# A Concurrent Approach for Testing Address Decoder Faults in eFlash Memories

Olivier Ginez, Patrick Girard, Christian Landrault, Serge Pravossoudovitch,
Arnaud Virazel, Jean-Michel Daga

# A Concurrent Approach for Testing Address Decoder Faults in eFlash Memories

O. Ginez [1, 2]    P. Girard [1]    C. Landrault [1]    S. Pravossoudovitch [1]    A. Virazel [1]    J.-M. Daga [2]

[1] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier – LIRMM
Université de Montpellier II / CNRS    161, rue Ada – 34392 Montpellier Cedex 5, France
Email: <lastname>@lirmm.fr    URL: http://www.lirmm.fr/~w3mic

[2] ATMEL Rousset – Libraries and Design Tools Department
Embedded Non-Volatile Memory Group – 13106 Rousset Cedex, France
Email: olivier.ginez@rfo.atmel.com    jean-michel.daga@rfo.atmel.com    URL: http://www.atmel.com

## Abstract

*The evolution of System-on-Chip (SoC) designs involves the development of non-volatile memory technologies like Flash. As any kind of memories, embedded Flash (eFlash) can be subjected to complex functional faults that are related to their particular technological process and to their integration density. In this paper, we address a major issue during eFlash testing, namely the test of Address decoder Faults (AFs), which is generally very time consuming with ad-hoc solutions presently used in industry. In the first part of the paper, we show the impact of AFs on the functional behavior of an eFlash. Next, we use an analogy with RAM memory testing to classify AFs with respect to their functional behavior. We then obtain AFs acting either as stuck-at faults or as state coupling faults. In the fourth part of the paper, we propose a concurrent approach for testing AFs acting on either the word line decoder or the bit line decoder. The proposed approach allows using a minimal number of programming operations during test application. Finally, we propose a compaction procedure to further reduce the test time of AFs. As a result, huge reductions in test time can be achieved; experiments on a 4 Mbits eFlash have shown that a test time reduction factor of 34x can be obtained when compared to the global eFlash test flow presently used in industry. An additional important feature of the proposed strategy is that it allows testing 100% of other critical faults in eFlashs (stuck-at, transition and state coupling faults) beside full coverage of AFs.*

## 1.    Introduction

The embedded memory resources are continuously increasing and will approach 94% of the System on Chip (SoC) silicon area in the next ten years [1]. This forecast highlights the main issue that we are currently facing in the fields of memory design and test. In fact, due to their high density, memories are considerably impacting SoC test time and are becoming the main contributor of the overall SoC yield loss. Consequently, efficient test solutions and repair schemes for memories are needed.

Different types of memory can be embedded in a SoC, such as SRAM, DRAM, EEPROM and Flash. The increased use of portable electronic devices such as mobile phones and digital camera produces a high demand for Flash memories. A Flash memory is a non-volatile memory that allows programming and erasing memory data electronically [2, 3]. The main-stream operation is based on the floating-gate concept on which charges can be stored and removed. The low-power consumption and high density of eFlash memories make them popular for portable devices. Unfortunately, their high integration density and particular manufacturing process steps make them more and more prone to inter or intra core-cell defects.

In the Flash testing domain, only very few papers can be found in the literature. Generally, all reported studies deal with 1T Flash cells (stacked gate / ETOX) [4, 5, 6] or 1½T Flash cells (split gate / local SONOS) [4, 5, 6]. In [7], we have considered the standard FLOTOX core-cells that are now often used in industry for eFlash memories. We have presented a qualitative analysis of actual defects and failure mechanisms that may occur in the FLOTOX core-cell array. All the defects were reported from experiments on the ATMEL 0.15μm eFlash technology.

The next step after the defect or failure mechanism analysis is i) their modeling from a functional point of view and ii) the development of efficient test strategies. In [8] and [9], we have enumerated a list of functional fault models that cover the actual defects occurring in NOR-based eFlash made of FLOTOX core-cells. Here, we address a particular type of faults, namely the Address decoder Faults (AFs) that occur in the addressing logic of an eFlash. These faults are particularly critical during eFlash testing as fully testing them with ad-hoc solutions is generally very time consuming.

In this paper, we propose a concurrent approach for testing AFs, such that a minimal number of programming operations can be used during test application. Compared

to existing solutions practiced in industry that are based on the use of the Diagonal 0 pattern [10], the proposed approach reduces the test time of AFs by one or more orders of magnitude without loss of fault coverage. As the test time of the proposed approach increases only logarithmically with the size of the memory array (while it increases linearly with the Diagonal 0 pattern approach), the test time reduction increases with the size of the eFlash memory. An important additional feature of the proposed strategy is that it allows testing 100% of other critical faults in eFlashs (stuck-at, transition and state coupling faults) beside full coverage of AFs.

The rest of the paper is organized as follows. Section 2 describes the basic operations of a FLOTOX core-cell and presents an eFlash memory from a functional point of view. Section 3 introduces AFs in the eFlash environment. Section 4 proposes a concurrent solution to test AFs with a significant test time reduction while achieving complete fault coverage of these faults. In Section 5, we propose a compaction procedure to optimize the previous solution and then further reduce the test time. Section 6 shows the interest of using the proposed test strategy to detect 100% of other critical faults in eFlashs (stuck-at, transition and state coupling faults) beside full coverage of AFs. Finally, Section 7 concludes the paper.

