

2-8-2011

Built-in controller for self-test and repair of 3D integrated Nand flash memory chip, used in radiation intense space environment

Naveen Nischal Purushotham

Follow this and additional works at: https://digitalrepository.unm.edu/ece_etds

Recommended Citation

Purushotham, Naveen Nischal. "Built-in controller for self-test and repair of 3D integrated Nand flash memory chip, used in radiation intense space environment." (2011). https://digitalrepository.unm.edu/ece_etds/210

This Thesis is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Electrical and Computer Engineering ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.

Naveen Nischal Purushotham

Candidate

Electical and Computer Engineering

Department

This thesis is approved, and it is acceptable in quality
and form for publication:


Approved by the Thesis Committee:



Chairperson



, Dr. James Lyke



, Dr. James F. Plusquellic

BUILT-IN CONTROLLER FOR SELF-TEST AND REPAIR
OF 3D INTEGRATED NAND FLASH MEMORY CHIP,
USED IN RADIATION INTENSE SPACE ENVIRONMENT

By

Naveen Nischal Purushotham

B.TECH., Electronics and Communication,
Jawaharlal Nehru technological University, 2008

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico
Albuquerque, New Mexico

December, 2010

© 2010, Naveen Nischal Purushotham

Dedication

To my father, P. James Ajit, my motivation personified, for his never ending support, encouragement and sacrifice. He has bestowed all his wisdom upon me and taught me to face failures and yet move forward with perseverance till I achieve my purpose.

To my mother, P. Shyamala Ajit for her unconditional love and sacrifice. Without whom I am incomplete.

To my grandmothers, P. Hemalatha Sukumar and P. Helen George for all their prayers and love.

To my sisters, K. Anne Praveen and P. Sowmya Amith for their unconditional support and love

Now thanks be to God who always leads us in triumph in Christ.

- 2 Corinthians 2:14

ACKNOWLEDGEMENT

I would like to thank Prof. Payman Zarkesh-Ha, my advisor and thesis chair, for his guidance, support and encouragement during my Master's program and also for being an outstanding teacher in the classrooms.

I would like to thank Dr. James Lyke, for introducing me to the problem and providing me with many insightful observations and valuable comments.

I am also grateful to Prof. James F. Plusquellic, for accepting to be as a committee member with very short notice. I would like to thank him for his time

I gratefully acknowledge Craig Kief and Dr. Steve Suddarth, my financial supporters, for making my life at graduate school most comfortable by always making sure I had everything I need. I would like to thank Craig for being there to patiently review my work. I am very fortunate to have him during my Master's, without whom it would have not been as successful.

Finally, I convey many thanks to my beloved, Bhavya and my friends Karthik, Shashank, Srinivasan, Bharath and Mallik and everyone else for filling my life with love, happiness and encouragement.

BUILT-IN CONTROLLER FOR SELF-TEST AND REPAIR
OF 3D INTEGRATED NAND FLASH MEMORY CHIP,
USED IN RADIATION INTENSE SPACE ENVIRONMENT

By

Naveen Nischal Purushotham

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico
Albuquerque, New Mexico

December, 2010

BUILT-IN CONTROLLER FOR SELF-TEST AND REPAIR OF 3D INTEGRATED
NAND FLASH MEMORY CHIP, USED IN RADIATION INTENSE SPACE
ENVIRONMENT

By

Naveen Nischal Purushotham

B.TECH. in Electrical and Communication Engineering, Jawaharlal Nehru
technological University, 2008

M.S. in Electrical Engineering, University of New Mexico, 2010

ABSTRACT

The perennial need for memory storage is further increasing with the advancements in technology. Both terrestrial and space related applications thrive on efficient ways and new technology for storing data that is Incorruptible and dependable. Research is continuously carried out on making storage systems vast and reliable. The development of 3D integration has spawned the idea of a new generation of memory based on 3D stackable chips. In an era where there is a continuous demand for larger, faster, denser and robust memories, 3D stackable memory settles in perfectly.

However, technology scaling is having a negative effect on the robustness (low yield and higher sensitivity to radiation effects) of the memories and 3D stackable memory is no exception. As the eagerness for using 3D stackable memory builds up because of its many advantages the major concern that stands as an obstacle for such a system is its yield and system reliability. It is an important consideration in critical applications related to space, avionics, and

defense. Even if a single memory domain fails in a stack of memory modules due to any kind of irregularity, it can lead to a total system breakdown.

The potential dangers for 3D memory include destructive errors such as physical errors in memory arrays, wear out faults, hard errors (e.g. stuck bits) & errors due to memory yield problems and non-destructive errors (soft errors) such as single event upsets (SEU). There is also an increased risk of the 3D technology class memories to be effected by multiple bit upsets (MBU) because of their natural vertical structure. Hence, 3D memory needs an efficient memory controller that can make the memory more reliable and robust against such dangers. The memory controller should not only accomplish the memory accessing, but it also should act as a potential healing system (ability to retrieve the data and exclude the failed memory) for the stack of memory. The concentration of the present work is focused on 3D Nand flash memory. It proposes an efficient and robust memory controller for 3D Integrated Nand flash memory chips used in space radiation environment that performs continuous self test and repair. It revives the system from all single point hard errors and soft errors.

The controller uses a novel and flexible memory mapping scheme (using PUF technology) and appropriate Error correction code (ECC) to protect against the above mentioned errors. The controller is implemented on a FPGA with three different kinds of ECC differing in the correction capability, space requirement, latency (delay) and power requirement. This work serves as a look up table for

space missions with varying mission requirements to make the choice of a particular type of ECC to go with the 3D Nand flash controller.

Table of Contents

	Page
List of Figures	xiii
List of Tables	xvii
Preface	xviii
Thesis Organization	xviii
Contributions of this Thesis	xix
Chapter 1 Introduction	1
1.1. “More than Moore” Approach – Exploring the Third Dimension	1
1.2. Hurdles for 3D Memories in Space Environment	3
1.3. Single Event Upsets and their Mechanism	4
1.3.1. Charge Collection Mechanism for CMOS	6
1.4. Special Concerns for 3D Flash Memory in Space	8
1.4.1. Ionization Damage from High-Energy Electrons and Protons	9
1.4.2. Microscopic Ionization Damage from Heavy Ions	10
1.4.3. Permanent errors – Stuck bits caused due to space radiation	11
Chapter 2 Background	12
2.1. 3D Integration Technology.....	12
2.2. Wafer level 3D memory integration example flow.....	14

2.2.1. Copper Bonding with TSVs for Memory Stacking.....	14
2.2.2. Temporary Bonding and Release.....	15
2.2.3. Wafer thinning.....	15
2.2.4. Vertical Interconnect.....	16
2.2.5. Alignment	17
2.2.6. Bonding	17
2.2.7. Final Metallization and Testing	18
2.3. 3D Memory Advantages	19
2.4. Error Detection and Correction (EDAC) Techniques Used for Memories in Space Applications.....	20
2.4.1. Hamming Code Algorithm.....	21
2.4.2. Reed-Solomon Codes.....	24
2.5. Flash Memory Mechanism	32
2.5.1. Floating Gate Devices.....	33
2.5.2. Charge Injection Mechanism.....	34
2.5.2. a) Hot Electron Injection.....	35
2.5.2. b) Fowler-Nordheim Tunneling.....	36
2.6. NAND Flash Technology and Architecture.....	40

2.7. Physical Unclonable Function(PUF) Technology.....	43
Chapter 3 Memory Controller.....	45
3.1. Introduction.....	45
3.2. Memory Controller Technique.....	47
3.3 Proposed Block diagram of hardware Specification.....	58
3.4. Benefits of the Controller	61
3.5. Limitations	62
Chapter 4 Implementation	65
4.1. Introduction	65
4.2. Hardware Description	66
4.3. Results	72
4.3.1. Functional Timing Simulation Results.....	72
4.3.2. Space and Power Calculation of FPGA	
Chip with 3 ECC Modules.....	78
Chapter 5 Conclusion.....	80
References.....	82

List of Figures

Figure 1.1: Schematic of 3D Assembly [2]	1
Figure 1.2: Heavy Ion vs. Proton Charge Deposition [10]	4
Figure 1.3: Single event Upset [30]	5
Figure 1.4: Electron concentration due to funneling in an n+/p silicon junction following an electron strike [7]	7
Figure 1.5: Alpen Effect [31]	7
Figure 1.6: Change in higher level storage locations in a multi-flash memory after irradiation with heavy ions. [5]	10
Figure 2.1: 3D Integrated Circuit	13
Figure 2.2 Graphical Representation of stacked memory/logic components integrated in a 3D chip. The structure is based on face-to-back stacking technology and TSV interconnection. [1]	14
Figure 2.3: Cross-sectional SEM of Si region with metalized TSV and connecting wiring level [1]	17
Figure 2.4: A close-up of the cross-sectional SEM image of a Cu via successfully bonded to a Cu pad [1]	18
Figure 2.5: Bit parity for p1 and p1' [19]	21
Figure 2.6: Decode schematic outline [21]	26

Figure 2.7: Encoder schematic outline [21]	29
Figure 2.8: Schematic cross-section of FG transistor [23]	34
Figure 2.9: Energy band diagram of a floating gate memory during programming by hot-electron injection [22]	36
Figure 2.10: Hot-electron injection mechanism for programming- in Flash memory [22]	36
Figure 2.11: Energy band diagram of a floating gate memory during programming by FN tunneling [22]	37
Figure 2.12: Uniform tunneling to program Flash memory [22]	38
Figure 2.13: Energy band diagram of a floating gate memory during erasing by FN tunneling [22]	39
Figure 2.14: Uniform tunneling to erase Flash memory [22]	39
Figure 2.15: Nand Flash cell [24]	41
Figure 2.16: Nand Flash device organized as 2048 Blocks [24]	42
Figure 2.17: Typical storage method [24]	43
Figure 2.18: Simple ring oscillator PUF [27]	44
Figure 3.1: Flow chart for Scan and Mark phase Algorithm	50
Figure 3.2: Flow chart for Discovery phase Algorithm	52

Figure 3.3: Bit format for NAND flash page	53
Figure 3.4: Flow chart for Data write request Algorithm	54
Figure 3.5: Flow chart for Data Read request Algorithm	55
Figure 3.6: Flow chart for Heal phase Algorithm	58
Figure 3.7: Block diagram of Hardware Specification for 3D NAND Flash Controller.....	59
Figure 4.1: Block diagram of Hardware Specification for 3D NAND Flash Controller implemented on a Virtex 4 FPGA	66
Figure 4.2: Block Memory in FPGA used to mimic 3D NAND Flash Memory	67
Figure 4.3: Hamming code Algorithm Block diagram [28]	69
Figure 4.4: Block diagram of Reed-Solomon Decoder showing the I/Os [29]	70
Figure 4.5: I/O specification for Reed-Solomon (5,3) Codec	71
Figure 4.6: Functional Timing Simulation of Boot Unit of 3D NAND Flash Controller	72
Figure 4.7: Functional Timing Simulation of Boot Unit of 3D NAND Flash Controller	73
Figure 4.8: Functional Timing Simulation of PUF Unit of 3D NAND Flash Controller	73

Figure 4.9: Functional Timing Simulation of ECC Unit (Hamming code) of 3D NAND Flash Controller	74
Figure 4.10: Functional Timing Simulation of ECC Unit (Reed Solomon (5, 3)) of 3D NAND Flash Controller	75
Figure 4.12: Functional Timing Simulation of ECC Unit (Reed Solomon (255,251)) of 3D NAND Flash Controller	75
Figure 4.13: Functional Timing Simulation of 3D NAND Flash Controller Data write	76
Figure 4.14: Functional Timing Simulation of 3D NAND Flash Controller showing UIDC counter and Mapping register	77
Figure 4.15: Functional Timing Simulation of 3D NAND Flash Controller Data Read	77

List of Tables

Table 2.1: Error Detection Responses	23
Table 2.2: Listing of transformation vectors [21]	32
Table 4.1: Results for 3D NAND Flash Controller	78

Preface

Thesis Organization

This thesis is organized in the following manner. Chapter 1 gives an introduction to 3D integration technology and the potential dangers of the advent of 3D integrated memories into space applications. It also covers the basic physical mechanisms of how a particle strike on semiconductor memories causes an upset in the logic value stored in it. The special concerns for 3D Nand flash memories in space environment are also discussed. Chapter 2 gives a detailed description of 3D integration technology and discusses a wafer level 3D memory integration example flow for better understanding. It also gives an overview of the error correction and detection codes used in traditional nand flash memories that can be exported to 3D nand flash chips. It then explains in brief the flash memory mechanisms and architecture. It concludes with an introduction to PUF technology and explains the mechanism of the ring oscillator PUF being used in the design of the memory controller. Chapter 3 discusses about a new memory controller technique, the algorithms involved in its proposed implementation and the hardware description. This chapter concludes with the discussion of the benefits and limitations of the memory controller technique. Chapter 4 gives the implementation details and results indicating the working of the memory controller technique. It concludes with the indication of the results obtained from power and space requirement calculations of the memory controller implemented with three different kinds of ECC and provides a trade-off

table. Chapter 5 concludes the thesis work with how this thesis work has satisfied a designer's requirement to have a simple yet efficient and robust memory controller for 3D nand flash memories used in intense space radiation environment.

Contribution of this Thesis

[1a] Naveen Purushotham, Srikanth Devarapalli, James Lyke and Payman Zarkesh-Ha, 'Self Healing Adjustable Memory System' AIAA-2010-3373, In Proceedings of AIAA Infotech@Aerospace 2010, Atlanta, Georgia, Apr. 20-22, 2010.

Chapter 1

Introduction

1.1. “More than Moore” Approach – Exploring the Third Dimension

For long, the semiconductor industry has been led by Moore’s law to achieve higher computing power, larger capacity versus lower cost in Integrated Circuits (ICs). The continuous scaling down of device dimensions as per Moore’s law has provided significant performance enhancements in transistors. However, as the demand for higher functionality increases, device scaling is becoming more challenging. The effect of smaller design rules is weakening in many types of ICs. To meet the requirements of the semiconductor rules, major chip manufacturers are investigating 3D IC technologies to stack chips vertically. Packaging based on 3D (Postsingulation of wafers into individual chips) and wafer level 3D integration (3D stacking prior to singulation of wafer into individual chips) is being considered by many companies. Many believe that 3D IC technology will make it possible to maintain the current pace of cost reduction. [1]

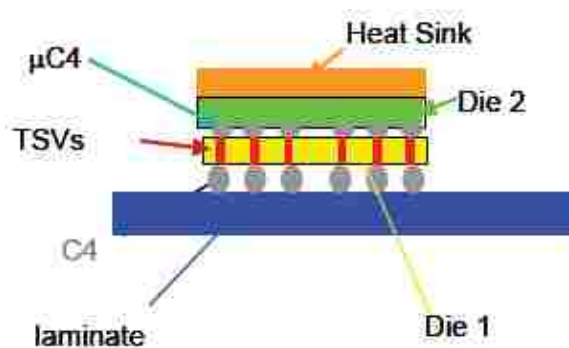


Figure 1.1: Schematic of 3D Assembly [2]

3D Integration has a plethora of potential advantages over the conventional 2D ICs including performance advantages of through-chip micron-sized interchip Vias for high speed multicore processors, high memory capacity with reduced processor- memory latency, heterogeneous integration of mixed signal ICs with high performance interconnects. In Particular, 3D memory devices have vast densities, superior speeds and they consume less power. Figure 1.1 shows a simplified schematic of 3D Integration Technology. With such groundbreaking features, 3D stackable chips are sure to pave their way into the space industry. However, there are certain immediate issues that the 3D ICs have to address namely (1) the overhead and design constraints of through silicon vias (TSVs) [a vertical electrical connection (via) passing completely through a silicon wafer or die]; (2) Power delivery and distribution in multiple strata; (3) heat dissipation across the 3D stack; and finally (4) reparability of the 3D stack. Some believe that the first real application of 3D technology will be in the memory area because it has been shown that the four constraints given above can be managed in memory applications quite well and significant system performance and power benefit can be accrued [2]. Due to the perennial need of memory storage in space related applications 3D memory is sure to take its place in this area.

More details of 3D integration technology are provided in the chapter 2 (for reference) and the rest of this document addresses 3D memory with 3D NAND flash memory being used as the main focus hardware.

1.2. Hurdles for 3D Memories in Space Environment

The harsh radiation environment in outer space makes it difficult for electronics to survive their expected mission duration. Space radiation effects can be broadly classified into 'Hard' and 'Soft' effects. A hard effect (or error) refers to physical damage caused to the silicon lattice structure due to the accumulation of radiation. A soft effect (or error) are not destructive to the device but can affect device availability and reliability. Soft errors (more commonly known as single event effects) refer to bit flips or change in stored data without any permanent damage to the chip. As the eagerness for using 3D stackable memory advances due to its many advantages, the major concern that stands as an obstacle for such a system is its yield and system reliability. It is an important consideration in critical applications related to space, avionics, and defense. If a single memory domain fails in a stack of memory modules due to any kind of irregularity, it can lead to a total system breakdown.

Reparability of the stack is a major reliability issue. Apart from the single event radiation effects the 3D memories are also prone to failures due to number of errors including physical errors in memory cells, wear out faults and stuck bits. Hard permanent destructive (or aging) errors are caused by defects in the silicon or metallization of the processor package, interconnect electro migration, time dependent dielectric breakdown (TDDB), thermal cracking and negative-bias temperature instability (NBTI) [3][4].

Basic influence of natural space radiation environment on flash memories are:

- (1) Macroscopic ionization damage from the interaction of many electrons and protons, producing a buildup of charge in the gate and isolation oxides,
- (2) Transient effects from the interaction of a single galactic cosmic ray or high-energy proton, causing upsets in the state machine, buffer or other digital regions of the flash memory,
- (3) Microscopic ionization damage from the charge produced by a single cosmic-ray heavy-ion in the gate region and
- (4) Microscopic catastrophic damage from high energy protons or galactic cosmic ray particles which can permanently increase the leakage current of the floating gate [5].

These effects are recently being observed in terrestrial environment as a result of technology scaling.

1.3 Single Event Upsets and their mechanism

Single-event phenomena occurs when a single charged particle strikes and deposits energy into space-borne electronics resulting in a fault. The particle

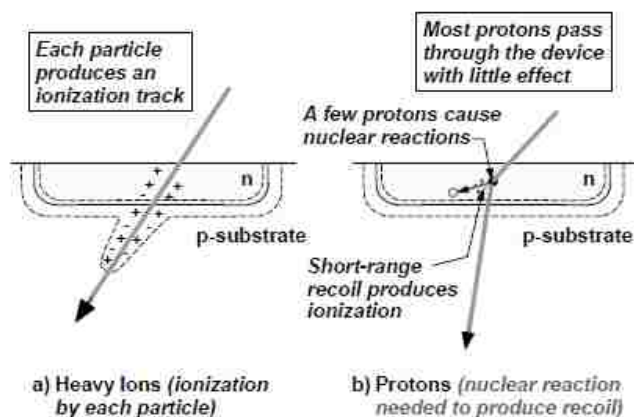


Figure 1.2: Heavy Ion vs. Proton Charge Deposition [10]

strike is broadly classified into by heavy ion strikes (direct ionization) and Proton/neutron strike (indirect ionization). Both the effects can be observed in Figure 1.2.

