EDACs and Test Integration Strategies for NAND Flash memories

Stefano DI CARLO, Michele FABIANO, Roberto PIAZZA, Paolo PRINETTO
-
Politecnico di Torino
Dipartimento di Automatica ed Informatica (DAUIN)
{Stefano.Dicarlo, Michele.Fabiano, Paolo.Prinetto}@polito.it, {R.Piazza.86}@gmail.com

***Abstract*—Mission-critical applications usually presents several critical issues: the required level of dependability of the whole mission always implies to address different and contrasting dimensions and to evaluate the tradeoffs among them. A mass-memory device is always needed in all mission-critical applications: NAND flash-memories could be used for this goal. Error Detection And Correction (EDAC) techniques are needed to improve dependability of flash-memory devices. However also testing strategies need to be explored in order to provide highly dependable systems. Integrating these two main aspects results in providing a fault-tolerant mass-memory device, but no systematic approach has so far been proposed to consider them as a whole. As a consequence a novel strategy integrating a particular code-based design environment with newly selected testing strategies is presented in this paper.**

I. Introduction

Mission-critical applications usually presents several critical issues: the required level of dependability of the whole mission always implies to address different and contrasting dimensions and to evaluate the tradeoffs among them. A mass-memory device is always needed in all mission-critical applications. NAND flash-memories could be used for this goal: in fact on the one hand they are nonvolatile, shock-resistant and powereconomic but on the other hand they have several critical drawbacks [12], [13], [14], [21]. Two main strategies have been adopted in order to improve dependability of flash-memory devices: on the one hand Error Detection And Correction (EDAC) techniques and on the other hand testing strategies and algorithms. However these two aspects have always been addressed separately: no systematic approach has so far been proposed to consider them as a whole. A novel strategy integrating a particular code-based design environment with newly selected testing strategies is presented in this paper.

The rest of the paper is organized as follows: Section II briefly addresses the reliability issues of NAND flash memories, Section III discusses the possible testing strategies, Section IV introduces the adopted correcting codes, Section V explains the architecture of our system integrating EDACs and testing while Section VI concludes our work.

II. Flash Memory Reliability Issues

Flash memories present several reliability issues especially because of their way of functioning [12]. Firstly the continuous scaling down of technology is making more critical the single cell reliability, while the reduction of the distance among the cells can lead to *Cell to cell Interferences* [14], [21]. Secondly the high electric fields needed for the program/erase operations and the thinner and thinner oxides imply charge losses and fluctuations problems (i.e., *Cycling Induced Degradation*). Finally operations on flash memories are intrinsically analog, forcing to have a high-dependable analog circuitry [3].

The most important reliability issues of NAND Flash memories can be split into *Permanent Faults* and *Transient Faults*.

A. Permanent Faults

Permanent Faults refer to permanent physical defects or conditions. They can be divided into three subclasses:

- *Flash Disturbances*: disturbances are faulty behaviors resulting from the floating gate (FG) technology [12], thus they do belong to flash memories, but not to the other memories [8], [19], [20], [23];
- *Circuit Level Faults*: defects can be present in the physical structure of the cell and are modeled as resistors and capacitors, while effects are analogically analyzed [11];
- *Cell to cell Interferences*: scaling down the physical distance between adjacent cells makes transistors closer, influencing each other and easily leading to change the state of the cell [14], [21];

[9] is a detailed exploration of Permanent Faults of NAND flash memories.

B. Transient Faults

Transient Faults refer to those physical phenomena or conditions that are not permanent, thus occurring randomly: they are usually due to charge trapping, charge fluctuations phenomena and are strictly related to the cycling induced degradation. These faults can be divided into four subclasses:

- *Cycling Induced Degradation*: since tunneling effect is exploited for program/erase operation, the needed high electric fields degrade the quality of the oxide; this results in a limited number of cycles per cell, after which reliability is not guaranteed anymore; the most important causes of the oxide degradation are the charge trapping and detrapping phenomena: charges in the oxide are accumulating cycle after cycle and eventually result in

978-1-4244-9556-6/10/$26.00 ©2010 IEEE

a shift of the V_{th} levels (i.e., *trapping*), while charges detrapping out the oxide depends logarithmically on time and on temperature (i.e., *detrapping*) [3], [13];
- *Anomalous Stress Induced Leakage Current (SILC)*: since oxide has defects and is stressed by high electric fields, electrons can tunnel from the inverted substrate to the FG in the case of positive gate stress; this effect is also known as *Trap-Assisted Tunneling (TAT)*, is particularly rare and is generated from the cooperation of close defects in the oxide and can lead to a shift of the V_{th} [13];
- *Random Telegraph Noise (RTN)*: readout operation can suffer from drain current fluctuations; the on-state current is said to be affected by RTN [13];
- *Few-electrons issue*: this issue usually refers to randomness of tunneling effect and to the related impossibility of engineering the data-retention time [13];

[13] is a detailed exploration of all the reliability issues of flash memories related to Transient Faults.

