Performance Analysis and Design of Systems Under Power and Reliability Constraints

Jona Babi
babijona@inf.uth.gr
July 4, 2012

Advisor: Georgios Stamoulis (University of Thessaly) georges@inf.uth.gr
Co-advisor: Nestoras Eumorfopoulos (University of Thessaly) nestemvo@inf.uth.gr

The thesis was carried out in collaboration with the Telecommunication Circuits Lab of the Swiss Federal Institute of Technology, Lausanne (EPFL).
I would like to thank
Dr. Georgios Karakonstantis and Prof. Andreas Burg for their idea, for the inspiring discussions and their support.

Ανάλυση απόδοσης και σχεδίαση συστημάτων ύπο περιορισμούς ισχύος και αξιοπιστίας

Πριν την τεχνολογία νανομέτρων, τα κύρια κριτήρια σχεδιασμού ολοκληρωμένων κυκλωμάτων ήταν η βελτιστοποίηση για ταχύτητα, έκταση και κατανάλωση ησχύος παραμελώντας τις φυσικές και ηλεκτρικές ιδιότητες των τρανζίστορ, οι οποίες παίζανε λιγότερο σημαντικό ρόλο. Συγκεκριμένα, κατά τη διάρκεια ζωής τους, τα τρανζίστορ θεωρούνταν ότι συμπεριφέρονται νομοτελειακά όπως είχαν σχεδιαστεί. Αυτό ήταν περίπου μέχρι το μέγεθος των 0,35 μm, επειδή το μέγεθος του τρανζίστορ ήταν πολύ μεγάλο σε σχέση με τα διαφορετικά είδη των διαδικαστικών διακυμάνσεων. Ωστόσο, με τη συνεχιζόμενη κλιμάκωση των κόμβων της τεχνολογίας μέχρι τα 65 nm, η φύση των μη ντετερμινιστικών ιδιοτήτων των τρανζίστορ άρχισε να αποκτήσει σημασία για τη σωστή λειτουργία των συσκευών. Στα 65 nm,επειδή το μέγεθος του τρανζίστορ είναι τόσο μικρός, η διαδικασία κατασκευής πάσχει από ασάφειες σχετικά με τα μεγέθη συσκευών που οδηγούν σε διαφορετικές παραλλαγές στα χαρακτηριστικά των τρανζίστορ. Τέτοιου είδους διαδικαστικές διακυμάνσεις περιλαμβάνουν τις χωρικές και χρονικές παραμετρικές διακυμάνσεις και απειλούν τη σωστή λειτουργία των νανομετρικών κυκλωμάτων.

Η λειτουργία των κυττάρων, στατικής τυχαίας προσπέλασης (SRAM) επηρεάζεται πολύ από όλες τις προαναφερθείσες διαφοροποιήσεις σε μεγαλύτερο βαθμό από ό, τι λογικά κυκλώματα. Τα αποτελέσματα των διακυμάνσεων στα κύτταρα μνήμης είναι πιο κυρίαρχα, δεδομένου ότι τρανζίστορ με πολύ μικρές διαστάσεις χρησιμοποιούνται για την κατασκευή της μνήμης, ώστε να ελαχιστοποιηθεί η κατεχόμενη περιοχή για την υψηλότερη πυκνότητα μνήμης. Οι συνέπειες των μεταβολών στο κελί SRAM εξαρτούνται από το είδος της μεταβολής που βιώνει. Για παράδειγμα, παραμετρικές μεταβολές οδηγούν σε διαφορετικές δυνάμεις των τρανζίστορ που στη χειρότερη περίπτωση προκαλούν αποτυχία του κυττάρου. Συγκεκριμένα, οι διάφορες αποτυχίες είναι οι εξής: αποτυχία του χρόνο πρόσβασης, αδυναμία διατήρησης, αποτυχία γραψήματος και αποτυχία διβάσματος. Εκτός από τις συνεχείς αποτυχίες μνήμης που προκαλούνται κατά τη διάρκεια της κατασκευής, υπάρχουν επίσης μη εμμένουσες αδυναμίες της μνήμης που δεν βλάπτουν το κύτταρο σε μόνιμη βάση, που οφείλονται κυρίως σε σωματίδια άλφα .

Οι σχεδιαστές έχουν καταλήξουμε σε διάφορες τεχνικές στο κυκλωματικό επίπεδο και στο επίπεδο της αρχιτεκτονικής, όπως κλιμάκωση του μέγεθος του τρανζίστορ, προσθέτοντας επιπλέον γραμμές / στήλες στη μνήμη του, προσθέτοντας τρανζίστορ σε ένα κελί μνήμης, τροποποίοντας δυναμικά το μέγεθος της μνήμης ή προσθέτοντας κώδικες διόρθωσης λαθών. Δυστυχώς, όλες αυτές οι τεχνικές υπάρχουν εις βάρος του αυξημένου μέγεθου του καλουπιού και της αυξημένης κατανάλωσης ισχύος, δεδομένου ότι βασίζονται στην προσθήκη πλεονάζον υλικό για τον εντοπισμό και τη διόρθωση τυχόν σφαλμάτων, ως εκ τούτου έρχεται σε αντίθεση με τα άλλα κύρια πρόκληση στο σχεδιασμό κόμβων νανομέτρων που είναι η κατανάλωση ενέργειας.

Επομένως, υπάρχει ανάγκη να αντιμετωπισθούν τέτοια ζητήματα, να μελετήθουν και να αναλύθουν και να βρεθούν νέες μεθόδους που επιτρέπουν την αξιόπιστη λειτουργία σε χαμηλή ισχύ και μικρό κόστος. Για το σκοπό αυτό, στην παρούσα διπλωματική εξετάζουμε μια νέα τεχνική για την αντιμετώπιση των ενδεχόμενων βλαβών της μνήμης, η οποία δεν έχει διερευνηθεί μέχρι στιγμής.

Η τεχνική εξερευνεί την αντοχή της αναπαράστασης δεδομένων (ο τρόπος που τα δεδομένα αποθηκεύονται στη μνήμη) σε επίπεδο συστήματος. Θα μπορούσε να είναι δυνατό διαφορετικές αναπαράστασεις δεδομένων να είναι πιο αξιόπιστες κάτω από ελαττώματα του υλικού με δεδομένη την κατανομή των εισαγόμενων δεδομένων. Για το σκοπό αυτό, στόχος μας είναι να βρούμε κάποια αναπαράσταση των δεδομένων που ελαχιστοποιεί τα μη-επίμονα σφάλματα στη μνήμη. Πιο συγκεκριμένα, για να επιτύχουμε το στόχο μας έχουμε σχεδιάσει έναν εξομοιωτή σφαλμάτων σε επίπεδο συστήματος. Η απόδοση αξιολογείται με βάση το μέσο τετραγωνικό σφάλμα των δεδομένων που γράφονται και διαβάζονται από τη μνήμη. Για μικρά μεγέθη bit είναι δυνατό να εκτελέστει μια εξαντλητική αναζήτηση για να βρούμε την καλύτερη αναπαράσταση των δεδομένων. Είναι ενδιαφέρον ότι τα αποτελέσματα έδειξαν ότι υπάρχουν πολλές αναπαραστάσεις που έχουν ένα καλύτερο μέσο τετραγωνικό σφάλμα από τις αναπαραστάσεις που χρησιμοποιούνται σήμερα για την αποθήκευση των δεδομένων στη μνήμη. Επιπλέον, έχουμε αναπτύξει έναν αλγόριθμο που εκτελεί το έργο της εύρεσης της καλύτερης αναπαράστασης. Η απόδοση του αλγορίθμου αξιολογείται με τη χρήση της Gaussian κατανομής για διάφορες παραμέτρους ώστε να προσεγγισθούν διαφορετικές κατανομές, όπως η Gaussian, Uniform, Άθροισμα Gaussians, Rayleigh και Laplace οι οπόιες μοντελοποιούν καλά κατανομές δεδομένων από δημοφιλείς εφαρμογές. Σε όλες τις περιπτώσεις, ο αλγόριθμος βρίσκει αρκετά καλές αναπαραστάσεις με σημαντικά λιγότερη υπολογιστική προσπάθεια και χρόνο από την εξαντλητική αναζήτηση ή την τυχαία αναζήτηση. Τα αποτελέσματα δείχνουν ότι ο αριθμός των αναπαραστάσεων οι οποίες είναι καλύτερες από την αναπαράσταση του συμπληρώματος ως προς δύο και η διαφορά μεταξύ της καλύτερης αναπαράστασης που βρέθηκε και της αναπαράστασης του συμπληρώματος ως προς δύο από την άποψη του μέσου τετραγωνικού σφάλματος εξαρτάται σημαντικά από την κατανομή των δεδομένων εισόδου και των παραμέτρων της.

Abstract

The impact of hardware defects and the resultant errors on the performance and yield of memory, potentially induced by voltage scaling and parametric variations, is analyzed. For modeling the memory and capturing the impact of the defects, a system-level fault simulator is designed. The performance is assessed based on the Mean Squared Error of the data written and read from the memory. An exhaustive search for the best data representation, which minimizes this Mean Squared Error in the memory, is performed for small bit sizes. Furthermore, an algorithm is presented to perform the task of finding the best representation with significantly less computational effort. The algorithm is analyzed using various input data distributions. The impact of the algorithm on memory yield and power is analyzed. To this end, a system model of a wireless communication channel containing a decoder is combined with our error resilient memory model and the improvement of the output quality is analyzed.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1
Introduction

Before nanometer technologies emerged, the main design criteria for integrated circuits were optimization for speed, area and power consumption neglecting the physical and electrical properties of the transistors which played a less important role. Specifically, during their device lifetime, the transistors were assumed to behave deterministically as they were designed. That was approximately until a feature size of 0.35 μm, because the transistor size was very large compared to the different kinds of process variations. However, with the continuing scaling of technology nodes down to the 65 nm regime, the nondeterministic nature of transistor properties started to become relevant for the proper operation of the devices. At this size, since the transistor size is so small, the manufacturing process suffers from larger imprecisions regarding device sizes that lead to different variations in transistor characteristics. Such process variations include both spatial and temporal parametric variations and threaten the correct functionality of nanometer circuits.

Parametric variations are categorized into two groups: intradie and interdie [1]-[3]. On one side, the variations of transistor strength that occur within the same die are called intradie. They can be caused mainly by random dopant fluctuations in the channel region resulting from lithography, chemical and mechanical polishing, line edge roughness and change of geometric dimensions compared to the specifications due to lithographic and etching techniques during the fabrication [1], [4]-[9]. Modeling the intradie variations is difficult because a large number of random variables are required [3]. On the other side, interdie variations are observed between different dies that might have been produced from different wafers. The causes for interdie variations are the fluctuations in length due to variations in the exposure time during fabrication and fluctuations in transistors width and oxide thickness.

Temporal variation is another problem that arises with decreasing transistor size. The temporal variations can be divided into two categories: aging-related and environmental. Aging variations include negative bias temperature instability (NBTI) [10]-[26], positive bias temperature instability [27], [28], hot carrier injection [29]-[31], time-dependent dielectric breakdown [32]-[34] and electro-migration [35]. Using the device for a long time decreases its strength because of aging issues (i.e. NBTI). Moreover, transistors experience temperature and voltage variations that are classified as environmental variations and associated with the variations in circuit functionality. For instance, temperature fluctuations occur when the large power consumption of a circuit is transformed into heat and the device is not equipped with mechanisms to limit the rise of temperature.

All the above variations result in large fluctuations of transistor threshold voltage (V_{th}), which degrade the overall system performance by affecting both logic (digital and analog) and memory. The variations in logic manifest increased delay and spread of the delay distribution. As a result, the delay specifications are not met and the yield is degraded [1], [2].

The operation of static random access memory (SRAM) cells is highly influenced by all the mentioned types of variations to a larger degree than logic circuits. The variation effects on memory cells are more dominant, since transistors with very small dimensions are used to build the memory in order to minimize the occupied area for higher memory density. The consequences of the variations on the SRAM cell depend on the type of variation that the cells experience. For instance, parametric variations lead to different strengths of transistors that in the worst case cause failure of

the cell. Specifically, different failures that can occur are: access time failure, retention failure, write failure and read failure. An access time failure is the violation of the delay specifications and it takes place during a read operation when the differential voltage across the bit lines is not sufficient to identify the correct value. A hold or retention failure is the loss of the cell information when the cell is in standby mode, for example as a result of a voltage drop. Write failures happen when it is not possible to write the cell, because the cell content cannot be toggled. Read failures occur when the data stored in the cell is flipped and the read value is not the correct one. Such failures are mainly the result of noise developed in the node that stores the zero that is becoming larger than the trip point of the inverter. The size and type (i.e. 6T, 8T) of the memory bit cell, the array organization and the supply voltage determine the degree of such failures.

During memory development, it is common that designers try to operate memories in the lowest possible voltages, since voltage downscaling is considered as one of the most effective methods for power reduction. However, operation at low voltages in combination with variations makes circuits very prone to failures [36]. Figure 1 depicts the dependence of the failure probability on the supply voltage for different cell types: medium sized 6T bit cells, 6T cells upsized by 15% and 8T bit cells in a 65 nm technology node [37].

