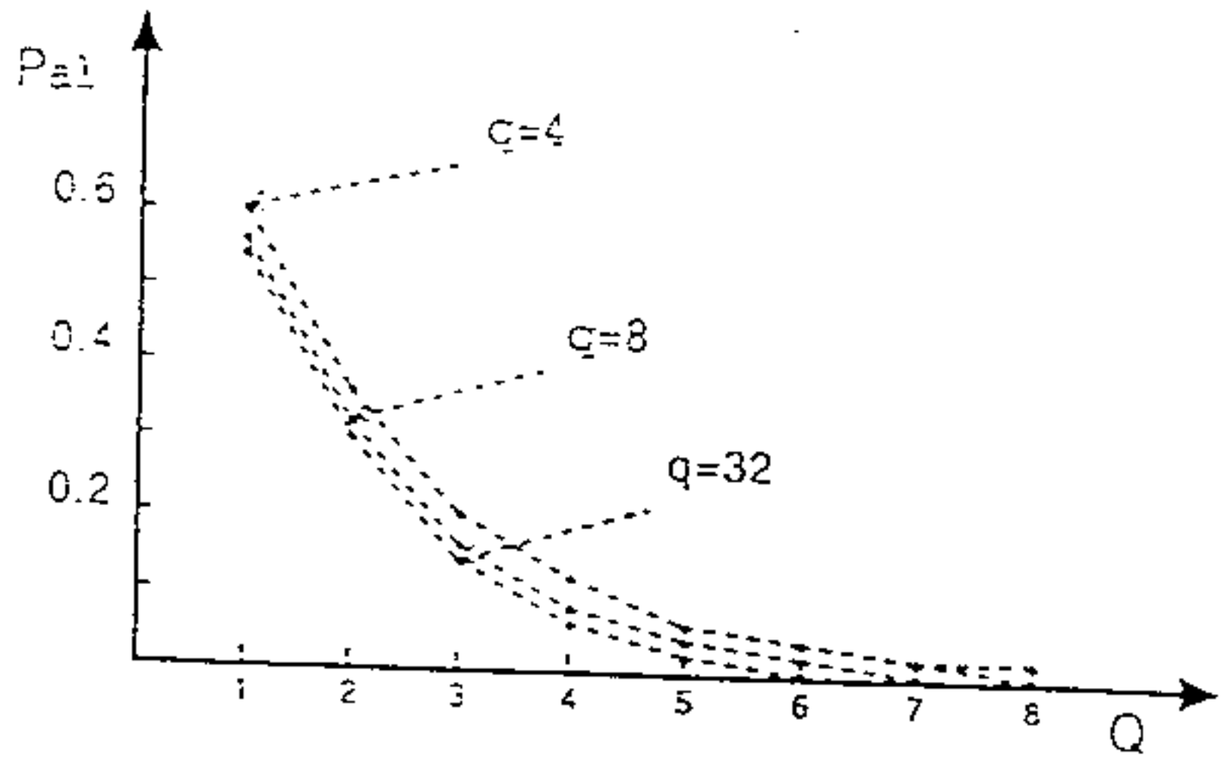

Figure 4: Aliasing probability P_{al} for CF_{id} $\langle\uparrow,0\rangle$, $\langle\uparrow,1\rangle$, $\langle\downarrow,0\rangle$ and $\langle\downarrow,1\rangle$ for pseudorandom tests based on simulation of LFSRs on a q-bit block-under-test

6 Hardware Implementations

In this section we describe hardware implementations for procedure TRC based on simulation of the twisted ring counter, over a q-bit(word) block of the memory-under-test. Similar implementations can be used for procedure PSR based on simulation of a q-bit LFSR over a q-bit (word) memory block.

The block diagram for the hardware implementation for a bit-oriented RAM with transparent testing is shown in Fig. 5.

The hardware implementation of the circular tests (both for TRC and PSR procedures) requires two($2m$) flip-flops and one(m) XOR gates for the test pattern generator (TPG), one(m) multiplexer (MUX) for the mode control and one signature analyzer (SA) per memory unit (m is the number of bits in a word). Counter modulo $\lceil \log_2 q \rceil$, counter modulo $\lceil \log_2 N \rceil$, the adder and the multiplexer in the control unit generate the address within a block-under-test. The finite state mashine (FSM) generates control signals corresponding to TRC or PSR. TRC generates test patterns for one q-bit block-under-test. For a small q and $m = 1$, the signature analyzer may be replaced by a q-bit shift register and, for the general case, it consists of r-bit signature analyzer, r-bit register and a comparator. In many cases one can take $r = m$.