III. Testing and Hard Errors Estimation

Testing strategies need to be explored in order to provide highly dependable systems. In fact flash-memories are prone to several kind of faults due to the floating gate technology [12] and a plenty of strategies have been proposed to tackle them [8], [17], [18], [19], [22].

[9] provides a complete exploration of the fault models of NAND flash memories and of the related testing strategies: in addition it presents a specific test algorithm for NAND flash memories, which is an extension of the Bridging Fault&Disturbances (BF&D) algorithm presented in [15].

In this section we want to provide a simple and effective method to estimate the maximum number of *Hard errors* for each page referring to the proposed test method of [9]. Our *Extended BF&D* algorithm programs the memory block twice: firstly with a chessboard pattern and secondly with the opposite chessboard pattern. After the whole block has been programmed, errors are detected reading *k*-times the pages, from top to bottom and vice versa by twos: therefore testing requires each data page to be read a total of k times.

Assuming $k = 2$ and naming $N_{TopBottom_i}$ and $N_{BottomUp_i}$ the bit errors counts, where i is the $i-th$ test section, Equation 1 is the empirical formula for the hard errors count estimation.

$$N_{BF\&D-Ext} = max(N_{TopBottom_1}, N_{BottomUp_1}) + \\ + max(N_{TopBottom_2}, N_{BottomUp_2}) \quad (1)$$

$N_{BF\&D-Ext}$ is the estimated number of *Hard errors* for a specific page: this estimation is simply computed choosing the maximum between $N_{TopBottom_i}$ and $N_{BottomUp_i}$. The overall estimation of hard errors is achieved summing the contributes of the $k = 2$ sections. Equation 1 let us associate each data page to a worst case number of hard errors.

IV. Error Detection And Correction

Error Detection And Correction (EDAC) is respectively the ability to detect the presence of errors and to correct them: EDAC techniques are needed to improve dependability of flash-memories. Designers should evaluate the most proper choice for their design, addressing many critical issues [5].

Each type of error correction code (ECC) is able to correct a well-defined number of errors, which is referred as the *error correction capability* of the particular code: particular tests are need in order to understand the correction capability needed by the specific NAND flash device.

A. Endurance Test

The occurrence of random errors in NAND flash memories is related to the transient faults described in Subsection II-B which are in turn strictly related to the cycling induced degradation issue. One method to characterize the reliability level of the technology with respect to the cycling induced degradation issue is known as *Endurance Test* [3].

The aim of the Endurance test is to characterize a specific NAND flash memory device in terms of charge losses and fluctuation phenomena versus cycling (i.e., write/erase cycles): it estimates the needed error correction capability and its changes with cycling. [7] exploits this approach: firstly the target NAND flash memory is carefully studied and tested to obtain as many information as possible on it. Then a typical endurance test can provide the following settings:

- each page is cycled until the blocks become completely unreliable (i.e., the data-sheet life time is reached);
- a subset of M blocks is tested to trade off the test time;
- at each cycle, data are read from blocks to detect errors;

The test is performed without ECC and at room temperature. At the end all the information about the errors of each block in function of cycling are collected and can be elaborated (e.g., results can be averaged, taken in the worst case, etc.).

[7] shows that the occurrence of errors is quite rare up to a certain aging, while a rapid growth of the errors reveals getting closer to the maximum number of cycles per block. In conclusion endurance tests help to choose the error tolerance (i.e., correction capability) needed to tackle the random errors related to cycling induced degradation.

B. Exploring the Tradeoffs of ECCs

Each ECC is characterized by it own error correction capability: a code with higher correction capability than another one is intuitively more "powerful". However the provided correction capability is strictly linked to the complexity devised to accomplish it: different codes provide different error correction capabilities with different complexity. As a consequence designers have to choose the most suitable solution for their specific design, evaluating all the tradeoffs and exploring all the design dimensions [6].

