AIAA Infotech@Aerospace 2010
20 - 22 April 2010, Atlanta, Georgia

AIAA 2010-3373

Self-Healing Adjustable Memory System

Naveen N. Purushotham[1] and Srikanth V. Devarapalli.[2]
Department of Electrical and Computer Engineering, Univ. of New Mexico, Albuquerque, NM, 87131

James Lyke[3]
Air Force Research Laboratory, Space Vehicles Directorate, Kirtland AFB, NM 87117-5776

and

Payman Zarkesh-Ha[4]
Department of Electrical and Computer Engineering, Univ. of New Mexico, Albuquerque, NM, 87131

The development of 3D integration has spawned the idea of a new generation of memory based on 3D stackable chips. It is believed that the first commercial application of 3D integration will be most likely in the commodity of memory space. In an era where there is a continuous demand for larger, faster, denser and robust memories, 3D stackable memory settles in perfectly. With technology advancements in space related applications, the need for memory storage in space systems has increased like systems in terrestrial related applications. Unfortunately 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. The present work proposes an efficient way to control a 3D stacked memory system along with a unique healing technique against physical errors in memory arrays, wear out faults, hard errors (e.g. stuck bits) and errors due to memory yield problems. It revives the system from all single point hard errors and detects all single point soft errors. It uses two main concepts for this purpose including PUF technology for memory mapping and RAID for data restoration. The estimated time delay increase due to using our proposed method is presented in this paper. The extra delay is a trade-off for achieving more robustness against a simple memory controller without any fault tolerance.

Nomenclature

UIDC = Unique ID for Specific Client
n = No. of Memory modules Assigned to a Specific Function including Memory module for Parity Information

I. Introduction

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 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. Some companies are considering vertical stacking of memory cells on a single silicon wafer. Many believe that 3D IC technology will make possible to maintain the current pace of cost reduction. 3D memory devices have vast densities, superior speeds and they consume less power. With such groundbreaking features, 3D stackable memory is sure to pave its way into the space industry because of the perennial need of memory storage in space related applications. However, like any other systems, the 3D memory stack is also prone to failures due to number of errors including physical

[1] Student, Department of ECE, UNM, Albuquerque, NM, 87106.
[2] Research Scholar, Department of ECE, UNM, Albuquerque, NM, 87106.
[3] Air Force Research Laboratory, Kirtland AFB, NM 87117-5776, and AIAA Senior Member.
[4] Assistant Professor, Department of ECE, UNM, Albuquerque, NM, 87106.

errors in memory cells, wear out faults, hard errors (stuck bits) and errors due space radiation environment. Hard wear out (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) [1, 2]. Other major type of error observed is the hard error or "stuck bits". One major cause for these errors is heavy ion induced hard errors[3,4]. Basic influence of natural space radiation environment on 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. 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. 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 paper 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 idea of "S*elf-Healing Adjustable Memory System*" is to have a global controller that 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.

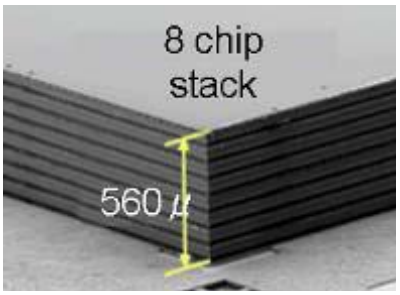

Figure 1. 3D Integrated Circuit. *Wafer-level processed stack package of memory*

In present work, Physical Unclonable Functions (PUF's) are used to assign unique identification code for each of the memory module. PUF's make use of process variations for giving an IC an identification, authentication, and activation. A simple ring oscillator in combination with a simple counter is used to take advantage of this fact and generate a basic PUF. The outputs generated from the ring oscillators on each of the memory modules will have a unique frequency. The outputs from the ring oscillator are given as clock inputs to the respective counters on each memory modules. The counter value generated from the counters will act as the PUF identities of the respective memory modules. An external pulse controls the run time of the counter. These PUF IDs are used for scanning through the memories and for collecting known good die information and then are later used by the global algorithms to rearrange the memories.

Apart from remapping the memory modules to the clients when errors occur, it is also necessary to restore the data lost in the memory modules. The degree of self healing of the system to a certain extent depends on the degree to which the data lost can be restored. Use of techniques such as modular redundancy is plausible but it imposes a large overhead on the system. A more efficient approach was adopted using RAID technology. Traditionally, RAID (Redundant Array of Independent Disks) has been used to support fast I/O performance and high reliability of storage system. RAID concept was introduced to provide high performance I/O subsystem by striping data across multiple low cost disk drives and storing parity information on one of the disks to generate the original data in case of a disk failure. Until now, the array of hard disk drive (HDD) is used to build up RAID systems[6.]