## 2. eFlash functional description

An eFlash core-cell is based on the floating-gate (FG) concept. There are two typical mechanisms to transfer electric charges from and into the FG: hot carrier injection (HCI) [3, 11] and the Fowler-Nordheim (FN) tunneling effect [3, 11]. The FN tunneling effect is used for charge injection or removal in FLOTOX core-cells. In our study, we use the FLOTOX core-cell structure presented in Figure 1 [3, 11].
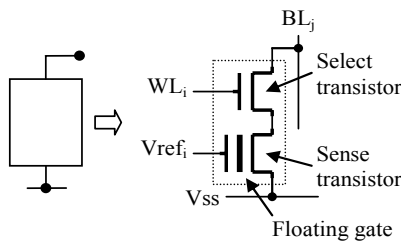


Figure 1: FLOTOX core-cell

The memory cell is composed of a select transistor and a sense transistor. The select transistor allows the selection of the targeted cell. It also avoids disturbances from the high voltage on the bit line when the cell is not selected. The sense transistor contains the floating-gate (FG) and the sense gate.

Three different operations can be performed on an eFlash: Erase, Write and Read. The Erase operation consists in injecting charges in the FG with a specific high voltage

combination. The high voltage must be applied on the Vref node of the sense transistor while its drain must be maintained at ground. During the Erase operation, the core-cell is 'on' and allows the node BLj to be pulled-down at the Vss potential. Note that the erase operation is performed simultaneously on all the cells of the same page and not on a cell by cell basis. At the end of the erase operation, charges in the FG have changed the VT of the sense transistor to a high VT (VTH in Figure 2) which corresponds to a logic '1'.

The Write operation consists in removing electrons from the FG by putting the Vref node at ground while maintaining BLj at a high voltage. With this operation, the sense transistor has a low VT (VTL in Figure 2) which corresponds to logic '0'. We call "VT window" the difference between VTH and VTL.
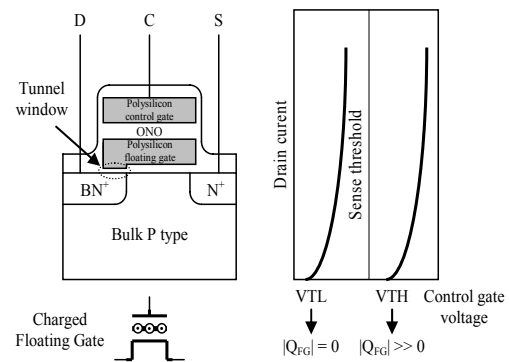


Figure 2: Vt modulation with charge quantity

In eFlash, the read operation is performed by a sense amplifier working in a current measurement mode. The Vref node is set around 0.7v during the read operation. If the sense transistor has a low VT (VTL), it delivers a current (between 10µA to 30µA) and the sense amplifier provides a logic '0' on its output. On the other hand, with the same Vref value, if the sense transistor has a high VT (VTH) there is no current through the bit line and the sense amplifier gives a logic '1'.

Two types of core-cell array can be used to realize an eFlash memory; NOR and NAND-based structures [3, 10]. Here, we present only the NOR-based structure which is the most often used in high-speed applications. In a NOR-based eFlash, core-cells are placed in parallel as shown in Figure 3.

A word line (WLi) is shared by all cells in the same row to form a page address. A core-cell array is defined according to a X-Y plan. Each core-cell in the array can hence be referenced through its X-Y coordinates.

Like in almost all memories, the address decoding is performed by using a tree of pass-gates as represented in Figure 4. In this, we can see an addressing path D defined from values (100) of the address bus (A0A1A2) through a tree of pass-gates.
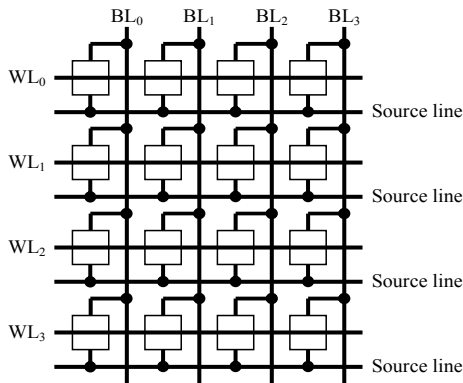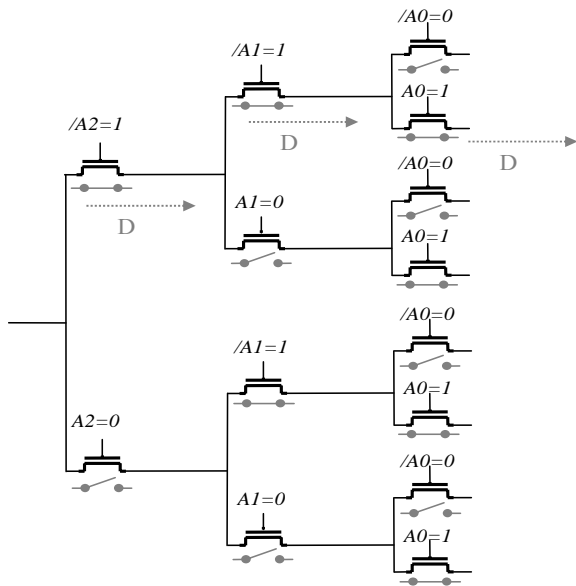
**Figure 3: NOR eFlash structure**



**Figure 4: Tree of pass-gates for the address decoding logic in an eFlash memory**

The functional scheme of an eFlash memory is presented in Figure 5. An eFlash is composed of a core-cell array (CORE), data latches (DLATCH), bit line and word line decoders and sense amplifiers. In addition, eFlash memories need a particular building block: a Charge Pump device for the High Voltage Generation (HVG) allowing the write and erase operations.

An eFlash has two different operating modes: the user mode and the test mode. In the user mode, an eFlash has two possible programming operations. The first one is the Functional Write (FW) operation that allows writing a data into a word. This FW operation is composed of an Auto Erase (AE) of the selected page containing the word to be written (without losing the content of the others words) followed by the write operation of the desired data. The time needed for this FW operation is 4ms for an eFlash designed with the ATMEL 0.15μm technology.