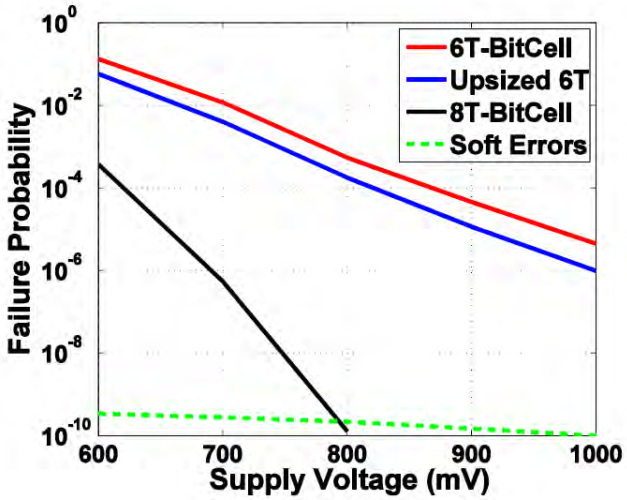


Figure 1: Memory failure probability (65nm)

Except of the persistent memory failures that are caused during manufacturing (e.g. stack-at -faults), there also exist non-persistent memory failures that do not damage the cell permanently, i.e. soft errors or erratic bit failures which are primarily due to alpha particles. Even if soft error rate slightly reduces as transistor size scale down, the number of errors increases because the probability of hitting a larger number of transistors by a particle increases [38]. Although the failure probability of the cell increases when soft errors happen, the overall failure rate in memories is still dominated by random dopant fluctuations induced errors [39] as it is shown in Figure 1.

Both persistent and non-persistent errors have a significant impact not only on the memory system, but also on the overall system. For instance, in a cache of a modern processor, a bit flip error during the decoding phase of the instructions may result to execute erroneously an instruction that may lead to complete failure of the system. The persistent errors cause defected cells which cannot be used for reliable storage having as a consequence the reduction of yield and thus the increase of manufacturing cost.

Designers have come up with various techniques at a circuit and architecture level, such as scaling the transistor size, adding extra rows/columns in the memory [40], [41], adding transistors to a memory cell [39] or modifying the memory size dynamically [42]. Unfortunately, all these techniques come at a cost of increased die area and power since they are based on the addition of

redundant hardware for the detection and correction of any errors, thus contradicting with the other main design challenge in nanometer nodes that is the reduction of power consumption.

Therefore there is a need to deal with such issues, study and analyze them and find new methods that allow reliable low power operation at a minor cost. To this end, in this report we investigate a new technique for dealing with potential memory failures, which has not been explored so far. Our approach exploits the resilience of data representation (the way that data is stored in the memory) at the system level for improving the output quality and the yield as well as the power consumption (by allowing lower voltages) of the memory. For this purpose a system level fault-simulator is developed, a performance metric and various input data distributions are analyzed. Furthermore, an algorithm that finds the best data representation in terms of the chosen performance metric is developed. The algorithm impact is assessed using a wireless communication system.

The rest of the report is organized as follows. The next chapter (Chapter 2) introduces an overview of the state of the art techniques that are designed to solve the above issues .Chapter 3 describes the system model of our solution and gives a general overview of the input data, performance metrics and the data representation in the memory. The following chapter (Chapter 4) discusses some simple, straight forward solutions. The quality of the solutions is assessed by comparison with results from Monte Carlo simulations. The next chapter (Chapter 5) presents our solution to the problem. A new algorithm is discussed. In chapter 6 the results of the new algorithm are presented and compared with the simple solutions of chapter 4. The impact of the algorithm on the output quality of a wireless communication system is shown. Moreover, the improvement of the memory yield and the power consumption is discussed.

Chapter 2
State of the art

Today there exist a lot of different techniques on different levels of the design hierarchy that address memory failures. Most of such techniques are based on the addition of redundant hardware or the design with safety margins achieved by transistor up-scaling. While such techniques provide sufficient robustness, they unfortunately come at a cost of increased area and power overhead.

2.1 Circuit Level

An important circuit level solution is scaling of the transistor size since failure probability and array yield strongly depend on the transistor size of a memory bit cell as well as the relatives ratios between them [43], [44]. The length and the width of different transistors in a SRAM cell affect the cell failure probability by modifying the nominal values of access time, the trip point of the inverter, read voltage, write time and minimum retention voltage as we discussed briefly in Section 1.

Some other circuit level solutions have to do with supply voltage (VDD) scaling. Designers limit or even up-scale the lower voltage in order to ensure stronger noise margins and more reliable operation. Another technique used after fabrication to compensate mainly the inter process variations is Adaptive Body Bias [28]. Specifically, each die is set to its own biasing voltage to reach the best frequency. This solution does not work for intradie variations, because a lot of different voltages are needed. It can work only if the die is divided into regions and if each region has a different body biasing.

2.2 Redundant Rows and Columns

Adding extra rows/columns is used to address the persistent manufacturing faults and consequently to improve the yield of SRAM arrays [41]. Specifically, a small number of redundant columns of SRAM cells is added to the array during the design stage. During the testing stage that takes place after the manufacturing, the defective cells are identified and the array is reconfigured to access one of the redundant columns instead of the column which contains the defective cell.

2.3 Bit Cells

Another approach is increasing the number of transistors in a cell. The 8T and 10T SRAM cells isolate the read and the write operations from each other, so it is much easier to optimize each of them separately under low voltage [45]-[50]. The area overhead of such bit-cells amounts to more than 30% and is a good price for the gained robustness, but these cells still suffer from pseudo-read errors because two or more of the words in a row share one word line.

Some of the SRAM cells have a build-in feedback mechanism that achieves process variation tolerance under a very low supply voltage. The design is based on the idea of Schmitt Trigger differential sensing. Overcoming the disadvantage of the read and write operations that the normal 6T cells have, Schmitt Trigger operation ensures a better read and write stability [48], [49].

2.4 High Level Solutions

A high level solution to the variations is modifying the memory architecture. For instance, a popular technique that is followed in cache memories in modern processors is designing a dynamical cache that resizes itself [51]. The resize happens when the cache finds not-working cells and maps them to working spare cells. Another way is a mechanism that deletes all the caches lines that are considered faulty. Cache line fault detection is implemented using an availability bit for each line [52], [53]. A cache line can be deleted only if the availability bit is turned off. A different way to locate the failures is using on-chip reliability sensors.

Cell-flipping technique is another high level technique to address the variation inside memory cells [16]. It relies on a memory structure that flips the cell state taking advantage of the unbalanced signal probability. The aim of this technique is the reduction of the Negative Bias Temperature Instability, allowing the PMOS operation to have periods of relaxation in between the continuing stress. Therefore the threshold voltage increase recovers dynamically.

Two other techniques that enable cache lines to operate at low voltages despite very high memory cell failure rates are the Word-disable scheme and the Bit-fix scheme [42]. The schemes identify and disable different sizes of the cache, individual words or pairs of bits. In the Word-disable scheme, if a 32-bits word contains defective bits, it is disabled. If a word is disabled, the other words are combined to form a line that has only non-failing words. Experimental results show that this scheme permits a 32kB cache to operate at 490mV. The Bit-fix scheme stores the location of the erratic bits and the correct value of the same bits. It allows a 2MB cache to operate at 475mV. During high voltage operation both schemes use 100% of the cache. At low voltages they sacrifice 50% and 25% cache capacity respectively.

2.5 Error Correcting Codes (ECC)

Instead of improving the structure of the cell of a circuit, a designer can build Error Correcting Codes (ECC) into the array. ECC is being added to data storages. It detects the errors at the data and when necessary corrects them. It is different from parity bits that only detect errors. When a word is sent to the memory to be stored, ECC calculates a code that corresponds to the bit sequence which is also stored along with the word. When a data word needs to be read the code is calculated again by the original algorithm and if the code stored in the memory and the one just calculated match, the data is free of errors and can be read. If not, then the word is corrected based on the comparison of the codes.

To correct the word that is going to be read from the storage, a simple ECC solution is to repeat each bit *n* times in order to create a group for each original bit and then consider as the correct bit the majority of the read bits in each group. For example with $n = 4$ the word 110 will be extended to 111111110000. This technique works for a minimum of $n = 3$. If the number of errors is smaller than $n/2$ in each group, then the word will be recuperated in every case. However, this approach is not practical, because when n increases, the overhead also increases.

2.5.1 Hamming Code and Extended Hamming Code

The Hamming code permits correcting single bit errors, by adding control bits to the information. However, a single error correction is not enough for many applications. Some other kind of ECC, called Double-bit Error Detection (SEC-DED) or extended Hamming correction codes can correct more errors but they also need to add to the control bits additional parity bits compared to the Hamming code.

2.5.2 Low-Density Parity-Check Codes (LDPC)

Low-Density Parity-Check Codes (LDPC) are encodings that use parity bits. The main idea is that each parity bit checks a number of bits and each bit is checked by a number of parity bits. One of the advantages of LDPC is that every code word has the same hamming distance from all the others. LDPC have two decoding algorithms: the first one check each digit and compares it with its parity check operators. Depending on what the majority of them indicate, the bit is flipped or not. The digits are checked repetitively until all bits settle to a fixed state. The second algorithm is similar to the first one but more accurate. Instead of deciding based on what the majority of the parity bits indicate about a bit, it computes the probability that a certain bit is one, taking into account all the other bits. It cycles through the bits until the bits are in a static state. These algorithms are not deterministic, since they depend on the order that the bits are checked.

2.5.3 Convolutional Codes

Convolutional codes process blocks not as an independent entity. They take into consideration the result of the previous bits. The initial state has a key that is known from both encoder and decoder. Each input bit is processed in multiple ways and has different output results. Therefore each output bit represents the results of different input bits. Convolutional codes can be implemented by different algorithms but they all depend on two variables: how many bits each state has and how many bits are produced per input bit. The main categories are the systematic and the recursive algorithms. The systematic code has an output that only depends on the input. The recursive code uses a prior output as part of the new input. The convolutional codes are fast and efficient. Their main disadvantage is that their accuracy depends on the input.

2.5.4 ECC Use for Improving Memory Reliability

The different ECC groups are exploited in order to create robust memories, i.e. some techniques utilize the SEC-DED [54] or the LDPC codes [55]. One of the approaches uses a two dimensional ECC that corrects multi-bit errors [56]. Specifically, this scheme is appropriate when contiguous bits in multiple rows and columns fail concurrently. The location of the defective bits defines the success of this scheme. If the defective bits are randomly distributed in each cache line, which happens as a result of random dopant fluctuations, this technique fails.

Another technique, called multi-bit segmented ECC (MS-ECC), addresses both persistent and non-persistent failures by using Orthogonal Latin Square Codes at the cost of more check bits [38], [57]. MS-ECC is an adaptive mechanism that enables different parts of the memory to be used for error correction. To increase the reliability during low-voltage operation, some associative ways in each cache set are used to store the ECC check bits for the remaining ways. The operating system can chose the number of associative ways used for storing ECC depending on the desired reliability level. Increasing the number of ways increases the reliability, but the cache capacity becomes smaller. This technique can correct 1-4 errors for each 64-bit segment.

2.6 Design Implications of Existing Methods

Using all these state of the art approaches for handling errors in the nanometer regime makes it possible to meet the design constraints, but these solutions come with disadvantages. The redundant hardware (i.e. redundant rows/columns in memory) increases the die area and limits performance.

But the most important aspect is that these techniques lead to significant power consumption overhead, which is another main design challenge in nanometer nodes. For example 8T and 10T cells, and also ECC can increase the power by more than 50% [39], [40], [42], [54], [58]. Techniques that achieve the reduction of power consumption are: multiple voltage and frequency islands [37] and supply voltage scaling [59]. Unfortunately, the best technique that is voltage scaling leads to more variations. In general, process variation resilience and reduction of power consumption have contradictory requirements so finding the best tradeoff is becoming very difficult and new methods with lower overhead are required.

Chapter 3
Proposed Approach

Memories are very important, because every digital system integrates at least one or even several different kinds of memory components. The minimum supply voltage for a processor as a whole is typically determined by the failures in the memory cells [42]. While researchers have explored different memory resilience techniques, the resilience of the data representation on a system level has not been explored. It might be possible that different data representations are more reliable under hardware defects given an input distribution. To this end, our goal is to find the data representation that minimizes non-persistent errors in the memory. More specifically, to reach our goal we design a system-level fault simulator. The performance is evaluated based on the Mean Squared Error of the data written and read from the memory. For small bit sizes it is possible to perform an exhaustive search to find the best data representation. Interestingly, results showed that there are a lot of representations that have a better Mean Squared Error than the representations that are used nowadays to store the data in the memory. Furthermore, we develop an algorithm that performs the task of finding the best representation. By analyzing the algorithm using various input data distributions, we find out that the algorithm can find one of the best representations using significantly less computational effort compared to an exhaustive search of the solution space.