Indirect Ionization:

This is a process in which a high energy particle such as a proton or neutron strikes the silicon atom resulting in an inelastic (or elastic) collision with the nucleus and releases a heavy ion called “nuclear recoil reaction”. In the case of an elastic collision, spallation reactions can occur in which the target nucleus is broken into two fragments (e.g., Si breaks into C and O ions) or alpha/gamma particles are emitted with recoil of daughter nucleus. These particles being much-

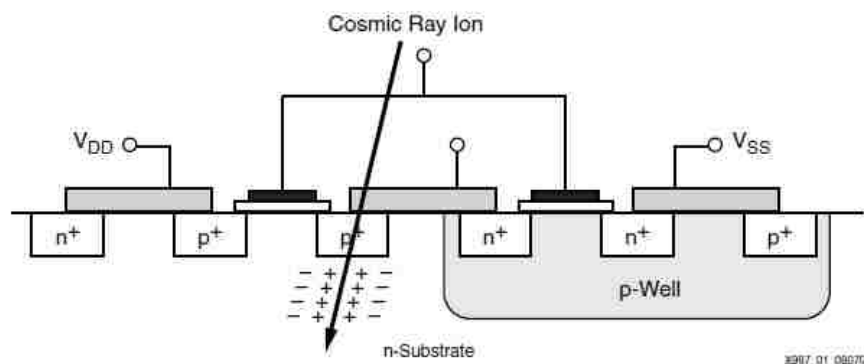


Figure 1.3: Single event Upset [30]

heavier than original proton or neutron deposit energy along their path. They deposit higher charge densities as they travel and therefore may be capable of causing a SEU. Figure 1.3 shows the Single event upset. In the case of an inelastic collision the particle does not travel far from the impact site and results in electron-hole pair generation near the impact area. These collisions typically have low energies [7].

Direct Ionization:

When a heavy ion strikes a semiconductor material it releases electron hole pairs along its path as it loses energy. Particles other than electrons, protons, neutrons and pions which have an atomic number greater than or equal to two can be classified as a heavy ion. Proton/Neutron strikes are generally less effective compared to direct heavy ion strikes. A direct heavy ion strike causes reaction in CMOS technology which is 5 orders of magnitude worse than a nuclear recoil event from protons and neutrons.[7][8]

1.3.1 Charge Collection Mechanism for CMOS

When a particle strikes a microelectronic device, the most sensitive regions are usually reverse-biased p/n junctions. The high field present in a reverse-biased junction depletion region can collect most of the particle-induced charge through drift processes, thereby resulting in a transient current at the junction contact. The drift process is a major mechanism that causes SEUs. However, the more important factor is the diffusion process (electrons diffusing from substrate to drain/bulk potential barrier), which contributes to the late time collection of the current at the struck node ensuring that a bit stays flipped.[7][9] Strikes on a depletion region can cause the carriers to diffuse into the vicinity of the depletion region field where they can be efficiently collected. This process of temporary depletion region extension is referred as funneling.

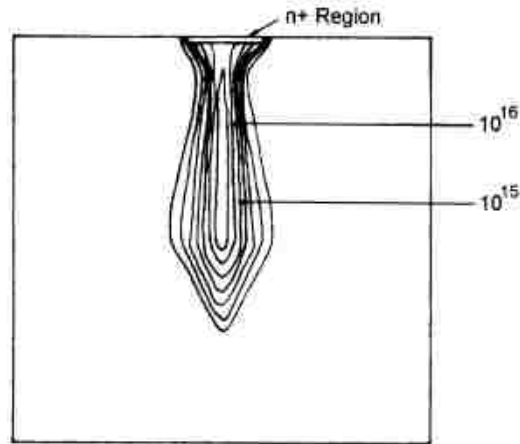


Figure 1.4: Electron concentration due to funneling in an n+/p silicon junction following an electron strike [7]

This funneling effect can increase charge collection at the struck node by extending the junction electric field away from the junction and deep into the substrate, such that the charge deposited some distance from the junction can be collected through the efficient drift process. Figure 1.4 shows the electrons concentration due to funneling [7].

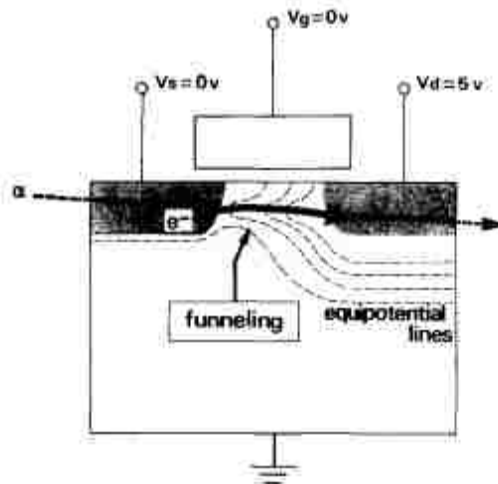


Figure 1.5: Alpen Effect [31]

The charge collection mechanism in submicron devices results from a disturbance in the channel potential of the device, referred as funneling effect. The effect is triggered by a particle strike that passes through both the source and the drain at near-gazing incidence as shown in Figure 1.5. Such a strike causes a significant source drain conduction current that mimics the “on” state of the transistor. This phenomenon is called the ALPEN effect [7]. ALPEN effect tends to increase as the channel length decreases.

Another effect known as the bipolar transistor effect is caused due to injection of electrons over the source/well barrier. For example, in an n-channel MOSFET, holes left in the well due to a particle strike raise the well potential, effectively lowering the source/well potential barrier. This lowered potential barrier causes the source to inject electrons into the channel. These electrons can be collected at the drain effectively increasing the original particle-induced current. This current increases the SEU sensitivity. Because the electrons are injected over the source/well barrier, this is referred to as a bi-polar transistor effect, when the source acts as the emitter, the channel as the base region, and the drain as the collector. Reducing the channel length effectively decreases the base width, and the effect becomes more pronounced [7].

1.4 Special Concerns for 3D Flash memory in space

Flash memories have unique design requirements that cause them to be more susceptible to radiation damage than conventional microelectronics. The high voltages required for erase and write operations require that some internal transistors are designed with thicker gate oxides and more lightly doped channel

regions compared to conventional digital logic transistors making them far more susceptible to radiation damage. The charge pumps that are required to generate the high voltages for erasing and writing are generally the most sensitive circuits. Single event upsets are far more difficult to deal with. The very complicated device architecture used in advanced flash devices causes their basic functionality to be affected by heavy ions and protons, and it is difficult to recognize and categorize these types of upsets because of limited visibility of internal operating conditions. In some cases the device does not completely function, but in some cases the error may be difficult to detect such as errors in buffers and page address registers. Fortunately, the overall error rate for those type of malfunctions is relatively small, allowing system mitigation with Error Correction and Detection EDAC [5].

Permanent errors are far more difficult problem for several reasons. First, tests for permanent errors are difficult and costly to perform. Second, some types of errors may occur even for devices that are not biased during the time that a heavy ion strike occurs. EDAC is a viable way to recognize this type of error.

1.4.1 Ionization damage from high-energy electrons and protons

In flash memories there is a macroscopic effect of charge build up in the gate and isolation oxides due to the influence of space radiation. Some of the excess charge is trapped at the interface region, changing the threshold voltage of the transistor. For the basic MOS transistor, the shift in threshold voltage scales with the square of oxide thickness. This has reduced the effect of total ionizing dose

in highly scaled devices. However, in flash memories the points of concern are the internal transistors used for the charge pump and erase/write control which have much thicker oxides for the requirement of high voltage giving them more cross section and making them more sensitive to total dose effects (and damage) compared to other technologies[5].

1.4.2 Microscopic Ionization damage from heavy ions

Microscopic ionization damage from heavy ions has been observed in multi-level flash devices. Such damage results in the increase of the cell leakage current. The net effect is the shift of the internal state of some of the internal memory cells. The most straightforward way to study this kind of effect is to examine the threshold voltage distribution within the device.

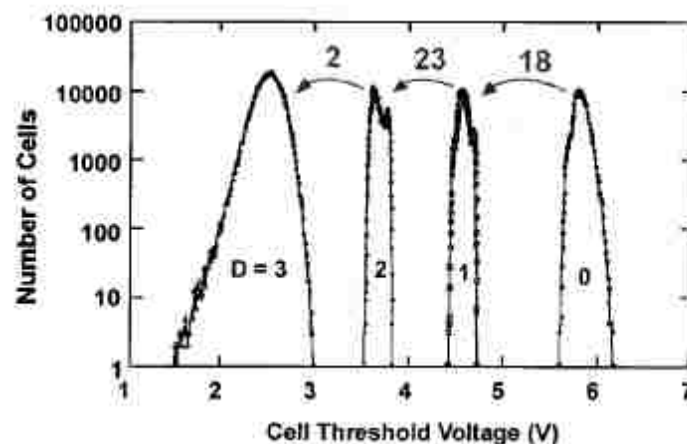


Figure 1.6: Change in higher level storage locations in a multi-flash memory after irradiation with heavy ions. [5]

In Figure 1.6, 18 cells with highest internal storage level were shifted to the next level, 23 from the second highest to the third highest, and 2 from that level to the

ground level. This is caused by microdose that changes the storage level of the charge “packet” within the floating gate. Although this changes the stored information, it does not cause permanent change in the device; rewriting the cell can be done to restore the information [5].

1.4.3 Permanent errors – Stuck bits caused due to space radiation

Permanent changes in thin oxide can occur due to the detailed interaction of intense ionization track from a single heavy ion with the gate region which produces localized damage in the structure of the gate. This effect can cause the leakage current to increase by several orders of magnitude. Although the change in leakage current is small, it can affect the ability of the floating gate to store charge over long periods. This is a permanent effect, and may limit the ability to use flash memories in space. The magnitude of the leakage current depends on the thickness of the gate region as well as the quality of the oxide. Although high electric fields are required in order to get large current changes (hard breakdown), soft breakdown can occur at very low electric fields and required only a single ion strike [5].

Chapter 2

Background

2.1 3D Integration Technology

What is a 3D IC?

A 3D IC contains at least 2 semiconductor layers and at least one layer of horizontal interconnect wiring between the semiconductor layers. Each semiconductor layer contains its own functional elements (transistors, diode, etc.) with electrical junctions. The semiconductor layers may be built sequentially or separately. Elements on the semiconductor layers are connected vertically with TSVs (through-silicon vias) or other vias, without employing the I/O drivers and associated electrostatic discharge (ESD) structures that are normally used to connect off-chip devices. [1]

3D integration refers to a variety of technologies that provide electrical connectivity between stacked multiple active device planes. There are a number of technology options to arrange ICs in a vertical stack. It is possible to stack ICs in a vertical fashion at various stages of processing: 1) Postsingulation 3D packaging (chip-to-chip), and 2) Presingulation wafer-level 3D integration (chip-to-wafer, wafer-to-wafer, and monolithic approaches).[1]. The most promising 3D technology is the wafer-level BEOL-compatible 3D integration technology. It is enabled by wafer alignment, bonding, thinning and inter-wafer interconnections. It uses TSVs to realize die-to-die interconnection. A 3D chip has a base stratum that interfaces to a laminate with one or more semiconductor strata vertically

attached to this base stratum. Power is supplied to the base layer from the laminate and all I/Os communicate to the laminate through the base layer. This

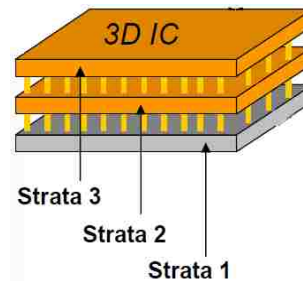


Figure 2.1: 3D Integrated Circuit

thinned base stratum is assumed to be mounted face down (though this is not a requirement) and for illustrative purposes, connected to the laminate through conventional bump connections (also called C4 connections). The second stratum is then mounted face-down on the first stratum. There are a variety of technological choices at this stage. One of which is a die-to-die connection using known good die, through the use of μ bumps and Through Silicon Vias (TSVs). The density of these connections is often discussed in terms of TSV pitch. The TSV diameter is a very critical parameter. A rule of thumb is that the TSV diameter must be of the order of the strata thickness. When the TSV diameters are in the $25\mu\text{m}$ range, it is possible to use solder based joining technology and mechanical alignment to perform die-to-die joining. When the TSV diameters drive down to below $10\mu\text{m}$ range, the joining process needs to be integrated with the silicon fabrication process using wafer scale processing rather than die based processing. This latter phase is needed to realize the full potential of 3D integration. The density offered by the $25\mu\text{m}$ TSVs is adequate for most

immediate memory applications that we are interested in and is a good place to start. However, the true potential of 3D integration can only be realized when the inter-strata interconnect pitch approaches sub $10\mu\text{m}$ [2].

2.2 Wafer level 3D memory integration example flow – IBM 3D integration approach [1]

An example of IBM 3D Integration Approach fabricated using a face-to-back layer transfer process is described in this section, which is a strong potential candidate for replacement of traditional planar circuit layout to enable future advanced CMOS technologies.

2.2.1 Copper Bonding with TSVs for Memory Stacking

As depicted in Figure 2.2, the “face-to-back” approach can be used to connect functional blocks of CMOS circuitry. More specifically, this method uses logic and memory components that traditionally (in 2D chip layout) reside side by side, and

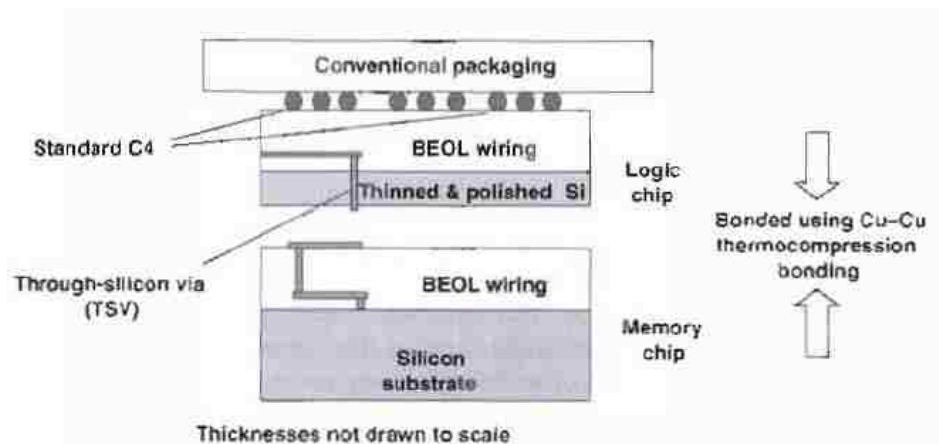


Figure 2.2 Graphical Representation of stacked memory/logic components integrated in a 3D chip.

The structure is based on face-to-back stacking technology and TSV interconnection. [1]

stacks them on top of each other providing a 3D IC solutions. The resulting “sandwich” components enable reduced overall chip/package size and increased speed of data flow among the various functional blocks of the chip.

2.2.2 Temporary Bonding and Release

In the “front-to-back” layer transfer approach, the use of carrier wafers is often required to protect the circuit layers and to provide mechanical stability during the layering process. IBM has historically used a glass handle wafer which has a thermal coefficient of expansion (TCE) matched to silicon. In this process, after the top wafer is fabricated and metalized a protective coating (usually nitride layer) is deposited, followed by the application of polymer-based layers that act as a bonding medium. Also, once the layer transfer is completed the detachment of the carrier has to be easy provided for reliable, fast, and cost-effective methodology. If the top wafer layer is not thinned below $\sim 100 \mu\text{m}$ it is possible to omit the handle wafer process all together, but protective thick layers have to be deposited to shield the circuitry during the thinning process (when wafers are attached to the grinding or polishing wheels and exposed to strong mechanical forces)

2.2.3 Wafer thinning

Wafer thinning is a necessary component of 3D integration as it provides capability of bringing layers closely together. However, its biggest challenge is the necessity of thinning a full Si wafer down to $\sim 5\%$ - 10% of its original thickness, with a required uniformity of $\sim 1\text{-}2 \mu\text{m}$. This is because, when bulk Si wafers are used in the creation of the integrated CMOS circuitry, there is no

natural etch stop like in the SOI- wafer case, and hence some of the original Si remains after thinning of the wafer is complete. The final thickness depends on thinning process control capabilities and is limited by the thickness-uniformity specifications of the silicon removal process (that being mechanical grinding and polishing, wet or dry etching), Successful thinning to a uniform Si thickness of few microns has been demonstrated, but usually $\sim 30\text{-}60\ \mu\text{m}$ of Si remains.

2.2.4 Vertical Interconnect

Perhaps the most important technology element for 3D integration is the 3D inter-connect. This is sometimes referred to as the TSV or the through-silicon inter-connect though in the case of our SOI 3D scheme, the via does not need to go through silicon since the substrate is removed. A vertical interconnect is necessary for 3D integration to truly take advantage of 3D for system-level performance. Without it interconnects would be limited to the periphery of the chip, and in this case, the interconnect density would be no greater than in conventional 2D technology. This interconnection method is essentially the same as a contact hole (or back-end-of-the-line (BEOL - like) process, with the difference that a much deeper hole has to be created vertically through the silicon material using a special etch process. IBM, Samsung, Tessera, Intel, Elpida, IMEC, and others are developing their TSV methodology optimizing the patterning and metallization process for their applications.

A variety of vertical through-silicon interconnect technologies have been developed by IBM and have been described in the literature [11, 12, 13]. Figure 2.3 shows a scanning electron micrograph(SEM) of a Via.

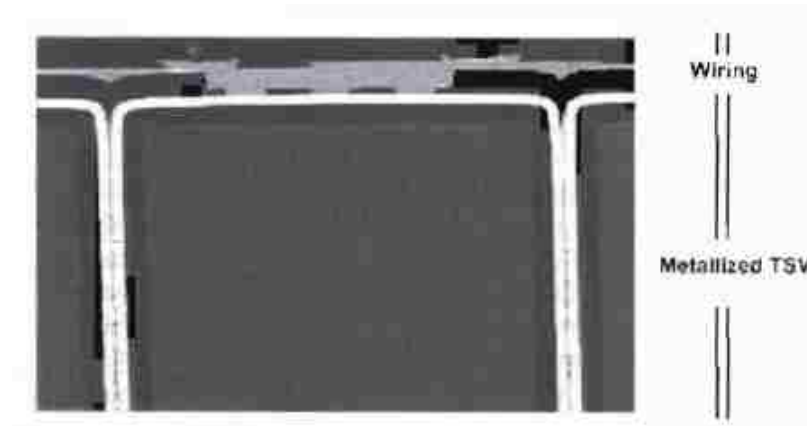


Figure 2.3: Cross-sectional SEM of Si region with metalized TSV and connecting wiring level

[1]

2.2.5 Alignment

In the TSV approach to 3D integration, the alignment requirement is not stringent. An alignment tolerance of $\sim 3\mu\text{m}$ is often sufficient, using Cu-Cu bonding pads of $\sim 10\mu\text{m}$, a tolerance which is within the capabilities of a standard alignment tool for bonding applications. However since this is a wafer-to-wafer 3D integration scheme, alignment and bonding are still key fabrication challenges. The key to good process control is the ability to separate the alignment and pre-bonding steps from the actual bonding process. Such integration design allows for better understanding of the final alignment error contributions.

