SRAM reliability improvement using ECC and circuit techniques

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

in

Electrical and Computer Engineering

MARK P. MCCARTNEY

B.S.E., COMPUTER ENGINEERING, CASE WESTERN RESERVE UNIVERSITY

M.S., ELECTRICAL AND COMPUTER ENGINEERING, CARNEGIE MELLON UNIVERSITY

CARNEGIE MELLON UNIVERSITY

PITTSBURGH, PA

DECEMBER, 2014

Keywords: SRAM, memory, reliability, safety, error control coding, ECC, cyber-physical systems, CPS, erasure-based error control coding, EB-ECC, erasure, runtime erasure detection, ternary-output sense amplifier, TOSA, wordline assertion comparator, WLAC

Dedicated to my brother,

"Because your own strength is unequal to the task, do not assume that it is beyond the powers of man; but if anything is within the powers and province of man, believe that it is within your own compass also."

Marcus Aurelius

my mother,

"Being human always points, and is directed, to something or someone, other than oneself – be it a meaning to fulfill or another human being to encounter. The more one forgets himself – by giving himself to a cause to serve or another person to love – the more human he is."

Viktor Frankl

and my father.

"We make out of the quarrel with others, rhetoric, but of the quarrel with ourselves, poetry."

William Butler Yeats

Abstract

Reliability is of the utmost importance for safety of electronic systems built for the automotive, industrial, and medical sectors. In these systems, the embedded memory is especially sensitive due to the large number of minimum-sized devices in the cell arrays. Memory failures which occur after the manufacture-time burn-in testing phase are particularly difficult to address since redundancy allocation is no longer available and fault detection schemes currently used in industry generally focus on the cell array while leaving the peripheral logic vulnerable to faults. Even in the cell array, conventional error control coding (ECC) has been limited in its ability to detect and correct failures greater than a few bits, due to the high latency or area overhead of correction [43]. Consequently, improvements to conventional memory resilience techniques are of great importance to continued reliable operation and to counter the raw bit error rate of the memory arrays in future technologies at economically feasible design points [11, 36, 37, 53, 56, 70].

In this thesis we examine the landscape of design techniques for reliability, and introduce two novel contributions for improving reliability with low overhead.

To address failures occurring in the cell array, we have implemented an **erasure-based ECC scheme (EB-ECC)** that can extend conventional ECC already used in memory to correct and detect multiple erroneous bits with low overhead. An important component of this scheme is the method for detecting erasures at runtime; we propose a novel **ternary-output sense amplifier** design which can reduce the risk of undetected read latency failures in small-swing bitline designs.

While most study has focused on the static random access memory (SRAM) cell array, for high-reliability products, it is important to examine the effects of failures on the peripheral logic as well. We have designed a **wordline assertion comparator (WLAC)** which has lower area overhead in large cache designs than competing techniques in the literature to detect address decoder failure.

Acknowledgments

My deepest gratitude goes to my advisor and committee chair Ken Mai for providing me the opportunity to pursue graduate research and for his countless hours spent refining and revising my work. He has guided my progress through a shift in research area and shared his wealth of expertise in SRAM and digital circuit design. He has a unique ability to imbue his students with a sense of hope and purpose.

I am also very grateful to my committee members Prof. James Hoe, Prof. Andrzej Strojwas, and Dr. Alexander Hoefler for reviewing my proposal and dissertation and providing their feedback. Their time and flexibility made this possible.

My parents and my brother lent their unfailing support and attention during these years. I would like to thank them for always being there for me.

I had many productive discussions with my co-authors and colleagues Dr. Mudit Bhargava, Dr. Umut Arslan, Dr. Jangwoo Kim, Weinan Ma, Craig Teegarden, Çağla Çakır, Burak Erbağcı, Dr. Yu Cai, Yun Du, Nail Etkin Can Akkaya, Ray Carley, and Rachel Dondero. Their input improved the quality of my work and my life outside of work. We shared long hours together collaborating on tapeouts and papers, and the camaraderie will be remembered long after the PCBs are discarded. Special thanks to Mudit Bhargava for our long conversations and his technical advice which proved invaluable to my research and understanding. His selfless effort in the custom design and layout of the core SRAM on the EB-ECC testchip was a major contribution to the design. Special thanks also to Craig Teegarden and Çağla Çakır who did tireless work on the custom design and layout of the CAM on the EB-ECC testchip.

During my time in Pittsburgh, I've had the fortune of crossing paths with a number of people who enriched my life: Dr. Jason Brown, Dr. Jeff Nelson, Osei Poku, Juvyrose Santos, Esq., Dr. Kai Yu, Sonia Singhal, Naresh Bhatti, Matt Beckler, Kaship Sheikh, Ekin Sümbül, Kevin Biswas, Eric Menendez, Vitaliy Gleyzer, Brandon Mendoza, and Dr. Patrick Bourke, among others too numerous to list here. Thank you all for your encouragement; I relished our time together and look forward to continued friendship and correspondence.

The staff at CMU are some of the kindest and most professional people I've ever met. I am profoundly appreciative of the efforts of Dr. Michele Keffer, Elaine Lawrence, Judy Bandola, Samantha Goldstein and all of the administrative staff in the ECE department for handling so many concerns and making life run more smoothly for all graduate students.

I am thankful for the inspiration of my undergraduate advisor Dr. Daniel Saab, whose guidance led me to graduate school in the first place and who always made time for my questions.

This work was financially supported in part by the Semiconductor Research Corporation and Freescale Semiconductor. While working at Freescale in Austin, Texas, I was managed by John Eagan and mentored by Alexander Hoefler, whose guidance clarified the need for reliability and safety guarantees in cyber-physical systems, providing the inspiration to focus on a practically useful set of objectives, which proved to be a turning point in my research. The completion of this dissertation owes much to that collaboration.

Contents

List of Figures

List of Tables

List of terms and abbreviations

τ_{FO4} delay for a fanout-of-4 inverter. xviii, 32

0WL no wordline. xvi, 22, 23, 62, 63, 67–71, 73

2WL multiple wordline. xvi, 22, 23, 62, 63, 67–71, 73

BISR built-in self-repair. 8, 10, 24, 89

BIST built-in self-test. xvii, 3, 8, 10, 24, 35, 74–76, 78, 83, 86, 87, 89

CAM content-addressable memory. vii, 3, 43, 73, 78, 79, 86, 88

CBL check bitline. 67, 68

CBLC check bitline ceiling. xvi, 68–70

CBLF check bitline floor. xvi, 68–70

CDF cumulative distribution function. xv, 19, 53, 54, 56–59

CLSA current-latching sense amplifier. 47

CMOS complementary metal-oxide-semiconductor. xvii, 71, 73–75

CRI capacitive resist implementation. xiv, xv, 47, 49–51

DAQ data acquisition. xvii, 82

DECTED double error correcting, triple error detecting. xiii, 31, 32, 34, 40, 42, 83, 85

DMR dual modular redundancy, in which two redundant versions of a system operate in lock-step, declaring an error if they disagree. 31, 35

DRAM dynamic random access memory. 4

EB-ECC erasure-based error control coding. vii, xvii, 2, 74, 75, 83, 85

ECC error control coding. v, x, xii–xv, xvii, xviii, 2, 3, 8, 10, 20, 21, 28–34, 36, 39, 42, 47, 52, 54, 58, 59, 65, 71, 73, 74, 76, 78–81, 83, 85–87

eDRAM embedded dynamic random access memory. 4, 5

FWL false wordline. 22, 23, 62, 73

GWF gate work function. 13

HCI hot-carrier injection. 7, 8, 10, 27

HSNM hold static noise margin. 18, 19

LDPC low-density parity check. 91

LER line-edge roughness. 13

MC Monte Carlo. 19, 50, 52, 71

MLC multi-level cells. 91

NBTI negative-bias temperature instability. 8, 10

NFET n-channel field-effect transistor. xii, 6, 12, 16, 17, 67

OPC optical proximity correction. 9

PCB printed circuit board. vii, xvii, 82

PDAI parallel device assist implementation. xiv, 47

PDF probability density function. xiv, xv, 48, 49, 53, 55, 56

PFET p-channel field-effect transistor. 6

POST power-on self test. 43

RBL reference bitline. 67, 68

RBLC reference bitline ceiling. xvi, 68–70

RBLF reference bitline floor. xvi, 68–70

RDF random dopant fluctuation. 13

ROM read-only memory. xvi, 3, 36, 66, 69, 73, 77–79, 86, 87

RSNM read static noise margin. 18, 19

SA sense amplifier. 53, 54

SECDED single error correcting, double error detecting. xiii–xv, xvii, xviii, 30–32, 34, 40–43, 53, 54, 58, 59, 71, 77–81, 83, 85–87

SET single event transient. 11, 62, 71

SEU single event upset. 11, 20, 23, 61

SNM static noise margin. 18

SOI silicon on insulator. 25

SRAM static random access memory. v, vii, xii, xiii, xv–xvii, 2–8, 11, 14, 19, 20, 23, 25–28, 31, 34, 39, 47, 58–62, 65, 66, 74, 75, 77, 79, 87, 89, 91

TDDB time-dependent dielectric breakdown. 10, 26

TMR triple modular redundancy, in which three redundant versions of a system vote to decide the correct output. 31, 35

TOSA ternary-output sense amplifier. xv, 38, 46, 52–54, 59, 85

VLSA voltage-latching sense amplifier. xv, 47–51

WBWL wrong-block wordline. 23, 62, 63, 73

WLAC wordline assertion comparator. xvi, xvii, 3, 68–73

WM write margin. 18, 19

Chapter 1

Introduction

Microelectronics have become ubiquitous in modern society, and their functionality is rapidly overtaking many of the tasks of which previously only humans were capable. While this development promises to bring the exponential growth curve of Moore's law[1] [49, 50] to bear in new environments, there is a challenge in designing computer systems for reliability. For safety-critical tasks in the automotive, industrial, and medical sectors, both software and hardware must be guaranteed to operate exactly according to the designers' intent, or to fail safely with advance warning. Many of these safety-critical systems can be classified as cyber-physical systems, which are "engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components." [54]. Each component of the hardware design must be designed for reliability to mitigate the risks inherent in systems which perform tasks interacting with humans in their physical environment, such as self-driving cars, robotic assembly equipment, or life-sustaining implantable medical devices. One important component of any hardware design is its memory system, which stores and retrieves the instructions and data that control the operation of the cyber-physical system.

In microprocessors and systems-on-a-chip (SoCs), the embedded memory system plays a significant role in the overall system performance, area, and power. In modern microprocessors

[1]Gordon Moore originally predicted in 1965 that the number of transistors on a die would double every year; this was revised by Moore in 1975 to a density-doubling period of 24 months with a performance-doubling period of 18 months, and has held remarkably consistent up to the present day.

and SoCs, the cache hierarchy is often in the critical path of the design, can utilize 25-50% of the die area [1, 13, 20, 40, 62], and can consume an appreciable fraction of the total power. Thus, the reliability and yield of the memory system is crucial to the reliability and yield of the entire system. However, as process technologies scale, ensuring reliability is becoming increasingly difficult due to a number of factors including process variability, susceptibility to energetic particle strikes, and aging.

In this thesis, we examine the landscape of design techniques for reliability, and introduce two novel contributions for improving reliability with low overhead, these being an erasure-based error control coding (EB-ECC) scheme with runtime erasure detection, and address decoder failure detection by wordline assertion comparator. To better understand the practical implications of these techniques, we have fabricated a prototype testchip containing early versions of EB-ECC (including vertical parity protection [35, 45]) and address decoder failure detection.

First, we provide background on static random access memory (SRAM) and its role within the memory hierarchy, the ways in which it can fail and the manner in which those failures manifest in both the cell array and the peripheral circuitry. By so doing, we supply the groundwork to evaluate our novel contributions for detecting and correcting these errors.

In Chapter 3 we present a review of conventional reliability/resiliency techniques. We focus on the following techniques which have received wide industry acceptance: read/write bias assist, 6T/8T cell choice, cell sizing, error control coding (ECC), and redundancy. Since the novel techniques developed by the thesis contributions are mostly orthogonal to these conventional techniques, with the exception of the 8T cell which is incompatible with our designs for runtime erasure detection, our discussion of them is qualitative in nature. In Chapter 4 we present our work in cell array protection with EB-ECC. By identifying erasures, or the locations of bits likely to be failing, we can double the number of bits correctable by the ECC being used. This is a widely-known concept in communications, but to the best of our knowledge has not been realized for the purpose of improving SRAM reliability in the literature. The challenge of using erasures in SRAM is in their identification; we present low-cost solutions for both runtime detection and offline storage of erasures.

In Chapter 5 we discuss peripheral logic fault detection and present our wordline assertion comparator (WLAC) design. We evaluate the differences between an existing technique and our novel technique, generating enough information to allow designers to make a decision as to which scheme is suitable for a memory with given array size and specifications. The WLAC design has area overhead advantages over the current state-of-the-art address decoder failure detection design of read-only memory (ROM) address storage for large SRAM sub-arrays.

Finally, in Chapter 6 we present a testchip implementation of a few of the above reliability techniques on a 55nm bulk CMOS process. The fabricated testchip contains the first implementation of vertical parity and erasure coding in SRAM and thus represents an important step in the understanding of the technique's requirements and overheads at the circuit level. The testchip's erasure coding scheme is designed with two sources of erasures: an on-die content-addressable memory (CAM) which can be loaded with detected erasure locations for offline erasure storage, as well as full-swing bitline read latency failure detection circuits which provide run-time erasure hints to the ECC decoder. All these separate options can be toggled via scan-chain-controlled on-die built-in self-test (BIST).

Chapter 2

Background

In the introduction, we provided motivation and context for the thesis; in this chapter, we discuss fundamental aspects of SRAM design, and the choices designers make to attain yield and reliability. We cover the causes and ways in which memories can fail to provide the groundwork to evaluate our novel contributions for detecting and correcting these errors. Classifications of both types of failures and the various components of memories where these failures may occur are presented.

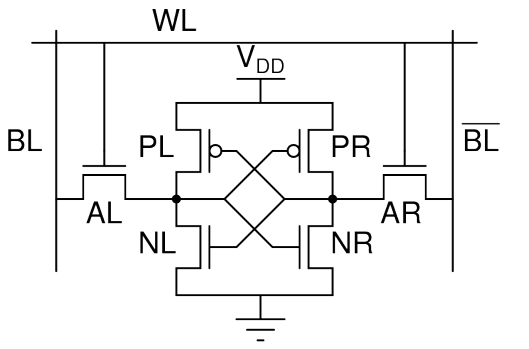
2.1 Memory system

In computer systems, the memory system is organized in a hierarchy in order to maximize system performance under cost constraints. The top level of this hierarchy is fast and small, while the bottom level is slow and large. They work together to give the illusion of being both fast and large by the process of caching, or keeping in the top levels of the cache hierarchy data which will be used in the near future. The cache hierarchy is divided between on-chip (embedded) and off-chip memory. The embedded memory space is currently dominated by SRAM because of its logic process compatibility and low latency, while off-chip memory is dominated by dynamic random access memory (DRAM) due to its density which reduces cost. Embedded dynamic random access memory (eDRAM) is also a commonly-used on-chip memory type for last level caches due to its density advantages over SRAM, though it requires additional fabrication steps. Recently, designers have had such strong demand for more capacity and bandwidth that they have packaged a separate eDRAM die alongside the CPU connected through on-package IO [22, 40].

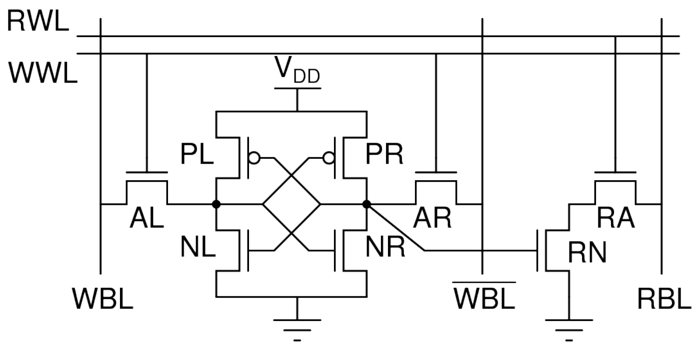
2.1.1 SRAM cell array

The fundamental element of SRAM is the cell, which contains bistable latching circuitry of two cross-coupled inverters with two access transistors (Figure 2.1a). These cells are arranged in two-dimensional arrays for compactness and simple access; a row of cells in the horizontal dimension share a wordline, while a column of cells in the vertical dimension share two or more bitlines (Figure 2.2). Under this arrangement, when the access devices of a row are turned on via a single wordline being asserted, each cell on that wordline is exposed to the bitlines, and can either perform a read or a write, depending on the state of the bitlines while the wordline is asserted. To perform a read, the bitlines are reset to V_{DD} prior to wordline assertion and then left floating. Thus, one of the bitlines will be pulled down by the cell's pulldown device and access device. This voltage difference between the bitlines can then be sensed by peripheral circuits. To perform a write, one bitline is driven to V_{DD} while the other is driven to V_{SS} by write drivers, and the wordline is asserted. When the access devices turn on, the cell's internal storage nodes will be overwritten to match the voltages on the bitlines.

Many variants on this basic six-transistor (6T) design exist to satisfy various requirements such as reliability, multiple-port access, or higher performance. Of these variants, we will focus on the 8T cell (Figure 2.1b) as it is most commonly employed to improve reliability over the 6T cell while maintaining similar area efficiency or incurring a small area overhead [14, 28]. When using the 8T cell without column multiplexing and operating on a single word per cycle, the read and write operations are separated and can be optimized separately, eliminating the 6T cell's conflict between improving read stability and writeability. Thus the original 6T portion of the 8T cell can be optimized for writeability without the concern that a read will disturb the value on the storage nodes. There are situations in which designers will still choose to



(a) 6T cell with **a**ccess device pair (left/right: AL/AR), pullup PFET (PL/PR) and pulldown NFET (NL/NR) pair forming bistable storage nodes.



(b) 8T cell. Added devices are read access (RA) and read pulldown (RN), while interconnect consists of write wordline (WWL) and read wordline (RWL), as well as differential write (WBL/$\overline{WBL}$) and single-ended read (RBL) bitlines.

Figure 2.1: SRAM cells

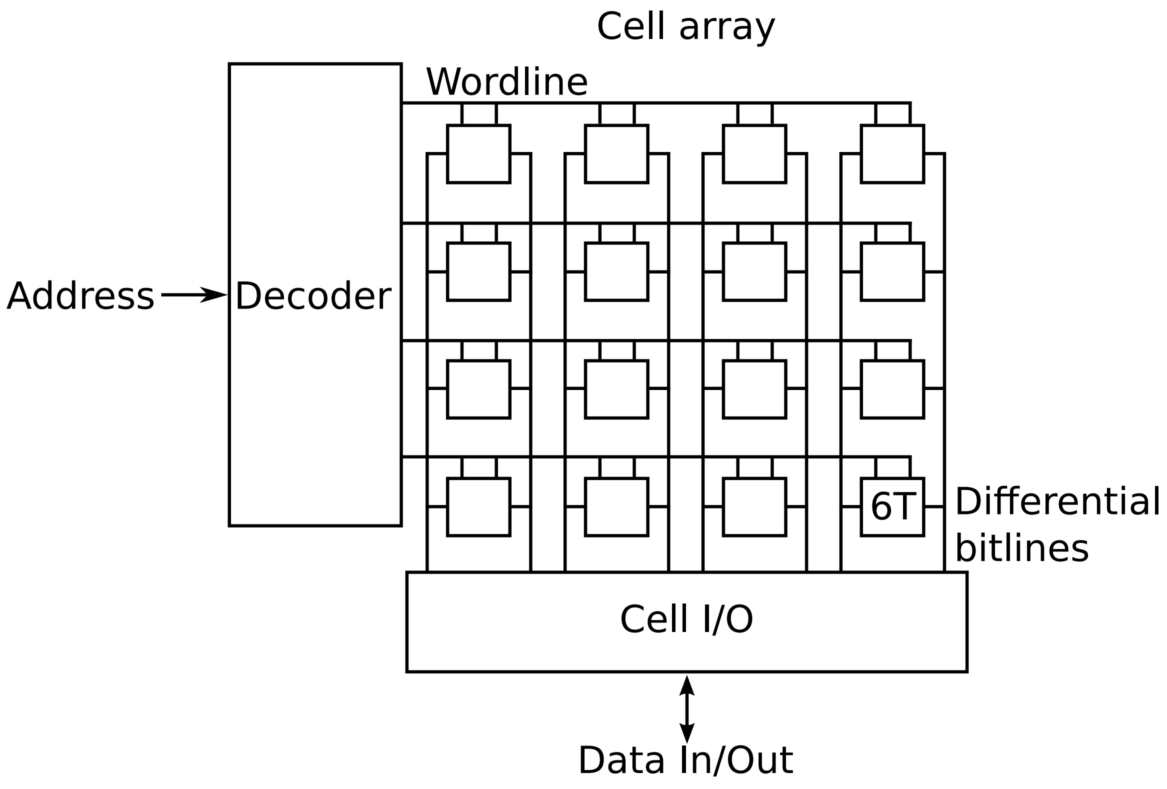


Figure 2.2: Simplified flow diagram of SRAM. Central boxes represent memory cells. Bitline can be single-ended or a pair of differential bitlines.

use column multiplexing with an 8T cell, however, which restores the risk of a read disturb on a write. In systems where the 8T cell's multi-port functionality is used to support a read and write in the same cycle, column muxing might be useful to the designer, particularly if the application does not require high density. Reducing the device sizing in the 8T cell to attain high density exacerbates the cell margins. The use of hot-carrier injection (HCI) techniques can make column muxing with 8T cells more desirable [26], but such a scheme has (to the best of our knowledge) yet to find commercial viability, perhaps due to the extra required manufacture-time testing burden.

