Minimal Test for Coupling Faults in Word-Oriented Memories

Ad J. van de Goor[1] Magdy S. Abadir[2] Alan Carlin[2]

[1]Delft University of Technology, Department of Information Technology and Systems,
Section Computer Engineering; Mekelweg 4, 2628 CD Delft, The Netherlands

[2]Motorola, Test and Logic Verification; 6200 Bridgepoint Parkway, Bldg 4, Austin, TX 78730, U.S.A.

E-mail: A.J.vandeGoor@ITS.TUDelft.nl; abadir@ibmoto.com; alan.carlin@motorola.com

Abstract: *Most industrial memories have an external word-width of more than one bit. However, most published memory test algorithms assume 1-bit memories; they will not detect coupling faults between the cells of a word. This paper improves upon the state of the art in testing word-oriented memories by presenting a new method for detecting state coupling faults between cells of the same word, based on the use of* m-*out-of-*n *codes. The result is a reduction in test time, which varies between 20 and 38%.*

Key words: *State coupling faults, word-oriented memories, tests, data backgrounds,* m-*out-of-*n *codes.*

1 Introduction

The problem with testing memories is in the amount of test time required to obtain a given fault coverage. Therefore there has been a trend away from the order $\Theta(n^2)$, $\Theta(n * \log_2 n)$ and $\Theta(n^{3/2})$ algorithms to linear (i.e., $\Theta(n)$) algorithms; whereby n represents the number of bits in the memory. The current trend is to reduce the test time even further, from $k_1 * n$ to $k_2 * n$; whereby $k_2 < k_1$. Most memory test algorithms have been designed to detect faults in memories which have a word width of a single bit [1]; i.e., $B = 1$. These memories are referred to as *Bit-Oriented Memories* (*BOMs*). Naturally, most industrial memories allow for writing and reading several bits simultaneously; e.g., 8, 16, or even 256 bits can be read and/or written at a time. These memories are referred to as *Word-Oriented Memories* (*WOMs*); they have the property that $B \geq 2$. The problem now is how to use the vast amount of existing BOM tests to cover faults in WOMs, at a minimal test cost? This problem only exists for *coupling faults* (*CFs*) [1, 2, 3], which have the property that the state of, or an operation applied to, the *aggressor cell* (*a-cell*) forces the state, or changes the state of the *victim cell* (*v-cell*). Figure 1 shows the following classes of CFs in a memory with four 5-bit (i.e., $B = 5$) words:

1. Inter-word CFs (Left case in figure)
The a-cell and the v-cell belong to *different* words.

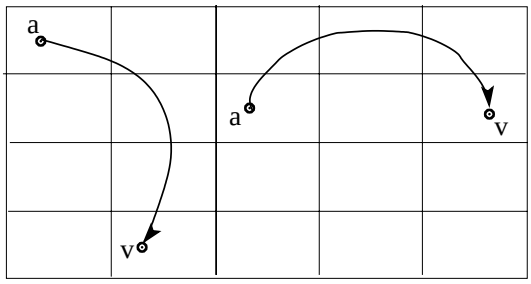

Figure 1: Inter and intra-word CFs

Classical BOM tests are designed to detect single-cell faults (e.g., stuck-at faults (SAFs), transition faults (TFs)) and inter-word CFs. Examples of BOM test are: MATS+ [6], which detects address decoder faults and SAFs, and March C- [1], which detects address decoder faults, SAFs, TFs, and most types of inter-word CFs.

2. Intra-word CFs (Right case)
The a-cell and the v-cell belong to *the same* word. Special WOM tests are required to detect this class of faults [1, 2, 3, 4, 5].