2.2.6 Bonding

Attachment of two of the functional levels of this 3D structure is completed by using a metal bonding process. This process is chosen as it not only secures two parts together but at the same time it provides for electrical connection between

them. More specifically, copper-to-copper (Cu-Cu) thermocompression bonding is used. Optimization of the quality of this bonding process is a key issue being addressed and includes provision of various surface preparation techniques. Postbonding straightening thermal annealing cycles, as well as use of optimized pattern geometry [14,15, 16]. These solutions combined with use of various seal designs address a key challenge of 3D technology, namely reliability and manufacturability of 3D ICs and packages. Figure 2.4 shows a close-up of the cross-sectional SEM image of a Cu via successfully bonded to a Cu pad.



Figure 2.4: A close-up of the cross-sectional SEM image of a Cu via successfully bonded to a Cu pad

[1]

2.2.7 Final Metallization and Testing

In the case of stacking memory and logic, or any other applications in which more complicated circuits/systems are involved, one has to consider an appropriate design for 3D interconnects (including choice of geometry and metallization) and optimized layout for metal levels of the various components. The critical challenge is to build 3D elements in parallel to a point where they can be pre-tested before layer transfer so that their functionality and fabrication process is verified. Then one has to be able to re-test circuit components right

after their vertical stacking so that any problems may be corrected before 3D parts are processed through any additional steps. To implement this reliability procedure the use of additional verification test circuits are often implemented.

2.3 3D memory advantages

Three-dimensional IC memories can provide a wide array of benefits that are catching the attention of many. They provide independent benefits like higher density, lower weight, lower susceptibility to soft errors, and addition of significantly new functionalities. Apart from these, there are also inter-related benefits which cause design trade-offs such as lower power, faster clock speed, lower latency, and lower cost.[17] Overall, the combined cost savings derived from a 3D IC memory can be as high as 50%. Certainly, some cost is added as each layer is bonded and thinned, and as TSVs must be added to all but one of the layers in the stack. This said, the cost reductions far outweigh the added stacking costs. The savings come from four major sources: memory wafer processing, array efficiency, testing and yield [1] [17].

3D memory provides enhanced reparability. Redundant elements can be shared from layer to layer within a 3D stack. As an example, if a repair requires more replacement than are available on the local array they can be borrowed from other layers. For clustered defects caused by particles, the ability to use spares on other layers significantly improves reparability. Other benefits can also be realized from 3D memories like lower soft error rates as a result of built in ECC as well as from thinner substrates, which provide less interaction volume for

incident particles or energies. Density improvements are obvious as each additional layer of memory in the 3D stack adds a mere 12um in height.

2.4 Error Detection and Correction (EDAC) techniques used for memory in space applications.

In Space computer systems, the contents of memory are protected by an error detection and correction (EDAC) code. Bit-flips caused by single event upsets (SEUs) are a well-known problem in memory chips and EDAC codes have been an effective solution to this problem.

An Error Detection and Correction (EDAC) or error-correcting code (ECC) is a system of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in different memories such as RAM, Flash memory, Hard disks, etc.

There are two basic types of Error Correction Codes: Block Codes and Convolution codes. Block codes are referred to as (n, k) codes. A block of k data bits is encoded to become a block of n bits called a code word. In convolution codes, the code words produced depend on both the data message and a given number of previously encoded messages, and the encoder changes state with every message processed. The length of the code word is usually constant. NAND Flash memories typically use Block Codes. NAND Flash includes extra storage on each page to store ECC code as well as other information for wear-

leveling, logical to physical block mapping, and other software overhead functions. The size of extra storage (spare area) is normally 16 byte per 512 byte sector but other sizes are also used. ECC algorithm correction strength (number of bit errors that can be corrected) depends on the ECC algorithm used to correct the errors (these algorithms may be implemented in either hardware or software). Simple Hamming codes can only correct single bit errors. Reed-Solomon code can correct multiple bit errors and is used on many of the current controllers. [18]

2.4.1 ECC Hamming code algorithm

The Hamming algorithm is relatively straightforward and easy to be implemented in software or hardware. The limitation of the Hamming algorithm is its limited error correction abilities. Hamming code is able to correct single bit errors and detect two bits errors.

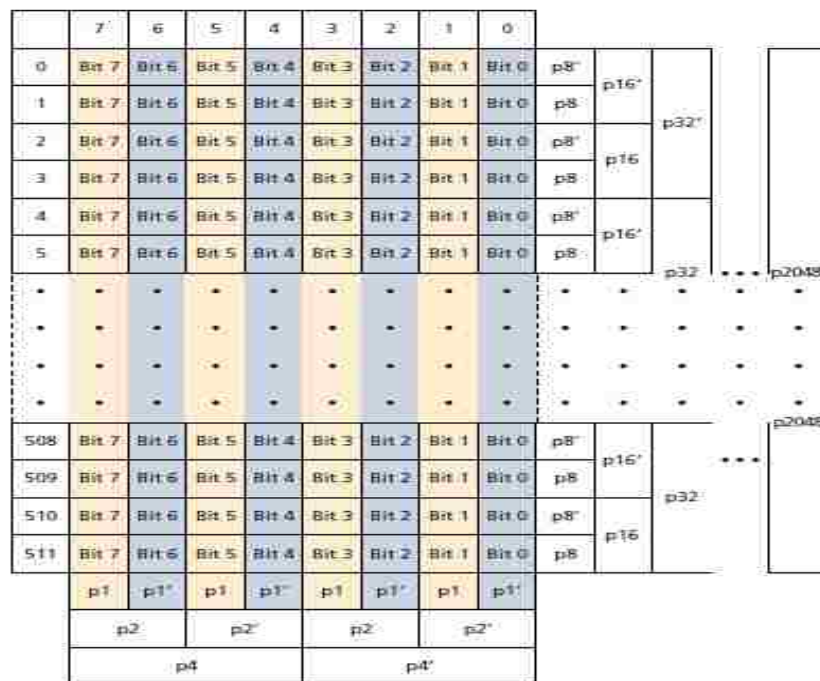


Figure 2.5: Bit parity for p1 and p1' [19]

The ECC algorithm used by the NAND Flash ECC module is based on block parity. The Hamming code for a 512-byte block uses 24 bits of coding information. Figure 2.5 show that p1 is the parity of every 7, 5, 3, and 1 bit of each byte of the 512 bytes in the sector. It shows that p1' is the parity of every 6,4,2 and 0 bit of every byte in this 512 bytes data sector while p1 is the parity of every 7,5,3,and 1 bit of each byte. The same principle holds for the value of bit p2, p2', p4, and p4' with the only difference being in use of different bits for parity generation. Their expression can be simply written as:

$$P1' = \text{bit6 XOR bit4 XOR bit2 XOR bit0}$$

$$P1 = \text{bit7 XOR bit5 XOR bit3 XOR bit1}$$

$$P2 = \text{bit7 XOR bit6 XOR bit3 XOR bit2}$$

$$P2' = \text{bit5 XOR bit4 XOR bit1 XOR bit0}$$

$$P4 = \text{bit7 XOR bit6 XOR bit5 XOR bit4}$$

$$P4' = \text{bit3 XOR bit2 XOR bit1 XOR bit0}$$

On the other hand, the remaining bits of those 24 bits of coding info are somehow derived in a different way in terms of the byte usage. In order to calculate such bits, an intermediate variable is necessary and defined as

$$\text{Rowparity} = \text{bit7 XOR bit6 XOR bit5 XOR bit4 XOR bit3 XOR bit2 XOR bit0}$$

Thus, bit p8' can be easily defined as the XOR of the row parities of every even-numbered (0,2,4,6,8...)row in the 512-byte matrix as shown in figure 7. Similarly, bit p8 will be defined as the XOR of the row parities of every odd-number row (1, 3, 5, 7, ...) in the matrix.

Hence, if each parity bit is updated and serially calculated as every byte is transferred into the ECC module, a typical bit can be described as follows:

$$P8' = P8' \text{ XOR (Rowparity AND not (rownumber (0)))}$$

Where the row number is the number of the rows (i.e., the leftmost column in figure 2.5). In this way, the other bits of hamming code can also be calculated by changing the corresponding row number when each byte of raw data is steamed in. To detect and correct errors, two sets of 24-bit Hamming codes are XORed to get the result. One of them is obtained by reading previously stored ECC result in the NAND flash device, and the other set comes from the result calculated with the data being transferred into the Hamming code block. The error detection responses are listed in Table 2.1.

Response	Description
No error	If the results is 0x00000000. Then There's no error as two sets of hamming code is same
1 bit error	If the result shows sets of 0s and 1s like 0x11111111, an error exists.
2 or more bit errors	More than one error exists if a random pattern of 0s and 1s are generated like 0x01101110

Table 2.1: Error Detection Responses

This approach of error detection is effective for two reasons. First, the bits used to calculate the parity for p_n and p_n' are disjointed sets which means that if one bit is used to calculate the parity for p_n , it will not be used to calculate the parity for p_n' . Suppose that one certain bit is flipped, it can be in the set of bits that calculate the parity for p_n or those that calculate parity for p_n' . Hence, only one of p_n or p_n' can be in error; the other one will be correct. Secondly, in each pair of

pn and pn' there must be a '1' and a '0' because of the bit used for calculating the parity. If a bit is flipped, it was used to calculate the parity of either pn or pn'. Thus, one of the two bits has to be erroneous [19].

2.4.2 Reed – Solomon codes

This section introduces byte-wide single error-correcting and double error detecting (255,252) cyclic Reed-Solomon block code in a very digital logic jargon as opposed to coding theory. A common procedure is used for both encoding and decoding, which is analogous to an arithmetic check-sum [20]. This formulation defines a (255,252) block code. The encoder accepts a block of bytes, to a maximum of 252 bytes. The encoder calculates three parity bytes from this data, and appends these bytes to the data, forming a complete block of up to 255 bytes. On reading the entire block from memory the decoder processes all the bytes to determine three syndrome bytes, from which error detection and correction may then follow. The transform is explained in two steps, it is best to first assume that a valid cyclic block exists whose construction will be explained in the second step. So we first look at the factored decoder and look in detail about the encoding scheme.

Factored Decoder:

Within a given, possibly erroneous code block, the notation V'_1 describes an individual l^{th} byte. The indices l from 0 to 2 identify the parity bytes, previously created by an encoder, while indices in the range from 3 up to $N-1$ identify the data bytes. A maximum value to N is 255.

Factored decoding in software is straightforward. Three syndrome bytes are derived: S_0 , S_1 , and S_2 . Derivation of each S_j involves a sequential 'look-up-table', either implemented in software, or by discrete logic. Starting with the highest-order byte and working downwards, the EX-OR sum of each byte and the latched output of a look-up table forms the address of the next look-up to that same table. Exactly the same procedure is used for each syndrome. Only the tables are different in each case.

$$\begin{aligned}
 &S_0 \leftarrow 0: S_1 \leftarrow 0: S_2 \leftarrow 0 && (1a) \\
 &\text{FOR } I = N - 1 \text{ DOWNT}O 0 \\
 &S_0 \leftarrow V'_1 \text{ (EX-OR) } F_0(S_0) \\
 &S_1 \leftarrow V'_1 \text{ (EX-OR) } F_1(S_1) \\
 &S_2 \leftarrow V'_1 \text{ (EX-OR) } F_2(S_2)
 \end{aligned}$$

In the above ' \leftarrow ' is an assignment operator, while '(EX-OR)' represents a byte-wide EXOR. The functions $F_j()$ represents the action of the look-up tables. A mathematical linkage exists over the entire code block, comprising data and check bytes. This linkage is being tested in three different ways. Calculation of syndrome bytes S_0 , S_1 , and S_2 can be in any order.

The total storage required for each table is just 256 bytes, while a complete calculation requires just 3 times N EXORs and 3 times N look-ups. In the theory of RS codes deriving this set of syndromes is a standard first step towards decoding. In the absence of error, which is hopefully the usual situation, all syndromes will calculate to zero. That would complete the decoding action. If one

or more syndromes are non-zero then an error has been detected. If all the S_j are non-zero then it may be possible to locate and identify a single error byte.

Confirmation and location of a single correctable byte is as follows. The process requires two further tables, and is likely always to be implemented in software. Each syndrome pattern is used as an 8-bit address to one inverse Galois field IGF() table. Stored in this table, against every address, there is a corresponding natural number, technically an exponent, in the range 0 to 254.

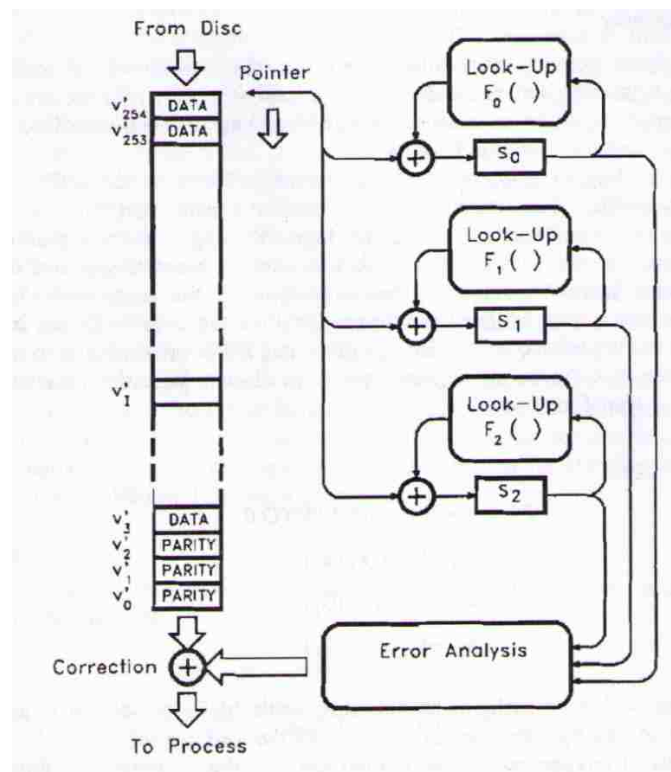


Figure 2.6: Decode schematic outline [21]

This IGF() look-up table, as addressed by byte-wide non-zero syndrome patterns (S_0 , S_1 and S_2), outputs natural numbers which are processed to give two independent estimates of the error locations, with values i and j .

The calculations are in conventional arithmetic, and read

$$i \leftarrow (\text{IGF}(S_2) - \text{IGF}(S_1) + 255) \text{ MOD } 255 \quad (1b)$$

$$j \leftarrow (\text{IGF}(S_1) - \text{IGF}(S_0) + 255) \text{ MOD } 255$$

They identify and confirm the locations of the erroneous byte, provided that $i=j$.

The error pattern itself may be computed from an equation

$$E \leftarrow \text{GF}((\text{IGF}(S_0) - i + 255) \text{ MOD } 255) \quad (1c)$$

Where $\text{GF}()$ is the second table, this time denoting a Galois field. This second table accepts a natural number in the range 0 to 254 as an address and yields a specific non-zero 8-bit error pattern. The process completes with the correction of the offending byte in the i th location with

$$V_i \leftarrow V_i \text{ (EX-OR) } E \quad (1d)$$

If only some of the syndromes are non-zero, or the calculated values i and j are unequal, then the decoder has detected some uncorrectable error pattern, which must be duly flagged as such. Figure 2.6 shows the outline of the decoding process. Summarizing:

<u>Non-zero syndromes?</u>			<u>Action/decision</u>
S_0	S_1	S_2	
N	N	N	no error detected
Y	N	N	uncorrectable error
N	Y	N	uncorrectable error
N	N	Y	uncorrectable error
N	Y	Y	uncorrectable error
Y	N	Y	uncorrectable error

Y Y Y ----- single error may be correctable (if $i=j$)

Factored Encoder:

Having reviewed a (standard) method of factored decoding, the equivalent procedure for encoding may now be described. Using the same standard algorithm, or discrete logic, as used in the decoder, remainders R_H are derived from the data bytes. We adopt a vector notation in which the symbol M_I represents a data byte, with index I in the range from 0 to $K-1$. The maximum value of K is 252.

In pseudo-code the algorithm is

```

R0 ← 0: R1 ← 0: R2 ← 0:
For I = K - 1 DOWNTO 0
    R0 ← MI (EX-OR) F0(R0)      (2a)
    R1 ← MI (EX-OR) F1(R1)
    R2 ← MI (EX-OR) F2(R2)
NEXT I

```

Up to this point the process is seen to be almost exactly the same procedure as that used for finding the syndromes S_H in the decoder (Figure 2.7). The only difference is that the derivation is (obviously) over three less bytes, since only the data bytes exist at this stage. These derivations of the R_H , if done in conventional, sequential software, can be in any order. The calculations require 3 times K EXORs and 3 times K look-ups for R_0 to R_2 . None of these remainder

bytes are of the correct parity, and must not be adjoined to the data bytes, as they stand.

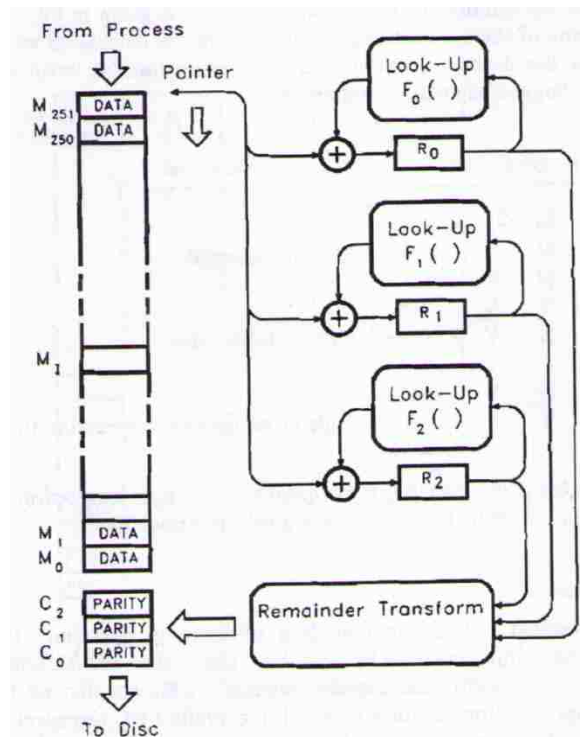


Figure 2.7: Encoder schematic outline [21]

The remainder bytes need to be transformed to the correct check bytes as required by the cyclic code formulation. It is easiest to envisage a 24-bit vector R , whose 8 least significant bits are taken from R_0 , the 8 middle significant bits are from R_1 and the 8 most significant bits are from R_2 . Notationally then

$$R = (R_0, R_1, R_2)$$

The remainder transform then runs as follows: read a bit at a time from R ; if and only if the k th bit $r[k]$ in R is a '1' then the algorithm selects the k th vector $C[k]$ in a given Table 2.2 and EXOR adds this selected row to a 24 bit register of an accumulating vector C . In pseudo code this reads

$$C \leftarrow 0 \quad (2b)$$

```

FOR  $k \leftarrow 0$  to 23
  IF  $r[k] = 1$  THEN  $C \leftarrow C \text{ (EX-OR)} C[k]$ 
NEXT  $k$ 

```

The operation is completed by partitioning the calculated vector C into three bytes. The 8 least significant bits go to C_0 , the middle most significant bits go to C_1 and the 8 most significant to C_2 . That completes the encoding. Formally the unchanged data bytes are 'promoted' up three steps when assigned to the code block i.e.,

$$V_{l+3} = M_l \text{ over } 0 \leq l \leq K \quad (2c)$$

The check bytes C_l then are formally assigned to the 'bottom' locations of the code block (as in Figure 2.7) as

$$V_l = C_l \text{ over } 0 \leq l \leq 3 \quad (2d)$$

In practice there is no need to waste time in physically shifting the data bytes as implied by (c). They may be loaded to memory in one 'go' and without further manipulation. The transform is far simpler to implement than to describe. The sequence requires on average only 12 look-ups and 12 3-byte wide EXOR additions, to finish encoding the complete code block. Its use is therefore trivial in computational load.