3.1 System Level Fault-Simulation Methodology

We develop a system level fault-simulation approach that is depicted in Figure 2. It consists of four stages: symbols and input distribution, memory model, metrics, and conventional mappings. The next sections explain analytically the stages.

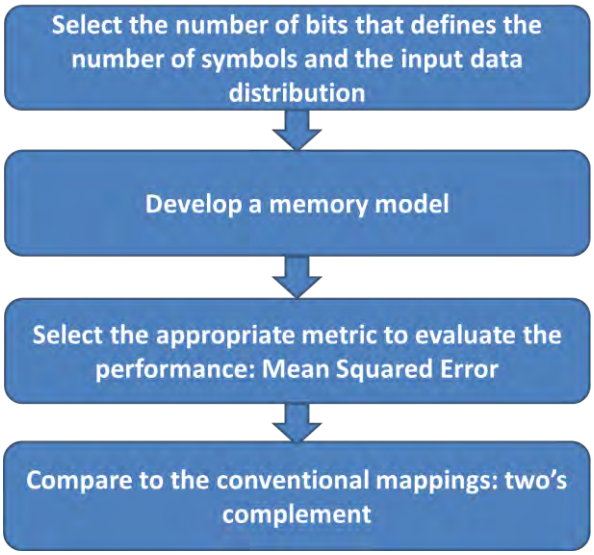


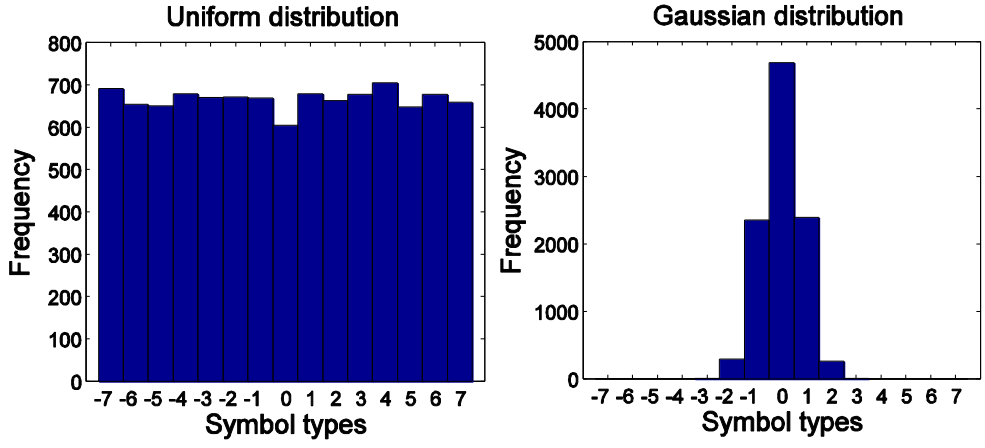
Figure 2: System level fault-simulation approach

3.2 Symbols and Input Data Distribution

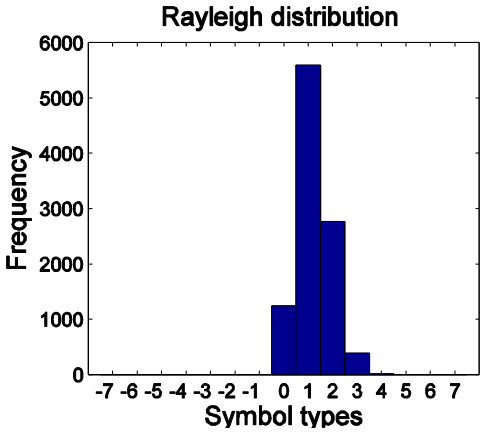
The number of different symbols that will be written and read from the faulty memory depends on how many bits are used to represent a data word. Let B be the total number of bits that are used to represent the data, then there will be $N = 2^B - 1$ different symbols that range from the signed integer values $-\lfloor N/2 \rfloor$ to $\lfloor N/2 \rfloor$. For instance, if each symbol is represented with 3 bits, there will be $N = 2^3 - 1 = 7$ symbols from $-\lfloor N/2 \rfloor = -3$ to $\lfloor N/2 \rfloor = 3$. Using B bits, there are $N = 2^B$ possible different symbols that range from $-\lfloor N/2 \rfloor - 1$ to $\lfloor N/2 \rfloor$, but for symmetry reasons we use one symbol less. We do not use the symbol with the larger absolute value. Hence there are an equal number of negative and positive symbols. We showed that when $B = 3$ and $N = 7$ our symbols are $\{-3, -2, -1, 0, 1, 2, 3\}$, three of them are positive and three of them are negative. But all possible symbols are $\{-4\} \cup \{-3, -2, -1, 0, 1, 2, 3\}$.

So far, we assumed that the symbols stored in the memory represent only integers. The symbols that represent fractional numbers are converted into integers by being multiplied by the appropriate power of two before being stored into the memory. The inverse procedure is applied upon reading the symbols. B still remains the total number of bits, but now is the sum of the number of bits used for the decimal part (B_{int}) , and the number of bits used for the fractional part (B_f) , $B = B_{int} + B_f$.

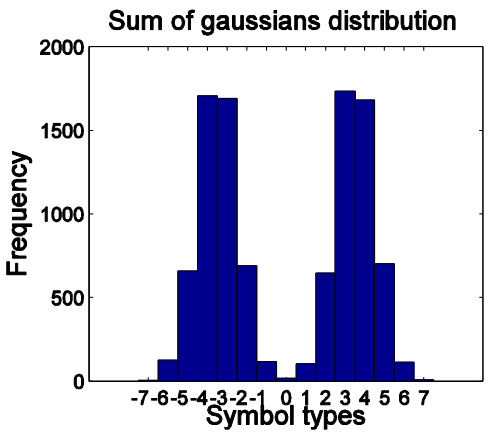
The distribution of the symbol types stored in the memory depends on the system that the memory will be included in. We utilize Gaussian distributions with different values for its parameters in order to emulate four other different input distributions: Uniform, sum of Gaussians, Rayleigh and Laplace as shown in Figure 3. The reason for choosing five different distributions is to be able to model different kinds of input data accurately. We chose these specific five distributions, because they are commonly encountered in practice and used in different areas i.e. statistics, multimedia and digital signal processing. For instance it is common that the data that is saved in the memory of a decoder, which is used in wireless communication channels (i.e. Viterbi decoder), follows a sum of Gaussians distribution. The Laplacian distribution is for example used in speech recognition to model priors on Discrete Fourier Transform coefficients. The Gaussian distribution is one of the well-known probability distributions in statistics and is encountered very often in practice and also in other sciences such as natural and social sciences. The central limit theorem uses the Gaussian distribution to model many practical problems. Emulating five different common distributions and changing their parameters, allows fitting other unknown data distributions to one of the five aforementioned distributions with a small approximation error.



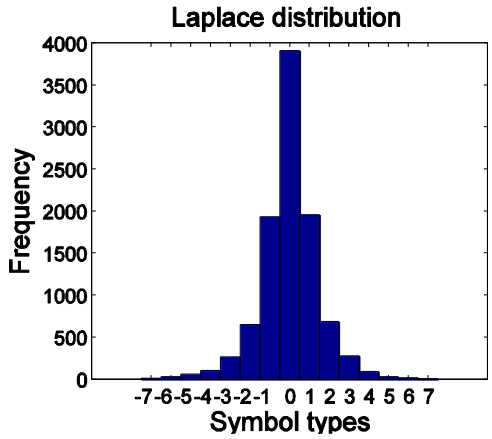
a) b)



c)



d)



e)

Figure 3: Symbol type distributions for $B = 4$
(a) Uniform
(b) Gaussian (variance $= 0.64$, mean $= 0$)
(c) Sum of Gaussians (variance $= 1$, mean $= 0$)
(d) Rayleigh
(e) Laplace ($\mu = 0, b = 0$)

3.3 Memory Model

Memory system models can be described on various levels of abstraction, starting from the very low transistor level, up to higher level descriptions, such as application programming interfaces. The selection of the memory system model is based on the main goal that will be explored. Our goal is to evaluate the impact of errors at the system level by exploring various input distributions. Since circuit level simulations are very costly in terms of time and processing power, we model the circuit failures as bit flips and the memory as a non-binary symmetric channel.

A binary symmetric channel (BSC), as shown in Figure 4, includes a transmitter which sends two different types of symbols (zero or one) and a receiver that receives the symbols through a communication channel. During transmission a symbol is 'flipped' with a crossover probability of p. A non-binary symmetric channel is a BSC which is capable of transmitting more than 2 different types of symbols.

We model the faulty memory as a non-binary symmetric channel. The communication channel is the faulty memory. The transmitted symbol is the symbol that is going to be written in the memory and the received symbol is the symbol that is going to be read from the same memory cell that the previous symbol was written to.

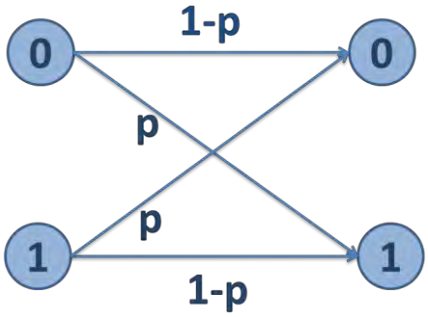


Figure 4: Binary symmetric channel (BSC)

Our goal is to minimize the mean difference between the value of the symbol written into the memory and the value of the symbol that is read back from the memory. We add 2 new stages to the memory model, one before and one after the memory as it is shown in Figure 5.

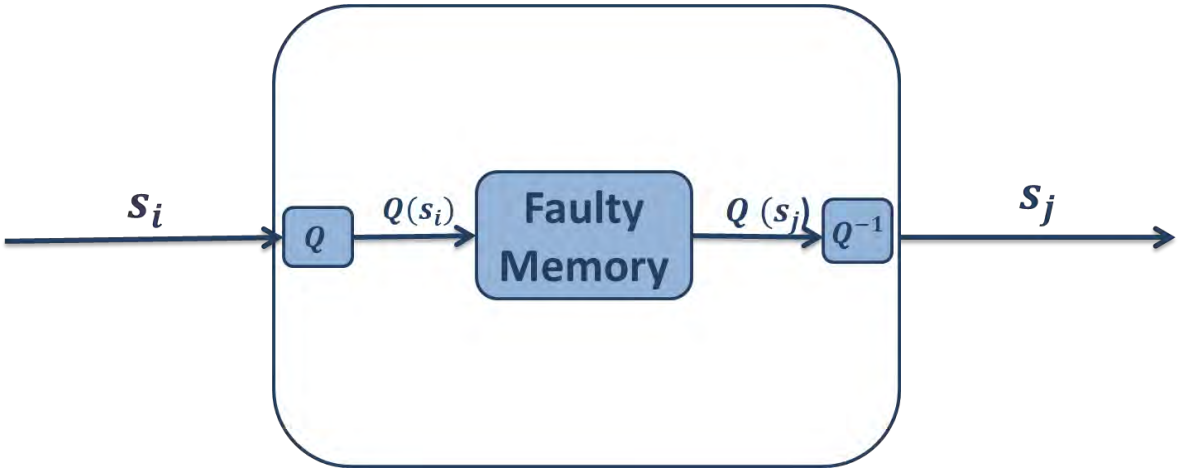


Figure 5: System model of memory with mapping and de-mapping stages

The first stage Q, which is the mapping, maps the input symbol s_i according to a look up table to another symbol $Q(s_i)$ that is then stored in the memory. The second new stage Q^{-1}, which is the de-mapping , de-maps the symbol $Q(s_j)$ that is read from the memory to s_j using the inverse look up table. A mapping is an ordered sequence of all the different types of symbols, each appearing only once. For instance for $B = 3$ the mapping $\{-1, -2, 0, 3, -3, 2, 1\}$ corresponds to the look up table shown in Table 1.

The number of possible mappings M depends exponentially on the number of symbols N:

$$M = N\,!$$

The number of symbols depends on the number of bits B (cf. Section 2.2), therefore the number of mappings depends on the number of bits:

$$M = (2^B - 1)!$$

s_i		$Q(s_i)$	
-3	101	-1	111
-2	110	-2	110
-1	111	0	000
0	000	3	011
1	001	-3	101
2	010	2	010
3	011	1	001

Table 1: Look up table for example mapping $\{-1, -2, 0, 3, -3, 2, 1\}$

The middle stage of the system in Figure 5 is the faulty memory which is modeled as a simple array. The memory size depends on the number of symbols that will be stored and on the number of bits used to represent each symbol. The injected errors are bit flips which are spread uniformly throughout the whole memory. The number of errors is defined by multiplying the size of the memory by the probability of a bit flip error P_e. The model is highly configurable and the aforementioned characteristics are parameterizable.

3.4 Metric: Mean Squared Error

Various metrics can be used to assess the memory performance depending on the memory model and on the design goal. The most relevant metrics to evaluate the impact of errors in high level memory models are the mean squared error (MSE) and the signal to noise ratio (SNR). We evaluate our performance in terms of MSE, since the MSE is always a basic component of the SNR. The MSE of an estimator quantifies the error between values implied by an estimator and the true values of the quantity being estimated. The MSE measures the average of the squares of the errors. In our case, the estimator is the symbol s_j that is read from the memory, and the true value is the symbol s_i written to the memor. The missing information for the estimator is the amount and the position of the errors in the memory.