Dekker [4] has designed a systematic way to solve the problem of detecting intra-word CFs, assuming that the faults between the cells of a given word are *State Coupling Faults* (*CFsts*); the proof is given in Section 2. The B-bit data patterns used to write to and read from the WOM are referred to as *Data Backgrounds* (*DBs*). Section 3 describes more recent work [2, 3], showing that a considerable reduction in test time can be obtained by separating the problems of detecting inter-word and intra-word CFs; it essentially reduces the test time from the *product* to the *sum* of the BOM and the WOM test times. In addition, extensions to more complex coupling fault models are presented. Section 4 shows that the methods of Sections 2 and 3 are based on the assumption that complemetary pairs of DBs had to be used, which is not required any more, given the separation of the tests for inter- and intra-word coupling faults. It thereafter proposes a new method for

deriving DBs for CFsts, based on the use of m-out-of-n codes [7], which allows for a significant further reduction of the WOM test time. Last, Section 5 ends with conclusions.

2 Traditional WOM tests

Dekker [4] published a method to derive a set of DBs and how to apply this set to cover CFsts in WOMs. The used intra-word coupling fault model is that of the state coupling fault (CFst), defined as follows: a v-cell, v, is forced to a given state, y, only if the a-cell, a, is in a given state, x. A formal syntax for describing this type of fault is [1]: $< S; F >$. Whereby S describes the state of the a-cell and F describes the resulting faulty state of the v-cell. Four different CFsts can be distinguished: $< 0; 0 >$, $< 0; 1 >$, $< 1; 0 >$ and $< 1; 1 >$. The CFst $< 1; 0 >$ means that if the a-cell is in state 1 it will force the 0 state in the v-cell. In order to detect all CFsts, it has to be verified that all four states are possible between any two bits in a word; i.e., $(b_i, b_j) \in \{(0,0), (0,1), (1,0), (1,1)\}$, for any b_i and any b_j. Dekker's solution is based on the observation that most (if not all) march tests contain read and write operations with some data value as well as the complementary data value. For example, a classical march test is the MATS+ test [6], which requires a test time of $5 * n$: $\{\Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0)\}$. It consists of three march elements, separated by the ';' symbol. The march element '$\Uparrow (r0, w1)$' means that if cell 3 is operated upon, a read operation (with expected value 0) is applied to cell 3, followed by the application of a write '1' operation; thereafter the same two operations will be applied to cell 4, etc. From MATS+ it can be seen that it applies the complementary data values 0and 1.

The objective for covering all CFsts is to design a minimal set of D DBs. Initially, only the states (0,0) and (0,1) have to be verified; because the other two states will be verified when the algorithm is executed with complementary data. The set of D DBs can be represented by a matrix $\mathbf{M}$, with D rows and B columns. A column of $\mathbf{M}$, denoted as *Vector* $\mathbf{V}c$, represents the DB values for cell c in a word, because that column represents the values successively written and read from cell c. For a minimal set of DBs the following requirements should be satisfied [4]:

1. No two vectors $\mathbf{V}i$ and $\mathbf{V}j$ may be equal. If they are equal, then the corresponding bits b_i and b_j will only be checked for the states (0,0) and (1,1). This results in a maximum number of different D-bit vectors, determined by the word length, as follows: $2^D = B$.
2. No two vectors $\mathbf{V}i$ and $\mathbf{V}j$ may be each others inverse. If they are each others inverse, then the bits b_i and b_j will only be checked for the states (0,1) and (1,0). This halves the maximum number of vectors; i.e., $2^{D-1} = B$, or $D = \log_2 B + 1$.

$$\mathbf{M} = \begin{vmatrix} b_{0,0} & b_{0,1} & \dots & b_{0,B-1} \\ b_{1,0} & b_{1,1} & \dots & b_{1,B-1} \\ \vdots & \vdots & \vdots & \vdots \\ b_{D-1,0} & b_{D-1,1} & \dots & b_{D-1,B-1} \end{vmatrix}$$

The number of DBs, D, required for checking the states (0,0) and (0,1) therefore is: $D = \lceil \log_2 B \rceil + 1$; if B is a power of 2, then this equation will become: $D = \log_2 B + 1$. The number of DBs required for detecting *all four* CFsts is: $D = 2 * (\lceil \log_2 B \rceil + 1)$.

