JOURNAL OF ELECTRONIC TESTING: Theory and Applications 16, 487–498, 2000

Testing Address Decoder Faults in Two-Port Memories: Fault Models, Tests, Consequences of Port Restrictions, and Test Strategy

SAID HAMDIOUI
Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95052, USA
said@cardit.et.tudelft.nl

AD J. VAN DE GOOR
Delft University of Technology, Faculty of Information Technology and Systems, Section of Computer Architecture and Digital Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
vdGoor@cardit.et.tudelft.nl

Received June 7, 1999; Revised December 22, 1999

Editor: S. Demidenko

Abstract. A two-port memory contains two duplicated sets of address decoders, which operate independently. Testing such memories requires the use of single-port tests as well as special two-port tests; the test strategy determines which tests have to be used. Many two-port memories have ports which are read-only or write-only; this impacts the possible tests for single-port and two-port memories, as well as the test strategy. In this paper the effects of interference and shorts between the address decoders of the two ports on the fault modeling are investigated. Fault models and their tests are introduced. In addition, the consequences of the port restrictions (read-only or write-only ports) on the fault models and tests are discussed, together with the test strategy.

Keywords: multi-port memories, single-port memories, fault models, address decoder faults, march tests, fault coverage, read-only ports, write-only ports

1. Introduction

Multi-port (MP) memories are memories which consist of P ports, with $P \geq 2$. The ports are used to access the memory cells simultaneously and independent of each other. MP memories are widely used and for different purposes; e.g., for providing multiple and concurrent access to a set of locations, and for synchronization of asynchronous processes. Because of this functionality, MP memories are widely used in high-speed processors. Although MP memories have been in use for a relatively long time, little has been published about testing them. An *ad-hoc test* technique with no specific fault model was described in [1]. Serial test algorithms for embedded MP memories were reported in [2]; however, the used fault models were restricted to shorts between word lines or bit lines. A *complex coupling fault* model and its test was developed in [3, 4]; this fault model is based on the traditional *idempotent coupling fault* (CF_{id}). The individual CF_{id}s, of which the complex coupling fault is composed, are too *weak* to sensitize a fault; however, their fault effects may be combined when the CF_{id}s are activated through different ports simultaneously. In [5, 6] a complete set of fault models (based on weak faults) for the memory cell array of *two-port* (*2P*) memories, together with their tests, has been established. In [7] the consequences of port restrictions on testing memory cell array faults in 2P memories have been investigated.

All fault models addressed in [1–4, 6] are for *memory cell array faults*. On *address decoders faults*, [5, 8] has reported new fault models (based on port interference) together with their tests. However, the tests are only valid for 2P memories consisting of ports having both the read as well as the write capability.

Many 2P memories have port restrictions; i.e., the ports allow only for read or write operations. These restrictions impact the possible tests for *single-port* (SP) as well as for 2P memories. The test strategy, which determines the set of SP and 2P tests to be performed for an optimal fault coverage, is also impacted by port restrictions.

In this paper, the effects of shorts and interferences between the two independent decoders on the fault modeling will be discussed; it is based on [5, 8, 9]. Since each decoder consists of a number of layers of gates, the interference can occur between an arbitrary layer of one port and any layer of another port. In addition, the impact of port restrictions on testing address decoder faults, together with the test strategy will be investigated. Section 2 discusses address decoder faults in 2P memories based on [5, 8]; it describes the influence of port interference on the fault modeling. Section 3 introduces the tests for the case that both ports have the read as well as the write capability. Section 4 describes the influence of port restrictions on testing address decoder faults, and gives a test for each type of two-port memory (e.g., a two-port memory having one read-only port and one write-only port). Section 5 derives the test strategy; while Section 6 ends with the conclusions.

2. Address Decoder Faults in 2P SRAM

Address decoder faults in 2P memories (2P-AFs) can be classified into two fault classes [5, 8]: the fault class involving a *single-port* (abbreviated as *AF1*) and the fault class involving *two ports* (abbreviated as *AF2*); see Fig. 1. The AF1 faults are address decoder faults (AFs) that occur in a single address decoder; i.e., they consist of the AFs which can occur in SP memories and are divided into no access faults and multiple access faults [10]. The AF2s consist of faults based on shorts between two different ports.

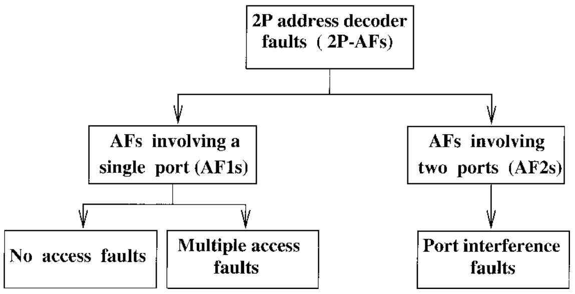
2P address decoder faults (2P-AFs)
AFs involving a single port (AF1s)
AFs involving two ports (AF2s)
No access faults
Multiple access faults
Port interference faults

Fig. 1. Classification of AFs in 2P memory.

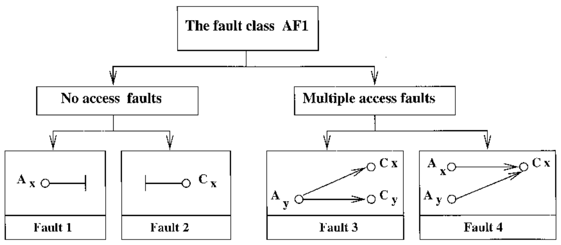

Fig. 2. AF1 fault class.

2.1. *The Fault Class AF1*

The AF1 fault class consists of faults in a single address decoder, caused by shorts and/or opens between the gates of the decoder. They can be divided into (see Fig. 2):

1. *No access faults*. They consist of:
   - Fault 1: with a certain address no cell is accessed.
   - Fault 2: there is no address with which a particular cell can be accessed.
2. *Multiple access faults*. They consist of:
   - Fault 3: with a certain address, multiple cells are accessed simultaneously.
   - Fault 4: a certain cell can be accessed with multiple addresses.

Because there are as many cells as addresses, none of the above faults can stand alone. They can only occur in one of the following combinations; see Fig. 3:

Fault A: Fault 1 + Fault 2.
Fault B: Fault 1 + Fault 3.
Fault C: Fault 2 + Fault 4.
Fault D: Fault 3 + Fault 4.

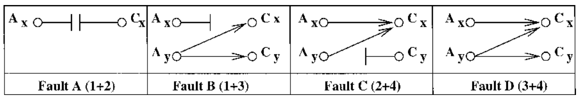

Fig. 3. Combinations of AF1 faults.