The second possible programming operation is the Page Write mode (PW). The eFlash has a page register allowing the user to program a whole page with the same duration as a FW operation. Also in this mode, the AE operation is preliminary performed.
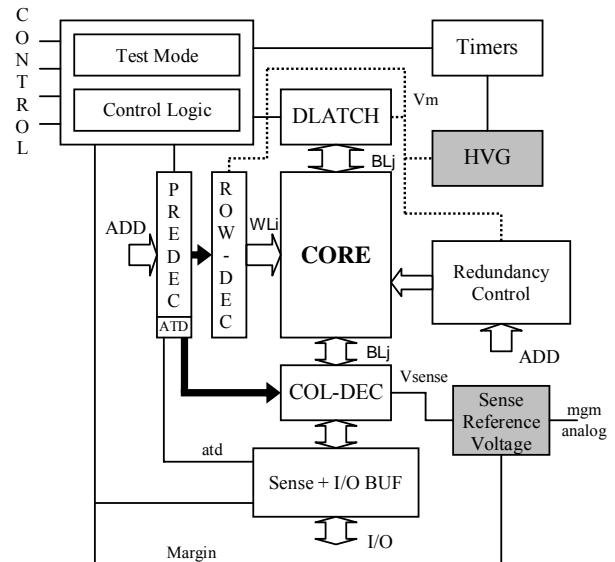


**Figure 5: Functional scheme of eFlash memories**

In the test mode, there are three possible programming operations. The two first operations allow erasing (all 1') or writing (all 0') the eFlash in one step and in 10ms: Chip Erase (CE) and Chip Write (CW). The third programming operation, called Concurrent Chip Write Pattern (CCWP), allows writing a particular pattern in different selected memory addresses. To explain this programming operation, let us consider again the address decoding logic presented in Figure 4. This address decoder has 3-bit addresses and thus allows selecting 8 word lines. To select one word line at a time, the corresponding address is applied on the $A_i$ address-bits and the complemented address is applied on the $/A_i$ address-bits, thus resulting in a unique address selection. When the CCWP operation is used, more than one word line has to be selected. This can easily be done by applying a certain address on the $A_i$ address-bits and a different address on the $/A_i$ address-bits. For example, let us assume that the CCWP operation has to write the same page on WL0, WL2, WL4 and WL6. To do that, we must apply 110 on A2, A1 and A0 respectively and 111 on /A2, /A1 and /A0 respectively. In summary, for a single page selection, the $A_i$ and $/A_i$ bits are complemented bits. When the CCWP operation is used, the value of $A_i$ and $/A_i$ bits depends on which word lines we want to select.

Moreover, note that whatever the operating mode (user and test), the read operation in a ATMEL 0.15μm eFlash memory takes 25ns irrespective of the word or memory size.

## 3.    AFs in an eFlash

Many studies dealing with AF testing for RAM memories exist and are based on March test algorithms [12]. AF testing for eFlash memories is much more critical as existing March algorithms are not applicable in this context for test time reason, i.e. the slow programming time of an eFlash. Practically, AF testing is generally done by performing a diagonal of '0' in the core-cell array [10]. This test pattern, called Diagonal 0 pattern, detects all AFs but its application time remains very high, e.g. about 4s for a 1024 pages eFlash.

Consequently, important efforts have to be done to find efficient AF test solutions that alleviate this test time problem. Beforehand, we present in this section some general basics of AFs in eFlash memories.

### 3.1    Different types of AFs

In this subsection we define different AF combinations involving addresses and their corresponding memory cells. To reduce the complexity of the representation, we only consider two addresses and their two corresponding memory cells. This representation can be applied for the word line decoders as well as for the bit line decoders. We note Addi and Addj the addresses corresponding to memory cells Ci and Cj respectively. We consider that Addi and Addj are addresses of an entire word line or bit line. In the same way, memory cells Ci and Cj can be considered as stand alone memory cells or as a set of memory cells sharing the same word line or bit line addressed with Addi and Addj.

As presented in [12], functional faults within the address decoders may result in the four following subtypes of AFs:

- **subtype1:** with a certain address, no cell will be accessed.
- **subtype2:** there is no address with which this cell can be accessed. A certain cell is never accessed.
- **subtype3:** with a certain address, multiple cells are accessed simultaneously.
- **subtype4:** a certain cell can be accessed with multiple addresses.

Note that AFs must be at least the combination of two subtypes from the above list. From these different subtypes, we can then classify the AFs into two families. These two families are described in Figure 6 and Figure 7.

Figure 6 illustrates the first AF family in which a cell can be accessed by a maximum of one address. This AF family is referred to as Single Access AFs (SA_AFs).
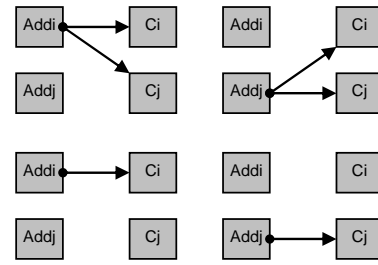


Figure 6: Single Access AFs (SA_AFs)

Figure 7 illustrates the second family of AFs in which a cell can be accessed by more than one address. This second family is referred to as Multiple Access AFs (MA_AFs).
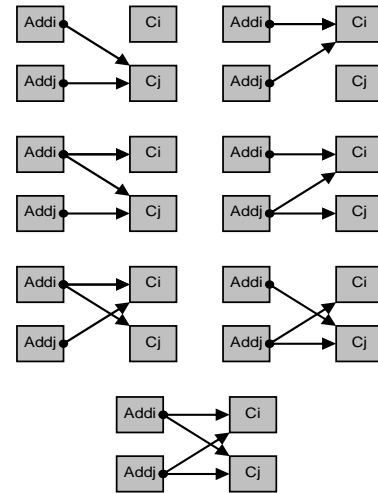


Figure 7: Multiple Access AFs (MA_AFs)

### 3.2    Basic operations for AF detection

As discussed in Section 2, an erase operation performed on the FLOTOX core-cell changes the threshold voltage of the sense transistor to a high Vt, which is then interpreted by the sense amplifier as a logic '1' (no current passes through the bit line during the read operation). During the write operation, the threshold voltage of the sense transistor changes to a low Vt interpreted as a logic '0' by the sense amplifier (a current passes through the bit line during the read operation). From these electrical behaviors, we can make some remarks that will be used to define the AF detection sequence:

- **Remark 1:** A read operation performs on a virgin FLOTOX core-cell (a cell not erased and not written) provides a logic '1' as no current passes through the cell.
- **Remark 2:** If an address does not access any cell, the data read is a logic '1' as the sense amplifier does not measure any current.