To compute the MSE we start with the error cost C or the square error, which is computed for two symbols s_i and s_j.

$$C\ (s_i, s_j)\ =\ \left|s_i - s_j\right|^2$$

The next step is to find out the crossover probability of s_i to be transformed to s_j. Transformation of s_i to s_j corresponds to transformation of $Q(s_i)$ to $Q(s_j)$ in our memory model. Each bit has a probability of P_e to be flipped, consequently each bit has 1-P_e probability to not change. The number of bits that are flipped is equal to the hamming distance $h(Q(s_i), Q(s_j))$ between $Q(s_i)$ and $Q(s_j)$. The Hamming distance between two vectors of equal length is the number of positions at which the corresponding vector elements differ. In other words, it equals the number of errors that transform one vector into the other. In our case, the vectors are the binary representation of $Q(s_i)$ and $Q(s_j)$. The total number of bits is B, hence the probability of $Q(s_i)$ being transformed into $Q(s_j)$ is:

$$P(s_i \to s_j) = P(Q(s_i) \to Q(s_j)) = {P_e}^{h(Q(s_i),Q(s_j))} * (1 - P_e)^{B - h(Q(s_i),Q(s_j))}$$

The product of the probability $P(s_i \rightarrow s_j)$ and the cost $C\ (s_i, s_j)$ gives the square error for s_i being transformed to s_j. Summing up the products for every s_j, we get the mean squared error for s_i being transformed to all the possible symbols. The MSE of one mapping is calculated from the probability of each symbol $P(s_i)$ which depends on the input data distribution described in Section 3.2. Multiplying $P(s_i)$ by the previous sum and summing it again for all the possible symbols gives the MSE for one mapping m:

$$MSE_m = \sum_i P(s_i) \sum_j P(s_i \rightarrow s_j)\ C(s_i, s_j)$$

$$m = 1, 2, \dots, M$$

3.5 Conventional Mappings

Four well-known data mappings are: ones' complement, two's complement, sign magnitude and gray code. The most popular one and the one that is usually used to store the data in the memory in almost all of today's digital systems is the two's complement format. Hence, to evaluate our results we will be comparing the MSE of every mapping to the MSE of the two's complement mapping.

3.5.1 Ones' and Two's Complement

Calculating the ones' complement of a binary number is done by inverting each bit that is one to zero and each bit that is zero to one. The two's complement is computed by computing first the ones' complement and then adding one using the binary representation. Another very easy way is to start from the last significant bit and keep the bits the same till the first one appears. Leave the one untouched and swap all the next bits till the most significant one.

The two's complement data representation is the most common of storing signed integers into memory devices. In this arithmetic the positive numbers do not change. For the negative numbers, the two's complement of their absolute value is stored into the memory. An B-bit two's-complement system can represent every integer in the range -2^{B-1} to $2^{B-1}-1$ while ones' complement can only represent integers in the range $-(2^{B-1}-1)$ to $2^{B-1}-1$.

3.5.2 Sign Magnitude

The sign magnitude representation is so well-known because it is close to the natural way of showing that a number is negative or positive which is putting the sign in the beginning of the number. In this system the first bit of the binary representation is used to represent the sign and the other ones to represent the amplitude or the absolute value. One of the disadvantages is that the zero is represented in two different ways 000000 (0) and 100000 (-0).

3.5.3 Gray Mapping

Gray code is a numeric format where the successive binary representation of the numbers differs in only one bit. The gray code for n-bits can be generated recursively from $\mathrm{n}-1$ bits. First writing down the $\mathrm{n}-1$ bits binary representation list, then reflecting the list and concatenating the first list with the second one and in the end putting as prefixes to the entries in the first list a zero and in the second one a one. For example, Table 2 shows generation of the $n = 3$ list from the $n = 2$ list.

2-bit list:	00, 01, 11, 10	
Reflected:		10, 11, 01, 00
Prefix old entries with 0:	000, 001, 011, 010,	
Prefix new entries with 1:		110, 111, 101, 100
Concatenated:	000, 001, 011, 010,	110, 111, 101, 100

Table 2: Generating 3-bit Gray code from 2-bit

The grey mapping can be applied just for positive numbers. In our case we have also negative numbers. Therefore, we shift the symbols in order that the smallest negative symbol corresponds to zero. Hence, all the numbers will be positive and afterwards the gray encoding can be applied.

Chapter 4
Exploration of Mappings

In this section we describe the methods for exploring the solution space and finding the mappings that can reduce the MSE under various hardware defects compared to the two's complement mapping. To verify our results we run 1000 Monte Carlo (MC) trials for each mapping, on a memory that contains 10000 random symbols, which are sampled from the same assumed input data distribution. The evaluation of the mappings using a MC based simulation flow is depicted in Figure 6. Initially some mappings are randomly selected or all of them are selected in case that it is computationally feasible. Afterwards, based on the input data distribution 10000 symbols are generated and stored in the memory according to the specific mapping. Then depending on the bit flip error probability P_e (explained in Section 3.4), an array instance which contains uniformly distributed fault locations throughout the whole memory is created. Furthermore, the MSE is evaluated for all the selected mappings. The last stage compares the MSE of each selected mapping with the MSE of the two's complement and returns the mappings with a better MSE than the two's complement.

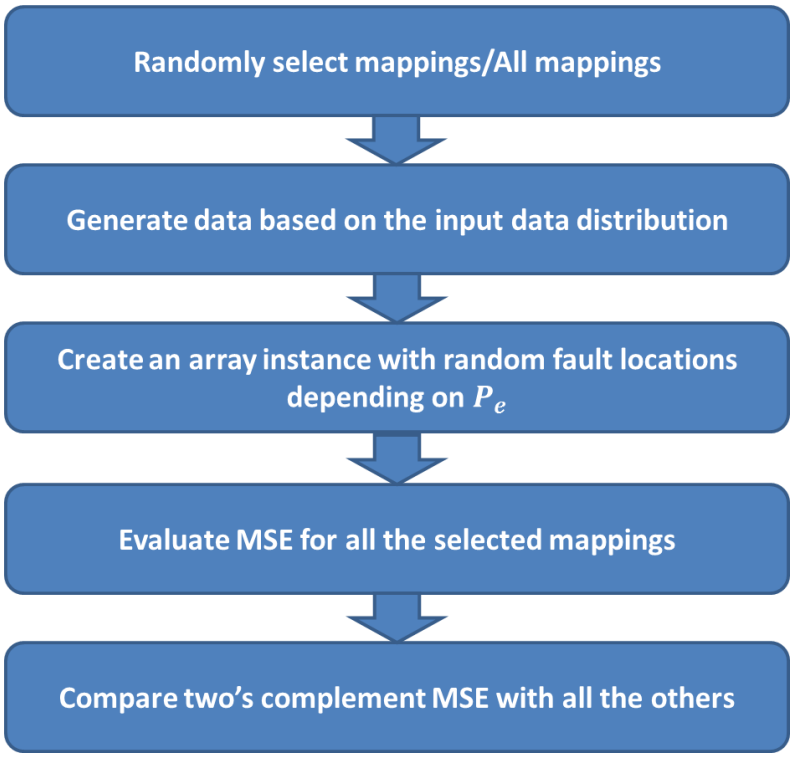


Figure 6: Evaluation of mappings using a Monte Carlo simulation flow

4.1 Exhaustive Search

The most straight forward method is to explore the complete solution space exhaustively. We run an exhaustive search for $B = 2$ and $B = 3$ bits, evaluating the MSE as shown in Section 2.4. The results show that there are a lot of other mappings with smaller MSE than the two's complement,

sign magnitude and gray code. For instance, in the case of $B = 3$ bits with input data generated by a Gaussian distribution with a variance of one, zero mean and $P_e = 0.1$, we evaluate all the 5040 possible mappings. As can be seen in Figure 7, there are more than 1000 mappings that perform better than the two's complement representation. Specifically, we find that a mapping that assigns $\{-3, -2, -1, 0, 1, 2, 3\}$ to the two's complement representation of $\{-1, -3, -3, 1, 0, 2, -2\}$ provides 45.3 % less MSE compared to the two's complement. Interestingly, sign mapping and gray encoding give 33.7% and 27.4% less MSE accordingly. In general, the mappings that are better than the two's complement achieve on average 20% less MSE, while the best of them achieve 50% less MSE.

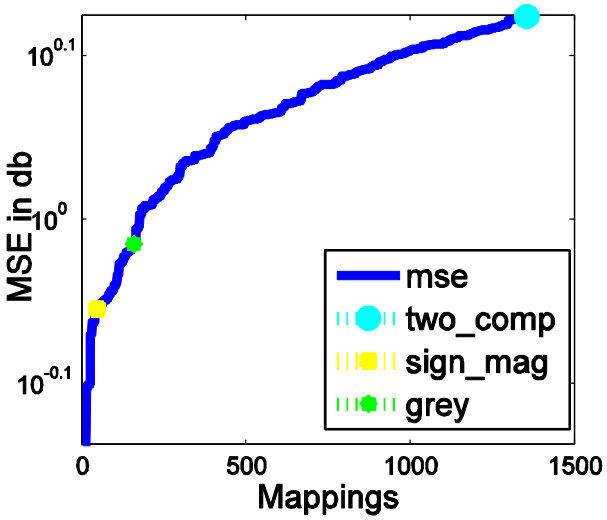


Figure 7: The mappings with a lower MSE than the two's complement (3 bits, Gaussian distribution, variance 1, mean 0)

To verify our results we run MC trials on a 30 kb memory as shown in Figure 8. In general the results of the exhaustive search and MC simulations are very close for various bit flip probabilities and number of bits. Table 3 presents the difference between the two methods for nine random mappings and the two's complement mapping for the instance of a Gaussian distribution with $B = 3$, variance one, mean zero and $P_e = 0.1$. The average deviation of the two methods for the specific example is 0.03976. For the same example and the same mappings shown in Table 3, Figure 8 depicts a comparison of the MSE in dB between the analytical method and MC trials results. We can graphically see that the two methods converge to similar results.

Mappings	Analytical	MC trials	Difference
1	0.6949	0.6931	0.0018
2	0.7292	0.7362	0.0070
3	0.7292	0.7364	0.0072
4	0.7945	0.7495	0.0450
5	0.8829	0.8393	0.0436
6	0.8965	0.9716	0.0751
7	0.9063	0.8511	0.0552
8	0.9063	0.8521	0.0542
9	0.9267	0.9504	0.0237
10	1.3319	1.2471	0.0848

Table 3: Comparison between the MSE calculated by the exhaustive search and the MC trials simulations

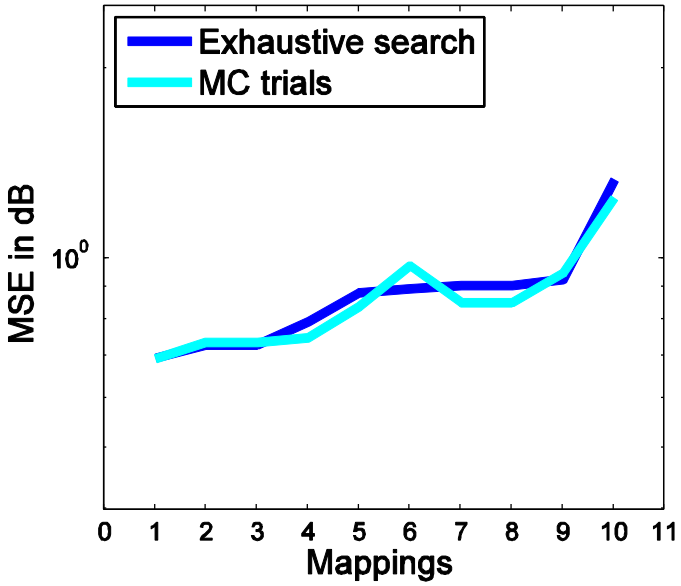


Figure 8: Comparison between the MSE in dB calculated by Analytical method and MC trials simulations

Unfortunately as the number of bits increases, the number of possible mappings increases exponentially, as can be seen in Figure 9. After $B = 7$ bits, the number of mappings in dB exceeds a level of 10^{213}. Therefore the exhaustive search of the results space is computationally impossible for a large number of bits. Since checking all the mappings is not feasible, the question that arises is if checking only some of the mappings makes it still possible to find better representations than the two's complement.

