March tests for word-oriented memories

A.J. van de Goor   I.B.S Tlili

Delft University of Technology
Faculty of Information Technology and Systems
Section Computer Architecture & Digital Technique
Mekelweg 4, 2628 CD Delft, The Netherlands
E-mail : vdgoor@duteca.et.tudelft.nl

Abstract

Most memory test algorithms are optimized tests for a particular memory technology and a particular set of fault models, under the assumption that the memory is bit-oriented; i.e., read and write operations affect only a single bit in the memory. Traditionally, word-oriented memories have been tested by repeated application of a test for bit-oriented memories whereby a different data background (which depends on the used intra-word fault model) is used during each iteration. This results in time inefficiencies and limited fault coverage.

A new approach for testing word-oriented memories is presented, distinguishing between inter-word and intra-word faults and allowing for a systematic way of converting tests for bit-oriented memories to tests for word-oriented memories. The conversion consists of concatenating the bit-oriented test for inter-word faults with a test for intra-word faults. This approach results in more efficient tests with complete coverage of the targeted faults. Because most memories have an external data path which is wider than one bit, word-oriented memory tests are very important.

Keywords: *Bit-oriented memories, word-oriented memories, memory tests, march tests, data backgrounds, fault models.*

1 Introduction

Word-oriented memories (WOMs) contain more then one bit per word; i.e., $B \geq 2$, whereby B represents the number of bits per word and usually is a power of two. Read operations read the B bits simultaneously, write operations write data into the B bits whereby the data to be written into each cell can be specified independently of the data for the other cells.

Traditionally, WOMs have been tested by repeated application of a test for *bit-oriented memories (BOMs)*, whereby a different data background (which depends on the used fault model for the faults between bits in a word) is used during each iteration. Traditional march tests for WOMs to detect *coupling faults (CFs)* [1, 3, 5] have used different types of *data backgrounds (DBs)* such as the Walking 1/O, the Marching 1/0 and the Bridging Fault (BF) patterns. The disadvantages of using these DBs are: (1) test time inefficiency, and (2) limited fault coverage for CFs.

This paper presents a new systematic way to convert march tests for BOMs to WOMs. The conversion consists of concatenating to the march test for *inter-word faults* (faults between words) a march test designed for *intra-word faults* (faults within words). For the construction of the test for intra-word faults a minimal *data background sequence (DBS)*, capable of sensitizing the targeted CFs, has to be established. This paper presents this technique for the idempotent CF (CFid), the disturb CF (CFdst) [2] and the state CF (CFst) intra-word fault models. Thereafter, a method of constructing the intra-word march test, given the DBS, is presented.

This paper is organized as follows. Section 2 describes the organization and fault models for WOMs; Section 3 describes WOM march tests for single-cell faults. Section 4 derives data background sequences for intra-word coupling faults; while a method to convert BOM march tests to WOM march tests for intra-word CFs is given in Section 5. The paper ends with conclusions in Section 6.

2 Organization and fault models of word-oriented memories

Most memories are WOMs with an external data path of B-bits, whereby for stand alone memories B may be 4, 8 or 16; for embedded (cache) memories B may even be 256 or more. This emphasises the importance of efficient WOM tests with a good fault coverage.

WOMs can be organized internally in many different ways (depending on where the B bits are physically located within a row of the memory cell array): (1) *adjacent* (the row consists of some number (w) of B physically adjacent data bits); (2) *Interleaved* (the B bits of a word are physically separated by $w-1$ bits of the other words within the row). This paper addresses march tests for WOMs for the case whereby the B bits are physically adjacent within a row; march tests for the interleaved case are a subset of the adjacent case [4].

The fault models for WOMs can be divided into the following classes:

1. *Single-cell faults*:
   These are the classical stuck-at faults (SAFs), transition faults (TFs) and data retention faults (DRFs).
2. *Faults between memory cells*:
   This class of faults consists of coupling faults (CFs). They can be further divided into the subclasses:
   a. *Inter word faults*: These faults are the classical CFs whereby the aggressor and victim cells belong to different words. Classical BOM tests are based on this subclass.
   b. *Intra-word faults*: These are CFs whereby the aggressor and victim cells belong to the same word. BOM tests have to be converted to be able to detect faults of this subclass.

3 WOM march tests for single-cell faults

SAFs, TFs and DRFs involve only a single cell. The fact that the memory word is B bits wide does not influence the detectability of these faults. March tests for BOMs can be converted to march tests for WOMs by taking into account that in the BOM tests, the `$r$0', `r1', `w0' and `w1' operations are applied to a single bit. In case of WOMs, an entire word of B bits has to be read or written; the data value of this word is called the *data background (DB)* [1].

When the fault model for faults within a word is considered to consist of single-cell faults (SAFs, TFs and DRFs), any BOM test can be converted into a WOM test as follows:

1. The `w0' operation should be replaced with a *`w-data background'* denoted as `$w_D$', whereby *any* data background is acceptable. For example: `w0...0' (an all 0s data background), `w01...01' (a data background pattern of a repeated sequence of `01'), etc.; whereby the length of the bit string is B bits. The `w1' operation should be replaced with a write operation which writes the *inverted data background*; i.e., `$w_{\bar{D}}$'.
2. The `r0' and `r1' operations should be replaced with the operations *`r-data background'* (`r_D') and `r-inverted-data background' (`$r_{\bar{D}}$').
3. In the equations expressing the required number of operations for a test, n (representing the number of cells in the chip) has to be replaced by n/B (representing the number of words in the chip).