2.1.2 SRAM peripheral circuitry

The peripheral circuits mentioned earlier consist of all the circuitry required to access the cells. We can categorize this logic into three paths: the address path, which drives the signals from the input address to the wordlines; the data path, which drives the signals from the bitlines to the input/output data ports; and the control path, which drives the command ports which indicate a read or a write to the local arrays. Since a memory may consist of several arrays of cells, an address decoder is required to determine which wordline in which array should be accessed and assert it. The remaining ports of the cell are the bitlines, which are controlled at the array periphery by reset devices, write drivers, and a sense amplifier to amplify the differential voltage on the bitlines to a digital data output. In the case of a single-ended bitline design, the sense amplifier may be replaced by an inverter.

Since the SRAM cell's devices are near-minimum sized for density while the wordline driver's devices can be sized quite large, it is in the designers' interest to minimize the bitline capacitance to reduce latency, while the wordline capacitance is not as critical a node. Thus, to improve the aspect ratio of the local cell array by reducing the bitline length while increasing the wordline length, column interleaving/multiplexing is commonly employed. It has additional benefits as well: it allows a relaxed 2, 4, or 8 cell pitch for shared bitline I/O circuits such as a large sense amplifier which would otherwise be quite difficult to lay out in a single SRAM cell's pitch. In addition, interleaving the columns of the SRAM spreads the bits of a single word apart so that an energetic particle strike of a certain radius is far less likely to upset multiple bits of the same word. For instance, a particle strike which would impact 3 bits of a word in an array without column interleaving would impact 1 bit in 3 separate words in an array with 4-way column interleaving, which can be corrected by a low-cost single-bit error-correcting code.

2.2 Error mechanisms in memory

Errors in memory can be detected via testing when the product is manufactured; in the case of these manufacture-time errors, there are multiple options. In situations where the error is minor, only affecting a single bit cell in a row or column of the memory, the error can be repaired by replacing it with a redundant row or column [24, 34]. If the error is sensitive to voltage or timing, it may be possible to employ product binning, configuring it to operate at a higher voltage, slower frequency, or even disabling portions of the memory so that the product can still be sold as a lower-performance or lower-capacity, less expensive option. In the worst case, if errors are caught at manufacture-time and cannot be dealt with through any other means, the product can be discarded before it reaches the consumer.

Errors that occur after manufacture-time testing are more pernicious and there are fewer options available to address them. BIST and built-in self-repair (BISR) mechanisms, or else some means of prevention, must be implemented to handle these run-time failures. Techniques to prevent failures, or to detect and correct them, provide a much-needed capability to the designer of reliable memory systems.

The well-known bathtub curve (Figure 2.3) illustrates the phases of product lifetime and the failure rate during those phases. Infant mortality failures in microelectronics are handled with burn-in test, while errors at runtime which can be caused by particle strikes, aging and wearout including HCI, negative-bias temperature instability (NBTI), electromigration, are handled with BIST and ECC among other countermeasures.

Errors can be coarsely classified along two orthogonal axes as either permanent (*hard*) or

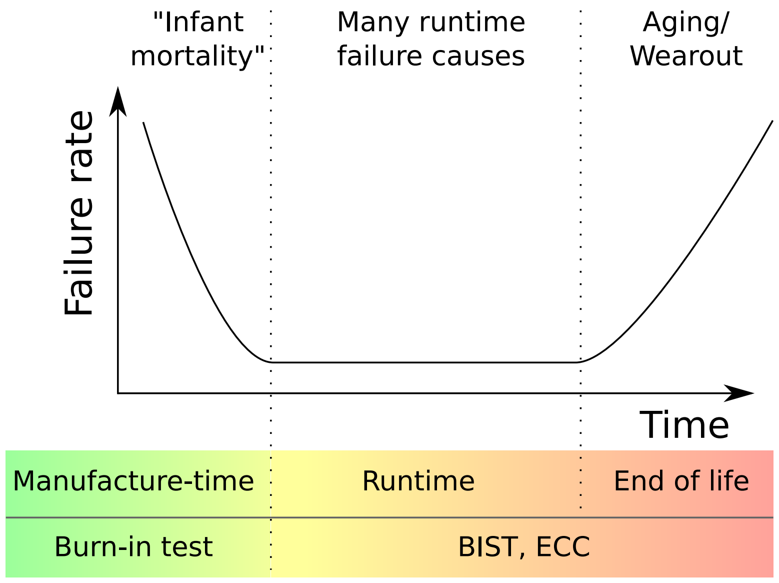


Figure 2.3: The failure rate over the product lifecycle is frequently depicted by this "bathtub curve". Broad categories of failure sources are shown above the curve, while lifecycle phases and conventional countermeasures are shown below.

non-permanent (*soft*)[1] and as occurring at either manufacture-time or during runtime [8]. We classify some common causes of errors along with conventional countermeasures here.

2.2.1 Hard errors

Hard errors which occur at manufacture-time may be caused by variability; process or mask defects such as particle/occlusion during fabrication which can cause bridge, stuck-at, or open faults. Conventional countermeasures include redundancy, disabling, or post-silicon tuning. At a lower level than these countermeasures, manufacturing defects are minimized by process engineers who employ computational techniques such as optical proximity correction (OPC) and tweak the design rules to limit the use of layout features which, during process characterization, display poor yield [59, 64]. One example of design rule constraints in modern process generations is that polysilicon gate direction is constrained to run in one direction in some designs, and for memory circuits where matching is very important, polysilicon gate lengths have

[1]Non-permanent errors can be further classified as transient if their cause is environmental or intermittent if their cause is non-environmental, but we do not make this distinction here.

been constrained to the same value within the cell array. The number of these design rules has been growing at a rapid pace in recent years as process engineers seek to simultaneously fully exploit the next process node's reduced-size geometry and limit the accompanying increase in manufacture-time errors. Recent research has presented a path away from this increasing design rule complexity by proscribing a limited set of regular pattern constructs or templates with which a wide variety of designs, including memory, can be synthesized. This differs from traditional standard-cell synthesized layouts in the way it minimizes the number of unique patterns, effectively making logic gates as regular and printable as memory [29, 51].

Hard errors occurring at run-time may be caused by energetic particle strikes which induce latch-up leading to thermal runaway, device aging/wearout including NBTI, time-dependent dielectric breakdown (TDDB), HCI, electromigration, or other causes. Conventional countermeasures include BIST/BISR, and ECC.

2.2.2 Soft errors

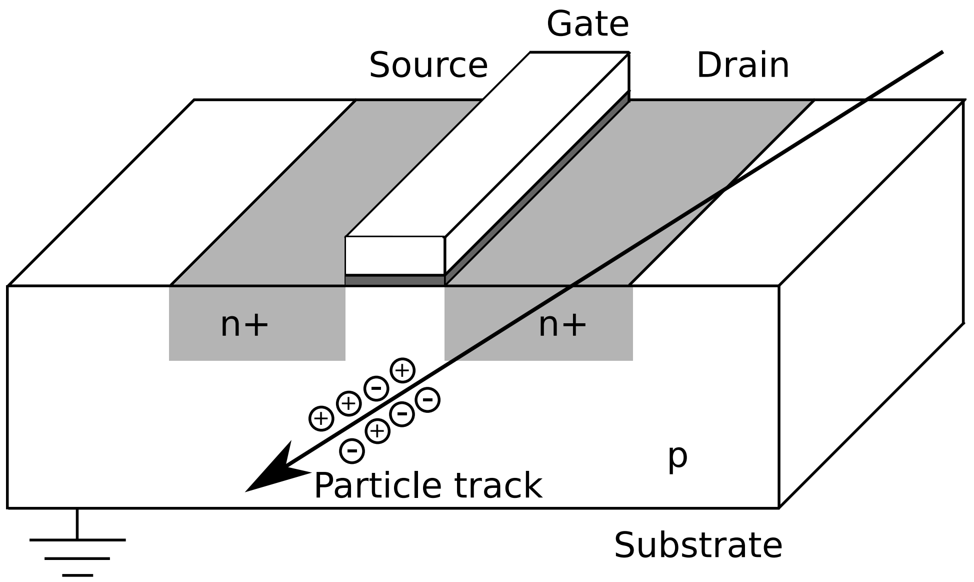
Soft errors are generally regarded as a phenomena which becomes apparent at run-time – at manufacture-time, if an error has occurred, is detected, and cannot be dealt with through one of the countermeasures mentioned above, the faulty product should not be shipped. Run-time soft errors may be caused by energetic particle strike including cosmic rays or α-particles; temperature variation; leakage; noise including power supply voltage droop, electromagnetic interference, capacitive coupling, or thermal noise; hazards and races due to poor design of critical timing paths; variation/physical irregularity in circuitry due to line edge roughness, random dopant fluctuation or other causes. Conventional countermeasures for these errors include bit interleaving in memory, ECC, fault detection, and read/write assist.

The energetic particle strike failure mechanism deserves further attention because it can often overcome designers' attempts to address it through design solutions.

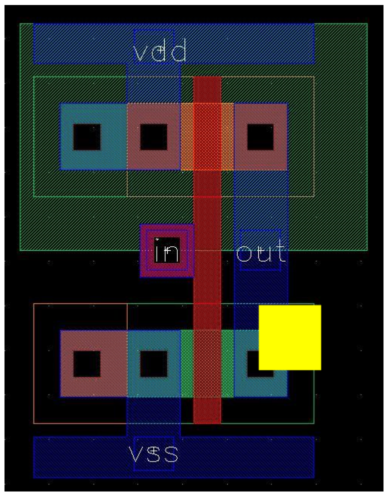
Energetic particle strikes

When an energetic particle strikes the silicon substrate, it excites electrons, separating positively charged ions and negatively charged electrons in the substrate via electronic stopping power. The energetic particle can be an alpha particle, which consists of two protons and two neutrons carrying a positive charge and represented by α or α^{2+}; it may also be a neutron created through the interaction of energetic cosmic rays with the Earth's atmosphere [18, 19, 83]. Electronic stopping power describes the energy loss of this energetic charged particle due to collisions with bound electrons in the substrate [30]. In the absence of a significant electric field nearby, the positive and negative charges will recombine. However, if this ionization occurs at a sensitive location near a transistor's channel, the electric field between the drain and body of the transistor separates the positive and negative charge carriers, preventing the recombination; next, the charge collected by the junction results in a transient current in the struck transistor from the substrate to the drain node (Figure 2.4). This is known as a single event transient (SET). Eventually, the charges dissipate and normal operation is restored, but the glitch may propagate if the disturbance is large enough in voltage and time. If the SET causes an incorrect value to be latched at some later node, it is then referred to as a single event upset (SEU). If an energetic particle were to strike a latch or an SRAM cell directly, it could easily cause a SEU without requiring the glitch to propagate through logic.

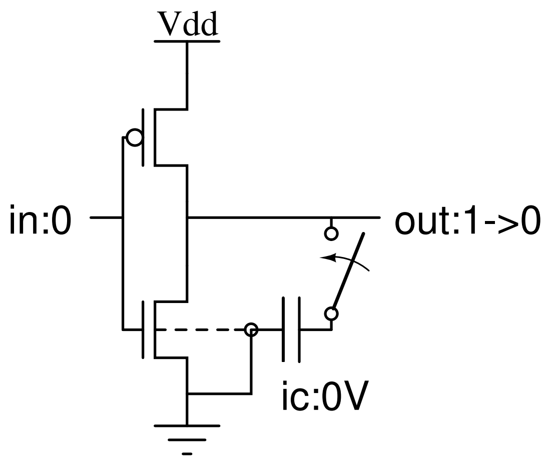
We present a study of particle strike locality in Section 5.1. Our investigation determines the potential functional fault modes resulting from energetic particle strikes on the address decode logic of an industrial design in the 40nm node. The study specifies the types of faults which will occur given a particle strike in a particular area of the physical layout. The analysis serves to guide our study of which techniques are the most effective at handling these faults given their overhead.



(a) Profile view of NFET and particle strike. When an energetic particle strikes the silicon substrate, the electric field between the drain and body of the transistor separates the positive and negative charge carriers, preventing recombination; next, the charge collected by the junction results in a transient current in the struck transistor from the substrate to the drain node.



(b) Layout view: yellow region indicates particle strike



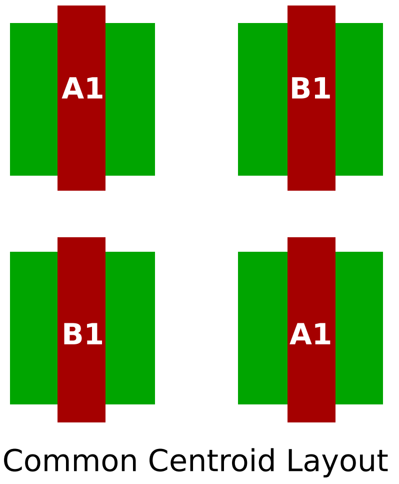
(c) Circuit diagram showing model of particle strike: switched capacitor between transistor drain and body with initial condition of 0 V potential.

Figure 2.4: Energetic particle strike causing current to flow across nominally off NFET until charge dissipates.

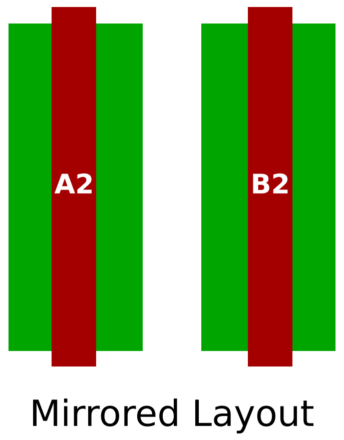
2.2.3 Parametric variability

One major contributing cause of both hard and soft errors in memory is the difference in parameters (such as threshold voltage V_t) of two identically designed transistors. The root causes of these parametric variations include gate work function (GWF) or oxide thickness variation, line-edge roughness (LER), and random dopant fluctuation (RDF) [16], which are exacerbated as transistor area decreases [57, 58]. Specifically, V_t variation due to RDF in the channel has been empirically shown to be inversely proportional to the square root of the channel area in the well-known Pelgrom model [58], reproduced in Equation 2.1. Note that this form of the Pelgrom model assumes two identically-drawn transistors in close proximity; devices farther from each other will suffer further mismatch due to global phenomena. In this equation, $\sigma(V_t)$ represents standard deviation of the device's threshold voltage, A_{Vt} represents the area proportionality constant specific to the process technology, and W and L refer to the device's width and length.

$$\sigma(V_t) = A_{Vt}\left(\frac{1}{\sqrt{WL}}\right) \tag{2.1}$$



(a) Systematic variation in horizontal and vertical dimension minimized as A1 and B1 will be equally affected.



(b) Systematic variation in vertical dimension minimized as A2 and B2 will be equally affected.

Figure 2.5: Careful layout including mirrored or common centroid configurations can minimize systematic variation, and is commonly employed in analog cells such as the sense amplifier.

This difference in parameters can cause the difference in latency between the worst and best cell to increase, and cause matching between devices in the cell to suffer and thus impact memory yield, reliability, and performance, since all these characteristics of a memory depend on its worst cell (after redundancy replacement). Device matching depends also on relative location; systematic variability sources may cause a gradient of a particular parameter across the die or wafer leading one side of a nominally matched pair of transistors to be consistently higher or lower in the parameter than the other. Careful layout including mirrored or common centroid configurations (see Figure 2.5) can minimize this systematic variation, and is commonly employed in analog cells such as the sense amplifier.

Additionally, researchers have proposed techniques for better matching of sensitive matched-pair transistors in analog applications through post-silicon statistical element selection rather than simple up-scaling of device sizing [4, 5, 31]. While incurring overheads due to the testing/configurability and non-volatile storage requirements, this improves reliability and yield in segments of the design which are highly sensitive to parametric variability.

We will describe the effects of parametric variability further in the next section, explaining how errors manifest in SRAM.

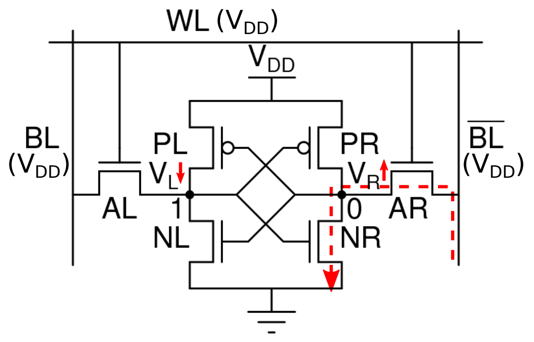
2.3 Error manifestations and cell failure types

Given that the aforementioned error mechanisms afflict embedded memory, there are a number of ways in which memory can fail. An understanding of these error manifestations is prerequisite to preventing, detecting, or correcting them. As before, we separate our discussion into two areas: the cell array and peripheral circuitry.

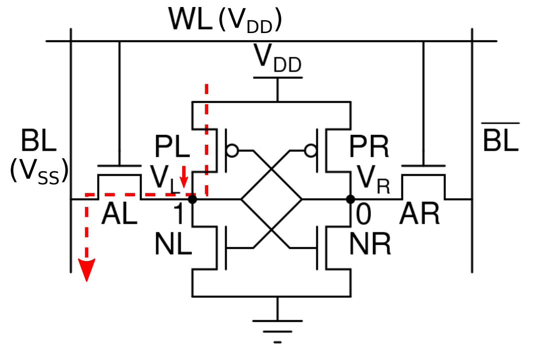
2.3.1 Cell array

There are four primary failure types associated with the cell and its local periphery: read disturb failure, write failure, access failure, and retention (hold) failure [36, 37, 52, 76] (Figure 2.6).

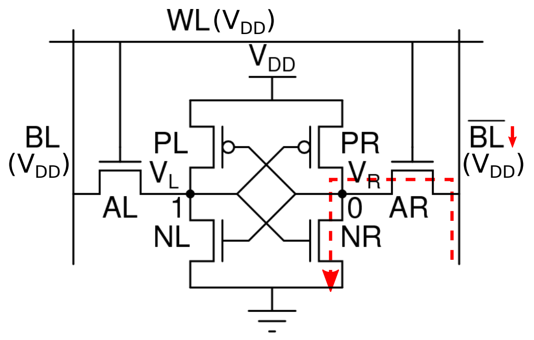
In general, the bitline capacitance is much greater than the cell's internal storage node ca-



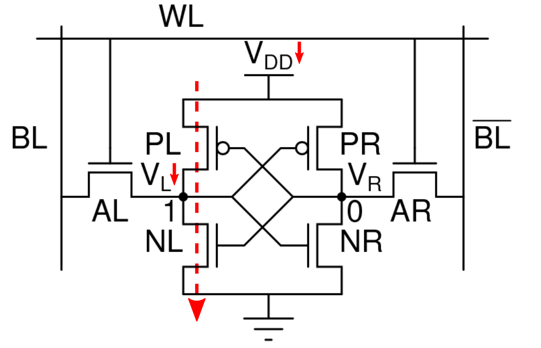
(a) Read disturb failure: Cell upsets when V_L drops to V_{SS}.



(b) Write failure: Cell fails to upset when V_L drops too slowly



(c) Read latency failure: Cell fails to read when $\overline{BL}$ drops too slowly



(d) Hold failure: Cell loses its value when V_{DD} drops too low and/or noise occurs at cell storage nodes

Figure 2.6: 6T cell failure types. Mismatched transistors and/or another fault within a cell can cause failure in cell operation (initialized as $V_L = V_{DD}$, $V_R = V_{SS}$).

pacitance, so when the wordline turns on the access transistor and the current flows from the bitline through the pulldown transistor, the internal storage node holding a 0 bumps up through voltage division over the two transistors. A read disturb failure may occur if this voltage rises far enough to turn on the opposite side's pulldown transistor (see Figure 2.6a). In designs which employ column multiplexing, a read disturb failure can occur during a write cycle, as there are cells activated by the wordline which are not intended to be written. These cells are generally referred to as "half-selected cells", as they are selected by the same wordline as the word activated for a write, but their bitlines remain charged or are left to float, rather than being driven for a write as those of the correctly column multiplexed bitlines. While there does not yet appear to be a consensus on terminology for this event in the literature, in this thesis we will describe this as a "half-select disturb failure". It has also been referred to in the literature as a partial write disturbance (PWD).