For a memory with a word-width of B bits, the set of DBs can now be constructed:

1. A DB with all 0's (and its inverse: all 1's)
2. A DB with alternating 0's and 1's
(i.e., 0101.0101. ... , and its inverse: 1010.1010. ...)
3. A DB with alternating pairs of 0's and 1's
(i.e., 0011.0011. ... , and its inverse: 1100.1100. ...)
4. A DB with alternating quadruplets of 0's and 1's
(i.e., 0000.1111.0000.1111. ... , and its inverse)
5. Etc., until $\lceil \log_2 B \rceil + 1$ steps have been performed

For a memory with $B = 16$, $D = 2*(\log_2 16+1) = 10$ DBs are required, as shown in Table 1. The reader can verify that all 4 states occur for any pair of cells.

In case B is not a power of 2 (i.e., $B = 2^q - r$; *and* $r > 0$), the set of DBs for the next higher power of 2 has to be generated, after which r columns of the matrix have to be eliminated. For example, if $B = 13$, the set of DBs for $B = 16$ has to be generated, from which $16 - 13 = 3$ columns have to be deleted; these may be any 3 columns of the set of 16 columns.

The proposed way the set of DBs is applied is as follows: (1) The BOM test (e.g., MATS+) is made word-wide, by replacing the value 0 in the algorithm with any B-bit DB, and by replacing the 1 value with its inverse; and (2) The word-wide BOM test is applied $D/2$ times, every time using a different complementary pair of DBs.

The result is that the application time of the WOM test becomes the product of the number of $DBs/2$ and the BOM test time. Many publications are based on this method for testing WOMs [1, 4, 5].

Table 1: Data backgrounds for CFsts and $B = 16$

Normal		Inverse	
0	0000 0000 0000 0000	1	1111 1111 1111 1111
2	0101 0101 0101 0101	3	1010 1010 1010 1010
4	0011 0011 0011 0011	5	1100 1100 1100 1100
6	0000 1111 0000 1111	7	1111 0000 1111 0000
8	0000 0000 1111 1111	9	1111 1111 0000 0000

As an example, the following MATS+ based set of tests covers intra-word CFsts in a memory with $B = 4$ ($D = 6$), with a test time of $5 * n/B * D/2 = 15 * n/4$.
1. $\{\Updownarrow\ (w0000); \Uparrow\ (r0000, w1111); \Downarrow\ (r1111, w0000)\}$;
Used DB pair '0000' and '1111'
2. $\{\Updownarrow\ (w0101); \Uparrow\ (r0101, w1010); \Downarrow\ (r1010, w0101)\}$;
Used DB pair '0101' and '1010'
3. $\{\Updownarrow\ (w0011); \Uparrow\ (r0011, w1100); \Downarrow\ (r1100, w0011)\}$;
Used DB pair '0000'and '1111'

3 Improved WOM tests

In [2, 3] it has been shown that the problem of testing for intra-word coupling faults can be divided into two parts, which can be handled independently:
1. Test for single-cell and inter-word coupling faults
This should be done with a word-wide BOM test such as MATS+, March C-, etc. For example, below, 4 out of 16 possible word-wide versions of MATS+, for $B = 4$, are shown.
1. $\{\Updownarrow\ (w0000); \Uparrow\ (r0000, w1111); \Downarrow\ (r1111, w0000)\}$
2. $\{\Updownarrow\ (w0001); \Uparrow\ (r0001, w1110); \Downarrow\ (r1110, w0001)\}$
3. $\{\Updownarrow\ (w0010); \Uparrow\ (r0010, w1101); \Downarrow\ (r1101, w0010)\}$
4. $\{\Updownarrow\ (w0011); \Uparrow\ (r0011, w1100); \Downarrow\ (r1100, w0011)\}$

