EXHAUSTIVE AND NEAR-EXHAUSTIVE MEMORY TESTING TECHNIQUES AND THEIR BIST IMPLEMENTATIONS

Debaleena Das and M.G.Karpovsky

Mailing address:

Prof. Mark G. Karpovsky
Department of Electrical, Computer and Systems Engineering
Boston University
44 Cummington Street
Boston, Massachusetts
02215 USA
Tel: (617) 353-9592
Fax: (617) 353-6440
Email: mr@buenga.bu.edu

Key Words:

Memory Testing

Pattern Sensitive Faults

Exhaustive Codes

Built-In-Self Test

EXHAUSTIVE AND NEAR-EXHAUSTIVE MEMORY TESTING TECHNIQUES AND THEIR BIST IMPLEMENTATIONS[1]

Debaleena Das and Mark Karpovsky, Fellow of IEEE

Research Lab. on Design and Testing of Computer Hardware

Boston University

Abstract: In this work we investigate the problem of detection and location of single and unlinked multiple *k out of n pattern sensitive faults* ($PS(n,k)$ fault model) in $n \times 1$ random-access memories (RAMs). This fault model covers all crosstalks between any k cells in $n \times 1$ RAMS. The problem of memory testing has been reduced to the problem of the generation of $(n, k-1)$-exhaustive backgrounds. We have obtained practical test lengths , for a memory size around 1M, for detecting up to 6-couplings by exhaustive tests and up to 9-couplings by near-exhaustive tests. The best known test algorithms up to now provide for the detection of 5-couplings only in a 1M memory, using exhaustive tests. Beyond these parameters, test lengths were impractical. Furthermore, our method for generation of $(n, k-1)$- exhaustive backgrounds yields short test lengths giving rise to considerably shorter testing times than the present most efficient tests for large n and k greater than 3. The systematic nature of both our tests enables us to use a *built-in self-test* (BIST) scheme, for RAMs, with low hardware overhead.

Key words: Memory testing, pattern sensitive faults, built-in self-test,
exhaustive codes, near-exhaustive codes.

Symbols and notations

The following notations are used in this text to denote faults :

- $\uparrow$ denotes a write 1 operation to a cell containing a 0;
- $\downarrow$ denotes a write 0 operation to a cell containing a 1;
- $\updownarrow$ denotes a write $\bar{x}$ operation to a cell containning a x, x $\in (0,1)$;

[1] This work was supported by the NSF under Grant MIP 9630096 and the NATO under Grant 910411

- $< I/F >$ denotes a fault in a single cell, I desribes the condition for sensitizing the fault and F describes the value of the faulty cell.

1 Introduction

In this work we describe a new method for *Random Access Memory* (RAM) testing based on the *PS(n,k)* fault model. This model includes most classical fault models for SRAMs (Static RAMs) and DRAMs (Dynamic RAMs).

The problem of memory testing for *PS(n,k)* faults has been reduced to the problem of the generation of $(n, k-1)$-exhaustive backgrounds, where n is the size of the memory under consideration and $k-1$ is a parameter from the fault model of the memory. A background is defined as a memory state that is loaded into the memory.

A new method for constructing the background matrix $B(n, k-1)$ with lesser mumber of rows (as compared to existing constructions) for n of the order of a million, i.e. the size of commonly used RAM chips, is described. . We also propose a construction for the near-exhaustive background matrix $B(n, k-1, \epsilon)$ which can be applied for detection of all $PS(n, k)$ faults with a probability of at least $1 - \epsilon$.

1.1 Memory modeling and Functional Faults

A functional model of a RAM, typically used for functional testing, is the 3-block model. The blocks are : address decoder, memory cell array and the read-write logic.
This fault model gives rise to the following classes of faults: [1]

1. **Faults in which a single cell is involved:** These are the stuck-at faults (SFs) and transition faults (TFs). Transition faults cannot occur in DRAMs because the cells are not implemented as bi-stable elements.

2. **Faults in which two cells are involved:** These are the coupling faults.

3. **Faults involving k cells:** These are considered typical for DRAM.

(a) **Neighborhood Pattern Sensitive Faults (NPSFs)**[1]: The k cells are clustered together in a physical neighborhood. There are three types of NPSFs: Active, Passive and Static (ANPSFs, PNPSFs and SNPSFs).

(b) **k-coupling fault :** The k cells are allowed to be located anywhere in memory. The k-coupling fault model is a generalization of the coupling fault mentioned earlier. A set of k cells is said to be *k-coupled* when a write operation to one cell ($\uparrow$ *or* $\downarrow$) produces a change in the contents of a second cell, subject to a particular data pattern in the remaining k-2 cells, which may be anywhere in the memory. k-coupling faults cover ANPSFs and SNPSFs (neighborhood is not fixed).

1.2 Testing Strategies for RAMs

Due to miniaturization of RAMs, the types of faults have become more complex and the emphasis now is on the detection of *pattern sensitive faults* (PSFs). PSFs were first studied by Hayes [2] and he observed that to obtain test sequences of practical length, we must greatly restrict the set of PSFs under consideration. Though it is practically justified to restrict the error behaviour associated with a particular cell to a well-defined neighborhood of the cell, it is not easy to specify the neighborhood, since the scrambling table may not be available to the designer. (The scrambling table describes the relationships between the logical addresses and physical addresses of the memory cells.)

Thus, there is the need for efficient tests to detect NPSFs regardless of the address mapping. The first step in this direction was the formulation of the *k-coupling fault model* by Nair et al [3]. The 5-coupling fault model covers active NPSFs with a Type I neighborhood [4] and the 9-coupling fault covers active NPSFs with Type II neighborhood [4]. Tests for 5/9-couplings cover active NPSFs and have to be used when the scrambling table is not known. The latest results in deterministic tests for the detection of single k-coupling faults in RAMs are by Cockburn and can be found in [5]. All the above tests were for bit-oriented memories. Extension to word-oriented memory has been done in [6].

Probabilistic tests for detecting *k-coupling faults* were proposed by Cockburn [8]. He suggested that effective probabilistic tests can be obtained by using random $n \times m$ matrices, where $m \gg 1$. Savir *et al.* [7] have proposed two test strategies to detect coupled cells in a n word by 1 bit RAM. In both strategies the data-in line is randomly driven. One of the strategies uses random selection of both the address lines and the read/write control. The other strategy sequentially cycles through the address space with deterministic setting of the read/write control. Probabilistic tests for coupling faults have also been described in [16] [17].

1.3 Proposed Fault Model and Main Results

The fault model considered in this work is the k out of n pattern sensitive fault model $PS(n, k)$, where n is the size of the memory under test. The $PS(n, k)$ fault model is one of the most general fault models which encompasses all the functional faults mentioned. According to this model, the contents or the ability to change the contents of any memory cell in a n-bit memory is influenced by the contents or change in the contents of any other $k - 1$ cells of the memory.

A hierarchy of the functional faults covered by the $PS(n, k)$ fault model is shown in Figure 1.