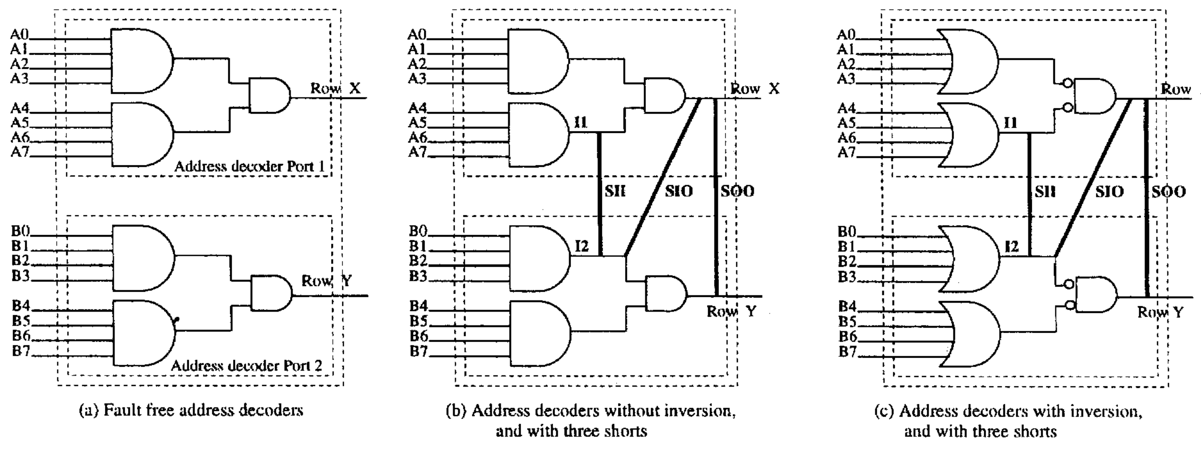
A0
A1
A2
A3
A4
A5
A6
A7
B0
B1
B2
B3
B4
B5
B6
B7
Row X
Row Y
Address decoder Port 1
Address decoder Port 2
I1
I2
SII
SIO
SOO
(a) Fault free address decoders
(b) Address decoders without inversion, and with three shorts
(c) Address decoders with inversion, and with three shorts

Fig. 4. Simple address decoders for 2P memories.

2.2. The Fault Class AF2

This fault class is based on the interference between the two different ports; that means shorts (i.e., *bridging faults*) between wires of the two different ports (Opens cannot cause port interference; they belong to the class AF1). In order to establish a set of fault models for the class of AF2s, shorts will be injected in the electrical circuits of the two address decoders; thereafter, functional faults will be derived, based on the fault behavior caused by shorts at the electrical level.

Fig. 4(a) shows two simple, fault free, address decoder circuits; one for the port 1 and one for the port 2. The decoders as well as their inputs are independent; i.e., they can access two different locations (or the same location) at any given time. Fig. 4(b) and (c) give equivalent circuits, each with three injected shorts: (1) a short between the two outputs lines of the two decoders (*SOO*), (2) a short between an internal line of one decoder and the output line of the second decoder (*SIO*), and (3) a short between two internal lines of the two decoders (*SII*).

In the rest of this paper the following notation will be used:

- A^1_X (A^2_X) denotes that row (column) line X is *selected* via port 1, P1, ($P2$) by an address. In the fault free case A^1_X has to *access* some cell C_x via P1. A^1_Φ denotes that *no* row (column) is selected via P1.
- "$A^1_X : A^2_Y$" denotes that the two rows (columns), X via P1 and Y via P2, are selected simultaneously.
- $A^1_{\bar{X}}$ denotes that row (column) X is *not* selected via P1; however P1 may well be used to select any row (column) $W \neq X$ or *no* row (column); i.e., $A^1_{\bar{X}}$ means: A^1_W, with $W \neq X$, or A^1_Φ.
- C^1_x denotes that the cell C_x in row (column) X is *accessed* via P1. The cell C_x can be in any column (row). C^1_ϕ denotes that *no* cell is accessed via P1.
- "$C^1_x : C^2_y$" denotes that the two cells, C_x via P1 and C_y via P2, are accessed simultaneously.
- $C^1_{\bar{x}}$ denotes that cell C_x is *not* accessed via P1. However, P1 may well be used to access any cell different than C_x; i.e., $C^1_{\bar{x}}$ means: C^1_w, with $w \neq x$, or C^1_ϕ.

In the remainder of this section we will focus on the row address decoders; the same applies to column address decoders.

2.3. Electrical Behavior of the Defective Circuit

In the following the electrical behavior of the address decoder circuits will be discussed in the presence of each of the shorts shown in Fig. 4; an overview is given in Table 1.

SOO: Short between the output lines. The presence of the short SOO in the address decoders (see Fig. 4 (b) and (c)) may cause a fault. Depending on the logic value of the output lines, two cases can be distinguished: *fault absent* and *fault present*.

A. Fault absent. This is the case when the two output lines have the same logic value (high or low);

Table 1. The faults in the presence of the three shorts.

Short		Fault	Number	Detection property
		If $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_x : C^2_y$	Fault SOO.1	*dAF*2
SOO		*If* $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_\phi : C^2_{\bar{y}}$	Fault SOO.2	*sAF*2
		If $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_\phi : C^2_y$	Fault SOO.3	*sAF*2
	Without inversion	*If* $A^1_X : A^2_Z$, $Z \neq Y$, *then* $C^1_x : C^2_z : C^2_y$	Fault SIO.A.1	*dAF*2
SIO		*If* $A^1_{\bar{X}} : A^2_Y$ *then* $C^1_{\bar{x}} : C^2_\phi$	Fault SIO.A.2	*sAF*2
	With inversion	*If* $A^1_X : A^2_Y$ *then* $C^1_x : C^2_\phi$	Fault SIO.B.1	*dAF*2
		If $A^1_{\bar{X}} : A^2_Z$, $Z \neq Y$, *then* $C^1_{\bar{x}} : C^2_z : C^2_y$	Fault SIO.B.2	*sAF*2
SII		*If* $A^1_X : A^2_Z$, $Z \neq Y$, *then* $C^1_x : C^2_z : C^2_y$	Fault SII.1	Equivalent with SIO.A.1
		If $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_\phi : C^2_{\bar{y}}$	Fault SII.2	Equivalent with SOO.2

i.e., (1) the two rows, X via port (P1) and Y via P2, are selected simultaneously (i.e., $A^1_X : A^2_Y$), or (2) are both not selected (i.e., $A^1_W : A^2_Z$; $W \neq X$ and $Z \neq Y$; A^1_W denotes that P1 intends to select row W, rather than X, which means that the decoder for row X is not used). Note that row X and row Y can access the same cell. In this case, the decoder circuits then behave as if no short is present. In short, the two cases can be described as:

1. *If* $A^1_X : A^2_Y$ *then* $C^1_x : C^2_y$
2. *If* $A^1_W : A^2_Z$; $W \neq X$ *and* $Z \neq Y$ *then* $C^1_w : C^2_z$

B. Fault present. This is the case when the two output lines have opposite logic values; that means that only row X or row Y is selected (i.e., $A^1_X : A^2_{\bar{Y}}$ or $A^1_{\bar{X}} : A^2_Y$); see Fig. 4(b) and (c). Assume the case $A^1_X : A^2_{\bar{Y}}$, then depending on the value of the resistance of the bridge, four sub-cases can be distinguished [11]. Note that P2 can be used to select row $Z \neq Y$, or no row.

