Design for Reliability Techniques in Embedded Flash Memories

Benoît Godard

UNIVERSITÉ MONTPELLIER II
SCIENCE ET TECHNIQUES DU LANGUEDOC

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ MONTPELLIER II

Discipline : Électronique, Optique et Systèmes.
Formation Doctorale : Systèmes Automatiques et Microélectroniques.
École Doctorale : Information, Structures et Systèmes.

Présentée et soutenue publiquement

par:

Benoît Godard

Ingénieur ENSPS

le 2 Juillet 2008

Techniques de Conception en Vue d'Améliorer la Fiabilité des Mémoires Flash Embarquées

(DESIGN FOR RELIABILITY TECHNIQUES IN EMBEDDED FLASH MEMORIES)

JURY

Patrick Girard, Directeur de recherche du CNRS, LIRMM	Président
Lionel Torres, Professeur, Université Montpellier II, LIRMM	Directeur
Gilles Sassatelli, Chercheur du CNRS, LIRMM	Co-Directeur
Régis Leveugle, Professeur, INP Grenoble, TIMA	Rapporteur
Jean-Michel Portal, Professeur, Univ. de Provence, L2MP	Rapporteur
Said Hamdioui, Assistant Professeur, Univ. de Delft, Pays Bas	Examinateur
Jean-Michel Daga, Docteur, Société Atmel Rousset	Examinateur

A MES PARENTS SANS QUI
RIEN NE SERAIT ARRIVÉ.
A CORALIE POUR SON SOUTIEN
DE TOUS LES INSTANTS.

Remerciements

Le travail présenté dans cette thèse a été réalisé dans le cadre d'un contrat CIFRE associant le Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier (LIRMM) et la société ATMEL.

Je voudrais en premier lieu exprimer mes remerciements à Monsieur Erich Palm, responsable du département “Libraries and Design Tools” pour m'avoir donné la possibilité d'effectuer cette thèse au sein de son département ainsi que de m'avoir fourni les moyens nécessaires à son accomplissement.

Je souhaite également remercier Monsieur Michel Robert, Directeur du LIRMM pour m'avoir donné l'opportunité de réaliser ma thèse au sein de son laboratoire.

Je tiens tout particulièrement à remercier mon directeur de thèse, Lionel Torres, Professeur à l'université Montpellier II, pour son encadrement professionnel, sa présence et son enthousiasme qui m’ont permis d’évoluer en thèse en toute sérénité.

J'adresse également tous mes remerciements à mon co-directeur de thèse, Gilles Sassatelli, dont les conseils et le bon sens furent essentiels dans l'accomplissement de ce travail.

Je remercie Jean-Michel Daga, Manager de l’équipe conception de mémoires non-volatiles de la société ATMEL. Son ouverture d'esprit et son pragmatisme furent précieux pour mener cette thèse à bien et je lui adresse ici de très sincères remerciements.

Je remercie également Régis Leveugle, Professeur à l'INP Grenbole, Jean-Michel Portal, Professeur à l'université de Provence de Marseille, Said Hamdioui, Assistant Professeur à l'université de Delft aux Pays Bas, et Patrick Girard, Directeur de recherche au CNRS à Montpellier, de m'avoir fait l'honneur de s'être intéressés à ces travaux en acceptant d'en être les rapporteurs, examinateur et président de jury. Qu'ils trouvent ici l'expression de ma plus sincère reconnaissance.

Enfin, je terminerais en remerciant tous mes collègues de la société Atmel et du LIRMM pour tous les bons moments que nous avons passés ensembles: Jimmy, Julien, Olive, Fab. S., Fab B., Vincent, Mary, Jeanine, Nico B, Salvo, Thom, Nico SJ.

Table of Contents

General Introduction

Semiconductor memories are usually considered to be the most strategic microelectronic components of digital logic system design, such as computers and microprocessor-based applications ranging from satellites to digital consumer electronics. The semiconductor memory business has burst at the beginning of 70's with the release of the first microprocessor developed by Intel. With more than 65% of growth between 1970 and 1980, it was the most dynamic market in all the semiconductor industry. At the beginning of 1980, improvements in manufacturing methods caused lower prices. More and more applications were designed with memories, and storage capacities increased by 70% each year during 80's. Since 2000, this growth has been maintained by the high demand in term of consumer electronic with hand-held devices, mobile phones, products based on memory cards and personal computers. Nowadays, memory parts have increased drastically and now achieve 25% of the microelectronic market.

Many factors must be considered to understand reasons of this growth and the presence of memories in electronic devices. For instance, it is well known that DRAM are used as stand-alone devices in personal computer. Similarly, high density Flash memories are used in devices requiring mass storage capacities. On the contrary, low densities of embedded Flash memories or ROM are used in microcontroller applications in order to access code as fast as possible. From these examples, we can see that the memory market is separated in two categories: the standalone market and the embedded market.

To illustrate the memory market growth, the stand alone Flash market represented close to 20 billion dollars in 2005. This market has grown faster than any other market in all the semi-conductor history due to the high demand of storage capability in mobile phones, MP3 players and cameras. The stand-alone flash market is mainly dominated by the race to cost per bit reduction, especially in the mass storage market driven by NAND flash memories. In this sector, integration densities must be improved as much as possible to remain competitive. Currently, Solid State Disk (SSD) is the new emerging application allowed by multi-gigabit NAND flash devices. Thanks to its advantageous performances, power consumption and non-mechanical structure, there is a strong probability that it will replace the traditional magnetic hard drive in a not-

so-distant future. As a consequence, taking shares into hard drive market will certainly contribute to further accelerate the flash market growth.

Concerning the embedded Flash memory market, parts are less important [1] and were about 500 millions dollars in 2005, this is due to lower volumes. However, this market remains relatively important for manufacturers supplying embedded memories in Systems-On-a-Chip designs. The growth of this market is intimately bounded to new trends in MCU market. In particular, the increasing part of Flash based solutions is a new noticeable characteristic of this business. This can be explained in part by a new trend in producing a wide range of products with high performance and low power characteristic, in smaller quantities, requiring faster time to market. Flash re-programmability provides a maximum of flexibility for code development. Shortened development time and faster introduction of new applications become possible. But, as SoC are gaining interest also in critical applications such as automotive, aeronautic or biomedical, an increasing demand from customer to develop methods and solutions to guarantee reliable products is growing.

SoCs with embedded memories are facing a technological issue. An increasing silicon area is dedicated to memories and storage elements. The Semiconductor Industry Association (SIA) confirms this trend forecasting that memory content will approach 94% within few years. As a result, memories will concentrate the major part of failures and will become the main detractor of the SoC reliability.

Considering the above mentioned context, the work of this thesis was focused on the memory reliability and, in particular, the reliability of Flash memories embedded in SoCs. The major advantage of such a technology is to be non-volatile by using the principle of the floating gate. It becomes possible to keep information even if power supply is removed. Flash memory cells are written and erased electronically by high electric field generation on a floating gate during a given period.

Usually, the reliability of Flash memories is improved by process optimizations or the use of better materials. Another solution consists in reducing electrical constraints imposed to memory cells. However, these techniques may not be sufficient to guarantee safety of a product. Indeed, as soon as a memory cell is failing, the full chip will be failing. That is why, memories should integrate fault tolerance techniques to avoid this problem.

In particular, Error Correcting Codes (ECC) are the most popular method to prevent memories from on-line random errors. Some parity bits are stored with information bits. Depending on the adopted ECC scheme, a certain number of errors can be detected and corrected. To enhance yield, designers choose row and/or column redundancy. During the test production phase, defective memory elements are disconnected and replaced with error-free redundancy elements. Throughout the years, these methods have been mixed and this research field has been gaining more and more interest. Architectures combining ECC and redundancy for yield enhancement, and/or reliability enhancement have been developed. However, as each types of memories have their own particularities, specific methods must be found for Flash memories at a minimal cost.

Therefore, the objective of this thesis was to find efficient fault tolerance techniques in order to improve the reliability of flash memories.

More particularly, this work aimed at finding solutions for a specific type of Flash memory cell called FLOTOX embedded in a NOR type Flash memory array. However, these solutions could be extended with some adjustment to other types of memory.

The manuscript of this thesis is organized as follow:

Chapter I is dedicated to the presentation of semiconductor memories. A brief description of volatile memories is given whereas more details on Flash and EEPROM are provided. Basics and architectures of such memories are exposed. At the end of the chapter, the importance of reliability in memories is pointed out.

Chapter II is focused on the study of reliability in Flash memories. After a short introduction on the concept of reliability and its mathematical definition, failure mechanisms in such a type of devices are exposed. To develop the fault tolerance technique described in Chapter III, a compact cell reliability model is required. For that purpose, we develop a model which allows to describe the cell reliability evolution depending on various parameters such as the time, the number of cycles and the temperature. This is the first important contribution of this work.

Chapter III presents a first error correction technique we named Error Correction by V_T analysis. The starting point of this original contribution is to consider Flash memories as analog devices and to build an improved ECC scheme at practically

no additional cost. In a second step, this scheme is integrated with an auto-reconfigurable redundancy structure. This concept is mathematically studied thanks to the reliability cell model derived from chapter II. Finally, at the end of this chapter, architecture and results are presented.

Chapter IV shows a second error correction technique based on a hierarchical error correcting code. The aim of this technique is to consider the operating granularity difference between read and program operations in Flash memories. A powerful scheme with extended correction capabilities at a minimal cost is then described. To prove the efficiency of this scheme, a mathematical modeling based on Homogeneous Markov Process is depicted. However due to the complexity of the scheme, the model of section 2 has not been employed for the benefit of a simplified reliability model. Next, this scheme is integrated with a specific reliability management scheme that will be described. At the end of the chapter, architecture, results and a discussion are provided.

This work has been carried out in collaboration with ATMEL (Rousset), a company specialized in ASIC, MCU and embedded Flash memory designs, under the framework of a CIFRE contract. This partnership has allowed an industrial validation of the results obtained in this thesis. It has also been validated by several publications in international conferences and patents specialized in the reliability and fault tolerance domains.

Chapter I - Semiconductor Memories: From Technology to Reliability

This chapter is dedicated to the presentation of the variety of semiconductor memories. A brief description of volatile memories (DRAM, SRAM) is given whereas non-volatile memories (EEPROM and Flash) are exposed in a more detailed way. This diversity comes from the willingness of the industry and the academia to find the perfect memory. A perfect memory is defined as a memory presenting only advantages. It would be non-volatile, low cost, requiring a reduced number of process steps, with a small size, compatible with a logic-CMOS process, highly reliable, low power and very fast. However, current memories only fit some of the above mentioned requirements, by consequent some effort must still be performed. At the end of this chapter, we highlight reasons for improving reliability in memory designs. As they are certainly the most strategic elements in system design, a special care on this aspect must be taken.

I.1. What is a System-on-Chip and an embedded memory ?

Stand-alone semiconductor developments such as Logic-CMOS, DRAM and FLASH are amongst the most competitive state-of-the-art technologies. However, going deeper and deeper into technological nodes makes their cost to increase exponentially and only few manufacturers can afford this price. For other companies, an exploration way is born from the System-on-a-Chip (SoC) concept. By definition, a SoC is a system requiring only few additional components to be functional. To achieve this requirement, microprocessor, memories and various peripherals should be embedded in the same die of silicon. The main objective for embedded technology development is to find the best solution to merge two or more processes into the same one. Large degrees of freedom are possible and no solution has became a standard. That is why, each founders have developed their own solution.

The main requirement for building a complete SoC is to make available large memories. The art of embedded memory process consists in the addition of mask steps to a standard logic process. In this field, a good process guarantees good memory densities and logic performances while keeping the number of additional steps low. Even if it complicates the technology development, embedded memories advantages are numerous [2]:

- Memory integration in a single chip reduces the product cost and size.
- On-chip memory interface provide small on-chip drivers. It contributes to limit the power consumption, heat, capacitive loads and wire length while high speed is possible.
- Pad limitation is no more a bottleneck. Word lengths can be extended to enable higher system performances.
- Memory size can be optimized and adjusted to fit applications' needs.
- Each semiconductor memory has its advantages and drawbacks. All have been subject to development for integration in embedded processes [3]. Research aims at new concepts to replace this variety by a unique embedded memory with only advantages and no drawbacks.

I.2. Semiconductor memories

I.2.i. Semiconductor memory structure

The high level modeling of a semiconductor memory is composed of few functional blocks shown in the Figure I.1:

- **Memory array** – contains memory cells. Its shape is square or rectangular to minimize area impact and makes the implementation easier. It can be organized hierarchically in few sub-matrices.
- **Addressing circuitry** – is composed of **address registers** storing the address coming from the memory inputs. These registers are connected to the **row decoder** and the **column decoder**. The row decoder selects a memory row. The column decoder selects columns that correspond to a word during write and read operations.
- **Logic control** – selects the operation to be done in the matrix. In the case of a read operation, the contents of selected memory cells is amplified by a **reading circuitry** before to be stored in data registers to be finally sent on the data bus. In the case of a write operation, data available on the data bus are stored in the **data registers** and written into the memory matrix thanks to the writing circuitry at the selected address.

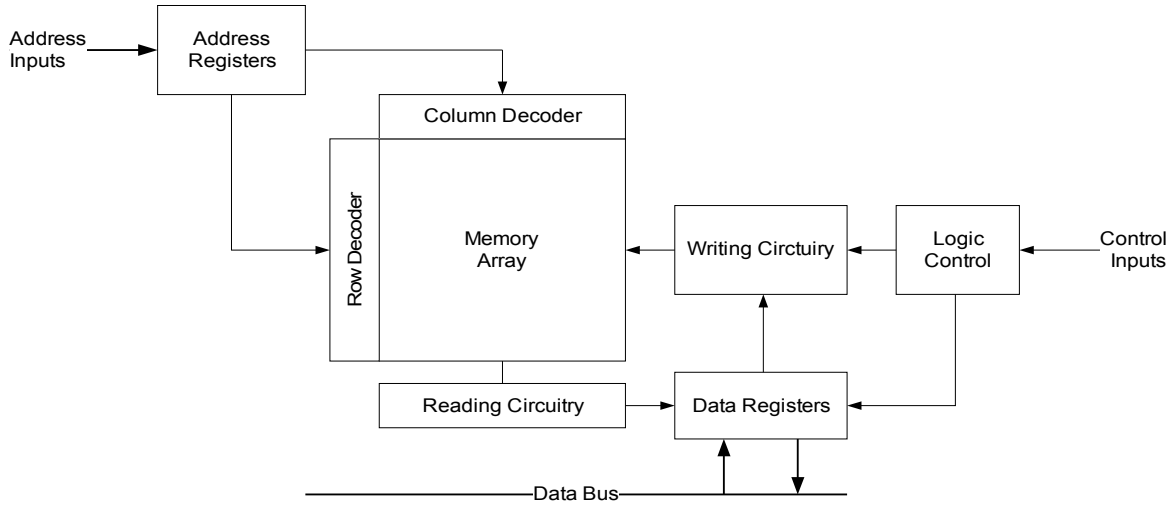

Figure I.1: Functional memory structure

Depending on the memory technology, some part of this basic structure must be adapted or removed. For instance, in EEPROM, writing circuitry is very complex. Charge pumps, regulators and timers are required to carry out write and erase operations. On the contrary, ROM does not have writing circuitry.

Semiconductor memories are usually classified in two main categories the volatile memories and the non-volatile memories.

I.2.ii. Volatile Memories

Main types of volatile memories are the Static Random Access Memories (SRAM) and Dynamic Random Access Memories (DRAM). The particularity of volatile memories is to lose information as soon as power supply is switched off.

I.2.ii.1. SRAM

The standard SRAM cell presented in Figure I.2 is made of two cross coupled inverters in addition with two access transistors. Such a structure retains data as long as a continuous power is provided. The memory cell size (125-160F^2 where F is the minimal length of the technology) is relatively important due to the 6-transistors composing its structure. In consequence, few nanoseconds operations are balanced with the memory cell size. The 6-T SRAM cell is highly compatible with the logic process. However, 4-T cells structure is possible and is employed to improve the density. Integration is balanced with the increase of process complexity due to additional polysilicon-layer resistors. SRAMs are very common and used in high speed buffer, registers, Translation Look-aside Buffer (TLB) and caches.

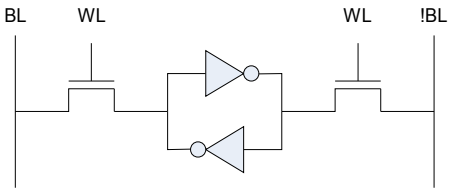

Figure I.2: 6-T SRAM memory cell

I.2.ii.2. DRAM

In DRAM, Figure I.3, information is stored as an electrical charge on a capacitor. It is called dynamic due to charge leaks even when continuous power is provided. In consequence, DRAM needs to be refreshed periodically to keep its content. The conventional cell structure is composed of one capacitor and one transistor. Its main advantage lies in its small size. The capacitor is made to have a high value on a minimum area.

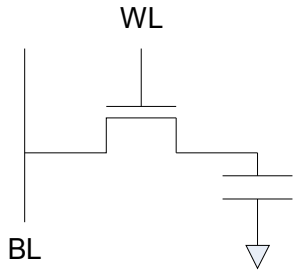

Figure I.3: 1-T DRAM memory cell

The main difficulty for embedding DRAM memories into a logic process is the following [3]: If the process is optimized for logic, high saturation current makes impossible the use of the conventional 1-T DRAM structure; on the contrary, if the process is optimized for memory with very low leakage current, the performance of logic transistors is low. To solve this issue, last technologies use dual gate processes with two gate oxide thicknesses. The DRAM part is made with the large oxide thickness whereas the logic part is made with small oxide thickness.

I.2.iii. Non-volatile memories

The main interest of non-volatile memories is to keep contents without continuous power supply. Several non volatile memories exist. The choice of such a memory is mainly constrained by three criteria which are the technology, economical constraints and applications needs.

I.2.iii.1. ROM

Historically, Read Only Memories (ROM) were the first class of non-volatile memories. They are mask programmable during the fabrication process. The memory effect is due to the existence or not of a CMOS transistor into the memory core. Thanks to its small size, large densities are available at low cost. However, mask programming is not flexible. A single code modification requires new masks and penalizes dramatically the time to market. Large volumes of identical memories are required to amortize the cost of these masks. ROM is highly compatible with logic process. However, its application is limited to storage of processors micro-code and well-debugged control code.

I.2.iii.2. Floating gate memories

As ROM memories lacks of flexibility, engineers were constrained to develop electrically programmable non-volatile memories. Thanks to its simplicity, the floating gate concept became mandatory. Further details on the floating gate concept will be given later in the section I.3.i. However, three technologies that have appeared successively need attention:

- **Electrically Programmable Read Only Memory** (EPROM) – The cell is written electrically but can be erased by UV rays. The writing operation can last up to 1 millisecond [4]. Hot electrons cross the oxide that separates the floating gate from the MOS transistor channel. The erase operation requires an exposition to UV rays by few tenths of minutes. Note that the EPROM core cell is only built with a double gate transistor as illustrated in Figure I.4. Thanks to this property, an EPROM has the same integration capability as a DRAM [5].

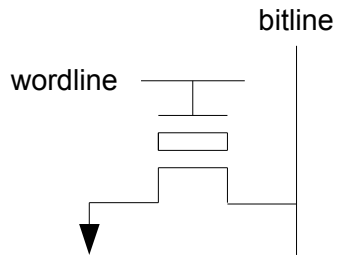

Figure I.4: EPROM cell

- **Electrically Erasable Programmable Read Only Memory** (EEPROM) – The main issue of EPROM was the UV erasing. EEPROM memory cell suppresses this constraint being programmable and erasable electrically. The cell is composed of two transistors in series. A first transistor is called control transistor and the second one is the floating gate transistor [6] as illustrated in Figure I.5. This structure has a low integration density due to the size of the 2-T cell memory device.

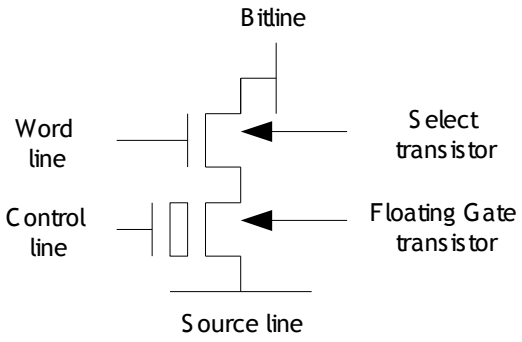

Figure I.5: EEPROM core cell

- **Flash EEPROM** – In fact, Flash memories are not linked to a technology but to a structure and more accurately to an operating granularity. Indeed, the name Flash comes from the particularity that the memory is erasable by a block or a page in one time. However, in almost all cases the Flash memory is built with a particular core cell technology based on a floating gate transistor. This core cell technology offers the advantage to have a reduced area and can be compared to the EPROM scalability. To write a logic content in this floating gate transistor, the HEI phenomenon is used whereas to erase this core cell, the tunneling effect called Fowler-Nordheim Tunneling [7] is preferred. This typical Flash core cell during the write and erase operations is shown in Figure I.6.

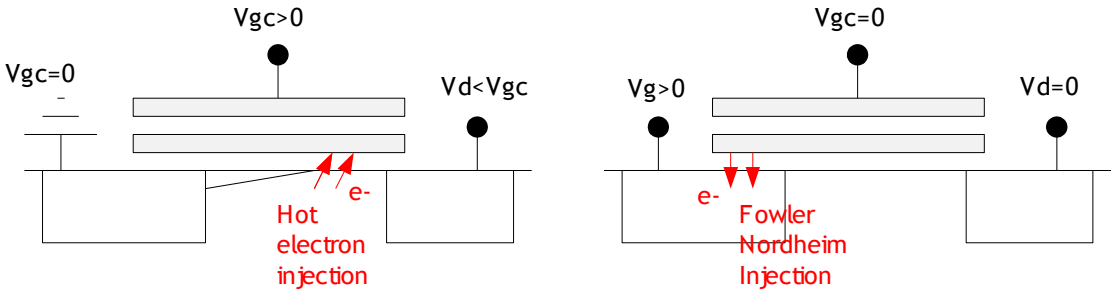

Figure I.6: Flash EEPROM core cell

I.2.iii.3. Future directions in floating gate cells

In order to further improve scalability of floating gate cells, it is admitted that new trapping concepts and new material should be introduced. Indeed, using a continuous floating gate as a single trapping node makes the cell very sensitive to oxide defects. Due to its continuous structure, one discharge path in the oxide is enough to discharge the floating gate, resulting in a failing bit. Because the amplitude of the stress induced leakage current through an oxide defect

increases when the tunnel oxide is thinned, oxide scaling is limited to about 80-90A. One very promising approach is to replace the floating gate by many discrete trapping nodes as illustrated in Figure I.7. This can be done by using an appropriate insulator such as nitride as in SONOS memories, or by semiconductor nanocrystals deposition. The basic idea is that in discrete-trap memories, a single leakage path due to a defect in the oxide can only discharge a single storage node. Using this concept, very good data retention and cycling characteristics have been obtained with tunnel oxides of 50A [8]. Thinner oxide means decreased voltage and better scalability. Anyway, a lot of work has still to be done to guarantee a limited spread in dot size and density, which is a mandatory condition to have this concept adopted in mass production [9]. In addition, the control gate to discrete traps coupling factor is an important parameter to optimize. Introducing high-K material such as HfO_2 into the control ONO stack is an efficient way to improve capacitive coupling [10], and decrease programming voltage. This approach could be combined with the introduction of nanocrystals as shown in Figure I.7, or with the current floating gate concept shown in Figure I.4.

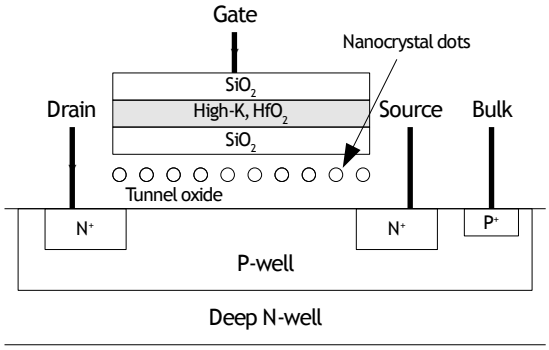

Figure I.7: Flash EEPROM core cell

The adoption of uniform channel programming using tunneling on a relaxed cell, together with the introduction of new trapping mechanism and high-K materials offers excellent perspectives to the embedded flash process to be integrated down to the 45nm node at least.

I.2.iv. Emerging memories

Research activities aims at finding a memory which would merge all characteristics to realize a perfect memory. Currently, two kinds of technologies seems to be the most promising in this field: the MRAM (Magnetoresistive Random Access Memory) and the PCRAM (Phase Change Random Access

Memory) or OUM (Ovonic Universal Memory). The brief review of these both memories is exposed in the following part.

I.2.iv.1. MRAM

MRAM principle is based on the modulation of resistivity of a ferromagnetic multilayer depending on the relative magnetization of its constituents. The physical phenomenon at the root of MRAM's invention is the magnetoresistance or TMR (Tunnel MagnetoResistance) which was put into evidence experimentally in 1975 by Julliere [11]. However, it is the discovery of the Giant MagnetoResistance (GMR) by Albert Fert and al. [12] which opened the ways to industrial applications. The first MRAM operational prototype was presented by IBM in 1999 [13] and shown a high potential with timings for program and read operations lower than 3ns.

MRAM is a nonvolatile thanks to the presence of a Magnetic Tunnel Junction (MTJ) in each cell. The simplest MTJ is composed of two ferromagnetic layers separated by a small insulating tunnel layer. The resistance of the MTJ depends on the relative orientation of the magnetic moments of the two ferromagnetic layers. If both magnetic moments are parallel, the resistance of the structure is low, however when they are anti-parallel, this resistance is high. Main advantages of MRAM are a unlimited number of write/erase cycles.

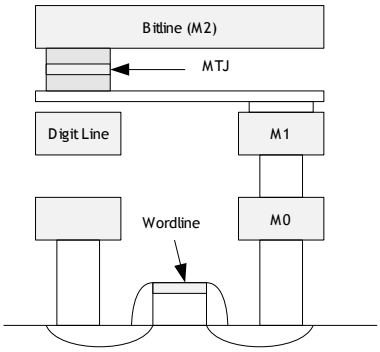

Figure I.8: Example of MRAM memory cell

I.2.iv.2. Phase change memories

Phase change memories are promising candidates as new generation of non-volatile memories. They are also name OUM (Ovonic Universal Memory). The material [14] they are constituted with is a chalcogenide glass which belongs to the same materials family used in rewritable optical compact disc such as CD-RW and DVD-RW. In optical disks, a low power laser beam is focused on a point of the disk to locally overheat the surface in order to switch the material phase from a

crystalline state to a amorphous state. The memory state is determined by the reflectivity of the material in this point which is different depending on the state of the material.

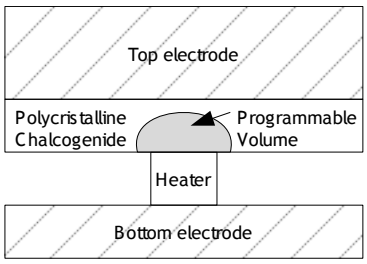

Figure I.9: Cross section of a PCRAM cell

In phase change memories, a current through a programmable volume results, by joule effects, in a phase change of this latter. Before the heating, the material is in a phase which resistivity is low. After the current crossing, the material is in an amorphous state, the resistivity is high. Time required to switch to amorphous state is typically lower than 30ns, however Samsung achieve a phase change speed close to 5ns. Some experiment have been carried out to determine experimentally the endurance of PCRAM which as been to 10^{12} without significant change of the resistance. Since 2001, research teams develop phase change memories prototypes [15] [16] and research is currently focused on the reduction of current and high density memory matrices.

I.3. Flash memory architecture

I.3.i. The floating gate concept

The floating gate concept is the common principle between an EPROM, an EEPROM and the Flash EEPROM cell. The Figure I.10 presents a cutaway view of a floating gate device [17]. Such a memory device is a transistor which threshold voltage can be adjusted according to the quantity of electrons present in the floating gate. The cell structure has a first gate of polysilicon named **Control Gate** (*CG*). A second polysilicon layer named **Floating Gate** (*FG*) is completely surrounded by a dielectric. The basic concept of a floating gate device can easily be understood determining relationships between the *FG* potential, that physically controls the channel conductivity, and the control gate circuitry which is controlled externally.

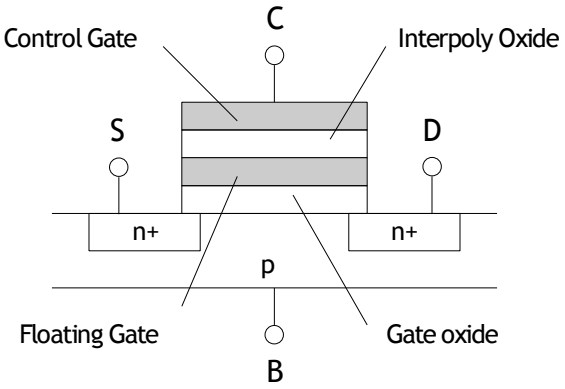

Figure I.10: The floating gate device

A simple capacity model is presented in the Figure I.11.The *FG* potential is expressed by:

$$V_{FG} = \frac{C_C}{C_T} \cdot V_C + \frac{C_S}{C_T} \cdot V_S + \frac{C_D}{C_T} \cdot V_D + \frac{C_B}{C_T} \cdot V_B + \frac{Q}{C_T} \quad \text{(I.1)}$$

Where, V_C, V_S, V_D, V_B are the control gate, source gate, drain gate and bulk gate potentials respectively. C_T denotes the total capacitance $C_B + C_S + C_D + C_B$ and Q is the charge of the *FG*.

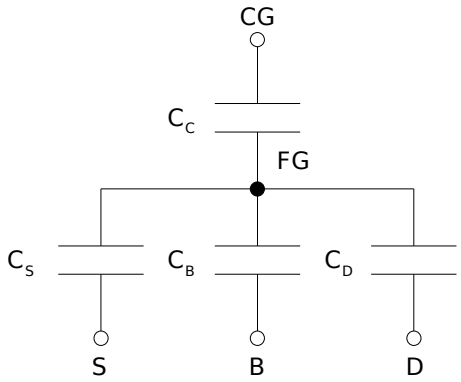

Figure I.11: Capacitive electrical model of a floating gate transistor

From equation (I.1), it can be noted that the *FG* voltage depends on the control gate voltage V_C and the charge of the floating gate Q but also depends on the biasing condition over the drain, the source and the bulk. If the bulk and the source are referenced to the ground, it comes:

$$V_{FS} = \frac{C_C}{C_T} V_{CS} + \frac{C_D}{C_T} V_{DS} + \frac{Q}{C_T} \quad \text{(I.2)}$$

And thus:

$$V_{FS} = \alpha_C (V_{CS} + f \cdot V_{DS} + \frac{Q}{C_C}) \quad \text{(I.3)}$$

Where, $\alpha_G = \frac{C_C}{C_T}$ denotes the coupling factor of the control gate and $f = \frac{C_D}{C_C}$ is the drain coupling factor.

In a floating gate transistor, the memory effect is due to $V_{T_{FS}}$ which is the threshold voltage that must be applied to the FG in order to reach the inversion of the surface population with $V_{DS} = 0$. Since the floating gate is not accessible, the derived voltage $V_{T_{CS}}$ is used. $V_{T_{CS}}$ is derived from equation (I.1). When the $V_{T_{CS}}$ voltage is applied to the control gate, $V_{T_{FS}}$ is applied to the floating gate:

$$V_{T_{CS}} = \frac{1}{\alpha_C} V_{T_{FS}} - \frac{Q}{C_C} \quad \text{(I.4)}$$

Contrarily to $V_{T_{FS}}$ that only depends on the technology, $V_{T_{CS}}$ depends on the number of charge inside the floating gate. This explains the non-volatile memory effect. Modulating the charge Q on the floating gate, it is possible to discriminate two states as shown in the Figure I.12:

- The erased state when $Q \ll 0$.
- The programmed state when $Q = 0$.

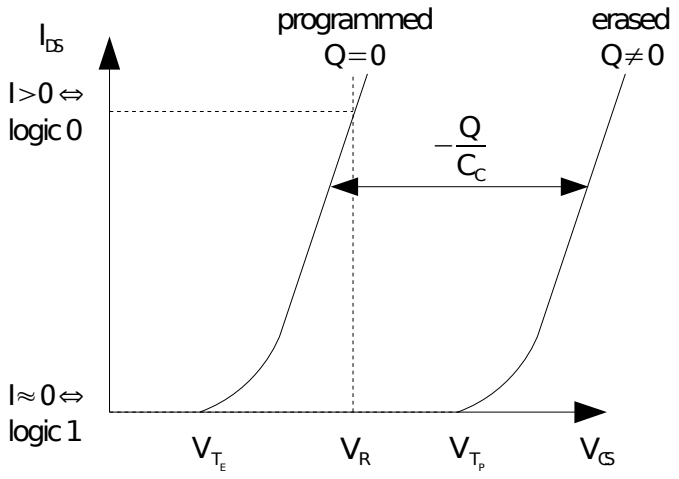

Figure I.12: I-V characteristic of a floating gate transistor for the erased and programmed states

Threshold voltages applied to the floating gate transistor are respectively:

$$V_{T_{CS}} = \frac{1}{\alpha_C} V_{T_{FS}} = V_{T_E} \quad \text{(I.5)}$$

$$V_{T_{CS}} = \frac{1}{\alpha_C} V_{T_{FS}} - \frac{Q_C}{C_C} = V_{T_P} \quad \text{(I.6)}$$

V_{T_E} and V_{T_P} are called erased threshold voltage and programmed threshold voltage.

Reading a floating gate device is made by a proper bias of the control gate such that a current flow could be detected or not through the device. By convention, if a current is detected, the device is ON which is traduced into a logic “0”. If no current is detected, the device is OFF which is traduced into a logic “1”.

Several solutions exist to transfer electric charges into or from the floating gate. The objective of these solutions is to pass though the thin oxide gate oxide (Figure I.10). Hot-electron injection or Fowler-Nordheim tunneling [7] are usually employed to perform write or erase operation in Flash memories.

I.3.ii. Memory cells description

I.1.i.a. ETOX cell structure

Figure I.13 shows a cutaway view of a standard Flash memory cell. This cell is named ETOX [18]. Originally, it has been derived from the EPROM memory cell with some significant modifications. For instance, the oxide between the bulk and the floating gate is very small. When an high voltage is applied on the source node whereas control gate is driven to the ground, an high electrical voltage appear in the oxide causing a tunneling effect from the floating gate to the source. Memory cell is erased. However, applied voltages significantly reduce source-bulk breakdown voltage. Consequently, source and drain are fabricated differently and the ETOX cell is not symmetrical, contrarily to the EPROM cell.

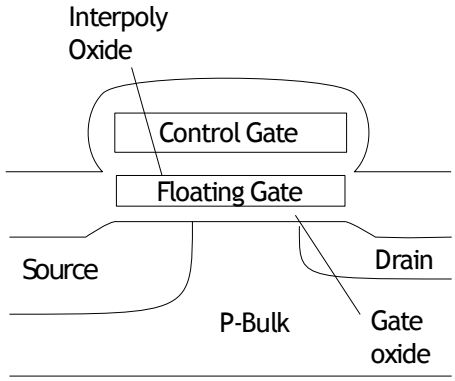

Figure I.13: ETOX memory cell

I.1.i.b. FLOTOX cell structure

The FLOTOX cell is composed of two MOS transistor in series: a Floating Gate transistor and a high voltage transistor named **Select Transistor**. The Floating Gate transistor contains information under the form of electrical charges. Charges are injected by Fowler-Nordheim Tunneling effects [19] trough a **Tunnel Window** which is a thin oxide layer located under the floating gate and the drain node.

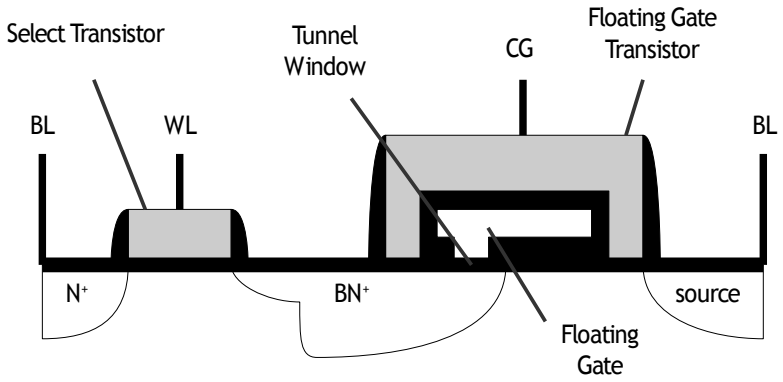

Figure I.14: FLOTOX memory cell

Memory cell programming is made thanks to Fowler-Nordheim Tunneling. It is a tunnel effect assisted by an electrical field. To happen, this phenomenon requires during a determined period of time a high voltage through the polysilicon-SiO_2-Silicon. This effect is possible thanks to the thin oxide layer (70-80 Å) of the Tunnel Window in the Floating Gate transistor.

To **write** the memory cell, that is to say to set a logic '0', a convenient high voltage is brought on the Word Line (WL) to set the Select Transistor ON. Next, the Control Gate is set to the ground and a high voltage V_M is applied on the drain of the floating gate device. V_M is around 12 to 20 V depending on the technology. In this state, electrons go from the floating gate to the drain. When the write operation is done, the floating gate transistor has a negative equivalent threshold voltage.