The hardware implementation of the test controller for deterministic and pseudorandom transparent testing depends on the size N of memory, as well as on the number of cells q within the memory block. As in the case of pseudoexhaustive testing (see Section 3, formula (3) and Table 5) the complexity of a test controller can be estimated as $O(\log_2 N) + O(\log_2 q)$.

The RAM may operate in the normal computing mode and in the transparent self-testing mode. Transparent periodic testing of the RAM is divided into $M(N, q, n)$

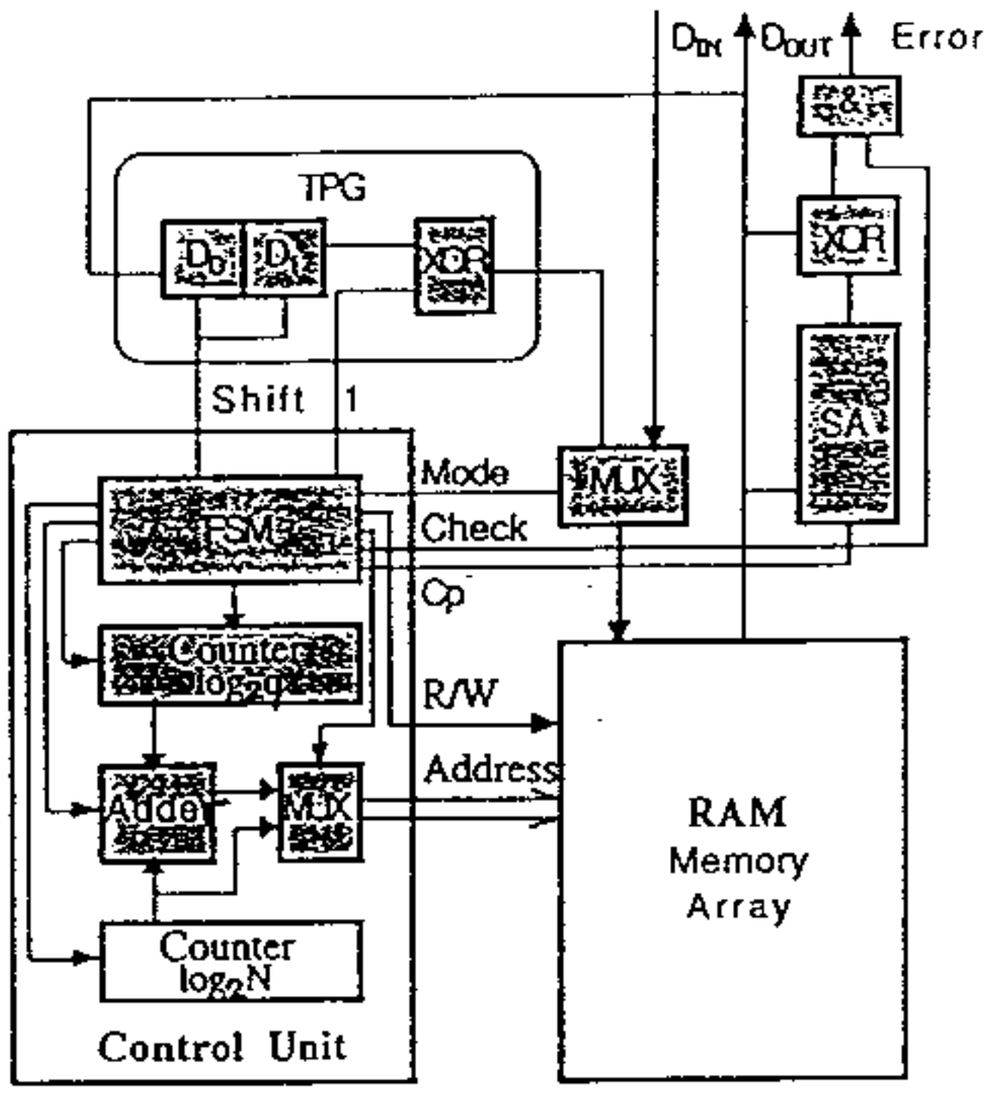

Figure 5: Block diagram for transparent self-testing for a bit-oriented RAM

subsessions (see Section 4), where $q \ll N$ is the size of a block used for one subsession. Every subsession consists of the following steps the sequences of which are determined by the TRC or PSR transparent testing procedures.