1. Both output lines have a high logic value. That means that the selection of row X (i.e., A^1_X) has as a consequence an erroneously selection of row Y (i.e., A^2_Y); whereby Y can access any cell, which may be cell C_x.

   If $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_x : C^2_y$
2. Both output lines have a low logic value. That means that A^1_X has as a consequence that no cell will be accessed; i.e.,

   If $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_\phi : C^2_{\bar{y}}$
3. The output line corresponding to row X has a high logic value and the output line corresponding to row Y has a low logic value. The circuits behave as if the fault is absent.
4. The output line corresponding to row X has a low logic value and the output line corresponding to row Y has a high logic value. In this sub-case A^1_X has as a consequence C^1_ϕ since row X becomes low. In addition, the cell C^2_y will be accessed since row Y becomes high.

   If $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_\phi : C^2_y$

SIO: Short between an internal and an output line. A similar way as used for the short SOO will be followed in order to analyze the electrical behavior of the address decoders in the presence of the short SIO. We will assume that the driver of the output line of the address decoder (i.e., row or column) is stronger than the driver of any internal line. Two cases are distinguished: fault absent and fault present.

A. Fault absent. The output line and the internal line have the same logic values. Depending of the type of address decoder circuits, without or with inversion (see Fig. 4(b) and (c)), this will require the simultaneous selection of row X and row Y, or the selection of only one of them; see below.

Without inversion: This requires that the two rows have to be selected simultaneously (see Fig. 4(b)): row X via P1 and some other row with $I2 = 1$ (i.e., B_0 through $B_3 = 1$) via P2; for the latter, row Y may be selected. Alternatively both rows are not selected. That is:

If $A^1_X : A^2_Y$ *then* $C^1_x : C^2_y$
If $A^1_W : A^2_Z$, $W \neq X$ *and* $Z \neq Y$, *then* $C^1_w : C^2_z$

With inversion: The output line (i.e., row X) and the internal line (i.e., I2; for which address Y will be chosen) have the same logic value, which means that at most one of the two rows, X or Y, is selected; see Fig. 4(c). That is:

If $A_X^1 : A_Z^2$, $Z \neq Y$, *then* $C_x^1 : C_z^2$; row X selected.
If $A_W^1 : A_Y^2$, $W \neq X$, *then* $C_w^1 : C_y^2$; row Y selected.

This can be rewritten in a more compact form as follows:

If $A_W^1 : A_Z^2$, $W \neq X$ *or* $Z \neq Y$, *then* $C_w^1 : C_z^2$

B. Fault present. This is the case when the output line and the internal line of the address decoders have opposite logic values. By using the assumption that the output line driver is stronger than the internal line driver, two sub-cases can be distinguished; the two sub-cases will be discussed for each decoder type (i.e., without and with inversion).

Without inversion; see Fig. 4(b). The two sub-cases are the following:

1. The output line has a *high* logic value; the fault present case now requires that the internal line has a low logic value. This sub-case occurs when row X is selected (i.e., A_X^1) *and* $I2 = 0$. The presence of the short SIO will drive the internal line I2 to a high logic value. If a row Z via port 2 is selected (i.e., A_Z^2); *and* A_Z^2 requires the inputs B_4 through B_7 to be high, then A_Y^2; i.e., row Y will be selected. Note that this has as a consequence that three cells may be accessed simultaneously. That is:

   If $A_X^1 : A_Z^2$, $Z \neq Y$, *then* $C_x^1 : C_z^2 : C_y^2$
2. The output line has a *low* logic value; the fault present case now requires that the internal line has a high logic value. This occurs when $A_{\bar{X}}^1$ *and* $I2 = 1$. In this sub-case row Y can not be selected since the output line forces the internal line to a low logic value. That is:

   If $A_{\bar{X}}^1 : A_Y^2$ *then* $C_{\bar{x}}^1 : C_\phi^2$

With inversion; see Fig. 4(c). The two sub-cases are described below.

1. The output line has a *high* logic value; the fault present case now requires that the internal line has a low logic value. This sub-case occurs when the rows X and Y are selected simultaneously. Since the short SIO will drive the internal line to a high value, row Y will not be selected. That is:

   If $A_X^1 : A_Y^2$ *then* $C_x^1 : C_\phi^2$
2. The output line has a *low* logic value; the fault present case now requires that the internal line has a high logic value. This occurs when both rows X and Y are not selected. If a row Z ($Z \neq Y$) is selected via P2 (A_Z^2) *and* A_Z^2 requires the inputs B_4 through B_7 to be low, then row Y will be selected erroneously. This is because the short SIO forces the internal line I2 to a low value. Note that P1 may be used to select any row different from X. That is:

   If $A_{\bar{X}}^1 : A_Z^2$, $Z \neq Y$, *then* $C_{\bar{x}}^1 : C_z^2 : C_y^2$

SII: Short between two internal lines. In a similar way as with the presence of the shorts SOO and SIO, the electrical behavior of the decoders in the presence of the short SII will be classified in the fault absent and the fault present cases.

A. Fault absent. This will be so when the internal lines have the same logic values. That means, for both types of the decoders, that the two rows X and Y are selected simultaneously, or both are not selected; see Fig. 4(b) and (c). That is:

If $A_X^1 : A_Y^2$ *then* $C_x^1 : C_y^2$
If $A_W^1 : A_Z^2$, $W \neq X$ *and* $Z \neq Y$, *then* $C_w^1 : C_z^2$

B. Fault present. This is the case when the two internal lines have opposite logic values. Depending on the value of the resistance of the bridge, four sub-cases can be distinguished for decoders with- as well as for decoders without inversion.

Without inversion: The two internal lines have different values only when one of the two input groups, A_4 through A_7 or B_0 through B_3, is high; see Fig. 4(b). Assume that $A_X^1 : A_{\bar{Y}}^2$, i.e., row X is selected and row Y is not selected (this requires that $I1 = 1$ and $I2 = 0$), then the following four cases can be distinguished:

1. $I1 = 1$ and $I2 = 1$. Both internal lines have a high logic value. That means that A_X^1 will force the the internal line I2 to be high. If $A_X^1 : A_Z^2$, whereby A_Z^2 requires the inputs B_4 through B_7 to be high, then row Y will be selected erroneously; i.e., A_Y^2, see Fig. 4(b). Note that the simultaneous selection of two rows (X and Z) has as a consequence that a third row will be selected. That is:

   If $A_X^1 : A_Z^2$, $Z \neq Y$, *then* $C_x^1 : C_z^2 : C_y^2$

2. $I1 = 0$ and $I2 = 0$. Row X will be not selected since the line I1 is forced to a low logic value; see Fig. 4(b).

   $If\ A_X^1 : A_{\bar{Y}}^2\ then\ C_\phi^1 : C_{\bar{y}}^2$

3. $I1 = 1$ and $I2 = 0$. This has no consequences since the circuits behave as if no short is present.
4. $I1 = 0$ and $I2 = 1$; (Note that the 'good case' assumed $I1 = 1$ and $I2 = 0$). Row X will be not selected, since the internal line I1 is forced to a low logic value.