To **erase** the memory cell, that is to say to set a logic '1', the Control Gate is set to V_M while the drain and the source are connected to the ground. In this state, electron goes from the drain to the floating gate. The floating gate transistor has a negative equivalent threshold voltage.

To **read** the memory cell, a voltage V_R is applied on the Control Gate. V_R is comprised between threshold voltages of the written state (V_{T_w}) and erased (V_{T_e})

memory cells. It is a sense amplifier that will detect the current through the Bit Line (BL).

I.3.iii. Memory Architecture

The functional architecture of an embedded Flash memory is given in the Figure I.15. It is composed of a core-cell array, sense amplifiers, a word-line decoder, a bit-line decoder, a load decoder and a page buffer. Some timers and control logic serve to sequence program and erase operations. Two particular dedicated blocks are necessary to perform special functions. A charge pump device allows High Voltage Generation (HVG) to perform program/erase operations; Sense Amplifiers (SA) are used to perform read operations.

Two types of core-cells array are usually employed in the design of embedded Flash memories: NOR and NAND based arrays.

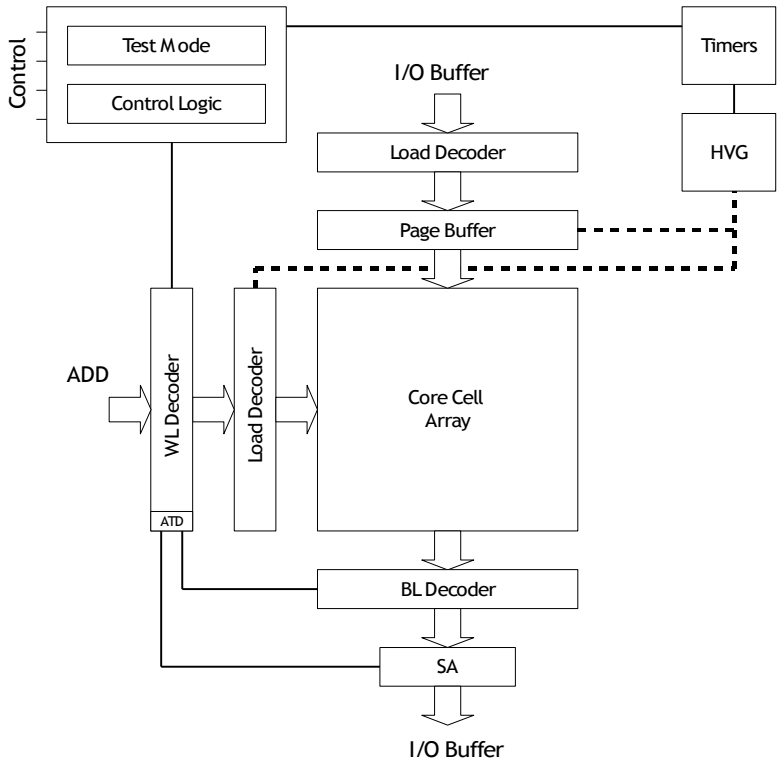

Figure I.15: Functional architecture of an embedded Flash memory

In NAND Flash arrays, Figure I.16.(a), the core-cells are placed in series. They form strips between two access transistors connected to a Bit Line (*BL*) and a Source Line (*SL*). A Word Line (*WL*) is shared by all the cells in a same row. During a write operation, all cells of one row are addressed together to form a page. Control Line (*CL*) and Ground Line (*GL*) allows control of the transistors.

Since the support of this work was based on NOR architectures, we will not give further details on NAND flash arrays.

In NOR flash array, Figure I.16.(b), the core-cells are placed in parallel. Each cell drain is connected to a Bit Line (*BL*), the gate of the floating gate transistor is connected to a Word Line (*WL*) and the source is connected to a global Source Line (*SL*).

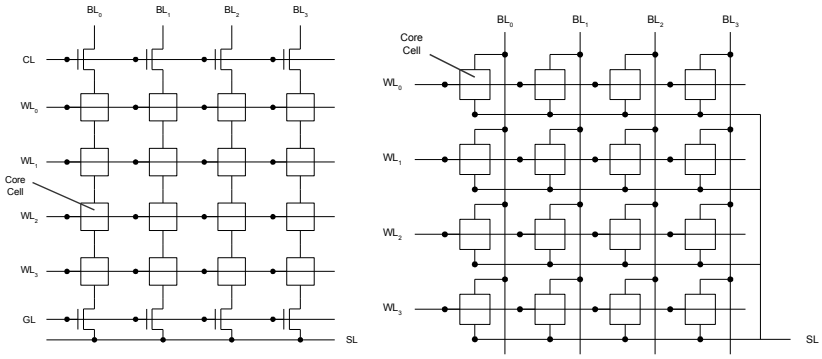

Figure I.16: Flash memory for (a) the NAND structure and (b) the NOR structure

Reading, loading and programming are the main operating modes accessible to an user. They must be well differentiated in term of speed and granularity:

- Reading is a fast operation (up to 40 MHz). To limit silicon area due to sense amplifiers, only a group of bits (32 to 128 bits) is read each time.
- Programming is made by page (64 to 256 bytes). It is a slow operation in the range of millisecond. Two steps are necessary to program a page: page erase, followed by the write operation. Erasing sets the overall page content to 'FF'. Then, some bits are written to '0' during write, according to the content to be programmed.
- Loading is a fast operation used to fill up Page Buffer with data before the programming operation. Multiple loading cycles are necessary because memory data input buffers are limited (usually 32 bits) compared to the page size.

I.3.iv. Embedded Flash memory technology comparison

There is no universal standard to implement memory cells. They are specifically tailored for the application, and can be divided into three main categories:

- The standalone NOR flash memories.

- The standalone NAND memories.
- The embedded flash memories.

I.1.i.c. The standalone market

The standalone flash memory market is driven by large memory capacities, very high volumes and standard interfaces. In that case, the cost per bit must be decreased as much as possible. As a matter of fact, the competition is mainly focused on memory cell scaling, while maintaining an acceptable level of performance and reliability. The 1-Transistor ETOX [17] cell is the dominant technology for stand-alone NOR memories. This is a good compromise to get an acceptable random read access time for code execution on large memories, at a reasonable cost per bit, especially when using the multi-level approach. The cost per bit can be lowered using the NAND cell approach [20], at the expense of an increased random read access time, making this type of memory more suitable for mass data storage, using a page access protocol. A summary of stand-alone NOR and NAND flash memory characteristics is provided in Table I.1.

Table I.1: Overview of standalone flash memories. *F* is the minimal length of the devices

	NOR Flash	**NAND Flash**
Purpose	Mainly code storage	Mass data storage
Cell density	10 F^2 for single level 5 F^2 for multilevel	4.5 F^2 for single level 2.3 F^2 for multilevel
Erase speed	0.1-1 s per sector	1 ms per page
Program speed	10 us per word	200 us per page
Read speed	Parallel access 20 MHz Random burst 66 MHz	Page serial access 10 us latency (first access) Up to 60 MB/sec, serial
Block density	Up to 512 Mb	Up to 4 Gb

From the user point of view, the NAND memories offer several advantages: low cost, erasing and programming speed, making them ideal memories for data management. NOR memories are more expensive, very slow to erase due to complex algorithms used to avoid over erase [17], but read operation is faster. For code execution, read access speed is mandatory. However, using a serial NAND memory and shadowing the code into an RAM memory for execution could be considered as a cost effective solution, mainly depending on code density and linearity [21].

I.1.i.d. embedded Flash technologies

A general purpose embedded flash technology must address several needs such as:

- High-speed in both erase and read operations.
- Low power supply and consumption.
- Very good reliability.

In terms of performance, such a technology has to combine the advantages of both NAND and NOR type architectures. In addition, the capability to be easily embedded into an advanced CMOS process at lowest cost possible must be considered. To achieve the performance needs, and considering the targeted memory capacities, the memory cell area can be relaxed compared to standalone technologies. For low to medium capacity, the memory block area is not only a question of memory cell density. It is the result of a trade-off between the memory cell area and the programming overhead area.

To illustrate the memory cell architecture impact on performances, and how performances could be improved by relaxing the memory cell area, both NOR and NAND stand-alone memory architectures can be considered.

NOR memories are characterized by poor erase speed performances which are a consequence of the 1T ETOX cell architecture. Time consuming erase algorithms must be implemented to avoid over-erased leaking cells [22] disturbing the read operation. Adding a select transistor serially connected to the floating gate device could make the resulting cell insensitive to the over-erase issue, as the minimum threshold voltage of the cell would be imposed by the select transistor. With such configuration, no more tight distribution of the erased cells would be needed, and erase delay could be reduced, at the expense of an increased cell area.

The very good erase performances of NAND memories can be explained by the cell architecture that includes two select transistors. However, in order to minimize as much as possible the equivalent NAND cell size, up to 64 floating gate devices are serially connected in between the select transistors [23]. This drastically reduces the memory cell read current, degrading the read performances. Minimizing the number of serially connected floating devices could be considered as a way to improve cell current and read speed, but once again, at the expense of an increased cell size.

The need for a high endurance, low current programming mode allowing massive parallel programming to speed-up the test, has mandated the Fowler-Nordheim tunneling mechanism to be used for both erase and program operations in most of the existing embedded flash solutions. Reducing the maximum high voltage value necessary to program the cell is desirable to simplify the integration of high voltage devices, and improve their density and performances. One solution is to move to a differential programming (using both positive and negative voltages) solution, on a triple well process. However, one limitation to this approach comes from the generation of disturbs that must be correctly evaluated.

In Table I.2, the main objectives to be reached by the e-flash technology and some possible technical solutions have been reported. The potential drawbacks associated with the proposed technical options are also given.

Table I.2: Technology objectives and possible technical options

Objective	Technical solution	Technical solution drawbacks
Low power, high endurance, parallel programming	FN tunneling for both erase and write	Single byte/word write delay
Avoid over-erase	Select device in series with FG device	Increased cell area
	State machine to improve control algorithm	Increased complexity and erase delay
Easy integration of high voltage device	Triple well, differential high voltage	Several pumps needed
		Potential disturbs

As explained before, standalone flash mainstream technologies are not suitable for an optimized e-flash solution.

The ETOX cell has been shrunk down to the 90nm for stand-alone NOR flash memories, but has several drawbacks limiting its use on e-flash applications especially for data management: over-erase increases erase delay, hot electron injection increases power consumption and limits endurance, disturb sensitivity limits programming flexibility. In addition, the circuitry overhead results in a poor block size density for low and medium memory capacity.

The NAND flash is low power and has good programming performances, but its poor read access performances do not fit code execution requirements. The 2T

FLOTOX cell has many of the required capabilities to build an ideal e-flash solution, and has been extensively used down to the 180nm node. Unfortunately, the FLOTOX memory cell cannot be satisfactory scaled below 1µm^2 due to the high voltage nature of its select device, and band-to-band tunneling issues resulting from drain programming.

Proprietary solutions have been developed in order to fit performance and reliability requirements of embedded flash at the minimum cost, with advanced technologies.

The 2T cell reported by [24] keeps most of the advantages of the FLOTOX cell, and can be scaled below 1µm^2 using advanced process rules. This is obtained by moving to channel program and erase on a triple well process and using tunneling. This protects the select device from high voltage biasing, and makes it scalable. Negative voltage is used to limit high positive voltage values in order to shrink HV-devices.

A 1T cell solution [25] allowing uniform channel programming, based on tunneling for both program and erase has been developed. Using channel erase and FN programming makes the cell scalable. However, in the absence of a select device, over-erase can result in leaky bits during read. One solution to avoid over-erase is to implement expensive and time consuming erase algorithms, in order to verify that all erased bits have a positive threshold voltage. Another solution presented in [25] is to use small sectors in combination with a special differential reading scheme to get rid of the leakage current coming from unselected cells in the selected transistor. This solution minimizes erase complexity, but increases read access time. A comparison of memory cell sizes at the 130nm node is provided in Table I.3.

Table I.3: Comparison of flash memory cell sizes (F: minimum length of the devices)

Cell Type	FLOTOX	e-Flash Proprietary	ETOX
Cell size 0.13 um node	~ 60 F^2	~ 25 F^2	~ 12 F^2

Even if there is no standard, e-flash proposed solutions share a lot of commonalities. Relaxed memory cell size to improve erase delay, FN tunneling for both erase and write in order to improve cycling capabilities and minimize

current consumption, are imposed by data management needs. This makes the e-flash a good candidate to replace not only the ROM, but the EEPROM as well.

I.4. Needs for reliability in embedded memories

I.4.i. Reliability engineering

Until the middle of the twentieth century, a good level of quality was assumed when an item could leave the manufacture without systematic defects or failures. However, as long as systems were made more complex, loss of operation and maintenance rapidly increased costs. To address this problem, a gaining interest has grown for the reliability engineering. Notions of reliability, availability, maintainability, safety have been developed.

Nowadays, a system must not only be failure free at the beginning of its operating time, but also during a defined time interval. However, this question cannot be answered by a simple yes or no depending on the result of a single test. Experience shows that only a probability of success can be given. To improve it, a specific engineering management scheme must be set during all the product life cycle. Decision scheme for reliability management at the very first step of the conception till the end of life of a product are mandatory. The field of reliability is gaining importance because of its effectiveness in the prevention, detection, correction of failure during design manufacturing and operating phase of products and systems.

I.4.ii. Reasons for growing failures in embedded memories

Technology scaling, demand for high capacity and application target are at the roots of high reliability needs. Indeed, the course for scaling and device integration poses severe problems of chip manufacturing, test and reliability. As technology is scaled down, defects that were negligible up to now must be taken into account. Devices are more sensitive to process drift, resulting in poor or null yield when process is not accurately tuned.

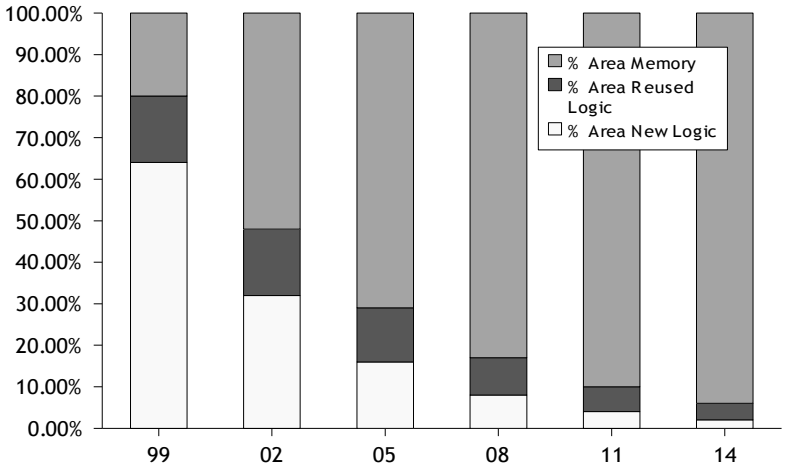

Figure I.17: Silicon area repartition in Systems-on-a-Chip

The high demand for embedded memories and storage elements constraints manufacturers to increase area dedicated to storage elements. As shown in Figure I.17, the Semiconductor Industry Association (SIA) [26] states that 94% of a SoC silicon area will be used by memories in 2014. The memory will be the main detractor of SoCs yield and reliability. For instance, Figure I.18 shows the evolution of the memory yield depending on the memory size in a 0.18µm process [27]. The yield of 20Mbits embedded memory product is 35% when the yield of 4Mbits was 80%. In particular, to put the yield back to an acceptable level, some solution must be integrated. For example, redundancy allows to reach theoretical yield above 95% for 20Mbits. Test repair and redundancy becomes a must in memory technologies.

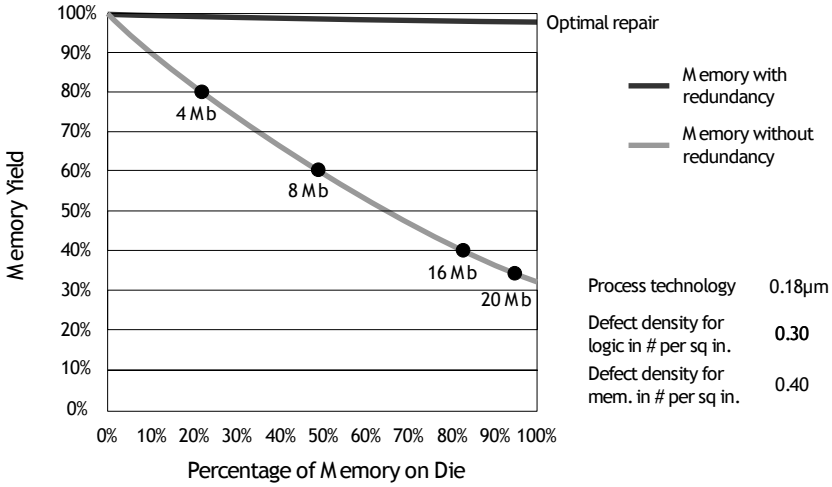

Figure I.18: Memory size versus yield

Multiples advantages of SoCs as all-in-one solutions has made it mandatory for various applications such as customer, industrial, automotive, aeronautic or biomedical products. However, in case of severe environments, devices are particularly stressed and 0ppm objective are difficult to reach if a special care on test and reliability has not been done.

I.4.iii. Impacts of the test on reliability

Test and reliability are complementary fields. The purpose of the test philosophy is summed in Figure I.19 [27]. Considering a set of manufactured chips, a perfect test should:

- detect all defects produced in a design and/or manufacturing.
- declare “pass” devices that are functionally good.
- declare “fail” devices that are functionally bad.
- detect all reliability related defects.

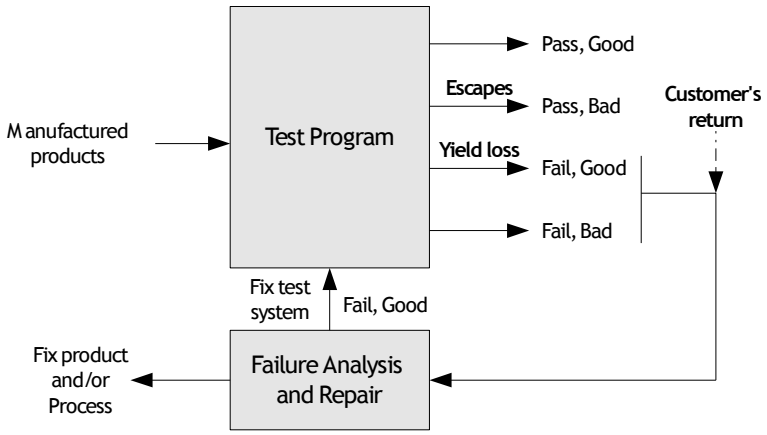

Figure I.19: Test philosophy

The purpose of test is to make $(Pass, Bad)$ and $(Fail, Good)$ sets of chips void, that is to say, without yield loss nor chip escapes. But, such a kind of exhaustive test does not exist because, in reality, it is impossible to test all defects. In general, specific tests are elaborated to cover only some realistic defects. This incomplete test coverage allows to optimize testing costs but may result in some yield loss and escapes.

Another practice of the test engineering is the memory repair already cited in section I.4.ii. This technique allows improving the yield by replacing defective elements of a chip with some failure free redundant elements. However, it has been shown by [27] that repaired memory chips are more sensitive to reliability

issues than non-repaired chips. This phenomenon is explained by the clustering effect: an area containing a lot of hard defect is more prone to latent defect. Latent defects and hard defects have the same origins but are different in terms of sizes and positions such that a latent defect will not be detected during production test contrarily to hard defects. With aging and stress, latent defect will become hard defects resulting in a SoC failure.

From a reliability point of view, escapes and latent defects are at the root of customer's return and reliability losses. They are very expensive and company killers in applications targeting 0 PPM (Part Per Million) returns.

I.4.iv. Memory design for reliability

To reduce the risk and improve reliability, specific methods must be developed to define, identify, and eliminate known or potential failures from the device before they reach the end user. The purpose of this work is to study and develop specific methods to improve embedded NOR Flash memories.

I.5. Conclusion

In this part, a review of volatile and non-volatile semiconductor memories was made. Thanks to their diversity, it is possible to address a large panel of applications going from server applications to portable devices. Table I.4 is a summary comparing embedded memories technologies and their performances. The perfect memory susceptible to be used in all electronic devices has not yet been discovered but is still the subject of a lot of research work.

Table I.4: Comparison between current embedded memories

	eSRAM	**eDRAM**	**eFlash**
Cell Size	120-140F^2	25-30F^2	15-30F^2
Extra Masks	0	+5	+10-12
Read performance	Up to 1GHz in 90nm	Up to 200MHz in 90nm	Up to 50MHz in 90nm
Limitations	Leakage issue requiring process options (V_T, T_{OX}) for portable applications	Large amounts of cache to justify process cost Large overhead	Low speed programming
Potential Issues	Leakage SNM Soft Errors	Leakage Stack capacitance value	Tunnel oxide L_{eff} scaling with HEI gate coupling
Solutions to push the limits	Design techniques Materials ECC	High K materials MIM cap ECC	3D structures High K materials Design techniques ECC
ITRS Prospects (2015)	Cell size: 0.15µm^2	Cell size: 0.0038µm^2 up to 10GBits	0,013um^2

This chapter was also the opportunity to introduce the support of our work which is related to Flash memories fabricated with FLOTOX core cell and integrated in NOR type arrays. The high flexibility, low power consumption, low access time and high density make these memories very suitable for portable devices. However, as their surface represents a growing importance in the system-on-chip and as critical applications may be nowadays addressed, the issue of the reliability of these memories must be considered.

Chapter II - Flash memory reliability

This chapter is dedicated to the study of the Flash memory reliability. In a first part, we provide definition on the concept of reliability and some mathematical prerequisites. Secondly, the problem of Flash memory reliability is discussed. Reliability for these devices is divided in two characteristics: endurance and retention. Mechanisms causing loss of these characteristics are reviewed. In the third part of this chapter, we are exposing the reliability cell model that has been employed in this work. As the main motivation of this thesis was to develop fault tolerant schemes for Flash memories, it was necessary to develop a model making possible reliability predictions. We have chosen to reuse and adapt the work of [28]*.We propose a compact reliability model for one cell depending on various parameters such as the time, the number of cycles and the temperature. Such a model has been useful in the validation of the fault tolerant scheme exposed in the Chapter III.*

II.1. Mathematical definition of the reliability

Reliability *- of a system is defined as the probability that the system will perform its required function under given conditions for a specified time interval.*

Failure *- happens when the item stops to perform its required function.*

The notion of reliability is based on four key points:

- **Probability** - There is always a probability of failure. Reliability engineering provides only probabilities of success at a specified statistical confidence level.
- **Required function** - It denotes the system task. When the system deviates from its required function, there is a failure. This notion is the starting point for reliability analysis.
- **Operating conditions** – They are related to the constraints applied on the system. It concerns its environment and condition of use.
- **Time interval** – It is the mission duration. Time t is often used as parameter. Reliability means that the system has a specified chance to operate without failure before time t. Other parameters may be relevant such as number of cycles.

The taxonomy to classify failures is well defined. They are categorized following their mode, cause, effect and mechanism:

- ***Mode*** is the symptom by which the failure is observed.
- ***Cause*** is the source of the failure. It can be extrinsic (bad operating condition, misuse) and intrinsic (wear out).
- ***Effect*** is the consequence of the failure. It can be non relevant, partial, complete, or critical.
- ***Mechanism*** is the process resulting in the failure.

Lifetime *(or* ***Time to Failure****) - is used to measure the quantity of service provided by a system. Usually, the time to failure of a system is measured by the number of hours it has effectively performed its required functions.*

The time to failure T is a random variable allowing evaluation of the system law degradation. The probability that the time to failure T to be comprised between t and $t+dt$ for a system is given by:

$$f(t)\cdot dt = Pr\,(t < T \le t + dt) \quad \text{(II.1)}$$

Where, $f(t)$ is **failure density probability**. The reliability function $R(t)$ and the unreliability function are given by:

$$R(t) = Pr\,(T > t) \quad \text{(II.2)}$$

$$F(t) = Pr\,(T \le t) \quad \text{(II.3)}$$

A system is usually characterized by its failure rate. The number of failure per units of time corresponds to the conditional probability that the system fails between t and $t + dt$ knowing it is reliable at time t:

$$\lambda(t) = Pr\,(t \le T < t + dt \,|\, T > t) \quad \text{(II.4)}$$

Common units for the failure rate are percent per thousand hours %/h and part per million per thousand hours PPM/K also called **Failure In Time** (FIT).

The four functions $f(t)$, $R(t)$, $F(t)$ and $\lambda(t)$ are linked together by formulas expressed in the Table II.1. .

Table II.1: Relation between *f(t)*, *F(t)*, *R(t)* and *λ(t)*

	$f(t)$	$F(t)$	$R(t)$	$\lambda(t)$
$f(t)$	-	$d\dfrac{F(t)}{dt}$	$-d\dfrac{R(t)}{dt}$	$\lambda(t)\cdot e^{-\int_0^t \lambda(x)\cdot dx}$
$F(t)$	$\int_0^t f(x)\cdot dx$	-	$1-R(t)$	$1 - e^{-\int_0^t \lambda(x)\cdot dx}$
$R(t)$	$\int_t^\infty f(x)\cdot dx$	$1-F(t)$	-	$e^{-\int_0^t \lambda(x)\cdot dx}$
$\lambda(t)$	$\dfrac{f(t)}{\int_t^\infty f(x)\cdot dx}$	$\dfrac{\frac{dF(t)}{dt}}{1-F(t)}$	$\dfrac{-1}{R(t)}\cdot\dfrac{dR(t)}{dt}$	-

For an electronic device, the failure rate $\lambda(t)$ evolution follows the bathtub curve presented in Figure II.1. This curve is composed of three regions. The first region corresponds to early failures or infant mortality. The failure rate decreases. The second region constitutes the useful life of the product. During this phase, $\lambda(t)$ is relatively constant and low. After the operating period $[t_0, t_1]$, the product enters in a third region called wear out. The failure rate increases sharply.

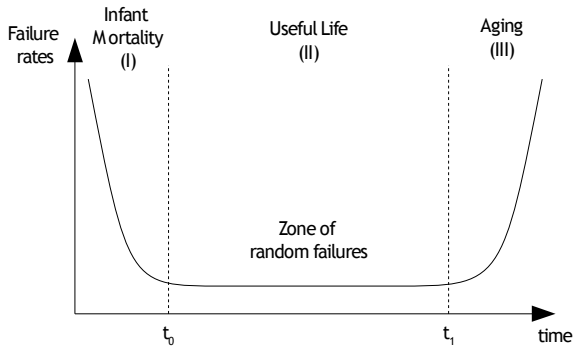

Figure II.1: General shape of the failure rate of a product during its life

Product victims of early failures are detected and rejected during the debug process. In addition, a « burn-in » phase may be employed to eliminate products that are not functional after a period t_0'.

A measurement of the operating time is given by the Mean-Time-To-Failure (MTTF). It corresponds to the mean of the *time-to failure T:*

$$MTTF = \int_0^\infty R(t) \cdot dt \tag{II.5}$$

The notion of reliability is the starting point for other concepts such as point availability, quality, safety and maintainability. All these concepts are the central notion of cost effectiveness [29].

***Availability** (or **Dependability**) - is expressed by the probability that the system will perform its required function under given conditions at a stated instant of time t.*

***Maintainability** - is the set of activities performed on a system to retain it in or to restore it to a specified state.*

In fact, maintenance is subdivided into:

- ***Preventive maintenance*** – is carried out at predetermined intervals and according to prescribed procedures, in particular to reduce wear out failures.
- ***Corrective maintenance*** – is carried out after fault recognition, and intended to put the system into a state in which it can again perform the required function. Corrective maintenance is also denoted ***repair***. It can include any or all of the following steps: localization, isolation, disassembly, exchange, reassembly, alignment, checkout.

***Quality** - is the totality of features and characteristics of a system that bears on its ability to satisfy stated or implied needs.*

Safety *- is the ability of a system not to cause injury to persons, or significant material damage or other unacceptable consequence during its use.*

Cost Effectiveness *- is the ability of a system to meet a service demand of stated quantitative characteristics, with the best possible usefulness to life-cycle cost ratio.*

Life-Cycle Cost *- is the sum of the costs for acquisition, operation, maintenance, and disposal of a system. In general for complex systems, higher reliability means higher acquisition cost and lower operating cost, so the optimal life cost and lower operating cost, seldom lies at extremely low or extremely high reliability figures.*

II.2. Floating gate reliability

The confidence in non-volatile memory reliability comes from the understanding of memory-cell failure mechanisms [30]. To characterize a floating gate memory array, the statistical distribution of cell characteristics on multiple chips has to be evaluated. Basically, the floating gate reliability mainly relies on two characteristics [31]:

Endurance *- is the capacity of a memory cell to keep good physical characteristic in order to be usable after multiple cycles of write/erase. Typically, a floating gate memory must be functional during at least* 10^5 *cycles.*

Retention *- is the capacity of the memory to retain information. Typically, a standard Flash memory cell must have 10 years retention.*

II.2.i. Retention

The threshold voltage V_T of a non-volatile memory depends on the number of charge stored in it. If the charge quantity varies, the threshold voltage will be modified. For instance, If there is a charge gain, the threshold voltage will increase. On the contrary, if there is a charge loss, the threshold voltage will decrease. Even when no voltage is applied on the floating gate, an electrical field is present through the tunnel oxide. The charge transfer occurs at any time. Quantitatively, let us consider that the floating gate capacitance is $C = 1\,fF$. A loss of $\Delta Q = 1\,fC$ causes $\Delta U = \Delta Q/C = 1V$ of threshold voltage variation. For 10 years retention, it supposes that less than 2 electrons can be lost per day [32]. Fault mechanisms leading to charge loss are classified in two categories:

- the extrinsic defects that are due to the device structure and manufacturing quality.

- the intrinsic defects are due to physical mechanisms used for programming operations.

The intrinsic mechanisms [33] are related to:

- **Field assisted electron emission** – that consists in the motion of electrons in an erased cell that can migrate to the interface with the oxide and from there, they can tunnel to the substrate. It results in a charge loss. In case of programmed cells, the opposite phenomenon can occur. Experiments have shown that these mechanisms are related to the coupling coefficient between the control gate and the floating gate, and also to the stress level.
- **Thermionic emission** – that corresponds to emission of carriers above the potential barrier. This phenomenon is negligible at low temperatures but becomes relevant at high temperatures.
- **Electron detrapping** – from the oxide, is a charge loss mechanism that reduces the program threshold voltage

The extrinsic mechanisms are related to:

- **Oxide defects** – that can cause charge loss or gain. The cause of defectiveness may be induced by process or fabrication defects but also by program/erase cycling. It may result in a phenomenon called “low temperature data retention” [34] and a poor data retention characteristic at moderate temperature. However, at high temperature, no data retention problem occur [35]. This results demonstrate that not all defects can be identified by temperature accelerated tests.
- **Ionic contamination** – that results in a shielding effect [33]. Ions, usually positive ones, are attracted by negative charges stored in the floating gate causing an effect similar to a charge loss. Memories can be affected by contaminations, which can penetrate through defects in passivation glasses or from chip edges. The quality of passivation layers have to be improved in order to reduce this effect.

II.2.ii. Endurance

Cycling is known to cause a fairly uniform wear-out of cell performances which eventually limits the memory endurance. The characteristic form of the wear-out is shown in the Figure II.2. Indeed, repeated program/erase cycles reduce

memory performances destroying memory oxides where charges are injected [36].

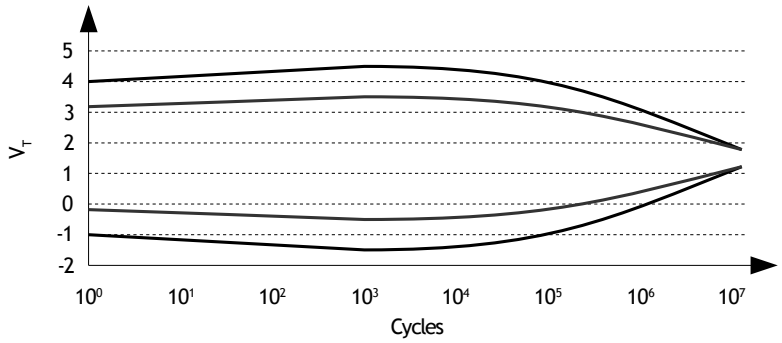

Figure II.2: Cell endurance cycling with fixed programming pulse with and distinct programming voltages

The current which crosses through the oxide creates interface states, traps and charge accumulations in the oxide. The main consequences of endurance stresses are:

- **Degradation of the transistor electrical characteristics** – this degradation is characterized by a loss of the mobility, the transconductance and the drain current. The decrease of transconductance and drain current may imply erroneous read.
- **Charge accumulation in the floating gate due to charge trapping in the oxide** – Charge trapped in the oxide results in shielding effects lowering the transparency of the tunnel barrier from a quantum point of view. Consequently, the Fowler Nordheim current decreases, resulting in a diminution of the stored charge in the floating gate and consequently a reduction of the programming window.
- **Oxide breakdown** – The programming method is an endurance driver. Indeed, the more injection zone is localized, the more created traps will be localized and dense. It is the case for memory using hot electron programming. It is also the case in memory that uses Fowler-Nordheim injection on recovering zone gate/drain or gate/source.

Experiments to test endurance are performed applying constant program/erase pulses [37]. The variations of program and erase threshold levels give a measure of oxide aging. In real devices, this corresponds to long program/erase times. The quality of tunnel oxide is again crucial for reliability assessment.

II.2.iii. Test and characterization of the reliability

In order to ensure high reliability and yield, the process monitoring of floating gate memories comprises three phases:

- In-line monitoring of the fabrication process.
- Parametric tests on dedicated structures at wafer level after the fabrication process.
- Reliability tests on test structures and circuits.

To evaluate quality and reliability of memories, non-destructive and destructive tests are performed through qualification tests:

- **Charge to breakdown measurement** on the oxide.
- **Endurance tests** to determine the distribution of the program/erase threshold voltages for a large number of cells and variations for a large number of cells.
- **Reliability test** of the high voltage transistors required for program/erase.
- **Data retention tests** to verify that information will be kept during the product lifetime. To verify this characteristic, bakes at high temperature are carried out. Results are extrapolated thanks to predictive models of temperature retention.

The programming quality and reading quality are performed during post production tests by a margin check. This check is made by modulating voltage V_R applied on the control gate of the floating gate transistor:

- To control the programming and the reading of a “1”, after a classical write, a reading is made applying a margin voltage V_R on the control gate of the memory core cell. The V_R is shifted up until a “0” is read instead of a “1”. The high margin V_H is an image of the equivalent threshold voltage of the memory cell.
- To obtain the low margin, synonym of a good programming and reading of a “0”, the memory cell is written and the V_R is shift down until a 1 is read instead of a 0. The more the **programming window**, difference between V_H high margin and V_L low margin, is high, the best programmings and readings are. This is a way to qualify the memory.

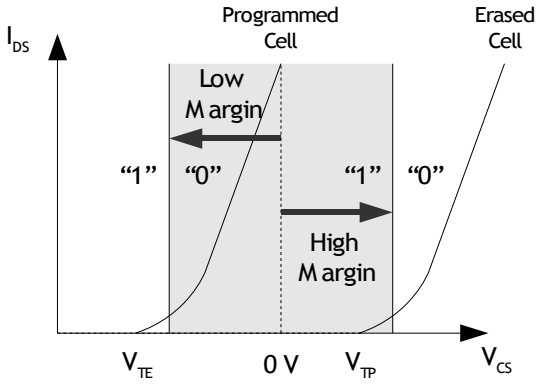

Figure II.3: Margin check by control gate voltage modulation

II.3. Cell reliability modeling

In our step toward the elaboration and evaluation of design techniques improving memory reliability, it appeared necessary to develop a compact reliability model. This part is focused on this contribution.

II.3.i. Objective of the modeling

Impact of various parameters can be modeled through a compact modeling of the floating gate reliability. Such a kind of model is important to make reliability predictions. It provides the probability that the threshold voltage of a cell V_T is lower than a voltage limit V_{limit} depending on factors such as time, number of cycles, temperatures, programming conditions.

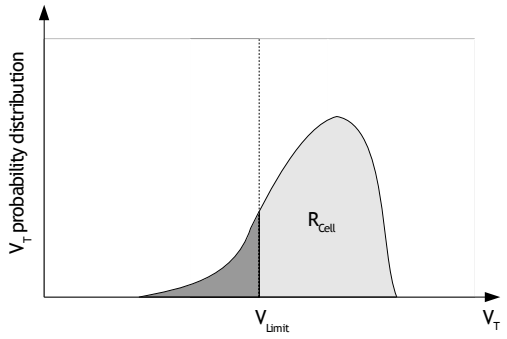

Figure II.4: Example of probability distribution of a cell threshold voltage

After a short explanation of the problem of the cell leakage characterization, the latest research trends in this field will be exposed. This part will be followed by a description of the compact reliability modeling that has been established.

II.3.ii. Problem of the cell leakage and its characterization

The threshold voltage value of the cell is directly related to the quantity of electrons Q_{FG} stored in the floating gate referred as by the relationship:

$$V_T = V_{T_0} + \frac{Q_{FG}}{C_{TOT}} \quad \text{(II.6)}$$

where, C, V_{T_0} are the equivalent capacitance of the floating gate and the virgin threshold voltage, respectively. Q_{FG}/C_{TOT} is also the floating gate potential V_{FG}. By convention, erased and written cells correspond to the logic value “1” and the logic value “0”. From a functional point of view, during the memory life, V_T are distributed over two populations as shown in Figure II.5, one for cells with high V_T (logic value “1”), and another for cells with low V_T (logic value “0”).