Converting the BOM MATS+ test $\{\Updownarrow(w0);\Uparrow(r0,w1);\Downarrow(r1,w0)\}$ to a WOM test with B = 4, using the DB `0101' results in the test : $\{\Updownarrow(w0101);\Uparrow(r0101,w1010);\Downarrow(r1010,w0101)\}$.

4 Data background sequences (DBS) for intra-word coupling faults (CFs)

In order to detect intra-word CFs, a BOM march test could be converted to a WOM march test by replacing the bit-wide `r0', `r1', `w0' and `w1' operations with operations which read and write a data background of B bits. This will result in the following fault coverage for WOM chips:

a. Inter-word CFs will be detected because the word-wide organization of the memory chip will not influence the fault coverage.

b. Intra-word CFs may or may not be detectable, depending on the dominance of the write operation on the CF.

   1. If the write operation dominates the CF (i.e., the value specified in the write operation will be stored in the cells), the CF will have no effect and therefore appears not to be present. A test to detect this non-dominating CF is therefore not required.
   2. If the CF dominates the write operation, such that the CF will manifest itself, it is required to detect the CF. In the case of idempotent CFs (CFids), disturb CFs (CFdsts) [2], and state CFs (CFsts) [1] the fault will not be detectable with algorithms for BOMs as shown below.

Figure 1 shows a 4-cell memory word (C_w, C_x, C_y and C_z), and the CFid whereby C_z is $<\uparrow;\uparrow>$ coupled to C_w (Note: The notation for the CFid $<\uparrow;\uparrow>$ means that an $\uparrow$ transition write operation applied to the aggressor cell causes an $\uparrow$ transition in the victim cell). Assuming that the memory word contains a `0...0' value, a `w1111' operation could be used to sensitize the CFid; however this would mask the CFid. The detection of this fault requires that the CFid is sensitized by the following write `w1$xy$0' whereby x and y may have arbitrary values, such that the CFid is not masked.

In the next subsections the required *data background sequences (DBSs)* for intra-word CFids, CFdsts and CFsts are derived.

4.1 DBS for intra-word inversion CFs

In the case of an inversion CF (CFin), the fault will be detectable with the algorithms for BOMs, converted to WOMs using *any* data background, because the value in the victim cell will be the inverse of the value expected by the read operation due to the CFin.

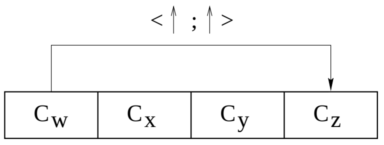


Figure 1: Intra-word CFid

4.2 DBS for intra-word idempotent CFs

In order to be able to detect intra-word CFids, for the case that the CFid dominates the write operation, the algorithms for BOMs have to be modified more rigorously. For the CFids within a B-bit memory word four subtypes exist : $<\uparrow;\downarrow>_{c_i,c_j}$ (Note: c_i denotes that cell-i is the aggressor cell and c_j denotes that cell-j is the victim cell), $<\uparrow;\uparrow>_{c_i,c_j}$, $<\downarrow;\downarrow>_{c_i,c_j}$ and $<\downarrow;\uparrow>_{c_i,c_j}$ where $i,j \in \{0,1,...,B-1\}$. Each subtype has $C_B^2 = B*(B-1)$ possible cases because any of the B cells can be the aggressor, while any of the $B-1$ non-aggressor cells can be the victim; the total number of CFids is therefore $4*B*(B-1)$. The purpose of this section is to find the minimal DBS which can sensitize all $4*B*(B-1)$ CFids.

Figure 2 shows the state diagram for sensitizing the CFids within a 2-bit WOM. The states (nodes) are numbered according to the value of the two cells c_1 and c_2 in the word; the arcs (which represent the transition write operations) are labeled with the sensitized faults. From Figure 2 we can see that:

1. A given CFid can be sensitized by different arcs (transition write operations). E.g., the CFid $<\uparrow;\uparrow>_{c_2,c_1}$ is sensitized by both arcs (S_{00},S_{01}) and (S_{10},S_{01}).
2. Each of the arcs formed by the states which are each others inverse; i.e., $(S_{00},S_{11}),(S_{11},S_{00}),(S_{01},S_{10})$ and (S_{10},S_{01}), is labeled with two CFids, furthermore those four arcs can sensitize all eight CFids.

The above means that the state diagram of Figure 2 can be simplified to that of Figure 3, where only four arcs are needed to sensitized all CFids. A fifth arc; e.g., (S_{00},S_{01}), is needed to connect the two pairs of arcs. The DBS for a 2-bit word is: $\mathbf{S}_2$ = 00, 11, 00, 01, 10, 01.