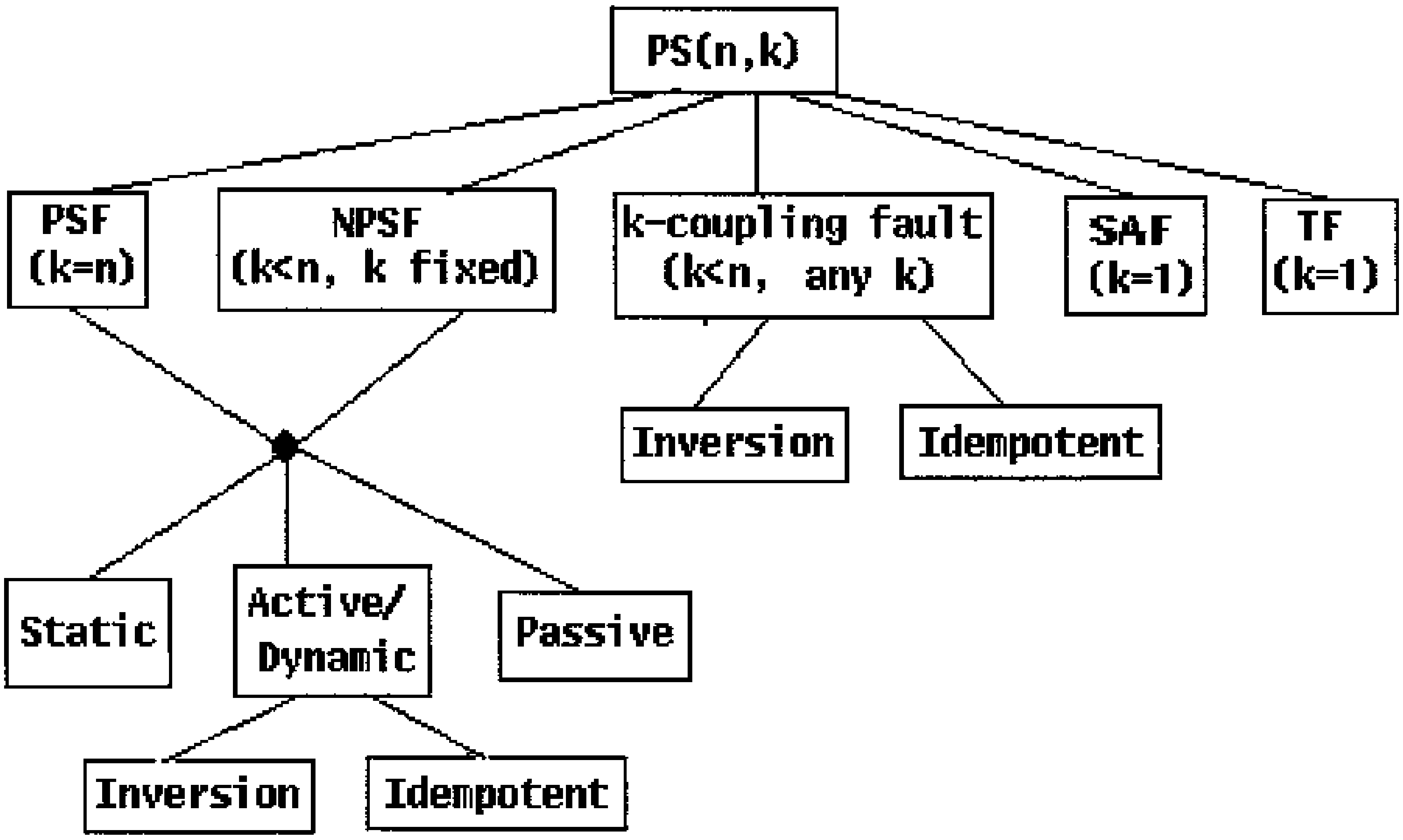

Figure 1: Hierarchy of Functional Faults Covered by PS(n,k) Fault Model

For $k < n$ and any k , we have the k-coupling faults which cover all active NPSFs (value of k depends on the size of neighborhood). TFs and SFs are covered by this model for k=1. We note that the $PS(n,k)$ fault model is slightly more general than the k-coupling model, since it includes the case when the memory consists of (possibly overlapping) blocks of size n (in this case n is smaller than the total size of the RAM) and crosstalks can occur only between k cells within the same block. This case was considered in [9]. In this paper we do not impose any limitations on the locations of the cells involved in crosstalks and n is equal to the total number of cells in the RAM.

This model covers all active and passive NPSFs (k=5 will cover Type I neighborhood and k=9 will cover Type II neighborhood). We construct both exhaustive and near-exhaustive tests for the detection of single k-coupling faults, using $(n, k-1)$-exhaustive backgrounds [5]. We have obtained practical test lengths, for a memory size around 1M, for detecting up to 6-couplings by exhaustive tests and 9-couplings by near-exhaustive tests. The best known results till date (by Cockburn [8]) is the detection of 5-couplings in a 1M memory, using exhaustive tests. Cockburn's algorithm is applicable for memories of any size. However, beyond these parameters, the test lengths may not be practical.

Furthermore, our method for the generation of (n, k)-exhaustive backgrounds yields much shorter tests than the tests generated by Tang and Chen [11] and used by Cockburn [5]. As an example, our test length based on (1M, 4)-exhaustive backgrounds is 50% shorter than the test used in [5].

We have also obtained practical test lengths, for a memory size around 1M to cover up to 9-couplings using near-exhaustive tests. These tests have a fault coverage very close to one.

2 Exhaustive Tests

For the k-coupling fault model, each cell can be affected by a crosstalk with at most $k-1$ other cells located anywhere in the memory. It will be shown in this chapter that the problem of testing such memories can be reduced to the generation of (n,$k-1$) exhaustive backgrounds, where n is the size of the memory and k is the size of the neighborhood . The k cells which form the neighborhood can be anywhere in the memory. A background is defined as a memory state that is loaded into the memory.

Definition 2.2.1 We define an $T_{n,k} \times n$ matrix $B(n,k)$ over the set {0,1}. Let B_k be any $T_{n,k} \times k$ submatrix of $B(n,k)$. $B(n,k)$ has the property that all B_k submatrices contain all the possible 2^k binary row vectors. $T_{n,k}$ is the smallest number of rows required to achieve this property.

The rows of $B(n,k)$ form (n,k)-exhaustive backgrounds.

Definition 2.2.3 We also define a matrix $B^1(n,k)$ such that:

$$\text{ith row of } B^1(n,k) = \text{ith row of } B(n,k) \oplus \text{(i - 1)th row of } B(n,k) \; i \in (1,2,\ldots,T_{n,k}-1)$$
$$\text{The 0th row of } B^1(n,k) = \text{0th row of } B(n,k).$$

Example 2.2.1 Consider n=3 and k=3. The corresponding $(n,k-1)$-exhaustive backgrounds is the matrix $B(3,2)$. $B(3,2)$ ans $B^1(3,2)$ are shown below:

$$B(3,2) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \quad B^1(3,2) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Consider the following testing algorithm.

- The test backgrounds used are the rows of a $B(n,k-1)$ matrix which form $(n,k-1)$-exhaustive backgrounds.

- For each background, perform the MARCH test. MARCH test is defined as a sequence of operations to be performed on each cell. In our test, the sequence of operations is

Read, $\updownarrow$, Read, $\updownarrow$. Let the addressing of the cells be in increasing order.

- The ith background is loaded as the modulo-two sum of the present background after the MARCH test and the ith row of $B^1(n, k-1)$, i $\in$ (1,2, ... , $T_{n,k-1}$). The all zero background is the 0th row of $B^1(n, k-1)$. (This is equivalent to writing the ith row of $B(n, k-1)$ if there is no distortion of data in the memory after the MARCH elements).
- After the MARCH sequence on the last background, each cell is read.

Example 2.2.1 contd. For the matrices $B(3, 2)$ ans $B^1(3, 2)$ the operations constituting the test are :

Load the first background : 000	MARCH test for first background
Load the second background : 000$\oplus$011	MARCH test for second background
Load the third background : 011$\oplus$110	MARCH test for third background
Load the fourth background : 101$\oplus$011	MARCH test for fourth background

Final Read Sequence for all bits.

	Backgrounds			
MARCH elements	000	011	101	110
$r(a_0)$	$\underline{0}00$	$\underline{0}11$	$\underline{1}01$	$\underline{1}10$
$w(\bar{a}_0)$	$\underline{1}00$	$\underline{1}11$	$\underline{0}01$	$\underline{0}10$
$r(\bar{a}_0)$	$\underline{1}00$	$\underline{1}11$	$\underline{0}01$	$\underline{0}10$
$w(\bar{a}_0)$	$\underline{0}00$	$\underline{0}11$	$\underline{1}01$	$\underline{1}10$
$r(a_1)$	$0\underline{0}0$	$0\underline{1}1$	$1\underline{0}1$	$1\underline{1}0$
$w(\bar{a}_1)$	$0\underline{1}0$	$0\underline{0}1$	$1\underline{1}1$	$1\underline{0}0$
$r(\bar{a}_1)$	$0\underline{1}0$	$0\underline{0}1$	$1\underline{1}1$	$1\underline{0}0$
$w(\bar{a}_1)$	$0\underline{0}0$	$0\underline{1}1$	$1\underline{0}1$	$1\underline{1}0$
$r(a_2)$	$00\underline{0}$	$01\underline{1}$	$10\underline{1}$	$11\underline{0}$
$w(\bar{a}_2)$	$00\underline{1}$	$01\underline{0}$	$10\underline{0}$	$11\underline{1}$
$r(\bar{a}_2)$	$00\underline{1}$	$01\underline{0}$	$10\underline{0}$	$11\underline{1}$
$w(\bar{a}_2)$	$00\underline{0}$	$01\underline{1}$	$10\underline{1}$	$11\underline{0}$

Table 1: Exhaustive Memory Testing Using B(3,2) And B^1(3,2)

a_0, a_1 and a_2 are the cells of the memory

(The cells which are written or read at every step are underlined)

Proposition 2.2.1 All NPSFs (Active, Passive and Static) with the size of the neighborhood equal to k and all k-couplings will be detected by the test algorithm defined.

Proof : Consider the following k-tuple of cells:

We define cells i and j to be the cells affected by crosstalk. Cell i is the cell which evokes the fault and cell j is the base cell which gets affected, if there is a fault. We also define the contents of the remaining $(k-2)$ cells of the neighborhood, which is necessary to evoke the fault, as the binary pattern P.

k-couplings : Since the test backgrounds are the rows of a $B(n, k-1)$ matrix, there will be at least one background in which the $k-2$ cells under consideration will contain P, and the cell j contains either 0 (to evoke a $\uparrow$ fault) or 1 (to evoke a $\downarrow$ fault). When we perform Write operations on cell i, we are at background which evokes the crosstalk, so either the $\uparrow$ transition or the $\downarrow$ transition will distort the contents of cell j.

Two cases can occur: If operations on cell j are done after cell i, the fault will be detected during the MARCH element of the same background when cell j is read. On the other hand, if operations on cell i come later, the fault will not be detected during that MARCH test. However, the method of loading the next background will propogate the error to the next background and it will be detected during the next MARCH test.

The proof is similar for ANPSFs since k-couplings cover ANPSFs. We note that for the detection of *inversion ANPSF*, it is sufficient to use a $B(n, k-2)$ matrix. Write operations to cell i will erroneously invert the contents of cell j, irrespective of its contents, when the remaining k-2 cells contain P.

PNPSFs: Either of the two $\updownarrow$ operations of the MARCH element will provoke the fault. Detection of the fault is similar to k-couplings.

Tests for *ANPSFs* and *PNPSFs* cover all *SNPSFs* [1]. Thus the proof is complete for all *NPSFs.*

We define "test length" to be the minimum number of Read/Write operations performed in a test.

Proposition 2.2.2 There exists a test, detecting all single and unlinked multiple $PS(n,k)$ faults in a n bit memory, with the test length = n($5T_{n,k-1}$ + 1). For large memories, the number of backgrounds ($T_{n,k-1}$) $\gg 1$. Hence, the minimum test length $\approx$ 5n$T_{n,k-1}$.