II. Background

This section briefly reviews 3D integration technology, RAID systems and PUF technology to facilitate better understanding of the 3D memory design and the proposed self healing adjustable memory system.

A. 3D Integration Technology

3D integration refers to a variety of technologies that provide electrical connectivity between stacked multiple active device planes. 3D integration technologies can be generally divided into three categories: (1) 3D packaging technology[7] (2) Transistor build-up 3D technology (3) Monolithic Wafer-level, back-end-of-the-line (BEOL) compatible 3D technology[8]

3D packaging technology is enabled by wire bonding, flip-chip bonding, and thinned die-to-die bonding. Transistor build-up 3D technology forms transistors layer by layer, on poly-silicon films, or on single-crystal silicon films. 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 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 μbumps 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 25μm 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 10μm 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 25μm TSVs is adequate for the 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 10μm[9].

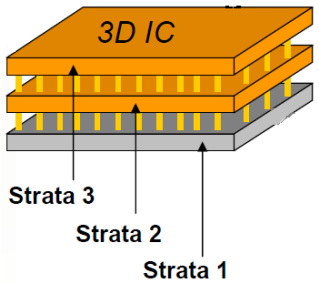

Figure 2. 3D IC.

B. RAID

RAID is a technology that provides high levels of storage reliability to computer storage systems (disk-drive components), it uses the technique of arranging the devices into arrays for redundancy. When multiple physical disks are set up to use RAID technology, they are said to be in RAID array. This array distributes data across multiple disks, but the array is seen by the computer user and operating system as one single disk. There are traditionally five standard levels of RAID, i.e. RAID 1 through RAID 5. The first level RAID is mirrored disks. It provides fault tolerance by mirroring data on a duplicate disk. It will function properly as long as one disk is functioning. The minimum number of disks for RAID 1 is two. In RAID 2, disks are synchronized and striped in very small stripes, often in single bytes/words. Hamming codes error correction is calculated across corresponding bits on disks, and is stored on multiple parity disks. The minimum number of disks for RAID 2 is three. RAID 3 and RAID 4 are striped set with dedicated parity while RAID 3 is byte-level parity and RAID 4 is block-level parity. The data is striped across two hard disks and the corresponding parity information is stored in the third disk. RAID 5 is the most commonly used architecture in RAID systems and is similar to RAID 4, except that the parity information is not stored on a dedicated drive. Instead the parity is interleaved across all the drives to make up a distributed parity system[10]. The minimum number of disks for RAID3 – RAID 5 is three.

C. PUF Technology

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 [11.] 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 [12]. 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. In our proposed technique, we use a simple ring oscillator PUF to generate a chip specific code as a result of the manufacturing variations among different chips. An illustration is shown in Fig. 3. 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.

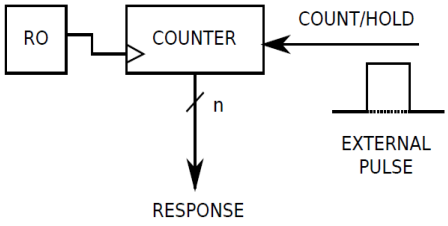

Figure 3. Simple Ring Oscillator PUF.

III. Memory Controller

Our proposed method is implemented using a global algorithm for rearranging the memories. The design is divided into three phases: (1) Discovery phase (2) Coalesce phase (run mode) (3) Heal phase

In the initial phase, information regarding the state and condition of each memory module in the stack is extracted and analyzed for stack organization. In the discovery phase, the known good die information is obtained. This information is used to mark the faulty memory modules by scanning through them individually. The discovery phase is a testing mechanism for the stack, the memory modules from the manufacturing phase can be tested for any irregularities in this phase. 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.