From a ground-bounce point of view, industrial results indicate that the '0000' and '1111' complementary pair of DBs is to be preferred; in addition, this pair is also required for detecting intra-word CFsts.
2. Test for intra-word coupling faults
This has to be done with a *separate test*, by writing and reading each of the DBs of Section 2 as follows: $\Updownarrow\ (w_{DB2}, r_{DB2}, w_{DB3}, r_{DB3}, \ \cdots\ , w_{DBD-1}, r_{DBD-1})$; i.e., each of the DBs is written to and read from each word. Note that the all 0s (DB #0 of Table 1) and all 1s (DB #1) DBs have already been applied by the word-wide BOM test, therefore they can be deleted from the test for intra-word CFsts. In addition, the single march element may be broken up into a number of march elements, and the address order of each march element may be chosen freely [2, 3]. Compared with Dekker's method the overall test time is reduced from the *product* to the *sum* of the BOM and WOM test times. For example, for a 4-bit memory ($D = 6$), using the MATS+ BOM test, Dekker's method requires a test time a test time of $5 * n/B * D/2 = 15 * n/4$, as compared with $5 * n/B + (D-2) * n/B * 2 = 13 * n/4$, for the improved method. Using the $10 * n$ BOM test March C- [1] for a 64-bit memory ($D = 14$), Dekker's method requires $10 * n/64 * 7 = 70 * n/64$, as compared with $10 * n/B + 12 * n/B * 2 = 34 * n/64$, for the improved method; a reduction by over 51%!

In [2, 3] additionally DBs for intra-word *indempotent CFs* (*CFids*) and *disturb CFs* (*CFds*s) have been established. A CFid is defined, using the $< S/F >$ notation, as: $<\uparrow;0>, <\uparrow;1>, <\downarrow;0>, <\downarrow;1>$. For example, the CFid $<\uparrow;0>$ means that an '↑' transition (i.e., a $0 \rightarrow 1$ transition) in the a-cell, forces a 0 in the v-cell. A CFds is defined as [9]: $< r0;0 >, < r0;1 >, < r1;0 >, < r1;1 >, < w0;0 >, < w0;1 >, < w1;0 >, < w1;1 >$; read as well as write operations can sensitize CFs, while only the data value 0 or 1 of the r and the w operations are relevant (rather than transitions). In addition to the fault types CFst, CFid and CFds, [2, 3] also introduce the following four classes of intra-word CFs:
1. *Unrestricted intra-word CFs* (*uCFs*)
The v-cell is only influenced by *a single a-cell*; the a- cell may be *any* of the B-1 non-v-cells. This fault model does not require knowledge about the layout of the cells in a word. It is used by Dekker [4].
2. *Restricted intra-word CFs* (*rCFs*)
The v-cell is only influenced by *a single a-cell*, which is *the physical neighbor* of the v-cell. This requires precise knowledge of the layout of the memory words and the routing of the I/O data lines.
3. *Concurrent-restricted intra-word CFs* (*crCFs*)
The v-cell is only influenced by the concurrent action or state of *two* a-cells, which are the physical neighbors of the v-cell.
4. *Concurrent intra-word CFs* (*cCFs*)
The v-cell is influenced by the concurrent action or state of the B-1 non-v-cells in the word; all a-cells perform the same sensitizing operation (Motivation: Ground and Vcc bounce).

For example, if the rCF fault model is used for CFsts, then only 4 DBs are required, independent of B, because it only has to be verified that all four states between two neighboring cells c_i and c_{i+1} are possible ($0 \leq i \leq B-2$). For an 8-bit memory these DBs are: 0000.0000, 1111.1111, 0101.0101 and 1010.1010.

Table 2 [2, 3] shows the number of DBs, D, and the test time, t, as a function of the three CF types (CFst, CFid and CFds) and the four classes of intra-word CFs (uCF, rCF, crCF and cCF). The values for D and t assume that the word-wide BOM test already applied the all 0s and all 1s DBs. The 2^{nd} and the 3^{rd} columns show the equations for D and t, while the last two columns show the value pair '*D;t*' for a memory with $B = 4$ and with $B = 32$, respectively. Note: The values of t in the 3^{rd}, the 4^{th} and the 5^{th} column have to be multiplied by n/B. From this table one can conclude that the test time increases with the complexity of the CF type (from CFst to CFid to CFds) and with the CF class (from rCF to uCF to crCF to cCF). Most industrial tests, however, only test for un-