A write failure occurs when the cell retains its value instead of the expected behavior of being overwritten by a wordline pulse with bitlines split (see Figure 2.6b). Prior to the wordline assertion, the storage nodes are held at their values by the cross-coupled inverters. Normally, the large capacitance ratio between the bitline capacitance and the internal storage node capacitance and the low resistance of the activated access transistor is sufficient to overcome the drain current of pullup device PL and pull down the internal storage node V_L. However, if mismatch causes the access device AL to have a higher resistance than average or the pullup device PL to have a higher drain current than average, the node V_L may not be pulled below the trip point of PR/NR before the wordline is de-asserted and thus the cell's value will not be overwritten.

A read latency failure, also known as an access failure, occurs when the cell's read current is insufficient to meet the design's timing requirements (read latency) (see Figures 2.7 & 2.6c). Within the cell which suffers from read latency failure, this may occur due to the access or pulldown NFETs having lower drain current than average due to a number of causes (see Section 2.2.1). But the failure cannot always be diagnosed as being caused solely due to variation within the cell; at the level of a single column, the leakage of other cells on the bitline may reduce the bitline differential due to the access NFETs having higher leakage current than average. Finally, the sense amplifier measuring the bitline differential of the column may suffer from mismatch in its nominally symmetrical structure causing input-referred offset.

In an equation, the requirement is shown in Equation 2.2, where N is the number of cells in a column, $T_{WL \to SAE}$ is the time between wordline assertion and sense amplifier enable assertion (the time the 6T cell has to produce a voltage differential on the bitlines), and C_{BL} is the bitline capacitance.

$$|V_{SAoffset}| < \frac{(ION - (N-1) * IOFF) * T_{WL \to SAE}}{C_{BL}} \tag{2.2}$$

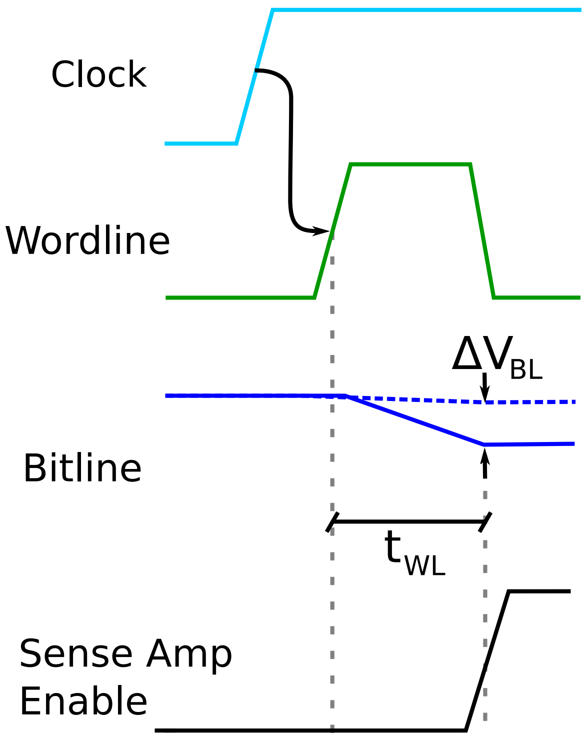
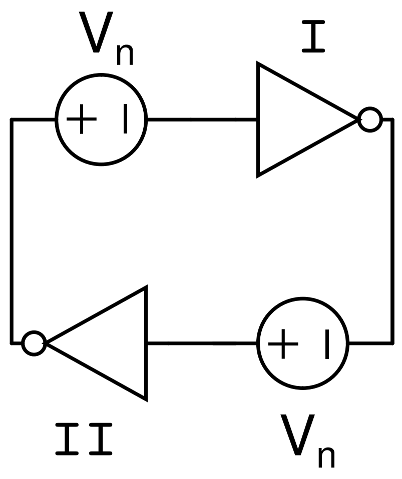


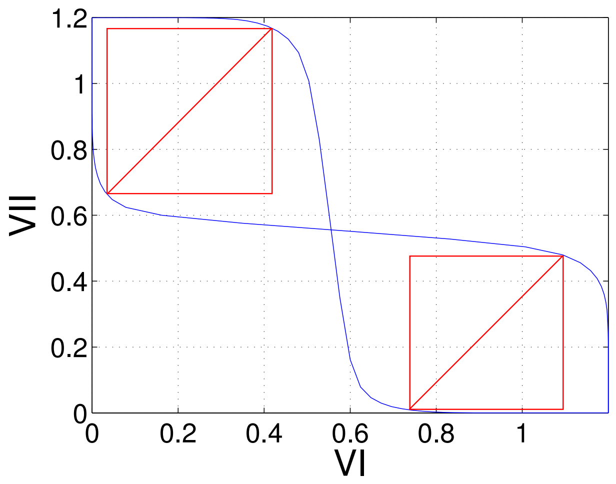
Figure 2.7: Depiction of read latency and critical $t_{WL \to SAE}$ delay. If $\Delta V_{BL} < V_{offset}$ at the time of SAE, the value on the bitlines is incorrectly sensed and a read latency failure will occur.

A hold failure, also known as a data retention failure, occurs when a cell loses its stored value after a period of time in which it is not accessed (see Figure 2.6d). It is often precipitated by a reduced cell supply voltage or noise on the internal storage nodes.

To quantify the manufacture-time yield of a memory design or the reliability of a cell during operation, we can measure cell metrics through simulation. Each metric indicates the magnitude of event or disturbance from nominal behavior which would have to occur at a certain point in the cell (or local to it) which would induce the failure type. For example, one commonly cited cell metric is the static noise margin (SNM), which quantifies cell stability and can be determined graphically by plotting the voltage transfer characteristics of the two cross-coupled inverters and measuring the sides of the largest squares inscribed between them [69]. The smaller of the two maximum squares' sides represents the cell's SNM. This plot is commonly known as the butterfly curve (see Figure 2.8). The read static noise margin (RSNM) and hold static noise margin (HSNM) describe versions of the SNM measured with the cell's access transistors turned on and off, respectively [52].



(a) Circuit for SNM measurement



(b) Butterfly curves illustrating SNM inscribed in squares

Figure 2.8: Static noise margin measurements [69]

The write margin (WM) can be determined in two ways. First, by setting both bitlines opposite the values stored at the cell's storage nodes (in Figure 2.6, $BL = V_{DD}$, $\overline{BL} = V_{SS}$) and sweeping the wordline from V_{SS} to V_{DD} to find the voltage at which the cell upsets, and subtracting this voltage from V_{DD} (as a more writeable cell should have a higher margin) [21, 72]. Second, by initializing the wordline and both bitlines to V_{DD} and sweeping the bitline on the side of the storage node initialized to V_{DD} to V_{SS} to find the voltage at which the cell upsets [23, 84].

This second measurement is also referred to as the write-trip point. Dynamic analysis of SRAM cells has found that the former static method of determining write margin (by sweeping the wordline from V_{SS} to V_{DD}) correlates best with the dynamic write time metric [74].

The metrics of RSNM, HSNM, and WM can be classified as functional noise margins; if any are below 0 due to manufacture-time variation, or some run-time disturbance exceeds the margin, the cell will fail to function due to read disturb failure, hold failure, or write failure. On the other hand, performance metrics of the cell include read current and leakage current observed at the bitline when the wordline is asserted or de-asserted. These are also significant to reliability, since one cell's poor read current can cause read latency failure, while an array's large leakage current may cause supply droop or exceed power requirements for the design. Measuring cell current is an indirect way of measuring the read latency independent of cell array choices such as bitline capacitance, which depends on column height and sense amplifier choice among other options. These measurements, specifically the ratio of read current to leakage current or more directly the read latency if the cell array choices are known, can generate a figure of merit which defines the risk of an access failure.

Unlike functional metrics, performance metrics cannot determine a cell failure independent of other array decisions and specifications. In order to conclude that a cell's read current is below the passing threshold and would suffer an access failure, a threshold must be established, and sometimes this threshold can be adjusted at manufacture-time using product binning or after manufacture-time using configurable self-timing [4]. But if a cell suffers a read disturb failure (which upsets the stored value of the cell), this happens regardless of conventional array design decisions.

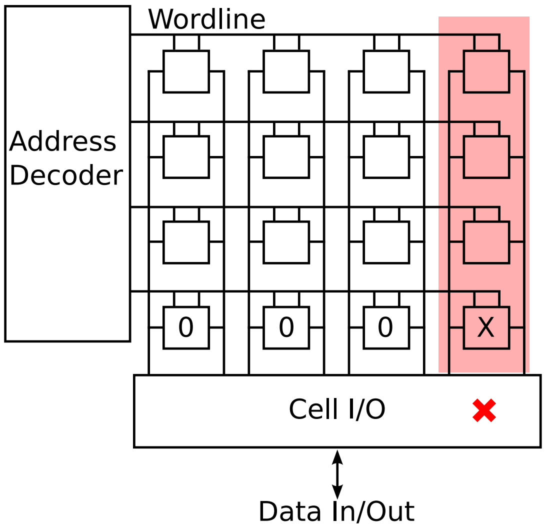
The average value of each metric across process and mismatch variations can be determined statistically for a given cell design and technology via Monte Carlo (MC) analysis. More importantly, the cumulative distribution function (CDF) of the metric can also be determined, allowing designers to give an expected yield figure for a memory of a given size, provided that the model files are accurate. MC analysis can estimate mean and standard deviation from a sample within a confidence interval given a certain number of samples. The 99.7% (3σ) confidence interval for a sample drawn from a normally-distributed population is defined by Inequality 2.3, where $\bar{x}, \sigma, N, \mu$ are defined as the sample mean, sample standard deviation, sample size and population mean, respectively [15, 65].

$$\bar{x} - 3\frac{\sigma}{\sqrt{N}} \leq \mu \leq \bar{x} + 3\frac{\sigma}{\sqrt{N}} \tag{2.3}$$

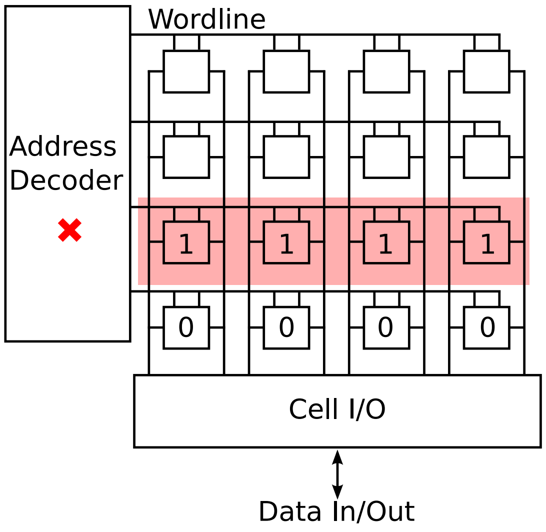
2.3.2 Peripheral circuitry

While most study has focused on the SRAM cell array, for high-reliability products, it is important to examine the effects of failures on the peripheral logic as well. Unlike in the cell array, where a failure in a storage element can cause immediate corruption of data, SEUs in the SRAM peripheral logic aren't generally able to cause an error in isolation. Rather, the effects of the SEU depend on the location as well as state of the memory at the time of the upset.

Failures occurring in certain segments of the peripheral logic manifest in ways unlike those which occur in the cell array. As classified in Section 2.1.2, the data path drives the signals from the bitlines to the input/output data ports. A particle strike or other failure mode affecting a portion of the data path would result in a disturbance to one or more bits of a word as opposed to the entire word (see Figure 2.9a). In such a situation, the resulting failure effects resemble those of a failure in the cell array. As such, these can be handled by the same techniques which handle failures in the array itself, such as ECC. Of course, not all failures are preventable or even detectable. The address path drives the signals from the input address to the wordlines/column select. A particle strike or other failure mode affecting a portion of the address path may manifest in a distinct way which is more challenging to detect. Since the final outputs of the address decoder are wordline and column select signals, the failure manifestations generally involve suppression of the intended wordline/column select signal and/or activation of an unintended wordline/column select signal. Since these failure manifestations involve the entire word rather than just a part of it, and could conceivably result in an undetectable failure (see Figure 2.9b).

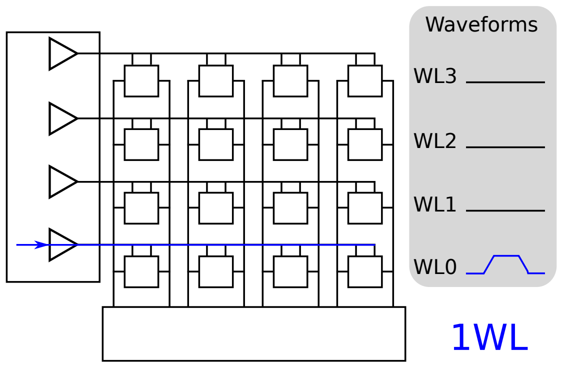


(a) Data path failure: one or more columns may have their data corrupted, similar to a failure somewhere in the cell array. These types of failures have conventionally been handled using interleaving and ECC.

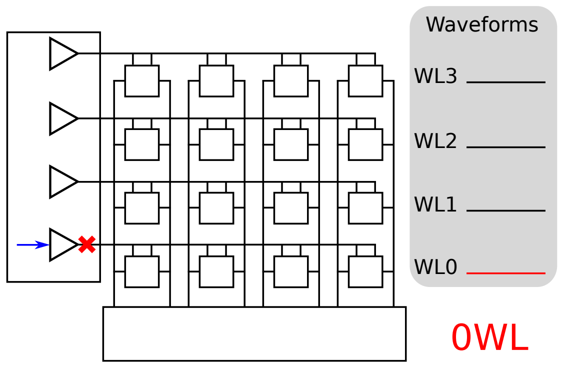


(b) Address path failure: another wordline whose data is opposite that of the intended wordline is asserted in error, potentially causing the entire word to be read incorrectly, but since the data from the aggressor wordline is a valid codeword, ECC will not detect that an error has occurred.

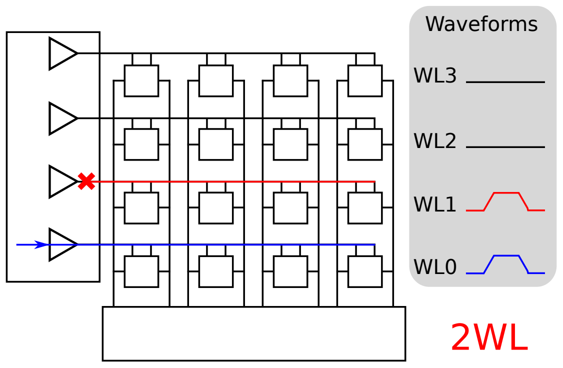
Figure 2.9: Failures occurring in the data path may be detectable or correctable by ECC; failures occurring in the address path are commonly neither detectable nor correctable by ECC and so must be dealt with through other means.



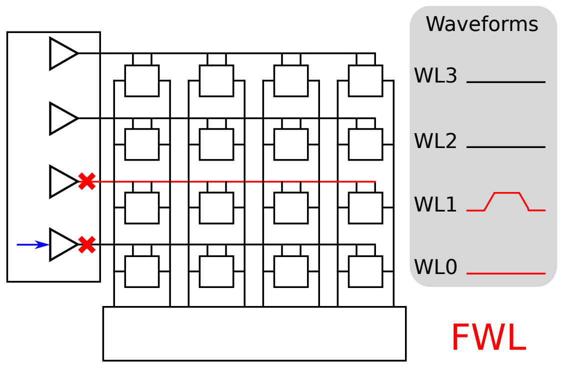
(a) 1WL: Correct operation



(b) 0WL failure



(c) 2WL failure



(d) FWL failure

Figure 2.10: Address decoder failure types.

The address decoder can fail in one of three ways:

1. 0WL failure (see Figure 2.10b): The desired wordline is not asserted, and no other wordline is asserted. Essentially the assertion of the wordline is suppressed.
2. Wrong-row failure: The desired wordline is not asserted, but one or more other wordlines is asserted. This type of failure has two sub-types depending on the proximity of the desired wordline to the improperly asserted wordline. If the two wordlines are in the same block, we term this a FWL (see Figure 2.10d). Otherwise, we term this a wrong-block wordline (WBWL). Sometimes we also refer to a WBWL failure as a "1WL" failure when we're examining the failures within a single block. The "1WL" terminology indicates a failure where the expected behavior was 0 wordlines in the block asserted but the actual behavior was 1 wordline in the block asserted.
3. 2WL failure (see Figure 2.10c): The desired wordline is asserted, but one or more other wordlines is asserted as well, which may cause contention on a read and data corruption on a write.

For simplicity, the terms "row" and "wordline" are used to refer to the address decode path's output; it should also be noted that the column select suffers from the same failure types.

Information on fault modes potentially resulting from these SEUs and their relative likelihood could guide designers in handling these faults using techniques which are most effective given their overhead. We present a study of particle strike locality in an industrial address decoder in Section 5.1. Additionally, we discuss conventional techniques for detecting peripheral logic failure in Section 3.6 and propose a new technique with lower area overhead for large cache designs in Section 5.2.

2.4 Summary

In this chapter we've discussed the basic structures of SRAM design: an overview of the 6T and 8T cell options, the array structure, and the peripheral circuitry. Failure causes and symptoms are examined, with a particular emphasis on failures which may occur at runtime, since BIST/BISR is generally limited in scope compared to testing and repair which can be performed at manufacture-time. The issues of parametric variability and functional noise margin are problems facing designers which cause difficulty maintaining strong guarantees of reliability. Next, we discuss conventional reliability techniques in Chapter 3 and novel reliability techniques in Chapters 4 and 5.

Chapter 3

Conventional reliability techniques

There exist a multitude of published memory design techniques which are employed to improve reliability/resiliency; of these, a subset have been adopted into wide commercial use. We discuss six of the most common solutions; specific implementations of these solutions differ in their overhead and efficacy so our discussion here is qualitative.

For the purposes of this thesis, what we consider to be a "reliability technique" is any circuit-level design choice which reduces the likelihood of a failure condition (or improves reliability) at the cost of some quantifiable increase in overhead, whether that be latency, area, or power. The limitation of circuit-level choices ensures that these techniques can be employed by the designer freely even after the technology decisions have been made relating to process steps which may improve reliability such as the use of silicon on insulator (SOI) to limit the effects of energetic particle strikes. These process-level choices are beyond the scope of this thesis.

3.1 Read/write bias assist techniques

Bias assist techniques modify one or more of the SRAM cell's ports (V_{DD}, V_{SS}, V_{BL}, V_{WL}, V_{body}) in order to reduce overall cell failure probability or increase cell read or write margins. Overcoming the inherent tradeoff between read stability/writeability generally requires designers to carefully choose their assist ports in combination with cell array architecture, because most modifications that improve read stability hurt writeability and vice versa. This is not an issue in single-ported cell arrays which have no column muxing/interleaving since in each cycle, only a read or a write occurs in a given word, and thus the read bias assist can be limited to cells active during a read, and similarly for the write bias assist.

To improve read latency and reduce failure rate, many designers have chosen to implement a form of read assist circuitry which boosts cell supply voltage or under-drives the wordline to limit cell upset during read [6, 60, 84].

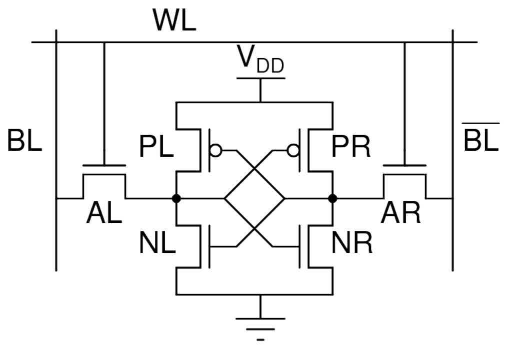
Conversely, to improve write performance and reduce failure rate, many designers have chosen to implement a form of write assist circuitry which collapses the cell supply voltage, boosts the wordline voltage, or drives the low bitline voltage below ground, to ensure the cell is properly written [6, 79]. Often these two assist techniques go hand-in-hand; the same supply node is chosen to be adjusted in opposite directions for a read or a write, or else one node is adjusted for read assist while the other is adjusted adversely for write assist [32].

One potential limiting factor with using negative bitline (NBL) as a write assist technique is that of a long-term reduction in reliability due to larger than VDD bias across the gate oxide, leading to potential BTI and TDDB issues. However, recent work has asserted that this is not a concern based on results of burn-in testing on VDDmin shift and standby leakage increase [17]. The only other caveat is a reported high energy overhead [85].