Proof: For the test constituitng of MARCH tests for each row of $(n,k-1)$-exhaustive backgrounds, we have:

The number of operations during the MARCH test is equal to 4 per cell per background.

The number of operations for loading a background is equal to 1 per cell per background.

The number of operations for the final Read is equal to 1 per cell.

The total number of operations is equal to $4nT_{n,k-1} + nT_{n,k-1} + n$.

In the matrix $B^1(n,k-1)$, a matrix element of "1" denotes that the cell changes state during the loading of the background whereas "0" denotes that the cell remains in the same state. It follows from the results presented in [13] that if $T(n,k-1) \geq log_2 n$, the backgrounds can be arranged in such a way that the number of changes in data from one background to another (ie the number of 1s in a row) is upper bounded by $\lceil n/2 \rceil$. The number of operations for loading a background will be reduced from $nT_{n,k-1}$ to $\lceil n/2 \rceil T_{n,k-1}$ if we perform the Write operation only for those cells which change states. Thus, the test length can be reduced to $9\lceil n/2 \rceil T_{n,k-1}$, if during the loading of a background, only those cells which change states are written.

2.1 Construction of (n,k)-Exhaustive Backgrounds

In this section, we will introduce a construction for (n,k)-exhaustive background matrices, that has fewer rows than published constructions, for large values of n. Several constructions for (n,k)-exhaustive backgrounds and estimations on the number of rows, $T_{n,k}$, can be found in [20], [21], [22] where these matrices are refered to as k-surjective arrays or k-independent sets.

We will first construct a matrix M whose property is defined in *Lemma 2.1.1*. *Lemma 2.1.2* deals with the construction of (n, k)-exhaustive background matrices using M.

Definition 2.1.1 Let $P_0, P_1, \ldots, P_{R-1}$ be different prime numbers with $P_0 < P_1 < \ldots < P_{R-1}$. We define an $R \times n$ matrix $M = (m_{ij})$, such that $m_{ij} = j \pmod{P_i}$. P_i is the prime number associated with row i.

We define M_k to be any $R \times k$ $(k \leq n)$ submatrix of M.

Definition 2.1.2: Consider any k columns of $M : j_1, j_2, \ldots, j_k$.

$0 \leq j_1 < j_2 \ldots < j_k \leq n - 1$

We define: $\triangle(j_1, \ldots, j_k) = \prod_{p>q}(j_p - j_q)$ where $p, q = (1, 2, \ldots, k)$

Definition 2.1.3: $\triangle(n, k) = max_{j_1,\ldots,j_k} \triangle (j_1, \ldots, j_k)$

Remark 2.1.1: Consider an ordered set of l primes:

$P = (P_0, P_1, \ldots, P_{l-1})$

Let X_m denote the ordered set of residues of a number "m" evaluated on P.

$X_m = (x_0, x_1, \ldots, x_{l-1})$ i.e. $x_i = m \bmod P_i$.

By the *Chinese Remainder Theorem*, X_m will be unique for any $m \in (0, \ldots, L-1)$ if $(\prod_{i=o}^{l-1} P_i > L)$.

Example 2.1.1: Consider $L = 5$. $P =$ (2,3) satisfies $(\prod_i P_i > L)$ then $X_0, \ldots, X_4$ should be distinct. This is illustrated below.

$X_0 \Leftrightarrow$ (0,0) $X_1 \Leftrightarrow$ (1,1) $X_2 \Leftrightarrow$ (0,2) $X_3 \Leftrightarrow$ (1,0) $X_4 \Leftrightarrow$ (0,1)

Lemma 2.1.1: All submatrices M_k of M will have at least one row in which all the matrix elements are distinct if $\prod_{i=0}^{R-1} P_i > \triangle(n, k)$.

Proof by contradiction: Let us assume that there exists a submatrix M_k with k columns in which there is no row where all the matrix elements are distinct. Let the columns in this matrix be $j_1, j_2, \ldots, j_k$. For any row i of M_k, there will be a repetition of matrix elements in at least one pair of columns. The difference between these pair of columns must be a multiple of P_i by the definition of M. Therefore, the residue of $\triangle(j_1, j_2, \ldots, j_k)$ evaluated on P_i will be zero. This result holds for all i.

$\Rightarrow X_{\triangle(j_1,j_2,\ldots,j_k)} = (0, 0, \ldots, 0)$.

However, X_0 is also equal to $(0, 0, \ldots, 0)$.

By Remark 2.1.1, since ($\prod_{i=0}^{R-1} P_i > \Delta(n,k)$), all numbers from 0 to ($\Delta(n,k)$ -1 should have had unique X_ms. Hence, there is a contradiction.

We note that the inequality $\prod_{i=0}^{R-1} P_i > \Delta(n,k)$ may be strengthened to $\prod_{i=0}^{R-1} P_i > C(k) \cdot \Delta(n,k)$; where $C(k) = C_2(k)C_3(k)\ldots C_p(k)$; p is the largest prime number lesser than P_0 and C_i is a proper fraction resulting from the absence of the prime number P_i in the product of the primes. As an example when the prime number 2 is not present in the product of the primes, the factors in $\Delta(n,k)$ cannot be 2 or a multiple of 2. The analytical solutions for $C_2(k)$ and $C_3(k)$ are given below :

$$C_2(k) = \begin{cases} 2^{\frac{-k(k-2)}{4}} & \text{for k even} \\ 2^{\frac{-(k-1)^2}{4}} & \text{for k odd,} \end{cases} \qquad 5$$

$$C_3(k) = 3^{-(k-3)}. \qquad 6$$

Remark 2.1.2 We have the following approximations for $\Delta(n,k)$ for k=3,4,5. The approximations tend to equality for n of the order of thousands.

- k=3 $\Rightarrow \Delta(n,3) \approx \frac{(n-1)^3}{4}$
- k=4 $\Rightarrow \Delta(n,4) \approx 0.018(n-1)^6$
- k=5 $\Rightarrow \Delta(n,5) \approx 3.3\text{x}10^{-4}(n-1)^{10}$

These values have been obtained by maximization using partial differentiation. These values hold for ns of the order of thousands and are thus applicable here.

Example 2.3.1: Consider n=10 and k=3.

$\Rightarrow \Delta(10,3) = \frac{(n-1)^3}{4} = 182.25$

$C(k) = C_2(k) = 2^{-1}$

$\Rightarrow \prod_{i=0}^{R-1} P_i > 2^{-1} \times 182.25, P_0 = 3$

$\Rightarrow R = 3, P_{R-1} = 7$

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 \\ 0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 0 & 1 & 2 \end{bmatrix}$$

The following Lemma gives the construction of $B(n,k)$ matrix from M.

Lemma 2.3.2 If every matrix element m_{ij} of M is replaced by the *jth* column of a (P_i, k)-exhaustive matrix $B(P_i, k)$, then the resulting matrix is a (n,k)-exhaustive matrix, $B(n,k)$. The number of rows of $B(n,k)$, $T_{n,k}$ is equal to $\sum_{i=0}^{R-1} T_{P_i,k}$, where $T_{P_i,k}$ is the number of rows in $B(P_i, k)$.

We will call matrices $B(P_i, k)$ $(i = 0, \ldots, R-1)$ as seed matrices for $B(n,k)$, where $\prod_{i=0}^{R-1} P_i > C(k) \bigtriangleup (n,k)$.

2.2 Estimation of testing times

The test times for the tests corresponding to memories of size 64K, 256K, 1M, 4M and 16M are computed assuming a 10 MHz test vector application rate. In order to compare the test times with Cockburn [5], we compute the test times for tests which detect active faults only. For the active faults, the second Read operation in the MARCH element is not necessary. Number of operations during the MARCH element is thus reduced to 3 per cell per background. The resulting test length is $7\lceil\frac{n}{2}\rceil T_{n,k-1}$.

The following tables give the test lengths for different memory sizes, n and different values of k. $Time_c$ are the test times given by Cockburn [5] and are given for comparison. $Time_c$ for k=3 corresponds to the test times of the S4CTEST and $Time_c$ for k=4 corresponds to the test times of the S5CTEST [5].

n	$c(3) \bigtriangleup (n,3)$	R	P_{R-1}	$T_{n,3}$	$Time$	$Time_c$	$\frac{Time-Time_c}{Time}\%$
64k	3.27×10^{13}	12	41	219	4.9secs	-	-
256k	2.1×10^{16}	13	43	246	22.05secs	22.9secs	3.9
1M	1.25×10^{17}	14	47	277	1.62mins	1.9mins	17.3
4M	8×10^{18}	15	53	308	7.17mins	8.34mins	16.3
16M	5.12×10^{20}	16	59	340	31.68mins	33.36mins	5.3

Table 2: Testing Times for 4-couplings

n	$c(4) \triangle (n,4)$	R	P_{R-1}	$T_{n,4}$	$Time$	$Time_c$	$\frac{Time-Time_c}{Time}\%$
64k	1.03×10^{26}	19	73	1594	35.7secs	-	-
256k	4.22×10^{29}	20	79	1704	2.54mins	3.28mins	29.0
1M	1.5×10^{33}	22	89	1924	11.23mins	19.24mins	71.3
4M	6.14×10^{36}	24	101	2144	50.02mins	1.28hrs	53.5
16M	2.52×10^{40}	26	107	2364	3.67hrs	5.13hrs	40.0

Table 3: Testing Times for 5-couplings

n	$C(5) \triangle (n,5)$	R	P_{R-1}	$T_{n,5}$	$Time$
64k	2.64×10^{42}	27	109	11875	5mins
256k	2.77×10^{42}	30	131	14038	23.9mins
1M	2.3×10^{54}	32	139	15494	1.6hrs
4M	2.4×10^{60}	35	157	17678	7.85hrs
16M	2.52×10^{66}	38	173	20486	-

Table 4: Testing Times for 6-couplings

As one can see from Tables 2 and 3, for a 1M RAM our approach results in a 17.3% savings in testing times for detection of 4-couplings and 71.3% for 5-couplings. Detections of 6-couplings in a 1M RAM with our approach requires 1.6 hours. In Tables 2,3 and 4, we have assumed the access time for Read and Write operations to be 100 nsecs. Only these two operations have been counted in the evaluation of test times.