- **Remark 3:** If an address selects two core-cells, which are connected to the same bit-line and containing opposite data ('0' and '1'), the data read is a logic '0'. In fact, in that case, the sense amplifier measures the current passing through the cell having the logic '0'. The same behavior occurs if an address selects two core-cells, connected to the same word-line, containing opposite data ('0' and '1'). This assumption is due to eFlash specificities.

### Detection of SA_AFs

Detection of SA_AFs can be performed using a global pattern approach. The solution consists in writing all the cells in one time (chip write: CW) and then read the expected value '0'. This basic detection of SA_AFs is described in Figure 8.

In the first example, SA_AFs #1, a write operation is performed in one time on Addi and Addj. Cells Ci and Cj contain a logic '0'. During the read sequence, Addi gives a logic '0' whereas Addj gives a logic '1'. The fault is sensitized and observed. For the dual representation SA_AF #1', the same phenomenon occurs as for SA_AF #1. For the second examples, SA_AFs #2 and #2', the same behavior occurs but this time the content of the unselected cell is always stuck-at logic '1'. Thus, we can conclude that this simple detection sequence (CW and Read) allows testing all SA_AFs.
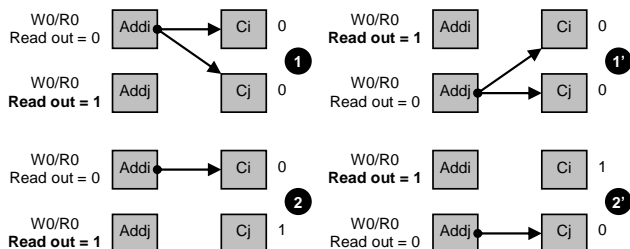


**Figure 8: Basic detection of SA_AFs**

### Detection of MA_AFs

Detection of MA_AFs is more difficult than SA_AFs and thus cannot be obtained with global patterns only, i.e. CE and/or CW patterns. We first initialize the array by performing a chip erase (CE) that initialize all cells at logic '1'. Then, we act a write on Addi only in order to obtain two opposite data on Addi and Addj. Finally, a read operation is performed. Figure 9 presents the MA_AFs detection sequence on Addi and Addj addresses.

For the MA_AFs #3, during the read operation of Addi and Addj, we read twice the content of cell Cj, a logic '0' instead of the expected '0' on Ci and '1' on Cj. For the MA_AFs #3', we also read twice the content of the same cell (Ci) which is fixed to a logic '0'.

The MA_AFs #4' example is more complex than the previous ones because this time, Addj selects two cells (Ci and Cj) containing opposite data. According to Remark 3, the data read with Addj is a logic '0'.
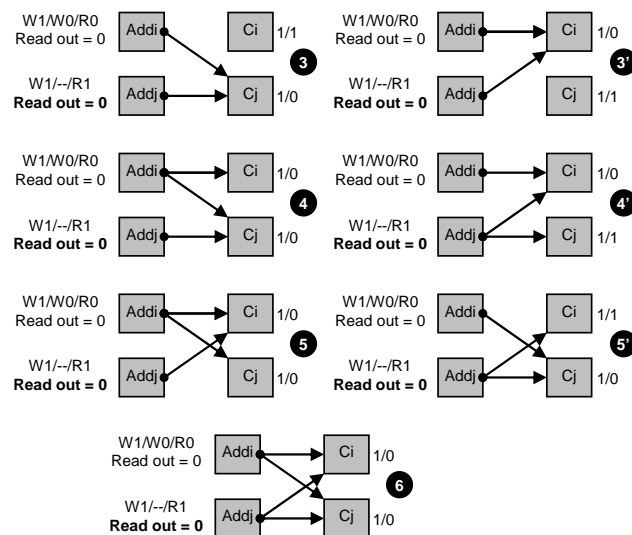


**Figure 9: Basic sensitization of MA_AFs**

For the MA_AFs #5, Addj addresses cell Ci containing a logic '0'. During the read operation, a logic '0' is read for both Addi and Addj. MA_AF #5' behaves as MA_AFs #4' because Addj selects two cells containing opposite data. Finally MA_AFs #6 gives two logic '0' during the read operation.

In each case the expected values were '0' for Ci and '1' for Cj but due to the MA_AFs, a logic '0' is always read on both Addi and Addj addresses. The proposed sequence allows the detection of all MA_AFs.

### 3.3 Similarities between AF in eFlash and RAM

Based on the previous AF analysis, we can try to optimize the global test flow of an eFlash memory by exploiting some similarities existing between SA_AFs / MA_AFs and classical RAM fault models such as stuck-at and coupling fault models.

For example, we can compare faults in the SA_AFs family to stuck-at '1' faults (SAF1) because the faulty address gives always a logic '1'.

For the MA_AFs family, the faulty behavior induced by each fault operates as a coupling fault. In fact, for the MA_AFs #3 and #3', one cell is selected by two different addresses, such that writing at a certain address induces a write on the other one. The same behavior occurs for the erase operation. In that case, a cell can be the victim and the other the aggressor or vice-versa. From the RAM test literature, MA_AFs #3 and #3' behave as state coupling faults, denoted as CFst<0,0> and CFst<1,1>.