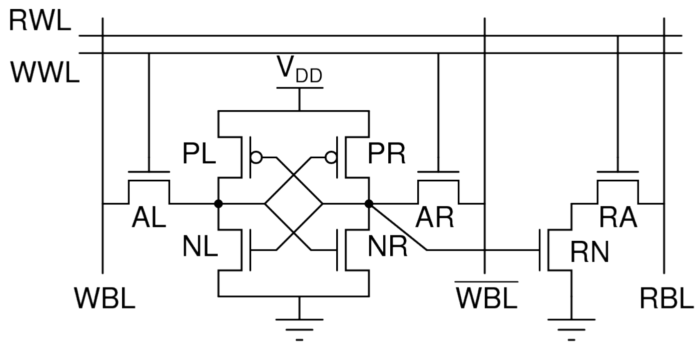
Body biasing has also been employed to reduce the impact of process variation, but it is generally used as a process-calibration method to recenter a design whose transistors' threshold voltage is skewed far from the modeled value.

3.2 Cell choice (6T/8T)

While the 6T SRAM cell (Figure 3.1a) has become the de facto standard in traditional designs (not including sub-threshold memory designs), in recent years the 8T SRAM cell (Figure 3.1b) has been commonly used to reduce failures and thus increase reliability, as well as separating the read and write paths in the cell so that optimization of read and write are no longer mutually



(a) 6T cell: High density and differential read bitlines, but inherent read stability/writeability tradeoff in design.



(b) 8T cell: reduced vulnerability to read disturb.

Figure 3.1: SRAM cells

exclusive [14, 28]. The read stability failure mode (Figure 2.6a) is completely removed (rendered identical to the hold stability failure mode) since the read bitline (RBL) no longer interacts with the cell's storage nodes. However, due to the risk of half-selected cells being upset on a write, column muxing might not be desirable for designers when using 8T cells. There are situations in which designers will still choose to use column multiplexing with an 8T cell, however, which restores the risk of a read disturb on a write. In systems where the 8T cell's multi-port functionality is used to support a read and write in the same cycle, column muxing might be useful to the designer, particularly if the application does not require high density. Reducing the device sizing in the 8T cell to attain high density exacerbates the cell margins. The use of HCI techniques can make column muxing with 8T cells more desirable [26], but such a scheme has (to the best of our knowledge) yet to find commercial viability, perhaps due to the extra required manufacture-time testing burden.

3.3 Cell sizing

One simple method of diminishing error likelihood from several different sources is to increase cell size by increasing transistor width, particularly access and pulldown transistors [33]; this increases area and power overhead but may improve latency. As discussed in Section 2.2.3, since the standard deviation of the threshold voltage and thus performance of each transistor in the cell is inversely related to its area [58], matching should improve as transistor size and thus cell size increases. This reduces failures due to variability, which can be a major contributor to failure even if it is not able to induce a failure by itself. However, increasing cell sizing limits density and is thus an undesirable solution for on-die last-level caches where high density is required.

3.4 Error control coding

The use of ECC allows a system to operate correctly in the presence of bit errors in the cell array. In the context of embedded memory, which generally has lower latency than off-chip memory and disk storage owing to its close proximity to the processing cores, the codes most frequently used are linear block codes, which operate on a block of data at a time rather than operating on a stream of bits, as is the case with cyclic codes. Block codes are thus more compatible with the fixed word-length environment of SRAM. A code, along with its encoder and decoder, is defined by its parameters (n, k, d_{min}), where n represents the length of the codeword, k represents the length of the message word, and d_{min} represents the minimum Hamming distance, defined as the distance between two codewords [43] (see Figure 3.2). The codeword is (n-k) bits larger than the message word; these extra bits are generated algorithmically from the value of the message word by the encoder, and are variously referred to as check bits or parity bits. The full codeword including check bits is used by the decoder to determine whether or not there has been an error and, if the number of bits in error is less than or equal to t, the corrected codeword. The rate of the (n, k, d_{min}) code is defined by Equation 3.1; a higher rate indicates a lower burden of added check bits and thus lower area penalty.

$$R = \frac{k}{n} \leq 1 \tag{3.1}$$

The designer must choose the code strength and data word size to meet system require-

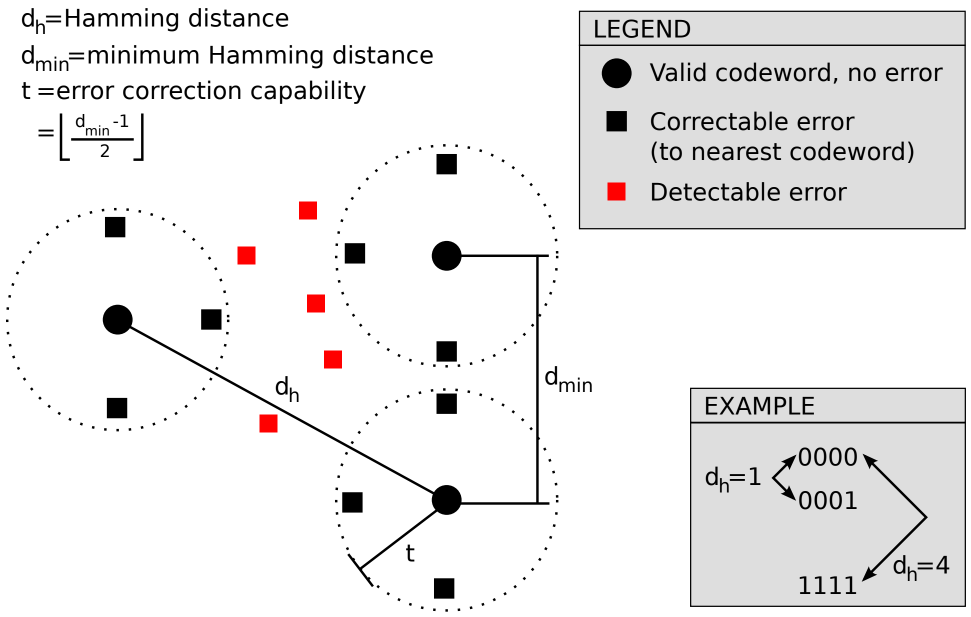
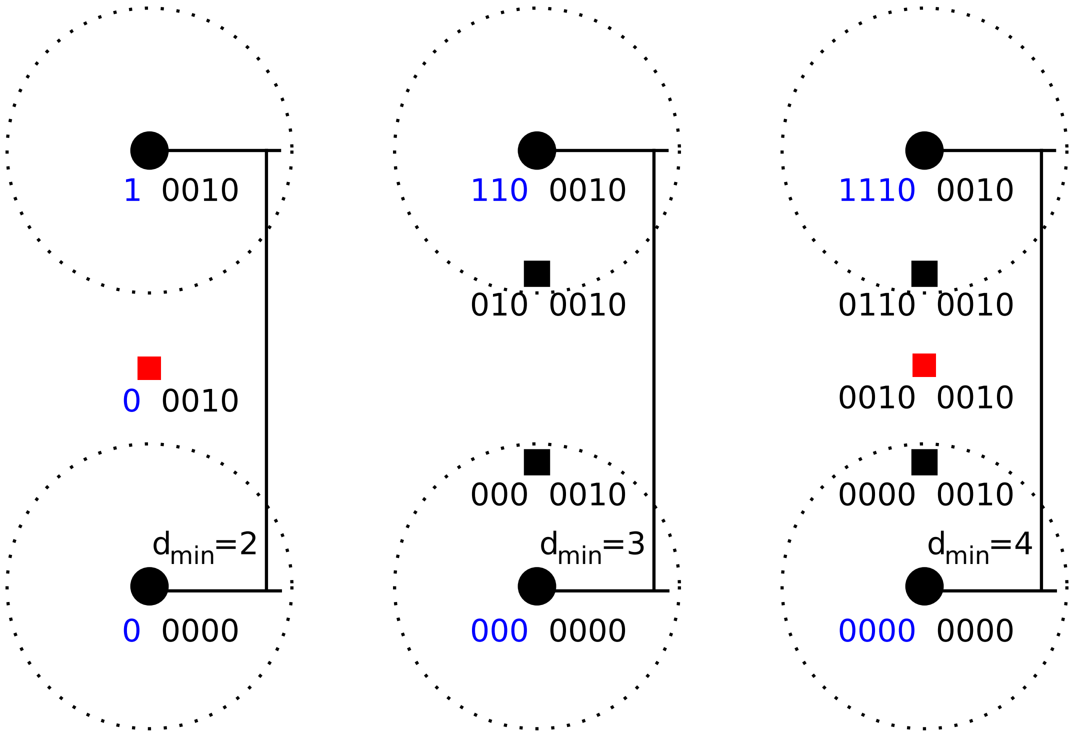

d_h=Hamming distance
d_{min}=minimum Hamming distance
t =error correction capability
$=\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$
LEGEND
Valid codeword, no error
Correctable error (to nearest codeword)
Detectable error
d_h
d_{min}
t
EXAMPLE
d_h=1
0000
0001
1111
d_h=4

Figure 3.2: Hamming spheres illustrate the limitations of error detection and correction. Each sphere, centered on a codeword, contains a number of vectors which are a maximum Hamming distance t from it and are thus correctable. Vectors outside the sphere are detectable as errors. Note that vectors which have more than t bits changed (distance) from their correct codeword may fall in the wrong sphere and thus be miscorrected. This represents an undetected error.

ments for error protection and circuit overheads. There is an inherent tradeoff between these two measures. A strong code has a large minimum Hamming distance (d_{min}). For example, a single-error-correcting Hamming code has a d_{min} of 3, which indicates that the least number of bits differing between two codewords is 3 (see Figure 3.3). Practically, since all Hamming codes have the all-zero codeword, this indicates the least number of 1's in a non-zero codeword.

3.4.1 Conventional ECC overheads

The tradeoff when using ECC is primarily a sacrifice of latency for yield and reliability. There is also an increase in area and power for the encode and decode logic as well as the storage of parity bits, and these grow exponentially with larger numbers of correctable bits. The latency overhead is generally higher for the decoder than the encoder in a code with $d_{min} > 2$. While the encoder operates on a k-bit message word input and computes (n-k) check bits output in a series of XOR operations, the decoder has multiple stages which begin with an n-bit codeword input



(a) Parity (5, 4, 2) **(b)** SEC (7, 4, 3) **(c)** SECDED (8, 4, 4)

Figure 3.3: Hamming sphere representation of ECC of various correction capacities. Black bits represent message word bits, blue bits represent added parity bits, black circles represent codewords, black squares represent correctable errors, red squares represent detectable errors, and the dotted circles represent decoding spheres whose radius is $t = \lfloor (d_{min} - 1)/2 \rfloor$. Comparison of (n, k) codes from d_{min}=2 to 4.

and conclude with a corrected n-bit codeword output and flags indicating whether the error was correctable or not.

Hsiao SECDED ECC is generally regarded as a de facto standard for error-correcting codes employed in cache memory due to its minimal area/latency overhead [27, 66]. Double error correcting, triple error detecting (DECTED) codes have been used in SRAM as well since at least 2009 for higher levels of cache with higher latency, but owing to their high overhead and variable latency of correction, they are generally kept at or above last-level on-die cache [41, 63, 67, 68]. Codes with higher d_{min} and correction capability are rare in mass-production microprocessor designs for the terrestrial environment. More frequently, companies with higher reliability/safety requirements for their memory designs will employ array and component[1] DMR or TMR to protect against a larger number of potential failures, especially particle strikes. In DMR, two redundant versions of a system operate in lock-step, declaring an error if they disagree; in contrast, in TMR, three redundant versions of a system vote to decide the correct output. However, these array-level or component-level redundancy schemes incur larger area penalties than redundancy on the row or column level, increasing cost.

As shown in Figure 3.4, the (72, 64, 4) SECDED decoder consists of a syndrome generator (27$\rightarrow$1 XOR tree), a syndrome decoder (two-level NAND), and 72 XOR gates to perform the correction of a single-bit error in the 72-bit word. The syndrome vector is an intermediate result in the error correction process which uniquely identifies the error location for a single-error-correcting code. For block codes which can correct multiple error bits per word ($d_{min} > 4$), the syndrome decode logic requires a significantly higher latency and area.

3.4.2 Multi-bit ECC

Despite the potential of inline multi-bit ECC to increase memory resilience and yield, its overhead, primarily due to the long latency of correction, has limited its use in the on-die memory system. Prior work has employed cache line disabling [77] or bit-fix [76] to minimize the archi-

[1]By "component", we refer to a large logic block of the memory such as the address pre-decoder or row decoder.

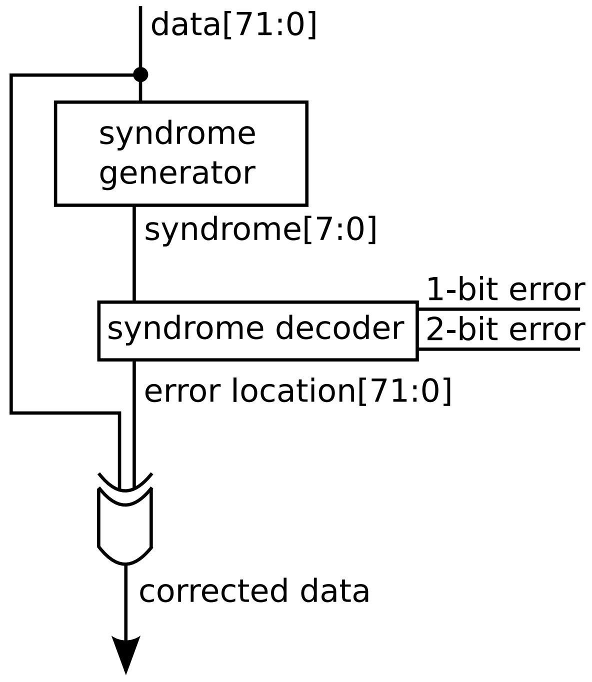


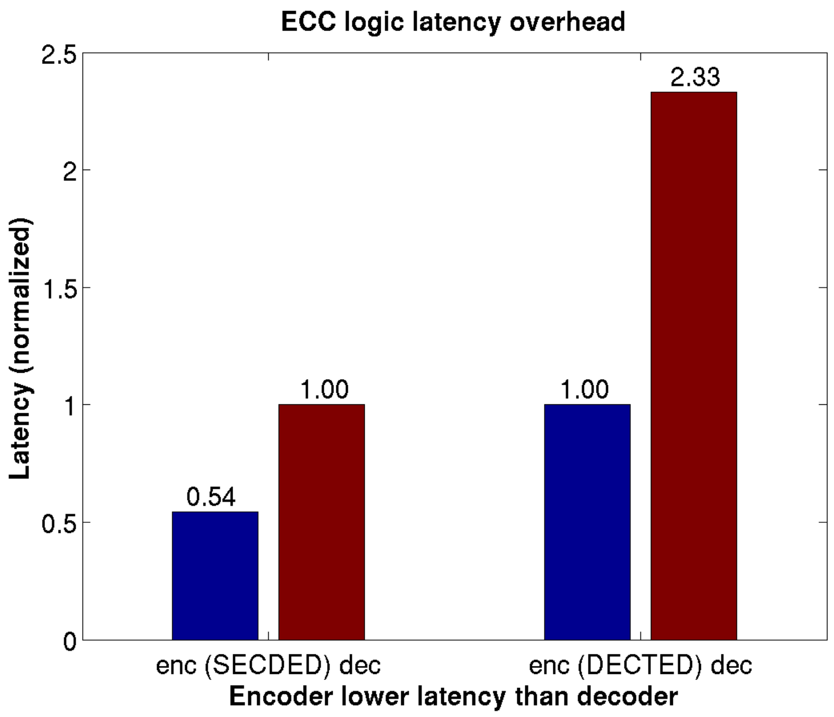
Figure 3.4: Decoder for (72, 64) SECDED ECC, minimum Hamming distance $d_{min} = 4$

ECC unit	Hsiao SECDED Latency (ns)	Hsiao SECDED Area (μm^2)	DECTED Latency (ns)	DECTED Area (μm^2)
Encoder	0.7	1168	1.3	2546
Decoder	1.3	2681	3	42976

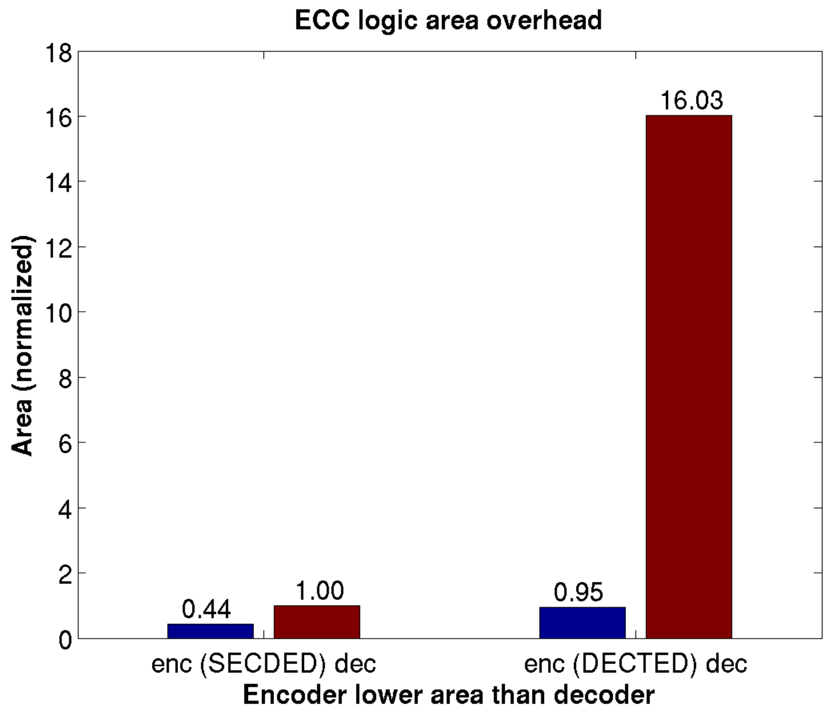
Table 3.1: ECC encoder/decoder latency and area overheads in IBM 90nm bulk CMOS process with 64-bit data word, τ_{FO4}=39 ps, and λ=40 nm [53]

ECC unit	Hsiao SECDED Latency (ns)	Hsiao SECDED Area (μm^2)	DECTED Latency (ns)	DECTED Area (μm^2)
Encoder	0.56	657	1.03	1432
Decoder	1.03	1508	2.4	24174

Table 3.2: ECC encoder/decoder latency and area overheads estimated by scaling in ST 65nm bulk CMOS process with 64-bit data word, τ_{FO4}=31 ps, and λ=30 nm [53]

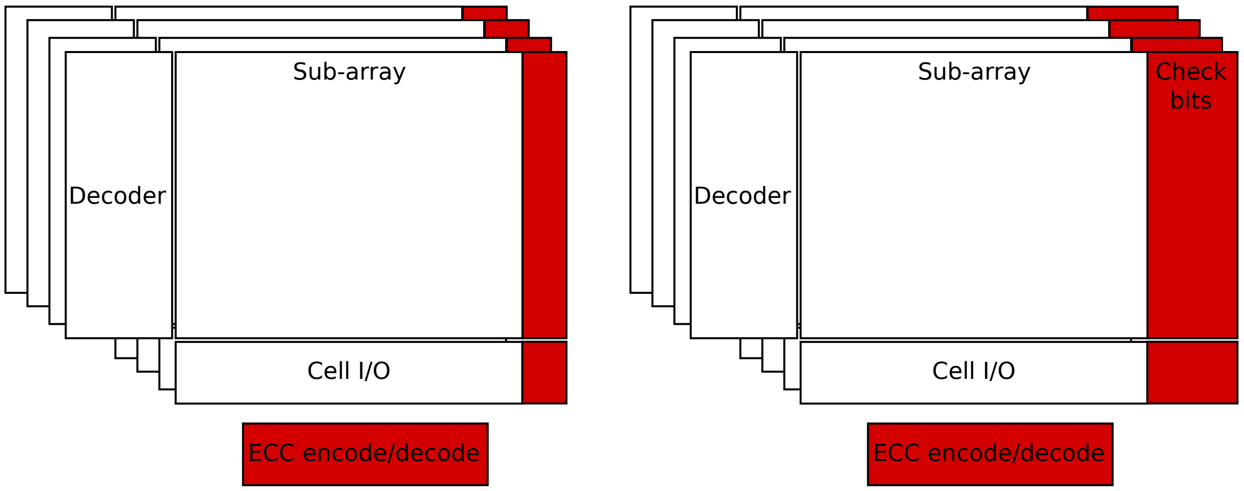


(a) Latency overheads



(b) Area overheads

Figure 3.5: Normalized overheads of ECC encoder/decoder: data from Table 3.1 [53]



(a) (72, 64, 4) SECDED ECC: 12.5% check bit overhead **(b)** (79, 64, 6) DECTED ECC: 23.4% check bit overhead

Figure 3.6: Illustration of area overheads. ECC encode/decode logic has fixed area regardless of array size, whereas parity/check bit storage has area overhead proportional to the array capacity and thus does not diminish in importance for a larger memory. Fixed area overheads and their relative area difference between SECDED and DECTED logic not to scale (see Figure 3.5).

tectural impact of these long latencies by ensuring that the latency hit is suffered only once per failure, after which the word is treated as unreliable, so it will not be corrected via long-latency inline multi-bit ECC again. Previously proposed low-overhead multi-bit soft error protection techniques perform poorly with randomly distributed hard errors because they assume either clustering error patterns, as in the case of block coding [43], or low error rates [35].