The Tables 2 and 3 show that the saving in time peaks around 1M. The peak results from the non-linear nature of our algorithm. Even beyond 16M, the benefits of our algorithm will not fall off.

3 Near-Exhaustive Probabilistic Tests

3.1 Survey of Probabilistic Tests

It has been shown in Section 2 that the problem of testing of memories, of size n and each cell being affected by crosstalks with at most $k-1$ other cells of the memory, can be reduced to the generation of (n,$k-1$)-exhaustive backgrounds. In this section we consider ϵ-exhaustive tests. ϵ-Exhaustive tests are those tests which will detect all $PS(n,k)$ faults with a probability $\geq (1-\epsilon)$. For ϵ-exhaustive tests, we will construct $(n,k-1,\epsilon)$-exhaustive backgrounds instead of (n,k)-exhaustive backgrounds.

$(n,k-1,\epsilon)$-exhaustive backgrounds is the matrix $B(n,k-1,\epsilon)$. $B(n,k-1,\epsilon)$ has the property that the fraction of $(k-1)$-tuple of columns, not containing all the possible 2^{k-1} binary row vectors, does not exceed ϵ. Thus $B(n,k-1,0) = B(n,k-1)$. Let us define B_{k-1} to be any submatrix of $B(n,k-1,\epsilon)$ having the same number of rows as $B(n,k-1,\epsilon)$ and number of columns equal to $(k-1)$. Henceforth, we shall refer to a submatrix B_{k-1} containing all possible 2^{k-1} binary row vectors as a "good" B_{k-1}. All other B_{k-1} submatrices will be refered to as a "bad" B_{k-1}.

It will be show below that the number of test backgrounds can be reduced substantially if we allow a small fraction ϵ of all the possible combinations of k columns not to be tested exhaustively. This effect holds even if ϵ approaches zero with the increase in n [14].

Chandra *et al.* considered the effectiveness of probabilistic methods for finding (n,k)-exhaustive codes for k $\geq$ 3 [15]. Cockburn [8] suggested that effective probabilistic tests for detecting *k-coupling faults* can be obtained by basing the tests on random $n \times m$ matrices, where $m \gg 1$. Other publications dealing with probabilistic tests for coupling faults are [16] [17] .

3.2 Construction of ϵ-Exhaustive tests

Let $H(2^s)$ be a matrix formed by the code words of a $(2^s, s)$-simplex code as the rows, where $s \geq k$ [18]. This is the Hadamard matrix over {0,1}. The columns of the matrix $H(2^s)$ form a s-dimensional linear space, with the vector of all zeroes deleted.

Proposition 3.1 For any $s \geq k$ and any $a < n$, there exists an ϵ-exhaustive test with $T_{n,k,\epsilon}$ backgrounds where $n = a2^s$, $T_{n,k,\epsilon} = 2^{s+1}$, $(1 - \epsilon) = (1 - \epsilon_v)(1 - \epsilon_h)$, $\epsilon_v = 1 - (2^s \prod_{i=0}^{k-2}(2^s - 2^i)(\prod_{i=0}^{k}(2^s - i))^{-1}$ and $\epsilon_h = 1 - a^k \begin{pmatrix} 2^s \\ k \end{pmatrix} (\begin{pmatrix} a2^s \\ k \end{pmatrix})^{-1}$.

The proof is given in [14].

One can use the following construction for the background matrix $B(n, k, \epsilon)$:

$$B(n,k,\epsilon) = \begin{bmatrix} \underbrace{\hat{H}(s) \quad \ldots \quad \hat{H}(s)}_{a} \end{bmatrix}$$

where, $\hat{H}(s) = \begin{bmatrix} H(s) \\ \bar{H}(s) \end{bmatrix}$ and $\bar{H}(s)$ is the negation of $H(s)$.

Fractions of good submatrices $B_k s$ i.e $(1 - \epsilon)$ for $B(n, k, \epsilon)$, constructed by Proposition 3.1, are given in Table 5.

ks	11	12	13	14
4	0.997	0.999	0.999	1.000
5	0.993	0.997	0.998	0.999
6	0.985	0.992	0.996	0.998
7	0.970	0.985	0.993	0.996
8	0.939	0.969	0.985	0.992
9	0.881	0.931	0.969	0.985

Table 5: $(1 - \epsilon)$ for $n = 2^{18}$ to $n = 2^{24}$

The number of rows of $B(n, k-1, \epsilon)$ is $= T_{n,k-1,\epsilon} = 2^{s+1}$ and similar to (1), the test length is $9\lceil n/2 \rceil T_{n,k-1}$.

Test lengths and testing times for detection of $PS(n, k)$ faults by MARCH Tests on $(n, k-1, \epsilon)$-exhaustive backgrounds for $n = 2^{18}$ to $n = 2^{24}$, for k-couplings with $k = 5, \ldots, 9$ and a probability of detection of at least 99% are given in Tables 6 and 7. We also present in Tables 6 and 7 test lengths and testing times for the case when $\hat{H}(s) = \begin{bmatrix} H(s) \\ \bar{H}(s) \end{bmatrix}$ is replaced by $\hat{H}_R(s) = \begin{bmatrix} H(s) \\ H_R(s) \end{bmatrix}$ where $H_R(s)$ is obtained from $H(s)$ by a random permutation of its columns. This approach results in the decrease of testing times but is rather difficult to implement for built-in self-testing of RAMS. In Tables 6 and 7, we assume an access time of 100ns.

k	5	6	7	8	9
Test Lengths	2.4×10^9	4.8×10^9	9.6×10^9	1.9×10^{10}	3.8×10^{10}
Test Lengths(random)	1.2×10^9	2.4×10^9	4.8×10^9	4.8×10^9	9.6×10^9
Test Times	4mins	8mins	16mins	32mins	1.06hrs
Test Times(random)	2mins	4mins	8mins	8mins	16mins

Table 6: Testing Times for Detection of PS(n,k) Faults with $\epsilon = 0.01$ and $n = 2^{18}$.

Test Lengths are calculated from Table 5. Test Lenghts(random) are the lengths obtained using $\hat{H}_R(s)$.

k	5	6	7	8	9
Test Lengths	9.6×10^9	1.9×10^{10}	3.8×10^{10}	7.6×10^{10}	1.5×10^{11}
Test Lengths(random)	4.8×10^9	9.6×10^9	1.9×10^{10}	1.9×10^{10}	3.8×10^{10}
Test Times	16mins	32mins	1.06hrs	2.11hrs	4.17hrs
Test Times(random)	8mins	16mins	32mins	32mins	1.06hrs

Table 7: Testing Times for Detection of PS(n,k) Faults with $\epsilon = 0.01$ and $n = 2^{20}$.

Test Lengths are calculated from Table 5. Test Lenghts(random) are the lengths obtained using $\hat{H}_R(s)$.

One can see from Tables 6 and 7 that all PS(n,k) faults can be detected in a reasonable time for $n \leq 2^{20}$ and $k \leq 9$ with a probability of at least 99%.

How bad is a bad B_k (k-tuple of columns of $B(n, k, \epsilon)$)? A bad B_k does not have all the 2^k binary row vectors, however it does have some of the k-bit binary row vectors and thus will contribute to the fault coverage. The results of the study on the vertical concatenation of two Hadamard matrices using negation (see Proposition 3.1) are shown below :

For two $2^{13} \times 2^{13}$ Hadamard matrices (s=13) and (k=6) we have:
Number of B_6 submatrices = 900,000, Number of good B_6 = 893,128,
Number of bad B_6 = 6,872.
216 of the bad B_6 have 16 out of the 64 possible combinations. 6,656 of the bad B_6 have 32 out of the 64 possible combinations.
Therefore, $1 - \epsilon_v = 0.992 + \frac{0.007}{2} + \frac{0.0024}{4} = 0.996$

For two $2^{13} \times 2^{13}$ Hadamard matrices (s=13) and (k=8) we have:
Number of B_8 submatrices = 900,000, Number of good B_8 = 886,257,
Number of bad B_8 = 13,743.
357 of the bad B_8s have 32 out of the 256 possible combinations. 4308 of the bad B_8s have 64 out of the 256 possible combinations. 9078 of the bad B_8s have 128 out of the 256 possible combinations.
Therefore, $1 - \epsilon_v = 0.985 + \frac{0.01}{2} + \frac{0.005}{4} + \frac{0.0004}{8} = 0.991$

For the simulation for k=6 and 8, random submatrices B_k are chosen since the total number of possible submatrices is too big. The results show that if the contribution of the bad $B_k s$ is included in the calculation of $(1 - \epsilon_v)$, as $\frac{\text{number of combinations in bad } B_k}{2^k} \times \frac{\text{number of bad } B_k}{\text{number of total } B_k}$, then the fraction $1 - \epsilon_v$ increases considerably.