Binary Bose-Chaudhuri-Hocquenghen (BCH) codes [1], [10], [16] are a well known correcting code technique for NAND flash-memories. [6] presents a brief overview of these codes and proposes a novel design environment aimed at supporting the design of BCH codes with a user-selectable error correction capability for NAND flash-memories: moreover it is under-integration with our FLARE design environment [4].

V. EDAC and Testing Integration

Section III showed that testing strategies are capable to esteem the number of possible hard errors per page: the main idea is to exploit these information as a feedback for the EDAC environment presented in [6], which in turn would automatically adapt its correcting power to the current state of the overall system.

The integration of the EDAC and the BIST strategies practically leads to the design of a fault-tolerant system with an adaptive error correction capability for flash-memory based mission-critical mass-memory devices.

A. Motivations

First of all the reliability characterization of flash-memories discussed in Section II and the optimization of the overall performances imply the need of a *user-selectable* tool in terms of error correction capability. In addition its *automatic* and *parametric* features help to tackle the high complexity of the EDAC system and to explore the design space tradeoffs [6].

Subsection II-A showed that pages can be affected from *Permanent Faults*: these faults could lead to *Hard Errors* [2] if proper conditions occur. The testing strategies of Section III can be adopted to esteem the number of hard errors per page: this figure could be exploited to evaluate the most proper error correction capability, making each page fault tolerant.

The integration of EDAC and testing strategies practically leads to the design of a fault-tolerant system.

B. System Overview

Figure 1 shows an overview of the system integrating EDAC and test. It is mainly composed by three functional blocks:

1) Test Unit: it let us know the the maximum number of hard errors occurred in a specific page due to permanent faults.

2) EDAC System: this block is the one of [6]; the user-selectable error correction system can code each written page with a different error correction capability t, thus we can assign different error correction capabilities to different pages and also change these values dynamically.

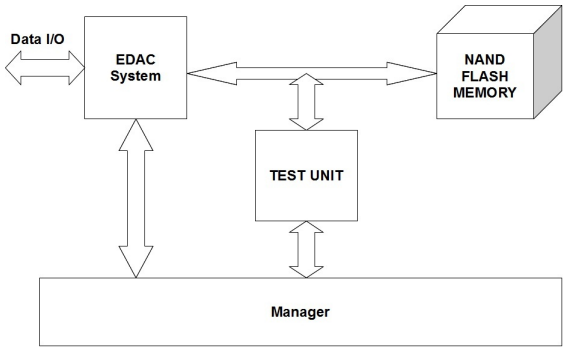

Figure 1. System Overview of EDACs and Test Integration

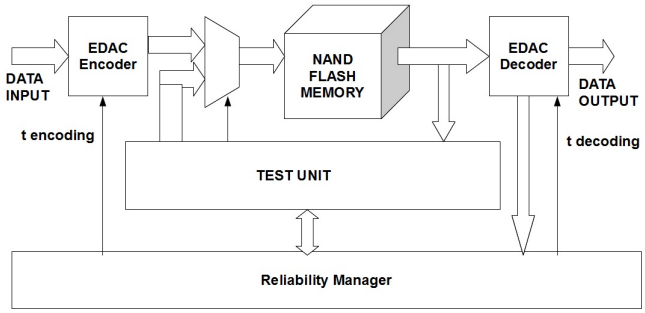

Figure 2. A more detailed System Overview

3) Reliability Manager: this block manages the EDAC System and the Test Unit block; it collects and elaborates the results of the Test Unit and feedbacks the EDAC System, setting the proper correction capability to pages; in addition it is responsible of the activation/deactivation of the test unit.

In the sequel we analyze more in detail the principles of working of these functional block.

C. Testing strategies

Several strategies are feasible for testing each block of the memory: let assume to exploit the algorithm of [9]. The test is data-destructive, so it can be executed only on blocks ready for deletion: here we will focus on the triggers for testing.

First of all each block can be tested before its first usage: it can be easily accomplished by testing each block at the first system boot. Therefore each page of data will be featured by a certain number of detected hard errors. Then we could test blocks for "unexpected behaviors": a possible strategy is that, if more than k errors are detected, the block is scheduled for testing as soon as possible. Another possible "unexpected behavior" is a decoding failure. However, at the end of the test session, the $i-th$ page will present N_{TEST_i} hard errors.