A. Discovery Phase

The first phase of the global algorithm is the scan and mark phases. The controller, first performs a sector by sector read back operation on the complete stack and collects information (known good die) regarding the state of each die. This is called the scan phase. The mark phase starts with the initiation of the PUF values. The PUFs used here are simple ring oscillator PUFs, each known good die is marked with a specific ID generated by the ring oscillator PUF. The spoiled memory modules are left alone with null values. The PUF values that we consider are 10 bit long, but can be modified to any custom length in order to achieve unique values. 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 finds and assigns a set of *n* memory modules to that particular function. Since the data is to be striped across multiple memory modules the value of *n* is decided by the number of memories that we choose to store the data in. If that number is chosen to be *n-1*, then the number of modules assigned is *n* since an additional memory is required to store the parity information. When a set of memory dies are assigned to a particular function it is first discovered by the controller using the UIDC counter. UIDC counter is a simple 10-bit counter. Counter value "0" is assigned for spoilt memory die. When creating the new mapping the controller first increments the UIDC counter and compares it against all the PUF values. If match is found the memory is discovered 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 assigned to the function for use. The same action is repeated, if multiple function request access. After all the functions are mapped the controller goes into the next phase, which is the coalesce or the run phase.

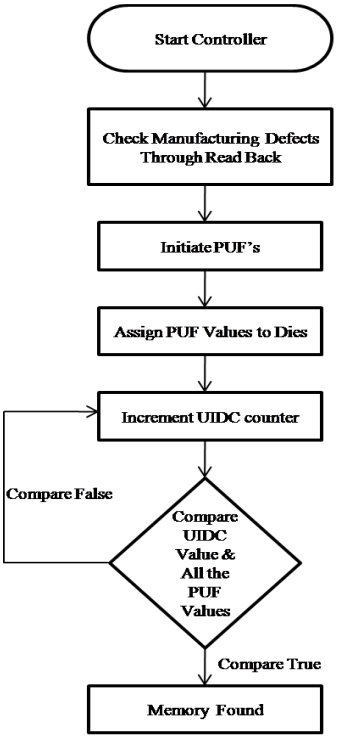

Figure 4. Discovery Phase. *Flow chart showing the algorithm for the discovery phase.*

B. Coalesce Phase (Run mode)

The Coalesce phase is basically the run mode for the system. After achieving a continuous map in the discovery phase. The controller now asserts a system ready signal to the function so that it can access memory. During a data write request the controller calculates the parity for the data bits during each write cycle. When the function writes *n-1* bits to an address, the controller arranges these *n-1* bits in *n-1* memory modules at the same address and the *n*th memory module gets the parity information corresponding to the data bits at the same address as shown in the Fig. 5. This arrangement helps the controller to restore data during a memory module failure. For example, if the function writes 8 bits to the memory at an address 0100011011 first the parity for the 8 bits is calculated, the parity used here is a simple odd even parity. The 8 bits are written across 8 memories at the same address and the 9th memory contains the parity bit for the corresponding 8 bits.

Similarly During a data read request the controller first calculates the parity for the *n-1* data bits and

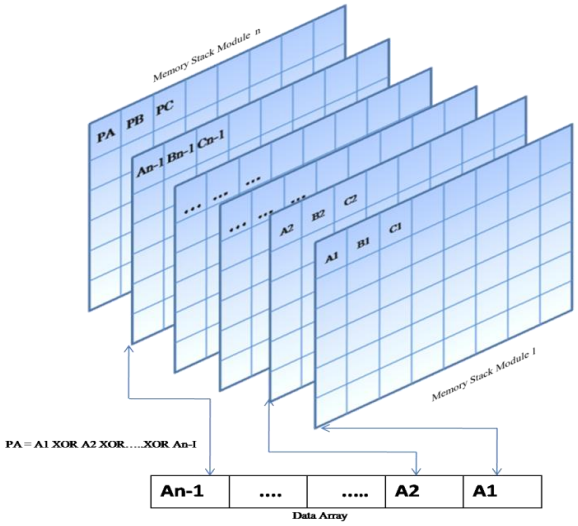

Figure 5. Data access arrangement in the stack.

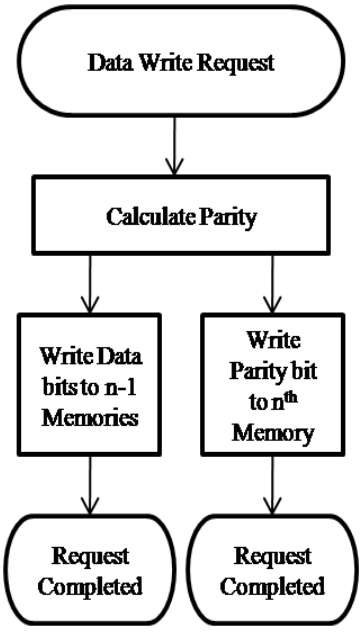

Figure 6. Coalesce Phase. *Flow chart showing the algorithm for the Data Write Request.*