   $If\ A_X^1 : A_{\bar{Y}}^2\ then\ C_\phi^1 : C_{\bar{y}}^2$

With inversion: The two internal lines have different values when only one of the two input groups, A_4 through A_7 or B_0 through B_3, is low (i.e., $I1 = 1$ and $I2 = 0$, or $I1 = 0$ and $I2 = 1$; see Fig. 4 (c)). Assume $A_X^1 : A_{\bar{Y}}^2$; i.e., row X is selected and row Y is not selected (This requires that $I1 = 0$ and $I2 = 1$), then the following four cases can take place.

1. $I1 = 0$ and $I2 = 0$. That means that the internal line I1 will force the the internal line I2 to be low. If $A_X^1 : A_Z^2$, whereby A_Z^2 requires the inputs B_4 through B_7 to be low, then row Y will be selected erroneously; i.e., A_Y^2, see Fig. 4(c). Note that the simultaneous selection of two rows (X and Z) has as a consequence that a third row will be selected. That is:

   $If\ A_X^1 : A_Z^2,\ Z \neq Y,\ then\ C_x^1 : C_z^2 : C_y^2$

2. $I1 = 1$ and $I2 = 1$. Row X will be not selected since the line I1 is forced to high logic value; see Fig. 4(c).

   $If\ A_X^1 : A_{\bar{Y}}^2\ then\ C_\phi^1 : C_{\bar{y}}^2$

3. $I1 = 0$ and $I2 = 1$. This has no consequences since the circuits behave as if no short is present.
4. $I1 = 1$ and $I2 = 0$. Row X will not be selected, since the internal line I1 is forced to a high logic value.

   $If A_X^1 : A_{\bar{Y}}^2\ then\ C_\phi^1 : C_{\bar{y}}^2$

It will be clear from the above that both types of decoders have the same behavior in the presence of the short SII.

Table 1 summarizes all faults discussed above; it lists the faults caused by each short for both types of decoders. Note that Short SOO as well as Short SII cause the same faults in both types of the decoders; and that Fault SOO.2 and Fault SII.2 are the same, as well as Fault SIO.A.1 and Fault SII.1. That means that the faults caused by Short SII are covered by faults caused by shorts SOO and SIO. Therefore, and from now on, we will focus only on the faults caused by shorts SOO and SIO. In addition, it should be noted that each of the faults of Table 1 occurs in two forms; e.g., for Fault SOO.1, the fault "$If\ A_X^1 : A_{\bar{Y}}^2\ then\ C_x^1 : C_y^2$," or the fault "$If\ A_{\bar{X}}^1 : A_Y^2\ then\ C_x^1 : C_y^2$ " can occur.

2.4. Classification of AF2s

By inspecting AF2 faults shown in Table 1, it will be clear that they can be divided into two fault subclasses: (1) the fault subclass that involves only the use of a *single* port in order to be detected (abbreviated as *sAF2s*), and (2) the fault subclass that involves the use of the two (*dual*) ports in order to be detected (abbreviated as *dAF2s*).

The *AF2s* involving a single-port in order to be detected, *sAF2s*, consist of the faults SOO.2, SOO.3, SIO.A.2 and SIO.B.2; see Table 1. They have the properties that: (1) the cell that the selected row intends to access (e.g., C_x) will not be accessed successfully (Fault SOO.2, Fault SOO.3 and Fault SIO.A.2), or (2) multiple cells will be accessed via the same port (Fault SIO.B.2). Therefore, the faults SOO.2, SOO.3 and SIO.A.2 are equivalent with Fault A, and Fault SIO.B.2 is equivalent with Fault D (see Fig. 3). Hence, *sAF2s are a subset of AF1s*, and any test which detects Fault A and Fault D, also detects faults SOO.2, SOO.3, SIO.A.2, and SIO.B.2. Note that the test has to be performed in such way that when applied via one port, all rows via the other port have to be *not selected* since the sensitization of *sAF2s* require that only one port has to be active at time. The *sAF2s*, which will be detected by tests for AF1s, will be not considered from here on.

The *AF2s* involving the use of both ports in order to be detected, *dAF2s*, consist of the faults SOO.1, SIO.A.1 and SIO.B.1; see Table 1. The fault SOO.1 requires the use of the two ports since the selection of a row via the first port (e.g., A_X^1) has as consequence that row Y via the other port (e.g., A_Y^2) will be selected; while the faults SIO.A.1 and SIO.B.1 require the simultaneous use of the two ports for their sensitization. To detect these faults new tests are needed. In the rest of this paper we will focus only on testing $AF2s$ involving the use of two ports simultaneously; see Table 2. Following the notation used in [10], the faults SOO.1,

Table 2. The reduced set of AF2s.

Fault	Name
If $A_X^1 : A_{\bar{Y}}^2$ *then* $C_x^1 : C_y^2$	Fault E
If $A_X^1 : A_Z^2$, $Z \neq Y$, *then* $C_x^1 : C_z^2 : C_y^2$	Fault F
If $A_X^1 : A_Y^2$ *then* $C_x^2 : C_\phi^2$	Fault G

SIO.A.1 and SIO.B.1 will be referred to as Fault E, Fault F and Fault G, respectively.

3. Tests for AFs in 2P SRAMs

$2P$ memories come in different forms depending on the type of ports they consist of. Each of the two ports of a 2P memory may have the capability to be a *read-only port* (*Pro*), a *write-only port* (*Pwo*), or a *read-write port* (*Prw*). The total number of ports $P = Prw + Pwo + Pro$. Therefore, four types of $2P$ memories can be distinguished based on the *port mix*:

- $(rw - rw)2P$ memories; $Prw = 2$.
- $(rw - wo)2P$ memories; $Prw = 1$ and $Pwo = 1$.
- $(rw - ro)2P$ memories; $Prw = 1$ and $Pro = 1$.
- $(wo - ro)2P$ memories; $Pwo = 1$ and $Pro = 1$.

The faults of the classes AF1 and AF2 occur in all types of 2P memories because each 2P memory contains two duplicated sets of address decoders, irrespective of the port type. However, the port restriction impacts the possible tests; e.g., two simultaneous write operations can not be applied to $(rw - ro)2P$ memories, nor to $(wo - ro)2P$ memories. In the rest of this section, first the notation of march tests extended for 2P memories will be given; thereafter, tests for AFs in $(rw - rw)2P$ memories will be discussed based on [5, 8].

3.1. Notation for March Tests for 2P Memories

The march notation of [10] is extended for 2P memories as follows [5, 8]:

- The operations applied in parallel to the ports are separated using colons, and the port number to which each of the set of the parallel operations is applied is determined implicitly; for example, the march element $(r0 : w1)$ denotes that a $r0$ operation is applied to P1, and $w1$ operation is applied to P2. Ports can also be specified explicitly, by superscripting the operation with the corresponding port number; e.g., $r0^1$ denotes a read operation via P1.
- The character 'n' denotes *no* operation, while the character '$-$' denotes *any* allowed operation. For example, $(r0 : n)$ denotes a $r0$ operation via $P1$, and no operation on $P2$.
- The cell to which the operation is applied can be specified explicitly by subscripting the corresponding operation. E.g., $(r0_{r,c})$ denotes a $r0$ operation applied to a cell with row r and column c.
- $\Updownarrow_{r=0}^{R-1}$: denotes $\Uparrow_{r=0}^{R-1}$ or $\Downarrow_0^{r=R-1}$, and $\Uparrow_{r_1=0}^{R-1} \Uparrow_{r_2=r_1+1}^{R-1}$: denotes a nested addressing sequence, whereby row r_1 goes from 0 to $R - 1$; and for each value of r_1, row r_2 goes from $r_1 + 1$ to $R - 1$.
- $C_{r,c}$ denotes the cell with row r and column c.