D. Static and Dynamic Error Correction Capability

The user-selectable EDAC System and the Test Unit are both managed by the so called *Reliability Manager* through a proper strategy: Figure 2 is a more detailed view of this block.

There are two main source of information for setting the error correction capability t: *Static* and *Dynamic* information.

Static information are based on considerations on the reliability of the adopted technology: these information can be obtained from *Endurance Tests* (ET, see Section IV) and are equal for all the pages of the memory. These information contribute to define the *Static error correction capability* t_{static}: Equation 2 shows this concept.

$$t_{static}(W/E_{cycles})_{TOT} = t_{static}(W/E_{cycles})_{ET} \quad (2)$$

Dynamic information are specific for each page and come both from test results and decoding results. In fact dynamic information are represented with the two following variables:

- N_{TEST_i}: it is the number of *Hard Errors* of the $i-th$ page detected during the most recent test session;

- N_{ECC_i}: it is the maximum number of errors detected/corrected in the $i-th$ page ever;

They define the *Dynamic error correction capability* $t_{dynamic_i}$ and is the one applied to the $i-th$ page. Equation 3 shows the dependence with N_{TEST_i}, N_{ECC_i} and t_{static}.

$$t_{dynamic_i} = f\left[t_{static}\left(W/E_{cycles}\right)_{TOT}, N_{TEST_i}, N_{ECC_i}\right] \tag{3}$$

f is the relation which practically defines the $t_{dynamic_i}$ to apply on the $i-th$ page: f will be referred as a *policy*.

E. Control Policy: an Example

A possible policy, combining together static and dynamic information, is reported in Equation 4: $t_{static} = t_{static}\left(W/E_{cycles}\right)_{TOT}$ for sake of simplification.

$$t_{dynamic_i} = \tag{4}$$

$$\begin{cases} min\left(t_{static} + N_{TEST_i}, t_{MAX}\right) & N_{ECC_i} \leq N_{TEST_i} \\ min\left(N_{ECC_i} + t_{static}, t_{MAX}\right) & N_{ECC_i} > N_{TEST_i} \end{cases}$$

N_{TEST_i} and N_{ECC_i} have null initial value: in particular $N_{ECC_i} = t_{MAX}$ in case of decoding failure, in order to force $t_{dynamic_i} = t_{MAX}$.

Firstly Equation 4 exploits the technological (i.e. static) information as reference, secondly run-time (i.e. dynamic) data coming from the EDAC system adapt them to the specific need of each page: whenever more errors than the number of hard errors are detected, the dynamic correction capability is conveniently increased. N_{ECC_i} is updated after each read (i.e. *decode*) operation, but also other strategies can be adopted.

In other words the *Reliability Manager* works as follows:

1) it collects and elaborates the outputs of decoding them, conveniently updating N_{ECC_i};
2) if needed, it activates the proper testing strategies, updating the N_{TEST_i};
3) it updates all the $t_{dynamic_i}$ according to the policy of Equation 4, with the values of N_{ECC_i} and N_{TEST_i} found at points 1 and 2;

The applied $t_{dynamic_i}$ is always covering all the *Hard errors* and also a margin of errors determined by t_{static}: a complete fault tolerant system is achieved, with respect both to transient and permanent faults [9].

VI. Conclusions

Error Detection And Correction (EDAC) and testing strategies are the main techniques to improve dependability of flash-memory devices. This paper considered them as a whole and presented an integrated design environment with powerful binary BCH codes and efficient testing methodologies: this lead us to a complete fault tolerant system, with respect both to transient and permanent faults.

Moreover the flexibility given by the interchangeable policies and algorithms will let us exploit this powerful design environment for a more detailed exploration of the dependability of NAND flash memories.

References

[1] J. Adamek. *Foundations of Coding: Theory and Applications of Error-Correcting Codes, with an Introduction to Cryptography and Informat*. John Wiley & Sons, Inc., New York, NY, USA, 1991.

[2] Robert Baumann. Soft errors in advanced computer systems. *IEEE Des. Test*, 22(3):258–266, 2005.

[3] J. Brewer and M. Gill. *Nonvolatile Memory Technologies with Emphasis on Flash(A Comprehensive Guide to Understanding and Using Flash Memory Devices*. 2008.