Table 2: Number of DBs D and test time t

CF type	# of DBs: D	Test time: t	$D;t$	
uCFst	$2 * \lceil \log_2 B \rceil + 2$	$4 * \lceil \log_2 B \rceil + 2$	6;10	12;22
rCFst	4	6	4;6	4;6
crCFst	8	14	8;14	8;14
cCFst	$2 * B + 2$	$4 * B + 2$	10;18	66;130
uCFid	$3 * \lceil \log_2 B \rceil + 3$	$6 * \lceil \log_2 B \rceil + 1$	9;13	18;31
rCFid	6	10	6;10	6;10
crCFid	12	19	12;19	12;19
cCFid	$3 * B = 2$	$5 * B + 3$	14;23	98;163
uCFds	$3 * \lceil \log_2 B \rceil + 3$	$7 * \lceil \log_2 B \rceil + 6$	9;20	18;41
rCFds	6	13	6;13	6;13
crCFds	12	27	12;27	12;27
cCFds	$3 * B + 2$	$6 * B + 9$	14;33	98;201

Table 3: Test time as a function of B

All 0s and 1s not in code			All 0s or all 1s in code		
B	C_m^n	t	B	C_m^n	t
3	$C_1^3 = 3$	6	3	$C_1^3 = 3$	8
4-6	$C_2^4 = 6$	8	4	$C_1^4 = 4$	10
7-10	$C_2^5 = 10$	10	5-10	$C_2^5 = 10$	12
11-20	$C_3^6 = 20$	12	11-15	$C_2^6 = 15$	14
21-35	$C_3^7 = 35$	14	16-35	$C_3^7 = 35$	16
36-70	$C_4^8 = 70$	16	36-56	$C_3^8 = 56$	18
71-126	$C_4^9 = 126$	18	57-126	$C_4^9 = 126$	20
127-252	$C_5^{10} = 252$	20	127-210	$C_4^{10} = 210$	22

restricted CFsts (uCFsts), because of potential bridges between cells within a word and between the I/O data lines. In addition, the uCFst model does not require any knowledge about the layout of the memory and the routing of the data lines to/from the I/O pads. Note additionally, that in case of repair, using spare rows and/or columns, the normal topology of the memory is altered, which makes it safe to use the uCFst model instead of the more time efficient rCFst model.

4 Optimal WOMs tests

As shown before, the traditional and improved methods established the minimum number of DBs based on the assumption that complementary pairs of DBs had to be used. This does not have to appply to the improved method of Section 3, such that a new opportunity for test time reduction presents itself! The problem to be solved now is: *What is the minimal number of DBs such that every pair of bits is tested for uCFsts (i.e.,* unrestricted *CFsts), assuming that the word-wide BOM test applies the all 0s and all 1s DBs.* The use of m-out-of-n codes, which are also called *constant weight codes* with weight m, solves this problem [7]. In an m-out-of-n code, each n-bit code word contains exactly m 1s. Note that the symbol n has already been defined in Section 1 to mean the number of bits in the memory; the context should make this unambiguous, however! The number of code words in an m-out-of-n code is: $C_m^n = n!/(n-m)! * m!$ A 2-out-of-5 code consists of $C_m^n = C_2^5 = 5!/(2! * 3!) = 10$ code words; see matrix $\mathbf{M}_2^5$ with $D = n = 5$ rows and $C_2^5 = 10$ columns. This code can be used for detecting intra-word uCFsts for a memory with $B \leq 10$.

$$\mathbf{M}_2^5 = \left| \begin{array}{cccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{array} \right|$$

A matrix of code words, constructed in the above way, detects all CFsts because all bit pairs (b_i, b_j) are exhaustively tested for the *two* states (0,1) and (1,0) by applying the $D = n$ patterns.