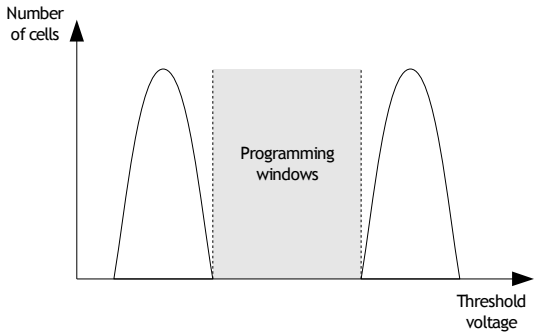

Figure II.5: Cell distribution

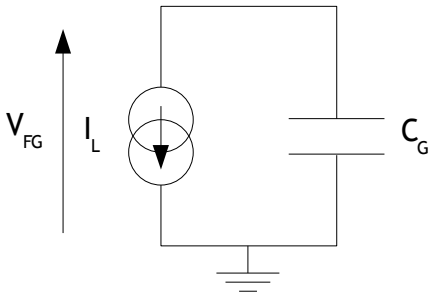

Figure II.6: Equivalent model of a defective floating gate cell

Because of charge leakage mechanisms distributions drift. With time, logic “1” and “0” tends to opposite values and becomes weak or erroneous. In the rest of this part, only one distribution that goes from high V_T values to low V_T values is considered. The charge leakage mechanism can be modeled by a capacitor being discharged by a current source through a thin oxide as shown in the Figure II.6.

The fundamental expression linking threshold voltage with leakage current is derived from (II.6) and expressed by:

$$\frac{dV_T}{dt} = \frac{-I_L}{C} \tag{II.7}$$

where, I_L is the charge leakage current. A general expression for I_L has been found [34] to cover various conduction mechanisms and to solve the charge conservation equation (II.7):

$$I_L = A \cdot E^{X+1} \cdot \exp\left(-sign(X) \cdot \left(\frac{B}{E}\right)^X\right) \tag{II.8}$$

where, A, B and X are parameters related to the conduction mechanism. Depending on the value of X, various conduction mechanisms can be described. For instance, X=1, -0.5, -1 correspond to Fowler-Nordheim, Poole-Frenkel and Trap-assisted-Tunneling mechanisms respectively.

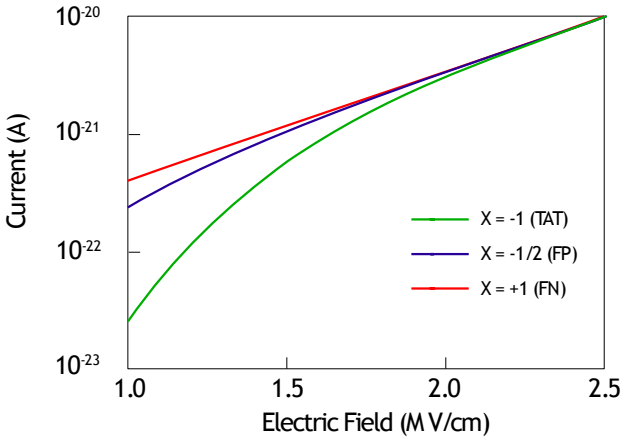

Figure II.7: Conduction model evaluated for multiple values of X

However, when low electrical fields are applied through an oxide, the accurate modeling of I_L is still a big issue. Currents are extremely low and difficult to measure experimentally. As illustrated in the Figure II.7 [34], current may vary from many orders of magnitude under low electrical fields depending on the conduction mechanism. In consequence, the conduction mechanism is usually chosen empirically based on experimental observation and not on an accurate knowledge of the physical mechanism.

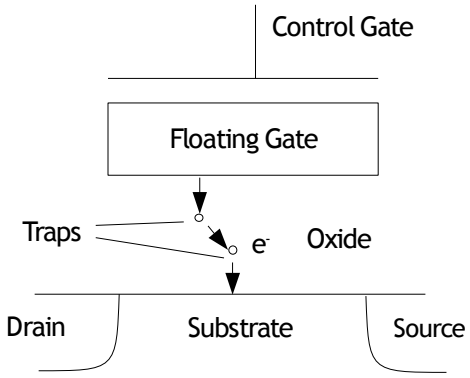

Figure II.8: Defective floating gate cell

II.3.iii. Stress Induced Leakage Current

Latest researches [38-41] have shown that the main retention issues are related to the *Stress-Induced Leakage Current* (SILC). SILC is the excess low field current across a thin gate oxide after a high field stress. SILC is the main detractor for floating gate reliability. Due to high field stresses, traps may appear in the insulating layer and create conduction paths. With oxide thickness scaling, SILC phenomenon becomes predominant. At the root of SILC, there would be a *Trap-Assisted Tunneling* (TAT) effect. The underlying physical phenomenon has been efficiently explained thanks to the percolation model [39,41,42].

Consider an oxide layer separating an anode and a cathode as shown in the Figure II.9 that has been exposed to a high electric field stress. Consequently, traps are present in the oxide with a given density D_{OT}. Assuming a path from the anode interface to the cathode interface passing by traps, the percolation distance x_{perc} is defined as the maximum distance between two elements of this path (trap-trap or trap-interface).

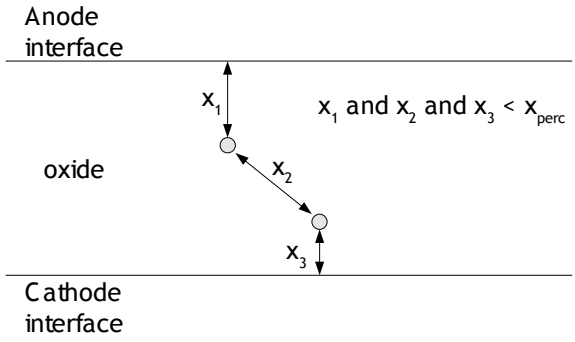

Figure II.9: two trap percolation path. The conduction is determined by the largest of the three distance x1, x2, x3

To describe the conduction mechanism in this path, the direct tunneling model is

used. Direct tunneling is a probabilistic quantum effect depending on the width of a potential barrier. If this barrier is thin enough, the probability for an electron to tunnel from one side to the other is very important, a current path exists.

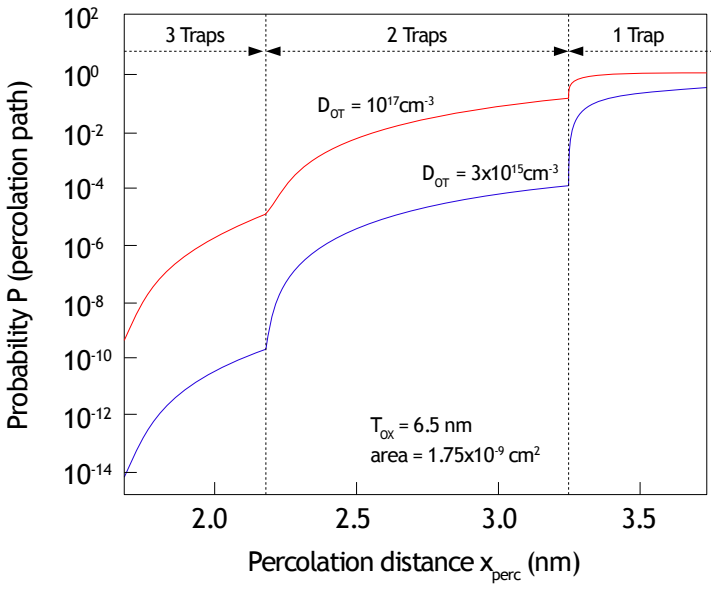

Figure II.10: Calculated failure fraction as a function of oxide thickness for distinct oxide qualities

The probability to find traps properly aligned between the anode and the cathode such that electrons can tunnel and form a current path (percolation path) in the oxide has been modeled. This major result is presented in the Figure II.10 [43]. For instance, if x_{perc} is above $t_{ox}/2$, a 1 trap path is involved. The probability for this path to be a percolation path is very important. The SILC phenomenon will be really important. When x_{perc} decreases, it models multiple trap percolation path, however, while the proper alignment is seldom the probability of percolation path decreases rapidly.

II.3.iv. Compact reliability modeling

For the purpose of this work, a high level model has been developed starting from the work of [28]. The V_T variation between cells is explained by parameters modulation that acts as V_T-shift on the cell threshold distribution. Thanks to experimental observations, assumptions on the V_T evolution have been made:

i. V_T drift is independent of the initial V_T value.

ii. V_T drift is linear with the logarithm of the time.

iii. V_T drift is linear with the logarithm of the number of write/erase cycles.

iv. V_T drift is linear with the inverse of the temperature.

I.1.i.e. Assumption 1 – V_T drift is independent of the initial V_T value.

In the general conduction model provided in the equation (II.8), if $X=-1$, the leakage current I_L can be written:

$$I_L = I_0 \cdot \exp\left(b_0 \cdot \alpha_G \cdot \left(V_T - V_{TNA}\right)\right) \quad \text{(II.9)}$$

Where V_{TNA} is the neutral threshold voltage of the cell, α_G is the gate coupling ratio, I_0 and b_0 are technological constants. Even if b_0 and I_0 varies slightly, all cells follow the same V_T-time curve shape. In the case where $X=-1$, the integration of equations (II.8) and (II.9) is very simple, all cells with an exponential leakage I-V have the following V_T-vs-time characteristic:

$$\frac{-1}{b}\ln\left(e^{\left(-b\cdot V_T\right)} - e^{\left(-b\cdot V_{T_0}\right)}\right) = \frac{-1}{b}\ln\left(\frac{t\cdot I_0\cdot b}{C_{TOT}\cdot\alpha_G\cdot e^{b\cdot V_{TNA}}}\right) \quad \text{(II.10)}$$

Where V_T is the threshold voltage of the cell at time t, V_{T_0} is the V_T at the initial state and $b = b_0 \cdot \alpha_G$. The left-side quantity is defined as $V_{T_{ADJ}}$:

$$V_{T_{ADJ}} = \frac{-1}{b}\ln\left(e^{\left(-b\cdot V_T\right)} - e^{\left(-b\cdot V_{T_0}\right)}\right) \quad \text{(II.11)}$$

In fact, the exponential $e^{-b\cdot V_{T_0}}$ becomes rapidly negligible in respect with the other term after few tenths of a volt. It means that V_T values becomes practically independent from the starting V_T point. This assumption is illustrated in the Figure II.11.

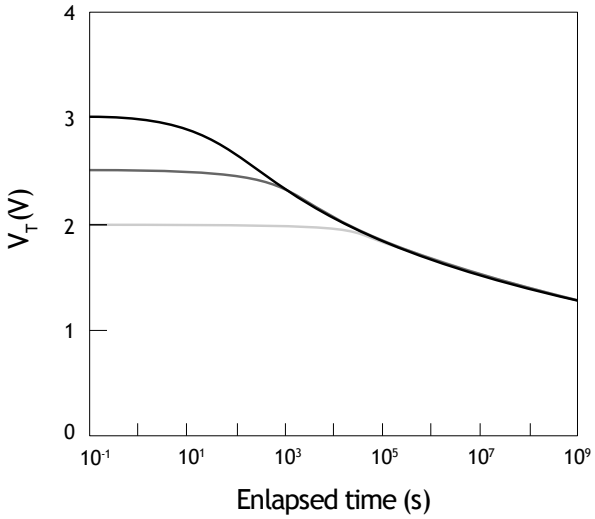

Figure II.11: V_T evolution in respect with the time depending on the initial threshold voltage

I.1.i.f. Assumption 2 – V_T drift is linear with the logarithm of the time

From this point, the equation (II.10) can be expressed by:

$$V_{T_{ADJ}} = \frac{-1}{b}\ln(t) - \frac{1}{b}\ln(I_0) - \frac{1}{b}\ln\left(\frac{b}{C_{TOT}\cdot\alpha_G}\right) + V_{TNA} \qquad \text{(II.12)}$$

With the previous model for the shape of the universal curve, an acceleration in time is equivalent to an arithmetic offset in $\ln(t)$. This time acceleration factor corresponds to a simple I_0 term variation.

II.3.iv.1. Assumption 3 – V_T drift is a linear function of the logarithm cycle count

A logarithmic dependence for $V_{T_{ADJ}}$ drift versus the cycle count n_{cycle} is assumed to model the cycle count. This dependence is added in the equation (II.12). The time t taken by a cell-V_T to cross some failure level will scale in a power-law depending on the number of cycles.

II.3.iv.2. Assumption 4 – drift is linear with the inverse of the temperature

To integrate the $V_{T_{ADJ}}$ evolution depending on the temperature, an Arrhenius law evolution is assumed. The Arrhenius law is also known in the literature as the $\frac{1}{T}$ model. In that case, V_T drift is linear with the inverse of the temperature for a

specific cycle count. A term in $\frac{1}{k_B \cdot T_{BAKE}}$ is added to (II.12). Data are extrapolated from experimentation at high temperature using Arrhenius law given by:

$$t_R(T) = t_0 \cdot \exp\left(\frac{E_a}{k_B} \cdot T\right) \quad \text{(II.13)}$$

T_R Is the retention time at temperature T (for a threshold limit defined previously), k_B is the Boltzmann constant, E_a is the activation energy and T the temperature. This relation can be re-written:

$$\ln\left(t_R\right) = \ln\left(t_0\right) + \frac{E_a}{k_B \cdot T} \quad \text{(II.14)}$$

To determine the curve equation representing the logarithm of t_R in function of $\frac{1}{T}$, only two points are needed. E_a is deduced from the slope of this curve.

Even if for the purpose of this work, the $\frac{1}{T}$-law is used, some limitation exist. This model is not fully satisfying because depending on the explored temperature range, the activation energy is different. In reality, t_R logarithm is not an affine function of $\frac{1}{T}$ [44]. To solve this problem, a unified model as been proposed thanks to an accurate analysis of the dependency of the temperature with the Fowler Nordheim current. This law let to interpret all the results:

$$t_R(T) = t_0 \cdot \exp\left(\frac{-T}{T_0}\right) \quad \text{(II.15)}$$

T_0 is a characteristic temperature of data retention. This expression is:

$$\ln\left(t_R(T)\right) = \ln\left(t_0\right) - \frac{T}{T_0} \quad \text{(II.16)}$$

The 1/T model overestimates the data retention as shown in the Figure II.12.

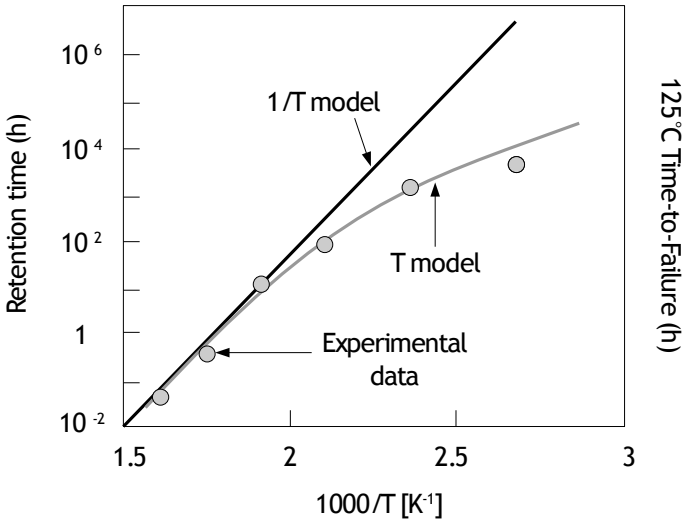

Figure II.12: Time extrapolation with T and 1/T laws

I.1.i.g. Statistical variations

The fraction of cells below any given $V_{T_{ADJ}}$ increases exponentially with $V_{T_{ADJ}}$, so the fraction of cells below a given $V_{T_{ADJ}}$ can be described by an exponential law depending on two parameters d_0 and d_1:

$$D\left(V_{T_{ADJ}}\right) = d_0\, e^{d_1 \cdot V_{T_{ADJ}}} \tag{II.17}$$

In optimized technologies, leaky cells are randomly distributed over the chip. Thus, a Poisson statistic of random defect density D can be used to describe this statistic in a block composed of N cells. The probability that the block's minimum V_T is below $V_{T_{ADJ}}$ is one minus the probability that all the cells of the block are above or equal to this value. For a block of N cells, it gives:

$$P_N = 1 - e^{-N \cdot D} = 1 - e^{-N \cdot d_0 e^{-d_1 \cdot V_{T_{ADJ}}}} \tag{II.18}$$

The result of this equation is called an extreme value distribution in the block-minimum V_T. To comprise an extreme value distribution variation in $V_{T_{ADJ}}$, a term of the form $d_1 \cdot \ln\left(-N \cdot \ln\left(1 - P_N\right)\right)$ is added in the equation (II.12) to model this offset.

II.3.iv.3. Recapitulation

As said previously, the model determines the probability that cell threshold is higher than a given voltage limit V_{Limit} depending on the time t, the number of cycles n_{cycles} and T the temperature:

$$R_{Cell} = P\left(V_T > V_{Limit}\right) = f\left(t, n_{cycles}, T, V_{Limit}\right) \quad \text{(II.19)}$$

All previous assumptions result in the following analytical model:

$$\ln\left(-\ln\left(1-P_N\right)\right) = \frac{1}{d_1 \cdot b}\cdot\ln\left(I_0\right) + \frac{1}{d_1 \cdot b}\cdot\ln\left(\frac{b}{C_{TOT}\cdot\alpha_G}\right) + \ln\left(d_0\right) + \ln\left(N\right) + \frac{\ln\left(t\right)}{d_1 \cdot b} - \frac{\alpha}{d_1 \cdot k_B \cdot T} - \frac{\beta\cdot\ln\left(n_{cycles}\right)}{d_1} + \frac{\left(V_{T_{ox}} - V_{TNA}\right)}{d_1} \quad \text{(II.20)}$$

The expression has been simplified by renaming constant factors in (II.20). The new expression depends on c_0, c_1, c_2, c_3, c_4 and variables such as the threshold voltage limit V_{Limit}, the time t, the number of cycles n_{cycles} and the temperature T:

$$\ln\left(-\ln\left(1-R_{Cell}\right)\right) = c_0 + c_1\cdot V_{Limit} + c_2\cdot\ln\left(t\right) + c_3\ln\left(n_{cycles}\right) + c_4\cdot\frac{1}{T} \quad \text{(II.21)}$$

where, constant c_0, c_1, c_2, c_3, c_4 are determined experimentally , with

$$\begin{aligned} c_0 &= \frac{1}{d_1\cdot b}\cdot\ln\left(I_0\right) + \frac{1}{d_1\cdot b}\ln\left(\frac{b}{C_{TOT}\cdot\alpha_G}\right) + \ln\left(d_0\right) \\ c_1 &= \frac{1}{d_1} \\ c_2 &= \frac{1}{d_1\cdot b} \\ c_3 &= \frac{-\alpha}{d_1} \\ c_4 &= \frac{-\beta}{d_1} \end{aligned} \quad \text{(II.22)}$$

II.3.iv.4. Erratic bits behavior

In Flash memories some bits may have an erratic behavior [45-47], it may be interesting to integrate them into the model in a specific way. This is our contribution to the model.

The erratic bit behavior is a floating gate specific issue that usually affects a ratio of cells randomly distributed in an array. For these bits, the threshold voltage may evolve by steps throughout time. These bits are at the root of reliability losses because they cannot be detected during the test production phase: even if a memory has passed tests successfully, it may be defective due to these bits. The underlying phenomenon is not accurately known but some explanations have been proposed. For instance, a convenient explanation would be that bi-stable traps in the oxide would create a kind of TAT effect. For a ratio of the memory life α_{ON}, these traps would be in an ON state, adding an additional current leakage I_{ON}

in the model. The rest of the time, traps would be in an OFF state. This effect is taken into account adding the constant term c_0' to (4):

$$\ln\left(-\ln\left(1-R_{Erratic}\right)\right)=c_0+c'_0+c_1\cdot Limit+c_2\cdot\ln\left(t\right)+c_3\ln\left(n_{cycles}\right)+c_4\cdot\frac{1}{T} \quad \text{(II.23)}$$

Erratic cells and normal cells are part of two independent distributions. The combination of (II.21) and (II.23) gives the reliability for one cell:

$$R_{Cell}=\alpha\, R_{Erratic}+\left(1-\alpha\right)R_{Normal} \quad \text{(II.24)}$$

where, α represents the ratio of erratic bits in a population. By the way, this expression depends on t, n_{cycles}, T and V_{Limit}. In our model, the aging of normal bits is responsible for the normal memory wear-out, whereas, erratic bits are abnormally increasing the in-line failure rate at the beginning of the memory life.

II.3.v. Parameter estimation

In order to calibrate the previous model, a 0.18 µm technology has been used. Parameters have been estimated and reported in Table II.2.

Table II.2: Estimated parameters for a 0.18um technology

Parameter	Value	Unit
t_{OX}	70	$\mathring{A}$
I_0	2.56×10^{-21}	A
α_G	0.885	–
C_{TOT}	1.47×10^{-15}	F
k_B	1.38×10^{-23}	$J\cdot K^{-1}$
α	2.403×10^{-20}	–
β	-1.086×10^{-1}	V
d_1	1.687×10^{-1}	V^{-1}
b	3.531	V^{-1}

II.4. Conclusion

This chapter was focused on the description of the Flash memory reliability. First of all, we provided basic notions on reliability. Secondly, mechanisms resulting in losses of endurance and retention have been exposed.

Next, a compact model of a single cell reliability has been proposed. This model is dependent from multiple parameters describing the behavior of a threshold

voltage drift. We have modeled the reliability of the non-volatile memory cell as the combination of two exponential distributions, one for the normal bits and one for the erratic bits. Between the distributions, only a log-time shift is applied, which is characteristic of an additional leakage current mechanism. From the one hand, the aging of the normal bits is responsible for the memory normal wear-out. From the other hand, erratic bits are increasing abnormally the in-line failure rate at the beginning of the memory life. In our modeling, no distinction is made to distinguish the cell-drift of the high threshold cells and low threshold cells. The model gives a rough estimation of what happen in non-volatile memory cells and is sufficient for reliability prediction that will be used in Chapter III.

Chapter III - Error Correction with V_T analysis

In this chapter an original error detection and correction scheme is exposed. It is a main contribution of this work and results were published in multiple conferences: [DATE07] [DDECS07]. The proposed technique is based on an analysis of the charge level into the floating gate. Associated with a standard ECC, this analysis improves the correction capacity of the standard ECC with practically no additional cost. In a second time, we consider that the memory has some redundancy elements which can be used to replace erroneous elements during the memory useful life. By merging the error correction scheme and redundancy, it is possible to improve the global reliability. This is demonstrated by a mathematical modeling based on the memory cell modeling established in chapter II. To put this scheme in application, a reliability management architecture is introduced and evaluated.

III.1. Motivation

Flash memories are analog devices allowing specific reliability enhancement methods. During a read operation, the modulation of the biasing conditions allows V_T level analysis. Bits with a weak charge level can be detected [48]. A cell refreshing scheme [49] and an error detection/correction scheme [50] based on V_T level analysis have already been proposed. However, in literature, architecture reliability evaluations are usually performed with a constant failure rate reflecting the SRAM cell reliability. Charge loss and cycling degradations are not taken into account even if multiple models for Flash reliability have been developed [28], [51]. In this part, we compare different methods to enhance Flash reliability using ECC, on-line redundancy, and V_T analysis. For this purpose, a memory array model using the cell reliability model exposed in section II.3 has been developed. The aim of this work is to help designers in choosing the most efficient reliability scheme to implement depending on the technology, memory architecture and reliability objective.

In order to introduce fault tolerance techniques used here such as error correcting codes and redundancy, the reader will refer to Annex A which provides some basic notions and literature examples.

III.2. Array conventions

Assume the memory array represented in the Figure III.1:

- In each word, additional bits are added for an error detection/correction system.
- In the array, spare rows are present in order to implement an on-line redundancy repair system.

Column redundancy is not considered here. Indeed, Flash are page-oriented during write/erase operations. These operations are time-consuming (few ms). To replace an entire column with redundancy, all pages of the array must be erased and written back in order to modify only one bit position. This on-line repair process is not realistic because the memory will not be available for a few seconds depending on the depth of the array. For instance, if the replacement of a bit position takes 4 ms and the array has 1024 pages, the column repair process will take more than 4 seconds. On the contrary, in case of row redundancy, only one page has to be programmed during the repair process.

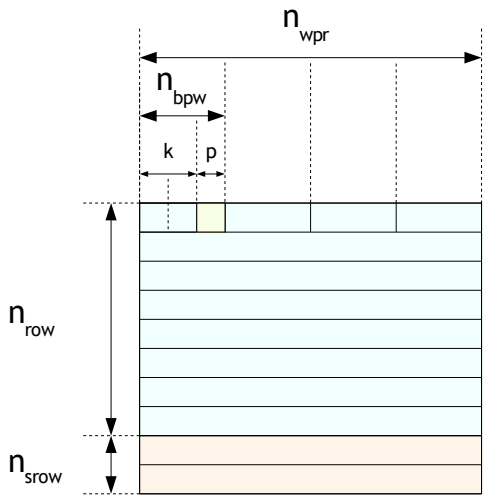

Figure III.1: Memory array modeling

Following notations are used to characterize the memory array:

- Each word is composed of k information bits and p parity bits with $n_{bpw} = k + p$ bits per word.
- Each row has n_{wpr} words.
- Each array has n_{row} normal rows and n_{srow} spare rows.
- cc and dc are defined as the error correction capacity and the error detection capacity associated to the error correcting code respectively.

In the rest of this part, we consider that reliability enhancement procedures incorporate three steps:

- ***Error Detection (ED)*** – An error state is detected in a memory word. This process is usually performed by a control of the likelihood based on an error detection/correction code.
- ***Error Localization (EL)*** – Locations of the erroneous bits in a memory word are determined. This step is performed using the capacity correction of an error correction code and/or a V_T analysis.
- ***Retrieval Mechanism (RM)*** – When a bit is detected to be weak or in error, the information sent to the user must be corrected. Additionally, operation can be performed on the memory array to physically repair it (using

redundancy), to refresh the data stored (using a refresh process) or to correct the information on the fly (using on-line detection/correction).

III.3. V_T analysis

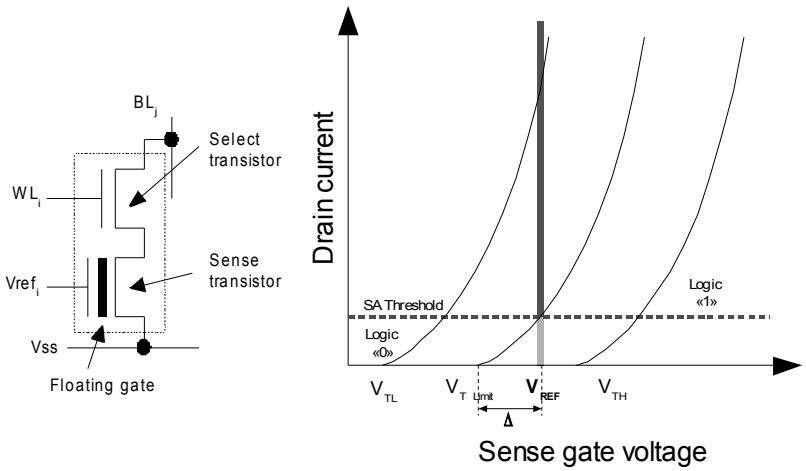

Figure III.2: FLOTOX with the floating gate concept

The floating gate cell current is sensed using a dedicated Sense Amplifier (SA) connected to the Bit Line of the cell. As referred in Figure III.2, for any given voltage biasing of the sense transistor's gate V_{REF}, there exists one single threshold voltages limit denoted $V_{T_{LIMIT}}$, such that:

- if the sense transistor has a V_T below $V_{T_{LIMIT}}$, it delivers a current I higher than I_{REF} and the sense amplifier provides a logic “0” on its output.
- if the sense transistor has V_T above $V_{T_{Limit}}$, it delivers a current I lower than I_{REF} through the bit-line and the sense amplifier provides a logic “1” on its output.

Where, I_{REF} is the SA threshold limit shown in Figure III.2.

It is important to notice the relationship $V_{REF} = V_{T_{LIMIT}} + \Delta$ with Δ being a constant between V_{REF} and $V_{T_{LIMIT}}$. Thanks to this principle, weak and erroneous states can be detected using a cell V_T sensing.

As illustrated in the Figure III.3, cells V_T are distributed over two distributions: one for cells with a high V_T (V_{T_H}), another for cells with a low V_T (V_{T_L}). We define

three biasing voltage V_{REF_L}, V_{REF_N} and V_{REF_H} for the sense transistor's gate such that $V_{REF_L} < V_{REF_N} < V_{REF_H}$. These references correspond equivalently to a low voltage limit V_L, a nominal voltage limit V_N and high voltage limit V_H. These V_{REF} voltages should be chosen in function of the programming window which is determined during post-production tests. In particular, V_{REF_N} must be centred to the middle of this window while V_{REF_L} and V_{REF_H} must be chosen between the low and high bounds of the programming window and V_{REF_N} respectively. For instance, if the bounds of the programming window are -2V and 2V, a priori, good values for V_{REF_L}, V_{REF_N} and V_{REF_H} could be respectively -1V, 0V and 1V.

Now, assume a memory cell has been programmed to a logic “1” (or logic “0”). As shown in the Figure III.3, this cell can be in one out of four states depending on its threshold voltage. These states are denoted: hard error, weak error, weak good, good.

For instance, to determine whether a cell stores a good logic-“1”, we must verify that the cell has a threshold voltage above the limit V_H. Then, a read operation with the high V_{REF} (V_{REF_H}) bias applied on the gate of the sense transistor must be carried out. A cell with a V_T higher than V_H will provide a logic-“1” whereas a cell with a V_T lower will provide a logic-“0”. The comparison of multiple read operations by modulating the sense transistor's gate voltage (V_{REF}) shown in Figure II.2 allows to determine the memory cell state. This type of operation is called ***margin read*** and is widely used to verify the programming window during post-production test.

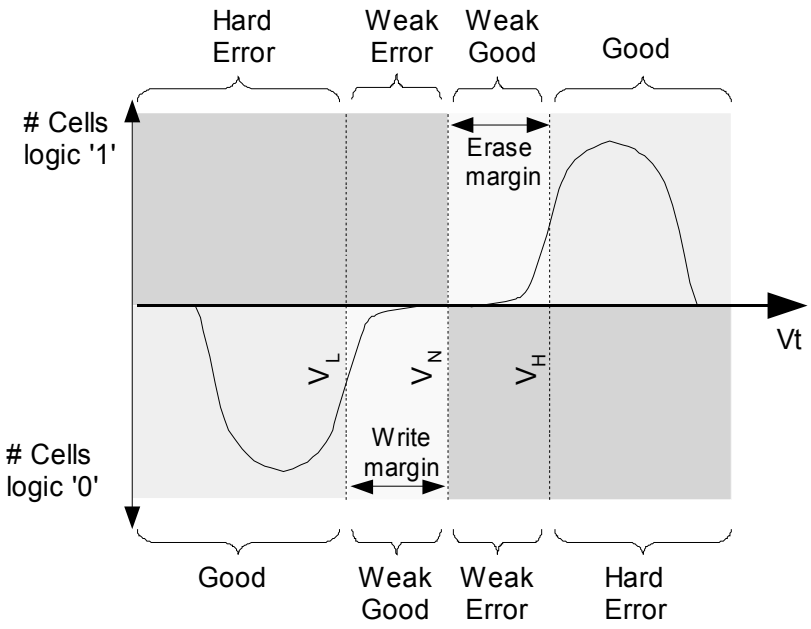

Figure III.3: Example of cell VT distribution

A new detection process, called ***V_T analysis***, can be defined to detect if a bit is weak (in the slice $\left[V_L, V_H\right]$):

- First, two read operations using V_L and V_H as threshold voltage limits are performed on a word. This is done by applying successively V_{REF_L} and V_{REF_H} on the gate of the sense transistor shown in Figure III.2.
- Next, weak bit locations are found making a bitwise comparison of read operation's results. Bits which are weak are set to 1, others are set to 0.

In the rest of this section, the concept of margin read is generalized and used to detect and locate bits with growing reliability issues. For instance, we will explain with more details how a 1EC-2ED ECC can be associated with margin read operations to provide a double error correction system.

To estimate reliability improvements, we develop a mathematical model and compare it with other reliability enhancement approaches in section III.4. Next, in section III.5, the underlying architecture is described and some results are provided.

III.4. Memory reliability modeling

III.4.i. Cell-V_T repartition in a word

The Figure III.4 shows an example of erase distribution. In all the section III.4, only this type of distribution will be considered to simplify the modeling. As we can see, the figure is composed of three threshold voltage limits: a low voltage V_L, a nominal voltage V_N and a high voltage V_H. When performing a read operation, one of these three V_T limits is chosen. Bits with V_T higher than this limit will correspond to logic value "1". In the same way, bits with V_T lower than this limit will correspond to logic value "0". We can note that selecting a V_T limit is equivalent to perform a read with a particular biasing of the cell control gate as explained in section III.3.

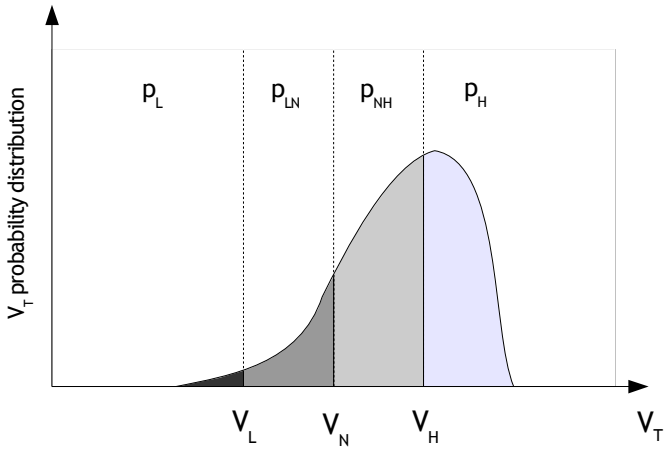

Figure III.4: VT probability distribution with VT limits

After an erase cycle, all the V_T distribution is shifted towards high V_T values. But, with time, the distribution drifts to lower values as illustrated in the Figure III.4. Then, the V_T of each bit in a word is located in one of the four V_T slices with a given probability. These probabilities are accessible through the model established in the section II.3:

$$R_{Cell} = P\left(V_T > V_{Limit}\right) = f\left(t, n_{cycles}, T, V_{Limit}\right) \quad \text{(III.1)}$$

$$\ln\left(-\ln\left(1-P_N\right)\right) = \frac{1}{d_1 \cdot b}\cdot\ln\left(I_0\right) + \frac{1}{d_1 \cdot b}\cdot\ln\left(\frac{b}{C_{TOT}\cdot\alpha_G}\right) + \ln\left(d_0\right) + \ln\left(N\right) + \frac{\ln\left(t\right)}{d_1 \cdot b} - \frac{\alpha}{d_1 \cdot k_B \cdot T} - \frac{\beta\cdot\ln\left(n_{cycles}\right)}{d_1} + \frac{\left(V_{T_{ADJ}} - V_{TNA}\right)}{d_1} \quad \text{(III.2)}$$

In the Table III.1, probabilities to be in each of the four slices are provided. Notions of hard failing bits and weak failing bits are also defined.

Table III.1: Bits conventions and associated probabilities

Slice	Type	Corresponding Probability
$V_T > V_H$	Good bits	$p_H = R_{cell}(V_H)$
$V_T \in [V_N, V_H]$	Weak good bits	$p_{NH} = R_{cell}(V_N) - R_{cell}(V_H)$
$V_T \in [V_L, V_N]$	Weak failing bits	$p_{LN} = R_{cell}(V_L) - R_{cell}(V_N)$
$V_T < V_L$	Hard failing bits	$p_{LN} = 1 - R_{cell}(V_L)$

Let us consider a word composed of N_{Cells}. There are a lot of different ways to distribute bits in the four slice. We can easily count for them and associate a probability. The probability of having respectively N_L, N_{LN}, N_{NH}, N_H bits in slices 1, 2, 3 and 4 is described by a multinomial repartition [52]:

$$p_W(N_L, N_{LN}, N_{NH}, N_N) = \frac{N_{Cells}}{N_L! \cdot N_{LN}! \cdot N_{NH}! \cdot N_H!} \cdot p_L^{N_L} \cdot p_{LN}^{N_{LN}} \cdot p_{NH}^{N_{NH}} \cdot p_N^{N_H} \quad \text{(III.3)}$$

Where, $N_{Cells} = N_L + N_{LN} + N_{NH} + N_H$. In other words, Equation (III.3) provides the probability of cells' V_T repartition in a word. To provide further explanation on Equation (III.3), we can see that this expression is composed of two terms $\frac{N_{Cell}}{N_L! \cdot N_{LN}! \cdot N_{NH}! \cdot N_H}$ and $p_L^{N_L} \cdot p_{LN}^{N_{LN}} \cdot p_{NH}^{N_{NH}} \cdot p_N^{N_H}$. The first term $\frac{N_{Cell}}{N_L! \cdot N_{LN}! \cdot N_{NH}! \cdot N_H!}$ is a count of the number of ways to have N_L bits in slice 1, N_{LN} bits in slice 2, N_{NH} bits in slice 3, N_H bits in slice 4. Once this value has been calculated, it must be associated with the probability that this configuration occurs, this is the second term $p_L^{N_L} \cdot p_{LN}^{N_{LN}} \cdot p_{NH}^{N_{NH}} \cdot p_N^{N_H}$.