3.2. Tests for AFs

As mentioned in Section 2.3, the *AFs* in 2P memories are divided into *AF*1*s* and *AF*2*s*. Moreover, AF2s consist of faults that are a subset of AF1s and faults which require new tests. Therefore the test procedure can be divided into two parts:

1. Test(s) to detect *AF*1*s*
2. Test(s) to detect *AF*2*s*.

For the detection of *AF*1*s* (and AF2s which are a subset of AF1s) a march test like MATS+, March X, etc. (see [10]) can be used. The test has to be applied via each port of the 2P memory separately. It should be noted that in order to detect *AF*2*s* which are a subset of AF1s, when the test is applied via the one port, the other port has to be not active (i.e., no row is selected via this port); see Table 1.

For the detection of *AF*2*s* (i.e., faults E, F, and G), a special two-port tests are required. In order to detect Fault F and Fault G (see Table 2), we have to generate all possible pairs of row addresses. This requires the generation of $C_2^R = \frac{R(R-1)}{2}$ addresses, whereby R represents the number of rows in the memory. That means that a test length of $O(R^2)$ for any functional test is required. Note that a functional test for Fault E requires only a test length of $O(R)$ since only one of the ports has to step through all addresses, while on the other port a single address (which does not change during the test), or no address, has to be specified.

The test detecting all *AF*2 faults of Table 2 (abbreviated as *Test AF*2*s*), for shorts in row decoders, is shown in Fig. 5. It should be noted that for detecting the same

```
Select two columns $c_1$ and $c_2$; $c_1 \neq c_2$;
for $all\ r$                        //$r \in \{0, 1, ..., R-1\}$
{ $w0_{r,c_1} : n$;                 // Initialize $c_1$ to 0
  $n : w0_{r,c_2}$;                 // Initialize $c_2$ to 0
}
for($r_1 = 0; r_1 < R; r_1 + +$)
{  for($r_2 = 0; r_2 < R; r_2 + +$)
   { $r0_{r_1,c_1} : r0_{r_2,c_2}$;          // Detection
     $w1_{r_1,c_1} : w1_{r_2,c_2}$;          // Sensitization
     $r1_{r_1,c_1} : n$;                     // Detection
     $n : r1_{r_2,c_2}$;                     // Detection
     $w0_{r_1,c_1} : n$;
     $n : w0_{r_2,c_2}$;
   }
}
for $all\ r$
{ $r0_{r,c_1} : -$;                 // Detection
}
```

Fig. 5. Test for AF2 in 2P memories.

faults in the column decoders, a similar test has to be applied to the column decoders (Note: shorts between row- and column decoders are not considered realistic, because of the topology of these decoders). In the figure, the first loop initializes the memory cells of two distinct columns c_1 and c_2 to 0. The second double loop generates all address pairs required in order to detect the faults; the operations "$w1_{r_1,c_1} : w1_{r_2,c_2}$" sensitize the faults, while the operations "$r0_{r_1,c_1} : r0_{r_2,c_2}$", "$r1_{r_1,c_1} : n$", "$n : r1_{r_2,c_2}$" and the operation "$r1_{r,c_1} : -$" (of the third loop) detect them. The two single write operations are added to the second loop to make the content of the cells equal to 0 after each loop; this is the expected value of the simultaneous read operations (i.e.,"$r0_{r_1,c_1} : r0_{r_2,c_2}$").

It should be noted that it is necessary to select *two* distinct columns; otherwise, e.g., Fault E : *"If A^1_X : $A^2_{\bar{Y}}$ then C^1_x : C^2_y"*, can *not* be sensitized when row $\bar{Y}$ accesses the same cell as row X. If in Fig. 4(b) and Fig. 4(c) A^1_X and $A^2_{\bar{Y}}$ specify rows in the same column, then Fault E will be *"If $A^1_X : A^2_{\bar{Y}}$ then $C^1_{r,c_1} : C^2_{r,c_1}$"* (Note: due to Fault E, cell C_{r,c_1} will also be accessed via P2). Therefore, when a single column would be used, Fault E would be masked. The fault would be sensitized via an operation applied to C^1_{r,c_1} and detected via a read operation applied to C^2_{r,c_1}; since both ports access the same cell C_{r,c_1}, masking will take place. The use of two columns will make Fault E behave as: *"If $A^1_X : A^2_{\bar{Y}}$ then $C^1_{r_1,c_1} : C^2_{r_2,c_2}$"*. This means that the fault will be not masked, but sensitized in cell C_{r_2,c_2}. A similar explanation can be given for Fault F and Fault G.

The fault coverage of the test of Fig. 5 is analyzed below for each of the faults (E, F and G).

Fault E; i.e., *"If A^1_X : $A^2_{\bar{Y}}$ then C^1_x : C^2_y"*. Depending on the value of Y ($Y \in \{0, 1, \ldots, R-1\}$) two cases can be distinguished:

1. $\bar{Y} = 0$ (i.e., $Y \neq 0$). In this case the fault will be sensitized by the operation "$w1_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = X$ and $r_2 = 0 (= \bar{Y})$. In the presence of the fault, 1 will not only be written into the cells C_{x,c_1} and $C_{\bar{y},c_2}$, but also into the cell C_{y,c_2} (due to Fault E) which content was 0. The fault will be detected by the operation "$r0_{x,c_1} : r0_{y,c_2}$".
2. $\bar{Y} \neq 0$ (i.e., $Y = 0$). In this case the fault will be sensitized by the operation "$w1_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = X$ and $r_2 \neq 0$ (e.g., $r_2 = 1$). In the presence of the fault, 1 will not only be written into the cells C_{x,c_1} and $C_{\bar{y},c_2}$, but also into the cell C_{y,c_2} which content was 0. The fault will be detected by the operation "$r0_{x+1,c_1} : r0_{y,c_2}$".

The second form of Fault E; i.e., *"If $A^1_{\bar{X}} : A^2_Y$ then $C^1_x : C^2_y$"* will be sensitized by the operation "$w1_{r_1,c_1} : w1_{r_2,c_2}$": (1) when $r_2 = Y$ and $r_1 = 0 (= \bar{X})$ if $X \neq 0$, and (2) when $r_2 = Y$ and $r_1 \neq 0$ (e.g., $r_1 = 1$) if $X = 0$. In both cases, 1 will not only be written into the cells C_{r_1,c_1} and C_{y,c_2}, but also into the cell C_{x,c_1} which content was 0. The fault will be detected by the operation "$r0_{x,c_1} : r0_{y,c_2}$" if $X \neq 0$ and by the march element M_2 if $X = 0$.