Extending the data background sequence to WOMs with 8-bit words, $W_8 = \{c_0,c_1,c_2,c_3,c_4,c_5,c_6,c_7\}$, requires the following steps:

1. *Level 0*: For each cell-pair (c_i,c_{i+1}) we apply the data background sequence $\mathbf{S}_2$ found for 2-bit words. This can be done by applying the sequence $\mathbf{S}_8^0$ shown in Table 1. From Table 1 we can see that all CFids between $(c_i,c_{i+1+k*2})$ are sensitized, where $k \in \{0,1...,\lfloor((B-1)-(i+1))/2\rfloor\}$; this is also shown in Figure 4, where the arc between two nodes (cells) c_i and c_j implies that all CFids between cells c_i and c_j are sensitized.
2. *Level 1*: For each cell-pair (c_i,c_{i+2}) we apply only the data background sequence $\mathbf{S''}_2$ = 01, 10, 01 ; this is sufficient because the sequence $\mathbf{S'}_2$ = 00, 11, 00 has already been applied in *Level 0* . As we can see from Table 2 and Figure 5 all CFids between $(c_i,c_{i+2+k*4})$ are sensitized, where $k \in \{0,1...,(\lfloor((B-1)/2-(i+1))/2\rfloor)\}$.
3. *Level 2*: For each cell-pair (c_i,c_{i+4}) we apply the data background sequence $\mathbf{S''}_2$. From Table 3 and Figure 6, we can see that all CFids between $(c_i,c_{i+4+k*8})$ are sensitized, where $k \in \{0,1...,(\lfloor((B-1)/4-(i+1))/2\rfloor)\}$.

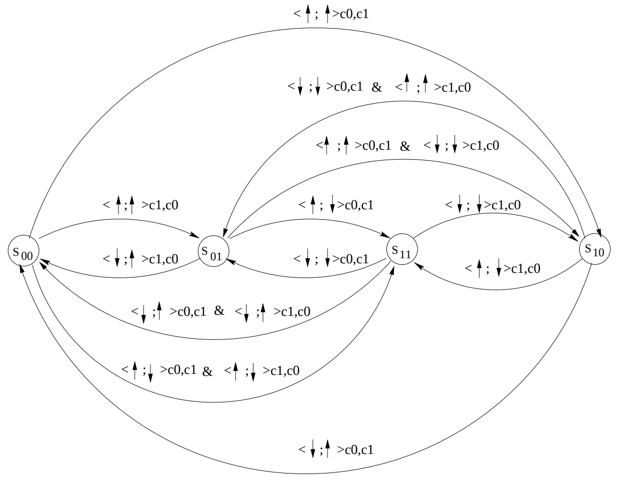
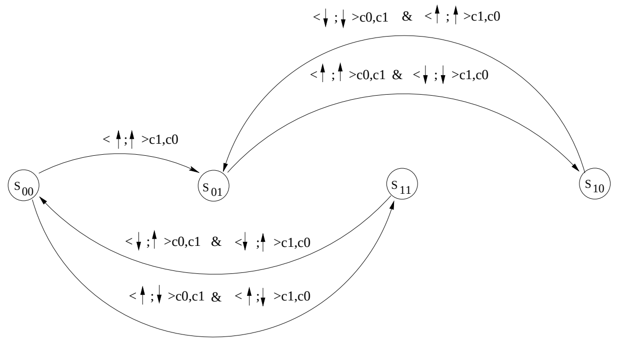


Figure 2: State diagram for sensitizing CFids (B = 2)



<↓ ;↓ >c0,c1 & <↑ ;↑ >c1,c0
<↑ ;↑ >c0,c1 & <↓ ;↓ >c1,c0
< ↑;↑ >c1,c0
S 00
S 01
S 11
S 10
<↓ ;↑ >c0,c1 & <↓ ;↑ >c1,c0
<↑ ;↓ >c0,c1 & <↑ ;↓ >c1,c0

Figure 3: Minimized state diagram for sensitizing (B = 2)

After *Level 2*, all CFids for an 8-bit WOM are sensitized. Table 4 shows the DBS $\mathbf{S}_8$ for 8-bit WOMs, it includes all three levels discussed above. The method of generat-

Table 1: DBS $\mathbf{S}_8^0$ at *Level 0*

#	State
	$c_0c_1c_2c_3c_4c_5c_6c_7$
0	0 0 0 0 0 0 0 0
1	1 1 1 1 1 1 1 1
2	0 0 0 0 0 0 0 0
3	0 1 0 1 0 1 0 1
4	1 0 1 0 1 0 1 0
5	0 1 0 1 0 1 0 1

Table 2: DBS $\mathbf{S}_8^1$ at *Level 1*

#	State	
	$c_0c_2c_4c_6$	$c_1c_3c_5c_7$
0	0 1 0 1	0 1 0 1
1	1 0 1 0	1 0 1 0
2	0 1 0 1	0 1 0 1

Table 3: DBS $\mathbf{S}_8^2$ at *Level 2*

#	State			
	c_0c_4	c_2c_6	c_1c_5	c_3c_7
0	0 1	0 1	0 1	0 1
1	1 0	1 0	1 0	1 0
2	0 1	0 1	0 1	0 1

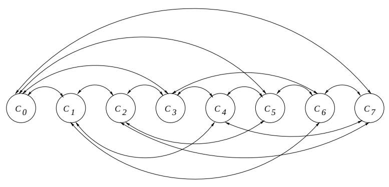


Figure 4: Graph for sensitizing CFids at *Level 0*

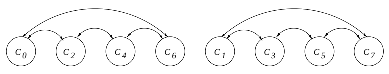


Figure 5: Graph for sensitizing CFids at *Level 1*

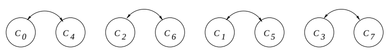


Figure 6: Graph for sensitizing CFids at *Level 2*

ing a DBS for 8-bit WOMs can be generalized for B-bit WOMs, where the *number of data backgrounds* needed to sensitize all CFids (denoted by (d)) within a word is: 6 $\{$the d used in *Level 0*$\}$ + 3 $\{$ the d used in each additional level$\}$ $*(\lceil\log_2 B\rceil - 1)$ $\{$the number of levels$\}$ = 3 + 3 $*\lceil\log_2 B\rceil$. The test length to detect all CFids within a B-bit word is then : 2 (write and read operations) $*d$ = 6 + 6 $*\lceil\log_2 B\rceil$. Historically, different DBSs have been used to detect CFs in WOMs; three commonly types of DBSs are used: the Walking 1/0 , the Marching 1/0 [3], and the Bridging fault (BF) DBS [1]. [4] has shown that the traditional DBSs do not cover all targeted faults and/or are less time efficient.