Essentially because the sub-set of all possible data bytes, which yield a common set of remainders, uniquely maps to a common set of three correct check bytes. It is required only to 'look-up' the latter via the former. Suppose for example that the following remainders are derived from some given data

$$\mathbf{R}_0 = 10000000 \quad \mathbf{R}_0 = 00000000 \quad \mathbf{R}_0 = 00000000 \quad (3a)$$

We would then form a 24 bit vector

$$\mathbf{R} = 100000000000000000000000 \quad (3b)$$

Table 2.2 then predicts that the correct check sequence would just be $C = C[0]$; in other words a vector

$$\mathbf{C} = 01110110 \quad 10001010 \quad 10001100$$

which partitions as

$$\mathbf{C}_0 = 0111011 \quad \mathbf{C}_1 = 10001010 \quad \mathbf{C}_2 = 10001100 \quad (4a)$$

and this would be true of every member of that sub-set of data which happens to generate that particular remainder vector R .

There are 23 other possible vectors \mathbf{R} with just one '1' located in some k th position of that vector and the rest of the bits are zero. Any data subset which happens to give rise to such a vector automatically has a correct check byte selected from the k th row in Table 2.2. More usually an arbitrary data collection which creates an arbitrary vector \mathbf{R} (with more than one '1' usually) requires just the linear combination of selected rows from Table 2.2. So Table 2.2 is universally applicable for all possible data. [21].

Select remainder bit $r[k]$

Bit no. k	3-byte vectors		
	$a_0 \dots a_7$	$a_8 \dots a_{15}$	$a_{16} \dots a_{23}$
0	01110110	10001010	10001100
1	00111011	01000101	01000110
2	10100101	10011010	01000110

3	11101010	01001101	10101001
4	01110101	10011110	11101100
5	10000010	01001111	01110110
6	01000001	10011111	00111011
7	10011000	11110111	10100101
8	10011110	00001000	10001010
9	01001111	00000100	01000101
10	10011111	00000010	10011010
11	11110111	00000001	01001101
12	11000011	10111000	10011110
13	11011001	01011100	01001111
14	01101010	00101110	10011111
15	01101010	00010111	11110111
16	11101010	10011110	01110110
17	01110101	01001111	00111011
18	10000010	10011111	10100101
19	01000001	11110111	11101010
20	10011000	11000011	01110101
21	01001100	11011001	10000010
22	00100110	11010100	01000001
23	00010011	01101010	10011000

Table 2.2: Listing of transformation vectors [21]

2.5 Flash Memory Mechanism

This section presents a basic overview of flash memory and its mechanism. Flash memory programming and erasing mechanisms using techniques such as Hot-carrier injection and Fowler-Nordheim tunneling are covered.

2.5.1 Floating Gate Devices

Non-volatile memory market share has been continuously growing in the past few years and further growth is foreseen, especially for Flash memories (in which a single cell can be electrically programmable and a large number of cells – called a block, sector, or page – are electrically erasable at the same time) due to their enhanced flexibility against EPROM's. Nonvolatile memory does not lose its data when the system or device is turned off. A nonvolatile memory (NVM) device is a MOS transistor that has a source, a drain, an access or a control gate, and a floating gate. It is structurally different from a standard MOSFET in its floating gate, which is electrically isolated, or "floating".

In flash memories, electrons were transferred from the floating gate to the substrate by tunneling through a 3 nm thin silicon dioxide (SiO_2) layer. Tunneling is the process by which an NVM can be either erased or programmed and is usually dominant in thin oxides of thicknesses less than 12 nm. Storage of the charge on the floating gate allows the threshold voltage (V_T) to be electrically altered between a low and a high value to represent logic 0 and 1, respectively. In floating gate memory devices, charge or data is stored in the floating gate and is retained when the power is removed. All floating gate memories have the same generic cell structure. They consist of a stacked gate MOS transistor as shown in Figure 2.8.

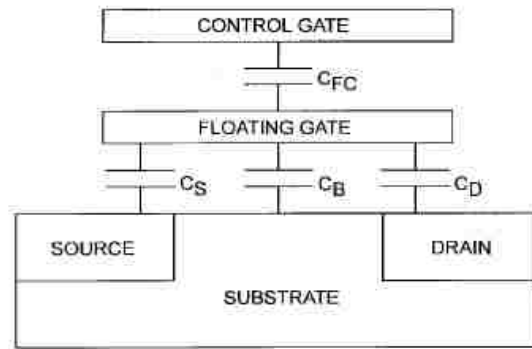


Figure 2.8: Schematic cross-section of FG transistor [23]

The first gate is the floating gate that is buried within the gate oxide and the inter-polysilicon dielectric (IPD) beneath the control gate. The IPD isolates the floating gate and can be oxide or oxide-nitride-oxide, ONO. The SiO_2 dielectric surrounding the transistor serves as a protective layer from scratches and defects. The second gate is the control gate which is the external gate of the memory transistor. Floating gate devices are typically used in EPROM (Electrically Programmable Read Only Memory) and EEPROM's (Electrically Erasable and Programmable Read Only Memory) [22] [23].

2.5.2 Charge Injection Mechanism

There are many solutions used to transfer electric charge from and into the FG. For both erase and program, the problem is making the charge pass through a layer of insulating material. The hot-electron injection (HEI) mechanism generally is used in Flash memories, where a lateral electric field (between source and drain) “heats” the electrons and a transversal electric field (between channel and control gate) injects the carriers through the oxide. The Fowler–Nordheim (FN) tunneling mechanism starts when there is a high electric field

through a thin oxide. In these conditions, the energy band diagram of the oxide region is very steep; therefore, there is a high probability of electrons' passing through the energy barrier itself. It is interesting to notice how these two mechanisms have been deeply investigated for MOS transistors in order to avoid their unwanted degradation effects. In Flash cells, they are exploited to become efficient program/erase mechanisms.

2.5.2 a) Hot Electron Injection (HEI)

The physical mechanism of HEI is relatively simple to understand qualitatively. An electron traveling from the source to the drain gains energy from the lateral electric field and loses energy to the lattice vibrations (acoustic and optical phonons). At low fields, this is a dynamic equilibrium condition, which holds until the field strength reaches approximately 100 kV/cm. For fields exceeding this value, electrons are no longer in equilibrium with the lattice, and their energy relative to the conduction band edge begins to increase. Electrons are "heated" by the high lateral electric field, and a small fraction of them have enough energy to surmount the barrier between oxide and silicon conduction band edges. Figure 2.9 and 2.10 show the Energy band diagram of a floating gate memory during programming by hot-electron injection and the HEI mechanism. For an electron to overcome this potential barrier, three conditions must hold [32].

- 1) Its kinetic energy has to be higher than the potential barrier.
- 2) It must be directed toward the barrier.

3) The field in the oxide should be collecting it.

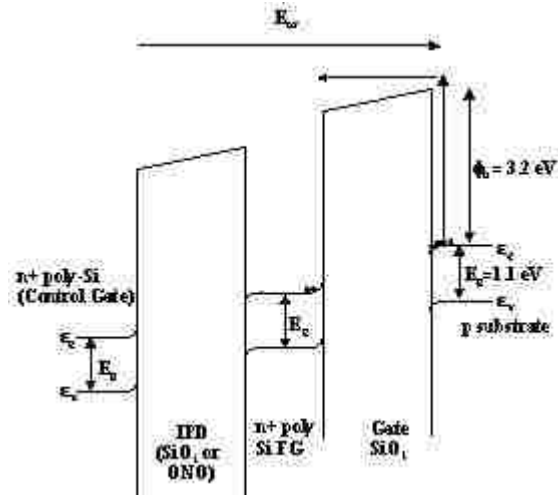


Figure 2.9: Energy band diagram of a floating gate memory during programming by hot-electron injection [22]

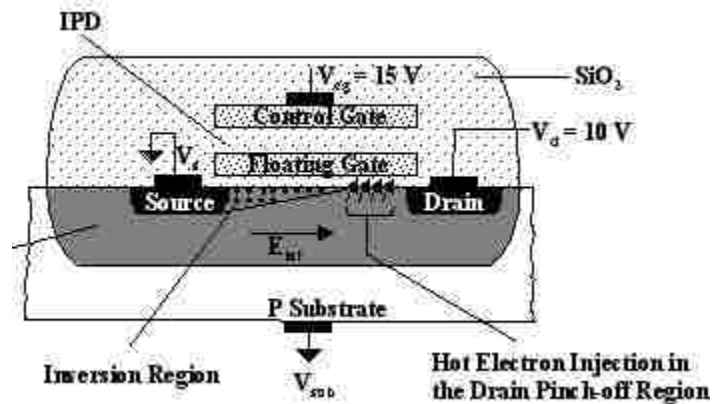


Figure 2.10: Hot-electron injection mechanism for programming in Flash memory [22]

2.5.2 b) Fowler-Nordheim Tunneling

One of the most important injection mechanisms used in NVM's such as flash is FN tunneling. When a large voltage V_{cg} is applied at the control gate during

programming, its energy band structure will be influenced as shown in Figure 2.11.

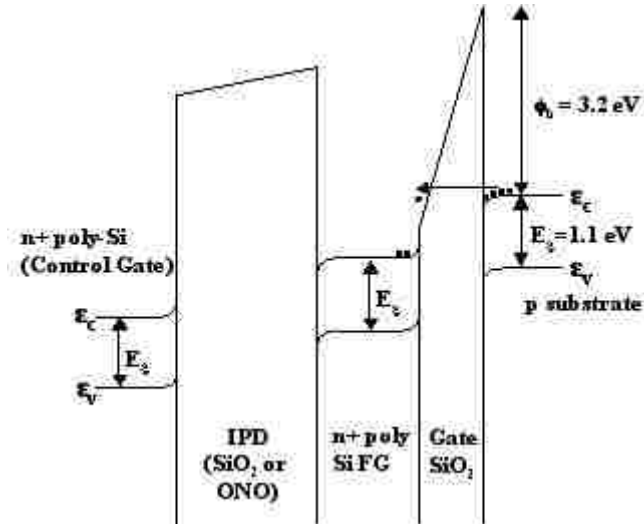


Figure 2.11: Energy band diagram of a floating gate memory during programming by FN tunneling

[22]

In the figure, e_c and e_v are the conduction and valence bands respectively, E_g is the energy band gap (1.1 eV for silicon), f_b is the Si-SiO₂ energy barrier (f_b is 3.2 eV for electrons and 4.7 eV for holes). The applied V_{cg} creates the electric field resulting in a potential barrier. This barrier provides a path for the electrons in the substrate to tunnel through the thin gate oxide (typically less than 12 nm) and eventually be collected in the n+ poly-Si floating gate. The bending of the energy bands of the IPD and the gate oxide are different due to the thickness differences between them. The IPD ranges from 25 nm to 45 nm while the gate oxide ranges from 5 nm to 12 nm. The electrons collected at the floating gate leads to a tunneling current density and is given by [33].

$$J = \alpha E_{inj}^2 \exp(-\beta/E_{inj}) \text{ ----- (1)}$$

With $\alpha = (q^3 / 8\phi_b \pi \hbar) (m / m^*)$ and $\beta = 4\sqrt{2}m^* (\phi_b^{3/2} / 3^n q)$

Where $h =$ Plank's constant

$\phi_b =$ Energy barrier at the injecting surface (3.2 eV for si-sio₂)

$q =$ charge of single electron (1.6×10^{-19} C)

$m =$ mass of a free electron (9.1×10^{-31} kg)

$m^* =$ effective mass of an electron in the band gap of sio₂ (0.42m)

$n = h/2\pi$

$E_{inj} =$ Electric field at the injecting surface = $V_{app} - V_{fb} / t_{ox}$ (v/cm)

$V_{app} =$ voltage applied across the tunnel oxide (V)

$V_{fb} =$ Flat band voltage (V)

$t_{ox} =$ Tunnel oxide thickness (cm)

Equation 1 shows that tunneling current density is exponentially dependent on the applied voltage, V_{app} , which influences the electric field, E_{inj} , across the gate oxide. Figure 2.12 shows a cross-section of a Flash memory with electrons tunneling uniformly with V_{cg} at positive potential while the source (V_s), the drain

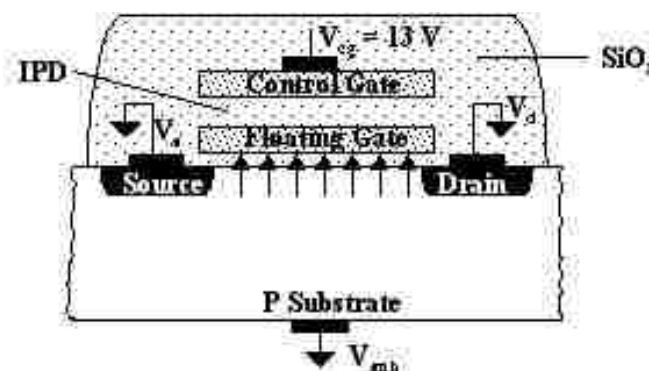


Figure 2.12: Uniform tunneling to program Flash memory [22]

(V_d), and the substrate (V_{sub}) are at ground potential. FN tunneling can also be used to erase an NVM. One of the methods is by applying a large negative voltage at the control gate. The energy band structure will be influenced as shown in Figure 2.13. The applied V_{cg} creates the electric field resulting in a potential barrier. This barrier provides a path for the electrons to tunnel from the floating gate to the substrate through the thin gate oxide

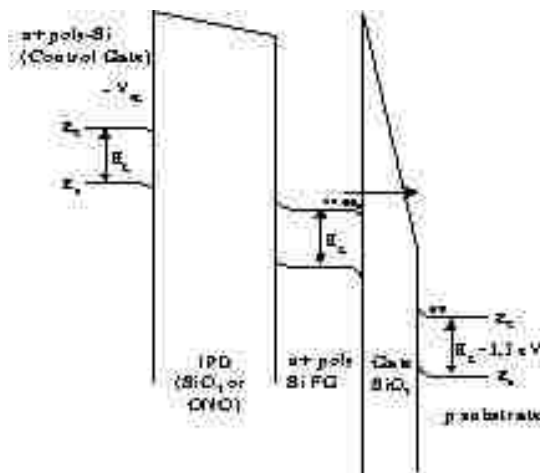


Figure 2.13: Energy band diagram of a floating gate memory during erasing by FN tunneling [22]

For uniform tunneling, a large negative V_{cg} is applied while for drain-side tunneling method, both a negative V_{cg} and a positive V_d are applied [22][23].

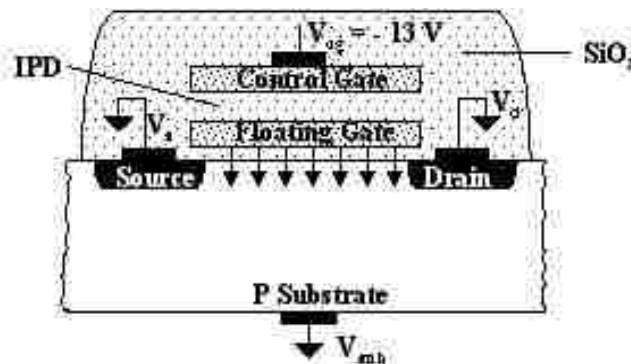


Figure 2.14: Uniform tunneling to erase Flash memory [22]

This is shown in Figure 2. 14.

2.6 Nand Flash Technology and Architecture

This section discusses the basics of nand flash from a designer's point of view. The nand flash array is grouped into series of blocks, which are the smallest erasable entities in nand flash device. A nand flash block is generally of arbitrary size like 128KB, 64KB etc. Erasing a block sets all bits to 1 (and all bytes to FFh). Programming is necessary to change erased bits from 1 to 0. The smallest entity that can be programmed is a byte. Although NAND Flash cannot perform Reads and Writes simultaneously, it is possible to accomplish Read/Write operations at the system level using a method called shadowing. Shadowing has been used on personal computers for many years to load the BIOS from the slower ROM into the higher-speed RAM. However, there is a limit to the number of times NAND Flash blocks can reliably be programmed and erased. Nominally, each NAND block will survive 100,000 Program/Erase cycles [24]. Figure 2.15 shows the layout and stick diagram of the Nand flash cell.

A technique known as wear leveling ensures that all physical blocks are exercised uniformly. To maximize the life span of a design, it is critical to implement both wear leveling and bad-block management. NAND Flash is very similar to a hard-disk drive. It is sector-based (page-based) and well suited for storage of sequential data such as pictures, video, audio, or PC data. Although random access can be accomplished at the system level by shadowing the data to RAM, doing so requires additional RAM storage. Also, like a hard-disk drive, a

NAND Flash device may have bad blocks and requires error-correction code (ECC) to maintain data integrity [24].