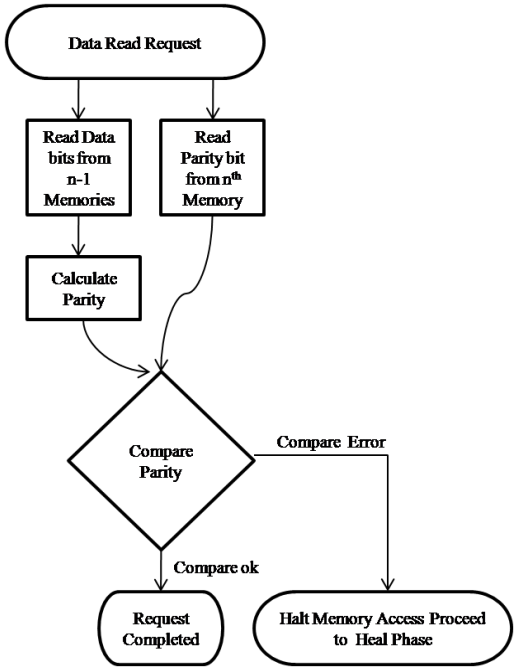

Figure 7. Coalesce Phase. *Flow chart showing the algorithm for the Data Write Request.*

compares it with the corresponding parity bit in the *n*th memory module. If the parity matches the data, read request is completed. If the parity doesn't match then an interrupt is asserted indicating the halt of memory access for data correction and the controller now proceeds into the heal phase for memory remap and data restoration. The flow chart for both the data write request and data read request are shown in Figs. 6 and 7.

C. Heal Phase

The heal phase is a correction phase in which the faulty die is excluded from the memory map and replaced with a new one and the data lost in the faulty die is restored. This is done from the parity information and the data from the other modules in the set. The first action taken by the controller is the identification of the faulty module. This is again done by read back. Once a parity mismatch occurs in the coalesce phase, an interrupt is sent to function to stop accessing the memory and in the heal phase a read back operation is performed on the last read address on all the memory dies in the set. This operation consists of writing '0' and '1' subsequently and reading them back, the faulty die will give back a bad bit in the presence of a hard error. When fault is discovered, the faulty die is excluded from the map and the controller proceeds to discovery phase to find a new die to replace it. Once a new die is selected within the discover phase, it is included in the new mapping for the respective function using recoalesce. The data lost in the faulty die is restored by traversing through each address and calculating data bits through parity information. Fig. 8 shows an illustration of data restore operation. For example if the memory stack module is the faulty die, then it is excluded from the map and is replaced by a new memory and the data can be restored by the following operation

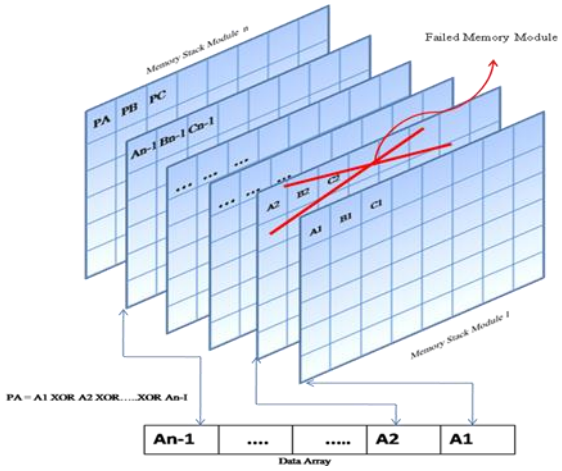

Figure 8. Data restore arrangement in the stack.

The bit A2 = PA xor A1 xor …xor An-1

The bit B2 = PB xor B1 xor …xor Bn-1 and so on..

If the faulty die was found to be the memory module with parity information then the new found die is stored with parity information with a similar calculation. Once this is done the system resumes access to memory like usual. The time lost in data restoration and remap can be decreased many fold by sectorizing the memory. In a sectorized memory the whole memory might not be excluded from the map but only a certain sector of it can be excluded depending on the extent of faults in the die. This decreases a lot of overhead on the system by removing excess parity calculation and also saves a lot of space on the memory stack.