4 Design of BISTed RAMs

RAMs are often embedded in large integrated circuits which makes it difficult to access the RAM inputs and outputs for testing purposes. The solution is to use a **built-in-self-test (BIST)** scheme. **BIST** also has other advantages like the reduction of expensive external test equipment and the possibilities of *at-speed* testing [19] [1]. In this section, we will discuss **BIST** implementation for the exhaustive tests described in Section 2 and the near-exhaustive tests described in Section 3.

4.1 System Architecture for BIST implementation of Exhaustive Tests

The architecture of the proposed **BIST** RAM scheme is given in *Figure 2.* The $n \times 1$ random access memory to be tested is the block labeled - Memory Under Test. Multiplexers are used to isolate the RAM from the external environment during the execution of the self-test. The internal details of the test generator are shown in *Figure 3.*

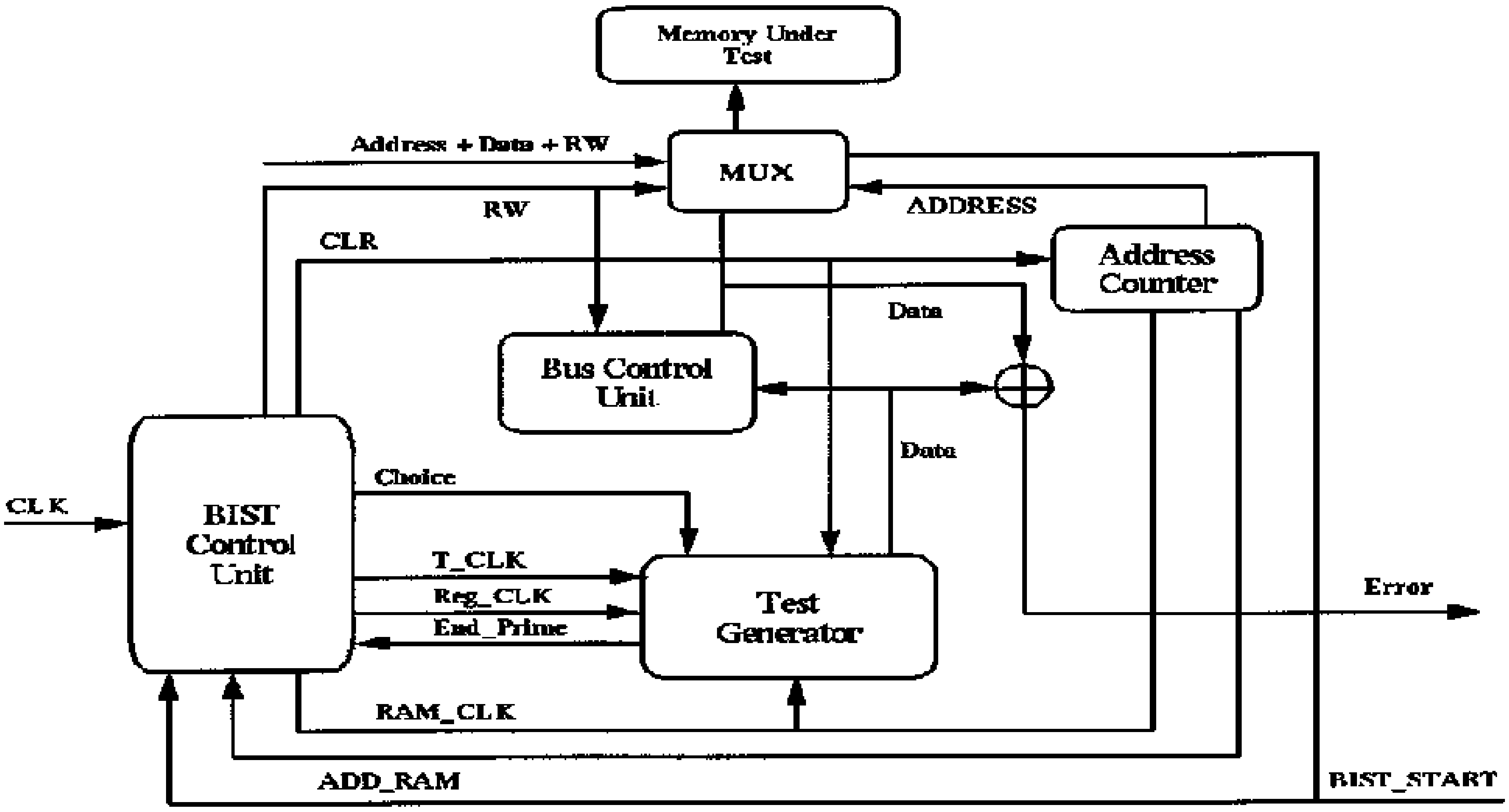

Figure 2: BIST Architecture for Exhaustive Testing

The location of the cell (or cells) which is faulty is given by the address generated by the Address Counter, for the RAM under test, at the time when the Error signal is set to one. Each time the Error signal is set to one, the address of the faulty cell can be stored and thus a black list of faulty cells generated. Once the faulty cells are located, the RAM can be reconfigured to eliminate the faulty cells.

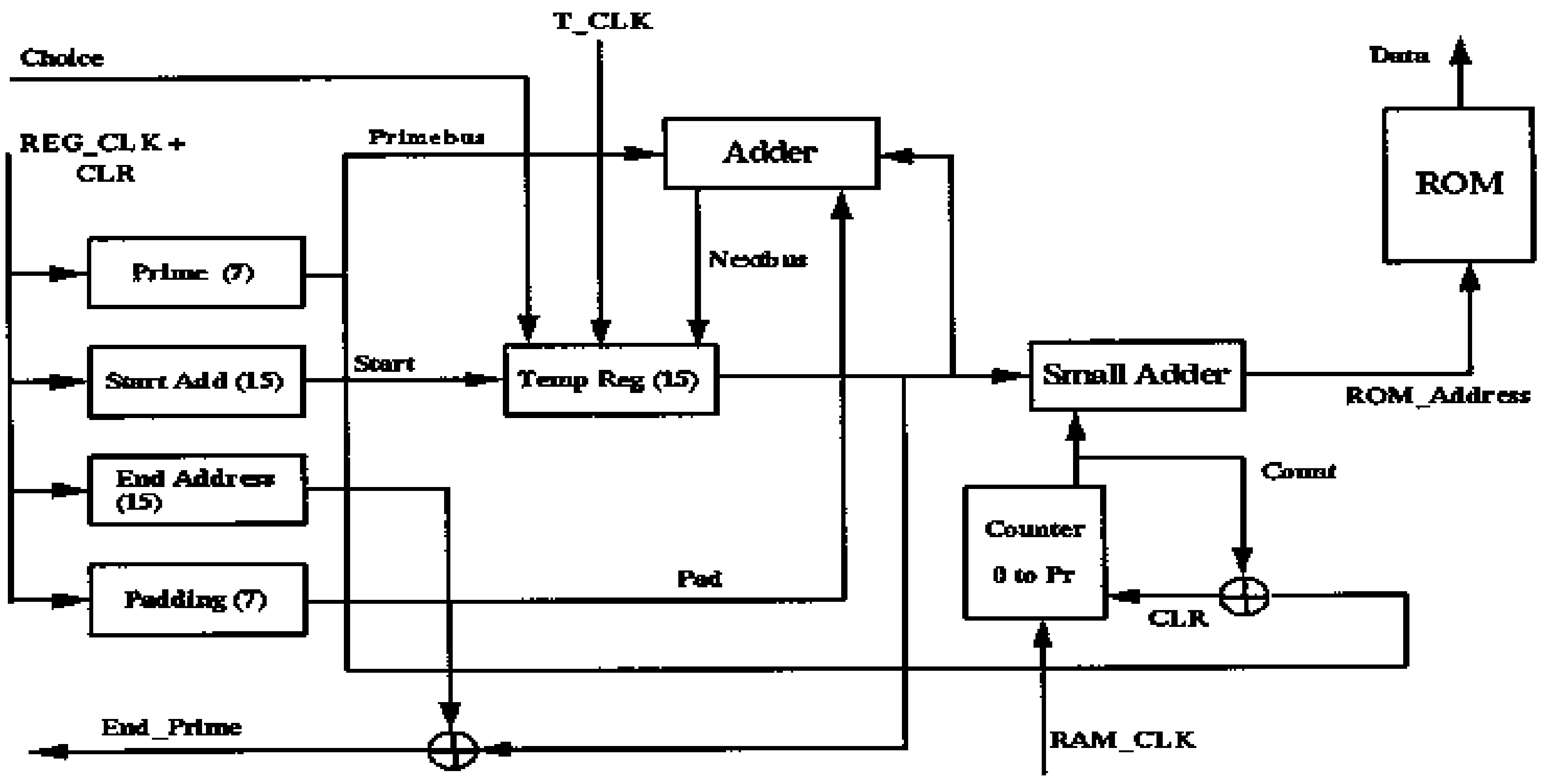

Figure 3: Test Generator for Exhaustive Testing

The *BIST* controller controls the sequence of the operations involved in the **BIST** using the internal status signals. This design has been done for ANPSFs and SNPSFs only. To cover PNPSFs as well, the only modification needed is an extra state between S_8 and S_9 which will correspond to the second Read operation in the MARCH element. The controller behaviour is illustrated in detail in *Figure 4*.