Fault F; i.e., *"If $A^1_X : A^2_Z$, $Z \neq Y$, then $C^1_x : C^2_z : C^2_y$"*. Since the second loop of the test generates all pairs of rows, the fault will be sensitized by the operation "$w1_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = X$ and $r_2 = Z$. In the presence of the fault, 1 will not only be written into the cells C_{x,c_1} and C_{z,c_2}, but also into the cell C_{y,c_2} which content was 0; whereby $Y \in \{0, 1, \ldots, R-1\}$, and $Y \neq Z$. The fault will be detected by the operation "$r0_{x,c_1} : r0_{y,c_2}$" if $Y > Z$ (i.e., in the same inner loop in which the fault is sensitized), and by the operation "$r0_{x+1,c_1} : r0_{y,c_2}$" if $Y < Z$ (i.e., in the next iteration of the inner loop; this will be for $r_1 = x + 1$). The second form of Fault F; i.e., *"If $A^1_W : A^2_Y$, $W \neq X$, then C^1_x : $C^1_w : C^2_y$"* will be sensitized by the operation "$w1_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = W$ and $r_2 = Y$; and detected by the operation "$r0_{x,c_1} : r0_{0,c_2}$" if $X > W$, and by the march element of the third loop if $X < W$.

Fault G; i.e., "*If* $A^1_X : A^2_Y$ *then* $C^1_x : C^2_\phi$" (or the second form of Fault G: "*If* $A^1_X : A^2_Y$ *then* $C^1_\phi : C^2_y$"). This fault will be sensitized by the operation "$w1_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = X$ and $r_2 = Y$, and detected by the operation "$r1_{r_1,c_1} : n$" or "$n : r1_{r_2,c_2}$".

As can be seen from the Fig. 5, the number of operations required to perform the test is $3R + 6R^2$; and therefore the time complexity of the test is $O(R^2)$ whereby R is the number of rows in the memory cell array. If we assume a two dimensional memory cell array with size n, then the time complexity of the test will be $O(n)$; i.e., linear with the size of the memory.

The test shown in C++ pseudo code in Fig. 5 can be rewritten in the compact march notation as follows: $\{\Updownarrow^{R-1}_{r=0}\ (w0_{r,c_1} : n, n : w0_{r,c_2})$; $\Uparrow^{R-1}_{r_1=0}\ (\Uparrow^{R-1}_{r_2=0}\ (r0_{r_1,c_1} : r0_{r_2,c_2}\ ,\ w1_{r_1,c_1} : w1_{r_2,c_2}\ ,\ r1_{r_1,c_1} : n\ ,\ n : r1_{r_2,c_2}\ ,\ w0_{r_1,c_1} : n\ ,\ n : w0_{r_2,c_2}))$; $\Updownarrow^{R-1}_{r=0}\ (r0_{r,c_1}\ : -)\}$. This test will be referred from now on as *March* $(rw - rw)AF2$.

To detect *AF*2*s* in column decoders, a similar test can be applied. In the test procedure, first two distinct rows have to be selected and their memory cells have to be initialized to 0, and then the loop has to be applied for all columns; i.e., from $c = 0$ to $c = C-1$.

4. Tests for Restricted 2P Memories

The test, March $(rw - rw)AF2$, given in Section 3.2 applies to $(rw - rw)2P$ memories only. In this section tests for detecting *AF*2*s* in each of the restricted 2P memory types will be derived.

4.1. Test for AF2s in (rw – wo)2P Memories

The $(rw - wo)2P$ memories allow for two simultaneous write operations, and only a single read operation. The test detecting *all AF*2*s* in such memories is given in Fig. 6, and is referred as March $(rw - wo)AF2$. It is the same as March $(rw - rw)$AF2 discussed in Section 3.2, and therefore it has the same fault coverage; however all read operations have to be applied via $Prw = \text{P2}$, and the simultaneous read operations have to be applied sequentially (i.e., "$r0^2_{r_1,c_1} : n$; $n : r0_{r_2,c_2}$") via the same port.

4.2. Test for AF2s in (rw – ro)2P Memories

The $(rw - ro)2P$ memories allow for simultaneous read operations and only for a single write operation. Therefore the sensitizing operations used in March $(rw - rw)AF2$ and March $(rw - wo)AF2$ (i.e., "$w1_{r_1,c_1} : w1_{r_2,c_2}$") can not be used for such memories. The march test for *AF*2*s* in $(rw - ro)2P$ memories is given in Fig. 7. By assuming that $P1 = Pro$ and $P2 = Prw$, the test of Fig. 7 *guarantees* the detection of one form of Fault E (i.e., "*If* $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_x : C^2_y$"), one form of Fault F (i.e., "*If* $A^1_X : A^2_Z$, $Z \neq Y$, *then* $C^1_x : C^2_z : C^2_y$") and one form of Fault G (i.e., "*If* $A^1_X : A^2_Y$ *then* $C^1_x : C^2_\phi$"). In addition, the test can detect the second form of Fault E (i.e., "*If* $A^1_{\bar{X}} : A^2_Y$ *then* $C^1_x : C^2_y$"), the second form of Fault F (i.e., "*If* $A^1_W : A^2_Y$, $W \neq X$, *then* $C^1_x : C^1_w : C^2_y$") and the second form of Fault G (i.e.,"*If* $A^1_X : A^2_Y$ *then* $C^1_\phi : C^2_y$") as shown below.

First, consider the first form of Fault E; i.e., "*If* $A^1_X : A^2_{\bar{Y}}$ *then* $C^1_x : C^2_y$". Depending on the value of Y, we can distinguish two cases:

1. $Y \neq 0$. In this case the fault will be sensitized by the operation "$r0_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = X$ and $r_1 = 0$. In the presence of the fault, the cell C_{y,c_2} (which content was 0) will be written with 1 via *Prw*. The fault will be detected by the operation "$n : r0_{y,c_2}$".
2. $Y = 0$. In this case the fault will be sensitized by the operation "$r0_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = X$ and $r_1 \neq 0$ (e.g., $= 1$). As a consequence of the fault, 1

$\{\Updownarrow^{R-1}_{r=0}\ (w0_{r,c_1} : n\ ,\ n : w0_{r,c_2})$;
$\Uparrow^{R-1}_{r_1=0}\ (\Uparrow^{R-1}_{r_2=0}\ (r0^2_{r_1,c_1} : n\ ,\ n : r0_{r_2,c_2}\ ,\ w1_{r_1,c_1} : w1_{r_2,c_2},\ r1^2_{r_1,c_1} : n\ ,\ n : r1_{r_2,c_2}\ ,\ w0_{r_1,c_1} : n\ ,\ n : w0_{r_2,c_2}))$;
$\Updownarrow^{R-1}_{r=0}\ (r0^2_{r,c_1} : -)\}$

Fig. 6. March $(rw - wo)AF2$ for *AF*2*s* in $(rw - wo)2P$ memories.