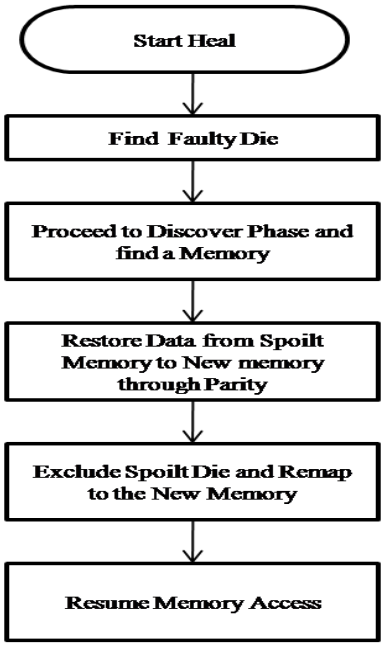

Figure 9. Heal Phase. *Flow chart showing the algorithm for the Heal Phase.*

D. Performance Analysis

The total time delay incurred in the system by using our method as a trade-off for achieving more robustness against a simple memory controller without any fault tolerance is calculated by the following analysis:

Case 1: System has no error

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

Where

$\tau_{\text{self-healing system}}$ = Time taken by Self Healing Adjustable System

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

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

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

$\tau_{\text{Parity calculation in write cycle}}$ = Sum of the Time taken to calculate parity during each write cycle

$\tau_{\text{Parity calculation in write cycle}}$ = Sum of the Time taken to calculate parity during each read cycle and comparing it with the existing parity information

Case 2: System with error

$$\tau_{\text{self-healing}} = \tau_{\text{normal-system}} + \tau_{\text{initial readback}} + \tau_{\text{PUF Calculation}} + \tau_{\text{Parity calculation in write cycle}} + \tau_{\text{Parity calculation in read cycle}} + \tau_{\text{data restore \& remap}}$$

Where

$\tau_{\text{data restore \& remap}} = \tau_{\text{readback}} + \tau_{\text{Parity calculation at each memory cell to restore data}}$

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

$\tau_{\text{Parity calculation at each memory cell to restore data}}$ = Sum of the Time taken to restore data bits at each memory cell by performing the Boolean xor operation on parity and remaining data bits.

IV. Conclusion

A new self-healing adjustable memory system is proposed to address fault tolerance in 3D memory architectures. We show that the proposed design approach has very little detection/correction overhead. The need for larger, cheaper, and more robust memories makes self-healing property as a necessary condition for future memory designs.

V. Future work

The current proposed technique does not support corrections for soft errors. Our next step in our research plan to develop a technique to heal the errors created by single event upsets.

Acknowledgments

This work was funded by Configurable Space Microsystems Innovations & Applications Center (COSMIAC), located in Albuquerque, NM.

References

[1] 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

[2] 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)

[3] Koga.R., Crain. S, Crawford. K., Yu. P.," Heavy ion induced hard errors in memory devices with sub-micron feature sizes," Radiation and Its Effects on Components and Systems, 2001. Page(s): 423 – 430

[4] Xapsos, M.A., "Hard Error dose distributions of gate Oxide arrays in the laboratory and space environment" Nuclear science, IEEE transactions on, Volume: 43, Issue:6, Part 1, Publication Year: 1996 , Page(s): 3139 - 3144

[5] A.H.Johnston., "Space Radiation effects in Advanced Flash memories" JPL.

[6] Kwanghee Park, Dong-Hwan Lee, Youngjoo Woo, Geunhyung Lee, Ju-Hong Lee, Deok-Hwan Kim., "Reliability and Performance Enhancement Technique for SSD array storage system using RAID mechanism" Communications and information technology 2009, ISCIT 2009, 9th International symposium on Digital Object, 2009, Page(s): 140 – 145

[7] Dreiza. M, Yoshida. A, Ishibashi. K, Maeda. T. , "High density pop (package-on-package) and package stacking development" Electronic Components and Technology Conference, 2007. ECTC '07. Proceedings. 57th Page(s): 1397 - 1402

[8] J.Q. Lu, T.S. Cale, and R.J. Gutmann, "Wafer-level three-dimensional hyper-integration technology using dielectric adhesive wafer bonding," *Materials for Information Technology: Devices, Interconnects and Packaging (Eds. E. Zschech, C. Whelan, T. Mikolajick)*, pp. 386–397, Springer-Verlag (London) Ltd, August 2005.

[9] Iyer S. S., "Three Dimensional Integration – Memory Applications." SOI conference 2009, IEEE, Page(s): 1- 5

[10] David A Patterson, Garth Gibson, and Randy H Katz. " A case of redundant array of inexpensive disks". COMPCON Spring '89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers. Year 1988

[11] Zhang Biyong., "Physically Unclonable Functions," Kerckhoffs Institute & Technical University of Eindhoven.

[12] 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.