4.3 DBS for intra-word disturb CFs

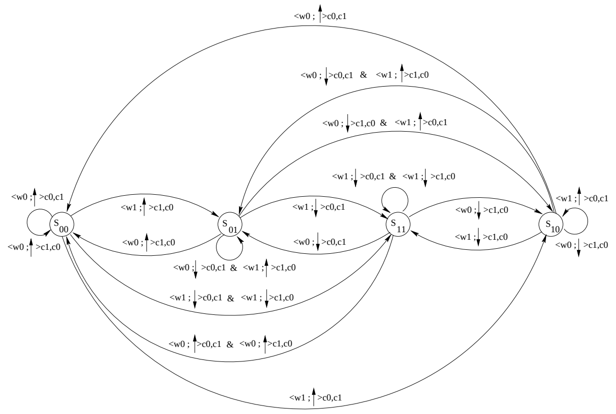
In order to be able to detect intra-word CFdsts, for the case that the CFdst dominates the write operation, the algorithms for BOMs have to be modified; similar to the way done for CFids (see Section 4.2). In the case of CFdsts, eight CFdsts subtypes exist (van de Goor, 1996): $< r0;\uparrow>_{c_i,c_j}, < r0;\downarrow>_{c_i,c_j}, < r1;\uparrow>_{c_i,c_j}, < r1;\downarrow>_{c_i,c_j}$ $, < w0;\uparrow>_{c_i,c_j}, < w0;\downarrow>_{c_i,c_j}, < w1;\uparrow>_{c_i,c_j}$ and $< w1;\downarrow>_{c_i,c_j}$; where $i,j \in \{0,1,...,B-1\}$. Each subtype has $C_B^2 = B*(B-1)$ possible cases; the total number of CFdsts is therefore $8*B*(B-1)$. Similar to the CFid case, we have to find the minimal DBS which can sensitize the $8*B*(B-1)$ CFdsts. Figure 7 shows the state diagram for sensitizing the CFdsts (only for write operations) within a 2-bit WOM. The states (nodes) are numbered according to the value of the two cells c_1 and c_2 in the word; the arcs (which represent the transition and

Table 4: DBS $\mathbf{S}_8$ for an 8-bit memory word

#	State	Level
	$c_0c_1c_2c_3c_4c_5c_6c_7$	
0	0 0 0 0 0 0 0 0	0
1	1 1 1 1 1 1 1 1	0
2	0 0 0 0 0 0 0 0	0
3	0 1 0 1 0 1 0 1	0
4	1 0 1 0 1 0 1 0	0
5	0 1 0 1 0 1 0 1	0
6	0 0 1 1 0 0 1 1	1
7	1 1 0 0 1 1 0 0	1
8	0 0 1 1 0 0 1 1	1
9	0 0 0 0 1 1 1 1	2
10	1 1 1 1 0 0 0 0	2
11	0 0 0 0 1 1 1 1	2

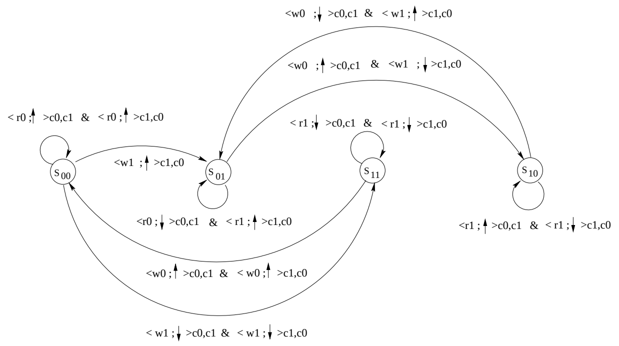
non-transition write operations) are labeled with the sensitized faults. Only write operations can change the state of a faultfree memory, which means that the DBS depends only on the write operations; therefore only the write operations are considered in this figure. From Figure 7 we can see that:

1. A given CFdst can be sensitized by different arcs (transition or non-transition write operations). E.g., the CFdst $< w0;\uparrow>_{c_1,c_2}$ can be sensitized by three arcs : (S_{00}, S_{00}), (S_{10}, S_{00}) and (S_{11}, S_{00}).