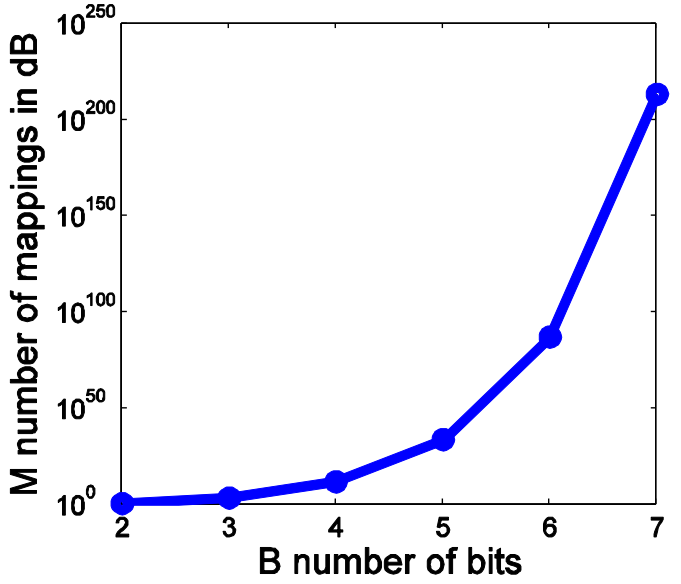


Figure 9: The number of mappings M as a function of the number of bits B

4.2 Random Mappings – Analytical Method

Since we have to choose some of the mappings to compare with the two's complement representation, the simplest way is random selection and evaluation of the results using an analytical method as introduced in Section 3. For a small number of bits until $B = 4$, the random selection of mappings results in finding some mappings that are better than the two's complement. For a larger number of bits, unfortunately choosing randomly representations does not find any mappings that have a smaller MSE than the two's complement. Therefore a better algorithm for choosing the mappings is needed. The proposed algorithm is explained in the next chapter.

Chapter 5
Algorithm for Choosing the Mappings

The analytical method that tries 5000 random mappings can find a few better mappings than the two's complement until $B = 4$ bits. For a larger number of bits this is not possible, because the ratio between the mappings with a lower MSE than that of the two's complement and all the possible mappings decreases significantly. Hence, we devise an algorithm that chooses the mappings in a systematic fashion and finds best mappings faster with a limited number of checked mappings.

5.1 Main Idea

The motivation for our algorithm stems from our observations during the exhaustive search simulations. Specifically, we observe that if we divide the best mappings into groups based on their first symbol, all the groups have a similar amount of mappings. For instance, in case of $B = 3$ bits there are 7 different symbols that can be written and read from the memory $\{-3, -2, -1, 0, 1, 2, 3\}$. Therefore there will be 7 different groups and each group will include all the mappings that start with a specific symbol, i.e. the group of -3 will contain mappings such as $\{-3, 2, 1, 0, -2, 3, -1\}, \{-3, 0, 1, 2, 3, -2, -1\}, \{-3, \dots, \dots, \dots, \dots, \dots, \dots\}$, the group of 0 will contain mappings such as $\{0, \dots, \dots, \dots, \dots, \dots, \dots\}$, etc. Figure 10 shows for the example of a Gaussian distribution with $B = 3$, variance one, mean zero and $P_e = 0.1$ that the number of different mappings in each group is similar. Therefore, if we make sure that the algorithm produces mappings that have as a first symbol all the possible different symbols, i.e. that the algorithm produces the same number of mappings for each group, we increase the probability of finding one or more good mappings, without having to exhaustively explore any of the groups.

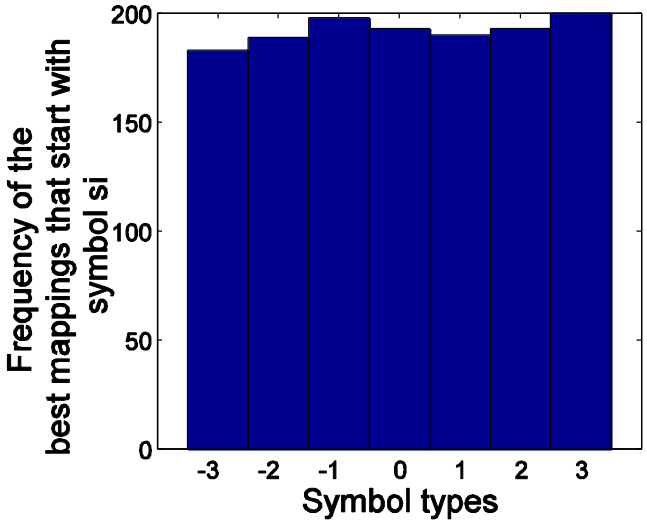


Figure 10: Best mappings groups based on their first symbol

5.2 Number of Generated Mappings

The algorithm generates a different number of mappings depending on the number of symbols N, hence depending on the number of bits B used to store the data in the memory. The total number of generated mappings M_a is the sum of the mappings in each of the groups that start with one different symbol as explained in Section 5.1. The number of mappings in each group is $N-1$, therefore the total number of mappings is:

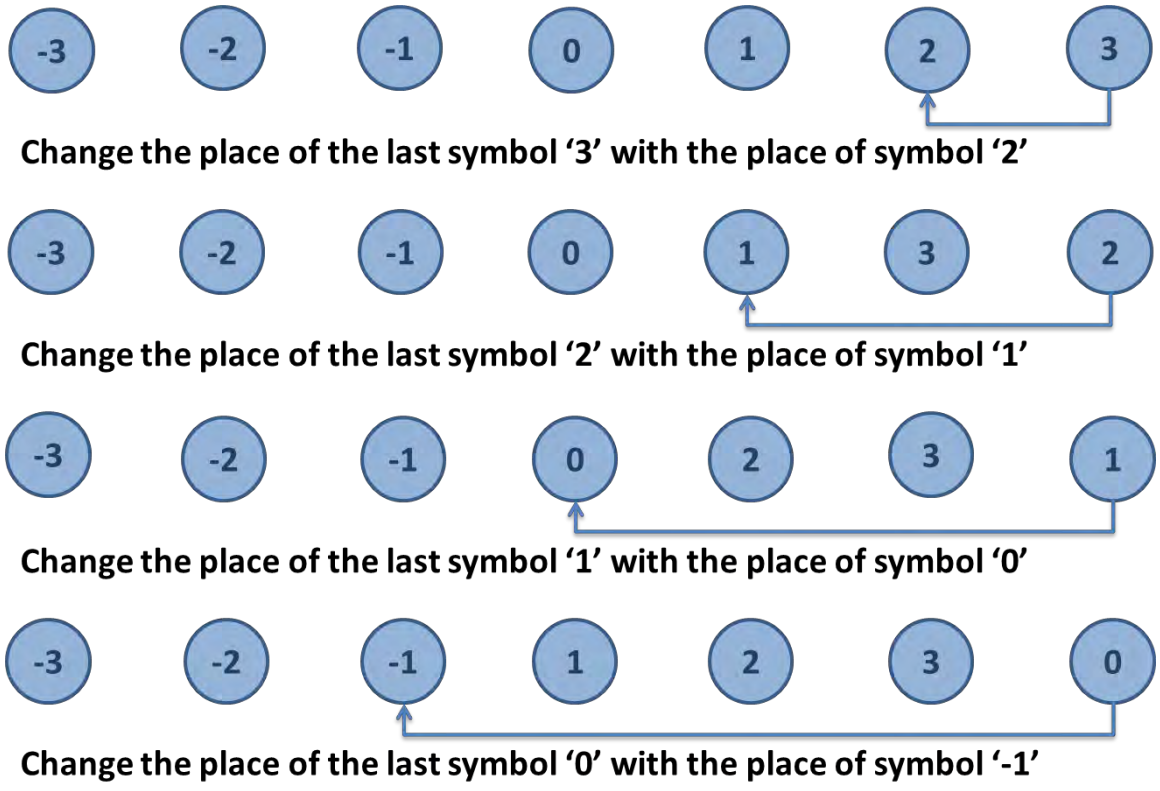
$$M\_a = N * (N-1)$$

For example for the $B = 3$ bit representation the number of mappings is: $B = 3 => N = 7 => M\_a = 7 * 6 = 42$

5.3 Generation of Mappings

Figure 11 shows an example of generating mappings for $B = 3$. In general, the mappings are generated each time by changing the place of the last symbol with all the other symbols of the mapping. The replacement of the symbols starts from the second last symbol and ends with the first symbol. When the first symbol is reached, the algorithm restarts from the symbol before the last. After every replacement, the new vector is used as initial vector for the next replacement. After M_a changes the algorithm stops because the next change returns the vector to the initial mapping (two's complement).

This method of symbol exchange allows for an equal number of generated mappings in each group. For instance, in the case of the $B = 3$ bit representation, all the possible symbols are -3, -2, -1, 0, 1, 2, 3, with each symbol generating 6 mappings that start with that symbol.



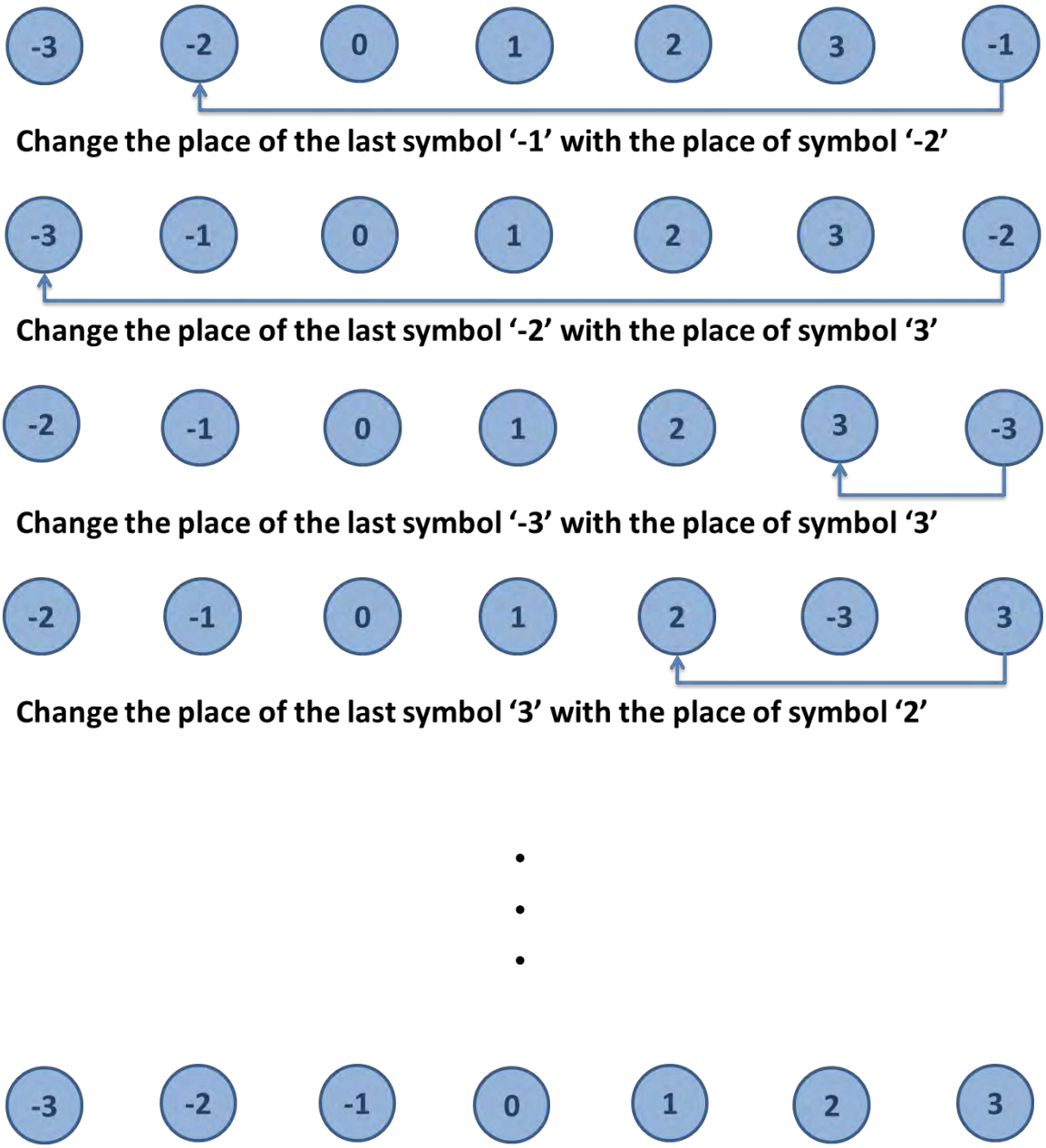


Figure 11: Visual explanation of how the algorithm generates the mappings

5.4 Pseudo-code

The algorithm pseudo-code is presented in Algorithm 1. *Mappings* is the array that stores all the possible mappings. The first entry of *Mappings* is initialized with the two's complement mapping as it is shown in Line 1. Line 2 initializes a counter to initially hold the second last symbol of the last generated mapping. In Line 3 the total number of mappings is defined and the mappings are counted until the algorithm generates the determined number of mappings. Since the first mapping is by default the two's complement one, the counting starts from the second mapping. Line 4 through 6 reset the counter that holds the symbol which is going to be exchanged if it has reached the first symbol of the mapping. In Line 7 through 9 the symbols are exchanged to form the next mapping. Line 10 updates the place of the symbol that is going to be exchanged with the last symbol, to form the new next mapping.

```
1-   Mappings(1,:) = two_complement;
2-   j = N-1;
3-   For i =2:1: M_a
4-                if (  j == 0 )
5-                        j = N-1;
6-                 end;
7-                 Mappings (i,:) = Mappings (i-1,:);
8-                 Mappings (i, N-1) = Mappings (i-1,j);
9-                 Mappings (i,j) = Mappings (i-1, N-1);
10-               j=j-1;
11-    end;
```

Algorithm 1: Mapping generation algorithm

5.5 Results: The Selected Mappings

For each selected mapping the algorithm calculates the mean squared error as explained in Section 2.4. The algorithm returns the mappings with a smaller mean squared error than the two's complement mapping. The algorithm results will be presented and compared in the next chapter.