Proof: The patterns applied to any two bits, b_i and b_j, come from the serialization of the two vectors (code words) $\mathbf{V}i$ and $\mathbf{V}j$ of an m-out-of-n code. Because of the fact that $\mathbf{V}i$ and $\mathbf{V}j$ are different code words, there must be a row in $\mathbf{M}$ for which $(\mathbf{V}i, \mathbf{V}j) = (0,1)$ *or* $(1,0)$. Consider that $(0,1)$ occurs, then also $(1,0)$ has to uccur because $\mathbf{V}i$ and $\mathbf{V}j$ contain an equal number of 1s. *Q.E.D.*

In order to allow for the maximum number of code words for a given n, m has to be: $m = \lceil n/2 \rceil$ *or* $\lfloor n/2 \rfloor$ [8]; from here on the $\lfloor n/2 \rfloor$ alternative will be assumed (note that either choice is optimal). This results in the following codes: $C_1^3, C_2^4, C_2^5, C_3^6, C_3^7, C_4^8, C_4^9, C_5^{10}$, etc; see Table 3, left part. The algorithm for establishing D for a particular B is based on the selection of C_m^n, such that the following ralationship holds, assuming $m = \lfloor n/2 \rfloor$: $C_{\lfloor (n-1)/2 \rfloor}^{n-1} < B \leq C_{\lfloor n/2 \rfloor}^n$. The left part of Table 3 shows the ranges for B; for example, consider $B = 8$, which falls in the range $7 \leq B \leq 10$. This range is derived from the column C_m^n, from which D is derived as folows: $D = n = 5$. The third column shows the test time: $t = 2 * D(*n/B) = 2 * 10(*n/8) = 20(*n/8)$, whereby the factor n/B is not included in the table.

A different set of code words has to be established for the case that the word-wide BOM test does not include the all 0s and/or the all 1s DB. By inspection, it can be verified that m-out-of-n codes are not able to generate a matrix $\mathbf{M}$ which contains all four data value pairs $(0,0), (0,1), (1,0)$ *and* $(1,1)$ for any pair of bits; however, they can be designed such that they do cover the $(0,0)$ or the $(1,1)$ case. If the word-wide BOM test does not cover the $(0,0)$ *and* the $(1,1)$ case, then either a row of all 0s has to be added to the matrix $\mathbf{M}$ and the m-out-of-n code has to generate the $(1,1)$ case, or a row with all 1s has to be added and the $(0,0)$ case has to be generated; we will assume the latter alternative from now on. This means that the $(0,0)$ case has to be generated by using m-out-of-n codes with $m = \lceil n/2 \rceil - 1$. They have the property that the number of 1s in a code word is one larger than the

Table 4: Test time comparison

B	Dekker	Impr.	Optim.	%
4	$2*n/4$*BOM	$10*n/4$	$8*n/n$	20
8	$3*n/8$*BOM	$14*n/8$	$10*n/8$	28
16	$4*n/16$*BOM	$18*n/16$	$12*n/16$	33
32	$5*n/32$*BOM	$22*n/32$	$14*n/32$	36
64	$6*n/64$*BOM	$26*n/64$	$16*n/64$	38

number of 0s. A matrix of code words, constructed in the above way, detects all uCFsts because all pairs of bits (b_i, b_j) are exhaustively tested for the four states: (0,0), (0,1) and (1,0) by the C_m^n code words, and (1,1) by the added all 1s row, such that $D = n + 1$.

Proof: The patterns applied to any two bits, b_i and b_j, come from the serialization of the two vectors (code words) $\mathbf{V}i$ and $\mathbf{V}j$ of an m-out-of-n code, followed by the all 0s pattern. $\mathbf{V}i$ and $\mathbf{V}j$ contain m 1s, but since $m = \lceil n/2 \rceil - 1$, there must be a k $(1 \leq k \leq n)$ such that $\mathbf{V}i(k) = \mathbf{V}j(k) = 0$. Also, since $\mathbf{V}i$ and $\mathbf{V}j$ are different code words, the positions of the 1s in $\mathbf{V}i$ must differ from those in $\mathbf{V}j$ in at least one position. Hence, there must be a p $(1 \leq p \leq n)$ such that $\mathbf{V}i(p) = 0$ and $\mathbf{V}j(p) = 1$. Similarly, there must be a q $(1 \leq q \leq n)$ such that $\mathbf{V}i(p) = 1$ and $\mathbf{V}j(p) = 0$. Hence, all patterns 00, 01, 10 and 11 are applied to b_i and b_j. *Q.E.D.* The right part of Table 3 shows the ranges for B and the corresponding values for C_m^n and t, whereby $D = n + 1$ and $t = 2 * D(*n/B)$.