<w0 ; ↑>c0,c1
<w0 ; ↓>c0,c1 & <w1 ; ↑>c1,c0
<w0 ; ↓>c1,c0 & <w1 ; ↑>c0,c1
<w1 ; ↓ >c0,c1 & <w1 ; ↓ >c1,c0
<w0 ; ↑ >c0,c1
<w1 ; ↑ >c1,c0
<w1 ; ↓ >c0,c1
<w0 ; ↓ >c1,c0
<w1 ; ↑ >c0,c1
S00
S01
S11
S10
<w0 ; ↑ >c1,c0
<w0 ; ↑>c1,c0
<w0 ; ↓>c0,c1
<w1 ; ↓ >c1,c0
<w0 ; ↓ >c1,c0
<w0 ; ↓ >c0,c1 & <w1 ; ↑ >c1,c0
<w1 ; ↓>c0,c1 & <w1 ; ↓>c1,c0
<w0 ; ↑>c0,c1 & <w0 ; ↑>c1,c0
<w1 ; ↑ >c0,c1

Figure 7: State diagram for sensitizing CFdsts (only write operations) within a 2-bit word



<w0 ; ↓ >c0,c1 & < w1 ; ↑ >c1,c0
<w0 ; ↑ >c0,c1 & <w1 ; ↓ >c1,c0
< r0 ; ↑ >c0,c1 & < r0 ; ↑ >c1,c0
< r1 ; ↓ >c0,c1 & < r1 ; ↓ >c1,c0
<w1 ; ↑ >c1,c0
S00
S01
S11
S10
<r0 ; ↓ >c0,c1 & < r1 ; ↑ >c1,c0
<r1 ; ↑ >c0,c1 & < r1 ; ↓ >c1,c0
<w0 ; ↑ >c0,c1 & < w0 ; ↑ >c1,c0
< w1 ; ↓ >c0,c1 & < w1 ; ↓ >c1,c0

Figure 8: State diagram for sensitizing CFdsts (read and transition write operations) within a 2-bit word

2. Each of the arcs formed by the states which are each others inverse: $(S_{00}, S_{11}), (S_{11}, S_{00}), (S_{01}, S_{10})$ and (S_{10}, S_{01}) is labeled with two CFdsts.
3. All arcs begining and ending in same node (representing non-transition write operations) are labeled with two CFdsts.

From the above it follows that all faults can be sensitized in more than one way, which means that the state diagram depicted in Figure 7 can be simplified. In [4] two simplified state diagrams have been given; Figure 8 shows the state diagram based on transition write operations. It has the advantage of requiring only 9 arcs and also covers CFids as shown in [4]. From Figure 8 we can construct the data background sequence $\mathbf{S}_t$, which depends only on transition write operations:
$\mathbf{S}_t = S_{00}, S_{11}, S_{00}, S_{01}, S_{10}, S_{01}$. Using this state sequence, a sequence of *read and write operations* (Ω_t) can be generated assuming initial state S_{00}:
$\Omega_t = w11, r11, w00, r00, w01, r01, w10, r10, w01$. Thus the number of operations needed to *sensitize* all CFdsts using the data background sequence $\mathbf{S}_t$ is 9. To *detect* all CFdsts using the operation sequence Ω_t, each of the above sensitizing write or read operations has to be followed by a read operation; this read operation detects the CFdst sensitized by the preceding write or read operation. The operation sequence is then:
$\Omega_t = w11, r11, r11, r11, w00, r00, r00, r00, w01, r01, r01, r01, w10, r10, r10, r01, w01, r01$. Ω_t now contains sequences of three identical read operations. The first read operation is required to detect faults sensitized by the preceding write operation; the second read operation is required to detect the faults sensitized by the preceding read operation; hence, the third read operation is redundant and can be removed. The 9^{th} operation $w01$ is needed only to connect the data backgroud sequence (see Figure 8), thus there is no need to check this write operation. The sequence $r01, r01, r01$ can therefore be removed; however, in order to detect CFdsts sensitized by the $r01$ operation, that operation has to be followed by a $r01$. The operation sequence Ω_t can now be simplified to:
$\Omega_t = w11, r11, r11, w00, r00, r00, w01, w10, r10, r10, w01, r01, r01$. The number of operations needed to detect all the CFdsts for a 2-bit WOM is now 13.

Extending the 2-bit DBS to a DBS for B-bit words, we can use the steps used for the CFids :

1. *Level 0*: For each cell-pair (c_i, c_{i+1}), we apply the DBS found for 2-bit words. This can be done by applying the sequence $\mathbf{S}_t$. All CFdsts between $(c_i, c_{i+1+k*2})$ are sensitized, where $k \in \{0, 1..., \lfloor((B-1)-(i+1))/2\rfloor\}$.
2. *Level 1*: For each cell-pair (c_i, c_{i+2}), we apply only the DBS $\mathbf{S}''_2$ = 01, 10, 01 ; this is sufficient because the sequence $\mathbf{S}'_2$ = 00, 11, 00 has already been applied in *Level 0* . All CFdsts between $(c_i, c_{i+2+k*4})$ are sensitized, where $k \in \{0, 1..., (\lfloor((B-1)/2-(i+1))/2\rfloor)\}$.
3. *Level 2*: For each cell-pair (c_i, c_{i+4}), we apply the DBS $\mathbf{S}''_2$. All CFdsts between $(c_i, c_{i+4+k*8})$ are sensitized, where $k \in \{0, 1..., (\lfloor((B-1)/4-(i+1))/2\rfloor)\}$.

...

$\log_2 B$. *Level* $\log_2 B - 1$: For each cell-pair $(c_i, c_{i+2^{\log_2 B-1}})$, we apply the DBS $\mathbf{S}''_2$. All CFdsts between $(c_i, c_{i+2^{\log_2 B-1}+k*2^{\log_2 B}})$ are sensitized, where $k \in \{0, 1..., (\lfloor((B-1)/(2^{\log_2 B-1})-(i+1))/2\rfloor)\}$.