III.4.ii. Array Reliability with detection/localization procedures

Three detection/localization procedures A, B and C are studied. As shown in Table III.2, procedures depend on the error correcting code implemented and so, on the number of parity bits added per word. Potentially, procedures A and B allow one error per word correction whereas the procedure C allows two errors correction.

Table III.2: Detection/localization procedures

		Procedure A	Procedure B	Procedure C
Error correction code		**Parity Code**	**Hamming Standard**	**Extended Hamming**
p		1	$\log_{2(k)}+1$	$\log_{2(k)}+2$
dc		1	1	2
cc		0	1	1
Detection		By the detection capacity of the ECC		
Localization	**1 error**	V_T analysis	ECC	ECC
	2 errors	-	-	V_T analysis
Total number of correctable errors		1	1	2

When a word is read, the on-line ECC mechanism is used to detect errors. If error correction capacity of the ECC is sufficient, errors are automatically corrected and the result is sent to the user. If the error correction capacity is exceeded but error detection capacity is still sufficient, a V_T analysis will determine weak bits in the slice $\left[V_L, V_H\right]$. Then, the following assumption is made:

- the weak bits discovered during the V_T analysis are failing bits that have not drifted enough to be hard errors.

Consequently, if the number of weak bits in the word equals the number of error detected, the inversion of weak bits allows to recover the correct word. To illustrate that purpose, let us consider the procedure C. In this case, the Extended Hamming Code is used, so up to two errors can be detected ($dc=2$) and only one can be corrected ($cc=1$). A read is performed on a word. If the word has a single error, the correction capacity is not exceeded. The ECC mechanism is able to transparently detect and correct the error. Now, if the word has two errors, the correction capacity is exceeded but not the detection capacity. The ECC mechanism is able to analyze the problem: two errors have been detected but not located. Hence, a V_T analysis is launched to locate weak bits. If two weak bits are found in the word, their values are inverted and the word is supposed to have been corrected. Table III.2 summarizes detection and localization procedures A, B and C.

Reliability enhancements with the three detection/localization procedures are now analyzed. For that purpose, we enumerate all reliable V_T repartitions in a word:

- ***Procedure A*** - A memory word is correct if:
  - For all cells, $V_T > V_N$ i.e. multiple bits may be weak but there is no error.
  - Or, one cell has a V_T in $\left[V_L, V_N\right]$ slice and all the others have $V_T > V_H$ i.e. one error is present due to one weak bit.

It corresponds to the probability:

$$p_{word}^{A} = \sum_{i=0}^{n_{bpw}} p_W\left(0,0,i, n_{bpw} - i\right) + p_W\left(0,1,0, n_{bpw} - 1\right) \quad \text{(III.4)}$$

- ***Procedure B*** - A memory word is correct if:
  - For all cells, $V_T > V_N$ i.e. multiple bits may be weak but there is no error.
  - Or, one cell has a $V_T < V_N$ and all the others have $V_T > V_N$ i.e. there is only one error.

It corresponds to the probability:

$$p_{word}^{B} = \sum_{i=0}^{n_{bpw}} p_W\left(0,0,i, n_{bpw} - i\right) + \sum_{i=0}^{n_{bpw}-1} p_W\left(0,1,i, n_{bpw} - i - 1\right) + \sum_{i=0}^{n_{bpw}-1} p_W\left(1,0,i, n_{bpw} - i - 1\right) \quad \text{(III.5)}$$

- ***Procedure C*** - A memory word is correct if:
  - For all cells, $V_T > V_N$ i.e. multiple bits may be weak but there is no error.
  - Or, one cell has $V_T < V_N$ and all the others have $V_T > V_N$ i.e. there is one error.
  - Or, two cells have a V_T in $\left[V_L, V_N\right]$ slice and all the others have $V_T > V_H$ i.e. there are two errors due to weak bits.

It corresponds to the probability:

$$\begin{aligned} p_{word}^{C} = & \sum_{i=0}^{n_{bpw}} p_W\left(0,0,i, n_{bpw} - i\right) + \sum_{i=0}^{n_{bpw}-1} p_W\left(0,1,i, n_{bpw} - i - 1\right) \\ & + \sum_{i=0}^{n_{bpw}-1} p_W\left(1,0,i, n_{bpw} - i - 1\right) + p_w\left(0,2,0, n_{bpw} - 2\right) \end{aligned} \quad \text{(III.6)}$$

There are $n_{wpr} \cdot n_{row}$ words per array. Consequently, reliability expressions for pages and arrays without redundancy in procedures A, B and C are obtained from equations (III.4), (III.5), (III.6):

$$R^{i}_{page} = \left(p^{i}_{word}\right)^{n_{wpr}} \quad \text{(III.7)}$$

$$R^{i}_{ECC} = \left(p^{i}_{word}\right)^{n_{wpr} \cdot n_{row}} \quad \text{(III.8)}$$

where, i is replaced by the chosen procedure A, B or C.

I.1.i.h. Array Reliability with on-line repair procedure

On-line repair with row redundancy can be considered as a retrieval mechanism:

- As soon as an error is detected, the entire page is replaced with row redundancy if available.

The corresponding reliability is given by the combinatorial probability:

$$R^{i}_{} = \sum_{k=0}^{n_{srow}} C^{k}_{n_{row}+n_{srow}} \cdot \left(R^{i}_{page}\right)^{n_{row}+n_{srow}-k} \cdot \left(1-R^{i}_{page}\right)^{k} \quad \text{(III.9)}$$

III.4.iii. Array Reliability with mixed repair and ECC procedure

In reality, when all redundancy rows have been used, the error detection/localization system still corrects errors and the memory continues to be reliable. In other words, the reliability expression (III.9) is used for memories whose number of pages in error is at least equal to the number of spare rows. Assuming $n_{srow} > 0$, the probability that some redundancy is still available is expressed by:

$$p^{i}_{n_{srow}} = \sum_{k=0}^{n_{srow}-1} C^{k}_{n_{row}+n_{srow}} \cdot \left(R^{i}_{page}\right)^{n_{row}+n_{srow}-k} \cdot \left(1-R^{i}_{page}\right)^{k} \quad \text{(III.10)}$$

With this procedure, the reliability of the array is:

$$R^{i}_{RED+ECC} = \left(1-p^{i}_{n_{srow}}\right) \cdot R^{i}_{ECC} + p^{i}_{n_{srow}} \quad \text{(III.11)}$$

III.5. Architecture for reliability management

Each time an operation is performed on the memory (write or read), the reliability structure is requested to prevent errors (endurance or retention) that may occur. The structure controls and corrects errors and eventually locates weak bits by V_T

analysis. While there is only one error in a word, the structure transparently corrects it. When double errors or weak errors are present in a word, the structure launches a self-Refresh and Repair process. The reliability structure avoids that the double error correction capacity per word may be caught out. Our reliability management scheme is composed of three functional parts:

- A ***Read process*** integrating a double error correction per word procedure.
- A ***Write process*** including an enhanced verification procedure. It manages margins during programming operation and verifies that there is only one error or one weak per word at the end of programming operations.
- A ***Refresh and Repair process*** activated if a double issue on a word is detected. This process can take place during memory idle time.

III.5.i. Architecture description

The memory architecture is separated in two parts. The first part Figure III.5 corresponds to the internal memory architecture. The second part Figure III.6 is an external wrapper that performs a smart reliability management. This wrapper can be easily appended to the memory design. A memory without this wrapper would be functional but no reliability improvement would be available.

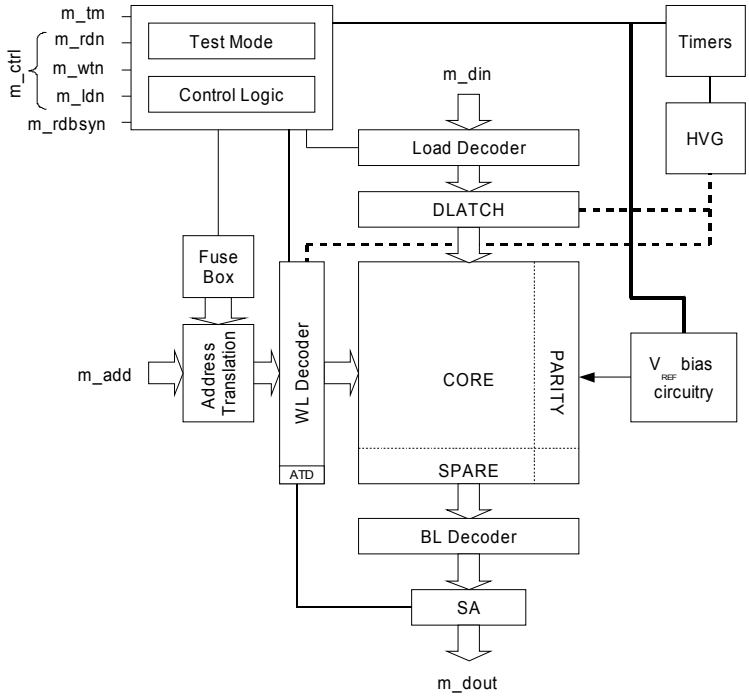

Figure III.5: Modified Flash memory architecture

The modified architecture of an embedded Flash memory is given in Figure III.5. It is composed of the standard blocks with some extra features for reliability purpose: the memory array has been extended to store the ECC parity (PARITY) and it integrates page redundancy (SPARE). A fuse box associated with some reconfiguration logic acts as a content addressable memory. It redirects memory operations on an error free redundancy element if the normal location is erroneous. Usually, the manufacturer programs the fuse box through a special test mode during the post-production phase. In this mode, address, control and data pins serve to program the fuse box. Another test mode also permits to read back the content of the fuse box through the output data bus. To end, a biasing circuitry generates voltages V_{REF_L}, V_{REF_N}, V_{REF_H} used for margin read operations. In the normal read mode, the V_{REF_N} bias is used. Other modes are made externally available through specific test modes thanks to *tm* pins.

The reliability manager shown in the Figure III.6 is a digital block. It constitutes an interface between the user and the memory. This block is composed of four parts. A reliability controller sequences one of the five modes: read, load, write, refresh and repair. An ***ECC encoder*** (ECCe) encodes word during user data loads. An ***Original Page Buffer*** (OPB) stores the content of a page to write. Typically, it is a small volatile memory of one page length with word granularity. To end, an ***Error & Weak Detector Locator*** (EWDL) detects, locates and corrects errors or weak bits in a memory word. The EWDL block is composed of an ECC decoder, comparison logic and two registers referred as A and B.

During the post-production phase, the Reliability manager is disconnected by use of the *tm* pins for the memory test purpose. During the normal mode, the Reliability manager is always connected. Consequently, the reliability manager substitutes memory inputs and outputs.

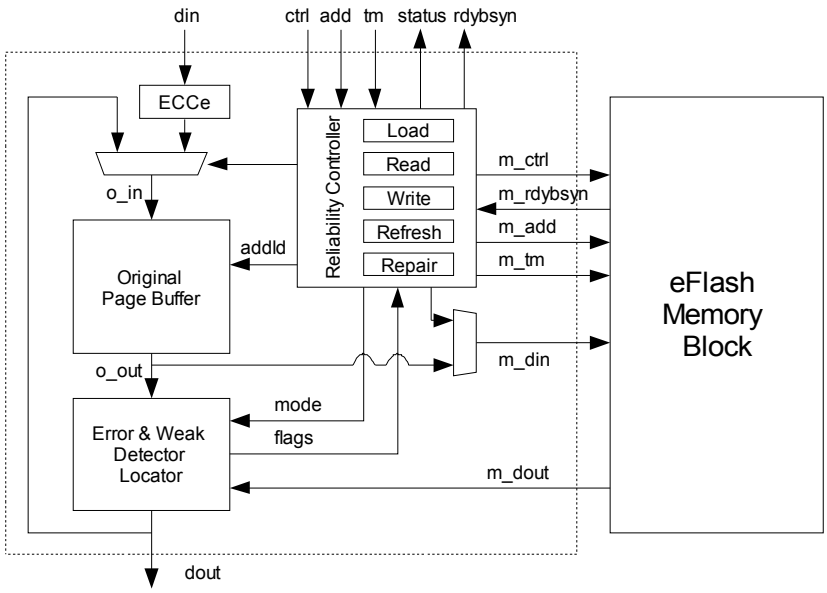

Figure III.6: Reliability manager

III.5.ii. Functional description

III.5.ii.1. Read process with double error correction

The double error correction scheme is activated when a user read operation is performed. All words are ECC encoded into the memory with an extended Hamming code. This code provides one error correction and double error detection per word. The code has been chosen in a way that an information word with all-“1” bits provides a parity with all-“1” bits. In fact, the ECC coding must be consistent with an erased word to avoid unwanted failures. An ECC word is composed of k information bits and $p = log_2(k) + 2$ parity bits. The Figure III.7 illustrates the sequence to perform a read and correct two bits per word. The functions are operated by the EWDL and sequenced by the reliability controller.

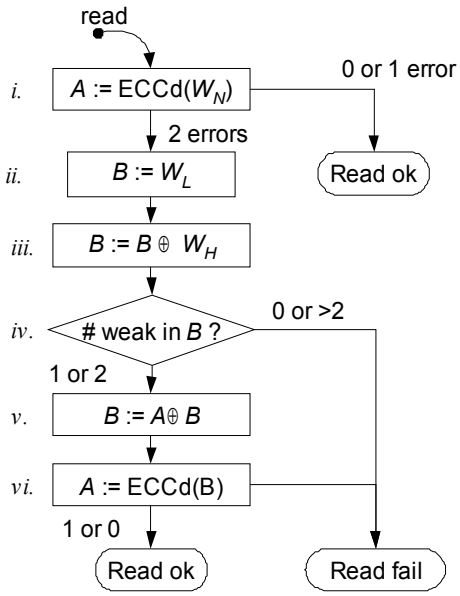

Figure III.7: Double error correction algorithm

The *ctrl* pins are set in a particular configuration so that the reliability controller enters in the read mode. First, the word W_N is read from the memory using the V_N limit (*i.*). W_N passes through the ECC decoder. If W_N has 0 or 1 error, W_N is corrected. A flag from the status pins is set to inform that the word is valid and the corrected data is sent via the *dout* pins. If W_N has 2 errors, a double error is signaled by the ECC decoder to the reliability controller. A flag is set to inform that the word is not valid. The word W_N not corrected is stored into the register *A*.

Then, a process is engaged to locate errors. Two reads are performed successively. The read with the V_L limit provides the word W_L during the step (*ii.*). The read with the V_H limit provides the word W_H during the step (*iii.*). Results are compared and stored in the register *B*. This comparison provides a pattern revealing weak bits location.

The number of ones in the weak bit pattern is counted for (*iv.*). If there is no weak or strictly more than two weak bits, the error cannot be located. The word is uncorrectable and the memory failed, a flag is set. If there are 1 or 2 weak bits, we suppose they are in error. They must be inversed. Consequently, the initial word W_N in *A* is compared (*v.*) with the weak bit pattern in *B*.

The result is a word with 1 or 0 errors. This error is a hard error that has not been yet corrected. The word passes through the ECC decoder to correct this last error (*vi.*). The read is done. A flag is set to inform that the word is valid and the corrected data is sent via the *dout* pins.

As a result, this correction scheme corrects a maximum of two errors in a word under following conditions:

- At least one of the erroneous bits is in weak error.
- Any good bit is weak.

III.5.ii.2. Load and Program with self-verification features

Embedded Flash memory write and erase operations always concern a page. During a write, cells of a page conserve their logical state or switch to a logic-0. During an erase, all cells of a page are simultaneously set to logic-1. To write the same word with different data, the entire page must be erased and rewritten.

The Figure III.8 describes the write algorithm. For an erase, the general concept is similar and is not shown here. It is composed of four phases: the ***User Load***, the ***Original Data Save & Load***, the ***Internal Write*** and the ***Last Verification***.

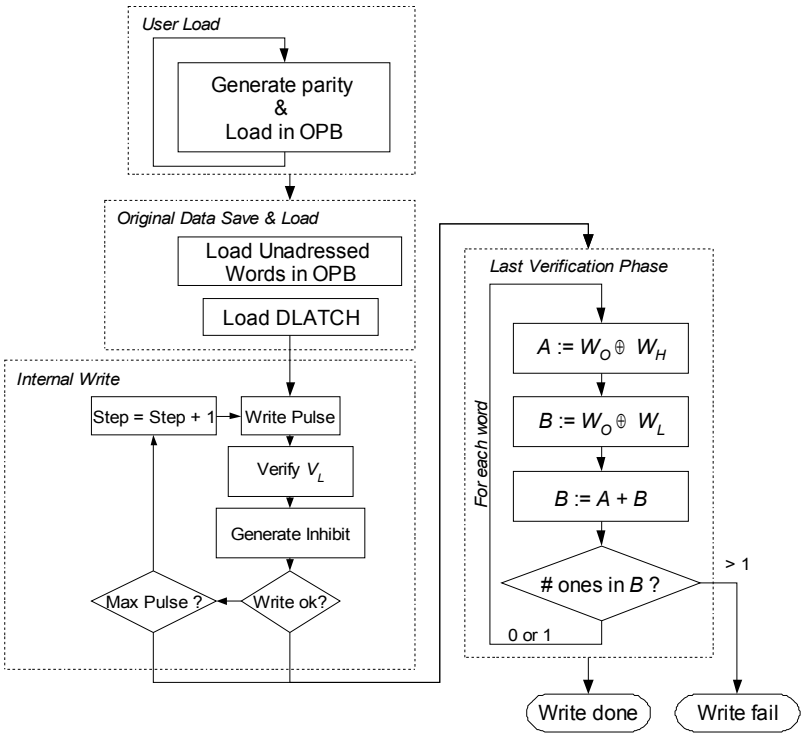

Figure III.8: Programming algorithm with error verification

During the User Load, user data are ECC encoded and stored into the OPB by the reliability controller.

Then, the user launches a write via the *ctrl* pins and the controller enters in the write mode. First, the ***Original Data Save & Load*** phase begins. Words that have not been previously loaded by the user are read back from the memory,

corrected and placed into the OPB. Next, the content of the OPB is sent to the DLATCH word by word.

The reliability controller activates the ***Internal Write*** of the memory. Several writing pulses followed by verify operations are performed on the page. It allows adjusting exactly the programming delay for each bit [36]. Bits non-uniformity and degradations are tracked after several programming cycles. After each pulse, verify steps take place, the page content is read back to detect bits that have been programmed with a sufficient margin V_L. The result is compared to the data stored in the DLATCH and an inhibition pattern is generated in order to stop the programming of written bits during the next pulse. Finally, when bits are programmed or when the number of writing pulses has reached a maximum, a ***Final Verification*** step is performed.

During this phase, functions are operated by the EWDL and sequenced by the reliability controller. For each word of the programmed page, margins are checked. To make so, two read are carried out. First, the word W_L is read with the V_L limit and compared with the corresponding original word W_O of the OPB. Bits that must be in a logic-“0” state are thus checked. Secondly, the word W_H is read with the V_H limit and compared with the corresponding original word W_O of the OPB. Bits that must be in a logic-“1” state are checked. Then, a logic operation on register *A* and *B* determines a pattern revealing position of bits that are in error or weak. When two bits in a word have problems, a page refresh is launched. A new erase followed by a write are performed on the page. If this refresh fails, the on-line repair scheme is activated. However, if no more redundancy resources are available, the write operation failed and thus the memory is declared non-functional.

III.5.ii.3. Refresh and Repair process

The proposed process is divided into two phases: a ***Refresh phase*** and a ***Repair phase***. During this process, the memory cannot be addressed. Phases and transitions are shown in the Figure III.9.

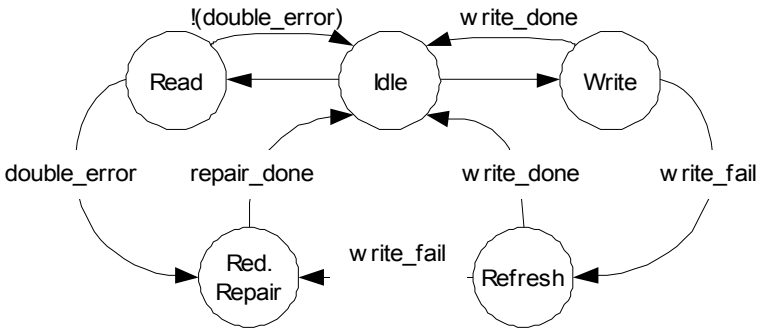

Figure III.9: Phase transition of the reliability scheme

During normal operation (read or write), the Refresh and Repair mechanism is not activated because the number of errors or weak bits per word is not critical. Write or read operations are completed successfully.

When a program operation fails because of multiple errors, the ***Refresh phase*** is launched. This operation is useful because after an erase, multiple write operation may have been performed on the page to program distinct words. However, this may disturb the V_T of not addressed cells that becomes weak or erroneous [30]. The refresh avoids a misuse of the redundancy resources by resetting cells-V_T before a write when it is necessary. As the original data is already present in the OPB, the page is erased and written back as soon as memory enters in this mode.

Next, if the ***Refresh phase*** results in a write fail, the memory enters in a redundancy ***Repair phase***. There are two cases depending on where the fail occurs. In the first case, the page belongs to the main memory. If a spare element is still available, a redundancy page replaces the faulty page. First, the faulty-page address is remapped to the redundancy-word by the fuse box programming. Next, data are written into the redundancy page. In the second case, the page belongs to redundancy elements. The faulty redundancy element is marked by programming of the reconfiguration circuitry. This step is necessary to avoid conflicts in the memory circuitry addressing.

If a read fails due to a double error or weak in a word, the ***Repair phase*** is launched. The page is considered unreliable which justifies the redundancy use. In this case, the repair may be postponed until the memory goes in idle mode.

III.6. Results and Discussion

The Figure III.10 is a comparison of the reliability between a standard eFlash array and eFlash arrays with detection/localization procedures developed in the

section III.4 after 10^5 program/erase cycles. As illustrative example, the V_T limits have been chosen as follow: $V_L = -1V$, $V_N = 0V$ and $V_H = 1V$. There are $n_{row} = 1024$ rows and $n_{wpr} = 64$ words per row. All curves are normalized in time by the ***Mean Time To Failure*** (MTTF) of the standard array noted $MTTF_{standard}$. Model parameters have been calibrated on a 180 nm eFlash technology, based on measurements performed on samples of a 2Mb memory.

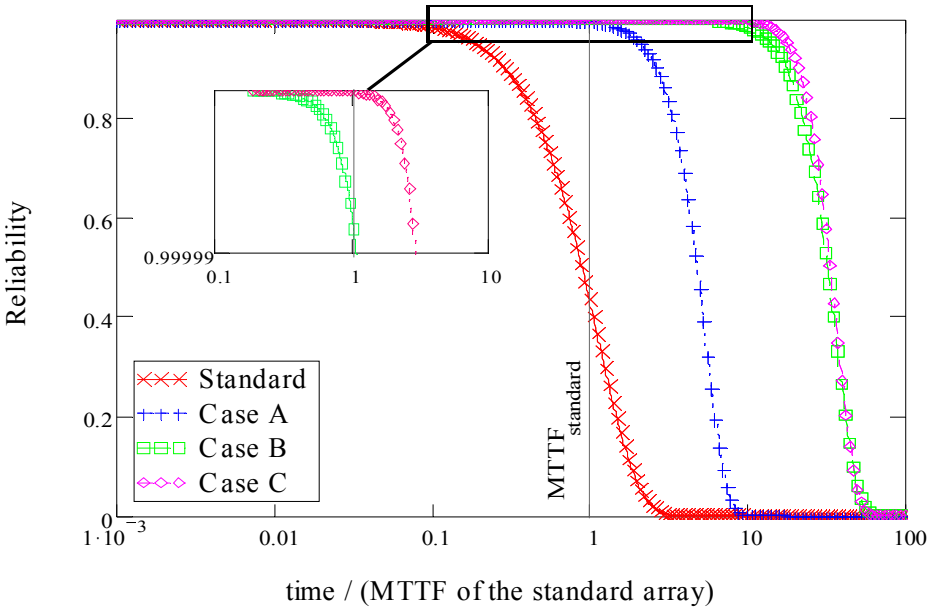

Figure III.10: Reliability of 2Mbits arrays using detection/localization procedures only

With the procedure A, only one error can be corrected using a V_T analysis. This scheme can correct one weak failing bit per word. As illustrated in the Table III.3, the MTTF is improved by a factor 4.67 with this procedure in comparison with a standard array. However, the hard failing bits are not reachable. Consequently, the reliability improvement is lesser than with the procedure B where one weak failing or one hard failing bit can be corrected thanks to the Hamming Correcting Code. In the procedure B, the MTTF is improved by a factor 25.1 in comparison with a standard array. Namely, if a memory has a MTTF equal to 1 year, then in procedures A and B, the MTTF will be respectively improved to 4.67 years and 25.1 years. In the procedure C, there is no noticeable improvement of the reliability compared to the procedure B even if an Extending Hamming Code and V_T analysis are used to correct up to two errors: the MTTF gain is only 26.3. Nevertheless, the reliability decrease occurs later in the procedure C than in the

procedure B. This phenomenon is observed focusing on the beginning of the curve decrease as shown in the Figure III.10. For instance, at time $MTTF_{standard}$, 56.3% of the standard arrays would have failed. In procedures A, B and C, only 6263, 7.2 and 0.08 parts per million (ppm) would have failed respectively. The two decades difference between procedures B and C may justify the adoption of the procedure C for products needing high reliability rates.

The impact of the erratic bits ratio on the reliability scheme is low. Indeed, if the ratio is increased by a factor 10^2, then, 9.7 ppm, 0.11 ppm would have failed in procedures B an C at time $MTTF_{standard}$.

2Mbits eFlash array with distinct word lengths have been reported in the Table III.3. Array overheads and MTTF gains are presented. MTTF gains are independent of the number of cycles due to the logarithmic dependence of n_{cycles} in our cell model. Our cost function is defined as the ratio between the array overhead and the logarithm of the MTTF gain. When the word length is increased the MTTF gain is reduced but the overhead impact decreases far more rapidly. In consequence, the cost function decreases. Array requiring moderate reliability improvement for a very low cost may adopt the procedure A. On the contrary, to be fully reliable, the procedures B or C should be adopted.

Table III.3: Summary of 2 Mbits arrays using detection and localization procedures only

	Procedure A			**Procedure B**			**Procedure C**		
Max. Correction	1			1			2		
Constant Parameters	$n_{row}=1024$, $n_{wpr}=64$, $V_L=-1$, $V_N=0$, $V_H=1$								
k	32	64	128	32	64	128	32	64	128
p	1	1	1	6	7	8	7	8	9
n_{wpr}	64	32	16	64	32	16	64	32	16
Array Overhead (%)	3.1	1.6	0.8	18.7	10.9	6.2	21.9	12.5	7.0
MTTF Gain	4.67	3.80	3.16	25.1	23.4	20.9	26.3	24.0	21.4
Cost $\frac{Overhead}{\log(MTTF\ gain)}$	4.6	2.8	1.6	13.4	8.0	4.7	15.4	9.1	5.3
# Defective arrays at $MTTF_{standard}$ **(ppm)**	6263	12663	17637	7.2	12.6	23.4	0.08	0.2	0.59

The Figure III.11 shows a reliability comparison between a standard array and arrays with mixed detection/localization procedures (A or B) and the on-line repair developed in the section III.4.iii after 10^5 program/erase cycles. The V_T limits have been chosen as follow: $V_L = -1\,V$, $V_N = 0\,V$ and $V_H = 1\,V$. There are $n_{row} = 1024$ rows and $n_{wpr} = 64$ words per row. The number of row redundancy is a parameter.

At first look, the on-line repair makes the reliability slope sharper. In Table III.4, we have reported the number of defective arrays at time $MTTF_{standard}$. This number becomes zero as soon as some redundancy is added. This observation is independent of the detection/localization procedures used. Thanks to Table III.4, we can note that the procedure A with 2 rows results in less defective arrays after $MTTF_{standard}$ than the procedure B with 0 rows. Consequently, the array becomes very reliable at time $MTTF_{standard}$ adding the online repair. The Figure III.11 shows also that the MTTF gain is increased adding few rows. But, the improvement reduces slowly with each new row. In the Table III.4, this is traduced by a slowdown of the slope of the cost function. As a result, only a few number of rows are useful. The procedure A associated with on-line repair would be a very good choice to manage reliability at a reduced array overhead cost.

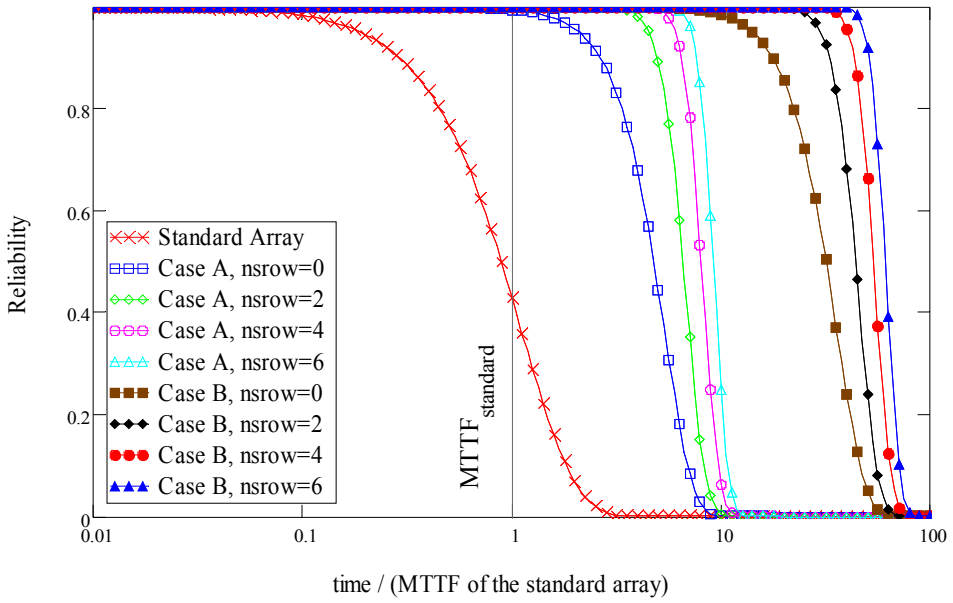

Figure III.11: Reliability of 2Mbits arrays using mixed detection/localization and on-line repair procedures

Table III.4: Summary of 2Mbits arrays reliability using mixed detection/localization and on-line repair procedures

	Procedure A				Procedure B			
Max. Cor.	1				1			
Constant parameters	k=32, n_{row}= 1024, n_{wpr}= 64, V_L= -1 , V_N= 0, V_H= 1							
p	1				6			
n_{srow}	0	2	4	6	0	2	4	6
Array Overhead (%)	3.1	3.3	3.5	3.7	18.7	19.0	19.2	19.4
MTTF Gain	4.67	6.31	7.76	8.91	25.1	41.7	50.1	57.5
Cost $\frac{Overhead}{\log(MTTF\ Gain)}$	4.6	4.1	3.9	3.9	13.4	11.7	11.3	11
# Defective arrays at $MTTF_{standard}$ (ppm)	6263	0.1	~0	~0	7.2	~0	~0	~0

Results are shown in the Figure III.12. Curves represent expected reliabilities after 10^5 cycles for a standard 2Mbits array and for 2Mbits arrays with double error correction and online redundancy repair. All curves have been normalized by the Mean Time To Failure of the standard array ($MTTF_{standard}$). Considered arrays are composed of n_{row} = 1024 pages, n_{wpr} = 64 words per page and $n_{bpw} = k + p$ bits per word. Each word contains $k = 32$ information bits with $p = 0$ parity bits for the standard array or $p = 7$ parity bits due to extended hamming code for arrays with reliability management. The number of spare rows n_{srow} for online redundancy is variable.

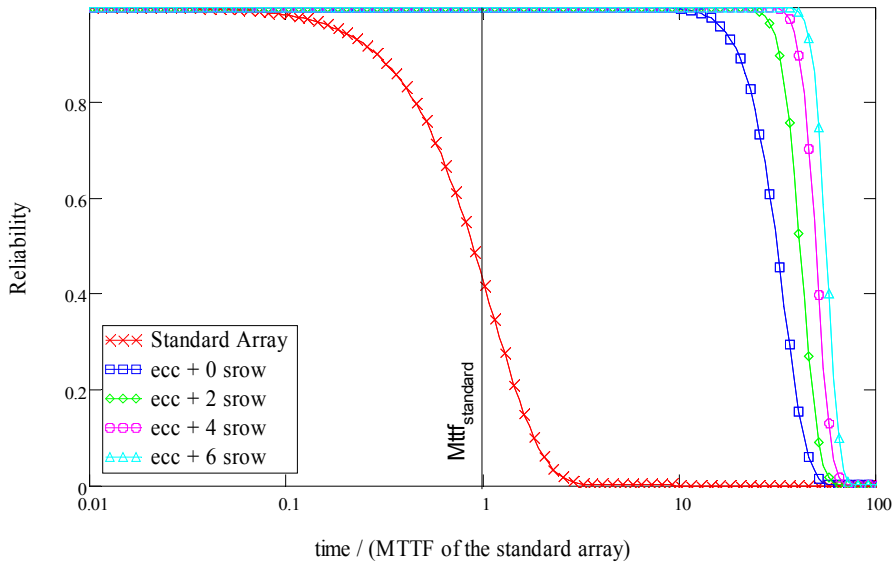

Figure III.12: Reliability of 2Mbits arrays with and without reliability management

The Figure III.12 shows that the reliability is greatly improved when a double error correction scheme is used. Moreover, the addition of on-line redundancy makes the reliability slope sharper with some reliability gain. Results have been reported in the Table III.5. The MTTF gain is 26 for an array with 32 bits words and double error correction. Consequently, if the $MTTF_{standard}$ is 1 year, the MTTF of an array with double error correction scheme without redundancy would be 26 years. Adding 2 rows for online redundancy will improve the MTTF gain to 42 years. The part per million (ppm) improvement is also very important at the time $MTTF_{standard}$. At $MTTF_{standard}$, more than 40% of standard memories would have fail. In the case of double error correction scheme, less than 1 ppm would have failed. Adding a low number of row redundancy would allow to reach 0 ppm.

Table III.5. Reliability improvement and cost for 2Mbits memories

	double error correction with online repair							
Max. Cor.	2							
Constant	k=32, n_{row}= 1024							
n_{wpr}	64				16			
k	32				128			
p	7				9			
n_{srow}	0	2	4	6	0	2	4	6
MTTF Gain	26.3	41.9	50.1	56.7	21.9	29.3	35.5	40.2
Ppm at $MTTF_{standard}$	0.08	~0	~0	~0	0.59	~0	~0	~0
(%) Total Memory Overhead	14.7	15.0	15.2	15.5	5.1	5.3	5.5	5.8

The implementation of the reliability management (Reliability manager, parity bits, row redundancy, fuse box and reconfiguration logic) introduces a global memory overhead which has been calculated. A 32 bits word induces an important array overhead in comparison with 128 bits word. This is mainly due to the array extension in order to store the parity. In case of 128 bits word, array overhead is approximately 5%, which should be a reasonable cost for application requiring very high reliability rates. As illustrated in the Table III.6, the area overhead due to reliability features depends a lot on memory size. In fact, periphery overhead represents a large area in small eFlash memories. However, it evolves lower than the memory array when the memory size is increased. Consequently, parity overhead represents a larger area with density and it justifies that the reliability architecture overhead increases.

Table III.6: Total memory overhead in function of the memory size

Memory size	512kBits		1024kBits		2048kBbits		4096kBits	
Constant Parameter	n_{srow} = 4							
n_{row}	256		512		1024		2048	
n_{bpw}	39	137	39	137	39	137	39	137
(%) Total Memory Overhead	9.7	4.0	13.5	5.1	15.2	5.5	16.8	6.0

III.7. Conclusion

In this chapter, we have introduced the original approach of error correction with V_T analysis [DATE07]. This method use a standard ECC in addition with a system able to locate bits storing weak logic values. If the ECC detect an error which is not able to correct, the method consists in considering that weak bit positions are very likely in error and should be corrected by flipping the bits content. The advantage of this scheme is to extent the error correction capacity of the standard ECC. For instance, with a Parity Code which is only able to detect one error, error correction with V_T analysis is able to correct up to one error. From a result point of view, it could improve the memory Mean Time To Failure (MTTF) by a factor 4,7 in a 2Mbits array with 32 bits per word.

In parallel, with the introduction of this technique, we also considered an architecture that could perform some reliability management [DDECS07]. The purpose of this scheme was to replace erroneous page by redundant page as soon as an error is detected. In term of results, reliability improvements are impressive. For instance, considering a 2MBits 32 bits per word memory using the Error Correction with V_T analysis to correct up to 2 errors, the MTTF is improved by a factor 26.3 compared to a memory without ECC. Now, if 6 rows redundancy are also available and managed by the reliability management architecture, the MTTF is improved by a factor 56.7. In term of area overhead, this architecture represents approximately a supplement of about 15%.

Chapter IV - Hierarchical error correction

In this chapter, a second reliability management scheme for NOR embedded Flash memories is exposed. This is a main contribution of this work. The originality of this approach relies on the use of an error correcting code well suited to NOR flash memories operational conditions. This code, named hierarchical code, improves the correction capabilities with a minimal impact on performances and area. The proposed reliability management scheme requires consider that depending on the number of errors in a page, it should be refreshed or repaired with redundancy. To prove the efficiency of this scheme, a mathematical modeling based on Homogeneous Markov Process is depicted. However, due to the complexity of the scheme, the model of chapter II has not been employed for the benefit of a simplified model of reliability. We will show how this scheme can improve the memory reliability at a very low cost.