For the MA_AFs #4, #4', #5 and #5', a complex faulty behavior like a coupling phenomenon also occurs. In this case, as a write or an erase at any address induces the same operation on its coupled neighborhood, the equivalent fault model is a disturb coupling fault. The equivalent models are: CFdst<w1,↑> and CFdst<w0,↓> corresponding to the two possible operations on a memory cell. Finally, in case of MA_AFs #6, the behavior can also be associated to a disturb coupling fault for which a cell can be either a victim or an aggressor.

By exploiting these similarities, we can develop an efficient test strategy to detect not only all AFs but also almost all other faults that may affect an eFlash.

## 4.    Concurrent approach for AFs testing

In this section, we propose an efficient strategy to test AFs using a minimal number of programming operations. Note that, the proposed test strategy is based on the concurrent programming capability available in almost Flash memories. Nevertheless, if this capability is not built-in, a small logic overhead can be added to the eFlash in order to get it. Moreover, by using the previous similarities, we want to optimize the global eFlash flow to test all fault models from the actual list of faults established in [8, 9]. The following subsections present the proposed test strategy for AFs testing, first in the word line decoder and next in the bit line decoder.

### 4.1    AFs testing in the word line decoder

For detecting SA_AFs, we have previously given the test sequence consisting in a full Chip Write (CW) followed by a read of the entire eFlash array. This sequence is able to detect all possible SA_AFs equivalent to the stuck-at one fault model. Such a test sequence can easily be performed as an eFlash has a special programming mode to act the same operation (erase and write) in one time on all core-cells. For a ATMEL 0.15µm eFlash built with FLOTOX core-cells, this special mode allows erasing or writing the whole memory in 10ms. Thus, the testing time of SA_AFs along the word line decoder will take 10ms for the programming operations and less than two milliseconds for the read operations. Note that the read operation depends on the eFlash memory size but remains always negligible in comparison with the programming time.

The test strategy is more complex for detecting MA_AFs. From Section 3.3, we have seen that MA_AFs could be equivalent to different coupling faults (CFst and CFdst). Coupling fault testing has to consider all combinations involving two cells (aggressor cells and victim cells). For MA_AFs testing, we have to consider all possible conflicts between two addresses which correspond to all possible address couples. Let us assume that the eFlash has M word lines. We can deduce the total number NB of faulty address couples as:

$$NB = M \times \frac{M-1}{2} \qquad (1)$$

Note that Equation 1 does not give the total number of possible MA_AFs but rather the number of possible faulty address couples. For example, let us consider a set of four distinct addresses (A0, A1, A2 and A3) for which the six possible faulty address couples are as follows:

A0 → A1 and A0 → A2 and A0 → A3

A1 → A2 and A1 → A3

A2 → A3

If we consider an eFlash memory of 1024 pages, NB reaches 523776 possible faulty address couples. From this very high number of possible faulty address couples, it is clear that the basic sensitization sequence to test MA_AFs is not applicable as it would take around 2100s.

A first solution to detect MA_AFs should consist in using a March test algorithm able to detect coupling faults. However, as mentioned previously, this class of solution is not applicable as March tests are not compatible with eFlash testing. The main solution generally used to detect AFs in an eFlash consists in using the Diagonal 0 pattern [10]. The goal of this pattern is to write a diagonal of 0's in the array. Figure 10 presents an example of a Diagonal 0 pattern applied on a 16x16 eFlash array.

```
0111111111111111
1011111111111111
1101111111111111
1110111111111111
1111011111111111
...
1111111111111101
1111111111111110
```
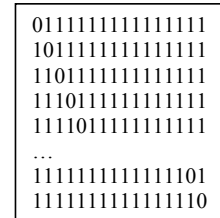
**Figure 10: Diagonal 0 pattern for a 16x16 eFlash**

Let us consider again the example of a 1024 pages eFlash. The resulting number of write operations used to perform the Diagonal 0 pattern will be 1024. This approach reduces considerably the test time of MA_AFs although it still remains significant, i.e. about 4s in this example. It is defined as follows:

$$T_{Diag0} = CE + PW \times NbPage \qquad (2)$$

Note that in Equation 2, we do not consider the time required for the read operation as it is considered as negligible compared to the programming time.

The test strategy proposed in this paper allows detecting all AFs in the word line decoder in a time much more lower compared to the Diagonal 0 pattern. To do that, we combine the March approach used to test coupling faults in word oriented memories [13] with the basic sensitization sequence presented in Section 3.2 for detecting MA_AFs. The resulting test strategy looks like

the method proposed in [14, 15] to detect open and short defects that may affect chip wire interconnections using the IEEE1149.1 standard.

The first pattern of the proposed strategy is a succession of '0' and '1' with a distance of δ = 1 between them. This pattern is equivalent to a checkerboard pattern. Here we have represented this pattern on a set of eight different addresses:

A0  A1  A2  A3  A4  A5  A6  A7

0   1   0   1   0   1   0   1

If we consider address A0 and the seven address couples possible with this address, we see that our first pattern is able to detect half of all MA_AFs combinations involving this address. Generally, with address A0 as a reference, the possible MA_AFs involving A0 and the addresses located at a distance of (k+1)*δ are tested with k ∈ {0, 2, 4 , …} and (k+1)*δ < number of addresses. In the previous example, the addresses coupled with A0 are A1, A3, A5 and A7.

The detection of the other address couples is done in the same way. The second pattern we apply is obtained by setting δ = 2:

A0  A1  A2  A3  A4  A5  A6  A7

0   0   1   1   0   0   1   1

With this second pattern, we can test the possible MA_AFs involving A0 and addresses located at a distance of (k+1)*δ with δ = 2 from A0. The possible faulty address couples are A0/A2 and A0/A6. With the same method, we can build a third pattern by setting δ = 4:

A0  A1  A2  A3  A4  A5  A6  A7

0   0   0   0   1   1   1   1

This third pattern allows the detection of the remaining faulty address couple A0/A4. Here too, the distance between A0 and A4 is (k+1)*δ with δ = 4.