[4] M. Caramia, S. Di Carlo, M. Fabiano, and P. Prinetto. Flare: A design environment for flash-based space applications. *Proceedings of High Level Design Validation and Test Workshop, 2009. HLDVT 2009. IEEE International*, pages 14 –19, nov. 2009.

[5] M. Caramia, S. Di Carlo, M. Fabiano, and P. Prinetto. Flash-memories in space applications: Trends and challenges. *Proceedings of East-West Design & Test Symposium (EWDTS)*, pages 18–21, September 2009.

[6] M. Caramia, M. Fabiano, A. Miele, R. Piazza, and P. Prinetto. Automated synthesis of edacs for flash memories with user-selectable correction capability. *Proceedings of High Level Design Validation and Test Workshop, 2010. HLDVT 2010. IEEE International*, pages 113 – 120, 10-12 June 2010 2010.

[7] Yuan Chen. Flash memory reliability nepp 2008 task final report. Technical report, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 2008.

[8] Sau-Kwo Chiu, Jen-Chieh Yeh, Chih-Tsun Huang, and Cheng-Wen Wu. Diagonal test and diagnostic schemes for flash memories. In *Proc. International Test Conference*, pages 37–46, 7–10 Oct. 2002.

[9] S. Di Carlo, M. Fabiano, R. Piazza, and P. Prinetto. Exploring modeling and testing of nand flash memories. *submitted to EWDTS 2010. IEEE International*, 2010. http://orion.polito.it/EWDTS10/ewdts10.pdf.

[10] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli. On-chip error correcting techniques for new-generation flash memories. *Proceedings of the IEEE*, 91(4):602 – 616, april 2003.

[11] Yea-Ling Horng, Jing-Reng Huang, and Tsin-Yuan Chang. A realistic fault model for flash memories. In *Proc. Ninth Asian Test Symposium (ATS 2000)*, pages 274–281, 4–6 Dec. 2000.

[12] Department IEEE Standards. Ieee 1005 standard definitions and characterization of floating gate semiconductor arrays. Technical report, 1999.

[13] D. Ielmini. Reliability issues and modeling of flash and post-flash memory (invited paper). *Microelectronic Engineering*, 86(7-9):1870 – 1875, 2009. INFOS 2009.

[14] Sung-Hoi Hur Jae-Duk Lee and Jung-Dal Choi. Effects of floating gate interference on nand flash memory cell operation. *IEEE Electron Device Letters*, 23 (5), 2002.

[15] A. Keshk. Flash memory testing for realistic fault modeling iceec2004. pages 503–506, 5–7 Sept. 2004.

[16] R. Micheloni, A. Marelli, and R. Ravasio. *Error Correction Codes for Non-Volatile Memories*. Springer Publishing Company, 2008.

[17] M. G. Mohammad and K. K. Saluja. Testing flash memories for tunnel oxide defects. In *Proc. 21st International Conference on VLSI Design VLSID 2008*, pages 157–162, 4–8 Jan. 2008.

[18] M. G. Mohammad and L. Terkawi. Fault collapsing for flash memory disturb faults. In *Proc. European Test Symposium*, pages 142–147, 22–25 May 2005.

[19] M. Gh. Mohammad, K. K. Saluja, and A. Yap. Testing flash memories. In *Proc. Thirteenth International Conference on VLSI Design*, pages 406–411, 3–7 Jan. 2000.

[20] M.G. Mohammad and K.K. Saluja. Flash memory disturbances: modeling and test. *VLSI Test Symposium, 19th IEEE Proceedings on. VTS 2001*, pages 218 –224, 2001.

[21] Mincheol Park, Keonsoo Kim, Jong-Ho Park, and Jeong-Hyuck Choi. Direct field effect of neighboring cell transistor on cell-to-cell interference of nand flash cell arrays. *Electron Device Letters, IEEE*, 30(2):174 –177, feb. 2009.

[22] J. C. Yeh, Kuo-Liang Cheng, Yung-Fa Chou, and Cheng-Wen Wu. Flash memory testing and built-in self-diagnosis with march-like test algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(6):1101–1113, June 2007.

[23] Jen-Chieh Yeh, Chi-Feng Wu, Kuo-Liang Cheng, Yung-Fa Chou, Chih-Tsun Huang, and Cheng-Wen Wu. Flash memory built-in self-test using march-like algorithms. In *Proc. First IEEE International Workshop on Electronic Design, Test and Applications*, pages 137–141, 29–31 Jan. 2002.