The improvement in reliability gained by stronger error-correction capability is tempered by an overhead of latency and area as shown in Tables 3.1 & 3.2. Scaling the area numbers generated in a 90nm technology is done by multiplying the area by the ratio of the process parameter λ, which is defined as half the minimum mask dimension or “feature size”, in this case the drawn length of a transistor channel [75]. Similarly, scaling the latency numbers is done by multiplying the delay by the ratio of the process performance parameter τ_{FO4}, or the delay of a fanout-of-4 inverter.

$$Area_{65nm} = Area_{90nm} * \frac{\lambda^2_{65nm}}{\lambda^2_{90nm}} \tag{3.2}$$

$$Latency_{65nm} = Latency_{90nm} * \frac{\tau_{FO4(65nm)}}{\tau_{FO4(90nm)}} \tag{3.3}$$

3.4.3 Vertical parity

Researchers have previously proposed a low-overhead multi-bit ECC technique which uses two-dimensional (2D) error coding (i.e., row-wise and column-wise) to provide both low VLSI overheads and high error coverage [35].

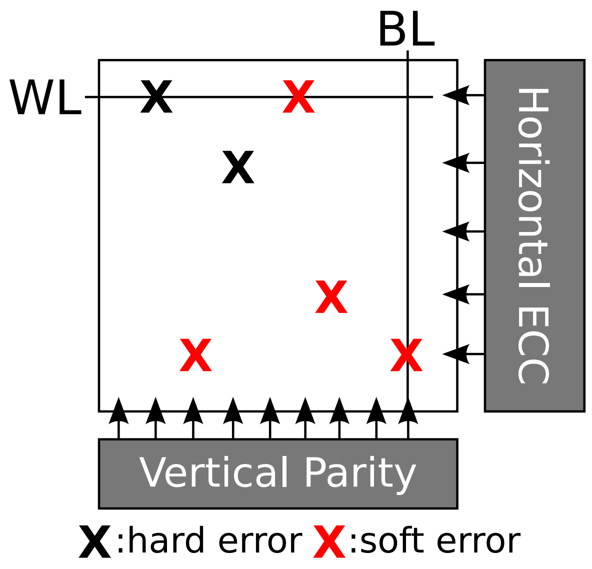


Figure 3.7: 2D coding / vertical parity illustration. When the horizontal ECC detects an uncorrectable error, the controller initiates a vertical parity recovery process. Note that the combination of a one-bit hard error and one-bit soft error can both be corrected, allowing for use of a word with a manufacture-time failure or marginal operation condition. [35]

Applying 2D error coding techniques to the on-die embedded memory system enables fast common-case error-free operation. The key innovation in 2D error coding is the combination of lightweight horizontal per-word error coding with vertical column-wise error coding. The horizontal and vertical coding can either be error detecting codes or error correcting codes. In our testchip implementation (see Chapter 6), the horizontal code is SECDED ECC, while the vertical code is a simple parity code which can only detect a single bit error (see Figure 3.7). However, when used in combination, the horizontal SECDED ECC and vertical parity form a single strong multi-bit ECC applied to the entire array. The vertical codes enable correction of multi-bit errors along rows up to and including entire row failures. The vertical parity rows are interleaved to increase coverage along the columns, with little impact on area, power, or delay.

Vertical parity update Before every write to a word in the array, that word must be read out to update the vertical parity: the new value of each bit of the vertical parity word is a 3-input XOR with the data read out of the array, the data written into the array, and the old value of the vertical parity. The vertical update logic can be pipelined in parallel with normal memory operations to mask this latency overhead. If a more complex code than vertical parity were used, instead of a simple read-before-write operation, multiple reads would have to be performed corresponding to every message word bit that goes into the parity computation which forms the vertical code word which is updated. This architectural overhead is what limits us to vertical parity in practice.

Vertical parity correction When the horizontal code detects an uncorrectable error, the controller initiates a vertical parity recovery process. The controller reads all data rows that share a vertical parity row with the erroneous data row and XORs their values together. The result of the XOR operation is the original value of the erroneous row, which is then written back to the proper location.

3.5 Redundancy

Redundancy on the row, column, or block level has long been employed in SRAM design to mitigate the yield loss due to a single bad cell or peripheral logic component to an entire array [24]. However, using hardware redundancy alone to address yield loss due to variability has limited efficacy since an entire row or column must be re-allocated with a redundant spare if even a single bit in that row or column fails. The advantage of using redundancy is that overhead in power and latency during normal operation is minimal. Redundancy replacement of a faulty memory cell can be accomplished at manufacture-time or at run-time, though most implementations employ fuses to store this configuration which are only configurable at manufacture-time. The replacement procedure consists of three phases: the testing phase, where faulty cells are located; the analysis phase, in which the rows and columns to be replaced are selected; and the repair phase, in which the configuration is written into fuses or some other non-volatile storage elements so that the replacement is preserved across power cycles. The area overhead therefore consists of the spare rows and columns of memory, the non-volatile storage elements and muxes for redirection of faulty memory rows and columns to spare rows and columns, and optionally BIST logic for testing to find faulty cells. In general, spare replacement using rows only, or columns only, is less effective in preventing yield loss, but simpler in implementation, particularly in the analysis phase [34].

Redundancy at higher levels has been used to meet high reliability requirements. Modular redundancy is used to guarantee reliability by operating multiple versions of the same block in parallel. In DMR, two redundant versions of a system operate in lock-step, declaring an error if they disagree; in contrast, in TMR, three redundant versions of a system vote to decide the correct output. These array-level or component-level redundancy schemes incur larger area penalties than redundancy on the row or column level, increasing cost.

3.6 Peripheral logic fault detection

At present, industrial designs focus most of their effort in error control on the cell array. Often, this decision is justified on the basis of fault likelihood; due to the large number of minimum-sized devices in the cell arrays, they are highly vulnerable to both failures due to random mismatch variability and failures due to energetic particle strikes. However, faults are not inconceivable in the peripheral logic outside the cell array, and thus in high-reliability applications such as those memories designed for the automotive, industrial, or medical sectors, it is no longer feasible to design the address decode path and other peripheral logic without fault detection.

We previously discussed error manifestations in the peripheral logic in Section 2.3.2. Traditional methods of error control applied to the cell array are generally unsuitable for peripheral logic failures since coverage will be poor if the data is not corrupted. For example, a failure in the address decode logic could go undetected on a read where the wrong data word is fetched, despite ECC protection, because the wrong data word will still be correctly encoded; rather than a single-bit failure, the entire word has failed. Instead, address decode failures can be detected through the addition of dynamic logic fault detection units to the address decode path and ROM address bits to each wordline, at the cost of area [82] (Figure 3.8). A comparison of the output ROM address bits to the input address bits will indicate definitively whether or not an address decode failure occurred. We refer to this technique interchangeably as a wordline encoder or as ROM address bits (ROM-ADDR).

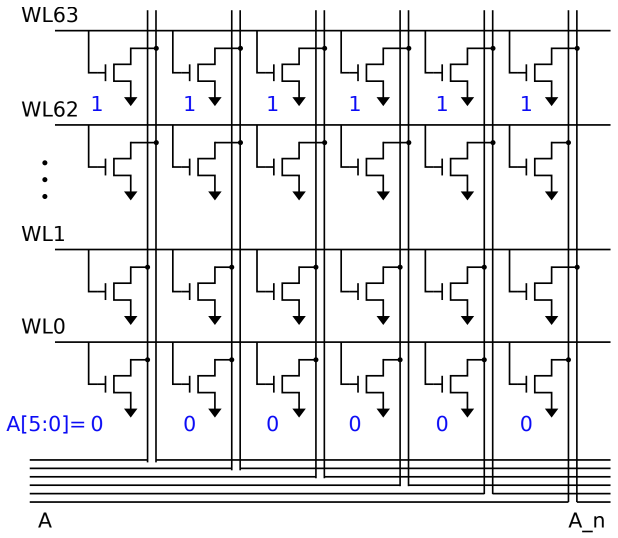


Figure 3.8: WL encoder enables detection of most common address decoder faults [82].

The wordline encoder is capable of detecting address path failure when combined with other error checker (EC) circuits such as a wordline NOR checker, write enable NOR checker, bitline and global bitline pre-charge checker, and replica columns for write and read checking. By storing both the true and complement forms of the address bits, aliasing due to multi-word failures is prevented. We propose a novel design to detect similar failure modes with better area efficiency for large caches in Section 5.2.

3.7 Summary

In this chapter we have discussed a number of conventional reliability techniques for memory. Read/write bias assists are commonly used to maintain high functional noise margins for the cell as devices and supply voltage shrink due to scaling and system performance pressures. The 8T cell is now widely used, especially in memories which place a higher emphasis on performance and multi-port access than density or cost. It also offers a respite from the conflicting read stability and writeability requirements, when used without column multiplexing. Up-sizing the cell similarly reduces failure rates at the cost of density and power. Error control coding has become routine in modern designs, although multiple-bit error correcting codes are still reserved for higher levels of the cache hierarchy due to their latency requirements. Redundancy has long been used as a yield-improvement technique. Finally, peripheral logic fault detection is a useful technique for safety-critical designs to protect against the risk of energetic particle strikes and other rare failure events.

In the following chapters we will present our novel reliability techniques, divided up into those focusing on the cell array (Chapter 4) and those focusing on the peripheral circuitry (Chapter 5).

Chapter 4

Cell array protection

"…memory locations…are just
wires turned sideways in time."

W. Daniel Hillis [25]

Following this metaphor, it should be possible to apply selected techniques for ensuring error-free communications over noisy channels to ensure reliable memory system operation.

This chapter explores a novel way of extending the widely-used communications technique of erasure coding to memory, and the overheads and benefits that can be realized. First, we discuss the theory of error control coding and erasure coding. Next, we introduce the circuit techniques which provide the erasure hints to the erasure coding logic, providing the means of doubling the correction strength of the code. These techniques include read latency failure detection for full-swing arrays via a bitline XNOR, and small-swing arrays using a ternary-output sense amplifier (TOSA).

Since a memory consists of a very large number of identical storage cells, each holding one bit of data, memory designers tend to be very conservative in their choices to minimize the individual cell failure rate. For a reasonably-sized[1] L2 cache of 8 Mb, a cell failure rate of 0.000057%[2] would result in an overall array failure rate of 99.2% without redundancy replace-

[1]In the 65nm generation, L2 caches range from 2 Mb [71] to 32 Mb [61].

[2]This corresponds to the 5σ point of the probability distribution, indicating that the cell is robust out to 5 standard

ment. While only 1 in 1.7 million bits will be in error given that cell failure rate, in an array of over 8 million bits, the odds that a failure will occur somewhere in the array are very high. Thus, in large arrays multiple means of dealing with erroneous or marginal cells are necessary. The techniques discussed in Chapter 3 are becoming necessary for continued scaling of SRAM; the erasure coding techniques we present in this chapter are largely compatible with the existing techniques and confer additional benefits.

4.1 Erasure coding theory

In order to correct an error in a data word, we must know both the location of the erroneous bit and the correct value. When a *random error* occurs in a data word, we know neither and must rely on the ECC to find both the location and correct value. However, an *erasure* can also occur, where we know the location of the erroneous bit, but do not know the correct value. Therefore, the ECC must only determine the correct value of the bit. In the context of communications, this concept of an erasure can be visualized as shown in Figure 4.1; if the signal's slew rate is poor or the sampling interval is timed at a point when the signal is still transitioning from one rail to the other, the digital value cannot be reliably sensed and an erasure is detected.

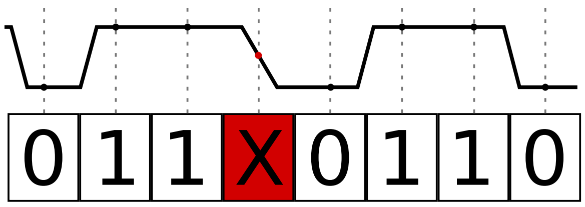


Figure 4.1: Illustration of an erasure (denoted by X) in a serial data stream. The dotted vertical lines indicate sampling intervals; when the signal's slew rate is poor or the sampling interval is timed at a point when the signal is still transitioning from one rail to the other, the digital value cannot be reliably sensed and an erasure is detected.

If we have an ECC with a minimum Hamming distance of d_{min}, the number of correctable random errors (e) and correctable erasures (f) must maintain the relationship: $2{\cdot}e+f < d_{min}$ [48]. Thus, **the number of correctable errors can be doubled using erasures**. If we wish to add

deviations from the mean.

Code	Correct		Detect
	Erasures	Random errors	Random errors
TED	0	0	3
TEC	3	0	0
SECDED	0	1	+1
SECTED	1	0	+2
DECTED	**2**	0	**+1**

Table 4.1: Potential implementations of a Hamming code with minimum distance 4. The SECDED implementation is commonly used, but with a source of erasure hints, two bits flagged as erasures can be corrected and an additional random bit error detected.

additional error detection capability beyond that which is correctable, the formulation with additionally detectable errors (g) is: $2{\cdot}e + f + g < d_{min}$.

The commonly used Bose-Chaudhuri-Hocquenghem (BCH) SECDED code has a $d_{min} = 4$ which is used to correct a single random error and detect an additional error ($e = 1, f = 0, g = 1 \rightarrow 2 * 1 + 0 + 1 = 3 < 4$). However, this discounts the possibility of erasures, which could enhance the correction capabilities of a $d_{min} = 4$ code. Table 4.1 shows the possible erasure and random error correction/detection capabilities of $d_{min} = 4$ codes. Of note is the ($e = 1, f = 1, d = 0$) code which can correct two erroneous bit in a data word, as long as one of them is an erasure, but eschews any additional detection capability. This code could be useful in designs that never expect to see more than two erroneous bits in a word. Also of interest is the ($e = 0, f = 2, d = 1$) code which can correct two erroneous bits as long as both of them are erasures, and it maintains an additional bit of detection. This code could be useful if we expect a small number of random soft errors and had an alternate method of correcting them as specified in Section 4.3.

4.2 Erasure detection

In a memory array, a bit is considered *erased* when it cannot reliably be read, written, and/or hold its data value. Detection of an erased bit can occur *offline* before the data access, thus providing prior knowledge of the erasure location, or at the time of the data access, thus requiring *runtime* determination of the erasure location.

4.3 Erasure correction

Once one or more erasures are detected, whether offline or at runtime, correction is a simple matter provided they do not exceed the code's minimum Hamming distance d_{min}. The general procedure for simultaneous correction of erasures and random errors is [48]:

1. Fill all erasure locations with 0's and decode as normal.
2. Fill all erasure locations with 1's and decode as normal.
3. Pick the corrected codeword with the smallest number of errors corrected as the correct codeword. Provided $2{\cdot}e + f < d_{min}$, the codeword will be correct.

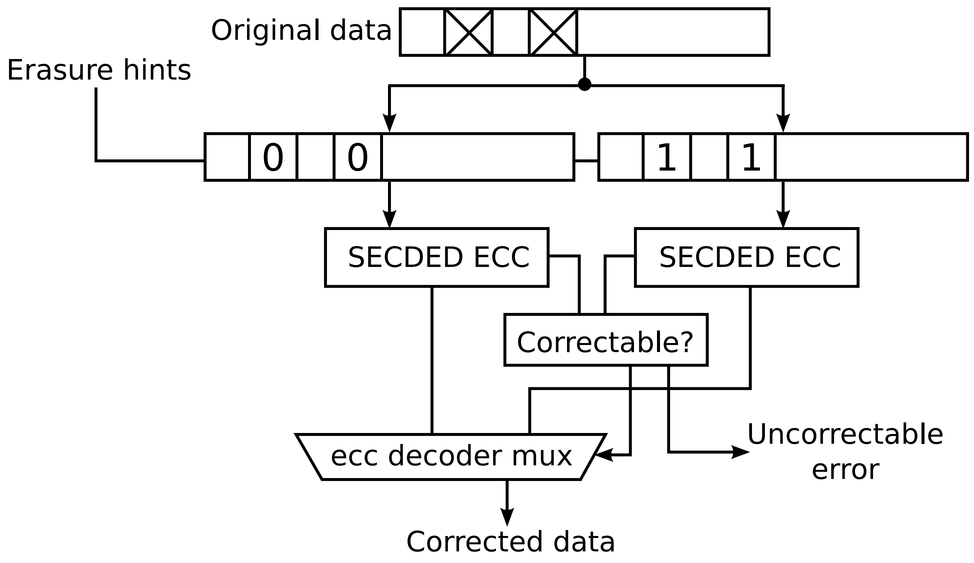


Figure 4.2: Correction mechanism with erasures (number of random errors $e = 0$, number of erasures $f = 2$, number of additional detectable errors $g = 1$) [37].

Figure 4.2 shows the hardware block diagram of the correction mechanism. Any errors occurring in erasure locations will appear as 0 & 2, 1 & 1, or 2 & 0 bit errors after masking the erasure locations to 00 and 11. These errors will be corrected by at least one of the SECDED decoders. If an additional error occurs in a non-erasure location, the scenarios will be changed to 1 & 3, 2 & 2 errors, or 3 & 1 errors. For 1 & 3 and 3 & 1 error patterns, both decoders will try to correct in non-erasure locations regardless of correctness. For 2 & 2 error pattern, neither decoder can correct the error. These scenarios are reported as a rare uncorrectable error event (e.g., soft error). In this way, this mechanism effectively works as a fast DECTED code; the latency of this double erasure correcting code is only slightly higher than that of a conventional SECDED code.

Figure 4.3 illustrates conceptually how a two-bit erasure pattern can reveal the correct codeword on the left side of the diagram, while the right side of the diagram shows the mechanical process of masking the erasures and muxing the properly corrected word to the corrected data output. Figure 4.4 illustrates the complementary scenario where the erasure locations' corrected values are not both 0 or 1, thus the output of each SECDED is a 1-bit corrected word.

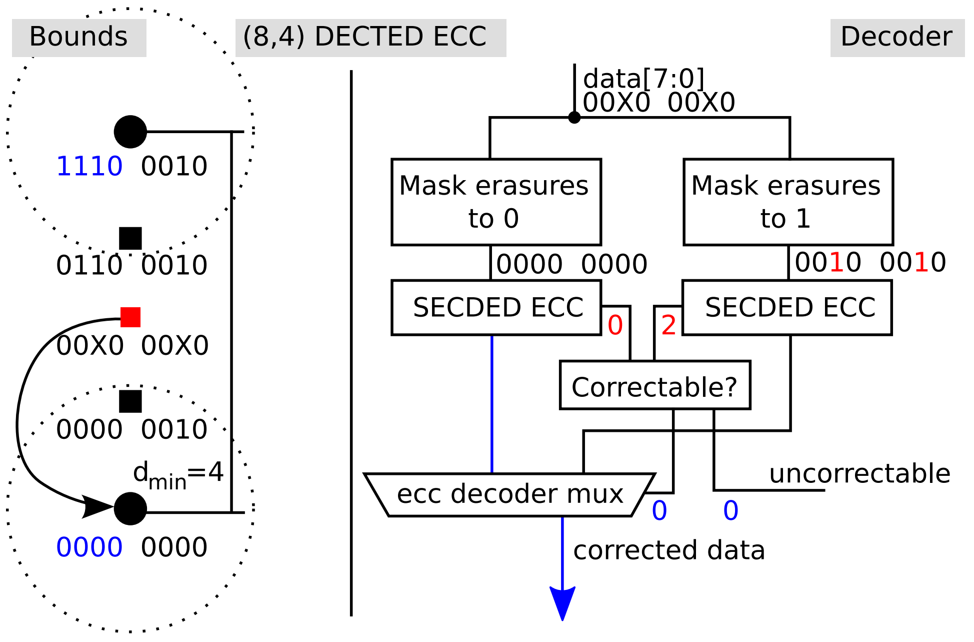


Figure 4.3: Correction of a two-bit erasure pattern: with the erasures masked to 0, the correct codeword is revealed (0 bits in error are detected by the SECDED ECC unit); on the other hand, with the erasures masked to 1, 2 bits in error are detected by the SECDED unit and thus the 0-masked word is muxed to the corrected data output.