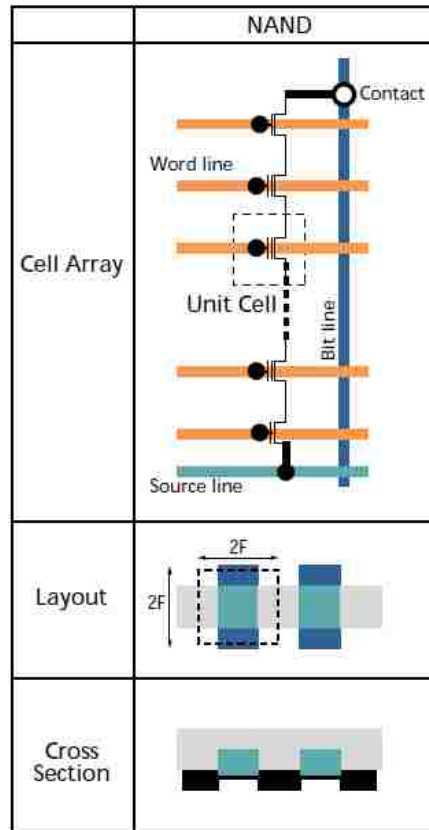


Figure 2.15: Nand Flash cell [24]

For instance, a 2Gb NAND Flash device is organized as 2048 blocks, with 64 pages per block (Figure 2.16). Each page is 2112 bytes, consisting of a 2048-byte data area and a 64-byte spare area. The spare area is typically used for ECC, wear-leveling, and other software overhead functions, although it is physically the same as the rest of the page. Many NAND Flash devices are offered with either an 8- or a 16-bit interface. Host data is connected to the NAND Flash memory via an 8-bit- or 16-bit-wide bidirectional data bus. For 16-bit

devices, commands and addresses use the lower 8 bits (7:0). The upper 8 bits of the 16-bit data bus are used only during data-transfer cycles

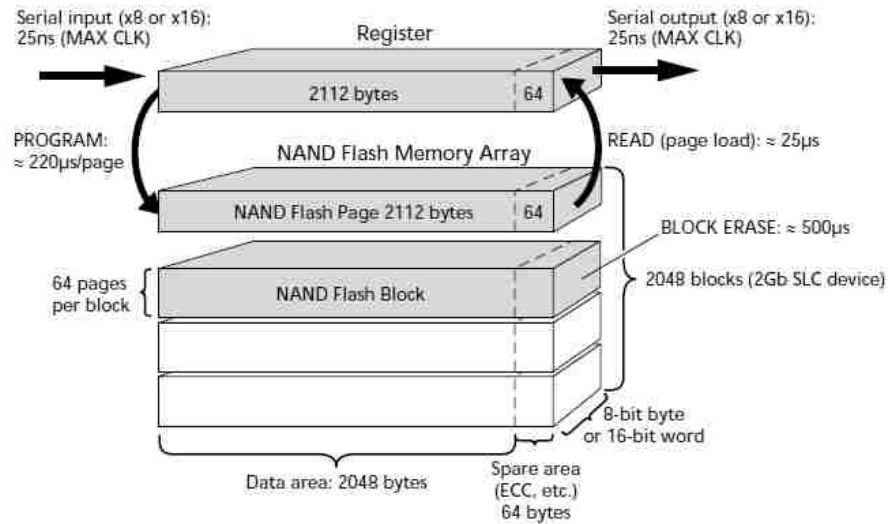


Figure 2.16: Nand Flash device organized as 2048 Blocks [24]

Storage method:

The two common methods for storing data and spare information in the same page are shown in Figure 2.17. The first method shows a data area of 512 bytes plus the 16-byte spare area directly adjacent to it; 528 bytes for the combined areas. A 2112-byte page can contain four of these 528-byte elements. The second implementation involves storing the data and spare information separately. The four 512-byte data areas are stored first, and their corresponding 16-byte spare areas follow, in order, at the end of the page.

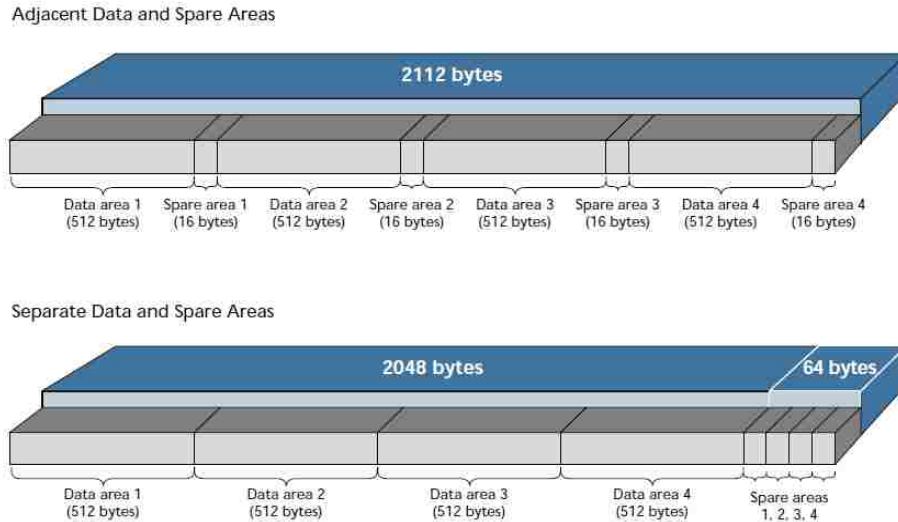


Figure 2.17: Typical storage method [24]

2.7 PUF technology.

In this section, we introduce the concept of Physical Unclonable Functions (PUFs). A PUF is a function that is embodied in a physical structure, so that it is easy to evaluate, but hard to characterize. The physical structure that contains the PUF consists of many random components. These random components are introduced during the manufacturing process and cannot be controlled. When a physical stimulus is applied to the structure, it reacts in an unpredictable way due to the presence of these random components [25]. The applied stimulus is called the challenge, and the reaction of the PUF is called the response.

PUFs inherit their unclonable property from the fact that every PUF has a unique and unpredictable way of mapping challenges to responses. Each die manufactured has unique physical characteristics as a result of slight variations in the ambient environment (temperature, physical location in the wafer, etc). While PUFs can be implemented with various physical systems our present

interest in this paper is on silicon PUFs (SPUFs) that are based on the hidden timing and delay information of integrated circuits. Even with identical layout masks, the variations in the manufacturing process create performance/delay differences among different ICs [26].

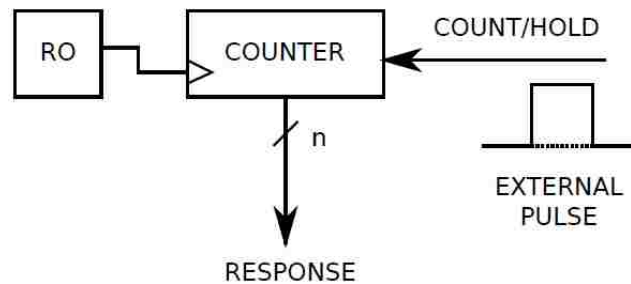


Figure 2.18: Simple ring oscillator PUF [27]

A simple ring oscillator PUF is explained to get an understanding of the concept of PUFs. The ring oscillator PUF is a design based on delay loops (ring oscillators) and counters. Each ring oscillator is a simple circuit that oscillates with a particular frequency. Due to manufacturing variation, each ring oscillator oscillates with a slightly different frequency. In order to generate a unique count value output from the ring oscillator is given as clock input counter. The output from the counter is the response of the PUF.

Chapter 3

Memory Controller

3.1 Introduction

In this chapter, Memory controller that implements a new technique to make the 3D Integrated Nand Flash memory chip more robust and fault tolerant against intense space radiation environment is proposed. The controller performs continuous self test and repairs itself in the case of any discrepancies such as physical errors in memory arrays; wear out faults, stuck bits and soft errors such as SEUs. Also, the controller provides protection against the possible effects of MBUs. The natural structure of the 3D integrated Nand flash memory provides the opportunity for charged particles in space to dig deeper in silicon going into the inner stacked layers hence causing a Multiple Bit Upset (MBU). It is an important consideration in critical applications related to space, avionics, and defense. Even if a single memory domain fails in a stack of memory modules due to any kind of irregularity, it can lead to a total system breakdown. A smart memory controller for such a system is therefore necessary. The memory controller should not only accomplish the memory accessing, but it also should act as a potential healing system (ability to retrieve the data and exclude the failed memory) for the stack of memory. The system will not recover from single or multiple module failures in the absence of such a healing application.

This section proposes an efficient way to control a stacked memory system along with a unique healing technique. It can also be used as a tester for finding manufacturing defects in the stackable memory. The global controller controls

and manages the memory modules and maintains a memory map of the various modules. It can either be a single-client or a multi-client system. The controller runs algorithms such that all the clients always see a contiguous memory. If, due to any reason, one or more of the memory modules fail, it automatically rearranges the mapping, so that the client attached to the spoiled memory space is assigned a new memory space. The controller uses a flexible remapping scheme (using PUF technology) and appropriate ECC to protect against these errors.

Proposed Memory Controller Properties:

- Flexible remapping scheme can be done at page, block and die level using PUF technology and logical to physical mapping concept.
- ECC is checked after each read and is readback to confirm the presence of a hard or a soft error(SEU & MBU)
- Uses one of Reed-Solomon code (255, 251), Reed-Solomon (5,3) Coder-Decoder in GF256 or Hamming code for ECC
- Uses special optional algorithm that detects the presence of MBUs in the flash block and is corrected immediately.

Cost Benefits

- Eliminates external test equipment – all tests are done on-board
- Reduces test time – no manual probe, and tests use internal bandwidth.
- Improves yield – conventional chips can handle fewer than 100 bad cells; chips with this controller can tolerate thousands of bad cells

Reliability Benefits

- Optional error and condition reporting
- Continuous “soft” error checking and correction
- Optional MBU detection and correction
- Continuous “hard” error detection and repair
- Longer projected lifetime per chip

3.2 Memory controller Technique

The controller can be best described in phases. There are basically four phases.

1. Scan and Mark phase
2. Discovery Phase
3. Coalesce Phase (Run Mode)
4. Heal phase

In the initial scan and mark phase, information regarding the state and condition of each memory module (page, block or single layer of Nand flash) in the stack is extracted and analyzed for stack organization and the known good die information is obtained. It works as a testing mechanism for the stack, the memory modules from the manufacturing phase can be tested for any irregularities in this phase. In the discovery phase, this information is used to mark the faulty memory modules by scanning through them individually. After getting the known good die information, the memory modules are collapsed to achieve a continuous map. The heal phase initiates only if there are any

irregularities in the memory modules. The heal phase has three functions, (1) Check for any bad memory and exclude it from the memory map, (2) Provide the client with a new memory module, (3) Restore the destroyed data.

Scan and Mark Phase:

This is the first phase of the global algorithm. At this point the state and condition i.e. the Known good die information of the 3D Nand flash memory is unknown. This is the phase in which the initial check on the yield of the chip is made. The controller is in this phase for only once in its total usage time. In this phase the controller first performs a sector by sector write-read back operation on the complete stack and collects information (known good die) regarding the state of each die, array and cell, hence called the scan phase. The controller sees the 3D nand flash memory as a collection of blocks sub divided into pages irrespective of the number of levels (stacks) that are present in the IC. It starts off by writing 0s to each page sequentially and after each and every memory cell is written into, it reads the data back. By doing this it can detect the presence of stuck-at 1s and it performs a similar operation by writing and reading back 1s to detect the presence of stuck-at 0 faults in the Nand flash memory.

Apart from these this operation detects the faults caused due to other physical errors in memory such as yielding errors and hard errors. Faulty cells during the scan are marked and the extent of damage is noted, this information is used by the controller to give an optional error and condition report but more importantly to decide if it should include the particular locations in the memory mapping or

not. This also helps the controller to decide if the remapping should be done at a cell, array or die level. Since the 3D chip can have possibly hundreds of levels there is very good chance of yield problems which will render the whole stack level useless in which case die- level remapping is done.

Once the initial scan is done and good knowledge is accrued regarding the state of the memory the mark phase is initiated. The controller sees the memory stack of the 3D Nand flash memory as a collection of pages. E.g. A 2Gb nand flash 2D chip is generally organized as 2048 blocks, with each block containing 64 pages and each page containing 2176 bytes or 17408 bits. So there is a total of 131,072 pages in a single layer, if we consider an 8 layer stack we have 1,048,576 pages in 3D nand flash chip.

The mark phase starts with the initiation of the PUF values. The PUFs used here are simple ring oscillator PUFs that generate 'n' bit unique values, where n is decided by the number of pages in the stack and can be modified to any custom length in order to achieve unique values. The information collected in the scan phase is used to mark the pages with specific IDs generated by the ring oscillator PUF. According to the condition report each good page is assigned a random PUF value and pages containing excess faulty cells are assigned null values.

At this point there is complete information with the controller regarding not only the state of each die and each memory cell but also IDs of all the possible pages which are write ready. This is the process that is followed by the memory controller for stack organization before it assigns memory to any function for

access. When a function needs memory in a system the memory controller sets the proper hand shaking signals for the access to occur. Figure 3.1 shows the algorithm for scan and mark phase.

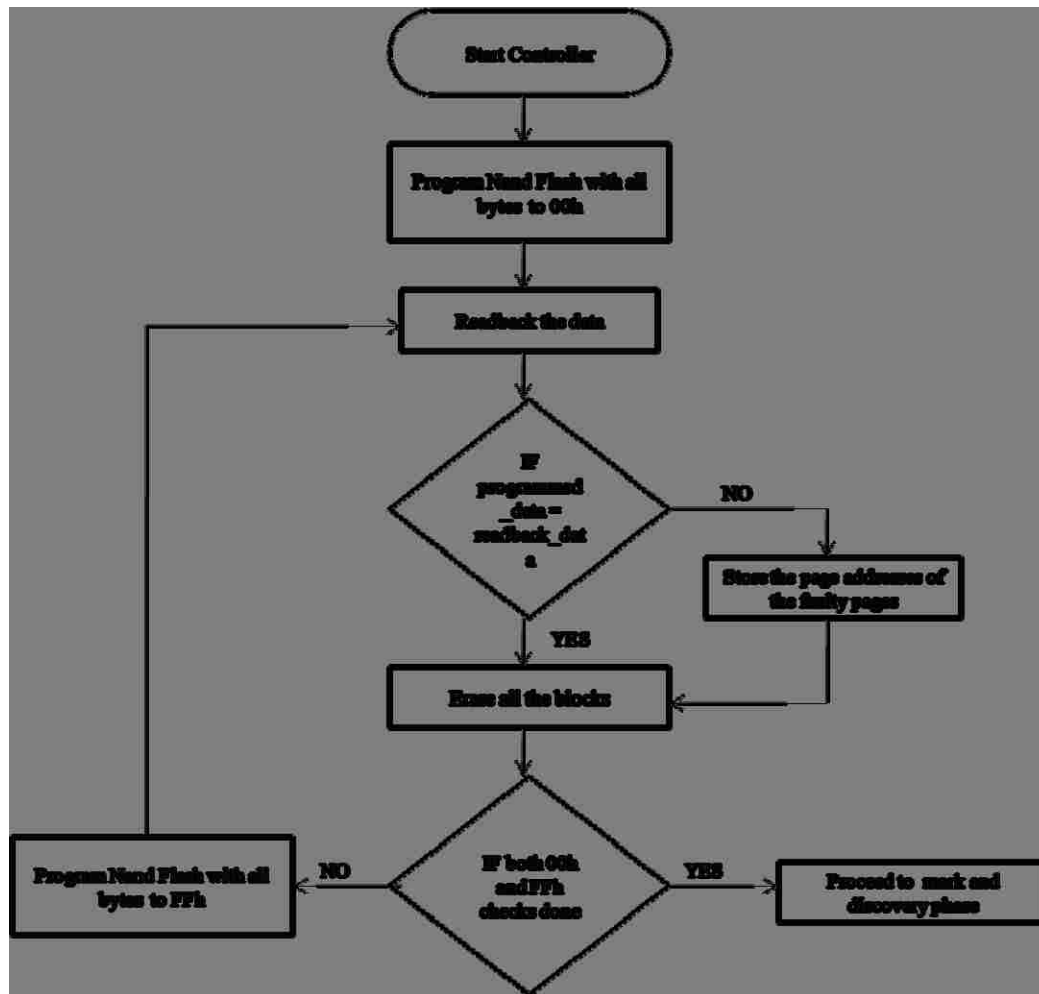


Figure 3.1: Flow chart for Scan and Mark phase Algorithm

Discovery phase:

This phase is the mapping phase of the controller. The controller in this phase creates memory maps using the logical to physical mapping concept. The client sees the whole stack as one single block of memory with a continuous set of

addresses (logically) but physically the locations of the addresses may not be continuous. The memory map is created in such a way that the clients always see a contiguous memory map and writes to the same addresses irrespective of the faulty memory modules in the stack that are spoiled either from manufacturing defects or hard and soft errors that pop up intermittently which have caused the physical locations to change to a different area. The nand flash pages are mapped in a unique way using the PUF IDs so that they can be assigned to the clients that need access to memory. This phase is basically run by the UIDC (unique ID for specific client) counter. It is a simple n bit counter where n is chosen depending on the number of pages in the stack and can be modified to any custom length. The algorithm for the discovery phase is shown in figure 3.2.

When creating the new mapping the controller increments the UIDC counter and compares it against all the PUF values. If match is found the memory is said to be discovered and the controller assigns this module (nand flash page) to the client for access and the UIDC counter is incremented to find the next module. When all the n dies are discovered a continuous map is made from the set and stored in a register. The spoiled memories from the scan phase are never discovered because their IDs are set to null value in the mark phase. The controller is in the discovery phase whenever a faulty memory is discovered to remap the memories.

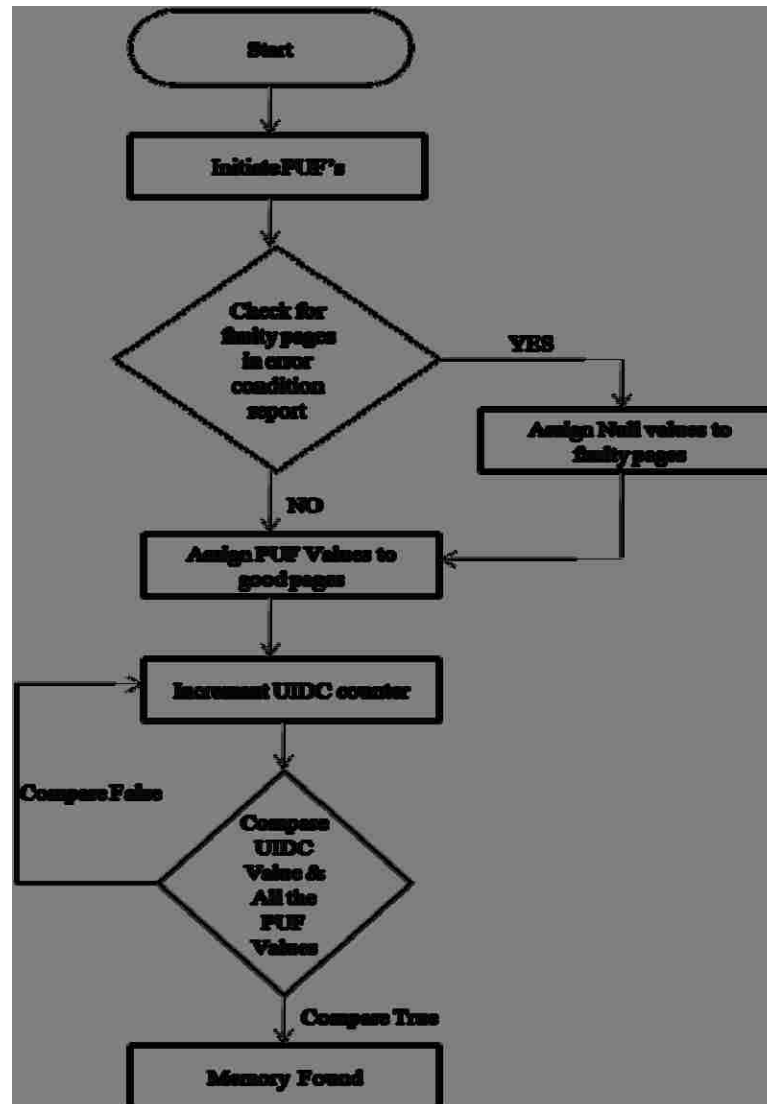


Figure 3.2: Flow chart for Discovery phase Algorithm

Coalesce phase:

The Coalesce phase is basically the run mode for the system. After achieving a continuous map in the discovery phase, the controller asserts a system ready signal to the client so that it can access memory. In this phase the controller performs data write and data read requests. During a data write request the

controller writes to the page discovered in the discovery phase. The coalesce phase runs along with the discovery in tandem, after each write to a page the controller goes into the discovery phase to find the address of a new page to write. Although the client writes to the next immediate address (logical address) it is actually written to the page chosen by the UIDC counter (physical address).