IV.1. Architecture for on-line reliability management

IV.1.i. Flash architecture and operating granularities

NOR Flash memories are accessed differently during reading and programming operations: Program is done per page or group of words and read is done per word. Read is a fast operation (up to 40 MHz). To limit silicon area and power consumption due to sense amplifiers, only a group of bits (32 to 128 bits) is read by operation. Programming is made by page (64 to 256 bytes) and it is a slower operation in the range of milliseconds. Usually, during programming operation, the whole content of the page is first erased to 'FF' and next written to user content. Consequently, there exists a granularity difference between the read operation (by word) and the program operation (by page). In the section IV.2, we will show how to use this difference in building an optimal error correction scheme.

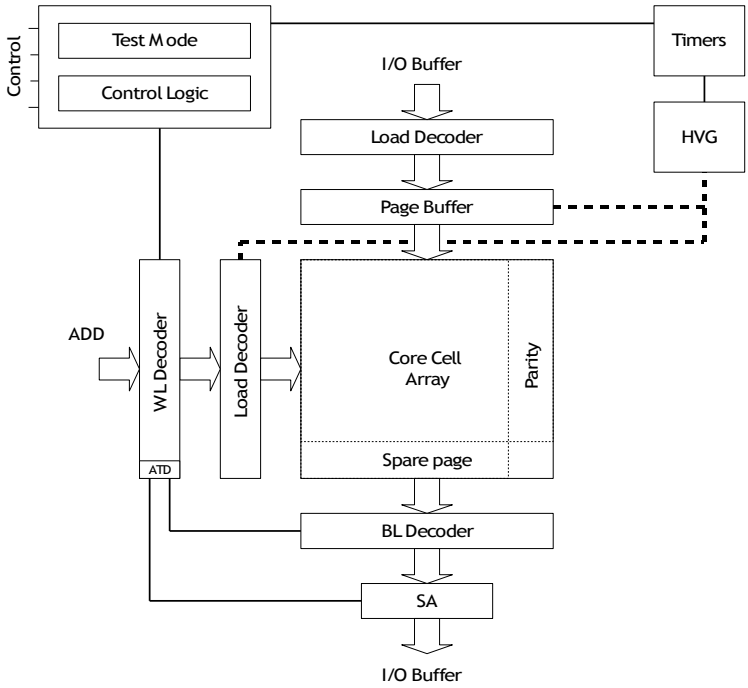

Figure IV.1: Flash memory architecture

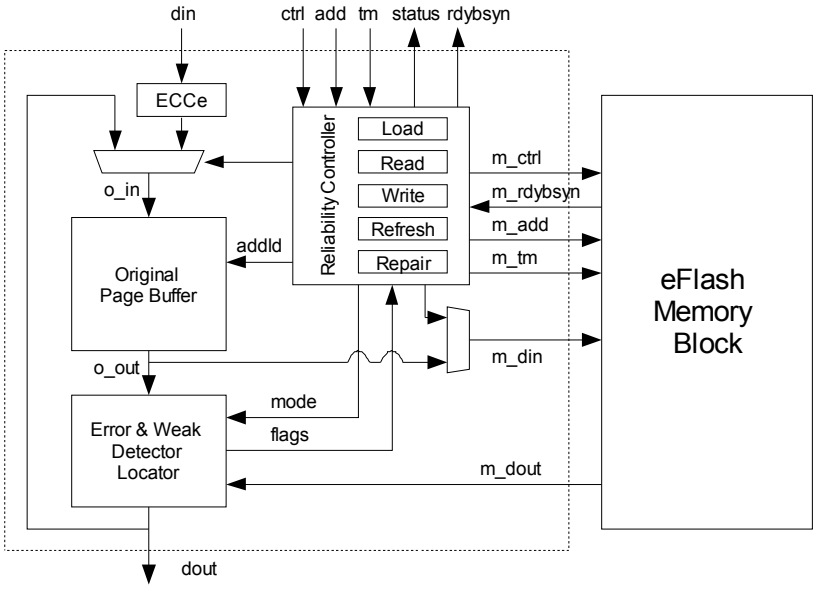

Figure IV.2: Reliability Management Unit

IV.1.ii. Memory for reliability purpose and reliability management unit

Consider the embedded NOR flash memory with following additional features:

- Additional bits on each page to store ECC parities.
- Spare redundancy pages for in application memory repair.

Additionally, assume that a Reliability Management Unit [DDECS07] is appended to the design in order to improve the reliability of the memory. This unit is shown in the Figure IV.2. When an erroneous page is detected, it can perform three reliability functions:

- Correct the page with ECC.
- Refresh the page to a correct content.
- Repair the page with redundancy.

The main problem of the reliability management is to determine when a page must be refreshed or repaired. This problem will be addressed in Section IV.4.

IV.2. Hierarchical error correction coding

IV.2.i. Principle

Since word sizes are relatively small (<128 bits) in NOR flash memories, the choice of the ECC is usually limited to a simple Parity Code or Hamming codes [50,53]. There exist two reasons for that. First, the parity overhead must be kept as small as possible while allowing some reliability improvements. In the Figure

IV.3, we can see parity overhead depending on the information length. When word length is low, parity overhead becomes very important for single error correction, approximately 20% for a 32 bits word length. Secondly, the impact on performance must be reduced to a minimum. If higher correction capabilities are required, performances and silicon area will be far more impacted due to the use of complex ECC such as BCH or Reed Solomon codes [54,55].

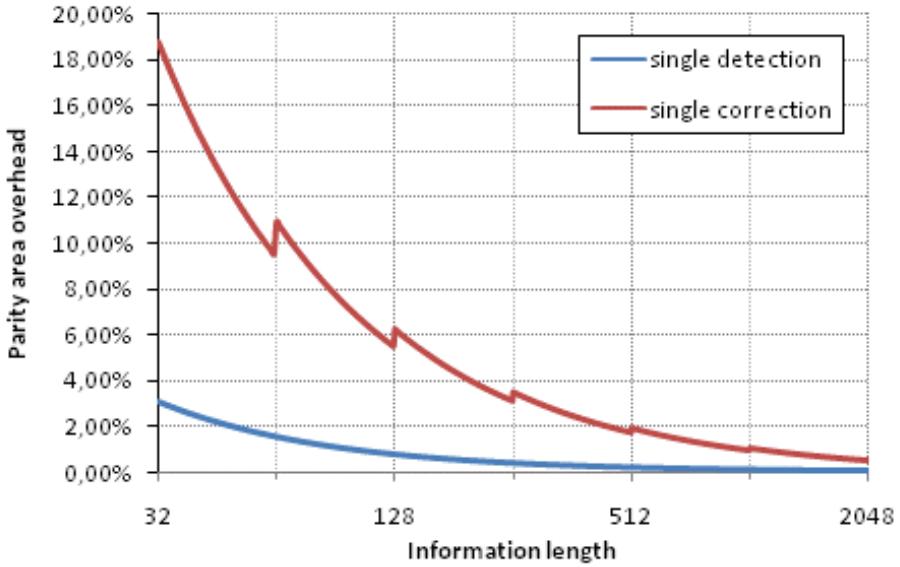

Figure IV.3: Area overhead depending on the information length

In traditional ECC schemes, detection and correction capabilities are used at the word level resulting in the page layout shown in the Figure IV.4. Each word (*W)* has an associated parity called Word Parity (*WP)*. Nevertheless, the number of bits necessary to bring error correction capacity can be very high when word length is low. Operating on larger information words, as shown in the Table IV.1, would allow to reduce the required number of parity bits.

W_0	WP_0	W_1	WP_1	W_2	WP_2	W_3	WP_3

Figure IV.4: Layout of a page with traditional coding scheme

Table IV.1: ECC code and parity overhead

Information word length	Parity word length			
	Parity Code	Hamming Code	Extended Hamming code	Binary BCH Code
MHD	2	3	4	5
DC	1	1	2	2
CC	0	1	1	2
# parity bits (k=# bits/word)	1	$\geq \log_2(k)+1$	$\geq \log_2(k)+2$	$\geq 2\log_2(k)+2$

In order to keep the global parity overhead low while having an improved error correction capacity, we propose to construct hierarchical codes. The underlying concept is to mix detection and correction capabilities through the distinct levels of granularity of a page.

A first code D (word code) with low correction and extended detection capabilities is used at word level to detect errors. When the number of error is too high for the word code D, a second code C (page code) with higher correction capability is used at page level to correct it. The basic layout of a page implementing a hierarchical coding is presented in Figure IV.5. Each word (W) has an associated parity called Word Parity (WP). Moreover, each page has an additional parity called Partial Page Parity (PPP).

Figure IV.5: layout of a page with hierarchical coding scheme

From a functional point of view, when a page is written into the memory, word parities and the partial page parity are computed and inserted into the page. Later, when a word is read, the corresponding word parity bits will be read too. If errors are detected in the word or its parity, additional steps occur. The entire page and the page parity are read back to correct these errors. When the process is done a corrected memory word can be sent to the user.

The main motivations for using hierarchical codes are:

- Working with a large number of bits reduces significantly the number of parity bits required for the same correction level. Putting a high correction capacity on the page rather than on each word is an efficient way to reduce parity overhead.
- The correction scheme for NOR Flash memories should comply with read/ write paradigm to keep memory performances high:
  - Parities must be computed and written with the corresponding user data during the same write operation in the same write element to match the write granularity and guarantee performances.
  - Parities must always be consistent with user data. If a refresh or write operation is triggered, parities must be computed and inserted into the page.

- ❍ As the number of errors is supposed to be low, the error correction process can be performed with only small performance degradations on the mean access time.

- Error correction applied to long memory pages implies a significant area overhead due to the implementation of high speed logic encoder and decoders. It should be reduced.

IV.2.ii. Code construction

Let us take the following conventions: a page is composed of w words and one partial page parity. Each word has n bits composed of k information bits and m_w parity bits (the word parity). The partial page parity has m_p bits. EC and ED are acronyms for Error Correction and Error Detection respectively.

Let e be a positive integer. Our objective is to find the parity generator matrix G of an $(e-1)$-EC e-ED per word and e-EC per page code. In other words, we are looking for a matrix allowing the calculation of word parities and partial page parity such that:

$$\left[WP_0\, WP_1 \cdots WP_{w-1}\, PPP\right] = \left[W_0\, W_1 \cdots \; W_{w-1}\right] \cdot G \qquad \text{(IV.1)}$$

Assume a matrix P with the following properties:

- P is a $k \times (m_w + m_p)$ parity generator matrix that generates a e-EC code.
- P can be decomposed in two sub-matrices D and C of dimension k x m_w and k x m_p respectively such that $P = \left[D \quad C\right]$ and D is the parity generator matrix of a $(e-1)$-EC e-ED code.

From P , a $(k \cdot w)\, x\, (m_w \cdot w + m_p)$ parity generator matrix G with the requested error detection and correction properties is obtained if:

$$G = \begin{bmatrix} D & O & \cdots & O & C \\ O & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & O & \vdots \\ O & \cdots & O & D & C \end{bmatrix} \qquad \text{(IV.2)}$$

Where O corresponds to null matrices of dimension k x m_w. It is very important to notice from Equations (IV.1) and (IV.2) that word parities $WP_0, \ldots, WP_{w-1}$ are computed independently from each other. It allows reading only one word with its associated word parity to correct up to $e-1$ errors and detect e errors. Now, if for

one $i \in [0, w]$, W_i WP_i and PPP have e errors, contribution of corrected words W_j is removed from the read partial page parity by computing $PPP_i = PPP \oplus \sum_{j \in [0,w]/i} W_j \cdot C$. Then, WP_i PPP_i are parity bits of the code P with errors for the word W_i. As P is e error corrector; W_i, WP_i and PPP_i can be corrected.

Practically, to construct G, a P matrix with e-EC properties is first chosen. Next, the main problem is to identify the sub-matrix D in P. However, this process is very easy when $e \in \{1, 2\}$. If e = 1, D is identified by a column vector of 1 corresponding to a standard Parity Code. If e = 2; D must be the parity matrix of an extended Hamming code, so:

- All rows of D must be different
- There must be an odd-number of 1 strictly superior to 1 on each row.

If it is not possible to find a matrix D in P, another P has to be chosen.

For instance, hierarchical codes suitable for a page composed of 4 words of 4 bits ($w = 4$ and $k = 4$) are presented:

$$D = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \tag{IV.3}$$

for $e = 1$ (1ED/word and 1EC/page hierarchical code).

$$D = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{IV.4}$$

for $e = 2$ (1-EC2-ED/word and 2-EC/page).

Then, the general form of the parity generator matrix G is given by:

$$G = \begin{bmatrix} D & 0 & 0 & 0 & C \\ 0 & D & 0 & 0 & C \\ 0 & 0 & D & 0 & C \\ 0 & 0 & 0 & D & C \end{bmatrix} \tag{IV.5}$$

The method to construct this code is reviewed in the section IV.2.iii.

Several hierarchical codes have been constructed for distinct word lengths k. The number of parity bits required has been reported in the Table IV.2 and compared

to other traditional codes. Moreover, the Bit Error Rate (BER) has been calculated and is shown in the Figure IV.6. The hierarchical coding scheme provides an error correction rate similar to the 2-EC scheme at a cost slightly higher than the 1-EC scheme in term of parity overhead.

Table IV.2: Parity overhead comparison (information page length = 1024 bits)

Code	Hamming Code 1EC per word			This work 1EC2ED/W 2EC/P			BCH Code 2EC per word		
# bits per words	m_w	m_p	overhead	m_w	m_p	overhead	m_w	m_p	overhead
32	6	-	+18.7%	7	7	+22.5%	12	-	+38.2%
64	7	-	+10.9%	8	8	+13.3%	14	-	+22.7%
128	8	-	+6.2%	9	9	+7.9%	16	-	+13.4%

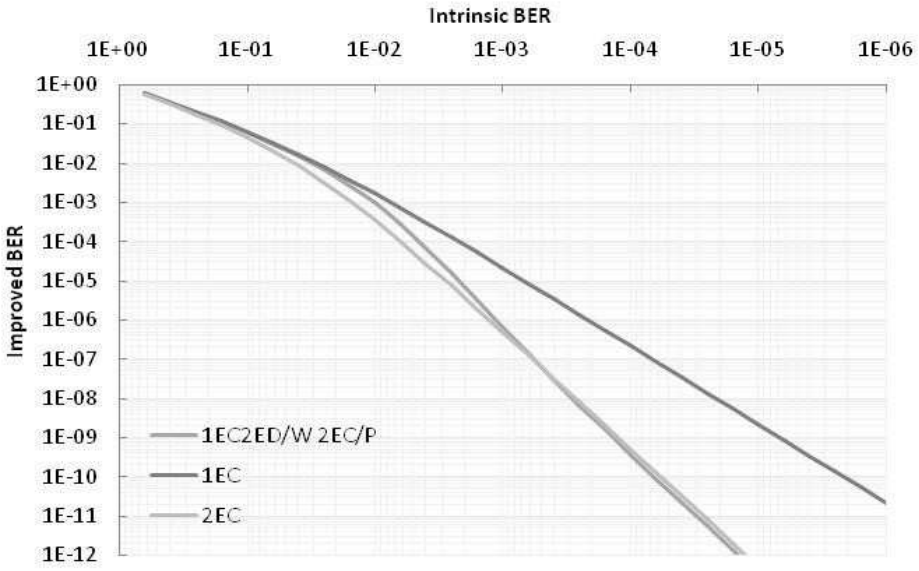

Figure IV.6: Improved Bit Error Rate (BER) comparison (A) Hamming Code (B) Hierarchical Code 1ED/W 1EC/P (C) BCH 2ED (D) HC 1EC-2ED/W 2EC/P

IV.2.iii. Example of generator matrix construction

In this example, we construct a hierarchical code which is a $1EC-2ED$ Extended Hamming code at word level and a $2-EC$ BCH code at page level. Assume that each word is composed of 4 bits. The BCH(31,21) with $t=2$ of generator polynomial $g(x)=1+x^3+x^4+x^9+x^{10}$ is considered. This code is a $2-EC$ code. The corresponding parity matrix G' associated to this code is:

$$P' = \begin{matrix} 1&0&0&1&1&0&0&0&0&1 \\ 1&1&0&1&0&1&0&0&0&1 \\ 1&1&1&1&0&0&1&0&0&1 \\ 1&1&1&0&0&0&0&1&0&1 \\ 1&1&1&0&1&0&0&0&1&1 \\ \mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0} \\ \mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{0} \\ \mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0} \\ 0&0&0&1&1&1&0&1&1&0 \\ 0&0&0&0&1&1&1&0&1&1 \\ \mathbf{1}&\mathbf{0}&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{0} \\ 0&1&0&0&1&1&1&1&1&0 \\ 0&0&1&0&0&1&1&1&1&1 \\ 1&0&0&0&1&0&1&1&1&0 \\ 0&1&0&0&0&1&0&1&1&1 \\ 1&0&1&1&1&0&1&0&1&0 \\ 0&1&0&1&1&1&0&1&0&1 \\ 1&0&1&1&0&1&1&0&1&1 \\ 1&1&0&0&0&0&1&1&0&0 \\ 0&1&1&0&0&0&0&1&1&0 \\ 0&0&1&1&0&0&0&0&1&1 \end{matrix} \quad \text{(IV.6)}$$

By proper lines' reduction, a parity matrix P with following characteristic is obtained:

- P is a $4\times(4+6)$ parity generator matrix that generates a 2-EC code.
- A 4×4 sub-matrix D is the parity generator matrix of a $1-EC\ 2-ED$ code.

$$P = \begin{bmatrix} 1&0&\mathbf{0}&\mathbf{1}&\mathbf{1}&\mathbf{1}&1&1&0&0 \\ 1&1&\mathbf{1}&\mathbf{0}&\mathbf{1}&\mathbf{1}&0&0&0&0 \\ 0&1&\mathbf{1}&\mathbf{1}&\mathbf{0}&\mathbf{1}&1&0&0&0 \\ 0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{0}&1&1&0&0 \end{bmatrix} \quad \text{(IV.7)}$$

The D sub-matrix is equal to:

$$D = \begin{bmatrix} 0&1&1&1 \\ 1&0&1&1 \\ 1&1&0&1 \\ 1&1&1&0 \end{bmatrix} \quad \text{(IV.8)}$$

And the C sub-matrix is:

$$C = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad \text{(IV.9)}$$

As the two last columns are null, corresponding parity bits are always null and can be removed. So the P matrix is:

$$C = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \text{(IV.10)}$$

This process of code reduction can easily be implemented by software. Additionally to optimize the coder/decoder complexity and the amount of logic in each parity tree (one per bits of parity), the number of 1 in the P matrix can be made equal and minimized [56].

IV.3. Coder/Decoder architecture

In this part, an example of architecture used for hierarchical error correction is presented. This architecture is a main contribution of this thesis and has been patented in [USPT08]. It is adapted to a hierarchical code where the word code is a Parity Code allowing 1ED and the Page Code is a Hamming code allowing 1EC.

IV.3.i. Encoder architecture and process

A schematic view of the encoding architecture is presented in the Figure IV.7. Writing a page into the memory is decomposed into three steps:

- loading operation
- encoding operation
- writing operation

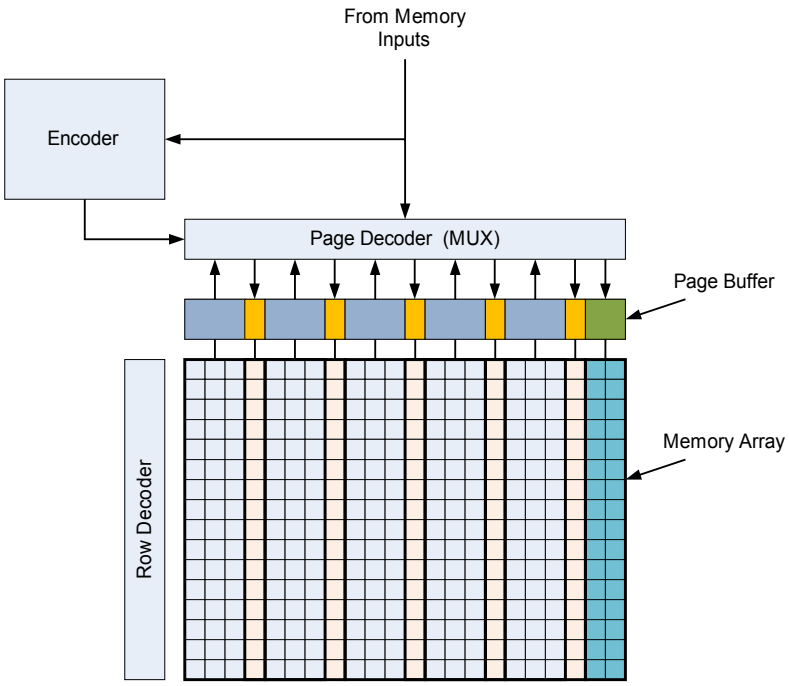

Figure IV.7: Memory encoding architecture

IV.3.i.1. Loading operation

During the load operation, user words are sequentially inserted into *Word Buffers* of the *Page Buffer*. The *Page Buffer* is shown in the Figure IV.8 and is decomposed in a *Word Buffer*, a *Word Parity Buffer* and a *Page Parity Buffer*.

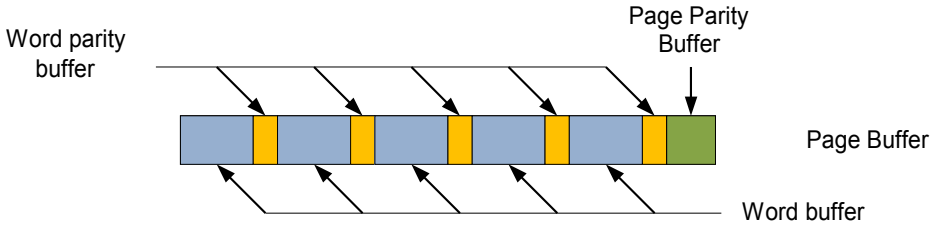

Figure IV.8: Page buffer

IV.3.i.2. Encoding operation

After the loading operation, a command to perform the write operation is launched. It triggers a first encoding step. *Word Parity* and *Page Parity* are computed and inserted into the corresponding buffers of the *Page Buffer*. A detailed description of the *Encoder* is illustrated in the Figure IV.9. It is composed of a *Parity Encoder Controller* and a *Parity Encoder*. The main function of the *Parity Encoder Controller* is to reset registers and sequence words in the parity encoder.

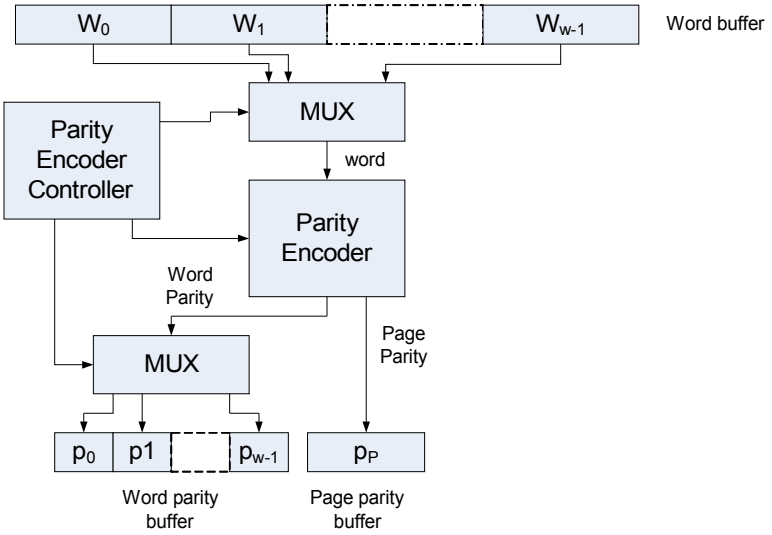

Figure IV.9: Parity encoder and controller architecture

For each word, the *Word Parity* bit is computed and inserted in the corresponding *Word Parity Buffer*. In parallel, *Page Parity* is sequentially computed. When all words have been processed into the *Parity Encoder*, the *Page Parity* is available at the output and is inserted in the *Page Parity Buffer*. When all parity are computed and inserted into the *Page Buffer*, the next operation begins.

The *Parity Encoder* structure is shown in the Figure IV.10. It is the functional part the G_P generator matrix encoding. The Parity Encoder is composed of two branches, one for the Word Parity calculation and another for the Page Parity calculation. The first branch processes a word of k bits and provides an output on q bits. The second branch sequentially processes words of k bits to provide an output on p bits.

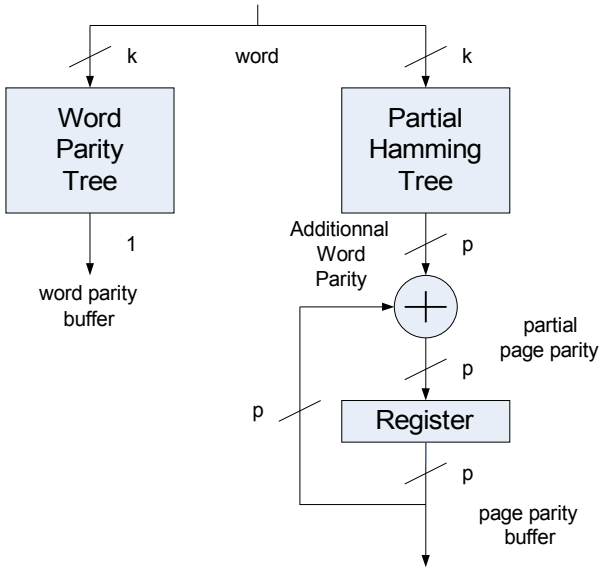

Figure IV.10: Parity encoder structure

IV.3.i.3. Writing operation

The content of the *Page Buffer* is programmed into the memory at the specified address location.

IV.3.ii. Decoder architecture and process

To manage and keep the page integrity, an *Error Decoder* structure is inserted between *Sense Amplifiers* and memory outputs as illustrated in the Figure IV.11.

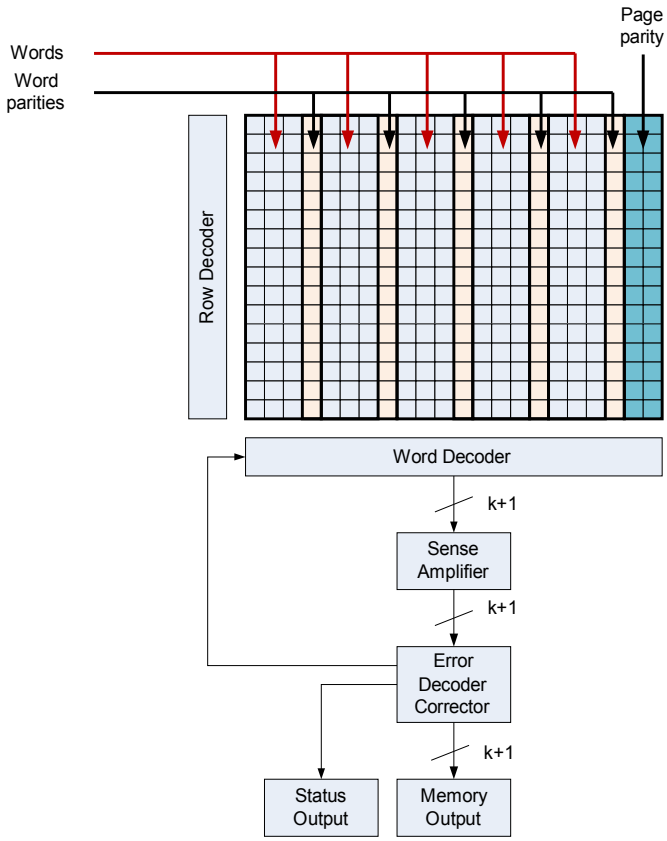

Figure IV.11: Memory decoding architecture

The *Error Decoder* structure is presented in the Figure IV.12 and is composed of the following blocks:

- *Registers* for storing syndrome, read word and corrected word.
- *Syndrome Register* ($m+q$ bits) stores the result of the syndrome computation.
- *Read Word Register* ($k+m+q$ bits) stores the word that user wants to read.
- *Corrected Word Register* ($k+m+q$ bits) stores the result of the correction process.

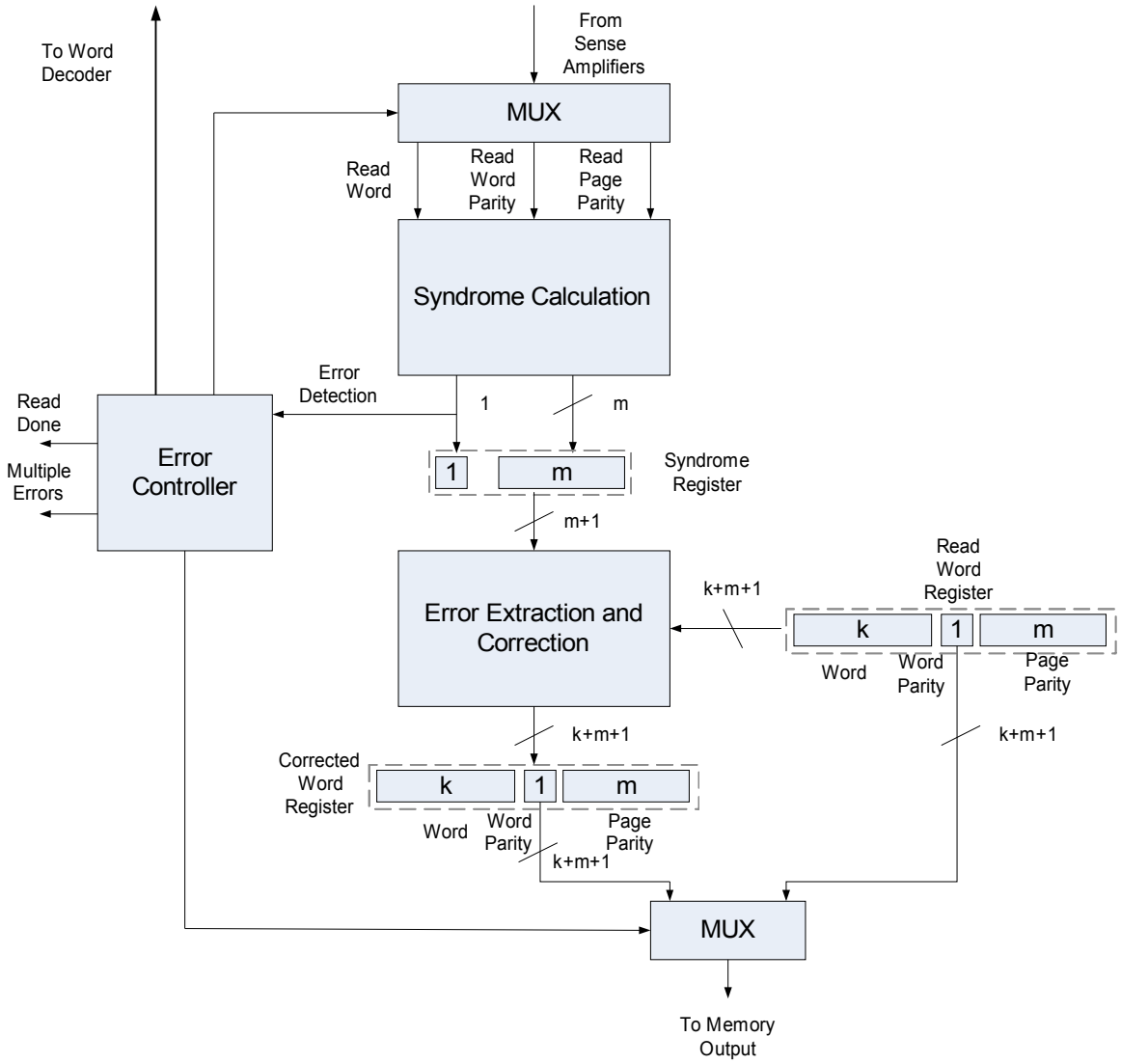

Figure IV.12: Error decoder structure

The *syndrome calculator* computes a syndrome of $m+q$ bits. q first bits of the syndrome provide the result of the word currently accessed. The m other bits provide the rest of the syndrome once the entire page has been read during a correction process.

***Error Extraction and Correction* is used** during the correction process. When the page has been re-read and the syndrome is computed, this block finds the error pattern and corrects the *Read Word Register*. The result of the correction is placed in the *Corrected Word Register*.

Additionally, two multiplexers are used in this structure. An *Input Multiplexer* selects destination of the data coming from the sense amplifier. An *Output*

Multiplexer selects the *Read Word Register* or the *Corrected Word Register* depending on if the word has been corrected or not.

Finally, an *Error Controller s*equences the operation during error correction process by

- Checking that only one error or no errors are detected in a page, else sends an Uncorrectable Error to the user.
- Resetting all the registers of the decoder.
- Informing user that the read operation is done (*Read Done*) or that too much errors are present in the page so it cannot be corrected (*Multiple Errrors*).

A detailed description of the *Syndrome Calculator* is provided in the Figure IV.13. It is composed of two branches in the same manner as the *Parity Encoder:*

- The first branch computes the *Error Detection* signal for a word. The *Word Parity* bit is recomputed from the *Read Word* thanks to the *Word Parity Tree*. Next, this *recomputed Word Parity* bit is "XORed" with the *Read Word Parity* to generate the *Error Detection* signal. If the *Error Detection* signal is 0, there is no error in the word, else it is 1, and there is an error in the word.
- The second branch computes the p last bits of the *Syndrome*. It first recomputes the *Page Parity* by reading sequentially all the memory words of the page (same process as the *Encoding*). When all words are processed, the recomputed *Page Parity* is present in the p bits of the *Recomputed Page Parity Buffer*. Secondly, the *Page Parity* is stored into the memory, then it is read and "XORed" with the content of the register to provide the p last bits of the *Syndrome*.

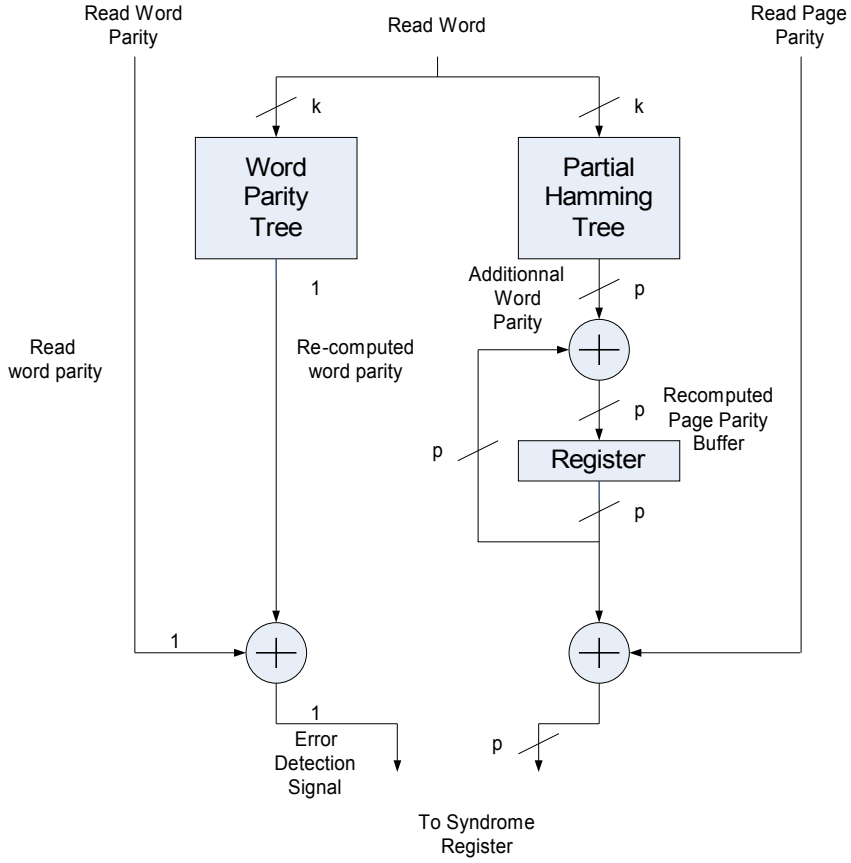

Figure IV.13: Syndrome calculator architecture

Each time a word is read, the first branch is used to detect error in memory words and to signal it to the *Error Controller* via the *Error Detection* signal. The second branch is only used for partial *Syndrome* computation when error correction is necessary.

When the *Syndrome* has been computed and latched into the *Syndrome Register*, it is used in the *Error Extraction and Correcting* block detailed in the Figure IV.12. An implementation example of such a block is presented in the Figure IV.14. This implementation is classical. It is composed of two blocks; an *Error Extraction* block and an *Error Correction* block:

- The *Error Extraction* is a combinatorial tree or a kind of Look-Up-Table. The combinatorial tree is the implementation of the matrix [P H]. Depending on the value of the *Syndrome* it returns the corresponding *Error Pattern*. In fact, the size of this combinatorial tree depends only on the word length, not on the page length as in a classical implementation of one page error correction. It is due to the repeated structure of H and P in the *Generator Matrix*, so logic overhead due to decoder is reduced.

- The *Error Correction* block (bitwise XOR) computes the corrected word by XORing the *Read Word* with the *Error Pattern*. The result is sent to the *Corrected Word Register*.