With the three previous patterns, all MA_AFs involving A0 are detected. Moreover, due to the periodicity of each pattern all the others possible faulty address couples corresponding to the others MA_AFs are tested.

The entire sequence presents a modulo 2 periodicity as the logic '0' and the logic '1' duration is multiplied by 2 between two consecutive patterns. We can also compute the limit number of patterns to generate with the help of the relation:

$$\log_2 W \tag{3}$$

where B is the total number of addresses.

In our example involving eight addresses, the number of patterns to generate is 3. The main advantage of the proposed sequence is the logarithmic relationship defining the number of patterns to generate. In fact, if we consider an eFlash memory of 1024 word lines, the proposed sequence will contain 10 patterns only. These 10 patterns can be applied using the special programming mode of the eFlash (concurrent chip write pattern: CCWP) resulting in 10 programming operations compared to 1024 programming operations in case of the Diagonal 0 pattern.

In order to illustrate the proposed test strategy, let us consider an eFlash consisting of 8 word lines and 4 bit lines. From Equation 3, three patterns are needed to detect all MA_AFs. These three patterns are applied using the CCWP (concurrent chip write pattern). This specific eFlash programming mode requires a chip erase (CE) before each pattern programming as presented in Figure 11. This example shows that six programming operations are required to apply the three initial test patterns detecting all MA_AFs in the word line decoder.

|         | CE   | CCWP.1 | CE   | CCWP.2 | CE   | CCWP.3 |
|---------|------|--------|------|--------|------|--------|
| $WL_0$  | 1111 | 0000   | 1111 | 0000   | 1111 | 0000   |
| $WL_1$  | 1111 | 1111   | 1111 | 0000   | 1111 | 0000   |
| $WL_2$  | 1111 | 0000   | 1111 | 1111   | 1111 | 0000   |
| $WL_3$  | 1111 | 1111   | 1111 | 1111   | 1111 | 0000   |
| $WL_4$  | 1111 | 0000   | 1111 | 0000   | 1111 | 1111   |
| $WL_5$  | 1111 | 1111   | 1111 | 0000   | 1111 | 1111   |
| $WL_6$  | 1111 | 0000   | 1111 | 1111   | 1111 | 1111   |
| $WL_7$  | 1111 | 1111   | 1111 | 1111   | 1111 | 1111   |

**Figure 11: MA_AFs testing in the word line decoder**

## 4.2    AF testing in the bit line decoder

The problem of AF detection in the bit line decoder is equivalent to the AF detection in the word line decoder. First, detecting SA_AFs is done in the same way as previously, i.e. a chip write CW followed by a read operation. Secondly, detecting MA_AFs is performed with the same type of patterns than for MA_AFs in the word line decoder. For example, Figure 12 presents the patterns in the case of a 8-bit line eFlash. For simplicity, Figure 12 only represents the page corresponding to WL0.

As for detecting MA_AFs in the word line decoder, the number of patterns to apply is given by the following relation:

$$\log_2 B \tag{4}$$

where W is the total number of bit lines. Moreover, as for testing MA_AFs in the word line decoder, we need to perform a chip erase (CE) between each CCWP pattern.

|        | $BL_0$ | $BL_1$ | $BL_2$ | $BL_3$ | $BL_4$ | $BL_5$ | $BL_6$ | $BL_7$ |
|--------|------|------|------|------|------|------|------|------|
| CE     | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    |
| CCWP.1 | 0    | 1    | 0    | 1    | 0    | 1    | 0    | 1    |
| CE     | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    |
| CCWP.2 | 0    | 0    | 1    | 1    | 0    | 0    | 1    | 1    |
| CE     | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    |
| CCWP.3 | 0    | 0    | 0    | 0    | 1    | 1    | 1    | 1    |

**Figure 12: MA_AFs testing in the bit line decoder**

## 4.3 AF testing summary

In this section, we have proposed a test strategy to detect all AFs that may affect the address decoders. Here we summarize the complete approach on a 4x4 eFlash. Figure 13 represents the sequence to test AFs in the word line decoders. A CW is first applied on the array followed by a read operation to detect all SA_AFs. Then, the four patterns allowing the MA_AFs detection are applied.

| CW | CE | CCWP.1 | CE | CCWP.2 |
|----|----|--------|----|--------|
| 0000 | 1111 | 0000 | 1111 | 0000 |
| 0000 | 1111 | 1111 | 1111 | 0000 |
| 0000 | 1111 | 0000 | 1111 | 1111 |
|  |  | 1111 |  | 1111 |
| ↑ |  | ↑ |  | ↑ |
| Read |  | Read |  | Read |

Figure 13: Word lines AFs testing

In the same way, Figure 14 presents the test sequence to detect AFs in the bit line decoder.

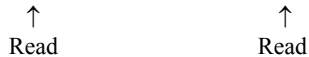| CE | CCWP.1 | CE | CCWP.2 |
|----|--------|----|--------|
| 1111 | 0101 | 1111 | 0011 |
| 1111 | 0101 | 1111 | 0011 |
| 1111 | 0101 | 1111 | 0011 |
| 1111 | 0101 | 1111 | 0011 |
|  | ↑ |  | ↑ |
|  | Read |  | Read |

Figure 14: Bit lines AFs testing

This sequence allows the detection of all MA_AFs in the bit line decoder. Note that, the detection of SA_AFs in the bit line decoder is already done during the word-line AFs testing by the CW operation in the previous sequence presented Figure 13.