The controller stores this memory mapping data in a data register that is used during data read requests. The data written to the page is first treated with an ECC engine and the encoded data is written to the page. For e.g. In a single page in the Flash memory there is 2112 bytes for data and 64 bytes for ECC redundant bits, each page has 4 sets of 512 bytes. This is shown in Figure 3.3. Different types of ECC can be used such as hamming codes, Reed-Solomon code, BCH codes etc. depending on the number of bytes needed to be corrected at the expense of more power, computation and latency trade off.

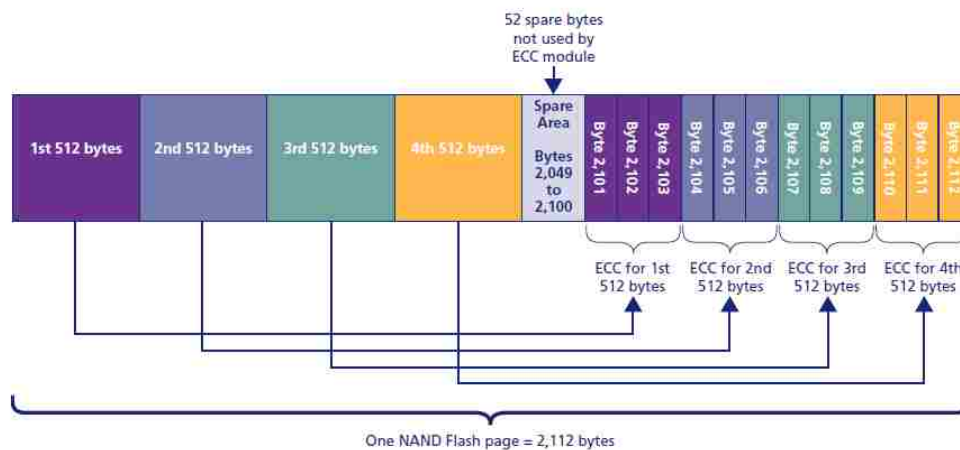


Figure 3.3: Bit format for NAND flash page

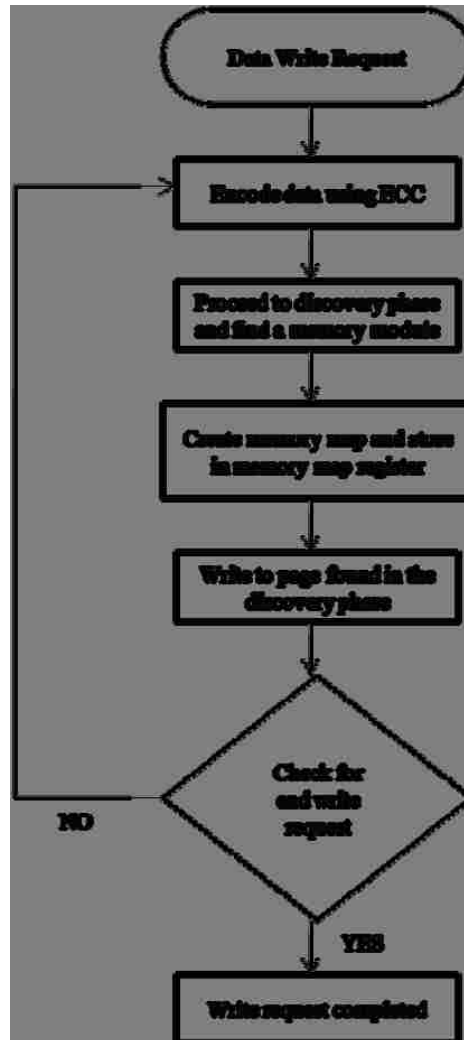


Figure 3.4: Flow chart for Data write request Algorithm

During a data read request the controller reads the physical addresses from the memory mapping register and completes the requests. Before completing a read request it performs an ECC verification, if verification is clean then it completes the request. If a fault is encountered in the ECC then an interrupt is asserted indicating the halt of memory access for data correction. The controller now proceeds into the heal phase where the distinction between a hard or a soft error is made and appropriate measures are taken to rectify the error and restore

health to the memory. Figure 3.4 and 3.5 show algorithm for the controller in the coalesce phase.

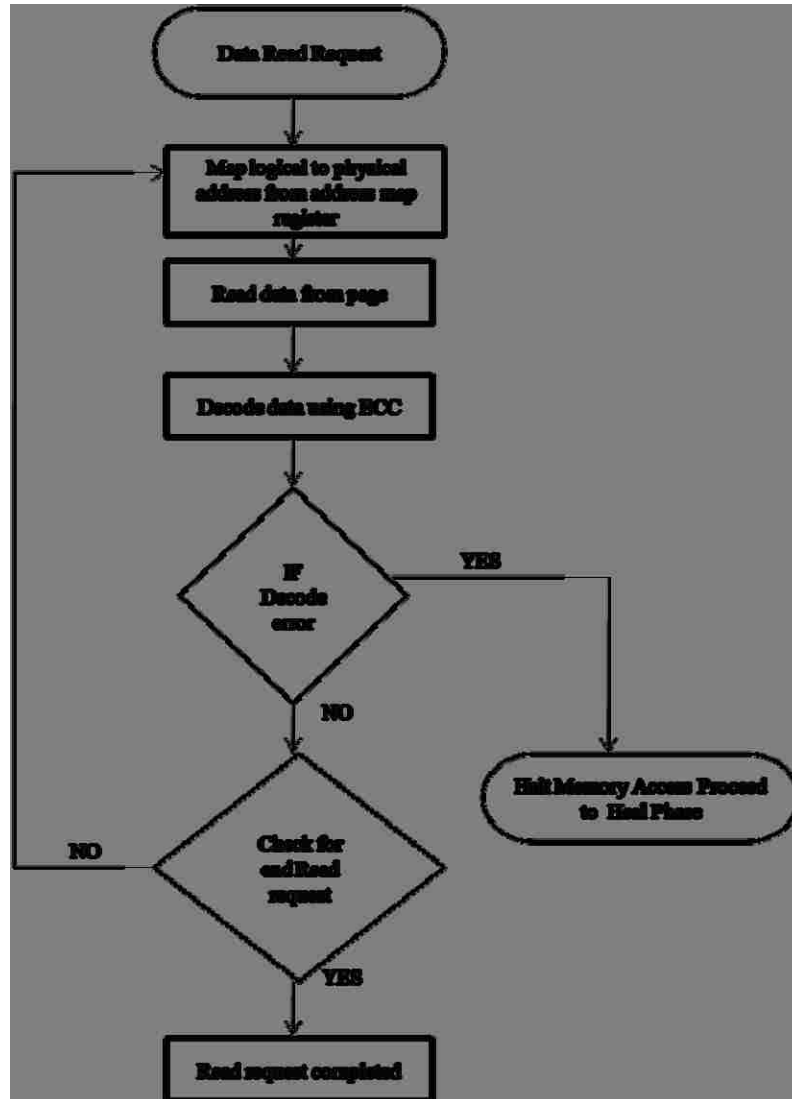


Figure 3.5: Flow chart for Data Read request Algorithm

Heal Phase:

The heal phase is the correction phase of the controller, the first action taken by the controller in this phase is to analyze the error that is detected in the coalesce phase. Depending on this analysis the controller decides the type of correction mechanism that must be applied. In case of a soft error, the controller restores the affected data through ECC decoding techniques provided the affected number of bits is in the permissible range of errors of the particular type of ECC being used, for e.g. A Reed-Solomon (255,247) decoder can correct up to 32 bits or 4 bytes while a simple hamming code can correct 1 byte. In the case of a hard error, the controller in addition to the retrieval of the affected bits has to remap the memory location of the affected site to a new location.

The remapping can be done at page, block or die level depending on the extent of the affect of radiation on the 3D stack. When a fault in the memory is detected in the coalesce phase, the controller first analyzes the fault by executing a simple algorithm which will decide the type of error. There are two types of algorithms, the first is to stop the data access and backup data to a data sink (like a memory buffer) and perform readback on the affected memory module. The controller first programs the particular module with a stream of 1s and reads back the data and then it programs a stream of 0s and reads it back. If the data programmed is equal to the data readback in both cases then the error is most likely a soft error and if the data readback is not equal then the error is qualified as a hard error and remap of the memory module is performed. The second method is to assume a soft error in each case and keep track of the areas being

affected. If a particular area pops up too frequently than the memory module is either affected by a hard error or is a very SEU sensitive node. In both cases the controller can proceed to remap.

The controller also performs an optional multiple bit upset (MBU) detection algorithm that will test the vicinity of the affected bit for an error. Since the memory chip is 3D there is a good chance of the radiation to dig deeper into the chip. During MBU detection, when a soft error is detected the controller not only corrects using ECC but also performs an ECC check on the area of the 3D die above and beneath the affected area and repeats the process recursively. If the ECC check fails it can be considered as a multiple bit upset and is corrected immediately. The memory mapping register is updated with the new mapping.

The remapping is done at page, block and die level by returning to the discovery phase, when a faulty memory module is encountered it is excluded from the memory map and the controller proceeds to discovery phase to find a new module to replace it. As said above, in the discovery phase the UIDC counter selects a memory module according to the random PUF value and replaces it with the faulty die. Once a new module is selected, it is included in the new mapping. At this point, the client still writes to the same logical address but the address now points to a different physical memory module. Figure shows the algorithm for Heal phase. Figure 3.6 shows the algorithm for the heal phase.

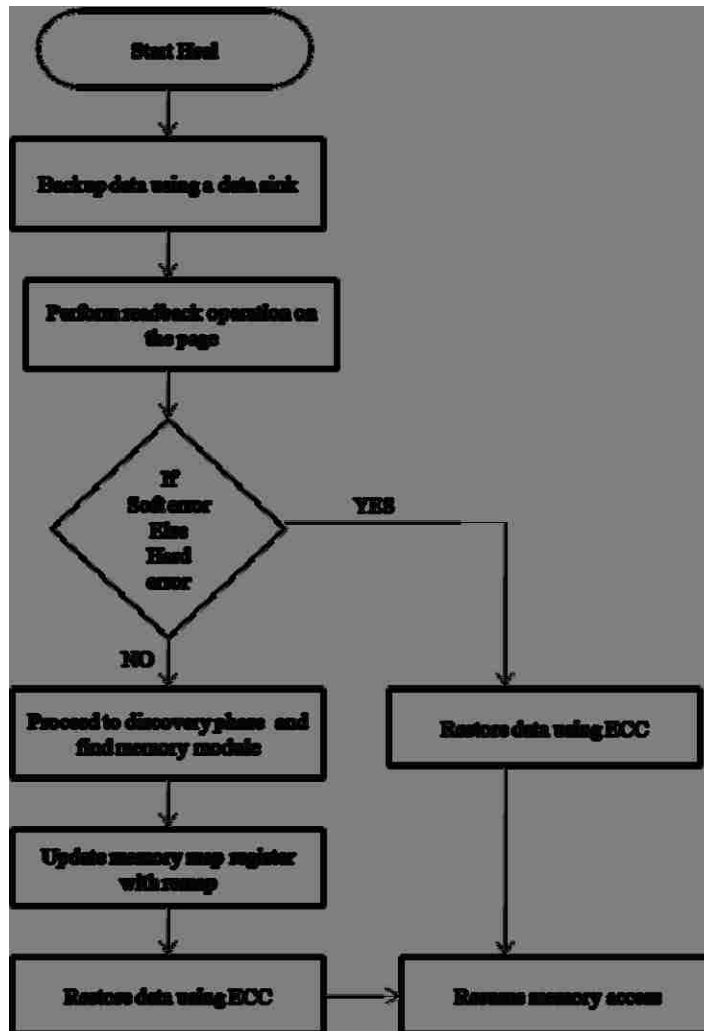


Figure 3.6: Flow chart for Heal phase Algorithm

3.3 Proposed block diagram of Hardware description

This section provides the details of the proposed implementation of the controller. Figure 3.7 shows a way of implementing the 3D Nand flash controller, each block is further explained in brief and chapter 4 gives a detailed explanation of the implementation of each block.

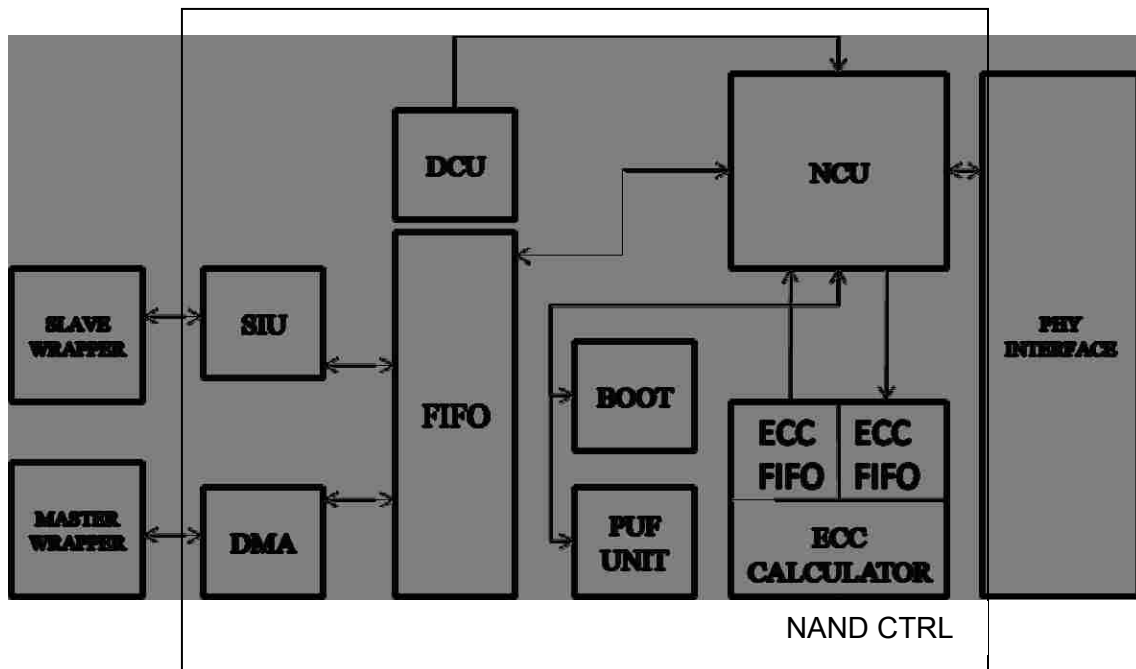


Figure 3.7: Block diagram of Hardware Specification for 3D NAND Flash Controller

The figure shows the top level description of the internal structure of the NAND flash controller design. The module within the solid line is the NAND control core, the modules outside are optional and are present on the chosen controller configuration.

Main components of the controller are:

FIFO – This unit provides FIFO queue interface to the other controller modules. Depending on the software configuration the one queue side will be data input modules the Slave interface unit (SIU) or direct memory access (DMA) module, the second side will always be NAND control unit (NCU)

Device Control Unit (DCU) - This is the main engine of the controller it controls the other modules basing on the current special function register values and the current controller state. The main control tasks of this module are:

- Provide the enable/disable signal to the DMA and SIU units when they try to get access to the FIFO module. Only one of those is active at a time.
- Enable/Disable the ECC module
- Provide the control signals to the NAND control unit
- Execute the boot sequence from boot unit
- Execute PUF sequence from PUF unit
- Control the interrupts

NAND control unit (NCU) - This unit is responsible for generation of 3D Nand flash device access sequences. Unit uses control signals provided by the DCU.

ECC - It is an error correction code calculator and a correction unit. A correction word is calculated for each 256 or 512B (or optionally over 512B) sub page of the NAND Flash memory page. During the read operation the unit can automatically correct bad bits without any interaction with the external system. It has a status register, the bits of which signal errors occurring during a read, and then inform if errors were corrected. It is possible to choose between a simpler unit that can correct only one error per 256B or 512B (or optionally over 512B) sub page and a more advanced unit that can correct multiple errors. The choice depends on the NAND Flash memory type that is in use. Depending on the end-user application, it is possible to choose between two solutions. The first one is

based on the Hamming algorithm that allows correction of one error for each 256B sub page. The other solution uses the Reed-Solomon algorithms. The ECC module has integrated FIFO that is used to transfer the calculated words to the NCU modules during encode process and to store the calculated partial syndromes during decode process.

DMA- This unit is responsible for fast transfer of the data between the external memory location and the controller

SIU - Unit provide the slave interface to the controller SFR registers and the FIFO module.

Boot unit - Unit initiates boot sequence on command from DCU. This unit heads the scan and mark phase of the controller.

PUF unit - This unit responsible for generation of random PUF values. It holds all the actual PUFs that generate the PUF values. It performs functions on command from the device control unit (DCU).

3.4 Benefits of the Controller

Most memory chips are tested for repair and remap only once, on the production line. If a bit becomes “stuck” at a later time (due to magnetism, radiation, heat, impact, or other damage) it cannot be repaired; the entire chip must be replaced [17]. With our controller, these “hard” bit errors are detected and remapped instantly. Many Memory devices ignore the random, recoverable

bit-flips caused by radiation; if a bit is flipped; the error remains until a new value is written to that location [17]. Our controller uses ECC (error checking and correction) or EDAC (error detection and correction) to detect and correct these “soft” errors in their memory banks, the complexity added by the ECC circuit can be reduced by using different levels of ECC based on various trade-off factors such as correction capability and latency. Also, because the controller continually monitors chip performance, it can detect and report unreliable behavior long before the chip actually fails using its error and condition reporting feature. This feature would allow failing parts to be detected and replaced before the pool of redundant bits are exhausted.

The controller also performs optional MBU detection and correction. Since 3D memory due to its natural structure has increased probability of being affected by an MBU this feature will help the performance of the memory by many fold. By using our controller testing the 3D nand flash chip is much easier since all the tests can be done on board without the need for external probing. In general the controller provides longer projected lifetime to the 3D nand flash chip in harsh radiation intense environment.

3.5 Limitations

The main limitation of the controller is the time delay incurred in the system as a trade-off for achieving more robustness. The delay incurred over a normal controller without any fault tolerance can be calculated using the following analysis:

Case 1: System has no error

$$T_{\text{our self-healing system}} = T_{\text{normal-system}} + T_{\text{initial readback}} + T_{\text{PUF Calculation}} \\ + T_{\text{ECC calculation in write cycle}} + T_{\text{ECC calculation in read cycle}}$$

Where

$T_{\text{our self-healing system}}$ = Time taken by Memory controller system

$T_{\text{normal-system}}$ = Time taken by a system with no fault tolerance

$T_{\text{initial readback}}$ = Time taken to check each memory page for errors by writing a '1' reading it back and then writing a '0' and reading it back

$T_{\text{PUF Calculation}}$ = Time taken for initiating the PUFs and generating a unique identification for each die

$T_{\text{ECC calculation in write cycle}}$ = Sum of the Time taken to encode data with ECC during each write cycle

$T_{\text{ECC calculation in Read cycle}}$ = Sum of the Time taken to decode data using ECC during each read cycle

Case 2: System with error

$$T_{\text{our self-healing system}} = T_{\text{normal-system}} + T_{\text{initial readback}} + T_{\text{PUF Calculation}} \\ + T_{\text{ECC calculation in write cycle}} + T_{\text{ECC calculation in Read cycle}} + T_{\text{data restore \& remap}}$$

Where

$T_{\text{data restore \& remap}}$ = T_{readback} + $T_{\text{remap and ECC calculation}}$

T_{readback} = Time taken by the controller to spot the faulty memory by readback

$T_{\text{remap and ECC calculation}}$ = Sum of the Time taken to restore data bits using ECC and finding new memory location using the PUF value.