Extending the *operation sequence* to detect all CFdsts in B-bit WOMs requires the following steps:

1. *Level 0*: For each cell-pair (c_i, c_{i+1}), we generate the operation sequence found for 2-bit words:

Table 5: DBS for a 4-bit word

#	State	Level
	$c_0c_1c_2c_3$	
0	0 0 0 0	0
1	1 1 1 1	0
2	0 0 0 0	0
3	0 1 0 1	0
4	1 0 1 0	0
5	0 1 0 1	0
6	0 0 1 1	1
7	1 1 0 0	1
8	0 0 1 1	1

Table 6: Operation sequence for a 4-bit word

#	Operation	Data	Level	#	Operation	Data	Level
		$c_0c_1c_2c_3$				$c_0c_1c_2c_3$	
0		0 0 0 0	0	10	r	1 0 1 0	0
1	w	1 1 1 1	0	11	w	0 1 0 1	0
2	r	1 1 1 1	0	12	r	0 1 0 1	0
3	r	1 1 1 1	0	13	r	0 1 0 1	0
4	w	0 0 0 0	0	14	w	0 0 1 1	1
5	r	0 0 0 0	0	15	w	1 1 0 0	1
6	r	0 0 0 0	0	16	r	1 1 0 0	1
7	w	0 1 0 1	0	17	r	1 1 0 0	1
8	w	1 0 1 0	0	18	w	0 0 1 1	1
9	r	1 0 1 0	0	19	r	0 0 1 1	1
				20	r	0 0 1 1	1

$\Omega_t = w11, r11, r11, w00, r00, r00, w01, w10, r10, r10, w01, r01, r01$. All CFdsts between $(c_i, c_{i+1+k*2})$ are detected, where $k \in \{0, 1..., \lfloor((B-1)-(i+1))/2\rfloor\}$.

2. *Level 1*: For each cell-pair (c_i, c_{i+2}), we apply only the operation sequence:
$\Omega''_t = w01, w10, r10, r10, w01, r01, r01$; this is sufficient because the operation sequence: $\Omega'_t = w11, r11, r11, w00, r00, r00$ has already been applied in *Level 0*. The first operation $w01$ in Ω''_t does not have to be followed by a read operation, because it is used only to connect the two *Level 0* and *Level 1* sequences. All CFdsts between $(c_i, c_{i+2+k*4})$ are detected, where $k \in \{0, 1..., (\lfloor((B-1)/2-(i+1))/2\rfloor)\}$.
3. *Level 2*: For each cell-pair (c_i, c_{i+4}), we apply the operation sequence Ω''_t . All CFdsts between $(c_i, c_{i+4+k*8})$ are detected, where $k \in \{0, 1..., (\lfloor((B-1)/4-(i+1))/2\rfloor)\}$.
...

$\log_2 B$. *Level* $\log_2 B - 1$: For each cell-pair $(c_i, c_{i+2^{\log_2 B-1}})$, we apply the operation sequence Ω''_t. All CFdsts between $(c_i, c_{i+2^{\log_2 B-1}+k*2^{\log_2 B}})$ are detected, where $k \in \{0, 1..., (\lfloor((B-1)/(2^{\log_2 B-1})-(i+1))/2\rfloor)\}$.

Table 5 and Table 6 show respectively the DBS and the operation sequence for 4-bit words, requiring only *Level 0* and *Level 1* operations as discussed above. The required number of DBs (d) to sensitize all CFdsts within B-bit words is: 6 {the d used in *Level 0*} + 3 {the d used in the other levels} $*(\lceil\log_2 B\rceil - 1)$ { the number of levels } = 3 + 3 $*\lceil\log_2 B\rceil$; which is identical to the d of CFids. The number of operations needed to detect all CFdsts within a B-bit words is: 13 {the number of operations used in *Level 0*} + 7 {the number of operations used in the other levels} $*(\lceil\log_2 B\rceil - 1)$ = 6 + 7 $*\lceil\log_2 B\rceil$.

4.4 DBS for intra-word state CFs

In order to be able to detect CFsts between cells in a word, all states of two arbitrary cells i and j should be checked; i.e. the states: $(i,j) \in$ (0,0), (0,1), (1,0), (1,1). Initially only the states (0,0) and (0,1) have to be checked, the other two states will be checked when the algorithm is executed with the inverted data backgrounds. The number DBs required for checking the states (0,0) and (0,1) is $d = \lceil\log_2 B\rceil$ + 1 [1]. If B is a power of 2 this will become: $d = \log_2 B$ + 1. In [1] a method for constructing the DBs for CFsts has been given. For a memory with B = 8 (a byte-wide memory) the DBs of Table 7 can be used. Note that because CFsts are only state, rather than transition dependent, the DBs can be applied in *any* sequence.

Table 7: 8-bit DBs for CFsts

#	Data background	
	Normal	Inverse
0	00000000	11111111
1	01010101	10101010
2	00110011	11001100
3	00001111	11110000

5 WOM march tests for intra-word CFs

Any given BOM march test can be converted to a WOM test which additionally covers intra-word CFs (i.e., CFids, CFdsts and CFsts). Such a WOM march test is a concatenation of two march tests: $\{$*inter-word* march test$\}\{$*intra-word* march test$\}$. The *inter-word* march test consists of a traditional BOM test (such as MATS+ or March C−), modified such that the bit-operations `r0', `r1', `w0' and `w1' are replaced with the word-operations `$r_D$', `$r_{\bar{D}}$', `$w_D$' and `$w_{\bar{D}}$'; whereby any data background value can be chosen for D. The *intra-word* march test is used to detect the intra-word CFs. It consists of a single march element with the following form:

1. For CFsts: $\Updownarrow (w_{D_0}, r_{D_0}, ..., w_{D_{d-1}}, r_{D_{d-1}})$, whereby D_0 through D_{d-1} are taken from the set of DBs from Table 7 (for $B = 4$), in such a way that both the normal and inverse values are covered. Note that the DBs can be applied in any order.
2. For CFids: $\Updownarrow (w_{D_0}, r_{D_0}, w_{D_1}, r_{D_1}, ..., w_{D_{d-1}}, r_{D_{d-1}})$, whereby D_0 through D_{d-1} represent the DBS of Table 4 (for $B = 4$).
3. For CFdsts: $\Updownarrow (w_{D_0}, r_{D_0}, r_{D_0}, ..., w_{D_{d-1}}, r_{D_{d-1}}, r_{D_{d-1}})$, whereby the DBOS (consisting of the operations together with the DBs) is taken from Table 6 (for $B = 4$).

The above intra-word test may be modified as follows without any impact on the fault coverage:

1. Extra read operations may be added (for example to make the test more symmetric and/or to detect possible faults of other fault models).
2. The single march element may be divided into any number of march elements, and for each march element the adressing order can be chosen freely.

The above freedom to modify the intra-word test allows for the following:

1. Test time reduction
   If some march elements of the intra-word test can be made identical to these of the inter-word test, those intra-word march elements can be removed.
2. Extra fault coverage for unanticipated faults
   This can be optimized when the intra-word march test has the following properties:
   a. It consists of many march elements
      Because each march element performs a sweep over the memory.
   b. All march elements start with a read
      This allows for the detection of CFs.
   c. The adressing orders of the march element of the intra-word march test should vary as much as possible. This maximizes the probability of detecting dynamic faults such as write recovery faults [3].

Below, two examples are given to show how BOM march tests can be converted into optimized WOM march tests.

Figure 9 shows how to convert March C−: $\{\Updownarrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0); \Downarrow (r0, w1); \Downarrow (r1, w0); \Updownarrow (r0)\}$ such that intra-word CFids are detected for a 4-bit WOM. The inter-word march test consists of 6 march elements, while the intra-word march test is designed such that it consists of 9 march elements whereby the similarity with the march elements of the inter-word test has been maximized. An extra initial read operation has been included in the intra-word test, and the last march element M'_{14} is composed of only a single operation which is needed to check the write operation of the march element M'_{13}. Because B = 4, only the cell values c_0 through c_3 and a DBS of 9 DBs, numbered 0 through 8, of Table 4 are used. From Figure 9 one can see that the march elements M'_6 and M'_7 are redundant (M'_6 = M_3 and M'_7 = M_4), and march element M_5 can be deleted because the `r0000' in $M'_8$ can be used to check the `w0000' in M_4. The removal of march elements M_5, M'_6 and M'_7 does not change the fault coverage of the inter-word test; the fault coverage of the intra-word test is maintained because the DBS of Table 4 is still being applied correctly. The optimized version of the WOM march test based on March C− is shown in Figure 10. The test length changes from $10 * n$ (for the BOM test) to $22 * n/4$ for 4-bit WOM. In general, for a B-bit memory, this will be : $(10 + 6 * \lceil \log_2 B \rceil) * n/B$.

Figure 11 shows how to convert March LR $\{\Updownarrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0, r0, w1); \Uparrow (r1, w0); \Uparrow (r0, w1, r1, w0); \Updownarrow (r0)\}$ (van de Goor, 1996) such that intra-word CFdsts will be detected for a 4-bit WOM. The conversion can be made as follows:

1. The read operation $r0$ is added to march element M_2 and $r1$ to M_4; the result is the march test: $\{\Updownarrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0, r0, r0, w1); \Uparrow (r1, w0); \Uparrow (r0, w1, r1, r1, w0); \Updownarrow (r0)\}$. These two read operations do not affect the fault coverage of the BOM march test.
2. Convert the obtained BOM march test to a 4-bit inter-word march test.
3. Concatenate the inter-word march with the intra-word march test which is obtained by a sequence of march elements, each consisting of three operations `$r_{D_i}, w_{D_{i+1}}, r_{D_{i+1}}$', or four operations `$r_{D_i}, w_{D_{i+1}}, w_{D_{i+2}}, r_{D_{i+2}}$' (see Table 6). The last march element of the intra-word march test consists of a single read operation which is used to check the last read operation in the previous march element (see Figure 11).
4. Optimize the resulting WOM march test by deleting the redundant march elements of the intra-word test (e.g., the sequence of operations in M'_6 and M'_7 has

already been used in M_2 and M_4). The march element M_5 can be deleted, because the `r0000' in $M'_8$ can be used to check the `w0000' in M_4. The optimized version of the WOM march test based on March LR is shown in Figure 12.