$\{\Updownarrow^{R-1}_{r=0}\ (w1^2_{r,c_1} : n\ ,\ n : w0_{r,c_2})$;

$\Uparrow^{R-1}_{r=0}\ (\Uparrow^{R-1}_{r_2=0}\ (r1_{r_1,c_1} : n\ ,\ n : r0_{r_2,c_2}\ ,\ w0^2_{r_1,c_1} : n\ ,\ r0_{r_1,c_1} : w1_{r_2,c_2}\ ,\ n : r1_{r_2,c_2}\ ,\ w1^2_{r_1,c_1} : r1^1_{r_2,c_2}\ ,\ n : w0_{r_2,c_2}))\}$

Fig. 7. March $(rw - ro)AF2$ for *AF*2*s* in $(rw - wo)2P$ memories.

will also written in cell C_{y,c_2} which content was 0. The fault will be detected by the operation "$n : r0_{y,c_2}$" of the next iteration of the inner loop.

Second, consider the first form of Fault F; i.e., "*If* $A_X^1 : A_Z^2$, $Z \neq Y$, *then* $C_x^1 : C_z^2 : C_y^2$". The fault will be sensitized by the operation "$r0_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = X$ and $r_2 = Z$. In the presence of the fault, 1 will not only be written into cell C_{r_2,c_2} but also in the cell C_{y,c_2} (via port *Prw*) which content was 0. The fault will be detected by the operation "$n : r0_{y,c_2}$" of the same iteration in which the fault is sensitized if $Y > Z$, or of the next iteration of the inner loop if $Y < Z$.

Third, consider the first form of fault G; i.e., "*If* $A_X^1 : A_Y^2$ *then* $C_x^1 : C_\phi^2$". This fault will be sensitized by the operations "$r0_{r_1,c_1} : w1_{r_2,c_2}$" when $r_1 = X$ and $r_2 = Y$, and detected by the operations "$n : r1_{y,c_2}$".

Consider now the second form of Fault E (i.e., "*If* $A_{\bar{X}}^1 : A_Y^2$ *then* $C_x^1 : C_y^2$"), the second form of Fault F (i.e., "*If* $A_W^1 : A_Y^2$, $W \neq X$, *then* $C_x^1 : C_w^1 : C_y^2$"), and the second form of Fault G (i.e., "*If* $A_X^1 : A_Y^2$ *then* $C_\phi^1 : C_y^2$"). These faults can not be sensitized using write operations via P1 since P1 = *Pro*. That means that the read operation is the only possibility to use. Assume the presence of the second form of Fault E, then applying the operation "$r0_{r_1,c_1} : w1_{r_2,c_2}$" (when $r_1 \neq X$ and $r_2 = Y$) will cause $P1$ to read two cells (i.e., C_{r_1,c_1} and C_{x,c_1}), which have different data values, simultaneously; while P2 will correctly write into cell C_{y,c_2} (see Fig. 7). A similar explanation can given by applying the operation "$w1^2_{r_1,c_1} : r1^1_{r_2,c_2}$". Depending on the technology of the sense amplifier, the value can be:

- *A* 0 *or a* 1 (*Stuck at fault behavior*): in this case the detection of the fault will be *guaranteed* by the operation "$r0_{r_1,c_1} : w1_{r_2,c_2}$" if it appears as SA1 and by the operation "$w1^2_{r_1,c_1} : r1^1_{r_2,c_2}$" if it appears as SA0.
- *The last read value*: in this case the detection of the fault will be *guaranteed* by the operation "$r0_{r_1,c_1} : w1_{r_2,c_2}$" since the last read value was 1 (by the operation "$r1_{r1,c_1} : n$"); see Fig. 7.
- *The OR logic function* of the read values. The detection of the fault is *guaranteed* by the operation "$r0_{r_1,c_1} : w1_{r_2,c_2}$" since all cells of column c_1 contain 1, and only the cell C_{r_1,c_1} contains 0; that means that the content of cell C_{r_1,c_1} is 0 and the content of C_{x,c_1} is 1.
- *The AND logic function* of the read values. The detection of the fault is *guaranteed* by the operation "$w1^2_{r_1,c_1} : r1^1_{r_2,c_2}$" since all cells of column c_2 contain 0, and only the cell C_{r_2,c_2} contains 1; that means that the content of cell C_{r_2,c_2} is 1 and the content of C_{x,c_2} is 0.
- *Random*: in this case the detection of the fault can *not be guaranteed*. However, applying the test multiple times can detect the fault probabilistically.

A similar explanation can be given for the detection of the second form of the Fault F and Fault G. However, the presence of second form of Fault G will has as a consequence that a cell will not be accessed; and therefore the read value depends on the type of sense amplifier (see above).

4.3. *Test for AF2s in (wo-ro)2P Memories*

The $(ro - wo)2P$ memories allow for only a single write operation, and/or a single read operation. The test detecting $AF2s$ in such memories is given in Fig. 8; it is referred as *March* $(ro - wo)AF2$. Note that this test is the same as that of Fig. 7, and therefore it has the same fault coverage as March $(rw - wo)AF2$; see Section 4.2. However, all read operations have to be done via $P1 = Pro$, and all write operations via $P2 = Pwo$.

5. The Test Strategy

As mentioned is Section 3.2, the detection of AF1s requires the application of the proper test via each port separately; while the detection of AF2s requires the application of the proper test via the two-port simultaneously. However, this is not always possible; e.g., a test consisting of write as well as read operations can not be applied via a *Pro* neither via a *Pwo*. In the rest of this section, the test strategy to detect AF1s, and AF2s for each type of 2P memory will be discussed.

$$\{\Updownarrow_{r=0}^{R-1} (w1^2_{r,c_1} : n \, , n : w0_{r,c_2});$$

$$\Uparrow_{r=0}^{R-1} (\Uparrow_{r_2=0}^{R-1} (r1_{r_1,c_1} : n \, , n : r0^1_{r_2,c_2} \, , w0^2_{r_1,c_1} : n \, , r0_{r_1,c_1} : w1_{r_2,c_2}, n : r1^1_{r_2,c_2} \, , w1^2_{r_1,c_1} : r1^1_{r_2,c_2} \, , n : w0_{r_2,c_2}))\}$$

Fig. 8. March $(ro - wo)AF2$ for $AF2s$ in $(ro - wo)2P$ memories.

```
A. To detect $AF1s$ :
   Apply a 1P-Test via P1, while Port P2 is
     not active;
   Apply a 1P-Test via P2, while Port P1 is
     not active;
B. To detect $AF2s$:
   Apply March (rw-rw)AF2 for row decoders;
   Apply March (rw-rw)AF2 for column decoders
```

Fig. 9. Test strategy for ($rw - rw$) memories.

5.1. *Test Strategy for (rw – rw)2P Memories*

In such memories, each port has the read-write capability; therefore march tests to detect AF1s can be applied via each port separately. Fig. 9 shows the test strategy that guarantees the detection of all AFs in $(rw - rw)2P$ memories. Step A guarantees the detection of all AF1s of each port. The test '1P-test' can be any appropriate test. Step B guarantees the detection of $AF2s$ in row decoders and column decoders; it uses the test described in Section 3.2.