Also, in the case where the mapping is done at page level the erase operation needs a good amount of data sink to backup memory because nand flash can be erased only at block level and not at page level, hence there is significant amount of space wastage and time delay incurred due to this operation. This will not be a big problem at block level and die level remapping where the erase operation will not erase valid data or erasure of data is tolerable.

Chapter 4

Implementation

4.1 Introduction

This section presents the proof of concept of the memory controller Implementation and simulation results. The overall idea of the implementation is to create a working memory controller equipped with three kinds of ECC and give a tradeoff table that can be referred to decide what kind is best suited for a particular space program's mission requirements. Three different kinds of ECC with different correction capability, space requirement, latency (delay) and power requirement were chosen namely, Hamming code, Reed-Solomon (255, 251) and Reed-Solomon (5,3).

We implemented our memory controller design in an FPGA (Virtex- 4) using VHDL. The block diagram of the implementation is given in Figure 4.1. The circuit differs from the concept block diagram given in chapter 3 in the fact that a stack of Block memory of the FPGA was used to mimic the 3D nand flash chip and the bus model is replaced by a client test circuit which is used to test the memory controller. This section first explains each block in the design and gives a brief introduction to the handshaking signals and I/O that will facilitate a good understanding of the results obtained. Then functional timing simulation results are shown to confirm the working of each block and the controller as a whole and finally the ECC modules are replaced to make a trade off table showing the correction capability of the controller, space requirement (no. of gates used) and

power requirement in all the three cases. The tools used are Xilinx ISE, ModelSim SE simulator. The power analysis was made using Xilinx Xpower analyzer.

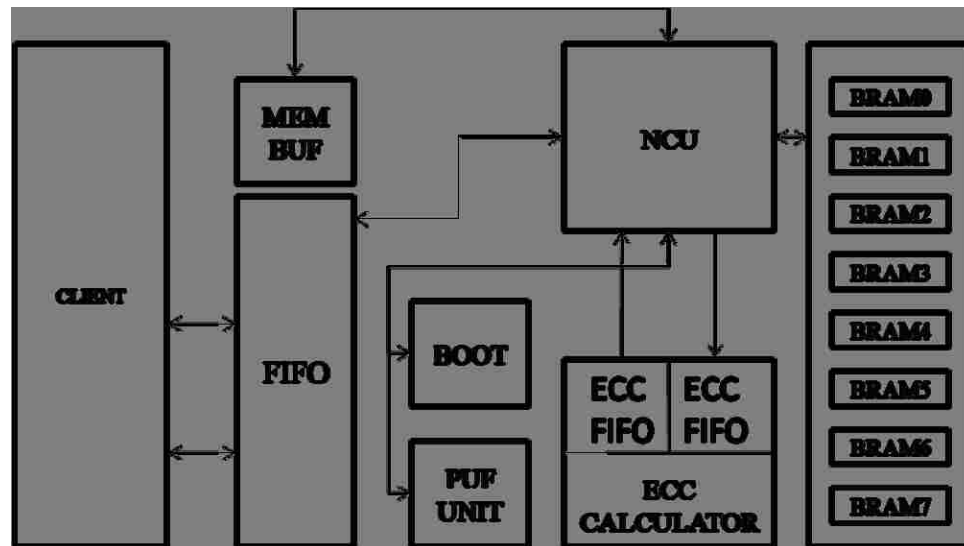


Figure 4.1: Block diagram of Hardware Specification for 3D NAND Flash Controller implemented on a Virtex 4 FPGA

4.2 Hardware description

Block Memory: The main controller is designed to access BRAMs in way to mimic the 3D nand flash structure (Figure 4.2). The design used 8 BRAM stack, each BRAM consisted of 1024 bits and represented a single page, two such BRAM constituted a block, and four such BRAM constituted a single layer of NAND memory. The 8 BRAM stack represented 2 layer 3D integrated NAND flash memory. The address of the BRAM represented the actual page address of NAND flash given as:

BRAM address = Actual page address

Where Actual page address = Block address concatenated with page address

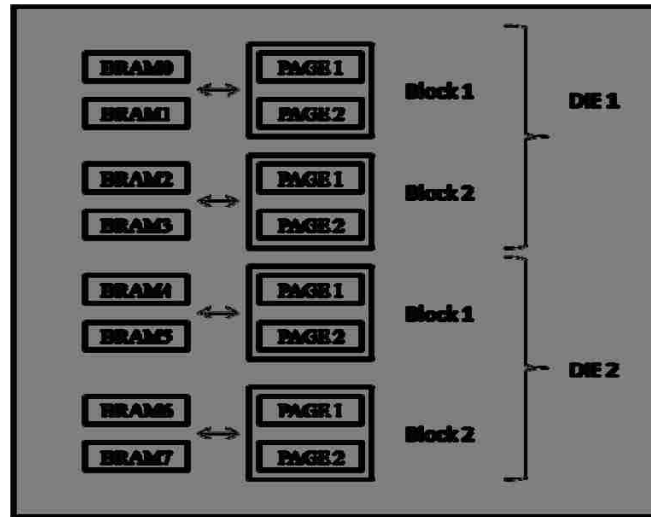


Figure 4.2: Block Memory in FPGA used to mimic 3D NAND Flash Memory

PUF unit: The PUF unit consists of counters; each counter is 10 bit and runs on a slightly different clock rate. For our purpose we have used different clock inputs to the counters to generate the random PUF values that are used to identify the pages. The number of pages in the memory decides the number of counters required in the PUF unit. In our design we have used 8 counters for the stack of 8 BRAMs.

Boot unit: The boot unit consists of a state machine and control registers that are controlled by the Nand control unit (NCU) during the scan and mark phase. The algorithm for the Boot sequence is provided in chapter 3 under scan and mark phase.

Memory Buffer: The memory buffer in the FPGA is implemented as a BRAM. This unit is used as a data sink during coalesce phase to store temporary

data while the data restoration and remapping is being accomplished by the controller.

Nand Control unit (NCU): This is the main engine of the controller it works as a device controller and also generates access sequences for accessing the BRAMs. It stores the memory map and performs logical to physical mapping while completing data read and data write requests. It generates proper status signals so that the client can accomplish communication with the controller.

FIFO: This unit provides queue interface to the Controller. The controller has to perform boot, PUF generation and mapping functions before it can generate access sequences hence, FIFO interface is essential to avoid loss of data and proper handshaking. Apart from this there is additional FIFO in the controller which is used by the ECC module, one for the encoder and one for the decoder.

ECC unit: The ECC In our implementation we use three different kinds of ECC namely:

- Hamming Code - Single Error Correction and Double Error Detection (SECCDED)
- Reed Solomon (5, 3) Encoder-Decoder in GF(256)
- Reed-Solomon (255,251) Decoder/Encoder

Hamming code:

The hamming code algorithm is a basic ECC with a correction capability of a single bit and a double bit detection capability. In our design we used the hamming algorithm core from Xilinx. The design is a piece of combinational logic

for data communication between the client and memory. The data bus is 16-bit wide, while the data written to memory is a 22-bit data word. When data is read back from the memory device, the stored parity bits are compared with a newly created set of parity bits from the read data. The result of this comparison, called the syndrome, will indicate the incorrect bit position in a single data error.

The figure shows the block level design of the hamming algorithm. This interface consists of the 16-bit processor data bus, $u_data [15:0]$, the read/write control signal, rw_n , and the error flag signal, $error_out [1:0]$. The right hand side describes the memory component interface, consisting of the memory data bus, $mem_data [21:0]$.

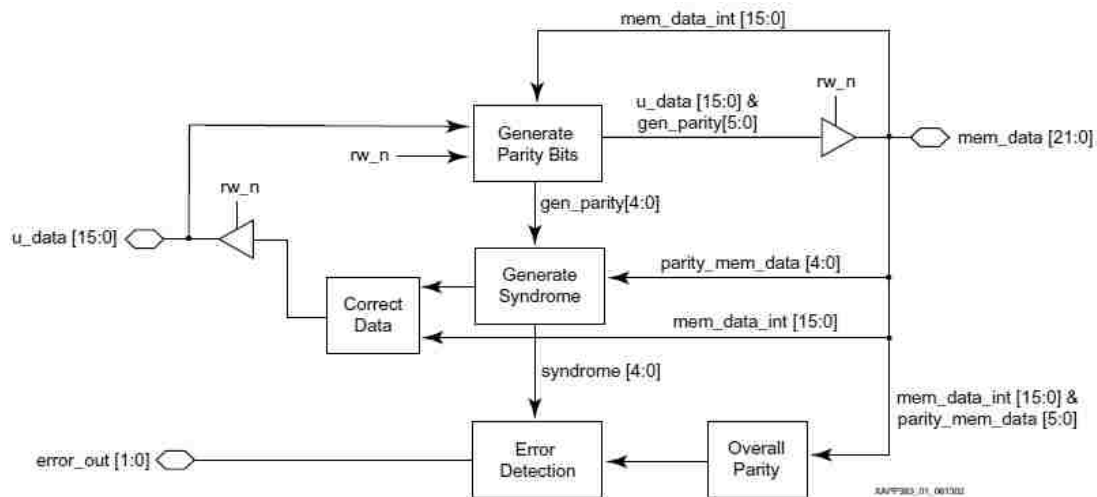


Figure 4.3: Hamming code Algorithm Block diagram [28]

The rw_n control signal from the client switches controller between read and write cycles. The rw_n signal will be equal to "1" for a processor read cycle and equal to "0" for a processor write cycle. The "Generate Parity Bits" block creates the parity bits to store with the processor data ($u_data [15:0]$) during a write

cycle. In a read cycle, this block is also responsible for creating one of the inputs in generating the syndrome; this block creates the parity bits with the data word stored in memory. The "Error Detection" block generates the error_out [1:0] flag based on the syndrome and the overall parity created from the data in memory.

Reed-Solomon (255,251) Decoder/Encoder

This core implements Reed-Solomon decoder for the 8-bit wide symbols. The core is designed to occupy fewer amounts of logic blocks, be fast and parametrizable. The main features are:

- 8-bit input and output data busses
- Fully synchronous and pipelined design using a single clock
- Symbol width of 8 bits
- Corrected byte number signaling
- Can correct 2 symbols (16 bits).

The block diagram of the decoder is as follows:

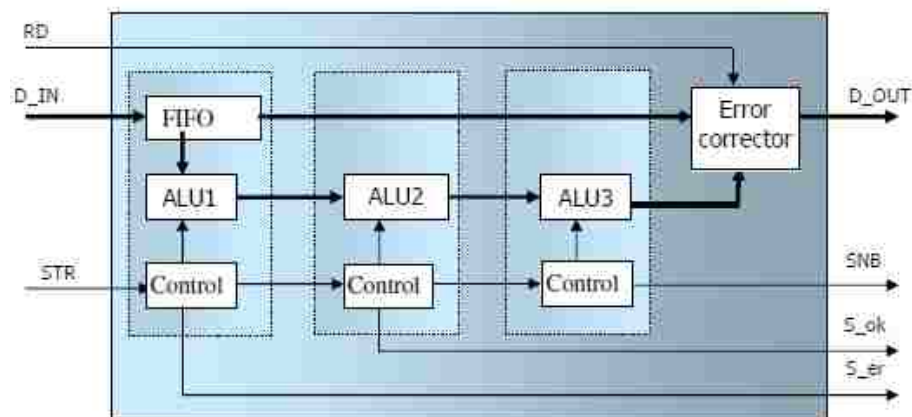


Figure 4.4: Block diagram of Reed-Solomon Decoder showing the I/Os [29]

Here signal D_IN is the input to the decoder and is only active after STR signal is pulsed. The SNB signal indicates the finish of decoding and the RD strobe is pulsed to accept the data of the D_out pin. Signals S_ok and S_er are status signals, S_er signal indicates the occurrence of an error and S_ok indicates that the error is fixed.

Reed Solomon (5, 3) Encoder-Decoder in GF(256)

This core implements Reed-Solomon decoder for the 8-bit wide symbols. The main features are:

- Symbol length: 8-bits.
- Coder: Takes 3-symbol message and encodes them into 5-symbol codeword.
- Decoder: Corrects 1-symbol (8bits) error in a codeword.
- No latency in decoding.

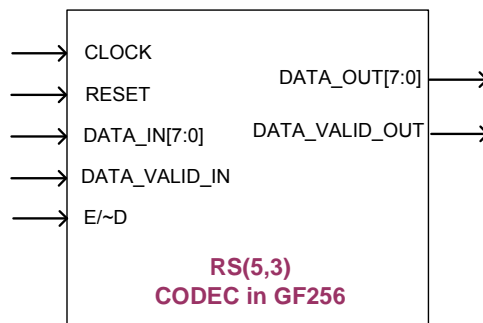


Figure 4.5: I/O specification for Reed-Solomon (5,3) Codec

The DATA_VALID_IN signal is an Active low that enables DATA_IN and E_D signals. The DATA_IN [7:0] signal takes the data from the client. Both message and codeword are fed to the CODEC from this port. When the E_D is high the

engine performs encoding and when low it performs decoding. E_D is monitored during DATA_VALID_IN is low, therefore should be constant unless mode of operation is not to be changed. DATA_OUT [7:0] is valid output data when DATA_VALID_OUT is low.

4.3 Results

4.3.1 Functional timing simulation

Boot unit functional timing simulation



Figure 4.6: Functional Timing Simulation of Boot Unit of 3D NAND Flash Controller

Figure 4.6 shows the functional simulation of the boot section which runs the boot sequence described in the scan phase in chapter 3. Each block ram representing a page in the 3D nand flash memory is written and readback with 00h and FFh values to all bytes to check for hard errors and yield errors such as stuck-at faults.

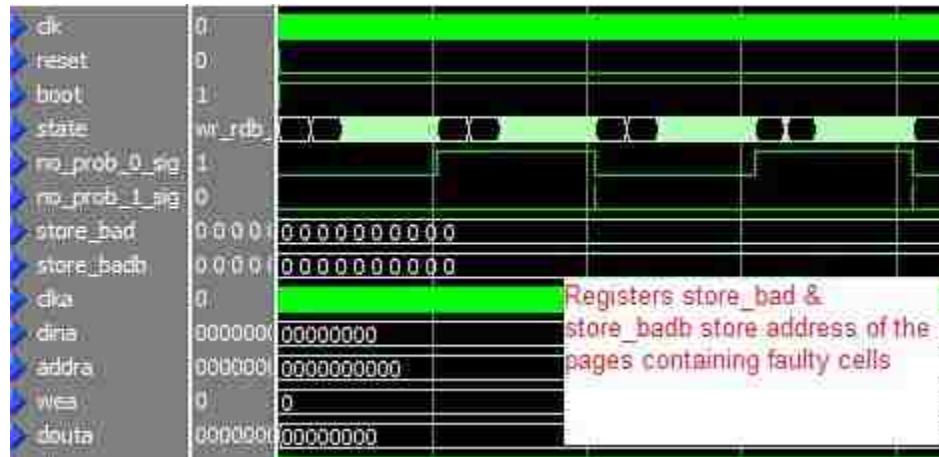


Figure 4.7: Functional Timing Simulation of Boot Unit of 3D NAND Flash Controller

Figure 4.7 shows the registers holding the address of the faulty memory cells that will be removed the memory map in the mark phase as described in chapter 3.

PUF unit functional timing simulation

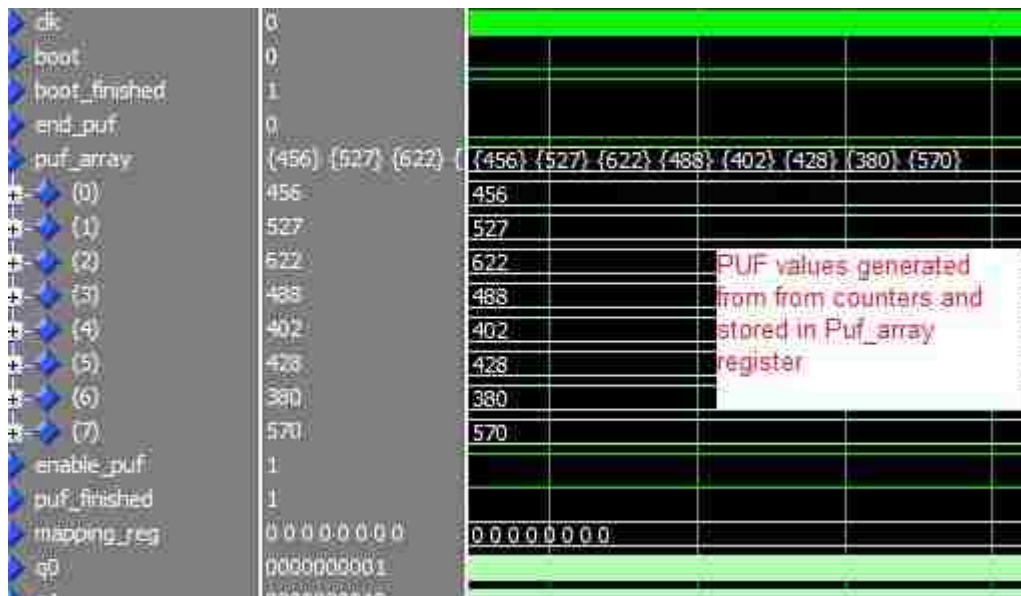


Figure 4.8: Functional Timing Simulation of PUF Unit of 3D NAND Flash Controller

Figure 4.8 shows PUF array holding the 10-bit PUF values generated by the counters. Each counter was supplied with clock varying by $0.1 \mu\text{s}$ which represent clocks coming from ring oscillators. These values are used as IDs for the memory modules in the mark phase.

ECC unit functional simulation

ECC module 1 – Hamming code algorithm

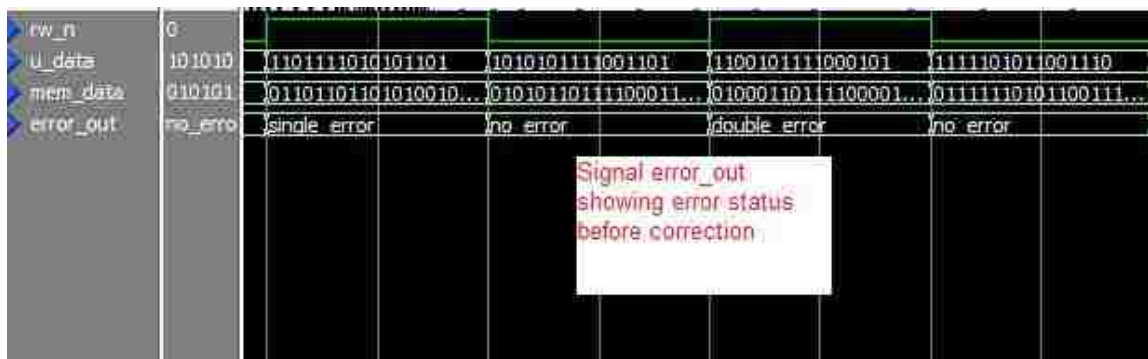


Figure 4.9: Functional Timing Simulation of ECC Unit (Hamming code) of 3D NAND Flash Controller

Figure 4.9 shows hamming algorithm with 1 bit correction and 2 bit detection capability. 16-bit `u_data` is the data line that takes data from the FIFO unit and `mem_data` is 22 bit that goes into the memory. This particular algorithm adds 5 parity bits for every 2 bytes of data.