Chapter 6
Results Comparison between the Algorithm, Analytical Method and MC Simulations

In the next sections we will be comparing the results of the algorithm with the results of the analytical method and MC simulations regarding their quality (MSE) and their computational complexity (algorithm runtime).

6.1 Reference Distribution

To explain the results, the Gaussian (or normal) distribution and its different parameters will be used as a reference. The Gaussian distribution is a continuous probability distribution known as the Gaussian function. Informally it is called the bell curve because it has a bell-shaped probability density function f:

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2}$$

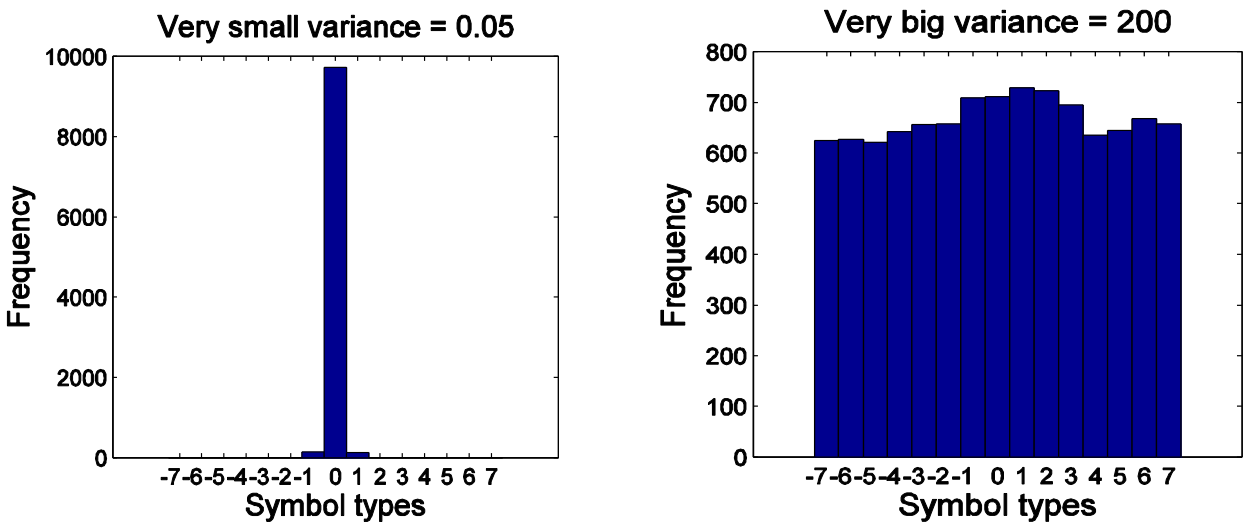
The parameter μ is the mean or expectation and σ^2 is the variance. σ is known as the standard deviation.

The normal distribution is one of the well-known probability distributions in statistics and is encountered very often in practice and also in other sciences such as natural and social sciences. The most important reasons for this are: The central limit theorem states that given certain conditions, the mean of a sufficient large number of variables from the same distribution with a finite mean and variance, will each be approximately normally distributed. Moreover, the normal distribution can be easily processed analytically, that means that a large number of results involving this distribution can be derived in explicit form.

We use different variances during the presentation of our results. Table 4 defines for every number of bits the exact value of small (0.3σ), medium (3σ) and big variance (27σ). All the following examples are illustrated utilizing one setup that uses a Gaussian distribution with zero mean. Changing the variance of the Gaussian distribution makes it possible to emulate also different other distributions as explained in Section 3.2. For example, since the number of bits that can be used to represent the data is not infinite, a Gaussian distribution with a very large variance tends to the uniform distribution as shown in Figure 12. Figure 12 also depicts a Gaussian distribution with a very small variance which tends to the Dirac delta distribution.

Gaussian	Small variance	Medium variance	Big variance
2 bits	0.01	0.1089	1
3 bits	0.1089	1	9
4 bits	0.64	5.29	49
5 bits	3.61	25	225

Table 4: Gaussian variances used for different bit length



a) b)

Figure 12: Gaussian distributions for $B = 4$ and mean zero (a) variance 0.05 (b) variance 200

We choose as number of bits $B = 3$, because for three bits we can explore the whole solution space using the MC trial simulations and the exhaustive search. Therefore it is possible to fully evaluate the algorithm performance in comparison with the two aforementioned methods and the analytical method. The number of performed MC simulations is 1000 on a 30kb memory. We assume that 10% of the bit cells have errors, that means specifically on average 3kb of the memory contains erroneous information. We choose 10% of the memory to contain wrong data, because we want to evaluate our algorithm in the worst case scenario [58]. In this scenario 10% of the memory suffers from failures induced by process variations or voltage scaling as discussed in Chapter 1. In our case, since we assume that all the cell failures are independent and the errors are uniformly distributed throughout the whole memory, the bit cell probability error P_e (cf. Section 3.4) is the same as the percentage of the memory that is erroneous.

6.2 Algorithm Runtime

As the number of bits B increases, the number of all possible mappings M grows exponentially. Our algorithm, as described in Chapter 5, checks a specific number of mappings M_a that grows polynomially. Figure 13 depicts both M and M_a from $B = 2$ to $B = 6$ by using a logarithmic scale for the number of mappings. As it can be seen, there is a very large difference between the number of checked mappings and all possible mappings. Hence, investigating only such a small number of

mappings makes the algorithm very fast compared to the exhaustive search. Table 5 presents more detailed numbers comparing all possible mappings and the number of checked mappings depending on the number of bits used to represent the data for $B = 2$ through $B = 7$.

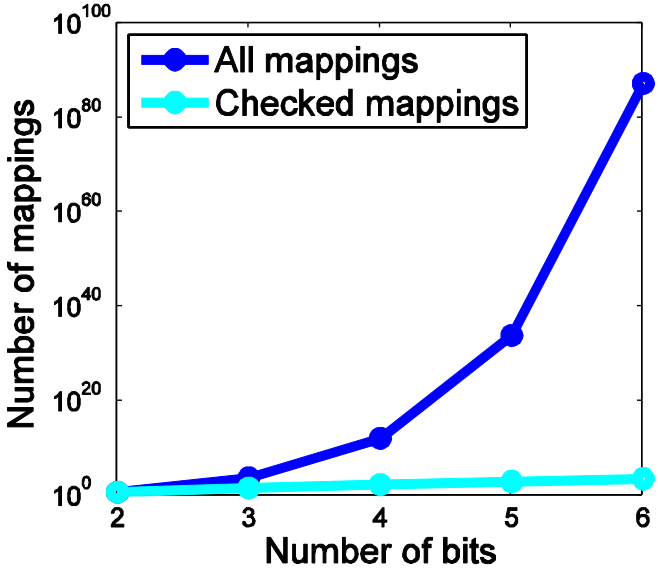


Figure 13 : Comparison of the number of all possible mappings and the number of the mappings that the algorithm checks

Bits B	Symbols N	Mappings M	Mappings M_a
2	3	6	6
3	7	5040	42
4	15	1.3077 e + 12	210
5	31	8.2228 e + 33	930
6	63	1.9826 e + 87	3906
7	127	3.0127 e + 213	16002

Table 5: Number of mappings generated by the exhaustive search and the algorithm

Our algorithm finds one of the mappings with smaller MSE than the two's complement in a very short time compared to not only the exhaustive search, but also compared to the analytical method which selects randomly mappings. Figure 14 compares results from the MC simulations, analytical method and the algorithm for medium (3σ) variance. The bars in the figure show percentages. The first bar shows how many mappings are checked from all the possible mappings. The second bar shows how many mappings are better than the two's complement of all the checked mappings. As can be seen in the figure, the algorithm checks a very small percentage of the mappings and around half of them are better than the two's complement. Specifically, our algorithm and the analytical method check 0.8% of all the possible mappings compared to the MC simulations that check 100% of the mappings. The algorithm finds that 48% of the checked mappings are better than the two's complement while the analytical method finds that approximately 35% of the investigated representations have a smaller MSE than the two's complement. MC simulations show that 28% of all the mappings are better than the two's complement.

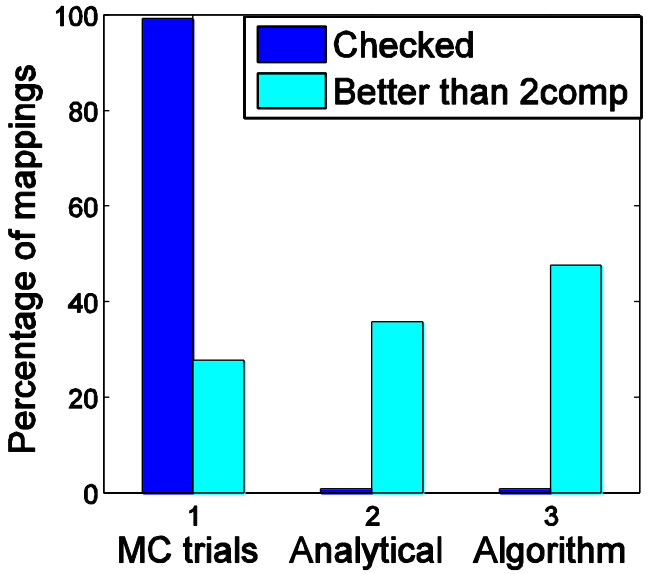


Figure 14: Percentages of checked mappings and mappings better than the two's complement

6.3 Dependence on the Distribution Parameters

The number of the mappings better than the two's complement that the algorithm finds, depends on the input data distribution. In case of a Gaussian distribution, the algorithm finds four times more mappings in case of a small variance compared to the case of a nine times larger variance as shown in Figure 15. This can be attributed to the fact that when the variance increases, the Gaussian distribution tends to approach the uniform distribution. For a uniform distribution, all possible values of the input data tend to appear with the same frequency in the memory. Therefore is not possible to find mappings which exploit the different frequencies of the symbols in the input data.

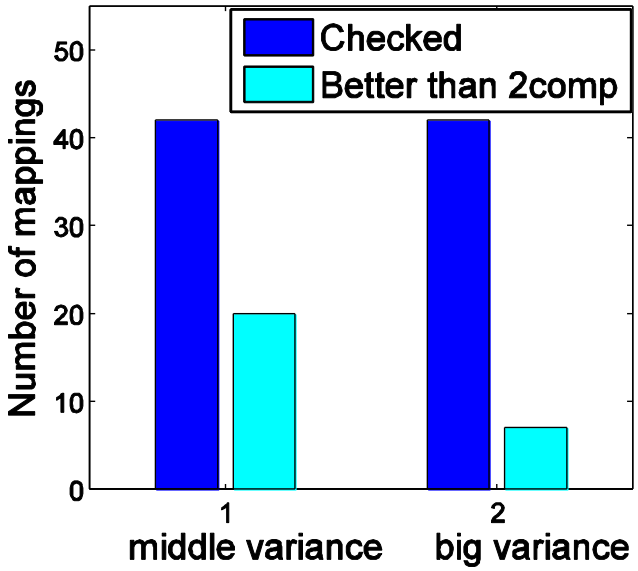


Figure 15: Number of mappings better than two's complement depending on input distribution parameters

In case that the data distribution tends to the uniform distribution, which happens for a variance greater than 9 (27σ) for $B = 3$, even if the algorithm cannot find a lot of mappings compared to how many it finds for the other distributions, it can find a few that have a good gain in dB, as depicted in Figure 16. The gain in dB is the absolute difference between the MSE of the best mapping and the two's complement mapping. Specifically, Figure 16 shows that in the case of a Gaussian distribution with a variance of one (0.3σ), the algorithm achieves a gain of 1.634 dB and for the same distribution with a variance of nine (27σ), it attains a gain of 1.3578 dB.