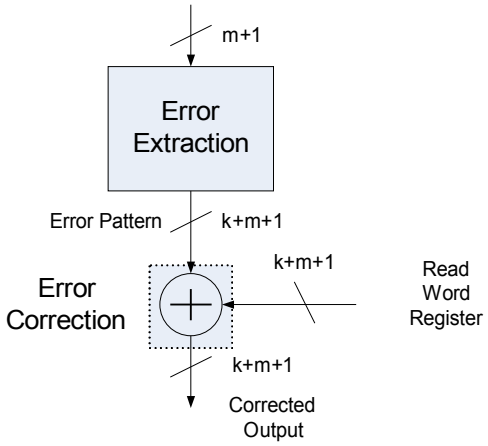

Figure IV.14: Error extraction and correction block

This part describes the sequence of operations for error correction shown in the Figure IV.15.

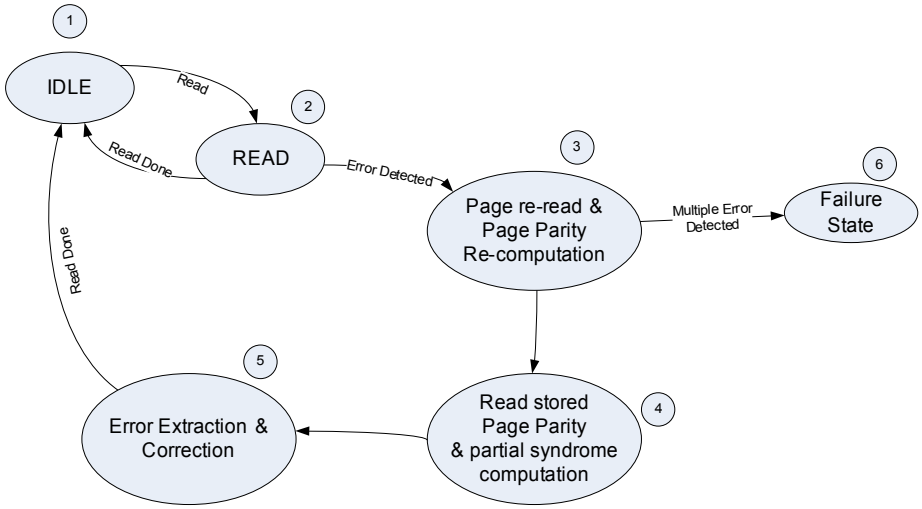

Figure IV.15: Operation for error correction

Assume the memory is in IDLE state. User specifies the address of a word to read and launches the memory operation.

i. The memory enters in read mode. The *Read Word* and *Read Parity Word* are read from the memory and stored into the *Read Word Register* Figure IV.12. In the same clock cycle, the *Read Word* and the *Read Parity Word* are processed by the first branch of the *Syndrome Calculator* block to

determine the *Error Detection* signal. If *Error Detection* signal is 0, the read done is signaled by the *Error Controller* and it outputs the *Read Word Register*.

ii. If *Error Detection* signal is 1, an error has been detected and the read done signal stays low. The value of the *Error Detection* signal is stored into the first bit of the *Syndrome Register*.

iii. Next, the page is re-read word per word. Words are sent to the *Syndrome Calculator* block. It re-computes the *Page Parity* and stores it into the *Recomputed Page Parity Buffer* Figure IV.13.

iv. After that, the *Page Parity* is read from the memory and "XORed" with the *Recomputed Page Parity Buffer* to provide the last p bits of the *Syndrome*. These p bits are stored into the *Syndrome Register*. In parallel, the *Page Parity* read from the memory is stored in the *Read Word Register*.

v. Next, the *Syndrome* is automatically sent to the *Error Extraction and Correction* block. It corrects the *Read Word Register* from the error and sends the result to the *Corrected Word Register*. The *Error Controller* selects the *Corrected Word Register* and sent it to the output of the memory. The read done signal is set high.

vi. During the state 3, if an error is detected in another word of the page, there are at least two errors in it. The error correction process cannot handle it. So, the *Multiple Error* signal is set high and the memory enters in a *Failure State*. The *Error Controller* sets the *Output Multiplexer* in a way that the *Read Word Register* is sent to the output of the memory.

IV.4. Reliability modeling

In this part, we propose to develop a reliability modeling of a memory with hierarchical coding. This contribution is based on a Markov process modeling which is a convenient method to perform complex reliability studies.

IV.4.i. Issue of the reliability management

Let us consider the Reliability Management Unit (RMU) that was presented in the Section IV.1.ii. Periodically, this unit counts for the number of errors in a page. This period denoted T_{sc} is called scrubbing period.

When errors are detected on a page, the objective of the Reliability Management Unit is to determine if a page should be leaved, refreshed or repaired. The use of the hierarchical correcting scheme with $e=2$ can help to make a choice among these techniques. The following reliability policy is defined:

- When a word has a double error, errors can be corrected by the page code. But, the page reliability becomes critical: a third error on this word would result in an unrecoverable memory failure. Consequently, when such a state is detected, the entire page should be replaced by one page redundancy.
- When multiple words have a single error, they are corrected by the word code. However, when a given number of single errors are detected in a page, a page refresh can be triggered. This limit is called refresh level and denoted *r*. After a refresh, the memory page returns to a failure free state.

IV.4.ii. Markov process modeling for a page

The behavior of the memory with RMU can be modeled by a Markov process [57]. This modeling is very well suited for determining the reliability of dynamic systems that can be reconfigured depending on a temporal evolution. In particular, we can use a Continuous Time Markov Chain (CTMC) for modeling our hierarchical code.

CTMC is a practical tool to describe a process without memory [58]. A CTMC is a continuous time stochastic process $\{X_t\}_{t\geq 0}$ with a state space S which satisfies the **Markov property** :

$$P\{X_{t+s}=j \mid X_s=i,\ X_u=i_u, 0\leq u<s\}=P\{X_{t+s}=j \mid X_s=i\} \quad \text{(IV.11)}$$

For all s , $t\geqslant 0$ and i , j , $i_u\in S$ and $0\leqslant u<s$.

Additionally, they are usually chosen time-homogeneous by adding the following property:

$$P\{X_{t+s}=j \mid X_s=i\} \quad \text{(IV.12)}$$

is independent of *s*.

In other ways, for any arbitrarily chosen time point *t*, the evolution of the process after *t* depends on *t* and the state occupied at *t*, but **not on the process evolution up to the time *t***. In the case of time-homogeneous Markov processes, dependence on *t* also disappears such that future evolution of the process depends only on the current state, but **not on the time elapsed.** In reliability theory, these processes

describe the behavior of repairable systems with constant failure and repair rates for all elements which is perfectly adapted to our study.

In this reliability study, one should notice that two assumptions have been made to facilitate mathematical modeling and computation:

- First, the Markov process is supposed Time Homogeneous [29]: transition rates between states are constant as a function of the time. This strong assumption is usual in reliability analysis even though it does not fit exactly the floating gate purpose where the failure rate depends on the number of write/erase cycles and time since the last programming operation [51].
- The second assumption simplifies the state modeling: we consider that a page is composed of words and associated word parities on which errors can occur. The partial page parity is not taken into account to reduce the number of states in the Markov Process.

The Continuous Time Markov Chain (CTMC) modeling of the proposed hierarchical code for one page is depicted in Figure IV.16. Each state $S_{i,j}$ of this graph is uniquely identified at time t by a couple (i, j) where, i is the number of words with a single error in the page and j is the number of words with double error in the page. Additionally, we introduce states $S_{i,1D}$ in which one unique double error in a word has been detected. The page is in ***unrecoverable error states*** if in either the F ***Failure state*** or the F_D ***Failure Detected state***. That is to say, at least two words have 2 errors or one word has 3 errors. States have been classified following six levels of criticality (S – **Safe**, M - **Medium**, C – **Critical**, C_D - **Critical Detected**, F - **Failed**, F_D – **Failed Detected**).

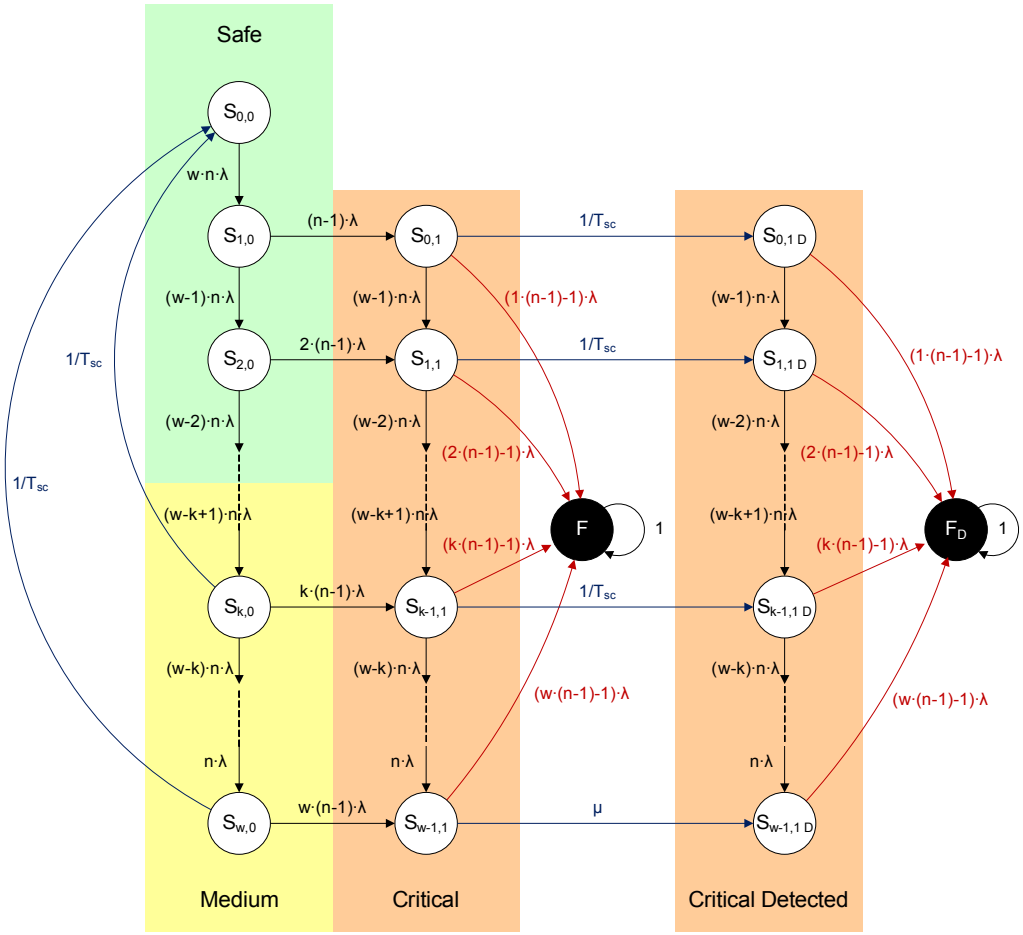

Figure IV.16: Markov modeling of the hierarchical coding scheme for a page

Considering λ, the failure rate of a memory, and T_{SC} the scrubbing operation period, following transition rates are defined:

- $(w-i)\cdot n\cdot\lambda$ for $0\leq i\leq(w-1)$ when the page transits from states $S_{i,0}$ to $S_{i+1,0}$. One more word has a single error but no word has double errors.
- $i\cdot(n-1)\cdot\lambda$ for $1\leqslant i\leqslant w$ when the page transits from states $S_{i,0}$ *to* $S_{i-1,1}$. One word has a double error while others have single or no error.
- $(w-i-1)\cdot n\cdot\lambda$ for $0\leq i\leq(w-2)$ when the page transits from states $S_{i,1}$ to $S_{i+1,1}$ or $S_{i,1D}$ to $S_{i+1,1D}$. One more word has a single error and a single word has double error.
- $((i-1)\cdot(n-1)-1)\cdot\lambda$ for $0\leq i\leq(w-1)$ when the page transits from states $S_{i,1}$ to F or $S_{i,1D}$ to F_D. At least, two words have two errors or one word has a triple error. In that case, hierarchical code cannot handle that.

Moreover, additional transition rates serve to describe the behavior of the Reliability Management Unit during refresh and repair operations:

- $\frac{1}{T_{SC}}$ for $0 \le i \le (w-1)$ when the page transits from a non-detected double error state $S_{i,1}$ to the corresponding detected double error state $S_{i,1D}$. These transitions will be used to trigger a repair with page redundancy.
- $\frac{1}{T_{SC}}$ for $0 \le i \le (w-1)$ when the page transits from a Medium criticality state $S_{i,0}$ ($r \le i \le w$) to the failure free state $S_{0,0}$.

The graph of Figure IV.16 can be expressed as a system of differential equations. Assume that each state is renamed with following notations:

- $S_{i,0} = S_{(i)}$ for $0 \le i \le w$.
- $S_{i,1} = S_{(w+i+1)}$ for $0 \le i \le w-1$.
- $S_{i,1D} = S_{(2 \cdot w+i+1)}$ for $0 \le i \le w-1$.
- $F = S_{(3 \cdot w+1)}$ and $F_D = S_{(3 \cdot w+2)}$.

The vector $P\left[S_{(0)}(t) S_{(1)}(t) \ldots S_{(3 \cdot w+2)}(t)\right]$ defines the probability to be in states $S_i(t)$, $0 \le i \le 3w+1$ at time t. The system of differential equations is given by:

$$\frac{d\,\boldsymbol{P}(t)}{dt} = \boldsymbol{M} \cdot \boldsymbol{P}(t) \qquad \text{(IV.13)}$$

$\boldsymbol{M}$ is the transition matrix composed of transition rates defined by the graph of Figure IV.16. An element of $\boldsymbol{M}$, $m_{i,j}$ with $i \neq j$, represents the rate related to the probability to transit from the state i to the state j. Additionally, an element of $\boldsymbol{M}$, $m_{i,j}$ with $i = j$ represents the rate related to the probability to stay in the state i. $m_{i,j}$ is defined as $-\sum_{i \neq j} m_{i,j}$. To determine $\boldsymbol{P}(t)$, the system of equation (IV.13) is solved numerically.

IV.4.iii. Reliability of an array with redundancy

Knowing the state probabilities' vector for a page $\boldsymbol{P}(t)$, we can compute the reliability of an array with redundancy. If the memory array is composed of n_p

page and n_{sp} spare page for redundancy repair, the array reliability is given by the combinatorial probability:

$$R(t) = \sum_{i=0}^{n_{sp}} \binom{i}{n_{sp}+n_p} \left(1 - P_{F_D}(t) - P_F(t)\right)^{(n_p+n_{sp}-i)} \left(P_{F_D}(t)\right)^i \quad \text{(IV.14)}$$

Where P_{F_D} and P_F are probabilities to be in states F_D and F respectively.

IV.5. Results and discussion

In this part, a 1Mbits memory array with following structure is considered: $n_p = 1024$, $k = 128$, $n_w = 7$, $n_p = 7$. r, n_{sp} and T_{sc} are variables. To illustrate the efficiency of the proposed scheme, a high failure rate λ = 5.6 FIT per bit (Failure In Time = Errors / 10^9 h) is assumed. This failure rate is related to devices working at extreme conditions of temperature. In that case, charge losses mechanisms resulting in retention errors are accelerated and in consequence the reliability of each cell is highly jeopardized [51].

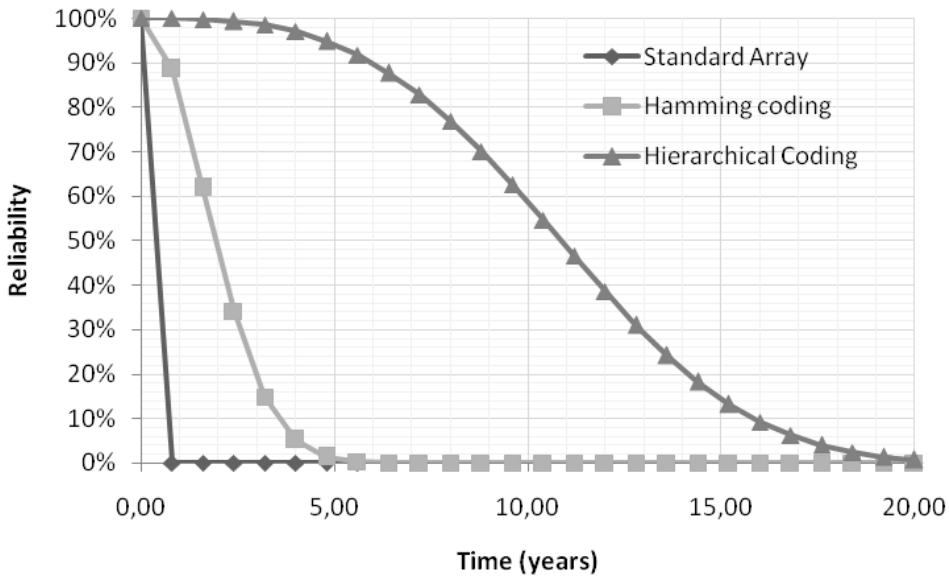

Figure IV.17: Reliability comparison between a standard array without any correction and arrays with Hamming coding or Hierarchical coding. (1Mbits, nsp= 0, 1/Tsc = 0, r = 0)

Figure IV.17 shows a fast decrease of the memory reliability in case of a standard array without any correction scheme or with a *1*-EC Hamming coding. On the contrary, reliability is largely improved when considering the hierarchical coding scheme. Even if the refresh and repair processes have not been activated ($1/T_{sc}$ = 0), the Mean Time To Failure (MTTF) reaches 10.9 years in comparison with the 2.1 years of the standard Hamming scheme.

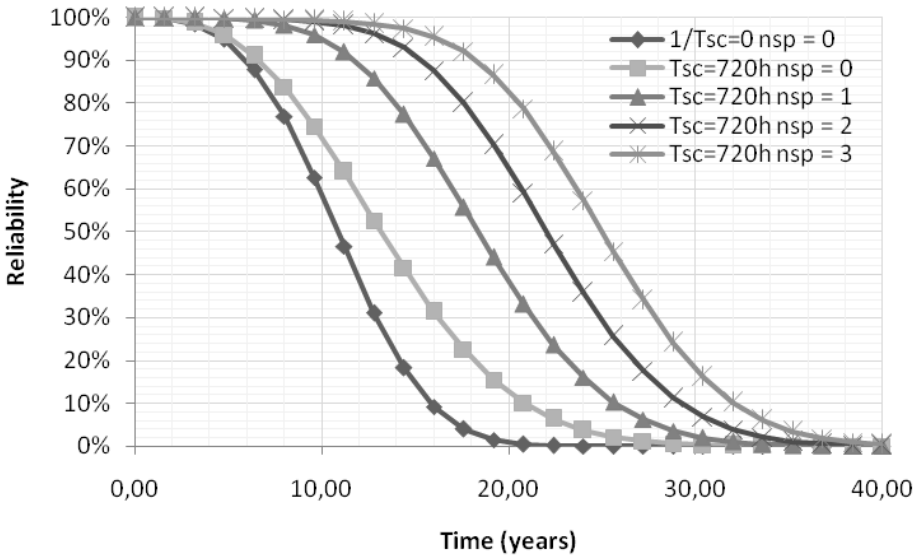

Figure IV.18: Impact of the page redundancy repair on the reliability (1Mbits, r = 2)

Figure IV.18 presents the reliability evolution of memories with Refresh and Repair processes engaged or not. If engaged, the scrubbing period T_{sc} has been fixed to 720 hours. The number of available pages for redundancy n_{sp} has also been modulated. The scrubbing allows a refresh of pages if they have more than 2 words with a single error ($r = 2$). The effect of this process is noticeable and the MTTF is improved by +24%. In that case, pages in a Medium criticality state are put back in the error free state. However, this behavior is limited to pages with single error words. Indeed, for pages with one double error word, it does not exist a refresh transition. The Markov modeling does not take into account the refresh of these pages when no redundancy is available. However, this behavior could be easily defined in the Reliability Management Unit and added to the Markov modeling.

Additionally, the effect of page redundancy is very important on reliability. MTTF improvements are +37.3%, +63.7%, +84.9% with 1, 2 and 3 pages redundancy in comparison with no redundancy case. Improvements increase slowly with higher number of redundancy pages.

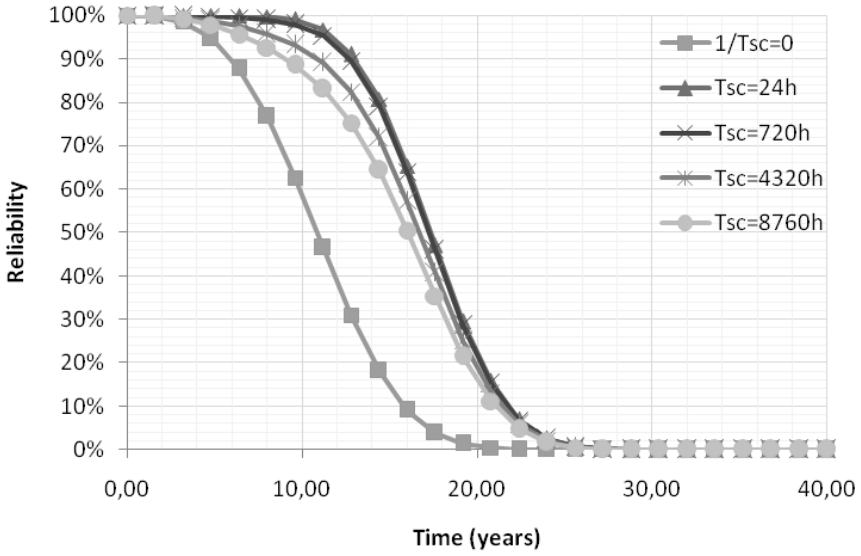

Figure IV.19: Impact of the scrubbing period on the reliability (1Mbits, r=4, nsp=2)

Figure IV.19 shows the impact of the scrubbing period on the reliability. This period corresponds to 1 day, 1 month, 6 months, 1 year or an infinite time. Even for a relaxed scrubbing period, improvements on MTTF are large. For instance, a 1 year period results in +42% of MTTF.

Impact on performances can also be predicted. When a word with double error is read, the access time of the memory is degraded. This delay is due to additional steps required to read back the content of a page plus one additional clock cycle for the page correction process. Let t_{acc} be the word access time, then the access time for double word error correction will be $t'_{acc} = t_{acc} \cdot (n_w + 1)$. The mean access time is defined by:

$$t'_{acc} \cdot (P_{C_D} + P_C) + t_{acc} \cdot (P_S + P_M) \quad \text{(IV.15)}$$

Where P_{C_D}, P_C, P_S and P_M are probabilities to be in Critical Detected, Critical, Safe and Medium criticality states respectively. If t_{acc} is *25ns* then t'_{acc} is *225ns*. The Figure IV.20 shows the mean access time evolution depending on the refresh level *r*. It can be compared with the evolution of the reliability presented in the Figure IV.21. Mean access time grows linearly after 10 years while reliability characteristic begins to decrease. With r = 2, the best reliability characteristic and mean access time are reached.

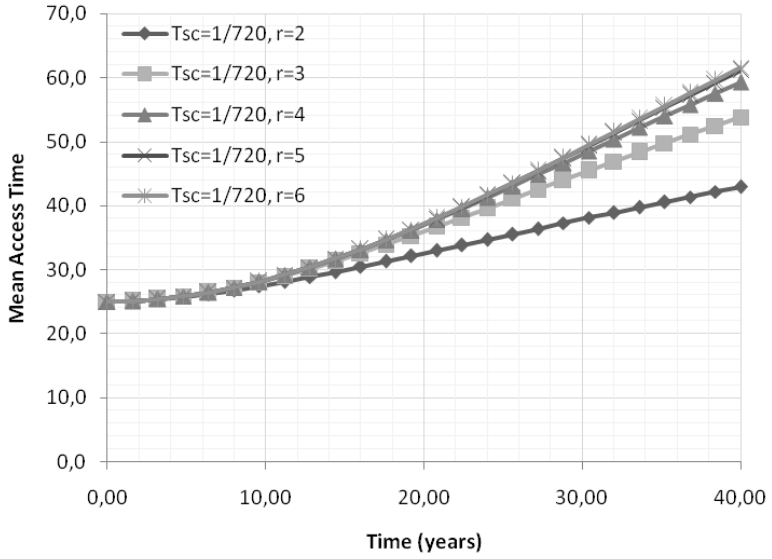

Figure IV.20: Mean access time evolution depending on the refresh level r

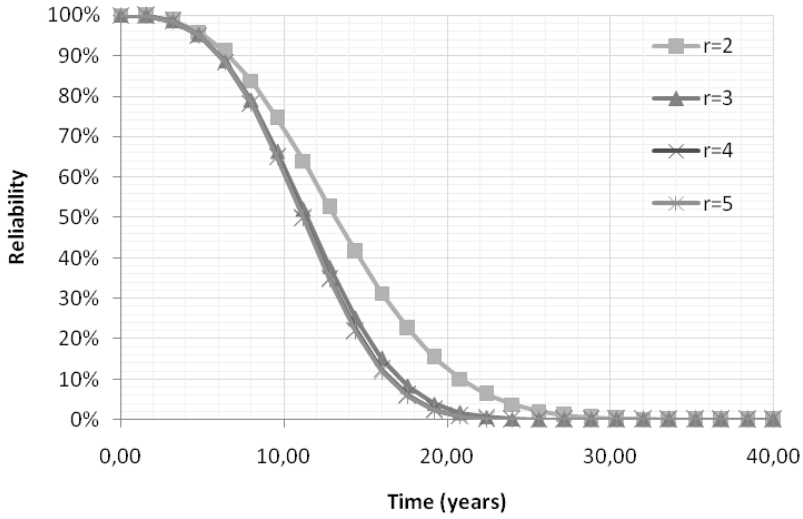

Figure IV.21: Impact of the refresh level on the reliability (1Mbits, nsp = 0, Tsc = 720h)

In Table IV.3 and Table IV.4, we have estimated the total overhead for memories using hierarchical codes and implementing the RMU and we compared it with some classical codes (Parity Code, Standard Hamming code and BCH 2EC code) for distinct word lengths.

In Table IV.3, each information word has a size of 32-bits. In that case, we can notice that the hierarchical code allowing to correct up to 1 bit per page (*e=1*) has an overhead around 3% which is largely smaller than the overhead of an Hamming code which allows 1 error correction per word. In fact, with 3.3%, the

overhead of the hierarchical code (*e=1*) is very close to the overhead with a Parity code (2.4%) when memory size is 4Mbits.

The hierarchical code (*e=2*) allows to correct 1 error per word and 2 error per page. The total overhead of this scheme is intermediate between the overhead of an Hamming code and the overhead of a BCH code whatever the memory size is. However, when the memory size becomes larger (4096Kbits), the hierarchical code overhead (18.1%) is closer to the overhead of an Hamming code (14.4%) than to the overhead of a BCH code (28.8%). As illustrated in Table IV.5, this trend is explained by the fact that for large eFlash memories, the part represented by the array is larger than the part due to the periphery and, by consequence, the total overhead is mainly due to parity bits storage.

Table IV.3: Total memory overhead depending on error correction scheme (32 bits per word)

	512Kbits	1024Kbits	2048Kbits	4096Kbits
Memory parameters	32 bits per word, 1024 bits/page, no redundancy			
Parity Code	1.3%	1.6%	1.9%	2.4%
Hierarchical e=1	3.7%	3.0%	3.0%	3.3%
Hamming	9.0%	10.3%	11.9%	14.4%
Hierarchical e=2	14.3%	13.9%	15.3%	18.1%
BCH 2EC	18.0%	20.5%	23.8%	28.8%

In Table IV.4, each information word has a size of 128 bits. We can notice that total overhead for the hierarchical code (*e=2*) is larger than the overhead of BCH 2EC for small memory size. In fact, this is mainly due to the logic necessary to perform coding/decoding operations. In fact, the hierarchical decoder (*e=2*) integrates an extended Hamming decoder and a BCH 2EC decoder. But, when the word length increases, the number of gates of these two decoders grows drastically and becomes non-negligible compared to the overhead due to parity bits. However, as soon as, the array part becomes more important, parity bits overhead outweighs the coder/decoder logic and hierarchical code (*e=2*) is better than the standard BCH 2EC.

For the hierarchical code (*e=1*), we can notice the same trend: the logic encoder/decoder are non-negligible for small memories. However, the overhead due to this part is rapidly masked by parity bits overhead as soon as array density exceeds 1024Kbits. This scheme is still efficient compared to the Hamming code.

In conclusion, the hierarchical code can significantly reduce the total overhead for memories with small word lengths (32 bits) compared to traditional codes whatever the memory size is. However, for memories with larger word lengths, the size of the array must be sufficiently important in comparison with the periphery to justify the use of the hierarchical code over a traditional code (Hamming, BCH).

Table IV.4: Total memory overhead depending on error correction scheme (128 bits per word)

	512Kbits	1024Kbits	2048Kbits	4096Kbits
Memory parameters	128 bits per word, 1024 bits per page, no redundancy			
Parity Code	0.6%	0.5%	0.6%	0.7%
Hierarchical e=1	7.7%	4.1%	3.1%	2.8%
Hamming	8.0%	5.7%	5.5%	5.9%
Hierarchical e=2	21.2%	12.4%	10.2%	9.9%
BCH 2EC	16.0%	11.5%	10.9%	11.8%

Table IV.5: Repartition between array and periphery in eFlash memories for various memory sizes

	512Kbits	1024Kbits	2048Kbits	4096Kbits
Periphery	60%	49%	39%	25%
Array	40%	51%	61%	75%

An estimation of the total overhead for 2Mbits arrays with a varying number of page for redundancy is proposed in the Table IV.6. As we can see, the total overhead evolves quasi-linearly in function of the number of row redundancy. Referring to Figure IV.18, it is fully justified to add few rows to improve the MTTF of memories whereas the impact on overhead is very low.

Table IV.6: Total overhead for 2Mbits arrays with varying number of page redundancy

	2Mbits array with 1024 bits per page							
Word length	32				128			
n_{sp}	0	2	4	6	0	2	4	6
Hierarchical code *e=1*	3.0%	3.2%	3.5%	3.7%	3.1%	3.3%	3.6%	3.8%
Hierarchical code *e=2*	15.3%	15.6%	15.8%	16.0%	10.2%	10.5%	10.7%	10.9%

IV.6. Conclusion

The objective of this chapter was to describe a second reliability management scheme for NOR embedded flash memories. The hierarchical coding scheme has been used as a way to improve the built-in self reliability management strategy. This technique uses the fact that Flash memories do not operate on the same number of bits during read and during program operations. In consequence, correction capacities can be distributed in an original way into the memory array to significantly reduce the cost usually associated with advanced error correction techniques.

This code has also been integrated in a reliability architecture having some redundancy resources. The reliability management policy considers that memory pages can be refreshed or repaired with redundancy periodically depending on the number of errors in it. A mathematical study based on Continuous Time Markov Chain has allowed to demonstrate the effectiveness of this structure. For instance, in case of a hierarchical ECC allowing 1 EC per word and 2 EC per page compared to a standard ECC allowing only 1 EC per word, the MTTF has been improved by a factor of 5.2. In addition, the integration of some row redundancy can substantially improve the MTTF of the memory in case of hierarchical ECC by many factors.

The ECC technique presented along this chapter has some similarities with the one developed in chapter III: an ECC with extended detection capacity is used in combination with a method to provide an additional error correction capacity. From a reliability point of view, the efficiency of Hierarchical ECC is better because all cells are read with the same biasing conditions and error extraction is uniquely based on mathematical properties of the code. On the contrary, in V_T error correction, there are multiple biasing conditions that must be adjusted conveniently to locate weak cells. Additionally, the assumption that a weak cell is likely to be in error is true but can result into mistakes. From an architecture point of view, Hierarchical ECC reduces modifications of the memory design because all required blocks such as decoders/encoders can be integrated outside the memory. However, in term of area overhead, a high order decoder will induce a higher area consumption compared to V_T error correction for the same level of correction. To conclude, hierarchical ECC is a good solution at high orders for products requiring high reliability rates. On the contrary, V_T error correction is a good solutions at low order for low cost products requiring low improvements.

General Conclusion

Different types of memories can be embedded in a SoC as SRAM, DRAM, EEPROM and Flash. In recent years, the increased use of portable electronic devices has produced a high demand for embedded Flash memories. Their advantages are to exhibit low power characteristics with some security features (lock bits). In addition, these memories allow In Situ Programming, resulting in very flexible solutions for code development and updates.

In parallel with the recent market evolution, SoCs with embedded memories are facing technological issues due to reliability and chip yield. As an increasing silicon area is dedicated to storage elements, memory reliability is becoming the main detractor of the SoCs reliability. Additionally, the diversity of applications covered by memory now integrates automotive, aeronautic and biomedical. For all these reasons, memory designs should be tailored to fit reliability needs of the targeted application while staying still cost competitive.

In this work, we have seen that eFlash memory where based on the floating gate principle on which charges can be stored and removed under the application of an high electrical field. Usually in embedded technologies, the Fowler-Nordheim (FN) tunneling is preferred for both erase and write operation in order to improve cycling capabilities and minimize current consumption. Nevertheless, with time and cycling, memory cell characteristics evolve mainly resulting in two kinds of reliability issues: retention and endurance. The endurance is the ability of the cell to endure write/erase operations keeping good electrical characteristics. The retention is the ability of the cell to keep information. Additionally, for some bits, the reliability characteristic may be naturally degraded resulting in errors randomly localized in the array. In particular, this is the case of fast moving bits and erratic behavior bits. The solution is then to integrate fault tolerance techniques.

Our thesis work took place in this context. In a first part, we studied embedded memories, Flash and their particularities resulting in Chapter I. Then, we focused on the reliability field and fault tolerance techniques which are usually employed in semiconductor memories. Thanks to this background, we established an analytical reliability model described in Chapter II. Next, the work was focused on

fault tolerance methods to improve memory reliability shown in Chapter III and IV.

The objective of a compact reliability model was to provide a generic equation for the reliability of a Flash memory cell. This work was based on the model developed by [28]. In fact, the resulting model has the characteristic of a Weibull distribution which usually traduce the wear-out of a device with time. The advantage of the developed model lies in its ability to traduce not only a temporal evolution but some other parameters evolution such as a number of cycles. All parameters of the model were calibrated with Atmel data.

The second step of the work was to look after fault tolerance techniques and reliability management scheme which could help in improving memory reliability while keeping cost as low as possible. In order to improve error correction, two ECC techniques were proposed: the Error Correction with V_T analysis and the hierarchical error correction. Large reliability enhancements have been reached with these two techniques.

The Error correction with V_T analysis [DATE07] has been described in Chapter III. It is a method which uses an ECC with a system that can locate bits storing weak logic values. If the ECC detect errors, the method consists in considering that weak bit positions are very likely in error and should be corrected by flipping the bit content. For instance with an ECC based on Parity Code (1 parity bit) allowing to detect one error, this scheme allows up to 1 error correction. The effectiveness of this method was studied with the flash reliability model established in Chapter II. In term of results, it could improve the memory Mean Time To Failure (MTTF) by a factor 4.67 in a 2Mbits array with 32 bits per word under given conditions.

The Error Correction with V_T analysis was also integrated in an architecture that could perform some reliability management [DDECS07]. The purpose of this scheme is to replace erroneous page by redundant page as soon as an error is detected. In such a configuration, large reliability enhancement are possible. For instance, considering a memory using the Error Correction with V_T analysis to correct up to 2 errors, the MTTF is improved by a factor 26.3 compared to a memory without ECC. Now, if 6 redundancy rows are also available and managed by the reliability management architecture, the MTTF is improved by a factor 56.7. In conclusion, potential improvements provided by online redundancy management are very large.

In Chapter IV, another fault tolerance technique based on a hierarchical error correcting code has been proposed. The aim of this technique was to consider the operating granularity difference between read and program operations in Flash memories. Indeed, in these devices, data are read per word and programmed per page. The hierarchical code is decomposed in two codes, a word code and a page code. The first is used for error detection and the second for error correction. During normal read operations, if an error is detected on a word being read, the full page is read back and error is corrected thanks to the page code. Schemes allowing 1 error detection (ED) per word and 1 error correction (EC) per page or 1 EC per word and 2 EC per page are possible. In particular, one encoder/decoder embodiment has been patented [USPT08].

In a second time we have considered the following reliability management scheme: periodically, page are refreshed or repaired with redundancy depending on the number of errors in it. A mathematical modeling based on Homogeneous Markov Process has been proposed [ETS08]. Due to the complexity of the scheme, the model of Chapter II has not been employed for the benefit of a simplified reliability model. However, large reliability improvements can still be seen. For instance, in case of a hierarchical ECC allowing 1 EC per word and 2 EC per page compared to a standard ECC allowing only 1 EC per word, the MTTF has been improved by a factor 5.2. In addition, the integration of some row redundancy can substantially improve the MTTF of the memory in case of hierarchical ECC.

The purpose of this thesis has led us to find solutions to improve reliability of NOR flash embedded memories. Even if proposed techniques were focused on this specific environment, they can be extended to other types of memories such as flash NAND, DRAM or MRAM for instance. A similar work can be done by considering specificities of each technology to tune the reliability management scheme. In addition, this work opens the way to some perspectives and future works.

First, a perspective would be to improve the Markov modeling presented in the Chapter IV. In particular, the reliability of a floating gate could be taken into account more accurately by assuming a non-constant transition over the time reflecting a non-homogeneous Markov process. Next, as long as NOR Flash memories are used for storing data and code, it would be interesting to derive

optimal repair and refresh policies for handling code and data based on real use cases.

Secondly, presented techniques were centered on the array and the memory cell. However, a non-negligible part of flash memories are constituted by the periphery which also includes high voltage devices. Thorough the memory life, these devices endure specific stresses which may result in a breakdown and a complete failure of the system. In consequence, a second perspective of our work lies in the study of the periphery. In a first step, it should be focused on the description of failure mechanisms in high voltage devices. Secondly, design techniques in addition with detection techniques could be developed to make this part more robust and safer.