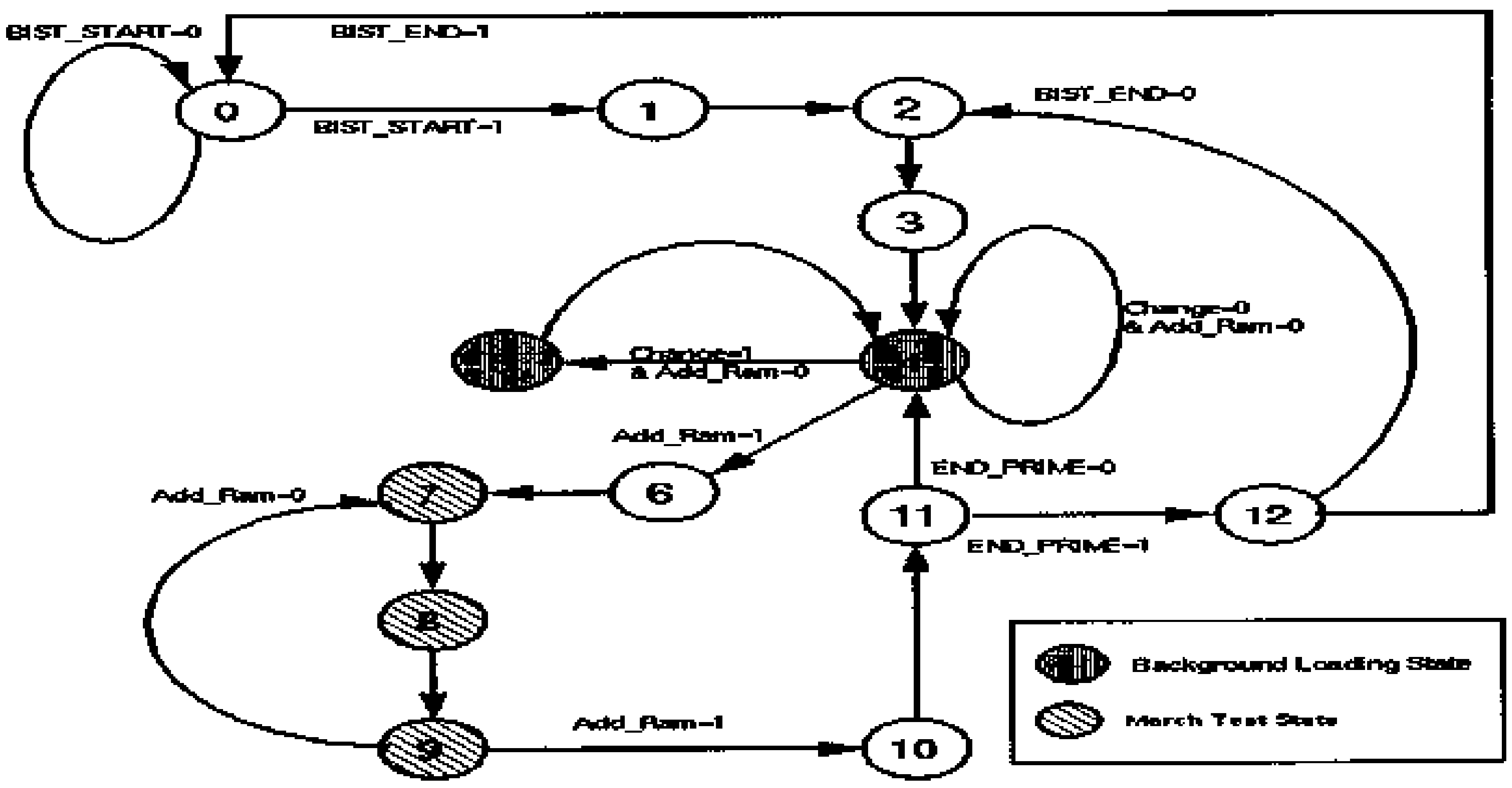

Figure 4: Controller State Diagram for Exhaustive Testing

4.2 System Arcitecture for BIST Implementation of Near-Exhaustive Tests

The BIST scheme using the *Hadamard matrix* is much simpler than the previous scheme. The sub-systems ROM and Address Generator for the ROM are not needed as there is no storage of seeds. Instead, there has to be a sub-system to generate the Hadamard matrix on the fly. One simple approach for the generation of the Hadamard matrix is based on the fact that each entry in the Hadamard matrix is the scalar product of the binary representations of its row and column numbers.

$H_{i,j}(s) =< i, j >= i_0 j_0 \oplus i_1 j_1 \oplus \ldots \oplus i_{s-1} j_{s-1}$,

where i and j are s-bit binary numbers; $\oplus$ stands for EXOR;

$i = (i_0, i_1, \ldots, i_{s-1}), j = (j_0, j_1, \ldots, j_{s-1})$.