ECC module 2 - Reed Solomon (5, 3) Encoder-Decoder in GF(256)

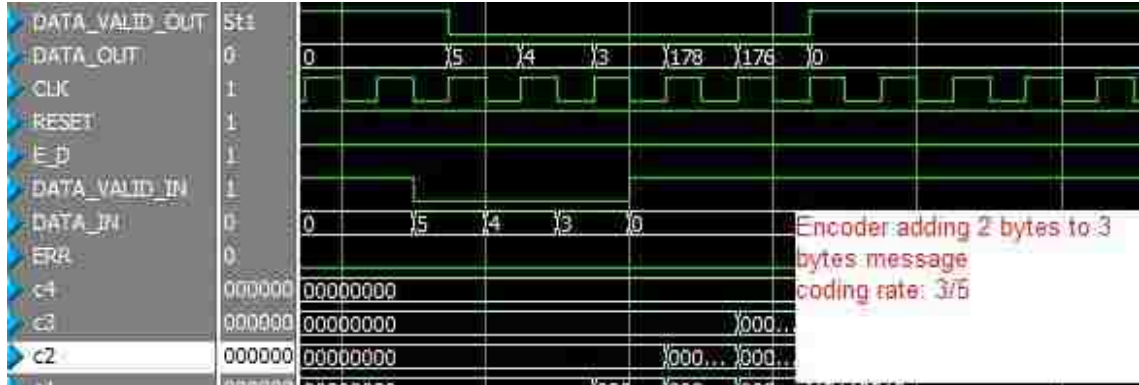


Figure 4.10: Functional Timing Simulation of ECC Unit (Reed Solomon (5, 3)) of 3D NAND Flash Controller

Figure 4.10 shows the functional timing simulation of Reed Solomon (5, 3) Encoder-Decoder ECC engine which corrects up to 8 bits.

ECC module 3 - Reed-Solomon (255,251) Decoder/Encoder

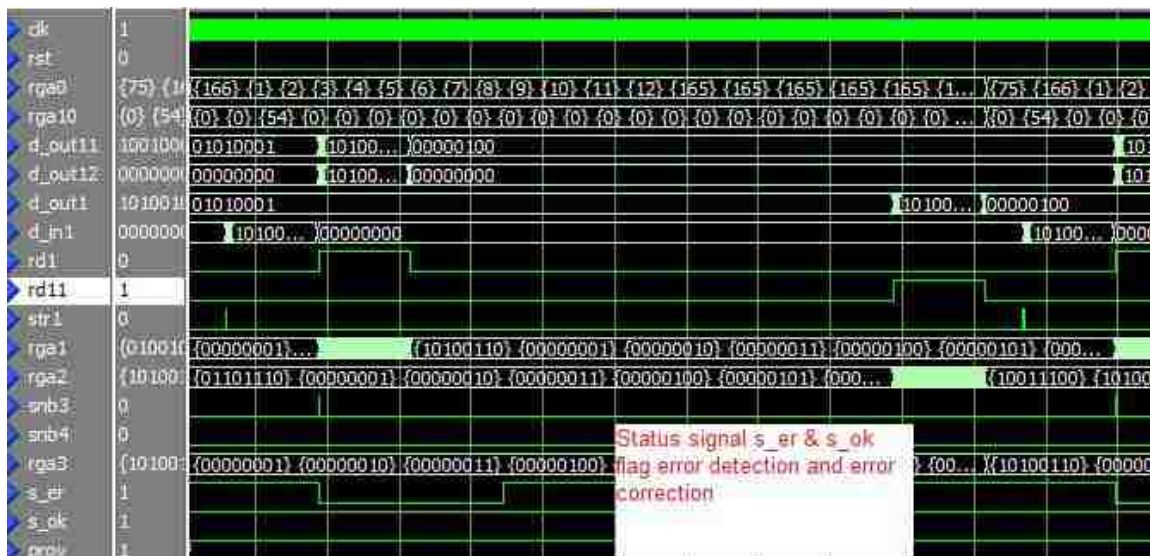


Figure 4.12: Functional Timing Simulation of ECC Unit (Reed Solomon (255,251)) of 3D NAND Flash Controller

Figure 4.12 shows the functional simulation of the Reed-Solomon (255,251) Decoder/Encoder which can correct up to 16 bits or 2 symbols

Controller functional simulation of writes and reads and remap due to errors.



Figure 4.13: Functional Timing Simulation of 3D NAND Flash Controller Data write

Figure 4.13 shows controller completing data write requests according to the mapping generated by the discovery phase.

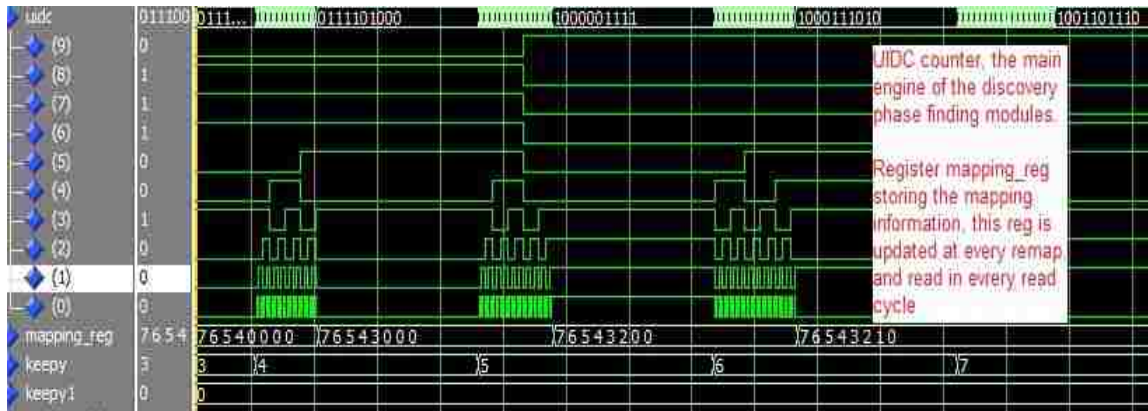


Figure 4.14: Functional Timing Simulation of 3D NAND Flash Controller showing UIDC counter and Mapping register

Figure 4.14 shows the simulation for the UIDC counter and the mapping register described in chapter 3.

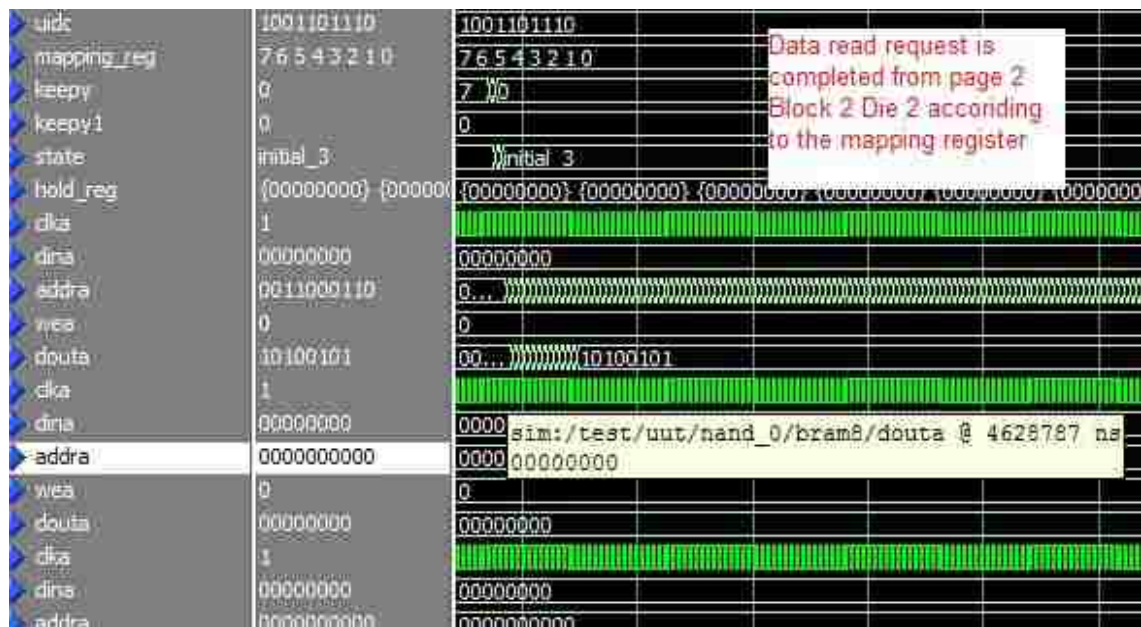


Figure 4.15: Functional Timing Simulation of 3D NAND Flash Controller Data Read

Figure 4.15 shows the simulation for completion of data read request

4.3.2 Controller space and power calculation of FPGA chip with 3 ECC modules

The power calculation is performed using the Xilinx Xpower analyzer. The total power consumed by the 3D Nand controller can be calculated with the following calculation:

Total Power = Total static power + Total dynamic power, Where

Total dynamic power represents the fluctuating power as your design runs. It represents the amount of power generated by the switching user logic and routing.

Total static power is the power drawn by the device when it is powered up, configured with user logic and there is no switching activity.

Table 4.1 summarizes the numerical results for the Nand flash controller implementation with 3 ECC modules.

ECC	Correction Capability (bits)	Slices Occupied	Operation Frequency (MHz)	Power (W)
Hamming code	1	632	207.419	0.7148
(5,3) RS code	8	863	186.935	0.7458
(255,251) RS code	16	13,736	137.071	1.0747

Table 4.1: Results for 3D NAND Flash Controller

The proposed 3D Nand flash controller is targeted on Virtex 4 FPGA that contains 26,624 slices. The simulation results show that the Nand controller using Hamming code uses a mere 632 slices on the Virtex 4 FPGA while consuming 0.7148 W of power but has a correction capability of just 1 bit and

runs at maximum frequency of 207.419 MHz while the controller with RS code (255,251) uses 13,736 slices and consumes 1.07 W of power but is capable of correcting up to 16 bits and runs at a maximum frequency of 135.017 MHz. The RS (5,3) code with a correction capability of 8 bits uses 863 slices and consumes 0.7458 W of power and runs at a maximum frequency of 186.935 MHz.

Thus the controller employing hamming code is more power efficient and requires less space on the chip. However, it can correct only one bit which is very unhelpful in harsh radiation environments. The controller employing Reed Solomon (255,251) code is requires more space and power but can correct up to 16 bits of faulty data.

Chapter 5

Conclusion

The cost versus capacity of memories for personal devices (such as cameras) has fallen significantly in recent years through conventional cost-reduction approaches such as incorporating smaller design rules. However, the effect of technology scaling is saturating in many types of ICs. Technology advancements in the form of 3D integration have made it possible to believe that the current pace of cost reduction can be maintained. It is believed that the first commercial application of 3D integration will be most likely in the commodity of memory space. Companies like Samsung, Tezzaron and Toshiba have come up with proof of concept 3D memory devices with vast densities and superior speeds.

With such groundbreaking features the advent of 3D memory in space related applications seems inevitable. However, a critical design challenge faced is the robustness of designs incorporating these 3D memories. Single event upsets have plagued electronic systems for a long time and are the major concern for space applications. Also, because of the relatively new fabrication techniques of 3D integration technology, yield problems are going to be inevitable till the processes are more standardized. So the effective use of 3D memories in space depends on the way these memories are controlled and protected against the potential dangers by incorporating fault-tolerance in them. Only the systems incorporating self test and repair can take full advantage of the 3D memory's special features.

The present work is focused on 3D Nand flash memory, a new memory controller system is proposed to address fault tolerance in 3D Nand flash memory. We show that the proposed design approach has very little detection/correction overhead and can revive the system against all single point hard errors and soft errors. The need for larger, cheaper, and more robust memories makes self-repairing property as a necessary condition for future memory designs.

References

- [1] Tan, Chuan Seng; Gutmann, Ronald J. and Reif, L. Rafael, Wafer Level 3-D ICs Process Technology, Series: Integrated Circuits and Systems, (Eds.) 2009, XII, 410 p. 145 illus., ISBN: 978-0-387-76532-7
- [2] Iyer S. S., "Three Dimensional Integration – Memory Applications." SOI conference 2009, IEEE, Page(s): 1- 5
- [3] Jared C. Smolens, Brian T. Gold, James C. Hoe, Babak Falsafi, and Ken Mai "Detecting Emerging Wearout Faults," The Third IEEE Workshop On Silicon Errors in Logic - System Effects (SELSE-3), Apr 2007
- [4] Abhishek Pillai, Wei Zhang, Dimitrios Kagaris., "Detecting VLIW Hard Errors Cost-Effectively Through A Software-Based Approach," 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)
- [5] A.H.Johnston., "Space Radiation effects in Advanced Flash memories" JPL.
- [6] Heather Quinn and Paul Graham, Xilinx Virtex FPGA Design Guide for Space, LA-UR- 08-04992
- [7] Paul E. Dodd, Lloyd W. Massengill, "Basic mechanism and modeling of single event upset in digital microelectronics," IEEE Transactions on nuclear Science, vol. 50, no. 3, June 2003
- [8] L. D. Edmonds, C. E. Barnes, L. Z. Scheick, An Introduction space radiation effects on microelectronics. JPL Publication 00-06

- [9] Aahlad Srinivasa M., "Single Event Upset Hardened CMOS Combinational Logic and Clock Buffer Design," Master's Thesis, University of New Mexico, December 2008.
- [10] Gary Swift., "Radiation effects and FPGAs." MAPLD06 seminar 2006.
- [11] Andry, P.S.; Tsang, C.; Sprogis, E.; Patel, C.; Wright, S.L.; Webb, B.C.; Buchwalter, L.P.; Manzer, D.; Horton, R.; Polastre, R. and Knickerbocker, J.; *A CMOS-compatible Process for Fabricating Electrical Through-vias in Silicon*. Proceedings of 56th electronic components and technology conference, San Diego, CA, pp 831-837
- [12] Tsang Ck, Topol AW(2006) 3D integrated circuits and silicon carrier packaging realization. Proceedings of 23rd VLSI VMIC Conference, Fremont, CA, September 25-28, 2006, VMIC no 06 IMIC-050 pp 61-69
- [13] Patel, C.S.; Tsang, C.K.; Schuster, C.; Doany, F.E.; Nyikal, H.; Baks, C.W.; Budd, R.; Buchwalter, L.P.; Andry, P.S.; Canaperi, D.F.; Edelstein, D.C.; Horton, R.; Knickerbocker, J.U.; Krywaczyk, T.; Kwark, Y.H.; Kwietniak, K.T.; Magerlein, J.H.; Rosner, J.; Sprogis, E.; *Silicon Carrier with Deep Through-Vias, Fine Pitch Wiring and Through Cavity for Parallel Optical Transceiver*. IBM T. J. Watson Res. Center, New York, NY. proceedings of 55th electronic components and technology conference, lake beuna vista, FL, pp 1318-1324
- [14] Chen KN, Tan CS, Fan A, Reif R(2004) "Morphology and bond strength of copper wafer bonding", *Electrochem Solid-State Lett* 7(1):G14-G16

- [15] Chen KN , Tsang CK, Topol AW, Lee SH, Furman BK, Rath DI, LU J-Q, Young AM, Purushothaman S, Haensch W (2006) "Improved manufacturability of Cu bond pads and implementation of seal design in 3D integrated circuits and packages", 23rd international VLSI Multilevel Interconnection(VMIC) Conference, fremont California. VMIC catalog No IMIC – 050- pp 195-202
- [16] Chen, Kuan-Neng ; Lee, Sang Hwui ; Andry, Paul S. ; Tsang, Cornelia K. ; Topol, Anna W. ; Lin, Yu-Ming ; Lu, Jian-Qiang ; Young, Albert M. ; leong, Meikei ; Haensch, Wilfried, Structure, Design and Process Control for Cu Bonded Interconnects in 3D Integrated Circuits. T. J. Watson Res. Center, IBM Corp., Yorktown Heights, NY. Electron Devices Meeting, 2006. IEDM '06. International. Session 13.5, pp 20- 22.
- [17] www.Tezzaron.com
- [18] Scott Chen, "What Types of ECC Should Be Used on Flash Memory?" Spansion application note.
- [19] TN-29-05, "ECC Module for Xilinx Spartan-3 Overview", Micron technical note.
- [20] Error control techniques, <http://cs.uccs.edu/~cs522/F2001code.pdf>
- [21] Hodgart M. S. (1992), "Efficient coding and error monitoring of spacecraft digital memory," International Journal of Electronics, 73: 1, 1-36.
- [22] Jitu J. Makwana, Dr. Dieter K. Schroder, A Nonvolatile Memory Overview, <http://aplawrence.com/Makwana/nonvolmem.html>

- [23] Paolo Pavan, Roberto Bez, Piero Olivo and Enrico Zanoni, "Flash Memory cells- An Overview," Proceeding of the IEEE, Vol. 85, No. 8, August 1997.
- [24] TN-29-19, "Nand Flash 101 Introduction", Micron technical note.
- [25] Zhang Biyong., "Physically Unclonable Functions," Kerckhoffs Institute & Technical University of Eindhoven.
- [26] Suh. G.E., Devadas. S., "Physical Unclonable Functions for Device Authentication and Secret Key Generation," Design Automation Conference, 2007, DAC, 44th ACM/IEEE, pages(s): 9-14.
- [27] Ryan L. Helinski, " Physical Unclonable functions,"
www.ece.unm.edu/~jimp/HOST/slides/RyansPUFslides.pdf.
- [28] XAPP 383, "Single Error Correction and Double Error Detection (SECCDED) with CoolRunner-II CPLDs", Xilinx Application note.
- [29] Unicore systems, "Reed-Solomon IP core", Rev 1.0 (2009).
- [30] Brendan Bridgford, Carl Carmichael, and Chen Wei Tseng, "Single-Event Upset Mitigation Selection Guide", XAPP 987, Xilinx Application note.
- [31] E. Takeda, K. Takeuchi, D. Hisamoto, T. Toyabe, K. Ohshima, and K.Itoh, "A cross section of α -particle-induced soft-error phenomena in VLSIs," IEEE transactions, Electron. Devices, Vol. 36, November 1989, pp. 2567-2575
- [32] B. Eitan and D. Froham-Bentchkowsky,"Hot-electron injection into the oxide in n-channel MOS devices,"IEEE Trans. Electron Devices, Vol. ED-28, no. 3, pp, 328-340,1981.

- [33] Lezlinger, M. and Snow, E.H(1969) Fowler-Nordheim tunneling in thermally grown SiO₂. Journal of Applied physics. 40, 278