Finally, even if memories are central elements of SoCs, they are only a part of the processing path. To build a reliable SoC, all elements of this path should be considered and built fault tolerant. It concerns memories, buses and functional elements such as CPU and finite-state-machines but also analog blocks and software parts. Another work perspective arises from the global management of the reliability. As in our approach, this could be made by a specific wrapper at system level. For instance, one function of this wrapper could be to change the communication mode in a bus to make the transmission of some critical information safer in spite of the performance. In parallel, the configuration of the wrapper could be changed by an external intervention of the user or by some specific software instructions. In fact, it is a conjunction of all these techniques which will help to build more reliable and safer products.

Annex A - Fault Tolerance Techniques

The objective of this annex is to provide basics of fault tolerance techniques applied to memories. The first part formalizes the fault tolerance process, it describes basic stages that a system may integrate to be more robust. Next, we are focusing on the error correcting code technique. ECC are introduced with a mathematical tutorial which may help the reader to understand the code construction of Chapter IV. Some codes are also exposed and some examples used in semiconductor memories are commented. Finally, an overview of redundancy repair method is performed. Principle and architectures are exposed. At the beginning, this technique was mostly used to address yield errors, but nowadays some literature examples exist for using it to avoid reliability issues.

A.1. Introduction

Environmental factor and potential errors cannot be completely avoided, a Flash memory is never immunized against failures. So, it must embed special features to handle unexpected errors. Four general techniques [59] have been identified to improve reliability in a system:

- ***Fault avoidance*** – refers to methods and tools employed during the system design to reduce the system susceptibility to faults.
- ***Fault detection*** – refers to techniques detecting faults when the memory is operating on. Usually, after a fault detection other techniques must be employed to correct the fault, or minimize its impacts on the system. It is for instance error detection codes, self-checking logic or watchdog timers.
- ***Fault removal*** – prevents the system from being affected by errors. Fault removal employs techniques such as error correcting code or some other ways to compensate the fault.
- ***Dynamic redundancy*** – refers to multiple techniques including retry, journaling, n-modular redundancy, reconfiguration or degraded mode service.

When a fault is present in the memory, some basic stages can be crossed by the system to handle it:

- ***Fault confinement*** – limits the zone affected by the error.
- ***Fault detection*** – detects that an error has happened. Detection can be on-line or off-line. If on-line the system is still functional during the detection. If off-line the system is unavailable during detection.
- ***Fault diagnosis*** – concerns mechanisms in charge of determining the nature and the location of the fault. Once done, the system can begin to recover from this error.
- ***Reconfiguration*** – When the faulty element is determined, a reliable system can isolate it and replace it with a new reliable element. This is redundancy. One another technique is to run the system in a degraded way affecting its performances but ensuring the service.
- ***Recovery*** – The system tries to eliminate the effects of the fault, by employing, fault masking, retry or roll back.

- ***System restart*** – If the effects of the fault are recovered, the system can try to restart in normal operating mode.
- ***Repair*** – During this stage, elements identified as faulty are repaired, this can be done either on-line or off-line.
- ***Reintegration*** – After the repair, the system goes from a degraded mode to a full operating mode.

For Flash memories embedded with microcontroller, previous stages are managed in system by software or hardware. Related techniques are:

- Error correcting codes [22,53]
- Redundancy repair [60]
- Refresh [49,61]
- Scrubbing [60,57]
- Bad block management [62,63]
- Wear leveling [64,65]

In the rest of this annex, we provide details on the two first techniques: the error correcting code technique and the redundancy repair technique. After an introduction of principles and mathematical background, some example taken from the literature will be commented.

A.2. Error Correction Coding

A.2.i. Introduction

The Error Correcting Codes (ECC) as well as the compression and cryptography theories are part of the coding theory [66]. As illustrated in Figure A.1, ECC is based on the concept that some level of data integrity can be ensured when data cross through a noisy communication channel if some redundant data are transmitted too.

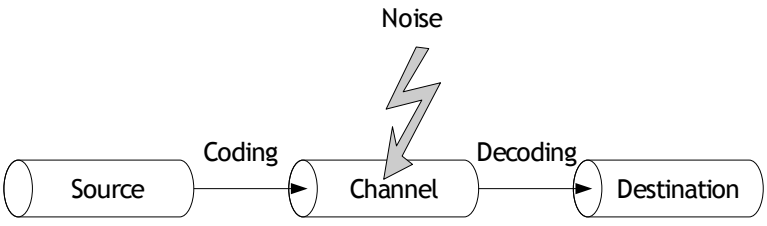

Figure A.1: Fundamental scheme of transmission channel

While Claude Shannon was developing the information theory, a first example of error correcting code has been proposed by Hamming in 1950 [67]. Since this time, the field of coding has evolved a lot. Today, it combines various disciplines in various field such as the group theory, linear algebra, numeric communication and signal processing.

Error correcting codes can be classified in two major classes: the block codes and the lattice codes as shown in the Figure A.2. A third class called concatenated codes has also been developed and is a mix of lattice codes and/or block codes. In this part, we will focus more particularly on linear block codes that are well suited to applications where information is processed by block of bits. This is the case of semiconductor memories.

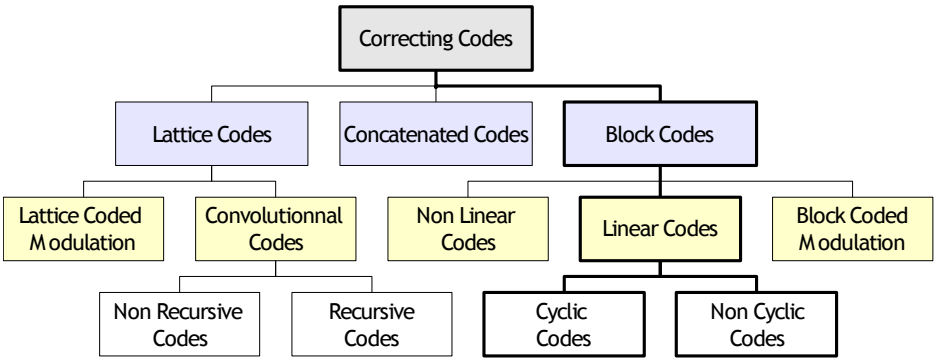

Figure A.2: Classification of Error Correcting Codes

A.2.ii. Block codes principle

A.2.ii.a. Notion of message

Let F_q be a finite alphabet of cardinal q, an element of F_q is called a ***symbol***. A ***message*** i of length k is represented by a vector:

$$i = (i_1, \ldots, i_k) \in F_q^k \tag{A.1}$$

The maximum number of distinct message is q^k. F_q^k is called the ***message space***. For instance if $q = 2$, $F_2 = \{0,1\}$ and a message of length n will be a binary vector of F_2^n. Let us consider a semiconductor memory. It can be modeled by a noisy communication channel. If an information i is written, stored or read, it can become erroneous and a corrupted data can be transmitted. It is because all possible messages of F_2^n are valid.

A.2.ii.b. Coding

To avoid this problem, valid messages must be interleaved with invalid messages. To do so, messages are translated from the message space to a new space called ***coding space*** *(or* ***codeword space****)*. Let n be an integer such that $n > k$ and C be a subset of F_2^n. C is called ***code***. n is the **length** of the code C. A coding function f_{coding} for a code C is an injective application such that:

$$\begin{array}{rccc} f_{coding}: & F^k & \rightarrow & F^n \\ & i=(i_1,\ldots,i_k) & \rightarrow & c=(c_1,\ldots,c_n) \end{array} \quad \text{(A.2)}$$

Such that $f_{coding}(F^k)=C$, by consequence f_{coding} is a bijection from F^k to C. A ***codeword*** c will be stored instead of i. The coding principle is exposed graphically in the Figure A.3. Elements of F_q^k and F_q^n are represented. As we can see, elements of C (valid codewords) are spaced by invalid codewords. To switch from the message space to the coding space, some redundancy ($p=n-k$ symbols) have been added. Valid codeword are sufficiently distinct from one to the other to correct a given number of errors.

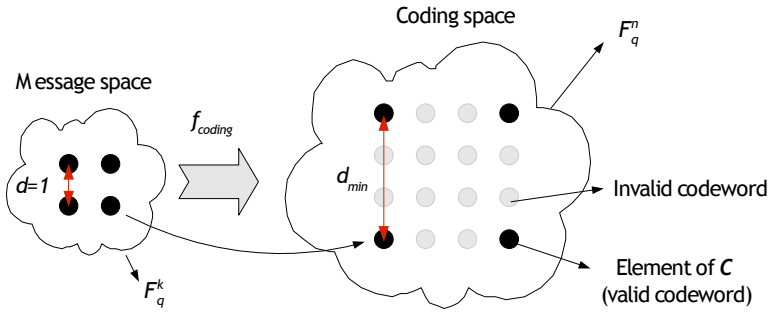

Figure A.3: Principle of encoding

To measure the separation between two codewords, we must introduce the notion of distance. The ***hamming distance*** between two words $x=(x_1,\ldots,x_n)$ and $y=(y_1,\ldots,y_n)$, noted **d(x,y)**, is the number i of position such that $x_i \neq y_i$.

Definition*: The* ***minimal hamming distance*** *of a code C is the minimal number d(x,y) such that x and y are elements of C. This distance is usually noted d_{min}. The minimal hamming distance is the most important notion of the ECC theory because it defines the code efficiency to detect and to correct a given number errors.*

Definition: C is a ***t-Error Correcting*** (*t-EC*) code when any set of at most t errors in the message can be corrected. C is *t-EC* if and only if the d_{min} of C is $d_{min} \geqslant 2 \cdot t + 1$. *t is called* ***error correction capacity*** of the code.

Definition: C is a ***d-Error Detecting*** (*d-ED*) code when any set of at most d errors in the message can be detected. C Is *d-ED* if and only if the d_{min} of C is $d = d_{min} - 1$ has a d_{min}. *d is called* ***error detection capacity*** of the code.

A code C is usually noted (n, M, d_{min}), where n is the length, M is the number of codewords and d_{min} is the minimal distance of the code.

A.2.ii.c. Decoding

Let us consider that the word $c \in C$ is transmitted. After passing the communication channel, the message $r \in F_q^n$ is received. As e errors may have occurred, c and r may be different such that $r = c + e$. A decoding function associated to C is a surjective application $f_{decoding}$:

$$\begin{array}{rccccc} f_{decoding}: & F_q^n & \rightarrow & C \subset F^n & \rightarrow & F_q^k \\ & r & \rightarrow & c' & \rightarrow & m' \end{array} \quad \text{(A.3)}$$

Such that $c' \in C$. r is usually decoded by using the ***principle of maximum of likelihood***. In other word, r is decoded in a word c' of C which is at the minimal distance of r. Graphically, in Figure A.4, if r is different from c due to errors e. Thanks to the principle of maximum of likelihood, r will be decoded into $c' = c$. Next, c' will be decoded into m' by applying the function f_{coding}^{-1}.

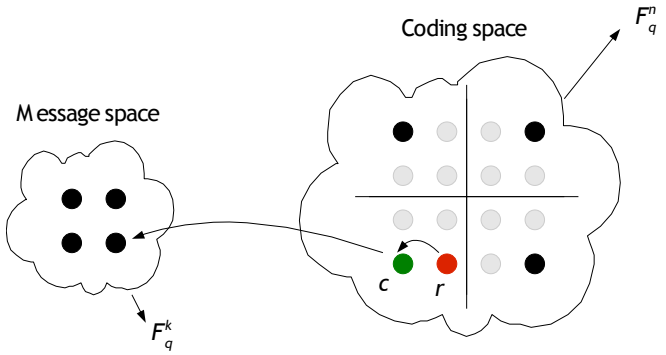

Figure A.4: Principle of decoding

They are three possible results of the decoding process:

- ***Correct decoding*** – the message is affected by at most t-errors, r is decoded into $c \in C$.
- ***Error detection*** - Errors can be only detected in the received message if $r \notin C$. It happens when r is at equidistance from multiple codewords of C. The decoding process is impossible but we can say that at least $t+1$ errors happen.
- ***Decoding problem*** – If there are at least $t+1$ errors. Either r is decoded into c' such that $c' \neq c$, it is a ***miscorrection***, either r is equidistant from multiple codewords of C, it is an error detection.

A.2.iii. Linear codes

If a code has no particular construction, error decoding and correction may be very difficult or impossible. Indeed, for each received message, it would consist in comparing it with all possible codewords of C and identify the closer one. In order to facilitate codes construction and manipulation, they are chosen with additional mathematical properties such as linearity. A linear code is a subset of F_q^n where F_q is a finite field. A element $x \in F_q^n$ will be noted $x = (x_1, \dots, x_n)$ where $x_i, i \in [\![1, n]\!]$ are the coordinates of x in the canonical base of F_q^n. Linearity is a property of codes allowing that the linear combination of two valid codewords to be a valid codeword. If $y \in F_q^n$, then $x + y = (x_1 + y_1, \dots, x_n + y_n) \in F_q^n$ and if $a \in F_q$, then $a \cdot x = (a \cdot x_1, \dots, a \cdot x_n) \in F_q^n$

***Definition**: The **weight** of a word $x = (x_1, \dots, x_n) \in F_q^n$, noted $w(x)$, is the number of position i such that $x_i \neq 0$. As $d(x, y) = w(x - y)$, the minimal distance d_{min} of a linear code C is the minimal weight $w(x)$ for $x \in C$ with $x \neq 0$.For instance, $w(1,1,0,1) = 3$.*

The three parameters n, k and d_{min} noted (n, k, d_{min}) of a linear code C defines the type of the code. There is always $d_{min} + k \leqslant n + 1$. This relationship is called the ***singleton bound***. Hence, for a fixed word length n, a high correction capacity (high d_{min}) is incompatible with a high number of words in the code (high k).

A.2.iv. Generator matrix and control matrix

To perform the coding and decoding of linear block codes, generator matrices must be introduced. A ***generator matrix*** of a linear code $C(n, k)$ is a matrix whose

rows are a basis of C. A generator matrix G has a length $k \times n$ of rank k. If $\{g_{1,} \dots , g_k\}$ is a basis of C then $C = \{a.G , a = (a_{1,} \dots , a_k) \in F_q^k\}$.

To encode a message, the process is just a simple matrix computation. If $m \in F_q^k$ is the transmitted message then the c codeword will be:

$$c = m \cdot G \tag{A.4}$$

Now, if we consider the orthogonal vectorial subspace of the code C where $< , >$ is the usual scalar product, then:

$$C^{\perp} = \{v \in F_q^n \mid \langle u , v \rangle = 0, \forall u \in C\} \tag{A.5}$$

$C^{\perp}$ is of dimension $n-k$ and can be considered as a linear code $C^{\perp}(n , n-k)$ whose generator matrix, noted H, is called ***control matrix***. A control matrix is a matrix of size $(n-k) \times n$.

An important proposition is the following:

$$C = \{u \in F_q^n \mid u \cdot H^T = 0\} \tag{A.6}$$

It allows to identify that a codeword is element of C by a simple matrix computation.

A.2.v. Systematic codes

Two codes C and C' of parameters (n , k) code are said ***equivalent*** if and only if there exists a permutation σ of coordinates transforming a codeword of C into a codeword of C'.

A linear block code $C(n , k)$ is equivalent to a code $C'(n , k)$ under of ***systematic*** form whose generator and control matrixes are respectively, $G' = (I^k \quad P)$ and $H' = (-P^T \quad I_n - k)$ with P a matrix of size $k \times (n-k)$.

Codes are usually chosen systematic: the initial message appears in clear into the codeword and the redundant information (parity) is appended at the beginning or at the end of the codeword.

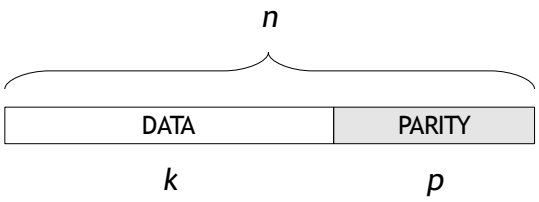

Figure A.5: systematic encoding

To illustrate the systematic encoding, assume the binary code $C(6,3)$ whose generator and control matrices are:

$$G=\begin{bmatrix}1&0&0&1&0&1\\0&1&1&1&0&1\\1&1&0&1&1&0\end{bmatrix} \text{ and } H=\begin{bmatrix}1&1&0&0&0&1\\1&1&0&1&1&0\\0&1&1&0&1&0\end{bmatrix} \quad (A.7)$$

By making the sum of the first and the third lines and after making the permutation (135426), we obtain an equivalent systematic code C':

$$G'=\begin{bmatrix}1&0&0&1&0&1\\0&1&0&1&1&1\\0&0&1&0&1&1\end{bmatrix} \text{ and } H'=\begin{bmatrix}1&1&0&1&0&0\\0&1&1&0&1&0\\1&1&1&0&0&1\end{bmatrix} \quad (A.8)$$

As we can see, the parity matrix P is:

$$P=\begin{bmatrix}1&0&1\\1&1&1\\0&1&1\end{bmatrix} \quad (A.9)$$

Note that the minimal distance d_{min} of this code is 3. Hence, it is a 1-Error Correcting code, commonly named binary Hamming code.

A.2.vi. Linear code decoding

Let us consider a code $C(n,k,d_{min})$. H is a control matrix of C and $t=\frac{d_{min}-1}{2}$ is the correction capacity of the code. The syndrome application is:

$$\begin{array}{rccc} S: & F_q^n & \rightarrow & F_q^{n-k} \\ & x & \rightarrow & H\cdot x \end{array} \quad (A.10)$$

The syndrome of $x\in F_q^n$ is $S(x)=H\cdot x$. Properties of the syndrome are the following

Properties: Let $y\in F_q^n$, then:

- $y\in C \Leftrightarrow S(y)=0$.

- If $y = c + e$ where c is the emitted codeword and e is the error, then $S(y) = S(e)$.
- If $w(y_1) \leqslant t$ and $w(y_2) \leqslant t$, then $S(y_1) = S(y_2) \Rightarrow y_1 = y_2$.

From the properties of the syndrome, a classical decoding method called the ***syndrome method*** can be established.

Assuming a $C(n, k, d_{min})_q$ code with t-Error Correction capacity and a control matrix H. Let consider the emitted codewords $c \in C$ and the error word $e \in F_q^n$ such that the received word $r \in F_q^n$ is affected by at most t errors.

First, we establish a ***decoding table*** of all eventual errors of $e \in F_q^n$ such that $w(e) \leqslant t$. For each $e \in F_q^n$ such that $w(e) \leqslant t$, the syndrome $S(e)$. Thanks to the third property of the syndrome, if $e \neq e'$ then $S(e) \neq S(e')$. A correspondence table between $S(e)$ and e can be constructed.

Next, we compute $s = S(r) = S(c + e) = S(e)$. If $s \neq 0$ then $e \neq 0$, errors are detected. Then, there are two cases:

- s is in the decoding table.
- Otherwise, we can say that r is affected by more than t errors, decoding is impossible.

This method is very effective but relatively expensive due to the computation of the decoding table.

A.2.vii. Code constructions

Multiple linear block code constructions exist. Binary hamming codes, Cyclic codes, Bose-Chaudhuri-Hocquenghem (BCH), Reed Solomon (RS), Golay codes are amongst the most popular linear bloc codes. They can all be classified depending on their mathematical structure, a map is dressed in to the Figure A.6.

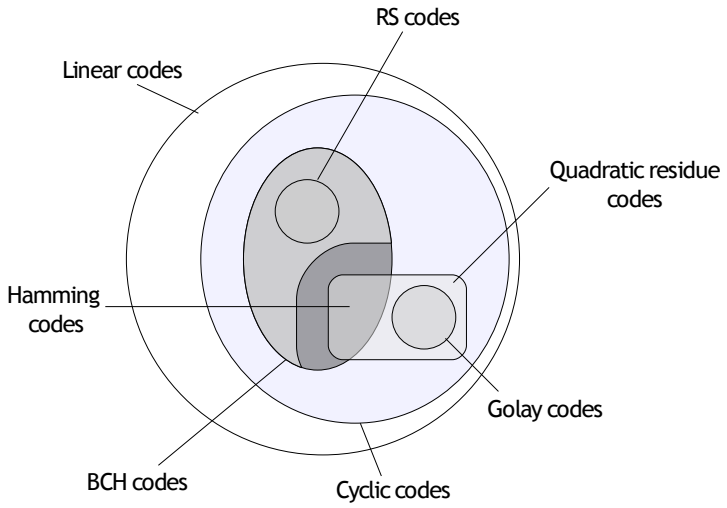

Figure A.6: Linear block code map

The main characteristic of linear block codes is summed by their parameters. For some of these codes, parameters have been reported in Table A.1.

Table A.1: Some code parameters

Code	Parameters
Parity	$(k+1, k, 2)$
Golay	$(23,12,7)$
Extended Golay	$(24,12,8)$
Binary Hamming	$(2^m-1, 2^m-m-1, 3)$
Extended Hamming	$(2^m, 2^m-m-1, 4)$
Generalized Hamming	$\left(\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m, 3\right)$
Generalized BCH	$(q^m-1, q^m-1-deg(g), d_{min} \geq \delta)$
Primitive Reed Solomon	$(2^m-1, k, n-k+1)$

A.2.vii.a. Hamming codes

Binary Hamming codes [68] are 1 error correcting code with minimal redundancy. They are usually denoted $(2^m-1, 2^m-m-1, 3)$.

A.2.vii.b. BCH codes

BCH codes is a class of cyclic codes [54]. Let consider the F_2 the binary finite field composed of elements $\{0,1\}$ and the extension field F_2^m composed of 2^m elements noted $(0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^m-2})$. A polynomial over F_2 is of the form:

$$f(X) = f_0 + f_1 \cdot X + f_2 \cdot X^2 + \ldots + f_n X^n \quad \text{(A.11)}$$

Where, $f_i, i \in [\![0, n]\!]$ are equal to 0 or 1 and the root of this polynomial are elements of the extension field. If β is any element of the extension field, β is a root of f if and only if $f(\beta) = 0$.

Let $\beta \in F_2^m$. The minimal polynomial of β is $f(X)$ where $f(X)$ is the polynomial of smallest degree of F_2 with β as root. The elements, β^{2^r}, for any $r > 0$ are called the conjugates of β. If β is a root of a polynomial over F_2, the conjugates of β are also roots of this polynomial. In consequence, conjugates of β have the same minimal polynomial as β.

A binary BCH code is determined by a generator polynomial, $g(X)$. For a t-Error Correcting code, $g(X)$ is the lowest degree polynomial with α, α^2, α^3, ..., α^{2t} as roots. If $f_i(X)$ for $i \in [\![1, 2t]\!]$ is the minimal polynomial of α, then $g(X)$ is the ***least common multiple*** (LCM) of $f_1(X), f_2(X), \ldots, f_{2t}(X)$. Since all conjugates have the same minimal polynomial, and all α^i for even i are conjugates of some odd powered α, this equation is equivalent to g(x) having $\alpha, \alpha^3, \ldots, \alpha^{2t-1}$ as roots and having $g(X) = LCM\left(f_1(X), f_3(X), \ldots, f_{2t-1}(X)\right)$. For any positive integer $m \geq 3$ and $t \leq 2^m - 1$, there exists a binary BCH code with block length n and parity check length of $n - k$ where k is the number of information bits. This code has a minimum distance of d_{min} and the following relationships hold parameters $\left(2^m, \geq 2^m - m \cdot t - 1, \geq 2 \cdot t + 1\right)$.

A.2.vii.c. Reed Solomon Codes

Reed Solomon (RS) [68] codes are a subset of BCH codes. Usually, they are denoted RS(n,k), where n represents the number of symbols of m bits (with $n \leq 2^m - 1$) of a codeword and k represents the number of the related information word. The encoding process starts from the k data symbols composed of m bits and adds the parity symbols to construct an n symbol codeword. Therefore, $n - k$ parity symbols are present.

A $RS(n, k)$ code can correct up to $2 \cdot er + re \leq n - k$, where er is the number of erasures and re is the number of random errors. For data transmission, a random error occurs when a symbol of the received codeword differs from the

transmitted codeword at an unknown location. An erasure is said to occur when the channel side information available from the receiver allows to localize the erroneous symbol in the code word. A Reed-Solomon codeword is shown in

Reed-Solomon encoding and decoding can be performed in hardware or in software. Reed-Solomon codes are based on finite fields theory and a Reed-Solomon codeword is generated using a special polynomial. All valid codewords are exactly divisible by the generator polynomial. The general form of the generator polynomial is given by:

$$g(x) = (x - \alpha^{i})(x - alpha^{i+1}) \ldots (x - alpha^{i+2t}) \quad \text{(A.12)}$$

A Reed-Solomon decoder tries to identify the position and magnitude of up to t errors (or $2t$ erasures) and correct them.

A.2.viii. ECC implementation in Flash memories

Up to now, we have introduced mathematical definitions of error correcting codes. We have also presented some classical linear block code constructions. In the rest of the Section A.2, we are providing three examples of ECC implementation into Flash memories found in the literature. They are illustrative examples to show which are memory constraints and which kinds of optimizations are possible by choosing a proper ECC.

A.2.viii.a. Standard implementation in NOR Flash and SRAM memories

One shot encoder/decoders are popular in NOR Flash and SRAM memories and iterative encoders/decoders are never used due to a problem of performance. An illustration of the one shot (also named parallel) coding/decoding process is given in Figure A.7. Indeed, thanks to a reduced word length, their implementation is relatively easy. Encoding is performed by a parity generator that is a simple XOR tree. Decoding requires computation of the parity by a parity generator and comparison with the stored parity. In a second step a LUT table allows to determine the error pattern. Finally, a bitwise XOR between the error vector and the read data word permits error correction. Codes usually employed in a standard implementation are Hamming codes or extended Hamming codes [69], because they allows one error correction without degrading too much performances of the memory design.

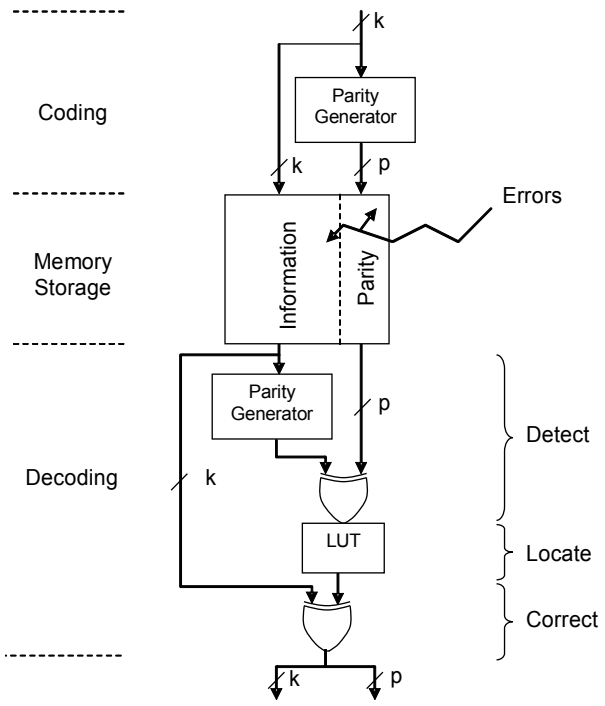

Figure A.7: Error correction implementation in semiconductor device

A.2.viii.b. NAND memories

The read operation in NAND flash memories is a two steps operation. First, the content of a full page is read and stored into a high speed memory buffer. Secondly, data are sent to the user serially. In consequence, ECC with large k and n parameters can be chosen to reduce parity.

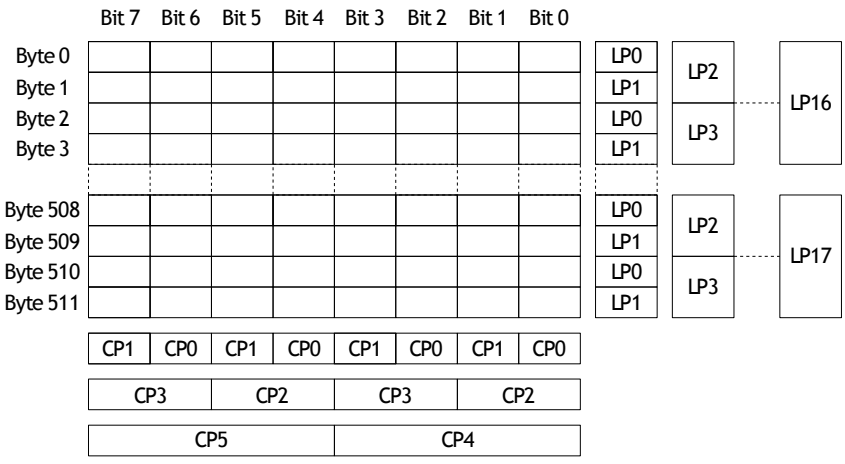

Figure A.8: Parity generation

In such type of application efficient encoding is possible [70], for each 512 bytes of data, a 1-EC code is used to generated a 24 bits parity. The parity is stored into 3 additional bytes that are appended to the memory page.

The parity computation is decomposed in two operations: a row parity computation (RP_i) and a column parity computation (CP_i). The layout of the code construction is shown in the Figure A.8. For instance, the bit CP_1 is obtained by XORing for all bytes the bits 6, 4, 2 and 0.

This code is able to detect 2 errors and correct up to 1 error. The decoding flowchart is available in the Figure A.9. Depending on an analysis of the number of “1” into the syndrome, the ECC decoder will be able to:

- Do nothing.
- Correct one error.
- Detect multiple errors.
- Declare a faulty behavior of the ECC decoder.

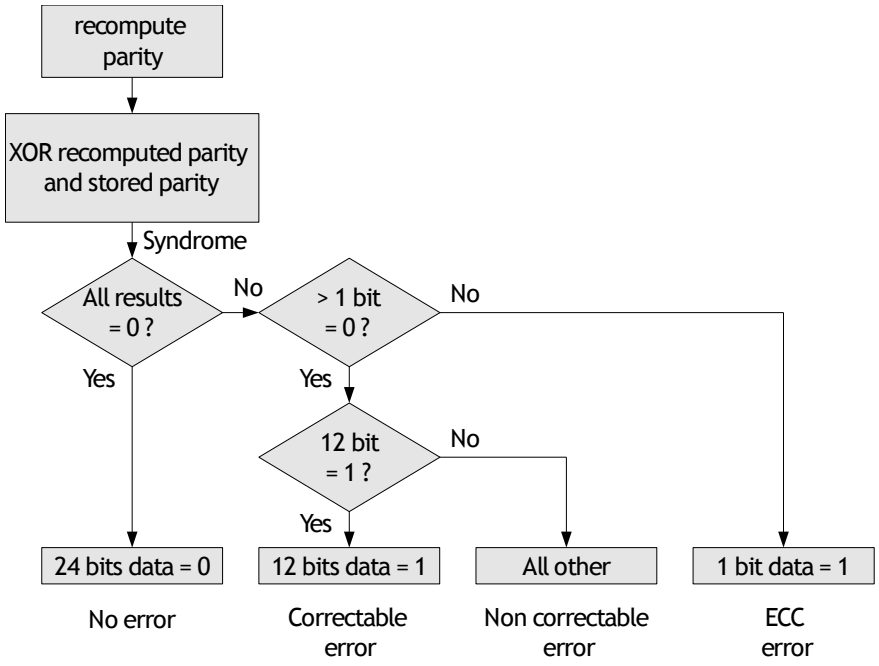

Figure A.9: Error correction flowchart

A.2.viii.c. Multi-level memories

An efficient ECC organization for multi-level memories has been proposed by [71]. Multi-level memories have the particularity to store more than one bit into one memory cell. In the context of floating gate memories, it can be performed by tightly controlling the charge injection in the cell.

Conventionally, Figure A.10, a 2^q-ary correcting code is used in the following way. One cell corresponds to one symbol that can have 2^q states. For each k cells,

$m = n - k$ parity symbols (equivalently cells) are added. Hence, a codeword is composed of n symbols. In any block of n cells, the ECC scheme is able to correct up to t symbols. This approach is straightforward but when the number of bits per cell increases, the encoding and decoding circuits become very complex. This results in large overhead in term of area and time. It mainly comes from the fact that addition and multiplication operators are required F_{2^q} for the implementation of this scheme.

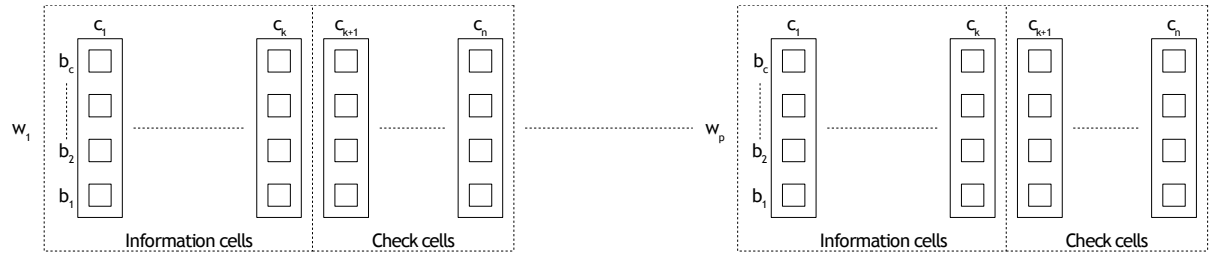

Figure A.10: Conventional ECC organization

The 2^q-ary correcting code and the word layout of the Figure A.10 can efficiently be removed by using a binary correcting code and the word layout of the Figure A.11. Bits of a single memory cell do not correspond to the same symbol but are bits of distinct codewords. Assuming that q is the number of bit per cells. For each k cells, there are $m = n - k$ parity cells and n cells contains q codewords. Each n bits layer block is protected a binary t-EC scheme.

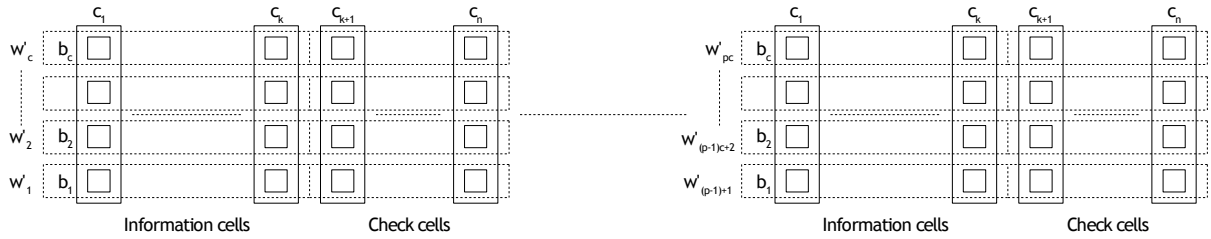

Figure A.11: New ECC organization

Evaluations have been performed considering a 4bit/cell storage solution in a 0.25µm CMOS process for the distinct approaches with 1-EC codes. As we can see, the parity cell overhead is higher with the proposed scheme. However, in term of coder/decoder delay and area, this solution is far more optimized than the traditional approach.

Table A.2: Comparison between the new ECC approach and a conventional ECC approach for 4-bit/cell storage

	New code HAMMING(71,64)	BCH(67,64) 16-ary symbols
Number of data cells (k)	64	64
Number of parity cells (m)	7	3
Coder gate count	278	1526
Decoder gate count	523	2949
Coding delay	2.5ns	3.9ns
Decoding delay	6.7ns	13.3ns

A.3. Redundancy

A.3.i. Principle

Memory repair consists in replacing some defective part of a memory with some redundant elements working correctly. This technique is known to be an efficient solution to solve the problem of yield loss due to deep sub-micron defects and process uncertainties in memory products. From a physical point of view, the array is still faulty but from a logical point of view, the array is fault free.

A.3.ii. Redundancy architectures

Several types of redundancy resources can be considered in Flash memories. For instance, row redundancy (or word or page redundancy), column redundancy (or bit redundancy), row redundancy (or word redundancy) and mixed row/column redundancy.

In row redundancy, Figure A.12, a memory row is replaced by a row redundancy. It allows to address issues in:

- row address decoders.
- memory cells of a single word, multiple words.

A content addressable memory (CAM) redirects the row addressing toward the main matrix or the redundancy. In an embedded Flash process two methods can be used:

- A non-volatile CAM made with floating gate devices.
- A volatile CAM and use a part of the non-volatile matrix to store the reconfiguration data. During start-up of the product, CAM is filled up with reconfiguration data, slightly increasing the power up delay.

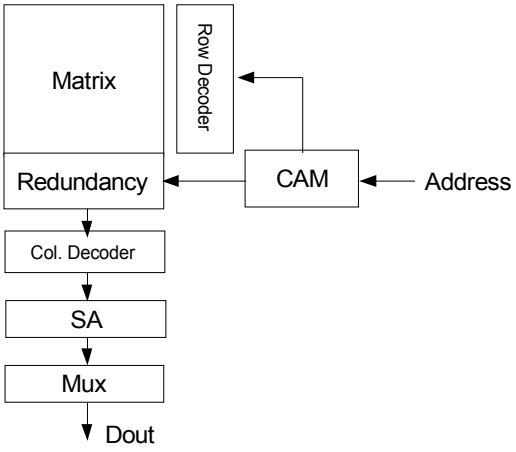

Figure A.12: Row redundancy architecture

In column redundancy, Figure A.13, the redundant element is the unit responsible for generating one bit of data. This unit can replace defects affecting column address decoder, columns of bits, columns multiplexers, sense amplifiers.