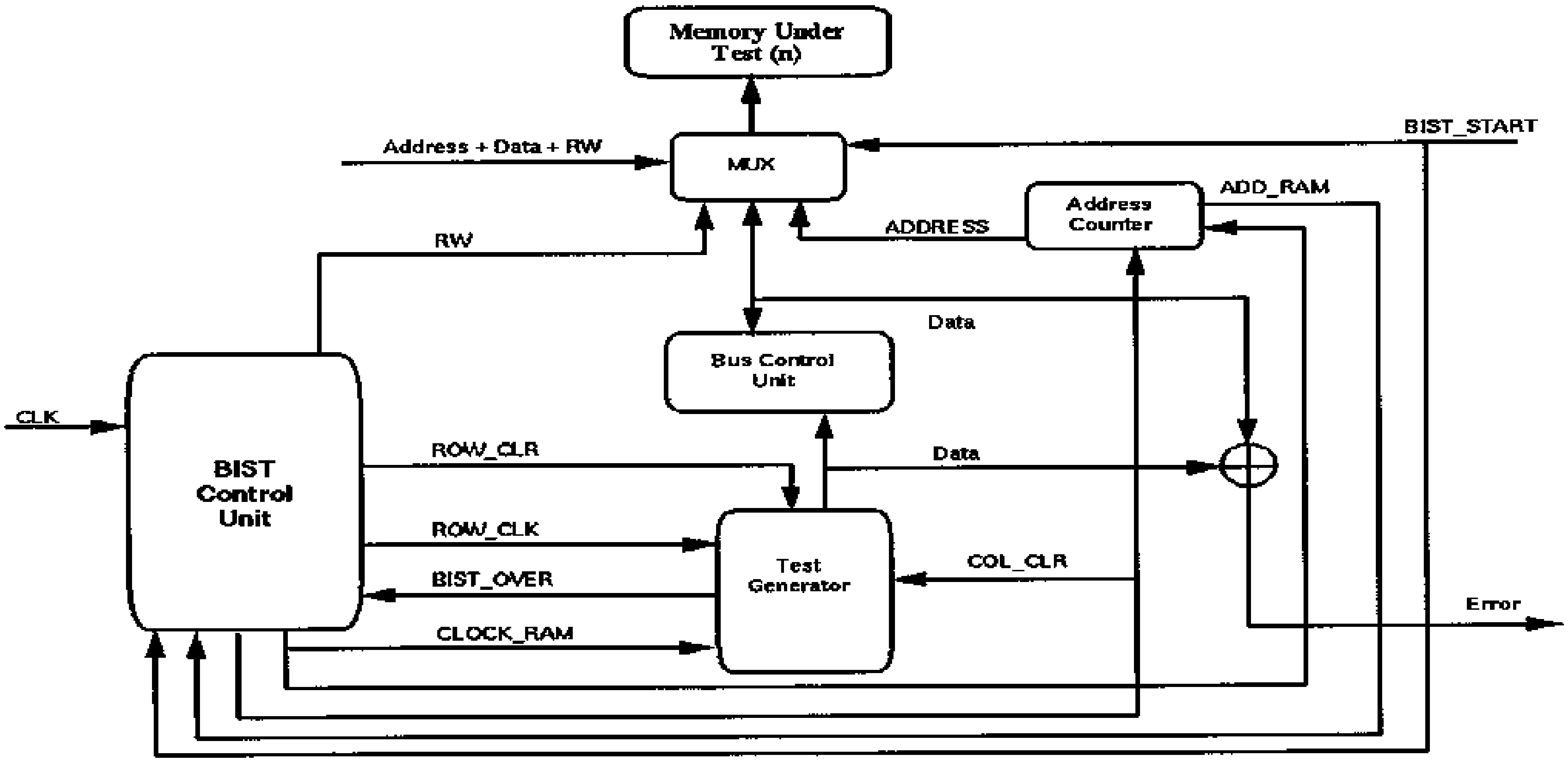

Figure 5: BIST Architecture for ϵ-Exhaustive Testing

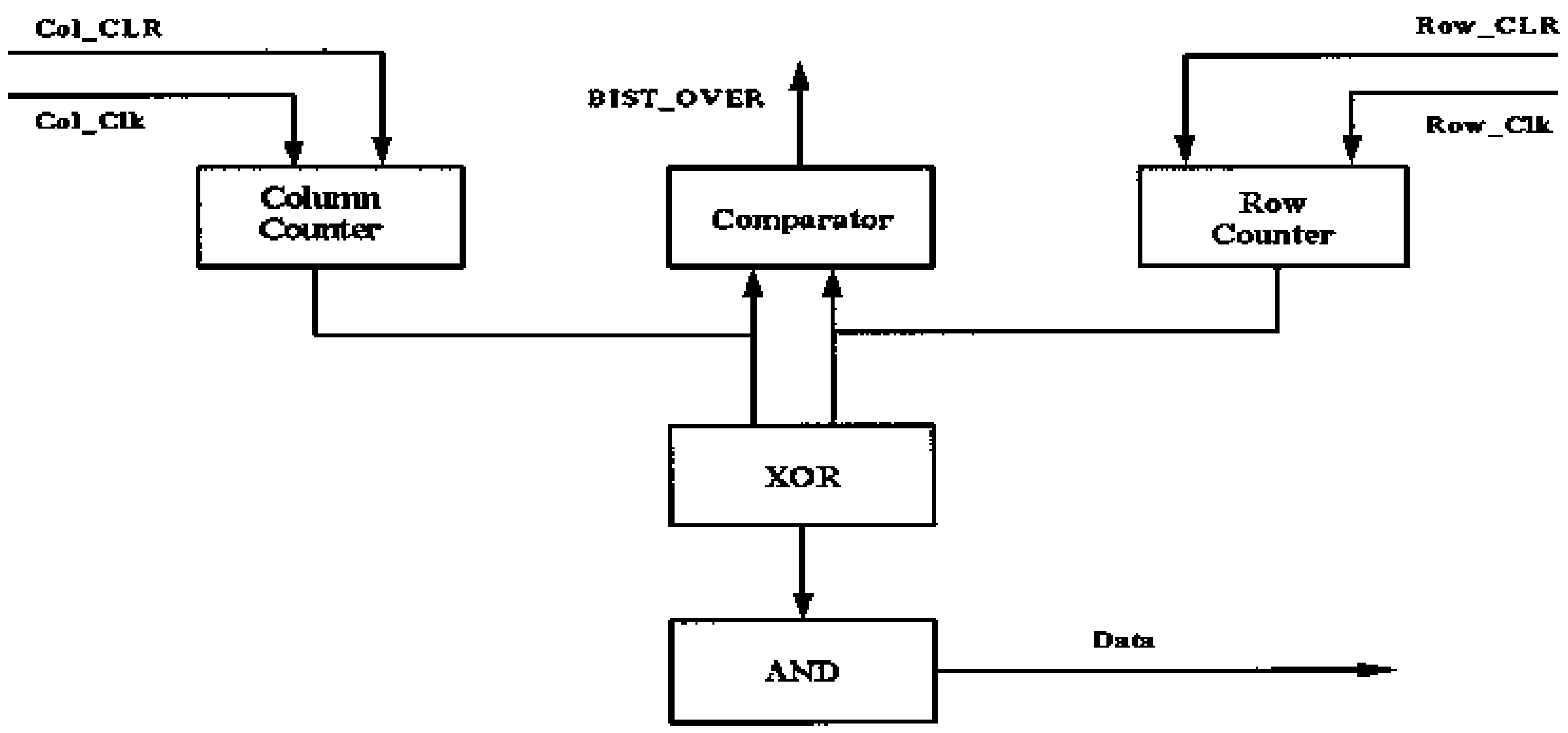

Figure 6: Test Generator for ϵ-Exhaustive Testing

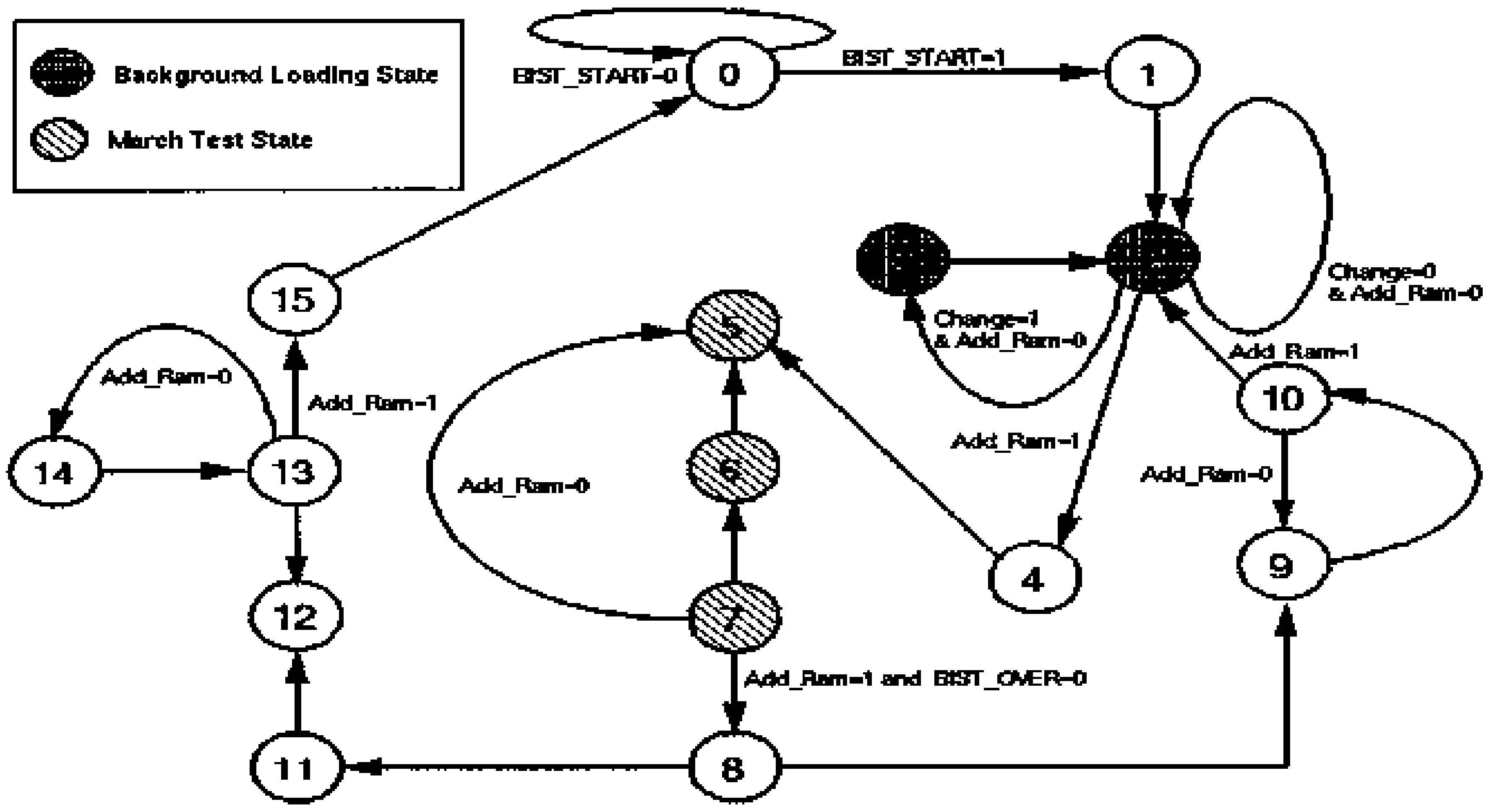

Figure 7: Controller State Diagram for ϵ-Exhaustive Testing

4.3 BIST Area Overheads

The BIST unit transistor counts and area estimations have been made by using Viewlogic's Powerview Toolkit to compile VHDL descriptions into standard cells designs. The TimberWolfSC standard cell placement and global routing tool have been used to generate the final physical design using a gates library.

For the exhaustive testing approach we have chosen $k = 5$. The transistor overhead for the **BIST** unit was 7,500 transistors in the control unit and 25,500 transistors in the ROM which stores the seeds for the backgrounds. Assuming that an SRAM has 7 transistors per cell, the hardware overhead in terms of transistors comes out to 0.47%. Even for a DRAM with one transistor per cell and ignoring the overhead for refresh logic, the overhead is only 3.3%. In terms of area, the overhead is leass than 1% for an SRAM and 6.8% for DRAM. Most of the overhead results from the ROM.

For the near-exhaustive testing approach we have chosen $k = 9$ and $\epsilon = 0.01$. The hardware overhead in terms of transistors was 2,600. This comes out to be 0.04% for an SRAM and 1.5% for a DRAM. In terms of area, the overhead is less than 0.2% for an SRAM and 1.5% for DRAM. The overheads are lower in the case of near-exhaustive tests as there is no necessity for the storage of seed matrices.

Most of the **BIST** circuit is independent of the size of the RAM. Thus, the larger the RAM, the lower the **BIST** overhead as a percentage of the whole circuit.

5 Conclusion

In this work we considered the problem of detection and location of single and unlinked multiple NPSFs (Active, Passive and Static) and couplings, with the size of the neighborhood equal k, in $n \times 1$ RAMs, for k=4,5,6,7,8,9 and n of the order 10^6 or 10^7. A fault model, the $PS(n, k)$ fault model, covering these faults was proposed. Then the problem was reduced to the generation of *exhaustive tests* (k=4,5,6) and ϵ-*exhaustive tests* (k=7,8,9). In Table 12 we summarize test times for our construction and for the best previously known constructions [5]. Our tests are labelled k-EXH or k-ϵEXH depending on whether it is a exhaustive or ϵ-exhaustive test respectively. A 10 MHz test vector application rate is assumed and our test times are computed for ANPSFs only to permit comparison with Cockburn's tests [5].