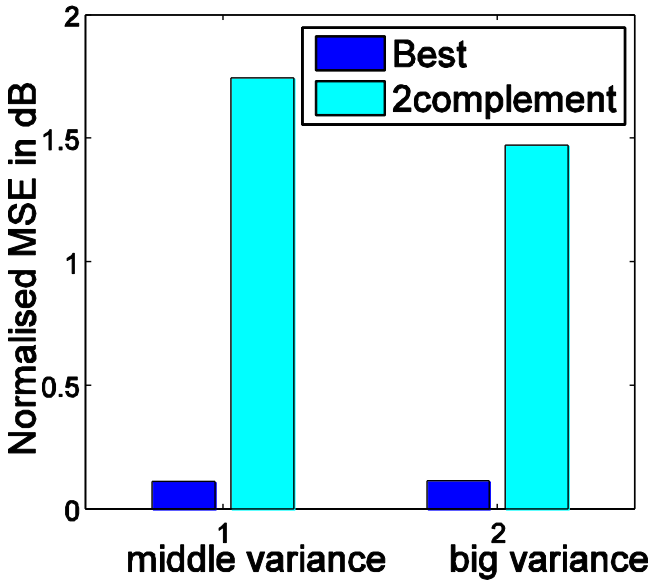


Figure 16: MSE of the best mapping depending on input distribution parameters

6.4 MSE Efficiency

The main goal of the algorithm is not to find as many mappings as possible better than the two's complement, but to find the one that has the largest reduction in terms of MSE compared to the two's complement representation. Figure 17 shows the relative gain in terms of MSE for the 20 best mappings ordered with increasing MSE. The computation is done with reference to the two's complement representation for the same setup described in Section 6.1 and 3σ variance. We note that 10 of the mappings provide more than 10% less MSE than the two's complement, 3 of the mappings more than 15%, 2 of the mappings more than 20% and the best mapping reduces the MSE by more than 30%. Specifically, if we choose the best mapping the MSE is decreased by 31.3 %.

A comparison between the algorithm results, the exhaustive search method and MC trial simulation for the same setup and parameters, shows that the algorithm finds a mapping with slightly less gain in terms of dB and percentage than the two other methods as shown in Figure 18. The gain in dB is the difference between the MSE of the two's complement mapping and the best mapping. The gain in percentage is the relative difference of the gain in dB. The exact values of both gains for the three methods are shown in Table 6. To summarize the results, the last row of Table 6 shows for each method the percentage of all the possible mappings that is checked. Interestingly, the algorithm finds a very good representation, while only checking a small percentage of all the mappings. For the specific case of $B = 3$, since the number of bits is very small, the exhaustive search is possible and the tradeoff that the algorithm performs between the performance in terms of MSE gain and the computational time may not be that obvious. For larger number of bits the tradeoff becomes significantly more apparent. For the instance of $B = 5$ and a variance of 3.61 using the previous setup, the gain in percentage is 10%, checking only a small fraction $1.1310\ \ 10^{-28}$ % of all

the possible mappings. For the instance of $B = 7$ and a variance of 441, the gain in percentage is X%, checking only $5.331 * 10^{-210}$ % of all the possible mappings. Table 7 shows a comparison of the speedup and the gain of MSE in percentage for the initial setup between the algorithm and the analytical method for different number of bits. For the case of 5 and 7 bits 0% means that the analytical method cannot find any mapping better than the two's complement. We note that the algorithm is 100 times faster than choosing random mappings, and as the number of bits increases the results of the algorithm have also a better quality than the analytical method. Note that the number of mappings that the analytical method checks is arbitrarily chosen as 100 times more than the number of mappings checked by the algorithm.

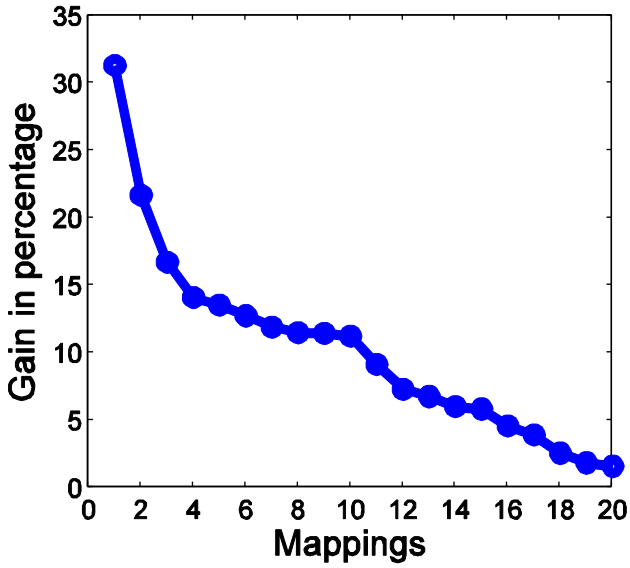


Figure 17: Gain of MSE in percentage for the mappings better than the two's complement

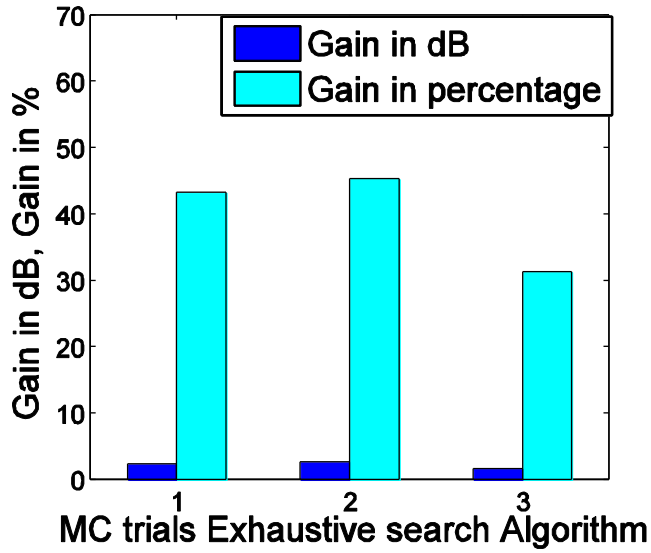


Figure 18: Gain of MSE in dB and percentage of the best mapping compared to the two's complement mapping

	MC trials	Exhaustive search	Algorithm
Gain in dB	2.289	2.6163	1.63
Gain in %	43.3	45.3	31.3
% of checked mappings	100%	100%	0.8%

Table 6: Gain of MSE in dB and percentage for the three methods

Number of bits	Analytical	Algorithm
3: Speedup MSE Gain in %	1.0 45.3 %	100 31.3 %
5: Speedup MSE Gain in %	1.0 0 %	100 10 %
7: Speedup MSE Gain in %	1.0 0 %	100 10 %

Table 7: Speedup and gain of MSE in percentage

6.5 Impact on Quality

In order to realize the significance of our approach, let us consider briefly a wireless system scenario and evaluate the potential impact on the quality of such a case study. A wireless system in a high level description includes different blocks as shown in Figure 19. The first block is the transmitter which transmits the signal over a channel. The receiver receives the distorted signal which includes the original signal and noise. A decoder is used to decode the received distorted signal in order to reconstruct the original signal. The information needed to decode the data is called Likelihood ratio (LLR), which is received and then stored in the memory. The distribution of the LLR input data usually follows a sum of Gaussians distribution and thus by modeling such data through a sum of Gaussians distribution we can evaluate the potential impact of our approach in such an application [60].

Figure 19: Wireless communication system

For instance, we could replace the memory system showed in Figure 11 with our memory model (cf. Section 3.3) and assess the Signal to Noise Ratio (SNR) in dB of the signal for the two's complement mapping and for mappings that our algorithm chooses. In this case the SNR is calculated as follows:

$$\boldsymbol{SNR = 10 * \log(Input^2/(Input - Output)^2)}$$

The input is the LLR data before being stored in the memory and the Output is the LLR data after being read from the memory.

For our case study we encode the LLR data as a sum of Gaussians distribution using three bits and we assume 10% random bit-flip errors in the memory of size 30 kb. Interestingly the results show that there is a significant improvement in terms of percentage gain of SNR, as shown in Figure 20. There are eight mappings that increase the SNR more than 20% compared to the SNR when the two's complement is used. Five mappings improve the SNR by more than 40%, two mappings by more than 60% and the best mapping by more than 80%. Therefore it is evident that our approach could improve the quality of the processed LLR data and potentially the decoding capability of the overall system.

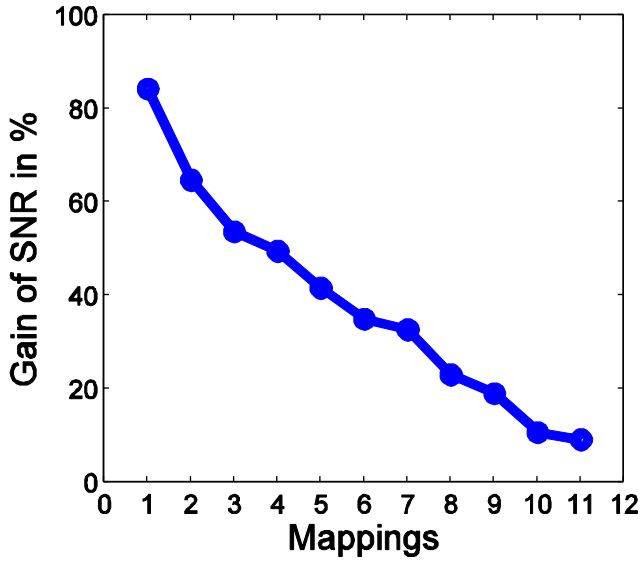


Figure 20: SNR gain in percentage

6.6 Power and Yield Improvement

Our algorithm achieves a better quality (quantified above in terms of MSE and SNR), which means that the output has a more satisfactory quality even if the memory has up to 10% errors. Since all the memory chips that contain up to 10% faulty cells or equivalently have a cell failure probability of $P_e = 0.1$ can now be used (cf. Section 6.1), the yield is significantly increased. For a model with a reliability of 100%, i.e. where dies are accepted only if they are error free, the yield for a memory array that contains K cells is:

$$Y = (1 - P_e)^K$$

In our case we accept memory chips that contain up to a specific number of errors F, hence the yield is redefined [37]:

$$Y(F) = \sum_{i=0}^{F} \binom{K}{i} x\, (P_e)^i\, (1 - P_e)^{K-i}$$

The yield can be rewritten as:

$$Y(F) = (1 - P_e)^K + \sum_{i=1}^{F} \binom{K}{i} x\, (P_e)^i\, (1 - P_e)^{K-i}$$

If the first part of the expression is replaced with the yield Y then $Y(F)$ corresponds to:

$$Y(F) = Y + \sum_{i=1}^{F} \binom{K}{i} x\ (P_e)^i\ (1 - P_e)^{K-i}$$

The above equation shows the yield improvement that can be achieved by not discarding chips with a number of defective cells smaller than F.

The improvement in quality achieved by the algorithm translates also into improvement of the yield of the memories, which is important since memories play a very significant role for the scalability of the supply voltage, for example failures in cache memory cells can determine the minimum supply voltage for a processor as a whole [32]. Although voltage scaling makes memories prone to errors as discussed in Chapter 1, using the proposed error resilient data representation for a given application enables aggressive voltage scaling, since more errors can be tolerated. Therefore the mappings found by the proposed algorithm offer the potential for power savings, since memory can be operated at a lower supply voltage. All the aforementioned improvements come at a low cost, since we do not use the expensive mechanisms of the techniques presented in Chapter 2. Specifically, our overhead is only two look up tables used to map and de-map the data going in and out of the memory.

Chapter 7
Conclusion

To minimize the impact of hardware failures on memory systems caused by process and power variations, an error resilient memory model is developed. The evaluation of the model performance is performed by using the Mean Squared Error of the data written and read from the memory. The simulations performed showed that for small bit sizes (e.g. 3 bits), the exhaustive search of the best data representation, which minimizes the Mean Squared Error, is possible. Since the computational time increases exponentially, the exhaustive search becomes infeasible for larger bit sizes. Only analyzing a small number of random data representations or mappings does not ensure that a representation with a lower Mean Squared Error than the two's complement will be found. For this purpose, an algorithm that systematically checks a small number of the mappings is developed. The algorithm performance is assessed using as main distribution the Gaussian, and emulating the Uniform, Sum of Gaussians, Rayleigh and Laplace distributions by the Gaussian distribution with changing mean and variance. In all the cases, the algorithm finds several good mappings using significantly less computational effort and time than the exhaustive search or the random search. The results show that the number of mappings better than two's complement and the difference between the best mapping found and the two's complement mapping in terms of Mean Squared Error depend considerably on the input data distribution and its parameters. The mappings that the algorithm generates improve not only the output quality and the yield of the memory by increasing its error resilience, but also permit the memory to operate under a very low voltage in order to reduce the power consumption.

Furthermore, the algorithm is implemented in a case study. Our error resilient memory system is included in a wireless communication channel system model that contains a decoder. The output data of the decoder has a lower bit error rate, when the data is stored in the memory using one of the mappings that our algorithm found, compared to when it is stored using the two's complement representation. The best result achieved is 80% more Signal to Noise Ratio (SNR).

Chapter 8
Future considerations

Future work should mainly focus on constructing the mappings number system, i.e. how to convert algorithmically a number expressed by the decimal system to its binary representation according to a specific mapping. This means indexing the bits and putting the right weight to each bit according to its position. The weight of each bit should not necessarily be a power of two. In this way the conversion from binary to decimal and the other way around would be possible without the use of a look-up table, through a mathematical formula.

Another interesting direction of future work is evaluation of the algorithm using advanced input data distributions and several case studies of other models that include a memory system.