1. Compute a signature for the initial state of the block of q cells. For small q (say, $q \leq 32$) cell contents are written into a shift register (SA) (see Fig. 5) and for larger q, SA is replaced by a primitive r-bit $LFSR$ $r \leq q$. This stage requires q memory cycles.

2. The block of q memory cells operate as the twisted ring counter (direct or inverse) or as the LFSR described by the polynomial $\varphi(x) = 1 \oplus x^1 \ominus x^q$ or $\varphi(x) = 1 \ominus x^{q-1} \ominus x^q$. The TPG generates test patterns by simulation of TRC or PSR procedures.

3. Compute a signature of the new state of the block and compare it with the one computed at step 1. For the RAM shown in Fig. 5 this step requires comparison of contents of the memory block with contents of SA. This step requires q cycles.

For the complexity $T[TRC(N, q, n)]$ of one test session we have in view of (14) for a 1-dim. memory and optimal value of $q = 2n$

$$T[TRC(N, 2n, n)] = M(N, 2n, n)T[TRC(2n, n)] = (32n - 2 + \frac{4}{n})N, \qquad (18)$$

and for a 2-dim. memory with $q = 4n$ we have

$$T[TRC(N, 4n, n)] = M(N, 4n, n)T[TRC(4n, n)] = (128n - 4 + \frac{4}{n})N, \qquad (19)$$

For the complexity, $T[PSR(N,q,n)]$, of one test session we have in view of (16) for a 1-dim. memory

$$T[PSR(N,2n,n)] = M(N,2n,n)T[PSR(2n,Q)] = (16Q + \frac{8}{n} - \frac{4Q}{n} + 14)N, \quad (20)$$

and for a 2-dim. memory

$$T[PSR(N,4n,n)] = M(N,4n,n)T[PSR(4n,Q)] = (32Q + \frac{8}{n} - \frac{4Q}{n} + 28)N, \quad (21)$$

where $T[TRC(2n,n)]$, $T[PSR(2n,Q)]$, $T[TRC(4n,n)]$ and $T[PSR(4n,Q)]$ are the complexities of TRC and PSR. The complexity, $T[TRC(N,q,n)]$, is $O(N)$ for $n \ll N$ and $O(N^2)$ for $q = N$ and $M = 1$, $(n = 0)$ as follows from (18). There is a possibility of generating the test with the complexity $O(N^3)$, for the case when $q = N$ and using TRC or PSR for all addresses $w \in \{0, 1, 2, \ldots, N-1\}$, with $n = q - 1$.

The hardware implementation of a transparent self-testing memory for a more common structure of a W word by m bit RAM requires m stages of the test pattern generator TPG, an m-bit signature analyzer ($MISR$), an m-bit register (RG), and a comparator. In this case, q words with m cells in a word are tested simultaneously in time due to operating as m parallel deterministic or pseudorandom circular test pattern generators (For word-oriented memories we assume as in [35] that couplings may appear only between bits in different words). As in the previous case, the self-testing procedure requires $M(N,q,n)$ subsessions and test complexities $T[TRC(N,q,n)]$ and $T[PSR(N,q,n)]$ are determined by (18) and (20). The only difference is that during steps 1 and 2 the compaction of the contents of q memory words is implemented by a $MISR$.

An important feature of our approach is the ability to combine off-line and on-line testing. For the PSR procedure, we can use on-line circuitry for the response evaluation and fault observation. In this case, we do not need to calculate and compare the signatures [32, 35].

If for any codeword X, which belongs to the code used for on-line error detection, the negation of the word $\overline{X}$ also belongs to the same code then there is the possibility of using the on-line testing circuits (syndrome evaluators) for off-line testing with circular test patterns, based on TRC, since for TRC, if X is an internal state of a q-bit block-under-test, than $\overline{X}$ is also generated by TRC.

For the case of 1-dim. parity check codes and even m (X and $\overline{X}$ belongs to parity check codes iff m is even) our approach (both TRC and PSR) has a simple hardware implementation. During the off-line test mode, this hardware implements a transformation of the memory word contents within the parity code codewords. For a faulty RAM, a memory word will be changed to a noncode word. A manifestation of any fault through the cyclic test pattern application can be detected in the *Read* mode by the on-line parity check circuit.