4.4 Offline erasure detection

For offline erasure detection, the memory testing community has already developed an extensive library of methods for detecting faulty storage cells, since this information is the basis for redun-

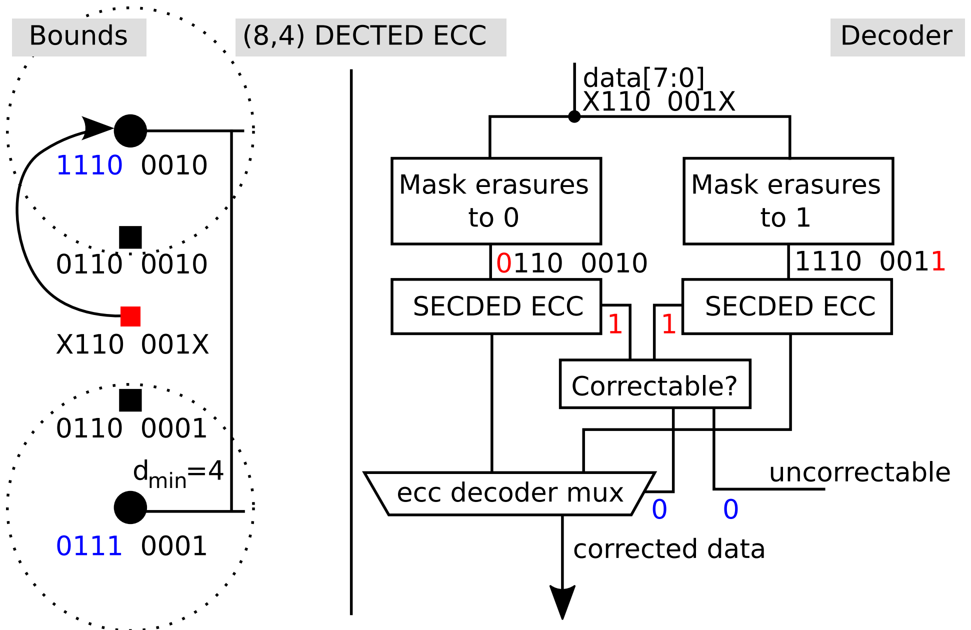


Figure 4.4: Correction of a two-bit erasure pattern: with the erasures masked to 0, the SECDED corrects a single-bit error. The same occurs with the erasure masked to 1, and so either output is the properly corrected data word.

dancy assignment. Redundant memory elements can be assigned at manufacture-time and/or in the field during power-on self test (POST) [13]. These tests use tailored test vectors that target specific error types, and in some cases, special test circuits that can find marginal cells. For example, special testing-only weakened write circuits can be used to find cells that are marginal in the write latency failure mode [2, 46].

Offline erasure detection requires storage of the erasure location and subsequent read out of the erasure map on every read access. The erasure map can be stored on a per word basis, in which case we must allocate a few additional bits of storage per word for the erasure pointer(s) ($log_2(n)$ bits for an n-bit data word for each erasure). Alternately, the erasure map can be stored globally in a searchable memory structure (*e.g.*, a CAM that uses the memory address as the key and the erasure vector as the data) which is accessed on every read to determine if the read memory address contains any erasures. Which of these methods is optimal depends on the memory parameters and error rate.

4.5 Runtime erasure detection

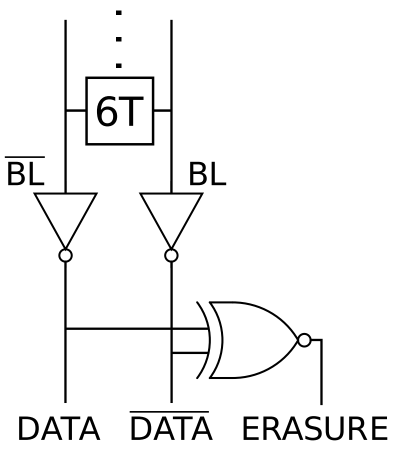
Four cell failure types were introduced in Section 2.2; the ease of detection of these failure types at runtime depends on whether the failure has an extrinsic manifestation outside of the cell and on whether the time of the failure can be pinpointed.

The simplest cell failure type to detect at runtime is an access failure, also known as a read latency failure, since it can be detected outside of the accessed cell (at the bitlines) and occurs at the time of the read access. The other three failure types do not necessarily immediately manifest at the time of occurrence, and thus are best detected through offline testing. A data retention fault, which occurs while a cell is idle, is known to be particularly difficult to detect [80]. An XNOR gate can be used to detect a read latency failure (Figure 4.5) for a full-swing bitline design. In this scheme, we treat each of the differential bitlines as independent single-ended bitlines which swing full rail (from V_{DD} to V_{SS}). On a proper read, one of the bitlines will be pulled to V_{SS} by the cell, and the other will remain precharged at V_{DD}. However, if the accessed cell has insufficient read current (*e.g.*, due to device variability) neither bitline will fall below the logic threshold of the sensing inverters, and both bitlines will remain high (output of both inverters will be low). Alternately, if there is too much leakage on the bitline due to the unaccessed "off" cells in the column, both bitlines will be pulled to V_{SS} (one by the accessed cell, and the other by the leaking cells), which is another detectable condition distinct from normal operation.

This technique described is suitable for full-swing bitlines, but an alternative design is required for small-swing bitlines. To determine whether a read latency failure has occurred at runtime, we employ a pair of sense amplifiers to determine whether the bitline differential is large enough to be an indication of a reliable read or whether it's so small that the output value cannot be determined reliably, in which case an erasure is declared.

4.5.1 Ternary-output sense amplifier

In this section, we demonstrate a method of runtime erasure detection which improves upon the direct bitline XNOR technique which is only suitable for paired full-swing bitlines. Recall



(a) Circuit implementation

$DATA$	$\overline{DATA}$	$ERASURE$	$Meaning$
0	0	1	Read latency failure
0	1	0	Value = 0
1	0	0	Value = 1
1	1	1	Access failure due to leakage

(b) Description of behavior

Figure 4.5: Runtime erasure detection circuit for access/read latency failures [37]. This design is suitable for full-swing bitlines.

that the read latency failure mode occurs when the single on cell's read current minus the off cells' leakage current is insufficient to create enough bitline differential to exceed the sense amplifier's input-referred offset. In an equation, this requirement is shown in Inequality 4.1, where N is the number of cells in a column, $T_{WL \rightarrow SAE}$ is the time between wordline assertion and sense amplifier enable assertion (the time the 6T cell has to produce a voltage differential on the bitlines), and C_{BL} is the bitline capacitance.

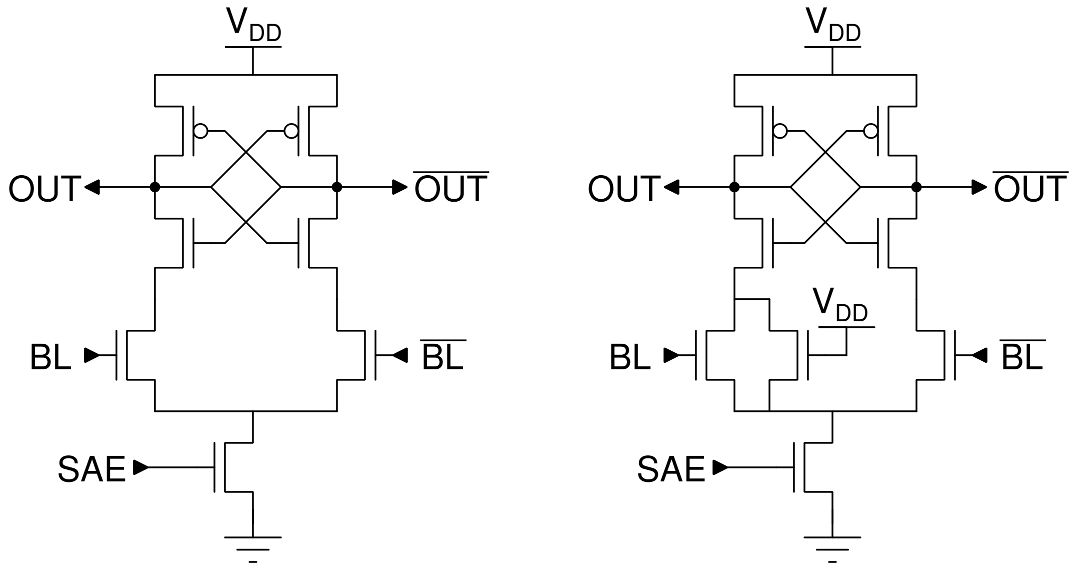
$$|V_{SAoffset}| < \frac{(ION - (N-1) * IOFF) * T_{WL \rightarrow SAE}}{C_{BL}} \tag{4.1}$$

Current solutions to this problem mainly involve minimizing the sense amplifier's input-referred offset through sizing or circuit techniques to improve the read current, but in some high-reliability applications, detection of the failure mode may be almost as valuable as correction.

The concept of a ternary sense amplifier has been discussed in the literature [78], but we believe our implementation of it to be unique in its low overhead and broad applicability for reduction of read latency failures when used in conjunction with an erasure coding scheme for error correction.

Building a ternary-output sense amplifier (TOSA) involves a modification of an existing sense amplifier design which outputs 0 or 1 for the bitline data with one that outputs an erasure signal as well, so that there are three possibilities for the output: 0, 1, or erasure (see Figure 4.11). At the circuit level, this is actually two separate lines, data and erasure. One simple way to build this concept using a small-swing differential sense amplifier as a building block is to allocate two sense amps, each with their input-referred offset skewed in opposite directions by a nominal voltage V_{skew}, so that their outputs will only agree if the bitline differential exceeds V_{skew} (in the nominal case absent variability). If they disagree, the erasure signal will be high, indicating that the bitline differential is not large enough to be considered reliable. This skewing can be accomplished by means of parallel device assist implementation (PDAI) for current-latching/"StrongARM" sense amplifiers (CLSA) (see Figure 4.6) or capacitive resist implementation (CRI) for voltage-latching/"latch-style" sense amplifiers (VLSA) (see Figure 4.7) [5, 38].

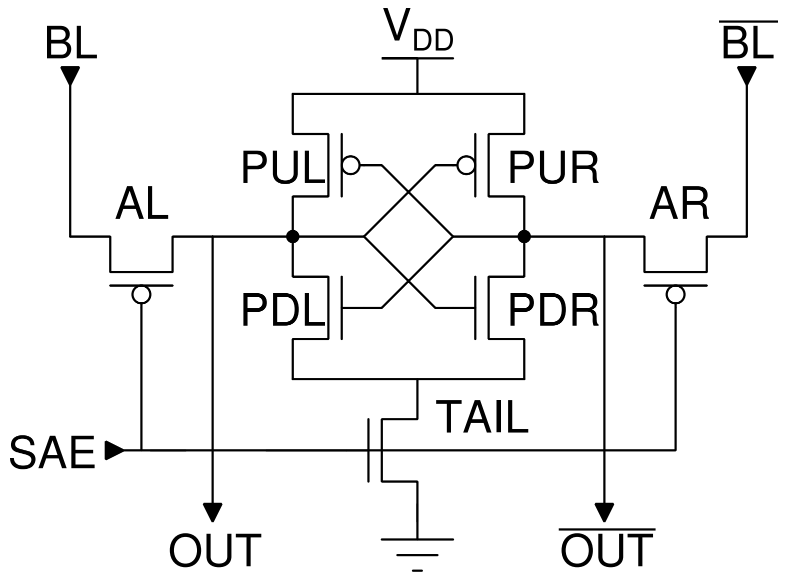
In the literature, this technique was used with kick structures on both sides of the sense amplifier which could be turned on or off, depending on which direction the device mismatch pushed the input-referred offset. In our incarnation, we allocated two sense amps with their inputs opposite, so that the one-sided kick structure acts in a roughly equal amount in opposite directions. We've simulated this design in a 65nm bulk CMOS process using voltage-latching sense amplifier (VLSA) and a one-sided CRI "kick", targeting an erasure range of 50 mV at nominal V_{DD} of 1.2 V; i.e., if the cell fails to develop 50 mV of bitline differential, the data from that column is considered to be an erasure by the ECC. Previous research has proposed using redundant sense amplifiers but to the best of our knowledge a multiple-sense amp design involving soft information to improve SRAM reliability has not been proposed [73].



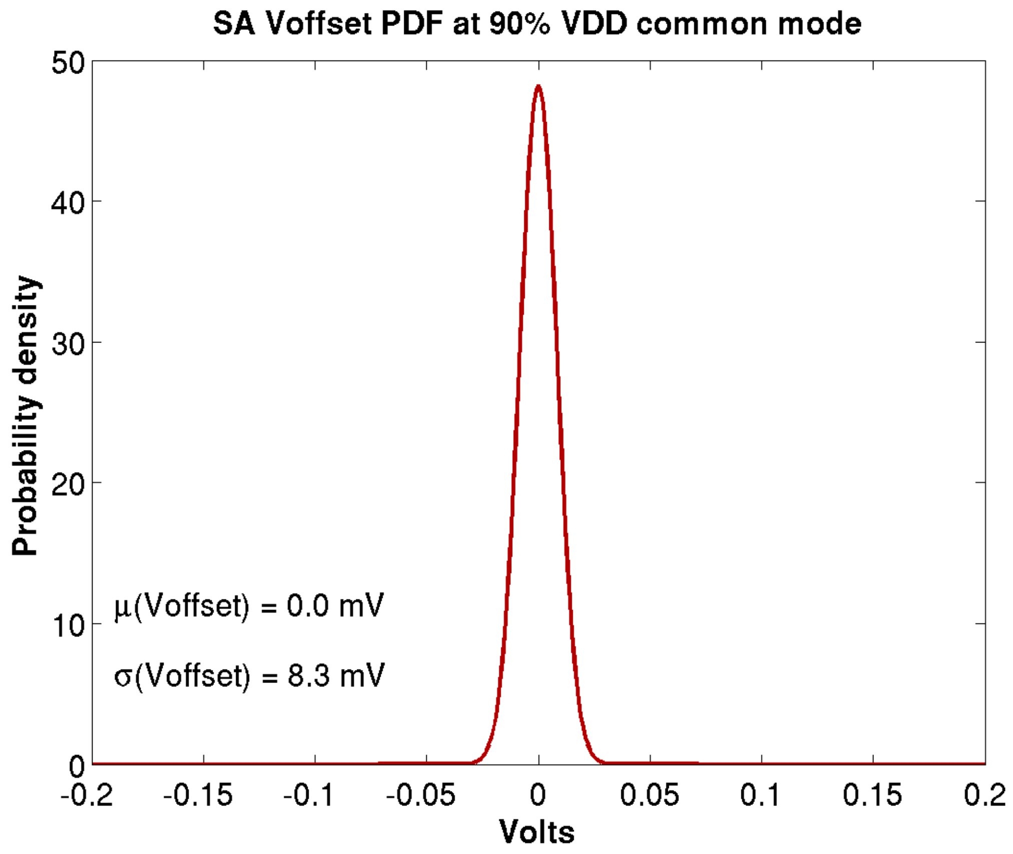
(a) CLSA **(b)** CLSA with PDAI kick structure

Figure 4.6: Current-latching sense amplifier and PDAI kick structure [5].

The device sizes shown in Figure 4.10 were chosen to minimize input-referred offset while maintaining acceptable area for either 2x or 4x column mux pitch in 65nm. A low propagation delay was also sought, as well as a high input common mode rejection ratio given an input common mode range from 0.7 V to 1.2 V. Over a 141,000-point MC analysis for mismatch variation, for the VLSA with CRI kick, $\mu(V_{offset}) = 34.3mV, \sigma(V_{offset}) = 8.4mV$ at 90% V_{DD} input

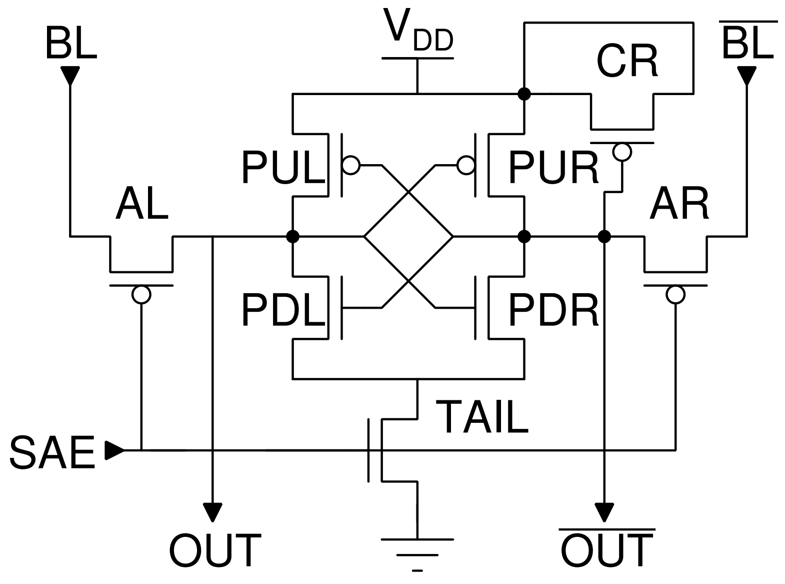


(a) VLSA

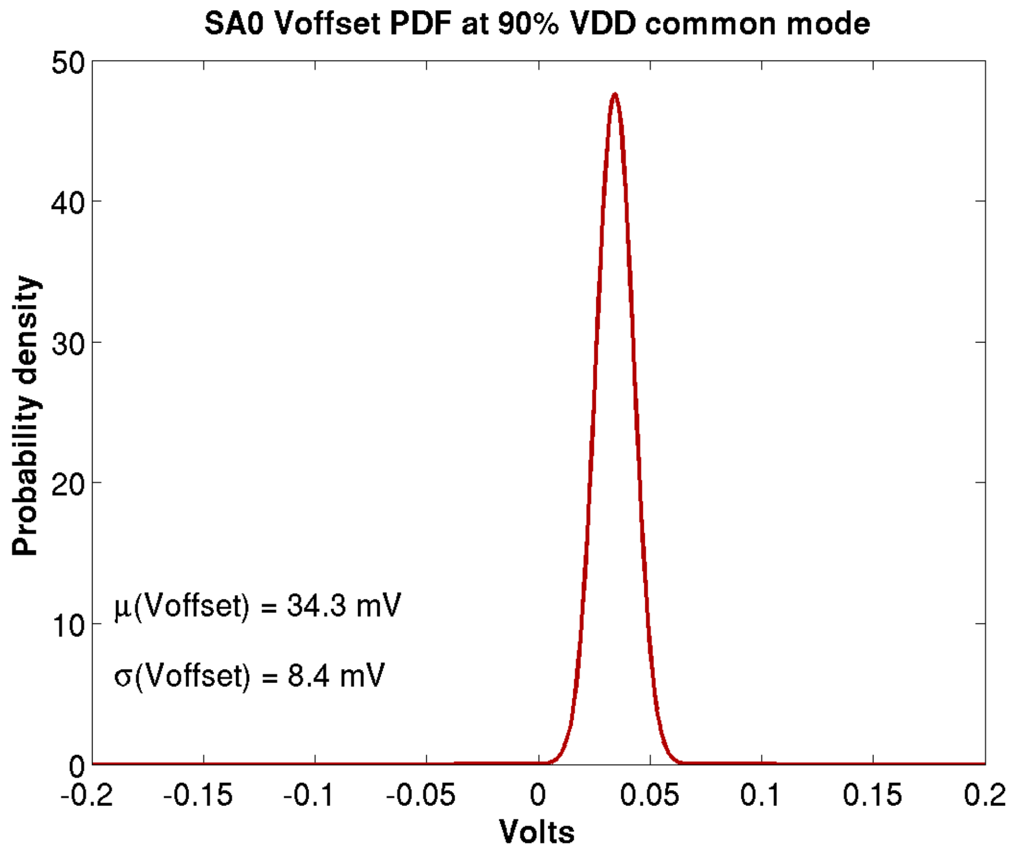


(b) VLSA input-referred offset PDF. V_{DD}=1.08, $\mu(V_{offset}) = 0V, \sigma(V_{offset}) = 8mV$ at 90% V_{DD} input common mode.

Figure 4.7: Voltage-latching sense amplifier and offset PDF.



(a) VLSA with CRI kick structure



(b) VLSA with CRI: input-referred offset PDF. V_{DD}=1.08, $\mu(V_{offset}) = 34mV, \sigma(V_{offset}) = 8mV$ at 90% V_{DD} input common mode.

Figure 4.8: Voltage-latching sense amplifier with CRI kick structure and offset PDF.