We can formulate the global test time of the proposed test strategy as follow:

$$T = CW + (CE + CCWP) \times (\log_2 B + \log_2 W) \quad (5)$$

in which CW, CE and CCWP each take 10ms. Once again, we do not consider in Equation 5 the time required for the read operation as it is negligible compared to the programming time.

Let us now compare the resulting test time of the proposed test strategy with that of the Diagonal 0 pattern. Considering a 1024x1024 eFlash array (B = 1024 and W = 1024), our test strategy will take around 410ms compared to 4s for the Diagonal 0 pattern. Our strategy reduces by a factor of 10 the resulting test time without degradation of the AF coverage.

## 5. Optimization of the AF test strategy

We have seen in Section 4 how to test AFs by considering separately the word line and the bit line address decoders.

In this section we show how these two approaches can be merged together to further reduce the resulting test time.

Let us first deal with the problem of MA_AFs testing. Thanks to the modulo 2 periodicity of the proposed test strategy, a compaction of the two previous sequences (Figures 13 and 14) is possible. In this compaction process, we consider the word lines and the bit lines as a set of cells sharing the same address line. Considering the basic MA_AFs detection sequence, we have to write two opposite data in two different word lines or bit lines. Let us consider the 8x8 eFlash example presented in Figure 15. In this sequence, each pattern (CCWP.1 to CCWP.3) is preceded by a CE and followed by a read operation.

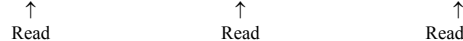| CE | CCWP.1 | CE | CCWP.2 | CE | CCWP.3 |
|----|--------|----|--------|----|--------|
| 1111 1111 | 0101 0101 | 1111 1111 | 0011 0011 | 1111 1111 | 0000 1111 |
| 1111 1111 | 1111 1111 | 1111 1111 | 0011 0011 | 1111 1111 | 0000 1111 |
| 1111 1111 | 0101 0101 | 1111 1111 | 1111 1111 | 1111 1111 | 0000 1111 |
| 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | 0000 1111 |
| 1111 1111 | 0101 0101 | 1111 1111 | 0011 0011 | 1111 1111 | 1111 1111 |
| 1111 1111 | 1111 1111 | 1111 1111 | 0011 0011 | 1111 1111 | 1111 1111 |
| 1111 1111 | 0101 0101 | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 |
| 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 |
|  | ↑ |  | ↑ |  | ↑ |
|  | Read |  | Read |  | Read |

Figure 15: Compaction of the proposed test strategy for MA_AFs detection in a 8x8 eFlash

As can be seen in the example of Figure 15, each pattern allows the detection of MA_AFs in the word line decoder and in the bit line decoder as well, since two opposite data are written in two different word lines or bit lines.

So, with the help of such a compaction technique, we reduce the resulting number of programming operations compared to the initial MA_AFs test strategy presented in Section 4. However, when applying this compaction technique, we must take care of the eFlash topology (square or rectangle). In case of a square array, the compaction has already been shown in Figure 15 for a 8x8 eFlash. The problem is different in case of a rectangle array. Let us consider a 4x8 eFlash example presented in Figure 16. This time, the array is composed of 8 bit lines and 4 word lines. In such case, detection of MA_AFs in the word line decoder requires 2 patterns (log2 4 = 2) while 3 patterns (log2 8 = 3) are needed for the bit line decoder. As more patterns are required for the bit line decoder, the last pattern is applied to the whole memory (see CCWP.3 in Figure 16).

In summary, for an eFlash using a rectangle array, the total number of patterns needed to detect all MA_AFs in both decoders is defined by the widest decoder. Then, we can define the global test time of the compacted test strategy as follow:

$$T_{AF_{op}} = CW + (CE + CCWP) \times \max(\log_2 B, \log_2 W) \quad (6)$$

in which, once again, we do not consider the time required for the read operation.
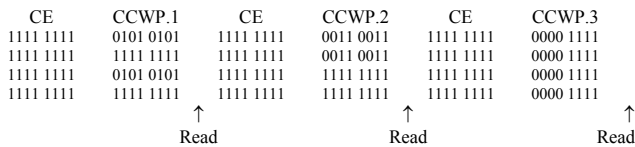
```
CE        CCWP.1      CE        CCWP.2      CE        CCWP.3
1111 1111  0101 0101   1111 1111  0011 0011   1111 1111  0000 1111
1111 1111  1111 1111   1111 1111  0011 0011   1111 1111  0000 1111
1111 1111  0101 0101   1111 1111  1111 1111   1111 1111  0000 1111
1111 1111  1111 1111   1111 1111  1111 1111   1111 1111  0000 1111
           ↑                      ↑                      ↑
         Read                   Read                   Read
```

**Figure 16: Compaction of the proposed test strategy for MA_AFs detection in a 4x8 eFlash**

Let us now compare the resulting test time of the proposed compacted AFs test strategy with that of the Diagonal 0 pattern. Considering again a 1024x1024 eFlash

array (B = 1024 and W = 1024), the compacted AFs test strategy will take around 210ms compared to 4s for the Diagonal 0 pattern. Our strategy reduces by a factor of 20 the resulting test time without degradation of the AF coverage.

## 6. Impact of the AF test strategy on the eFlash test flow

In this section, we show the interest of using the proposed strategy for testing AFs in the global eFlash test flow. Let us first start by a summary of our previous work on the eFlash array. Studies presented in [7, 8, 9] have conducted to a fault list related to actual defects occurring in the array of FLOTOX core-cells. As a main result, we give the following list of faults that have to be tested in order to guarantee the eFlash array functionality:

- Stuck-at-faults

- Transition faults

- State coupling faults (CFst<0,0>, CFst<1,1>) limited to core-cells in the same word line

- To this fault list, we obviously add AFs.