Author/Test	*256K*	*1M*	*4M*	*16M*
Nair et al.	15.1s	1mins 7.2s	4mins 55.7s	21mins 30.2s
Papachristou & Sahgal	11.4s	50.4s	3mins 41.9s	16mins 8.0s
Suk & Reddy	2.6s	10.4s	41.7s	2mins 46.9s
Franklin & Saluja	36mins 14.3s	4hrs 28.5mins	17hrs 54.2mins	5d 2.2hrs
S2CTEST	0.3s	1.0s	4.2s	16.8s
S3CTEST	2.4s	10.3s	44.5s	3mins 11.3s
S3CTEST2	2.6s	11.4s	49.1s	3mins 29.7s
S4CTEST	22.9s	1min 54.2s	8mins 20.4s	33mins 21.4s
S5CTEST	3mins 16.7s	19mins 14.3s	1hrs 17.0mins	5hrs 7.8mins
4-EXH	22.05s	1min 37.2s	7mins 10.2s	31mins 40.8s
5-EXH	2mins 32.4s	11mins 13.8s	50mins 1.2s	3hrs 40mins
6-EXH	23mins 54s	1hrs 36mins	7hrs 51mins	

Table 8: Comparison of Test Times

It can be seen from the comparison of test times that our test times are considerably shorter than Cockburns (S4CTEST and S5CTEST). Furthermore, we have extended the tests to 6-EXH which detects 6-couplings. 5-EXH like the S5CTEST has the convenient property of being able to detect all NPSFs of Type I neighborhood regardless of the mapping from logical cell addresses to physical cell locations. The table illustrates the penalty in increased time that is paid for this. SC5TEST takes roughly 111 times longer to test a 1 Mbit RAM than Suk and Reddy's near-optimal test which requires the knowledge of the mapping from the logical cell addresses to physical cell locations. Our 5-EXH on the other hand takes roughly 65 times longer to test a 1M RAM than Suk and Reddy's near optimal test. If each cell can be assumed to have three rather than four nearest neighbors, then 4-EXH can be used instead of 5-EXH. The test length penalty is then reduced, for a 1Mbit RAM, from 65 to roughly 10.

Test	Test lengths	Test times
7-ϵEXH	3.8×10^{10}	1.06hrs
8-ϵEXH	7.6×10^{10}	2.11hrs
9-ϵEXH	1.5×10^{11}	4.17hrs

Table 9: Test Times for $n = 2^{20}$ and $\epsilon \leq .01$

Test times for detection of 7,8 and 9-couplings by near-exhaustive tests with a probability $\geq$ 99% for a 1M RAM are given in Table 14. The fraction of undetected faults for $7-\epsilon$EXH, $8-\epsilon$EXH and $9-\epsilon$EXH, ϵ is actually smaller than indicated if we take into account the contribution of the bad B_k submatrices. Test times and/or ϵ can be further reduced by performing another vertical concatenation using an independent random permutation of the Hadamard matrix.

Finally, we have described a BIST scheme for our proposed test and have evaluated the hardware area overhead. For a 1 Mbit memory, the BIST area overhead for the detection of 5-couplings is less than 1% for SRAM and 6.8% for a DRAM. For the detection of 9-couplings with a probability $\geq$ 99%, the BIST area overhead is less than 0.2% for SRAM and 1.5% for DRAM.

6 Appendix

Seed Matrices used for the construction of exhaustive backgrounds :

- k=3

  All the seed matrices are taken from the construction by Sloane [20].

- k=4

  $B(5,4)$: Optimal solution is obtained using the method of *constant weight vectors* [21]. $B(7,4)$ and $B(11,4)$: Obtained using a heuristic-guided computer program [5]. The $B(11,4)$ seed is used for $B(7,4)$ also.

$B(13,4)$ upto $B(37,4)$: Constructed using the deterministic algorithm given by Chandra *et al* [15].

$B(41,4)$ upto $B(107,4)$: Constructed using the method based on MDS codes [22] with the base matrix as $B(11,4)$ matrix.

n	$T_{n,4}$	n	$T_{n,4}$	n	$T_{n,4}$	n	$T_{n,4}$	n	$T_{n,4}$	n	$T_{n,4}$	n	$T_{n,4}$
5	16	7	22	11	22	13	50	17	56	19	56	23	76
29	99	31	103	37	104	41	110	43	110	47	110	53	110
59	110	61	110	67	110	71	110	73	110	79	110	83	110
89	110	97	110	101	110	103	110	107	110				

Table 10: Number of Rows $T_{n,4}$ in the Seed Matrices for k=4

- k=5

  $B(5,5)$ and $B(7,5)$: Optimal solution is obtained using the method of *constant weight vectors* [21].

  $B(11,5)$ and $B(13,5)$: Obtained using a heuristic-guided computer program.

  $B(17,5)$ upto $B(47,5)$: Constructed using the iterative procedure given by Tang and Chen [11].

  $B(53,5)$ upto $B(61,5)$: Constructed using the method based on MDS codes [22] with the base matrix as $B(8,5)$ matrix.

  $B(67,5)$ upto $B(113,5)$: Constructed using the method based on MDS codes [22] with the base matrix as $B(11,5)$ matrix.

  $B(127,5)$ upto $B(167,5)$: Constructed using the method based on MDS codes [22] with the base matrix as $B(13,5)$ matrix.

n	$T_{n,5}$	n	$T_{n,5}$	n	$T_{n,5}$	n	$T_{n,5}$	n	$T_{n,5}$	n	$T_{n,5}$	n	$T_{n,5}$
5	32	7	42	11	101	13	104	17	294	19	294	23	294
29	294	31	294	37	294	41	294	43	294	47	294	53	392
59	392	61	392	67	707	71	707	73	707	79	707	83	707
89	707	97	707	101	707	103	707	107	707	109	707	113	707
127	728	131	728	137	728	139	728	149	728	151	728	157	728
163	728	167	728										

Table 11: Number of Rows $T_{n,5}$ in the Seed Matrices for k=5

Acknowledgements

The authors would like to thanks Dr. L.B.Levitin of Research Lab on Design and Testing of Computer Hardware, Boston University for many helpful discussions. We also wish to thank the editor and the reviewers for their valuable comments.

References

[1] A.J. van de Goor, *"Testing Semiconductor Memories, Theory and Practice,"* John Wiley & Sons, Chichester, UK, (1991)

[2] J.P.Hayes, "Detection of Pattern-Sensitive Faults in Random-Access Memories," *IEEE Transactions on Computers*, **vol.c-24**, pp 150-157, (1975)

[3] R.Nair, S.M.Thatte, and J.A.Abraham, "Efficient Algorithms for Testing Semiconductor Random-Access Memories", *Dig. of Papers, 7th Intl. Conf. on Fault-tolerant Computing*, Los Angels, CA, pp 81-87, (1977)

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

[5] B.F.Cockburn, "Deterministic Tests for Detecting Single V-Coupling Faults in RAMs," *J. Electronic Testing*, **vol.5**, no.1, pp 91-113, (1994)

[6] M.G.Karpovsky, V.N.Yarmolik and A.J van de Goor, "Pseudoexhaustive Word-Oriented DRAM Testing", *Proc. 1995 European Design and Test Conference*, (1995)

[7] J.Savir, W.H.McAnney and S.R.Vecchio, "Testing for Coupled Cells in Random Access Memories," *IEEE Transactions on Computers*, **vol.c-40**, no. 10, pp 1177-1180, (1991)

[8] B.F.Cockburn, "A 20 MHz Test Vector Generator for Producing Tests that Detect Single 4- and 5-Coupling Faults in RAMs," *Records of the 1993 IEEE Int. Workshop on Memory Testing*, San Jose, CA, (1993)

[9] M.G.Karpovsky and V.N.Yarmolik, "Transparent Memory Testing for Pattern Sensitive Faults," *Proc. International Test Conference*, pp 862-867, (1994)

[10] M.G.Karpovsky and V.N.Yarmolik, "Transparent Memory BIST", *Proc. International Workshop on Memory Tech.*, pp 106-112, (1994)

[11] D.T.Tang and C.L.Chen, "Iterative Exhaustive Pattern Generation for Logic Testing," *IBM J. of Res. Develop.*, **vol.28**, no.2, pp 212-219, (1984)

[12] A.J.Goor and C.A.Verruijt, "An Overview of Deterministic Functional RAM Chip Testing," *ACM Computing Surveys*, **vol.22**, no.1, (1990)

[13] L.B.Levitin and M.G.Karpovsky, "Traveling Salesman Problem in the Space of Binary Vectors," *Proc. of International Symp. on Information Theory*, Norway, (1994)

[14] L.B.Levitin and M.G.Karpovsky, "Exhaustive Testing of Almost all Devices with Outputs Depending on Limited Number of Inputs," *Open Systems and Information Dynamics*, **vol.2**, no.3, pp 1-16, (1994)

[15] A.K.Chandra, L.T. Kou, G.Markowsky, and S.Zaks, "On Sets of Boolean n-Vectors With all k-Projections Surjective," *Acta Informatica*, **vol.20**, pp 103-111, (1983)

[16] R.David, A.Fuentes and B.Courtois, "Random Pattern Testing versus Deterministic Testing of RAMs," *IEEE Transaction on Computers*, **vol.c-38**, pp 637-650, (1989)

[17] R.David, J.A.Brzozowski, H.Jurgensen, "Random Test Lengths for Bounded Faults in RAMs," *Proc. of the Third European Test Conference, Rotterdam*, IEEE Computer Society Press, Los Alamitos, pp 149-158, (1993)

[18] F.J.Macwilliams and N.J.A.Sloane, "The Theory of Error-Correcting Codes", *North Holland Publishing Company*, Amsterdam, (1977)

[19] M.Franklin and K.K.Saluja, "Built-In Self-Testing of Random-Access Memory," *IEEE Transaction on Computers*, pp 45-55, (1990)

[20] N.J.A. Sloane, "Covering Arrays and Intersecting Codes," *J. Combinatorial Design* , **vol.1**, no.1, pp 51-64, (1993)

[21] D.T.Tang and L.S.Woo, "Exhaustive Test Pattern Generation with Constant Weight Vectors," *IEEE Transaction on Comp.*, **vol.c-32**, no.12, pp 1145-1150, (1983)

[22] L.B.Levitin and M.G.Karpovsky, "Efficient Exhaustive Test Based on MDS Codes," *IEEE International Symposium on Information Theory*, Ann Arbor, (1986)