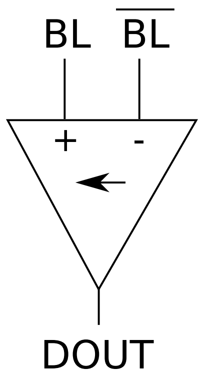


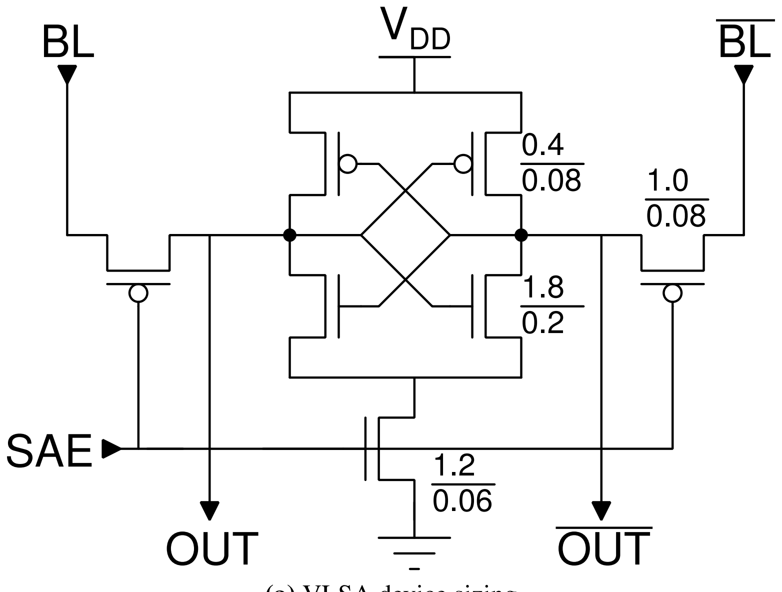
Figure 4.9: Symbolic representation of VLSA with CRI kick in same configuration as Figure 4.8.

common mode (V_{DD}=1.08 V, see Figure 4.8). Since the variation in input-referred offset is primarily a result of mismatch between the nominally identical sides of the sense amplifier structure, the results from a MC analysis for process and mismatch variation should show nearly identical $\sigma(V_{offset})$. This was found to be the case: over a 14,000-point MC analysis for process and mismatch variation, for the VLSA with CRI kick, $\mu(V_{offset}) = 34.2mV, \sigma(V_{offset}) = 8.5mV$ at 90% V_{DD} common mode. Information on the methodology of our simulation infrastructure is provided in Appendix A.

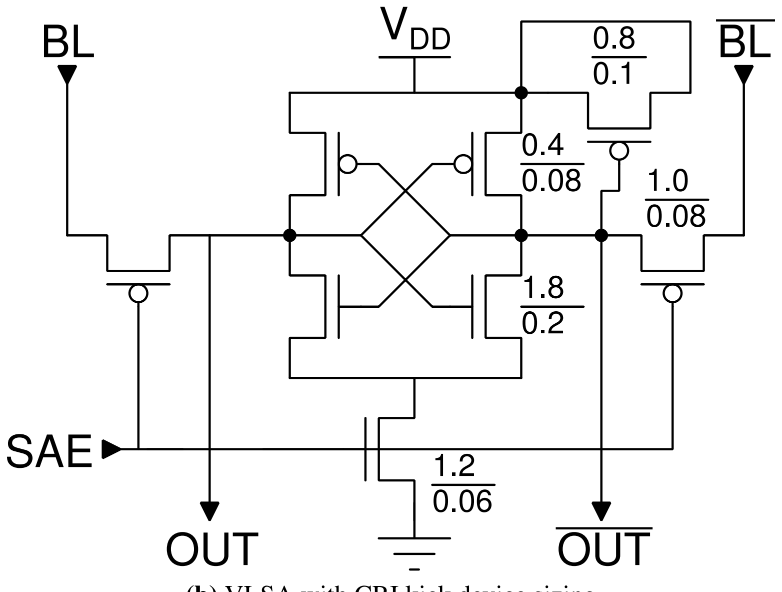
Of course, since any sense amplifier has some variation in its input-referred offset, this erasure range will have per-column variation. We've shown input-referred offset voltage of each of the two sense amplifiers at one design point (see Figure 4.14).

4.5.2 Overhead

Since the design is based on only a slight modification to a typical sense amplifier, it will double the sense amp area, with the addition of an XNOR gate on the output; amortized over a 128-cell column, this represents approximately 6% increased local array area overhead. Also, the added sense amplifier increases the capacitance on the bitline, which reduces bitline differential voltage developed by the cell, which may impact latency depending on the specific design parameters.



(a) VLSA device sizing



(b) VLSA with CRI kick device sizing

Figure 4.10: Sense amplifier sizing; all measurements in microns.

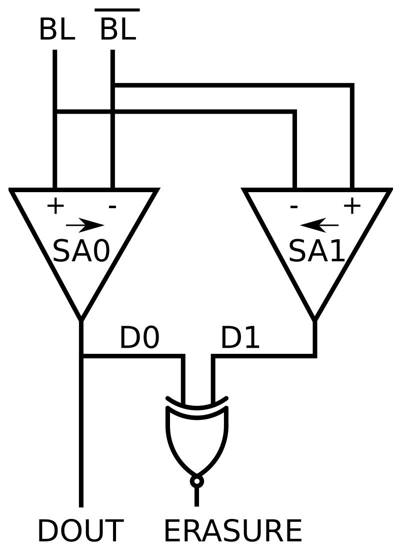


Figure 4.11: Ternary-output sense amplifier (TOSA) block diagram. Each sense amp's input-referred offset is "kicked" away from 0 in opposite directions.

4.5.3 Simulation methodology

Since read latency does not have a minimum threshold at 0 as static noise margins do, we must establish constraints according to a canonical design which is chosen to be roughly similar to published industry designs of the same process generation. At our chosen technology node of 65nm, we establish the following targets: for L2 cache, 8 Mb capacity; for L3 cache, 100 Mb capacity. We disregard the L1 cache space in our analysis since there has been very little headroom for reliability techniques which impact latency at that level of the cache hierarchy.

As is done commonly in industry, we pick a shared subarray size for both the L2/L3 cache targets and use it as a building block for the much larger overall cache size. Our MC analysis is performed at the cell level, and statistics are generated using the assumptions of a subarray target size. The subarray target is 128 rows and 64 bits/word with 4-way column multiplexing (thus 256 bits/row before ECC check bits). This will force the column output to be a small-swing sense amplifier, which is common, although not required, in dense cache designs such as those in L2/L3.

The frequency target is set at 2 GHz, which is comparable with Intel's quad-core Itanium 65nm processor [71], and other processors of that generation. While large caches can pipeline their access, the wordline-to-bitline-to-digital data output interval cannot be pipelined, and thus represents the minimum single-cycle delay. Thus we use the 2 GHz target to define a 500 ps access latency for the 65nm subarray. Simulations show that for a 500 ps access latency, a limit of 300 ps wordline-to-sense amplifier enable delay is appropriate, given the delay of other logic in the critical path such as the row decoder, wordline driver, and sense amplifier output latch.

Using the overall cell failure probability, we can calculate memory yield based on the memory capacity. Furthermore, it is reasonable to assume that run-time reliability would be proportionate to manufacture-time yield as determined by the overall cell failure probability. Cells with poor margins are more likely to fail at run-time, so improvements in parametric yield will also generate cell arrays which have lower likelihood of failures due to marginal cells.

4.5.4 Results

Figures 4.12 & 4.15 illustrate the PDFs of the sense amplifier's input-referred offset compared with the bitline differential developed after 300 ps of wordline assertion by the worst-case (lowest ION) cell in the 128-cell column. The bitline differential is computed by the right-hand-side of Inequality 4.1; in this expression, $T_{WL \to SAE}$ and C_{BL} are optimistically considered constants[3]. I_{ON} and I_{OFF} are fit to distributions and sampled to form the combined distribution of ΔV_{BL}.

The probability of undetectable error when using a conventional sense amplifier (SA) is shown at the X=0 point of Figure 4.13; from calculations of the CDF value it is effectively 0 for both L2 and L3 array sizes. The probability of detectable erasure when using a TOSA is shown at the X=0 point of Figure 4.16; it is 0.0011% for L2 array size and 0.014% for L3 array size. An erasure occurs when the bitline differential developed by the 6T cell is less than the input-referred offset of SA0, but greater than that of SA1. Up to two erasures per word can be corrected using a SECDED code configured with erasure masking logic. Finally, the probability of undetectable error when using a ternary-output sense amplifier is shown at the X=0 point of Figure 4.17; it is effectively 0 for both L2 and L3 array sizes.

[3]We neglect the variability of bitline capacitance mismatch and of read self-timing, a complex system we choose not to model here but which is well-studied in the literature [3, 4, 39, 55].

An error only occurs when the bitline differential is so low and the SA1 input-referred offset is so high that the bitline differential crosses both SA0 and SA1 offsets, leading DOUT to resolve incorrectly and ERASURE to be 0. Note that the CDF in Figure 4.17 is shifted to the right farther than the CDF in Figure 4.13. This indicates that the probability of undetectable error is lower for the TOSA than for the SA; although calculations show effectively 0 error rate for both SA and TOSA designs (though TOSA will have a very small rate of erasures which can be corrected), the noise margin for TOSA is larger than for a conventional SA design, allowing for safer operation through the system's lifetime. Figures 4.18, 4.19 & 4.20 show direct comparisons between a word, L2, and L3-sized array with SECDED ECC and a conventional SA or TOSA which provides erasure hints to the ECC decoder. All three lines represent uncorrectable errors where they cross the Y-axis (or an event overcomes their noise margin), but the blue line for the probability of 3 detected erasures in a word still represents a detectable failure condition, whereas the green and red lines represent undetectable failure conditions. Notice that there is a broad safety band between the blue and red lines – this indicates that at run-time, it is highly likely that read latency failures will be detectable by the TOSA in one or more columns. More importantly, in the unlikely event of undetectable read latency failures in other columns, the detectable failures will signal to the ECC decoder that the word is to be considered uncorrectable so these undetectable failures will not silently corrupt the data.

An additional advantage of the use of TOSA for small-swing designs or bitline XNOR for full-swing designs is that it can be used to aid manufacture-time testing in finding storage cells which have marginal read latency. Some memory designs are built with configurable self-timing for the read latency; this has been used to reduce the rate of read latency failures and improve system yield [4, 55]. When a conventional SA is used, determining the minimum sense amplifier enable timing at which a read latency failure occurs requires a standard test pattern to be performed in which data written in is stored in an error-free memory elsewhere in the system and compared against the data which is read out of the memory. If SECDED ECC is available, one or two of these read latency failures per word can be detected without needing to resort to a standard

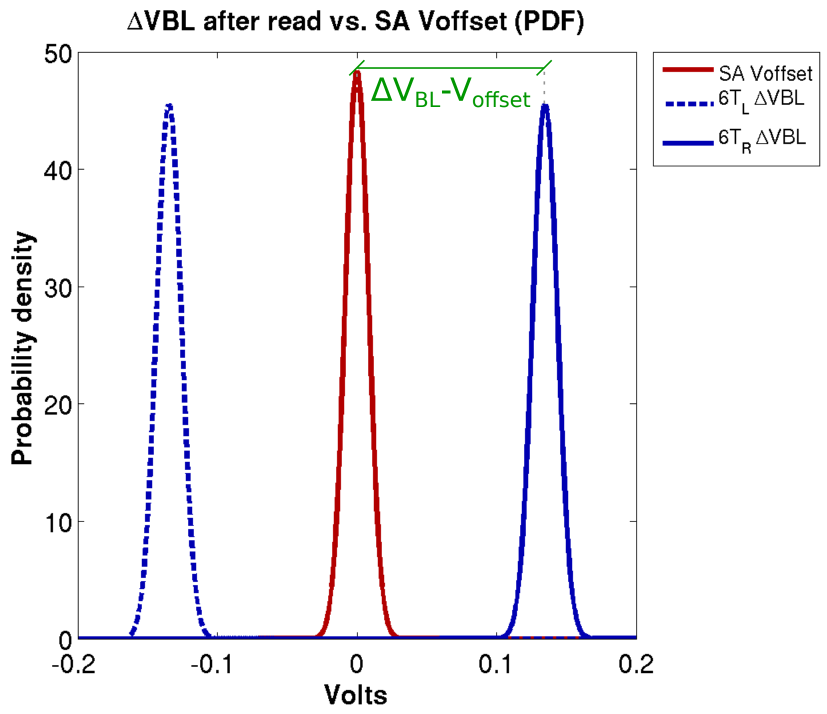


Figure 4.12: PDF of the sense amplifier's input-referred offset compared with the bitline differential developed after 300 ps of wordline assertion by the worst-case (lowest ION) cell in the 128-cell column. V_{DD}=1.08V, T=27° C.

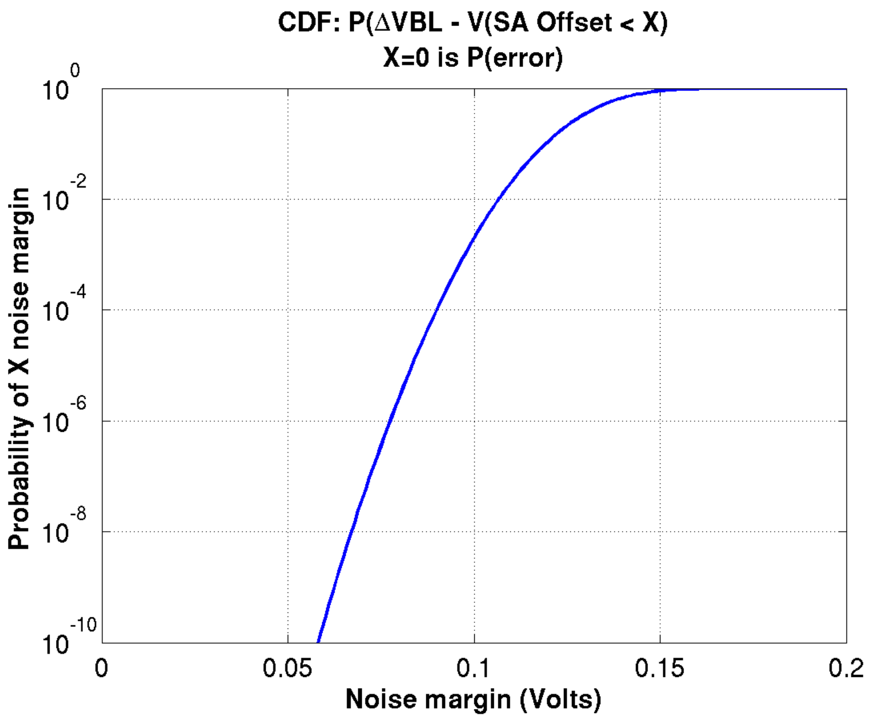


Figure 4.13: CDF of 6T cell bitline swing vs. sense amplifier offset, V_{DD}=1.08V, T=27° C.

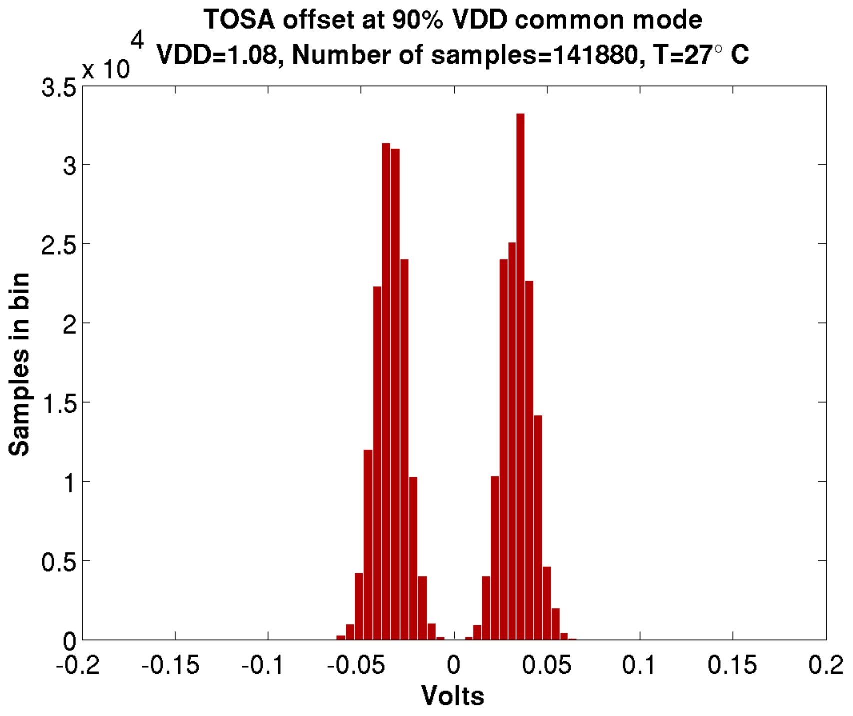


Figure 4.14: Histogram of ternary output sense amplifier offsets, V_{DD}=1.08V, T=27° C.

test pattern, but more failures than that may go undetected. Ideally, the read latency failure rate could be determined transparently during operation rather than requiring a dedicated test to be performed – this is made possible with the use of TOSA. Even without the use of ECC, the TOSA or bitline XNOR can detect any number of read latency failures and declare them erasures; only two of these can be corrected with SECDED ECC and thus higher bit failure counts will result in lost data, but in systems built with support for architectural replay, the data which was erased can be reconstructed after an exception is declared. This operates on the same principle as a similar technique for in situ error detection and correction for sequential elements [7].

4.6 Summary

In this chapter we discussed the extension of the communications technique of erasure coding to SRAM. We presented the design and simulation results of a novel technique for runtime erasure detection, the ternary-output sense amplifier. We evaluated the technique's advantages and

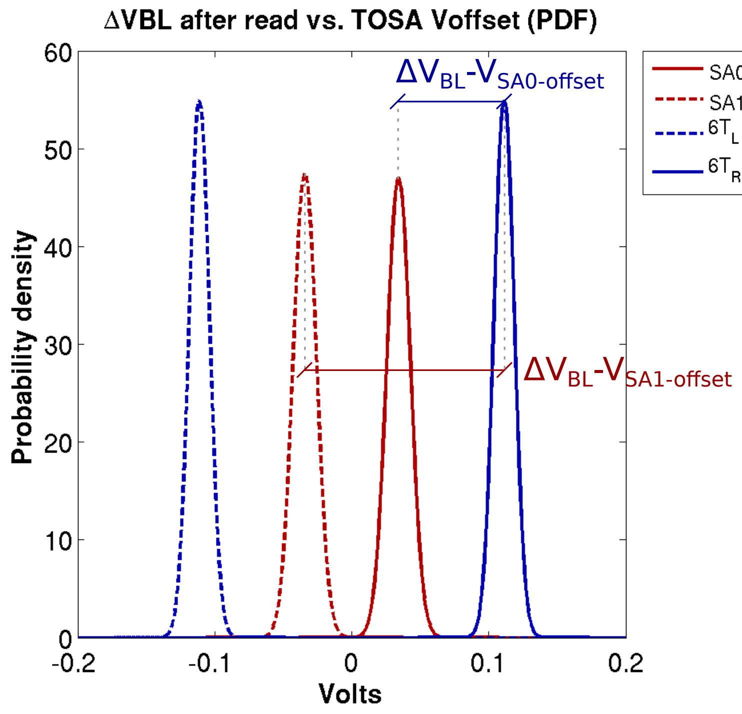


Figure 4.15: PDF of each of the TOSA's input-referred offsets compared with the bitline differential developed after 300 ps of wordline assertion by the worst-case (lowest ION) cell in the 128-cell column. Note that the mean bitline differential is lower than in Figure 4.12 due to the increased capacitance on the bitline from the extra SA. V_{DD}=1.08V, T=27° C.

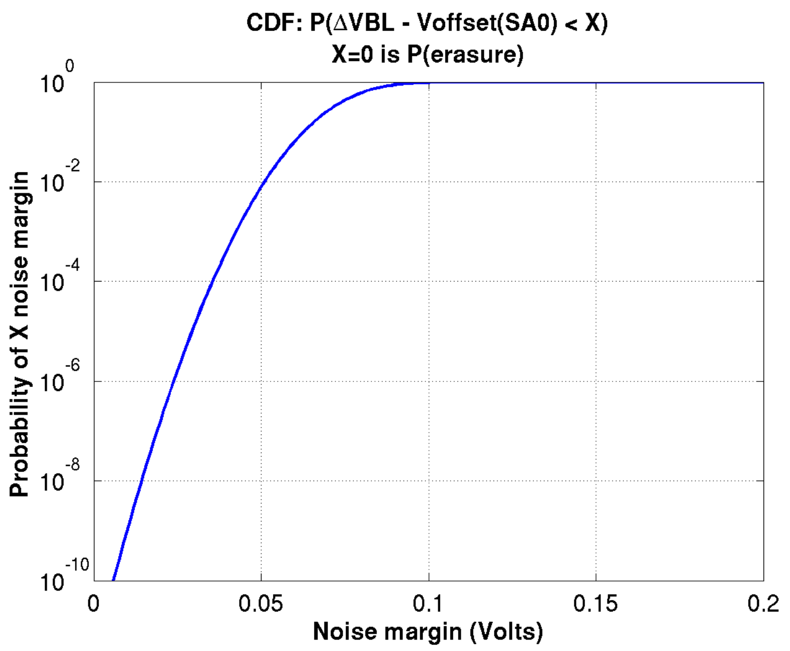


Figure 4.16: CDF of 6T cell bitline swing vs. TOSA offsets. This indicates the likelihood of detectable erasure. V_{DD}=1.08V, T=27° C.