A global test flow is generally used to test eFlash [10]. It is composed by basic patterns as CE (Chip Erase), CW (Chip Write), CKB (Checkerboard), CKBI (Checkerboard inverse) and Diag0 (Diagonal 0 pattern). Considering the previous fault list, we have to check if this global test flow is able to detect all the faults. To do that, we have developed a fault simulation tool to evaluate the efficiency of the global test flow. The resulting fault coverages are reported in the second column of Table 1. From these data, we can see that the global test flow is able to detect 100 % of SAFs, TFs and AFs. Concerning the CFst model, only the lines CFst* are important as they reflect the CFst<0,0> and CFst <1,1> fault coverage limited to cells sharing the same word line. The last line in Table 1 reports the resulting test time for three eFlash sizes (1, 2 and 4Mbits eFlash). Once again, the time required for the read operations is negligible compared to programming times.

With the help of our fault simulation tool, we have also evaluated the efficiency of the proposed test strategy in both its initial version (Section 4) and its improved version

with compaction (Section 5). Results are reported in Column 3 and Column 4 respectively. A first comment is that both AFs test strategies detect all SAFs, TFs and AFs. The AFs test strategy in its compacted version offers the better test time but it does not detect all CFst*. Only the initial AFs test strategy allows detecting all CFst*.

These comparisons show the interest of the proposed test strategy in detecting all AFs as well as all faults that may affect the eFlash array. Moreover, the test time of the proposed approach increases only logarithmically with the size of the memory array and not linearly as for a conventional global test flow. From the test time data in Table 1, the initial AFs test strategy reduces the test time by a factor of 9.8, 18.6 and 34 for a 1 Mbits, 2 Mbits and 4 Mbits eFlash respectively.

|  |  | Global test flow | AFs test strategy | Compacted AFs test strategy |
|---|---|---|---|---|
|  | SAF | 100 % | 100 % | 100 % |
|  | TF | 100 % | 100 % | 100 % |
|  | AF | 100 % | 100 % | 100 % |
| CFst | <0,1> | 55 % | 100 % | 25 % |
|  | <1,0> | 55 % | 67 % | 25 % |
| CFst* | <0,0> | 55 % | 100 % | 50 % |
|  | <1,1> | 55 % | 100 % | 50 % |
| Test time | 1 Mbits eFlash | ~ 4.1s | ~ 420ms | ~ 220ms |
|  | 2 Mbits eFlash | ~ 8.2s | ~ 440ms | ~ 240ms |
|  | 4 Mbits eFlash | ~ 16.4s | ~ 480ms | ~ 280ms |

**Table 1: Test sequence evaluations**

## 7. Conclusion

In this paper, we have address AF testing in eFlash memories. These faults have been studied and classified in two families, SA_AFs and MA_AFs, according to their faulty behaviors. Based on this classification, we have proposed a test strategy that reduces drastically the resulting test time compared to the Diagonal 0 pattern.

This new AFs test strategy has been compared with a global test flow generally used to test eFlash. From these comparisons, we have shown that the proposed solution is the only one that covers all fault models that may affect an eFlash, i.e. all SAFs, TFs, AFs, and CFst are tested. Moreover, our solution drastically reduces the test time compared to the global test flow generally used; by a factor of 34 for a 4Mbits eFlash.

## 8. References

[1] Semiconductor Industry Association (SIA), "International Technology Roadmap for Semiconductors (ITRS)", http://www.sia-online.org/home.cfm, 2005.

[2] "IEEE Standard Definitions and Characterization of Floating-gate Semiconductor Arrays", IEEE 1005-1998, Revision of the IEEE std. 1005-1991.

[3] P. Pavan et al, "Flash Memory Cells – An Overview", Proc. of the IEEE, vol. 85, N° 8, August 1997, pp. 1248-1271.

[4] M. Mohammad and K. Saluja, "Flash Memory Disturbances: Modeling and Test", Proc. of the IEEE VLSI Test Symposium, pp. 218-224, 2001.

[5] M. Mohammad and K. Saluja, "Simulating Disturb Faults in Flash Memories Using SPICE Compatible Electrical Model", IEEE Trans. on Electron Devices, vol. 50, N° 11, November 2003, pp. 2286-2291.

[6] Y.-L. Horng, J.-R. Huang and T.-S. Chang, "A Realistic Fault Model for Flash Memories", Proc. of IEEE Asian Test Symposium, pp. 274-281, 2000.

[7] O. Ginez, J.-M. Daga, M. Combe, P. Girard, C. Landrault, S. Pravossoudovitch and A. Virazel, "An Overview of Failure Mechanisms in Embedded Flash Memories", Proc. of IEEE VLSI Test Symposium, pp. 108-113, 2006.

[8] O. Ginez et al, "Retention and Reliability Problems in Embedded Flash Memories: Analysis and Test of Defective 2T-FLOTOX Tunnel Window", to appear in Proc. of IEEE VLSI Test Symposium, 2007.

[9] O. Ginez et al, "Electrical Simulation Model of the 2T-FLOTOX Core-Cell for Defect Injection and Faulty Behavior Prediction in eFlash Memories", to appear in Proc. of IEEE European Test Symposium, 2007.

[10] A.-K. Sharma, "Semiconductor Memories: Technology, Testing and Reliability", IEEE Press, Piscataway, 1997.

[11] K. Itoh, "VLSI Memory Chip Design", Springer, Berlin, Germany, 2001.

[12] A.J. van de Goor, "Testing Semiconductor Memories, Theory and Practice", COMTEX Publishing, 1998.

[13] A.J. van de Goor and I.B.S. Tlili, "March tests for word-oriented memories", Proc. of Design Automation and Test in Europe, pp. 501-506, 1998.

[14] W.H. Kautz, "Testing of Faults in Wiring Interconnects", IEEE Trans. Computers, vol. 23, N° 4, April 1974, pp. 358-363.

[15] M. Abramovici, M.A. Breuer, A.D. Friedman "Digital Systems Testing and Testable Design", Computer Science Press, 1990.