Thus, if the RAM has an on-line test circuit, it can be used for the off-line transparent self-testing procedure and allow us to reduce hardware and time overheads for BIST. For

the case of parity check codes, there is one extra column for the parity and an additional circuit for on-line testing. To implement the transparent off-line circular test pattern, we have to generate, in parallel, the circular test patterns for all columns of the RAM. The circular test patterns are generated for information columns only. Check bits will be generated by the *Encoder* and the output response will be analyzed by the *Decoder* of the corresponding code. For a RAM with the on-line testing capability we do not need an m-bit *MISR* or m-bit register and comparator, which simplifies the hardware implementation.

7 Conclusions

In this paper we have presented unified approach for memory testing based on PS(n,k) fault model and pseudoexhaustive transparent memory testing and and deterministic pseudorandom circular tests. It was shown that proposed test procedures have high fault coverages for simple faults, as well as for pattern sensitive faults.

Pseudoexhaustive testing provides for a high fault coverage compared with a circular testing. At the same time, it requires more hardware for storage of pseudoexhaustive patterns. Circular memory testing, (TRC and PSR), require low hardware overheads for BIST implementations and provides relatively high fault coverage. For pseudorandom circular testing, fault coverage grows with an increase in testing time. It was shown also that BIST implementations of circular memory testing have some attractive features, such as combined utilization of on-line and off-line circuitry, low hardware and time overheads and non-destructive off-line testing.

Acknowledgment

The authors wish to thank Prof. Lev B. Levitin and Ms. D.Das of the Boston University, Boston, the reviewers and the editor for the valuable comments.

References

[1] T.Yamada, A.Fujiwara, and M.Inoue "COM (Cost Oriented Memory) Testing", *Proceedings International Test Conference*, Baltimore, September 1992, p. 259.

[2] A.Tuszynski "Memory Chip Test Economics", *Proceedings International Test Conference*, Washington, September 1986, pp. 190-194.

[3] M.Nicolaidis "Transparent BIST for RAMs", *Proceedings International Test Conference*, Baltimore, September 1992, pp. 598-607.

[4] S.K.Jain and S.H.Stroud "Built-In Self-Testing of Embedded Memories", *IEEE Design and Test of Computer*,Vol. 3, No. 5, October 1986, pp. 27-37.

[5] Y.Zorian and V.K.Agarwal "On improving the effectiveness of the standard BIST approach", *Proceedings 6th International Conference Custom and Semicustom ICs* , November 1986.

[6] K.T.Le and K.K.Saluja "A Novel Approach for Testing Memories Using a Built-in Self Testing Technique", *Proceedings International Test Conference*, Washington, September 1986, pp. 830-838.

[7] R.Dekker, F.Beenker and L.Thijssen "Fault Modeling and Test Algorithm Development for Static Random Access Memories", *Proceedings International Test Conference*, Washington, September 1988, pp. 343-352.

[8] S.Nair, F.Agricola and W.Maly "Failure Analysis of High-Density CMOS SRAMs", *IEEE Design and Test of Computer*, Vol. 10, No. 2, June 1993, pp.13-23.

[9] A.J. Van de Goor and C.A.Verruijt "An overview of deterministic functional RAM chip testing", *ACM Computing Surveys*, Vol. 22, No. 1, March, 1990.

[10] A.J. Van de Goor "Testing Semiconductor Memories, Theory and Practice", *John Wiley and Sons*, Chichester, 1991.

[11] J.P.Hayes "Detection of Pattern-Sensitive Faults in Random-Access Memories", *IEEE Transactions on Computers*, Vol. C-24, No. 2, 1975, pp. 150-157.

[12] V.N.Yarmolik and M.Nicolaidis "Exact Aliasing Computation And/Or Aliasing free design for RAM BIST", *Proceedings of Workshop on Memory Testing*, San Jose, August 1993.

[13] C.Nair, S.M.Thatte and J.A.Abraham "Efficient Algorithms for Testing Semiconductor Random-Access Memories", *IEEE Transactions on Computers*, Vol. C-27, June 1978, pp. 572-576.

[14] D.S.Suk and S.M.Reddy "A March Test for Functional Faults in Semiconductor Random-Access Memories", *IEEE Transactions on Computers*, Vol. C-30, No. 12, 1981, pp. 982-985.

[15] M.Marinescu "Simple and efficient algorithm for functional RAM testing", *Proceedings International Test Conference*, 1982, pp. 236-239.