Table 4 shows a test time comparison between Dekker's method, the Improved method (Impr.), and the Optimal method (Optim.) (for the case that the word-wide BOM test covers the (0,0) and the (1,1) cases), for some values of B. Th % column shows the test time improvement of the optimal over the improved method. In Section 3 it has already been shown that Dekker's method is inferior as compared with the Improved method, especially because it heavily depends on the used BOM algorithm. Table 3 shows that an improvement of 20 to 38% has been obtained over the latest published methods.

5 Conclusions

Most memories are word-wide; i.e., $B > 1$. Memory test algorithms are traditionally designed for bit-oriented memories (BOMs), for which $B = 1$. These algorithms will not detect coupling faults between cells of the same word, called intra-word CFs. Many types of intra-word CFs exist, however, due to the likelihood of bridging defects, state CFs (CFsts) are considered to be the most important fault type. The detection of CFsts requires that all four states of any two cells within a word be verified to be possible. This is done using a set of data backgrounds (DBs). A new method, based on m-out-of-n codes, has been presented to derive an optimal set of DBs. In addition, it has been shown that the problem of detecting inter-word CFs and intra-word CFs should be separated, to allow for minimal test time. The new method allows for a reduction of 20 to about 40 % in test time for intra-word CFsts.

The question still remains as to when this test should be applied, as it depends on the topological organization of the memory [2, 3]. In case of an adjacent memory organization, the cells of a word are physically adjacent; this typically will be the case for small SRAMs (e.g., 64 words of 32 bits). Then the introduced test fully applies. In case of an interleaved, or also called folded, memory organization (e.g., this will be the case for an SRAM of 1024 words * 4-bits), the B bits of a word are physically separated by bits of other words, the introduced test only partially applies: It has to be performed on a single word of each fold to insure the absence of CFsts in the paths to/from the I/O data lines. In case of a sub-array organization (e.g., a 1M*4 DRAM may consist of 4 independent 1M*1 sub-arrays) the introduced test only has to be applied to one word of each sub-array to insure the absence of CFsts in the paths to/from the I/O data lines.

References

[1] A.J. van de Goor, "Testing Semiconductor Memories: Theory and Practice", *ComTex Publishing, Gouda, The Netherlands*, 1998. http://ce.et.tudelf.nl/~vdgoor/

[2] A.J. van de Goor and I.B.S. Tlili, "March Tests for Word- Oriented Memories", *In Proc. Design Automation and Test in Europe (DATe98)*, Paris, pp. 501-508, 1998

[3] A.J. van de Goor et al., "Converting March Tests for Bit-Oriented Memories into Tests for Word-Oriented Memories". *In Records of the IEEE Int. Workshop on Memory Technology, Design and Testing*, San Jose, CA, pp. 46-52, 1998

[4] R. Dekker et al., "Fault Modeling and Test Algorithm Development for Static Random Access Memories", *In Proc. of the IEEE Int. Test Conf.*, pp. 343-352, 1988

[5] R.P. Treuer and V.K. Agrawal, "Fault Location Algorithms for Repairable Embedded RAMs", *In Proc. of the IEEE Int. Test Conf.*, pp. 825-834, 1993

[6] M.S. Abadir and J.K. Reghbati, "Functional Testing of Semiconductor Random Access Memories", *ACM Computing Surveys*, **15**(3), pp. 175-198, 1983

[7] T.R.N. Rao and E. Fujiwara, "Error-Control Coding for Computer Systems", *Prentice-Hall International Editions, Englewood Cliffs, NJ*, 1989

[8] D.A. Anderson, "Design of Self-Checking Digital Networks Using Coding Techniques", *Coordinated Sci. Lab. Report R527. Champaign: University of Illinois Press*, 1971

[9] A.J. van de Goor and G. Gajdadjiev, "March LR: A memory test for realistic linked faults", *In Proc. of the* 14^{th} *IEEE VLSI Test Symposium (VTS'96)*, pp. 272-280, 1996