As the number of memory element is important, replacing a column permits to repair a larger number of defects in comparison with row repair. For instance, this technique stays the most used in laser reconfiguration techniques.

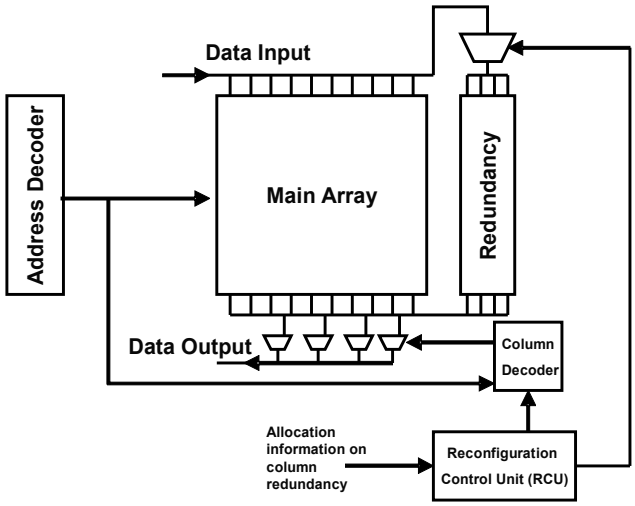

Figure A.13: Column redundancy architecture

In order to reduce the redundancy resources requirements, the redundancy architecture may be constrained depending on the memory matrix architecture as illustrated in the Figure A.14. For instance [72], assume the memory array is shared into part. One column redundancy can replace distinct bits into each of these blocks. It provides a better redundancy granularity. This method allows to limit the number of sense amplifiers, reduce the area and power consumption.

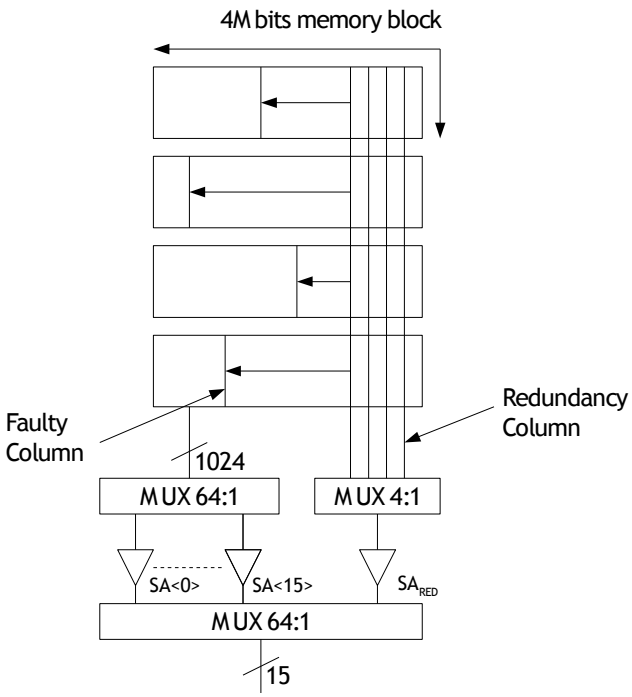

Figure A.14: Block diagram of column redundancy architecture for flash memory

A.3.iii. Repair methodologies: external repair versus built-in repair

Test and repair of memories are usually performed by using a probe station at wafer-test stage. The repair step is external and composed of three phases:

- A software phase where faults are detected and located in the memory using a test algorithm.
- A software phase where the best way to use redundancy resources is determined depending on the fault information.
- An hardware phase where the defective parts are disconnected and the redundancy part are connected to reconfigure the memory using some reconfiguration logic. In normal operation, the reconfiguration information must be conserved and is usually written in non-volatile device fused by laser for instance.

For embedded flash product, Built-In Self Test and Built-In Self Repair can be considered as a cost-effective solution to address the problem of redundancy. The Built-In Self Repair (BISR) for memories is an autonomous repair technique in which, the phases of detection, localization and replacement of faulty elements are realized inside the circuit. The BISR design is usually composed of three parts:

- The built in self test (BIST) module – performs test and fault location, collecting information for BIRA.
- The built-in redundancy algorithm (BIRA) – performs redundancy analysis/ allocation under certain conditions (redundancy architecture, test algorithm, memory type…)
- The address reconfiguration module – to redirect the memory operation toward the redundancy resources.

As for external repair, the internal repair is composed of the same three phases which are realized respectively by the BIST (detection location phase), the BIRA (redundancy analysis phase) and the reconfiguration module (repair phase).

A.3.iv. Internal repair advantages and recommendations

Contrarily to external repair (laser fuse), the self integrated repair, does not require external material. In case of embedded memories, this technique has many advantages:

- The integrated self repair circuitry allows to avoid an expensive repair equipment based on laser technology. The laser technique uses electrical devices that are not part of the standard CMOS process and thus implies a significant increase of the manufacturing cost. This equipment may also give birth to new defect.
- It allows to significantly reduce the repair time.
- The BISR could offer the possibility to repair the memory during the fabrication process to improve the yield but also during the field to improve the reliability.
- The BISR may be useful to repair buried memories because of issue of observability or controllability.

As in the case of BIST, a BISR has a cost in term of area and performance. The only solution is to minimize timing and area additional cost:

- Limit the time necessary to perform a repair operation.
- Implement algorithm that could manipulate efficiently redundant resources to repair a set of faults wherever they are located (on main matrix or on the redundancy matrix)

- Minimize the cost area and the temporal penalty induced by the use of the BISR.
- The repair circuitry must be integrated with the memory not within the memory. This is very important to avoid internal structure modification of the memory.

An efficient BISR implementation combines antagonistic requirements at a reasonable cost in term of repair efficiency, area cost and time. By consequence, it is a difficult task. Actually, repair algorithms have a lot of limitations. This is justified because error probability does not exceed tenth errors by megabits. The actual trend is to reduce the area cost over a high repair capacity.

A.3.v. Repair algorithms

A.3.v.a. 2-D redundancy problem

Let's a memory have column and row redundancy. During production test, defective cells are detected and must be replaced by spare redundancy. The redundancy analysis problem is then the following:

- Choose the minimum number of spare rows and columns that cover all the faulty cells.

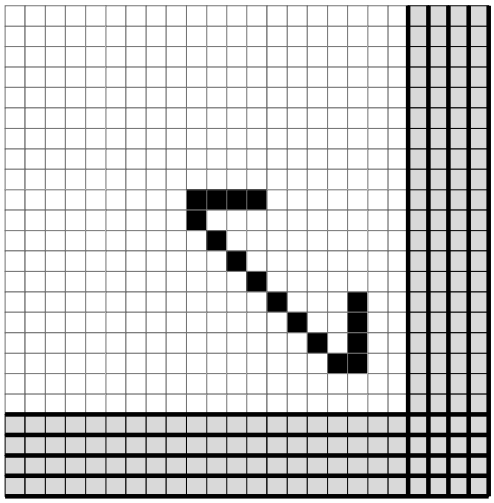

Figure A.15: Error bitmap of a memory

The complexity of the 2-D redundancy analysis is NP-complete. The time impact to determine the optimal solution is crucial. The objective of a redundancy algorithm is to reduce the time necessary to find a solution.

Typically, 2-D redundancy analysis algorithms are decomposed in a must-repair phase followed by a final-repair phase [73]:

- Must-repair phase: It consists in replacing with a spare row (spare column), a row (columns) that contains more defective cells than the number of free spare columns (rows).
- Final-repair phase: Various redundancy allocations policies can be employed such as exhaustive search [74], repair most, row-first, column-first, repair most using local bitmap, essential spare pivoting (ESP) [75].

Historically, algorithms were set up to mainly reduce the temporal penalty due to solution search. These algorithms require full or a partial error bitmaps only compatible with external repair solutions. With the emergence of embedded memories and systems on a chip, new algorithms were required.

A.3.v.b. Built-in redundancy algorithms

The goal of the built-in redundancy algorithm (BIRA) is to allocate redundancy properly in parallel with the test execution. The BIRA will be executed at the same time than of BIST or a Serial Test Interface (STI).

BIRA is a compromise between hardware and timing overhead. The choice of the algorithm is of primary importance. With such an algorithm, the solution search will not be exhaustive. It may conduct to some yield loss even if a repair solution exits. As seen in the Figure A.16, depending on the error bitmap and the employed algorithm, memory will be repaired or not. Additionally, to limit the size of the error bitmap, a local bitmap must be constructed [75].

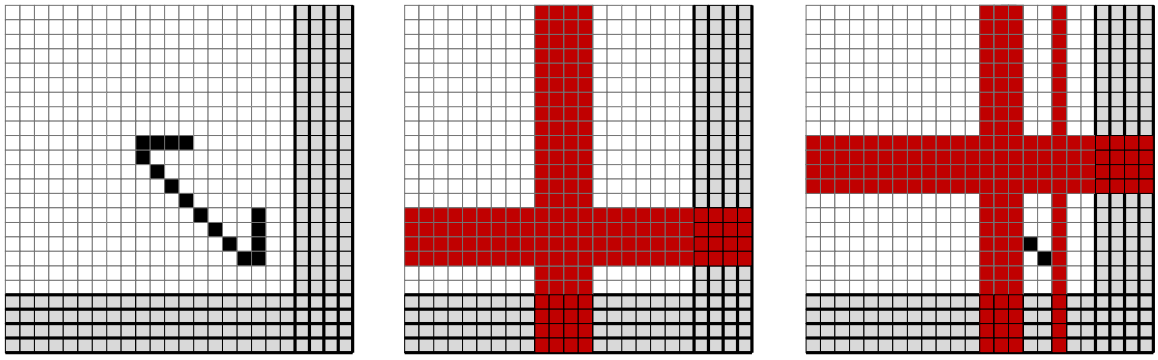

Figure A.16: Memory repair with distinct algorithms (a) error bitmap (b) exhaustive search solution (c) repair most solution

By order of complexity, algorithms that suit embedded memories are:

- ***Row-first*** – rows redundancy are first allocated. When no more rows are available columns redundancy are allocated.
- ***Column-first*** – columns redundancy are first allocated. When no more columns are available rows redundancy are allocated
- ***Essential Spare Pivoting (ESP)*** [75]– Error positions are logged into a bitmap as test protocol is proceeding. At any moment, if a new error shares the same address than a previously logged one, a redundancy element (column or row) is allocated to replace the element containing these two errors.
- ***Local repair-most (LRM)*** [75]– a local bitmap is filled successively on new errors. When the bitmap is full, a repair-most policy is employed.
- ***Local optimization (LO)*** [75]– a local bitmap is filled successively errors. When the bitmap is full, an exhaustive search is performed to clear it.

The complexity of algorithm is correlated with the repair efficiency. Local repair-most and Local optimization algorithms have the best repair rates [75] but their costs is higher in term of area and timing. Row-first, Column-first and ESP have been analyzed and compared in a Flash-NOR context [76]. ESP seems to be a good choice amongst all these solution.

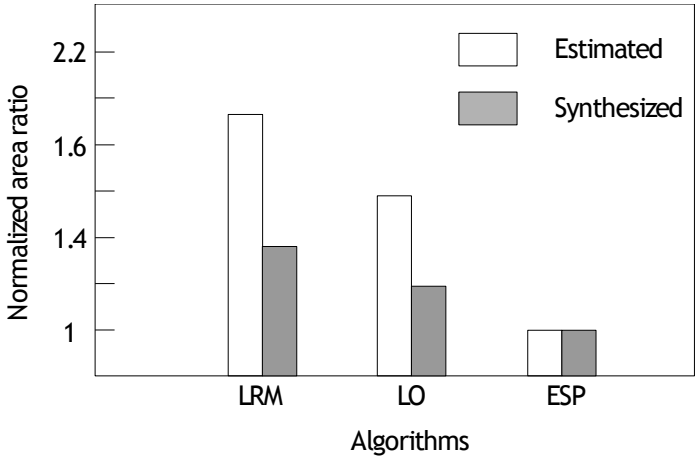

Figure A.17: Area overhead of distinct repair algorithms

A.3.vi. On the fly built-in redundancy repair

Many Built-In Self Repair methodologies for memories have been described into literature [75-82,74]. The main purpose of these architectures is to improve the memory yield during the post production phases. As it is not the purpose of this work, we will not describe it. On the contrary, example of repair architecture

activated during the normal memory use are seldom. In [60], a Built-In Self Repair (BISR) methodology has been described. It allows identification of both hard faults and soft errors during the field. To reach this goal, it uses two techniques: redundancy repair and ECC. This methodology targets to improve memory yield with redundancy and avoid the loss of reliability. The operating process during normal operation is summed in Figure A.18. If an error is identified as a soft error, ECC corrects it with no other actions. If it is identified as a hard fault, the memory is repaired by redundancy during idle cycles if some redundancy elements are still available.

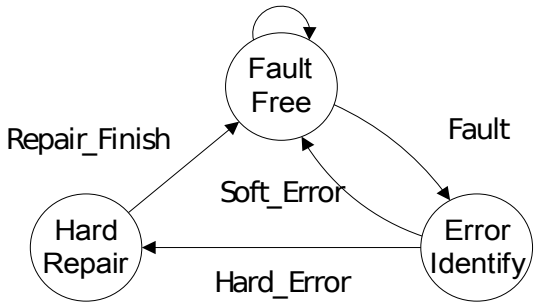

Figure A.18: Phase transition of the proposed scheme

This repair scheme has been proved to be very efficient for SRAM memories as shown in Figure A.19. For instance, let us consider a $32K \times 64$ memory with Hamming code, 8 redundant rows, 4 redundancy columns and a constant error $\lambda_L = 10^{-8}$ per hour. Such a memory with ECC only has an MTTF of 19.758 hours. When the redundancy repair scheme is activated the MTTF is 21.170 hours, it represents an improvement of about +7.1%.

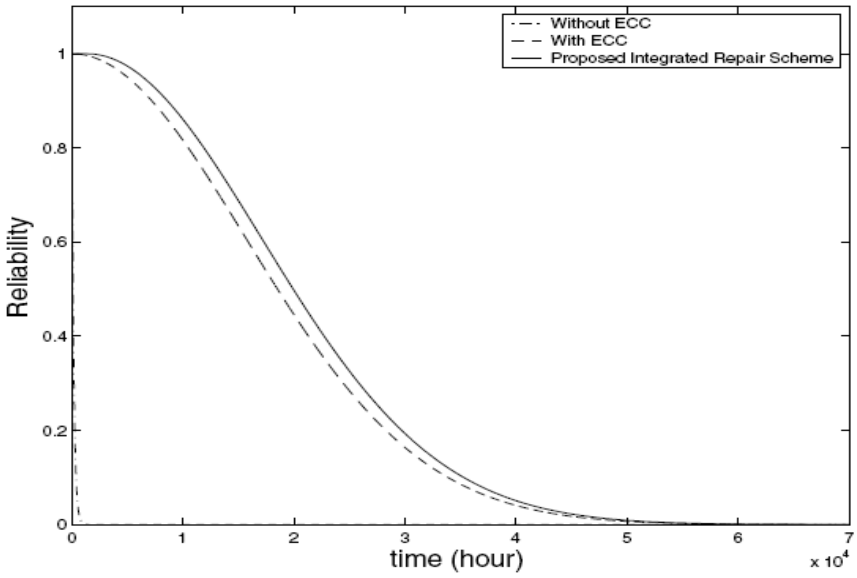

Figure A.19: Reliability of the 32K x 64 memory with ECC (dashed line), without ECC (dot-dashed line) and that using the proposed method (solid lines)

A.4. Conclusion

In this annex, we have reviewed two fault tolerance techniques used in semiconductor memories.

ECC in Flash memory context have been introduced. As we have seen, depending on the memory constraints (standard NOR, NAND, multi-level), the code employed is usually optimized to reduce penalties in term of parity bit overhead, performances and coder/decoder complexity.

In another part, we have reviewed the redundancy technique. Redundancy repair is a conventional technique to improve memory yield. It becomes a complex problem when multiple kinds of redundancy elements are available, Nowadays, there also exists literature examples where redundancy is used to improve the memory reliability.

These two techniques where the starting point of this work. To sum up our scientific step, we chose to answer two questions from previous observations :

- How to optimize the ECC scheme employed in NOR Flash memories?
- What are the expected improvements and impacts if some redundancy elements are also added to improve the NOR Flash reliability?

Bibliography

[1] J. Agin, H. Boyce, and T. Trexler, “Overcoming test challenges presented by embedded flash memory,” *Electronics Manufacturing Technology Symposium, 2003. IEMT 2003. IEEE/CPMT/SEMI 28th International*, 2003, pp. 197-200.

[2] W. Chen, *The VLSI Handbook*, CRC Press, 2006.

[3] R. Rajsuman, “Design and test of large embedded memories: An overview,” *Design & Test of Computers, IEEE*, vol. 18, 2001, pp. 16-27.

[4] W.D. Brown and J. Brewer, *Nonvolatile Semiconductor Memory Technology: A Comprehensive Guide to Understanding and Using NVSM Devices*, Wiley-IEEE Press, 1997.

[5] H. Maes et al., “Trends in semiconductor memoires,” *Microelectronic Journal*, vol. 20, 1989, p. 9.

[6] G. Yaron et al., “A 16K E/SUP 2/PROM employing new array architecture and designed-in reliability features,” *Solid-State Circuits, IEEE Journal of*, vol. 17, 1982, pp. 833-840.

[7] A.K. Sharma, *Semiconductor Memories: Technology, Testing, and Reliability*, Institute of Electrical & Electronics Enginee, 2002.

[8] R. Steimle, “Integration of silicon nanocrystals into a 6V 4Mb non volatile memory array,” 2004.

[9] B. De Salvo, “How far will silicon nanocrystals push the scaling limits of NVMs technologies,” *IEDM Tech. Digest*, 2003, p. 597.

[10] E. Spitale, “Effect of high-k materials in the control dielectric stack of nanocrystal memories,” *IEDM Tech. Digest*, 2004, p. 161.

[11] M. Julliere, “Tunneling between ferromagnetic films,” *Physics Letters A*, vol. 54, Sep. 1975, pp. 225-226.

[12] M.N. Baibich et al., “Giant magnetoresistance of (001)Fe/(001)Cr magnetic superlattices,” *Physical Review Letters*, vol. 61, Nov. 1988, pp. 2472-2475.

[13] S.S.P. Parkin et al., “Exchange-biased magnetic tunnel junctions and application to nonvolatile magnetic random access memory (invited),” *Journal of Applied Physics*, vol. 85, Avril. 1999, pp. 5828-5833.

[14] N. Yamada et al., “Rapid-phase transitions of GeTe-Sb2Te3 pseudobinary amorphous thin films for an optical disk memory,” *Journal of Applied Physics*, vol. 69, Mar. 1991, pp. 2849-2856.

[15] S. Lai and T. Lowrey, “OUM - A 180 nm nonvolatile memory cell element technology for stand alone and embedded applications,” *Electron Devices Meeting, 2001. IEDM Technical Digest. International*, 2001, pp. 36.5.1-36.5.4.

[16] M. Gill, T. Lowrey, and J. Park, "Ovonic unified memory - a high-performance nonvolatile memory technology for stand-alone memory and embedded applications," *Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International*, 2002, pp. 202-459 vol.1.

[17] P. Cappelletti et al., *Flash Memories*, Kluwer Academic Publishers, 1999.

[18] V. Kynett et al., "An in-system reprogrammable 32 K×8 CMOS flash memory," *Solid-State Circuits, IEEE Journal of*, vol. 23, 1988, pp. 1157-1163.

[19] B. Van Zeghbroeck, *Principles of Semiconductor Devices*; http://ece-www.colorado.edu/~bart/book/contents.htm.

[20] K. Takeuchi et al., "A source-line programming scheme for low voltage operation NAND flash memories," *VLSI Circuits, 1999. Digest of Technical Papers. 1999 Symposium on*, 1999, pp. 37-38.

[21] D.C. Richard, "The explosive word of serial flash," *Electronic Component News*, Aug. 2005; www.ecnmag.com.

[22] S. Gregori et al., "On-chip error correcting techniques for new-generation flash memories," *Proceedings of the IEEE*, vol. 91, 2003, pp. 602-616.

[23] K.-T. Park et al., "A 64-Cell NAND Flash Memory with Asymmetric S/D Structure for Sub-40nm Technology and Beyond," *VLSI Technology, 2006. Digest of Technical Papers. 2006 Symposium on*, 2006, pp. 19-20.

[24] T. Ditewig et al., "An embedded 1.2 V-read flash memory module in a 0.18 um logic process," *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*, 2001, pp. 34-35, 425.

[25] C. Peters, "A 130nm high-density embedded EEPROM as universal memory for code and data storage on 1T FN/FN flash cell," *Proceedings of the Non-Volatile Semiconductor Memory Workshop*, 2004, pp. 55-56.

[26] *International technology roadmap for semiconductors (ITRS)*, Semiconductor industry association, 2006; http://www.itrs.net/.

[27] S. Hamdioui, G. Gaydadjiev, and A. van de Goor, "The state-of-art and future trends in testing embedded memories," *Memory Technology, Design and Testing, 2004. Records of the 2004 International Workshop on*, 2004, pp. 54-59.

[28] H. Belgal et al., "A new reliability model for post-cycling charge retention of flash memories," *Reliability Physics Symposium Proceedings, 2002. 40th Annual*, 2002, pp. 7-20.

[29] A. Birolini, *Reliability Engineering: Theory and Practice*, Springer, 2003.

[30] O. Ginez et al., "An overview of failure mechanisms in embedded flash memories," *VLSI Test Symposium, 2006. Proceedings. 24th IEEE*, 2006, p. 6 pp.

[31] "IEEE standard definitions and characterization of floating gate semiconductor arrays," 1998.

[32] P. Pavan, L. Larcher, and A. Marmiroli, *Floating Gate Devices: Operation and Compact Modeling*, Kluwer Academic Publishers, 2004.

[33] P. Pavan et al., “Flash memory cells-an overview,” *Proceedings of the IEEE*, vol. 85, 1997, pp. 1248-1271.

[34] P. Kuhn et al., “A reliability methodology for low temperature data retention in floating gate non-volatile memories,” *Reliability Physics Symposium, 2001. Proceedings. 39th Annual. 2001 IEEE International*, 2001, pp. 266-270.

[35] G. Tempel, “Abnormal charge loss of flash cells at medium temperatures,” *Proceedings of the Non-Volatile Semiconductor Memory Workshop*, 2000.

[36] J.B. Razafindramora, “Modélisation et caractéristation de transistors MOS appliquées à l'étude de la programmation et du vieillissement de l'oxyde tunnel dans les mémoires EEPROM,” Dec. 2004.

[37] Yow Wern Shiong Eric et al., “Reliability characterization of a 0.6 /spl mu/m FLOTOX EPROM process,” *Semiconductor Electronics, 2002. Proceedings. ICSE 2002. IEEE International Conference on*, 2002, pp. 490-499.

[38] R. Degraeve et al., “Analytical model for failure rate prediction due to anomalous charge loss of flash memories,” *Electron Devices Meeting, 2001. IEDM Technical Digest. International*, 2001, pp. 32.1.1-32.1.4.

[39] L. Larcher and P. Pavan, “Statistical simulations to inspect and predict data retention and program disturbs in flash memories,” *Electron Devices Meeting, 2003. IEDM '03 Technical Digest. IEEE International*, 2003, pp. 7.3.1-7.3.4.

[40] R. Degraeve et al., “Statistical model for stress-induced leakage current and pre-breakdown current jumps in ultra-thin oxide layers,” *Electron Devices Meeting, 2001. IEDM Technical Digest. International*, 2001, pp. 6.2.1-6.2.4.

[41] F. Schuler et al., “Physical description of anomalous charge loss in floating gate based NVM's and identification of its dominant parameter,” *Reliability Physics Symposium Proceedings, 2002. 40th Annual*, 2002, pp. 26-33.

[42] R. Degreave et al., “A new analytic model for the description of the intrinsic oxide breakdown statistics of ultra-thin oxides,” *Reliability of Electron Devices, Failure Physics and Analysis, 1996. Proceedings of the 7th European Symposium on*, 1996, pp. 1639-1642.

[43] R. Degraeve et al., “Analytical percolation model for predicting anomalous charge loss in flash memories,” *Electron Devices, IEEE Transactions on*, vol. 51, 2004, pp. 1392-1400.

[44] B. De Salvo, “Analysis of the electrical transport and reliability of floating gate non volatile memory insulators,” 1999.

[45] T. Ong et al., “Erratic Erase In ETOX/sup TM/ Flash Memory Array,” *VLSI Technology, 1993. Digest of Technical Papers. 1993 Symposium on*, 1993, pp. 83-84.

[46] A. Chimenton and P. Olivo, “Erratic erase in flash memories - part I: basic

experimental and statistical characterization," *Electron Devices, IEEE Transactions on*, vol. 50, 2003, pp. 1009-1014.

[47] A. Chimenton and P. Olivo, "Erratic erase in flash memories - Part II: Dependence on operating conditions," *Electron Devices, IEEE Transactions on*, vol. 50, 2003, pp. 1015-1021.

[48] J. Portal, H. Aziza, and D. Nee, "EEPROM memory: threshold voltage built in self diagnosis," *Test Conference, 2003. Proceedings. ITC 2003. International*, 2003, pp. 23-28.

[49] Y. Furuta and T. Okumura, "United States Patent: 4218764 - Non-volatile memory refresh control circuit," Aug. 1980.

[50] F. La Rosa, "United States Patent: 6735733 - Method for the correction of a bit in a string of bits," May. 2004.

[51] A. Hoefler et al., "Statistical modeling of the program/erase cycling acceleration of low temperature data retention in floating gate nonvolatile memories," *Reliability Physics Symposium Proceedings, 2002. 40th Annual*, 2002, pp. 21-25.

[52] J. Harthong, *Probabilités & statistiques: De l'intuition aux applications*, Diderot Arts et Sciences, 1980.

[53] T. Tanzawa et al., "A compact on-chip ECC for low cost flash memories," *Solid-State Circuits, IEEE Journal of*, vol. 32, 1997, pp. 662-669.

[54] L. Joiner and J. Komo, "Decoding binary BCH codes," *Southeastcon '95. 'Visualize the Future'., Proceedings., IEEE*, 1995, pp. 67-73.

[55] S. Morioka and Y. Katayama, "Design methodology for a one-shot Reed-Solomon encoder and decoder ," *Computer Design, 1999. (ICCD '99) International Conference on*, 1999, pp. 60-67.

[56] M.Y. Hsiao, "A Class of Optimal Minimum Odd-weight-column SEC-DED Codes," *IBM Journal of Research and Development*, vol. 14, 1970, p. 395.

[57] G. Cardarilli et al., "Data integrity evaluations of Reed Solomon codes for storage systems [solid state mass memories]," *Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings. 19th IEEE International Symposium on*, 2004, pp. 158-164; http://ieeexplore.ieee.org/iel5/9332/29643/01347836.pdf.

[58] E. Pardoux, *Processus de markov et applications - algorithmes, réseaux, génome et finance : cours et exercices corrigés*, Dunod, 2007.

[59] J. DeVale, "Traditional Reliability," 1998; http://www.ece.cmu.edu/~koopman/des_s99/traditional_reliability/.

[60] Chin-Lung Su, Yi-Ting Yeh, and Cheng-Wen Wu, "An integrated ECC and redundancy repair scheme for memory reliability enhancement," *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on*, 2005, pp. 81-89.

[61] C. Hu and F. Hsu, “United States Patent: 5511020 - Pseudo-nonvolatile memory incorporating data refresh operation,” Apr. 1996.

[62] “AN1819 - Application note: Bad block management in nand flash memories,” May. 2004.

[63] C. Park et al., “A low-cost memory architecture with NAND XIP for mobile embedded systems,” *Hardware/Software Codesign and System Synthesis, 2003. First IEEE/ACM/IFIP International Conference on*, 2003, pp. 138-143.

[64] “AN1822 - Application note: Wear leveling in single level cell nand flash memories,” Nov. 2004.

[65] L. Chang, “On efficient wear leveling for large-scale flash-memory storage systems,” *Proceedings of the 2007 ACM symposium on Applied computing*, Seoul, Korea: ACM, 2007, pp. 1126-1130.

[66] J. Dumas et al., *Théorie des codes : Compression, cryptage, correction*, Dunod, 2006.

[67] R. Hamming, “Error detector and error correcting codes,” *The Bell System Technical Journal*, 1950.

[68] R.H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, John Wiley & Sons, Ltd. (UK), 2002.

[69] C.L. Chen and M.Y.B. Hsiao, “Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review,” *IBM Journal of Research and Development*, vol. 28, 1984, pp. 124-134.

[70] “AN1823 - Application note: Error correction code in single cell nand flash memories,” Nov. 2004.

[71] S. Gregori et al., “An error control code scheme for multilevel Flash memories,” *Memory Technology, Design and Testing, IEEE International Workshop on, 2001.*, 2001, pp. 45-49.

[72] S. Matarress and L. Fasoli, “A method to calculate redundancy coverage for FLASH memories,” *Memory Technology, Design and Testing, IEEE International Workshop on, 2001.*, 2001, pp. 41-44.

[73] N. Achouri, “Techniques d'auto-réparation pour les mémoires à grandes densités de défauts,” Apr. 2004.

[74] T. Kawagoe et al., “A built-in self-repair analyzer (CRESTA) for embedded DRAMs,” *Test Conference, 2000. Proceedings. International*, 2000, pp. 567-574.

[75] Chih-Tsun Huang et al., “Built-in redundancy analysis for memory yield improvement,” *Reliability, IEEE Transactions on*, vol. 52, 2003, pp. 386-399.

[76] Yu-Ying Hsiao, Chao-Hsun Chen, and Cheng-Wen Wu, “A built-in self-repair scheme for NOR-type flash memory,” *VLSI Test Symposium, 2006. Proceedings. 24th IEEE*, 2006, p. 6 pp.

[77] M. Choi et al., “Optimal spare utilization in repairable and reliable memory cores,” *Memory Technology, Design and Testing, 2003. Records of the 2003 International Workshop on*, 2003, pp. 64-71.

[78] V. Schober, S. Paul, and O. Picot, “Memory built-in self-repair using redundant words,” *Test Conference, 2001. Proceedings. International*, 2001, pp. 995-1001.

[79] Xiaogang Du et al., “At-speed built-in self-repair analyzer for embedded word-oriented memories,” *VLSI Design, 2004. Proceedings. 17th International Conference on*, 2004, pp. 895-900.

[80] D. Bhavsar, “An algorithm for row-column self-repair of RAMs and its implementation in the Alpha 21264,” *Test Conference, 1999. Proceedings. International*, 1999, pp. 311-318.

[81] M. Nicolaidis, N. Achouri, and L. Anghel, “A diversified memory built-in self-repair approach for nanotechnologies,” *VLSI Test Symposium, 2004. Proceedings. 22nd IEEE*, 2004, pp. 313-318.

[82] Jin-Fu Li et al., “A built-in self-repair scheme for semiconductor memories with 2-d redundancy,” *Test Conference, 2003. Proceedings. ITC 2003. International*, 2003, pp. 393-402.

Scientific Contributions

U.S Patent

[USPT07] B. Godard, O. Ginez and J.-M. Daga, “Method and System for Providing a Nonvolatile Content Addressable Memory using a single FloTOx Element”, US Patent, USPTO-11650104, January 2007.

[USPT08] B. Godard, J-M Daga, “Error Detecting/Correcting Scheme for Memories”, US Patent, USPTO-12031289, February 2008.

Publications in international conferences proceedings

[DDECS07] B. Godard, J-M Daga, L. Torres, G. Sassatelli, “Architecture for Highly Reliable Embedded Flash Memories”, 10th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, Krakow, Poland, April2007.

[DATE07] B. Godard, J-M Daga, L. Torres, G. Sassatelli, “Evaluation of Design for Reliability Techniques in Embedded Flash Memories”, IEEE International Conference on Design Automation and Test in Europe, Nice, France, April 2007.

[ETS08] B. Godard, J-M Daga, L. Torres, G. Sassatelli, “Hierarchical Code Correction and Reliability Management in Embedded NOR Flash Memories” 13th IEEE European Test Symposium, Verbania, Italy, May 2008.

Publications in national conferences proceedings (France)

[JNRDM07] B. Godard, J-M Daga, L. Torres, G. Sassatelli, “Évaluation de Techniques pour la Fiabilisation de Mémoires Flash Embarquées”, Journées Nationales du Réseau Doctoral de Microélectronique, Lille, France, 14-16 Mai 2007.

Glossary

Abbreviation	Description
ASIC	Application Specific Integrated Circuit
BCH	Bose Chaudhuri Hocquenghem
BER	Bit Error Rate
BL	Bit Line
CG	Control Gate
CMOS	Complementary Metal Oxide Semiconductor
CTMC	Continuous Time Markov Chain
DRAM	Dynamic Random Access Memory
EC	Error Correction
ECC	Error Correcting Code
ED	Error Detection
EEPROM	Electrically Erasable Programmable Read Only Memory
EL	Error Localization
EPROM	Electrically Programmable Read Only Memory
ESP	Essential Spare Pivoting
FG	Floating Gate
FIT	Failure In Time
FLOTOX	FLOating gate Thin OXide
FMEA	Failure Modes and Effects Analysis
FP	Frenkel Poole
GL	Ground Line
HEI	Hot Electron Injection
MCU	Microcontroller Unit
MOS	Metal Oxide Semiconductor
MRAM	Magnetic Random Access Memory
MTJ	Magnetic Tunnel Junction
MTTF	Mean Time To Failure
OUM	Ovonic Universal Memory
PCRAM	Phase Change Random Access Memory
ROM	Read Only Memory
RS	Reed Solomon
SIA	Semiconductor Industry Association
SILC	Stress Induced Leakage Current
SL	Source Line
SoC	System-on-a-Chip
SRAM	Static Random Access Memory
RM	Retrieval Mechanism
TAT	Trap Assisted Tunneling
TLB	Translation Lookaside Buffer
UV	Ultra Violet
WL	Word Line

List of Figures

List of Tables

Techniques de conception en vue d'améliorer la fiabilité des mémoires Flash embarquées

Résumé : Les mémoires non-volatiles de type Flash sont présentes dans un grand nombre de circuits visant des applications électroniques portatives. Leur non-volatilité et flexibilité en font des mémoires extrêmement populaires. Néanmoins, la fiabilité devient une caractéristique à améliorer en raison des besoins en surface grandissants et de leur intégration dans des applications sensibles. Des solutions de tolérance aux fautes peu coûteuses et faciles à intégrer doivent être mises en place. Tout d'abord, cette étude s'est portée sur l'analyse et l'étude de la fiabilité des Flash. Il fut l'occasion d'établir un modèle de fiabilité d'une cellule à grille flottante. Ce modèle a été ajusté suivant les paramètres issus d'une technologie Flash 180nm. Dans un second temps, deux techniques de tolérance aux fautes mêlant codes correcteurs d'erreurs et redondance ont été mises au point. La première technique, nommée correction d'erreurs par analyse de $V_{T,}$ fournit des capacités de correction accrues par l'analyse du niveau de programmation des cellules mémoire. Une étude mathématique puis une architecture de fiabilisation ont été proposées. Dans cette étude, on suppose que des ressources de redondance sont disponibles afin de réparer la mémoire lorsqu'une erreur est détectée. La seconde technique, appelée correction d'erreur hiérarchique, utilise des capacités de correction distribuées dans la mémoire Flash afin de réduire significativement le coût associé à une correction d'erreur avancée. Cette technique a été intégrée dans une architecture de fiabilisation disposant de ressources de redondance. Une étude basée sur les Chaines de Markov à Temps Continu a démontré l'efficacité de cette structure. Ces techniques constituent des solutions alternatives aux schémas standards utilisés dans l'industrie. Elles augmentent significativement le temps moyen à la défaillance du système sans faire exploser la surface requise à l'intégration une structure de tolérance aux fautes.

Mots clés : Fiabilité, mémoire Flash embarquées, sûreté de fonctionnement, tolérance aux fautes, codes correcteurs d'erreur, réparation par redondance.

Design for reliability techniques in embedded Flash memories

Abstract: Flash memories are non-volatile memories present in a growing number of integrated circuits used in portable electronic devices. The non-volatility, low power consumption and flexibility make them extremely popular. Nevertheless, the reliability is a characteristic to improve as far as area needs increase and critical applications are targeted. Effectives fault tolerance solutions that are low cost and that can be easily integrated must be found. In a first time, the work of this thesis was focused on the analysis and the study of Flash reliability. It was the occasion to establish a reliability model for a floating gate cell depending on various parameters. This model was adjusted depending on parameters coming from a 180 nm technology. In a second time, the work was dedicated on the development of two fault tolerance techniques merging error correcting codes and redundancy. The first technique, called error correction by V_T analysis, provides an extended correction capacity by analyzing the charge level of cells. A mathematical study and next, a reliability architecture have been proposed. In this study, it is supposed that a given number of redundancy resources are available to repair the memory as soon as an error is detected. The second developed technique, called hierarchical error correction, uses the fact that correction capacities can be distributed in an original way into the memory to significantly reduce the cost usually associated with advanced error correction techniques. This code has also been integrated in a reliability architecture having some redundancy resources. A mathematical study based on Continuous Time Markov Chain has allowed to demonstrate the effectiveness of this structure. Developed techniques are alternatives solutions to standard scheme used in the industry. They allows to improve significantly the mean time to failure of a system by many times at a reasonable area cost.

Keywords: Reliability, embedded Flash memories, safety, fault tolerance, error correcting codes, redundancy repair.

Université de Montpellier II : Science et Techniques du Languedoc

LIRMM : Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

161 Rue Ada – 34392 Montpellier Cedex 5