[16] B.F.Cockburn "Deterministic Tests for Detecting Single *V*-Coupling Faults in RAMs", *Journal of Electronic Testing: Theory and Applications*, Vol. 5, 1994, pp. 91-113.

[17] B.F.Cockburn "A Transparent Built-In Self-Test Scheme for Detecting Single *V*-Coupling Faults in RAMs", *IEEE International Workshop on Memory Technology Design and Testing*, August 8-9, 1994, San Jose, CA, pp. 119-124.

[18] J.Savir et. al. "Testing for Coupled Cells in Random-Access Memories" , *Proceedings International Test Conference*, Washington, September, 1989, pp. 439-451.

[19] C.A.Papachristou and N.B.Saghal "An Improved Method for Detecting Functional Faults in Random-Access Memories", *IEEE Transactions on Computers*, Vol. C-34, No. 2, 1985, pp. 110-116.

[20] Z.Barzilai, D.Coppersmith and A.Rosenberg "Exhaustive Generation of Bit Pattern with with Application to VLSI Self-Testing" , *IEEE Transactions on Computers*, Vol. C-31, No. 2, 1983, pp. 190-194.

[21] D.T.Tang and C.L.Chen "Iterative Exhaustive Pattern Generation for Logic Testing" , *IBM J. Res. Develop.*, Vol. 28, No. 2, 1984, pp. 212-219.

[22] G.Cohen, M.Karpovsky and L.Levitin "Exhaustive Testing of Circuits with Outputs Depending on Limited Number of Inputs" , *IEEE International Information Workshop* , Caesarea, 1984.

[23] N.J.A.Sloane "Covering Arrays and Intersecting Codes", *J. Combinatorial Design*, Vol. 1, No. 1, 1993, pp. 51-64.

[24] L.B.Levitin and M.G.Karpovsky "Exhaustive Testing of Almost All Devices with Outputs Depending on Limited Number of Inputs", *Open Systems & Information Dynamics*, Vol. 2, No. 3, 1994, pp. 1-16.

[25] G.Cohen, P.Godlewski and M. G. Karpovsky "Exhaustive Testing of Combinatorial Circuits", *Traitement du signal, revue scientifique francaise publiee par le GRETSI*, Vol. 1, No. 2-2, 1984, pp. 224-226.

[26] N.Alon "Explicit Construction of Exponential Sized Families of k-independent sets", *Discrete Math.*, Vol. 58, 1986, pp. 191-193.

[27] P.Busschbach "Constructive Methods to Solve the Problem of: s-surjectivity, conflict resolution, coding in defective memories" *Tech. Rep. 84D005, Ecole Nationale Superieure des Telecom.*, Desember, 1984.

[28] D.T.Tang and C.L.Woo "Exhaustive Test Pattern Generation with Constant Weight Vectors", *IEEE Transactions on Computers*, Vol. C-22, No. 12, 1983, pp. 1145-1150.

[29] J.Savir, W.H.McAnney and S.R.Vecchio "Testing for Coupled Cells in Random-Access Memories" *Proceedings International Test Conference*, Washington, August 1989, pp. 439-451.

[30] W.Bleickardt "Multimoding and Its Suppression in Twisted Ring Counters", *The Bell System Technical Journal*, November, 1968, pp. 2029-2050.

[31] M.G.Karpovsky and V.N.Yarmolik "Transparent Memory BIST", *IEEE International Workshop on Memory Technology Design and Testing*, August 8-9, 1994, San Jose, CA, pp. 106-111.

[32] M.G.Karpovsky, V.N.Yarmolik "Transparent Memory Testing for Pattern Sensitive Faults", *Proceedings International Test Conference*, Washington, October 1994, pp.368-377.

[33] P.H.Bardell, W.H.McAnney, J.Savir "Built-in Test for VLSI: *Pseudorandom Techniques*", John Wiley and Sons, New York, 1987.

[34] J.Savir, W.H.McAnney "A Multiple Seed Linear Feedback Shift Register", *Proceedings International Test Conference*, Washington, September 1990, pp.657-659.

[35] M.Nicolaidis "Efficient UBIST for RAMs", *VLSI Test Symposium*, April 1994, pp.158-166.

[36] M.G.Karpovsky, V.N.Yarmolik, A.J. van de Goor "Pseudoexhaustive Word-Oriented DRAM Testing", *Proceedings European Test Conference*, March 1995.