The test length changes from $14 * n$ (for the BOM test) to $30 * n/4$ for a 4-bit WOM. In general, for a B-bit memory, this will be: $(16 + 7 * \lceil \log_2 B \rceil) * n/B$.

$$\{\underset{M_0}{\Updownarrow(w0000)};\underset{M_1}{\Uparrow(r0000,w11111)};\underset{M_2}{\Uparrow(r1111,w0000)};$$
$$\underset{M_3}{\Downarrow(r0000,w1111)};\underset{M_4}{\Downarrow(r1111,w0000)};\underset{M_5}{\Updownarrow(r0000)}\};$$
$$\{\underset{M'_6}{\Uparrow(r0000,w1111)};\underset{M'_7}{\Downarrow(r1111,w0000)};\underset{M'_8}{\Uparrow(r0000,w0101)};$$
$$\underset{M'_9}{\Downarrow(r0101,w1010)};\underset{M'_{10}}{\Uparrow(r1010,w0101)};\underset{M'_{11}}{\Downarrow(r0101,w0011)};$$
$$\underset{M'_{12}}{\Uparrow(r0011,w1100)};\underset{M'_{13}}{\Downarrow(r1100,w0011)};\underset{M'_{14}}{\Updownarrow(r0011)}\}$$

Figure 9: WOM march test based on March C−

$$\{\underset{M_0}{\Updownarrow(w0000)};\underset{M_1}{\Uparrow(r0000,w11111)};\underset{M_2}{\Uparrow(r1111,w0000)};$$
$$\underset{M_3}{\Downarrow(r0000,w1111)};\underset{M_4}{\Downarrow(r1111,w0000)}\};\{\underset{M'_5}{\Uparrow(r0000,w0101)};$$
$$\underset{M'_6}{\Downarrow(r0101,w1010)};\underset{M'_7}{\Uparrow(r1010,w0101)};\underset{M'_8}{\Downarrow(r0101,w0011)};$$
$$\underset{M'_9}{\Uparrow(r0011,w1100)};\underset{M'_{10}}{\Downarrow(r1100,w0011)};\underset{M'_{11}}{\Updownarrow(r0011)}\}$$

Figure 10: Optimized WOM march test based on March C−

$$\{\underset{M_0}{\Updownarrow(w0000)};\underset{M_1}{\Downarrow(r0000,w11111)};\underset{M_2}{\Uparrow(r1111,w0000,}$$
$$r0000,w1111);\underset{M_3}{\Uparrow(r1111,w0000)};\underset{M_4}{\Uparrow(r0000,w1111,}$$
$$r1111,w0000);\underset{M_5}{\Updownarrow(r0000)}\};\{\underset{M'_6}{\Uparrow(r0000,w1111,r1111)};$$
$$\underset{M'_7}{\Downarrow(r1111,w0000,r0000)};\underset{M'_8}{\Uparrow(r0000,w0101,w1010,r1010)};$$
$$\underset{M'_9}{\Downarrow(r1010,w0101,r0101)};\underset{M'_{10}}{\Uparrow(r0101,w0011,w1100,r1100)};$$
$$\underset{M'_{11}}{\Downarrow(r1100,w0011,r0011)};\underset{M'_{12}}{\Uparrow(r0011)}\}$$

Figure 11: WOM march test based on March LR

6 Conclusions

The paper has presented a systematic way for constructing march tests for word-oriented memories; starting from march tests for bit-oriented memories. The fault models for inter-word and intra-word faults can be chosen independently. An optimal set of data backgrounds and

$$\{\underset{M_0}{\Updownarrow(w0000)};\underset{M_1}{\Downarrow(r0000,w11111)};\underset{M_2}{\Uparrow(r1111,w0000,}$$
$$r0000,r0000,w1111);\underset{M_3}{\Uparrow(r1111,w0000)};\Uparrow(r0000,w1111,$$
$$\underset{M_4}{r1111,r1111,w0000)}\};\{\underset{M'_5}{\Uparrow(r0000,w0101,w1010,r1010)};$$
$$\underset{M'_6}{\Downarrow(r1010,w0101,r0101)};\underset{M'_7}{\Uparrow(r0101,w0011,w1100,r1100)};$$
$$\underset{M'_8}{\Downarrow(r1100,w0011,r0011)};\underset{M'_9}{\Uparrow(r0011)}\}$$

Figure 12: Optimized WOM march test based on March LR

operation sequences has been derived for each of the intra-word CFs (CFids, CFdsts and CFsts). It has been shown that a word-oriented test can be obtained from a bit-oriented test by concatenating the bit-oriented test with a set of march elements which detect the intra-word CFs. Optimization techniques are presented to reduce the number of march elements of the concatenated intra-word test.

Examples for 4-bit word-oriented memories are given which show that a word-oriented version of March C− requires $10*n$ operations when B = 1 and $22*n/4 = 5,5*n$ operations when B = 4. Similary, March LR has been converted to cover intra-word CFdsts whereby the test length has been changed from $14 * n$ for B = 1 to $30 * n/4 = 7,25 * n$ for B = 4.

The traditional WOM tests are based on repeating the BOM test with a sequence of data backgrounds, while the proposed method extends the BOM test with an intra-word test. [4] has shown that the traditional method is therefore less time efficient and/or does not cover all targeted intra-word faults.

References

[1] Dekker, R. (1990). *"A Realistic Fault Model and Test Algorithms for Static Random Access Memories"*, IEEE Trans. on computers **C-9)** (6), pp. 567 - 572.

[2] van de Goor, A.J. (1996) *"March LR: A test for Realistic Linked Faults"*, In Proceedings of the 14^{th} VLSI Test Symposium-1996, pp. 272 - 280.

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

[4] Tlili, I.B.S. and van de Goor, A.J. (1997). *"Tests for word-oriented memories"* Technical Report No. 1-68340-44(1997)08, Delft University Technology, Department of Electrical Engineering, Delft, The Netherlands.

[5] Treuer, R.P. and Agarwal V.K. (1993). *"Fault Location Algorithms for Repairable Embedded RAMs"*, In Proceedings of the ITC 1993, pp. 825 - 834.