5.2. *Test Strategy for (rw – wo)2P Memories*

To detect the AF1s, the test has to be applied via each port. However, this is not possible via *Pwo*, since it has a write-only capability while tests require write as well as read operations. To detect AF1s for *Pwo*, the test has to be applied in such way that the write operations will be done via *Pwo* and the read operations via *Prw*. Fig. 10 shows the test strategy for $(rw - wo)2P$ memories which guarantees the detection of all AFs. The Step A.1 guarantees the detection of all AF1s for *Prw*. Step A.2 guarantees the detection of all AF1s for *Pwo*. The Step B guarantees the detection of $AF2s$ in $(rw - wo)2P$ memories and uses March $(rw - wo)AF2s$ of Section 4.1.

```
A. To detect $AF1s$ :
   1. Apply a 1P-Test via Prw, while Pwo is
        not active;
   2. Apply a 1P-Test in such way that :
        write operations will be done via Pwo;
        and read operations via Prw;
B. To detect $AF2s$:
   Apply March (rw-wo)AF2 for row decoders;
   Apply March (rw-wo)AF2 for column decoders
```

Fig. 10. Test strategy for ($rw - wo$) memories.

5.3. *Test Strategy for (rw – ro)2P Memories*

The $(rw - ro)2P$ memories have one *Prw* and one *Pro*. AF1s of *Prw* can be detected by applying a test via *Prw*, while those of *Pro* can be detected by applying a test in such way that the write operations will be done via *Prw* and the read operations via *Pro*. Fig. 11 shows the test strategy that detects AFs in $(rw - ro)2P$ memories. A similar explanation can be given as for the strategy of $(rw - wo)2P$ memories.

5.4. *Test Strategy for (wo – ro)2P Memories*

The $(wo - ro)2P$ memories have one *Pwo* and one *Pro*. Note that no test can be applied via *Pwo* nor via *Pro*. The only possibility is to apply a test in such way that the write operations will be done via *Pwo* and the read operations via *Pro*. Fig. 12 shows the test strategy for $(wo - ro)2P$ memories. Step A guarantees the detection of $AF1s$ since they can be sensitized via *Pwo* and detected via *Pro*; however, it can not specify whether the detected AF1 belongs to *Pwo* or to *Pro*. Step B guarantees the detection of $AF2s$ in $(wo - ro)2P$ memories, using March $(wo - ro)AF2s$ of Section 4.3.

```
A. To detect $AF1s$:
   1. Apply a 1P-Test via Prw, while Pro is
        not active;
   2. Apply a 1P-Test in such way that :
        read operations will be done via Pro;
        and write operations via Prw;
B. To detect $AF2s$:
   Apply March (rw-wo)AF2 for row decoders;
   Apply March (rw-wo)AF2 for column decoders;
```

Fig. 11. Test strategy for ($rw - ro$) memories.

```
A. To detect $AF1s$:
   Apply a 1P-Test in such way that :
        read operations will be done via Pro;
        and write operations via Pwo;
B. To detect $AF2s$:
   Apply March (wo-ro)AF2 for row decoders;
   Apply March (wo-ro)AF2 for column decoders
```

Fig. 12. Test strategy for ($wo - ro$) memories.

6. Conclusion

In this paper address decoder faults in two-port memories have been analyzed; the effects of interference and shorts between ports have been investigated and new fault models have been introduced. These fault models are divided into faults requiring a single-port (AF1s) and faults requiring two-ports (AF2s) in order to be detected. AF1s can be covered with tests for address decoder faults in single-port memories, while AF2s require a special test. In addition, the impact of the port restrictions (i.e., the ports which allow only for read or write operations) on testing AF2s has been investigated. Four types of two-port memories have been identified. The test for each type of 2P memory, together with the test strategy, has been presented. The time complexity of the functional tests is $O(R^2)$, whereby R is the number of rows (or columns) in the memory.

References

1. M.J. Raposa, "Dual Port Static RAM Testing," *Proc. IEEE International Test Conference*, 1988, pp. 362–368.
2. B. Nadeau-Dostie, A. Sulburt, and V.K. Agrawal, "Serial Interfacing for Embedded-memory Testing," *IEEE Design and Test of Computers*, Vol. 7, No. 2, pp. 52–63, 1990.
3. V.C. Alves and M. Nicolaidis, "Detecting Complex Coupling Fault in Multi-Port RAMs," *IMAG Research Report No. RR978*, Feb. 1991.
4. M. Nicolaidis, V.C. Alves, and H. Bederr, "Testing Complex Couplings in Multi-Port Memories," *IEEE Transactions on VLSI Systems*, Vol. 3, No. 1, pp. 59–71, March 1995.
5. S. Hamdioui, "Fault Models and Tests for Multi-Port Memories," Technical Report No. 168340-28(1997)-11, Delft University of Technology, Faculty of Information Technology and Systems, Delft, The Netherlands, Oct. 1997.
6. A.J. van de Goor and S. Hamdioui, "Fault Models and Tests for Two-Port Memories," *16th IEEE VLSI Test Symposium*, April 1998, pp. 401–410.
7. S. Hamdioui and A.J. van de Goor, "Consequence of Port Restrictions on Testing Two-Port Memories," *International Test Conference ITC'98*, Washington, USA, Oct. 1998, pp. 63–72.
8. S. Hamdioui and A.J. van de Goor, "Address Decoder Faults and their Tests in Two-Port Memories," *Memory Technology, Design and Testing*, San Jose, California, USA, Aug. 1998, pp. 97–103.
9. S. Hamdioui and A.J. van de Goor, "Consequence of Port Restrictions on Testing Address Decoder Faults in Two-Port Memories," *Seventh Asian Test Symposium*, Singapore, Dec. 1998, pp. 340–347.
10. A.J. van de Goor, *Testing Semiconductors Memories, Theory and Practice*, ComTex Publishing, Gouda, The Netherlands, 1998. E-mail: vdGoor@cardit.et.tudelft.nl
11. M. Renovell, P. Huc, and Y. Bertrand, "Bridging Faults Coverage Improvement by Power Supply Control," *14th IEEE VLSI Test Symposium*, 1996, pp. 338–343.

Said Hamdioui received the Master of Science degree from Delft University of Technology, The Netherlands in 1997. Since then, he is working towards the Ph.D. degree in Testing of Multi-Port Memories. Currently, he is an intern with intel Corporation, CA, which is a part of his Ph.D. program. His research interests include design of realistic fault models, design of test algorithms, as well as design for testability, and build in self testing for a large variety of multi-port memories.

Ad van de Goor obtained an MSEE degree from the Delft University of Technology, in Delft, The Netherlands. He additionally obtained an MSEE and a Ph.D. degree from Carnegie-Mellon University. He worked for Digital Equipment Corporation, in Maynard, Mass, as the chief architect of the PDP-11/45 computer, and for IBM in The Netherlands and in the USA, being responsible for the architecture of embedded systems. Currently he is professor of Computer Architecture at the Delft University of Technology. His main research interest is testing logic and memories. He has written two books and over 120 papers in the areas of computer architecture and testing. He is a member of the editorial board of the *Journal of Electronic Testing: Theory and applications*.