References

[1] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," *IEEE Design Autom. Conf.*, pp. 338–342, Jun. 2003.

[2] K. A. Bowman, S. G. Duvall, and J. D. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE J. Solid-State Circuits*, vol. 37, no. 2, pp. 183–190, Feb. 2002.

[3] A. Shrivastava, D. Sylvester, and D. Blaauw, "Statistical Analysis and Optimization for VLSI: Timing and Power," *New York: Springer*, 2005.

[4] A. Asenov, A. R. Brown, J. H. Davies, S. Kaya, and G. Slavcheva, "Simulation of intrinsic parameter fluctuations in decananometer and nanometer-scale MOSFETs," *IEEE Trans. Electron Devices*, vol. 50, no. 9, pp. 1837–1852, Sep. 2003.

[5] M. Hane, Y. Kawakami, H. Nakamura, T. Yamada, K. Kumagai, and Y. Watanabe, "A new comprehensive SRAM soft error simulation based on 3D device simulation incorporating neutron nuclear reactions," *Simul. Semiconductor Processes Devices*, pp. 239–242, 2003.

[6] S. R. Nassif, "Modeling and analysis of manufacturing variations," *Custom Integrated Circuit Conf.*, pp. 223–228, 2001.

[7] C. Visweswariah, "Death, taxes and failing chips," *Design Autom. Conf.*, pp. 343–347, 2003.

[8] A. Bhavnagarwala, X. Tang, and J. D. Meindl, "The impact of intrinsic device fluctuations on CMOS SRAM cell stability," *IEEE J. Solid State Circuits*, vol. 36, no. 4, pp. 658–665, Apr. 2001.

[9] X. Tang, V. De, and J. D. Meindl, "Intrinsic MOSFET parameter fluctuations due to random dopant placement," *Trans. Very Large Scale Integr. (VLSI) Syst*., vol. 5, no. 4, pp. 369–376, Dec. 1997.

[10] B. E. Deal, M. Sklar, A. S. Grove, and E. H. Snow, "Characteristics of the surface-state charge (Qss) of thermally oxidized silicon," *J. Electrochem. Soc*., vol. 114, pp. 266–273, 1967.

[11] E. H. Nicollian and J. R. Brews, "MOS Physics and Technology," *New York: Wiley-Interscience*, 1982.

[12] C. E. Blat, E. H. Nicollian, and E. H. Poindexter, "Mechanism of negative bias temperature instability," *J. Appl. Phys*., vol. 69, pp. 1712–1720, 1991.

[13] M. F. Li, G. Chen, C. Shen, X. P. Wang, H. Y. Yu, Y. Yeo, and D. L. Kwong, "Dynamic bias temperature instability in ultrathin SiO2 and HfO2 metal-oxide semiconductor field effect transistors and its impact on device lifetime," *Jpn. J. Appl. Phys*., vol. 43, pp. 7807–7814, Nov. 2004.

[14] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," *Design Autom. Conf*., pp. 370–375, 2007.

[15] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An analytical model for negative bias temperature instability," *Int. Conf. Comput.-Aided Design*, pp. 493–496, 2006.

[16] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, “Impact of NBTI on SRAM read stability and design for reliability,” *Int. Symp. Quality Electron. Design*, pp. 210–218, 2006.

[17] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, “Adaptive techniques for overcoming performance degradation due to aging in digital circuits,” *Asia-South Pacific Design Autom. Conf.*, pp. 284–289, 2009.

[18] E. Karl, P. Singh, D. Blaauw, and D. Sylvester, “Compact in-situ sensors for monitoring negative bias-temperature-instability effect and oxide degradation,” *Int. Solid-State Circuits Conf*., pp. 410–623, 2008.

[19] W. Wang, V. Reddy, A. T. Krishnan, S. Krishnan, and Y. Cao, “An integrated modeling paradigm of circuit reliability for 65 nm CMOS technology,” *Custom Integr. Circuits Conf*., pp. 511 514, 2007.

[20] W. Wang, Z. Wei, S. Yang, and Y. Cao, “An efficient method to identify critical gates under circuit aging,” *Int. Conf. Comput.-Aided Design*, pp. 735–740, 2007.

[21] K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy, “Efficient transistor-level sizing technique under temporal performance degradation due to NBTI,” *Int. Conf. Comput. Design*, pp. 216–221, 2006.

[22] B. C. Paul, K. Kang, H. Kuflouglu, M. A. Alam, and K. Roy, “Temporal performance degradation under NBTI: Estimation and design for improved reliability of nanoscale circuits,” *Design Autom. Test Eur*., pp. 780–785, 2006.

[23] B. C. Paul, K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy, “Impact of NBTI on the temporal performance degradation of digital circuits,” *IEEE Electron Device Lett*., vol. 26, no. 8, pp. 560–562, Aug. 2005.

[24] K. Kang, M. A. Alam, and K. Roy, “Characterization of NBTI induced temporal performance degradation in nano-scale SRAM array using IDDQ,” *Int. Test Conf*., pp. 1–10, 2007.

[25] K. Kang, S. P. Park, K. Roy, and M. A. Alam, “Estimation of statistical variation in temporal NBTI degradation and its impact in lifetime circuit performance,” *Int. Conf. Comput.-Aided Design*, pp. 730–734, 2007.

[26] K. Kang, S. Gangwal, S. P. Park, and K. Roy, “NBTI induced performance degradation in logic and memory circuits: How effectively can we approach a reliability solution?,’’ *Asia South Pacific Design Autom. Conf.*, pp. 726–731, 2008.

[27] S. Jafar, Y. H. Kim, V. Narayanan, C. Cabral, V. Paruchuri, B. Doris, J. Stathis, A. Callegari, and M. Chudzik, “A comparative study of NBTI and PBTI (charge trapping) in SiO2/HfO2 stacks with FUSI, TiN, Re gates,” *Very Large Scale Integr. (VLSI) Circuits*, pp. 23–25, 2006.

[28] F. Crupi, C. Pace, G. Cocorullo, G. Groeseneken, M. Aoulaiche, and M. Houssa, “Positive bias temperature instability in nMOSFETs with ultra-thin Hf-silicate gate dielectrics,” *J. Microelectron. Eng.*, vol. 80, pp. 130–133, 2005.

[29] T. H. Ning, P. W. Cook, R. H. Dennard, C. M. Osburn, S. E. Schuster, and H. Yu, “1 _m MOSFET VLSI technology: Part IV-hot electron design constraints,” *Trans. Electron Devices*, vol. 26, no. 4, pp. 346–353, Apr. 1979.

[30] A. Abramo, C. Fiegna, and F. Venturi, “Hot carrier effects in short MOSFETs at low applied voltages,” *Int. Electron Device Meeting*, pp. 301–304, 1995.

[31] Y. Taur and T. H. Ning, “Fundamentals of Modern VLSI Devices,” *New York: Cambridge Univ. Press*, 1998.

[32] JEDEC Solid State Technology Association, ‘‘Failure mechanisms and models for semiconductor devices,’’ *JEP122-A*, 2002.

[33] M. T. Quddus, T. A. DeMassa, and J. J. Sanchez, “Unified model for Q(BD) prediction for thin gate oxide MOS devices with constant voltage and current stress,” *Microelectron. Eng*., vol. 51–52, pp. 357–372, May 2000.

[34] M. A. Alam, B. Weir, and A. Silverman, “A future of function or failure*,” IEEE Circuits Devices Mag.*, vol. 18, no. 2, pp. 42–48, Mar. 2002.

[35] D. Young and A. Christou, “Failure mechanism models for electromigration,” *IEEE Trans. Rel.*, vol. 43, no. 2, pp. 186–192, Jun. 1994.

[36] S. Garg and D. Marculescu, “System-level process variation driven throughput analysis for single and multiple voltage-frequency island designs,” *ACM Trans. Design Autom. Electron. Syst.*, vol. 13, Sep. 2008.

[37] G. Karakonstantis, C. Roth, C. Benkeser, A. Burg, “On the exploitation of the inherent error resilience of wireless systems under unreliable silicon” *IEEE Design Autom. Conf.*, pp. 510-515, June 2012.

[38] Z. Chishti, et al., “Improving Cache Lifetime Reliability at Ultralow Voltages”, *Micro*, pp. 89-99, Dec. 2009.

[39] K. Zhang, “Embedded Memories for Nanoscale VLSIs”, *Springer,* 2009.

[40] Shi-Ting Zhou, et al. “Minimizing Total Area of Low-Voltage SRAM Arrays through Joint Optimization of Cell Size, Redundancy, and ECC,” *IEEE ICCD*, 2010.

[41] S. E. Schuster, “Multiple word/bit line redundancy for semiconductor memories,” *IEEE J. Solid State Circuits*, vol. SC-13, pp. 698–703, Oct. 1978.

[42] C. Wilkerson, et al., “Trading off Cache Capacity for Reliability to Enable Low Voltage Operation,” *35th International Symposium on Computer Architecture (ISCA-35)*, pp. 203-214, June 2008.

[43] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, “Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*., vol. 24, no. 12, pp. 1859–1880, Dec. 2005.

[44] J. M. Rabaey, “Digital Integrated Circuits: A Design Perspective,” *Upper Saddle River, NJ: Prentice-Hall*, 1996.

[45] L. Chang, D. M. Fried, J. Hergenrother, J. W. Sleight, R. H. Dennard, R. K. Montoye, L. Sekaric, S. J. McNab, A. W. Topol, A. D. Adams, K. W. Guarini, and W. Haensch, “Stable SRAM cell design for the 32 nm node and beyond,” *Very Large Scale Integr. (VLSI) Technol*., pp. 128–129, 2005.

[46] B. H. Calhoun and A. Chandrakasan, “A 256-kb 65-nm sub-threshold SRAM design for ultra-low voltage operation,” *IEEE J. Solid State Circuits*, vol. 42, no. 3, pp. 680–688, Mar. 2007.

[47] I. Chang, J. J. Kim, S. P. Park, and K. Roy, “A 32 kb 10 T Sub-threshold SRAM array with bit interleaving and differential read scheme in 90 nm CMOS,” *IEEE J. Solid State Circuits*, vol. 44, no. 2, pp. 650–658, Feb. 2008.

[48] J. P. Kulkarni, K. Kim, S. Park, and K. Roy, “Process variation tolerant SRAM design for ultra low voltage application,” *Design Autom. Conf*., pp. 108–113, 2008.

[49] J. P. Kulkarni, K. Kim, and K. Roy, “A 160 mV robust Schmitt trigger based subthreshold SRAM,” *IEEE J. Solid State Circuits*, vol. 42, no. 10, pp. 2303–2313, Oct. 2007.

[50] T. Kim, J. Liu, J. Keane, and C. H. Kim, “A 0.2 V, 480 kb subthreshold SRAM with 1 k cells per bitline for ultra-low-voltage computing,” *IEEE J. Solid State Circuits*, vol. 43, no. 2, pp. 518–529, Feb. 2008.

[51] A. Agarwal, B. C. Paul, H. Mahmoodi, A. Datta, and K. Roy, “A process-tolerant cache architecture for improved yield in nano-scale technologies,” *Trans. Very Large Scale Integr. (VLSI) Syst.*, pp. 27–38, 2005.

[52] D. A. Patterson, P. Garrison, M. Hill, D. Lioupis, C. Nyberg, T. Sippel, and K. Van Dyke, “Architecture of a VLSI instruction cache for a RISC,” *Int. Symp. Comput. Architecture*, pp. 108–116, 2003.

[53] D. C. Bossen, J. M. Tendler, and K. Reick, “Power4 system design for high reliability,” *IEEE Micro*, vol. 22, no. 2, pp. 16–24, Mar./Apr. 2002.

[54] Y. Emre, et al., “Memory Error Compensation Techniques for JPEG2000,” *IEEE SiPS*, 2010.

[55] F.Zhang, “LDPC Codes for Rank Modulation in Flash Memories,” IEEE ISIT , pp. 859-863, Jun. 2010.

[56] J. Kim, et. al, “Multi-bit Tolerant Caches Using Two Dimensional Error Coding”, *Micro-40*, pp 197-209, 2007.

[57] H. Y. Hsiao et al., “Orthogonal Latin Square Codes,” *IBM Journal of Research and Development*, Vol. 14, Number 4, pp. 390-394, July 1970.

[58] I. J. Chang, et al., “A Priority-Based 6T/8T Hybrid SRAM Architecture for Aggressive Voltage Scaling in Video Applications.,” *IEEE Trans. on CSVT*, 2011.

[59] K. Zhang, U. Bhattacharya, Z. Chen, F. Hamzaoglu, D. Murray, N. Vallepalli, Y. Wang, B. Zheng, and M. Bohr, “A 3 GHz 70 Mb SRAM in 65 nm CMOS technology with integrated column-based dynamic power suppl,” *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 146–151, Jan. 2006.

[60] K.Sayana, J.Zhuang and K.Stewart, „Short Term Link Performance Modeling for ML Receivers with Mutual Information per Bit Metrics,” *IEEE GLOBECOM,* pp.1-6, 2008.