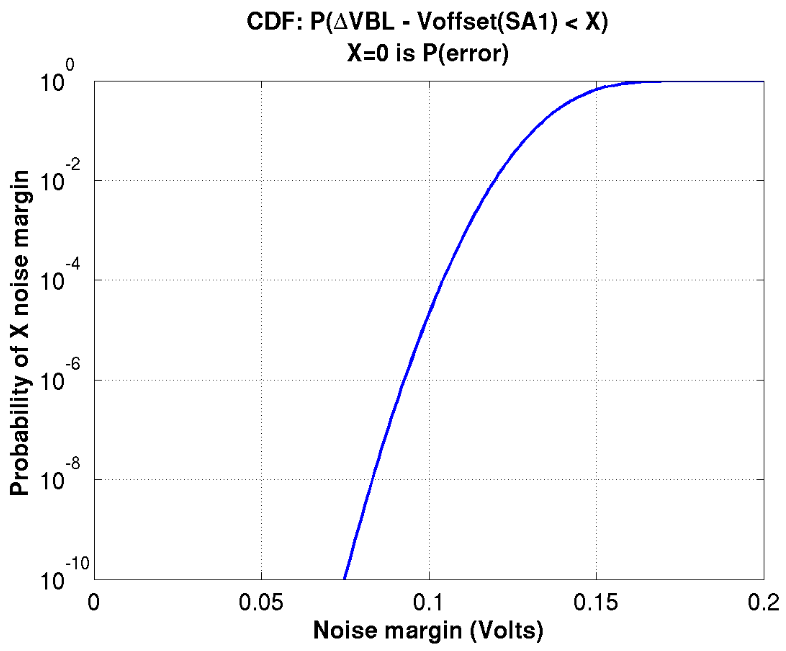


Figure 4.17: CDF of 6T cell bitline swing vs. TOSA offsets. This indicates the likelihood of undetectable error. V_{DD}=1.08V, T=27° C.

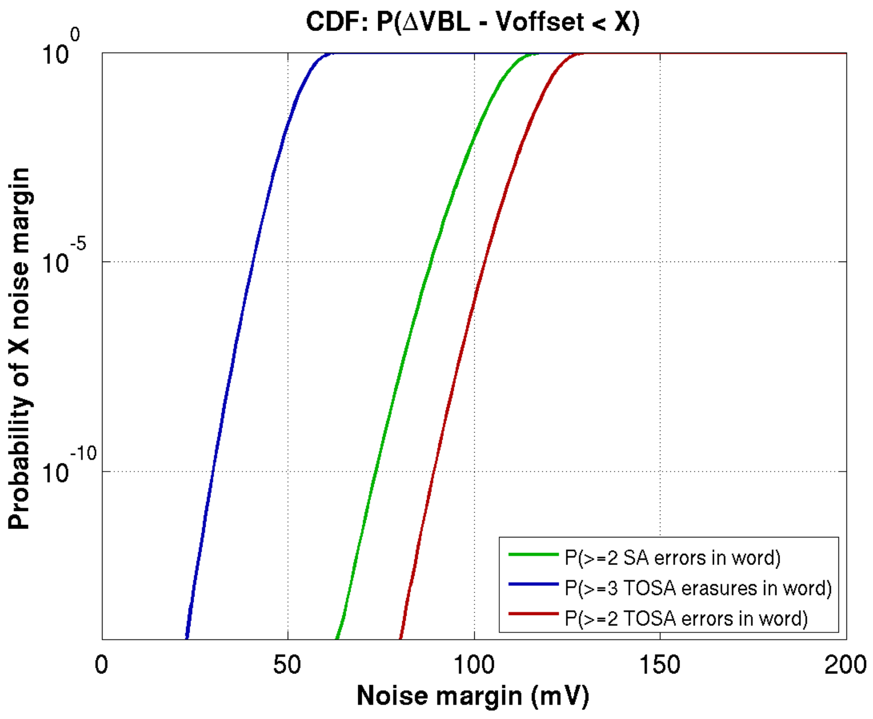


Figure 4.18: CDF of 6T cell bitline swing vs. SA and TOSA offsets. This indicates the likelihood of uncorrectable error in a 72-bit word with SECDED ECC. V_{DD}=1.08V, T=27° C.

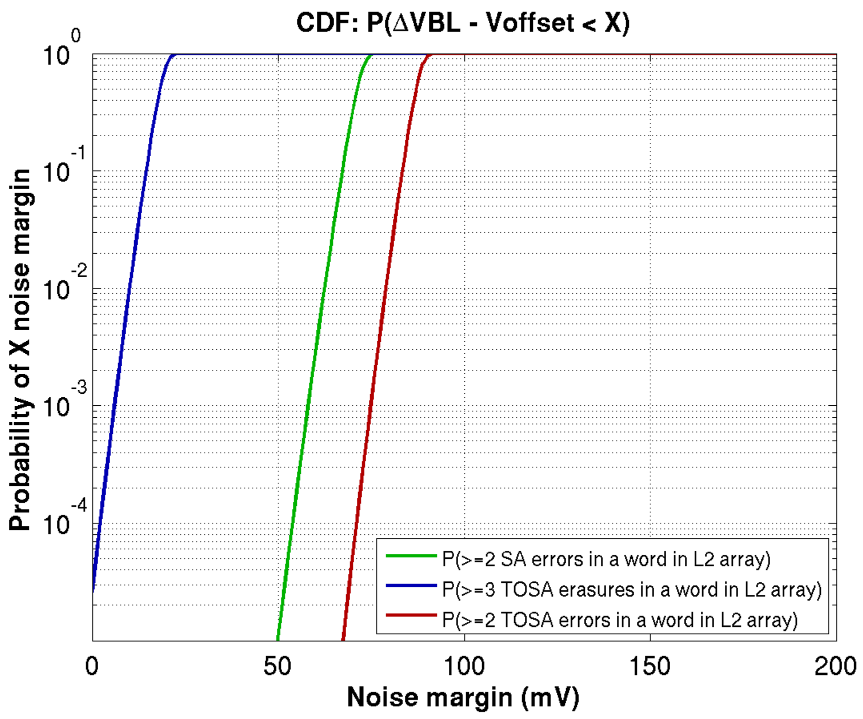


Figure 4.19: CDF of 6T cell bitline swing vs. SA and TOSA offsets. This indicates the likelihood of uncorrectable error in an 8Mb L2 SRAM with SECDED ECC. V_{DD}=1.08V, T=27° C.

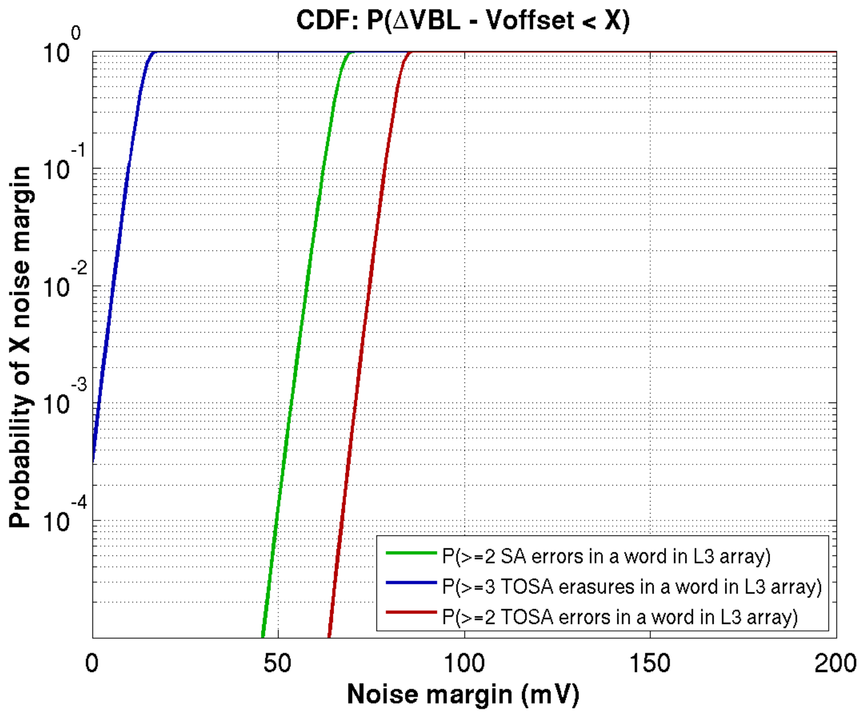


Figure 4.20: CDF of 6T cell bitline swing vs. SA and TOSA offsets. This indicates the likelihood of uncorrectable error in a 100Mb L3 SRAM with SECDED ECC. V_{DD}=1.08V, T=27° C.

estimated local array area overhead of 6% for a 128-cell column design in 65nm. In Chapter 5 we discuss peripheral logic failure detection circuits and present a novel technique of detecting address decoder failures which has benefits for area overhead in large cell array designs.

Chapter 5

Peripheral circuitry protection

While most study has focused on the SRAM cell array, for high-reliability products, it is important to examine the effects of failures on the peripheral logic as well. Unlike in the cell array, where a failure in a storage element can cause immediate corruption of data, single event upsets (SEUs) in the SRAM peripheral logic aren't generally able to cause an error in isolation. Rather, the effects of the SEU depend on the location as well as state of the memory at the time of the upset. Information on fault modes potentially resulting from these SEUs and their relative likelihood could guide designers in handling these faults using techniques which are most effective given their overhead.

To this end, we have conducted an investigation into the potential functional fault modes resulting from energetic particle strikes on the address decode logic of an industrial design in the 40nm node. The study specifies the types of faults which will occur given a particle strike in a particular area of the physical layout. The analysis serves to guide our study of which techniques are the most effective at handling these faults given their overhead.

5.1 Particle strike locality

We performed the analysis by writing a SKILL script to examine the layout of an industrial SRAM row decoder for vulnerabilities to particle strikes in certain windows. Locally, a particle

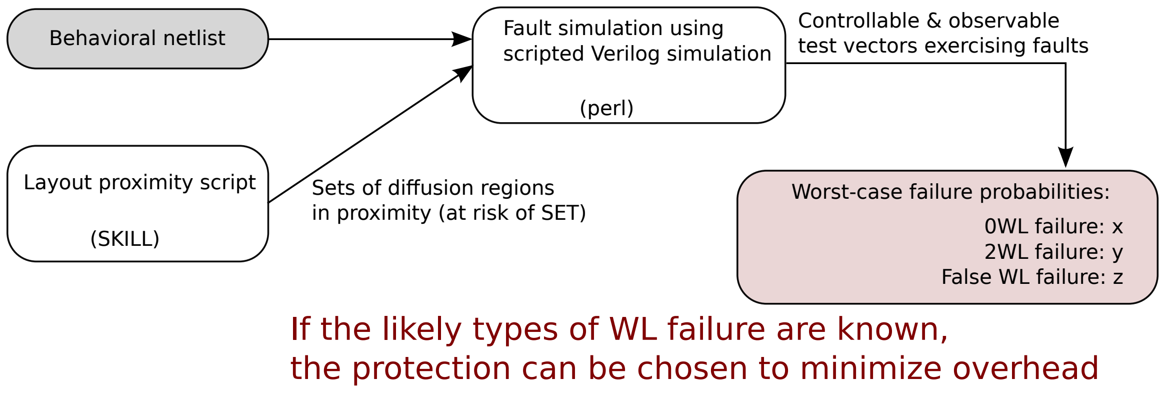

Behavioral netlist
Layout proximity script
(SKILL)
Sets of diffusion regions
in proximity (at risk of SET)
Fault simulation using
scripted Verilog simulation
(perl)
Controllable & observable
test vectors exercising faults
Worst-case failure probabilities:
0WL failure: x
2WL failure: y
False WL failure: z
If the likely types of WL failure are known,
the protection can be chosen to minimize overhead

Figure 5.1: Particle strike effect study: Process.

strike can be pessimistically modeled as a stuck-at fault, which we chose to do to study the worst-case functional effects of particle strikes on SRAM peripheral logic. The goal was to show the possible consequences on decoder behavior of a particle strike in that window, or in other words, the potential faults that could be the result of a SET due to a particle strike. The results of the SKILL proximity script are then fed into a Verilog model of the decoder to determine through fault simulation the results of each set (see Figure 5.1).

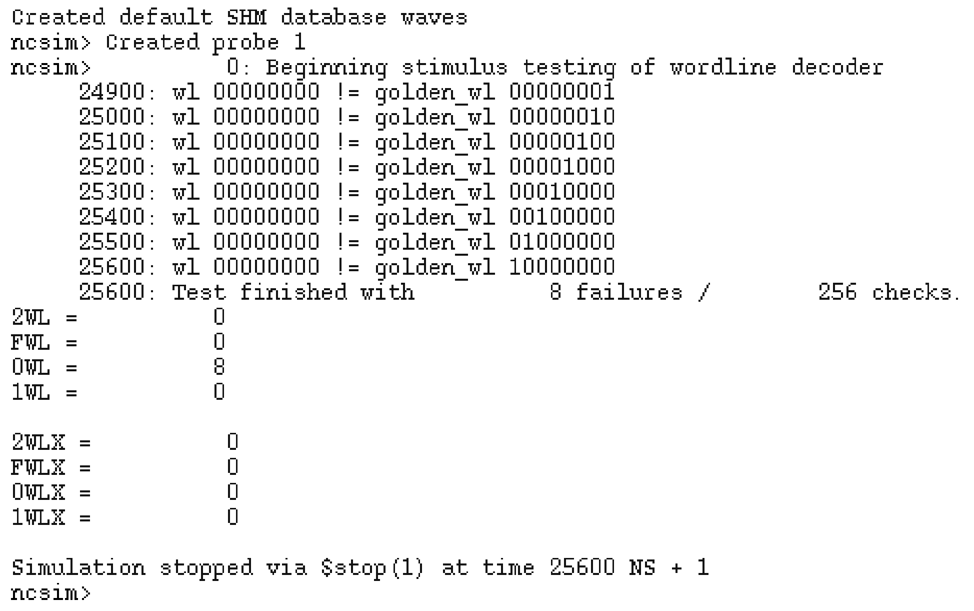
The address decoder can fail in one of three ways:

1. 0WL failure: The desired wordline is not asserted, and no other wordline is asserted. Essentially the assertion of the wordline is suppressed.

2. Wrong-row failure: The desired wordline is not asserted, but one or more other wordlines is asserted. This type of failure has two sub-types depending on the proximity of the desired wordline to the improperly asserted wordline. If the two wordlines are in the same block, we term this a FWL. Otherwise, we term this a WBWL. Sometimes we also refer to a WBWL failure as a “1WL” failure when we’re examining the failures within a single block. The “1WL” terminology indicates a failure where the expected behavior was 0 wordlines in the block asserted but the actual behavior was 1 wordline in the block asserted.

3. 2WL failure: The desired wordline is asserted, but one or more other wordlines is asserted as well, which may cause contention on a read and data corruption on a write.

For simplicity, the terms “row” and “wordline” are used to refer to the address decode path’s output; it should also be noted that the column select suffers from the same failure types.



```
Created default SHM database waves
ncsim> Created probe 1
ncsim>          0: Beginning stimulus testing of wordline decoder
     24900: wl 00000000 != golden_wl 00000001
     25000: wl 00000000 != golden_wl 00000010
     25100: wl 00000000 != golden_wl 00000100
     25200: wl 00000000 != golden_wl 00001000
     25300: wl 00000000 != golden_wl 00010000
     25400: wl 00000000 != golden_wl 00100000
     25500: wl 00000000 != golden_wl 01000000
     25600: wl 00000000 != golden_wl 10000000
     25600: Test finished with          8 failures /        256 checks.
2WL =          0
FWL =          0
0WL =          8
1WL =          0

2WLX =          0
FWLX =          0
0WLX =          0
1WLX =          0

Simulation stopped via $stop(1) at time 25600 NS + 1
ncsim>
```

Figure 5.2: Fault simulation: example

The results of this study are shown in Figure 5.5. This demonstrates the relatively small likelihood of false wordline failures due to energetic particle strikes, at least in the chosen industrial row decoder layout. This is intuitively understandable: each of the other failure types requires only that a single logic path be asserted or de-asserted, while a FWL failure requires that the desired wordline’s logic path be de-asserted and in addition that another wordline’s logic path be asserted. While not inconceivable, the combination of these two events in a single particle strike is less likely than either of them occurring independently.

This result shows us that if we could design a fault detection technique that would catch all 0WL, WBWL, and 2WL failures, we would be ensuring resiliency in almost all circumstances against energetic particle strikes in the address decoder. We present such a technique in Section 5.2.

In deciding which segment of the periphery is most vulnerable, we chose the latter portion of the address path. Our reasoning for choosing this is that hardening techniques exist to secure the address latches [12, 42], and a particle strike on a portion of the data path would result in a

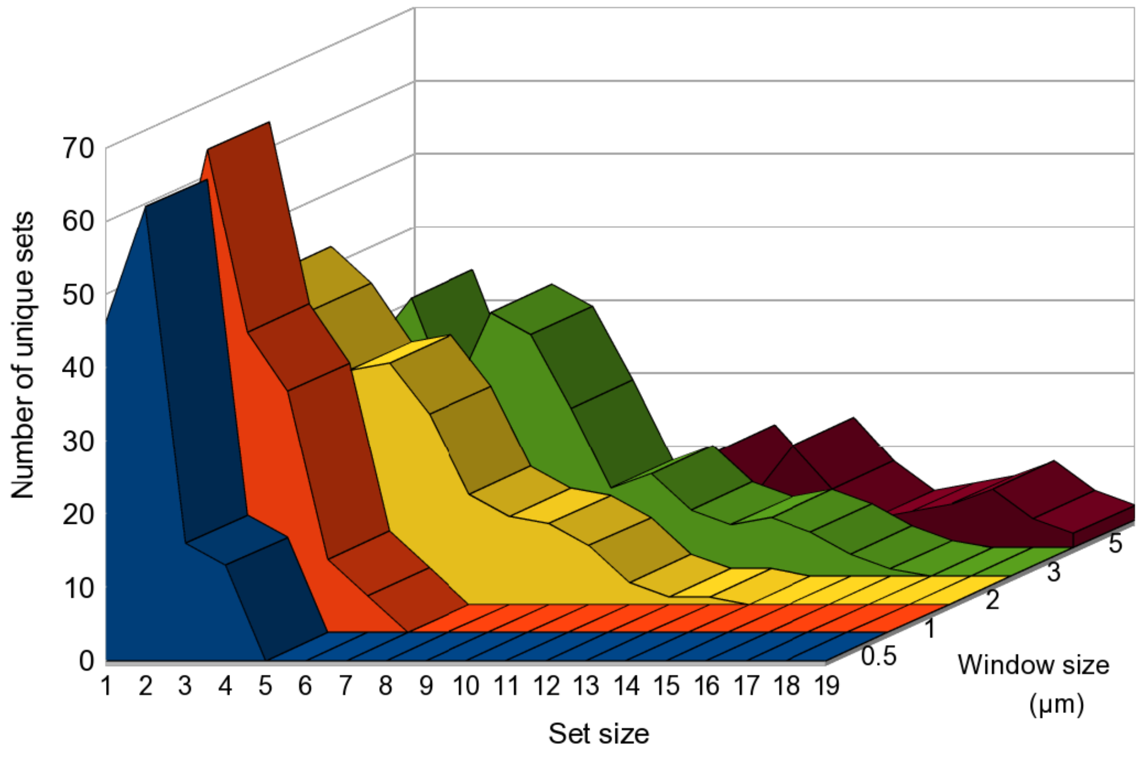
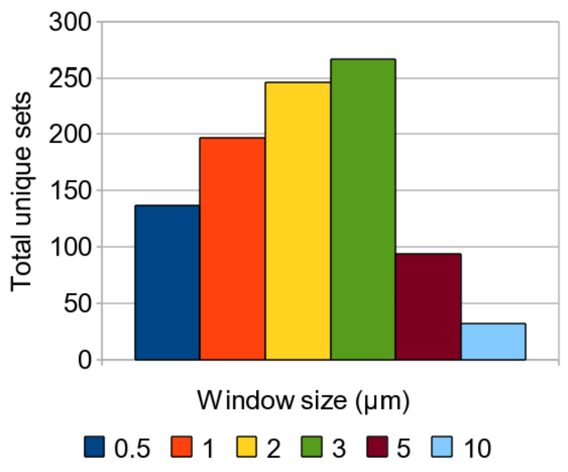


Figure 5.3: Layout proximity: number of unique sets, by number of nodes per set, found in each window size



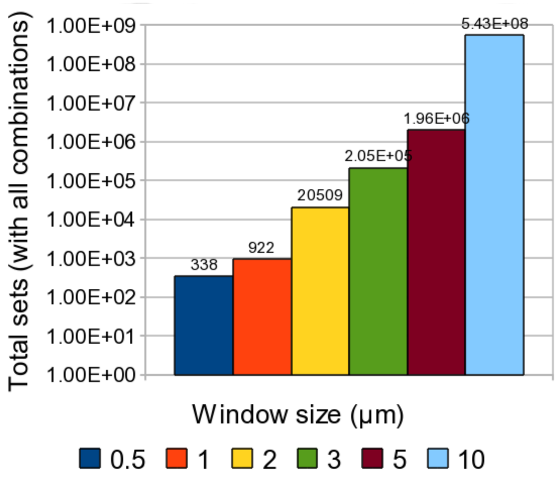


Figure 5.4: Layout proximity: number of sets found in each window size

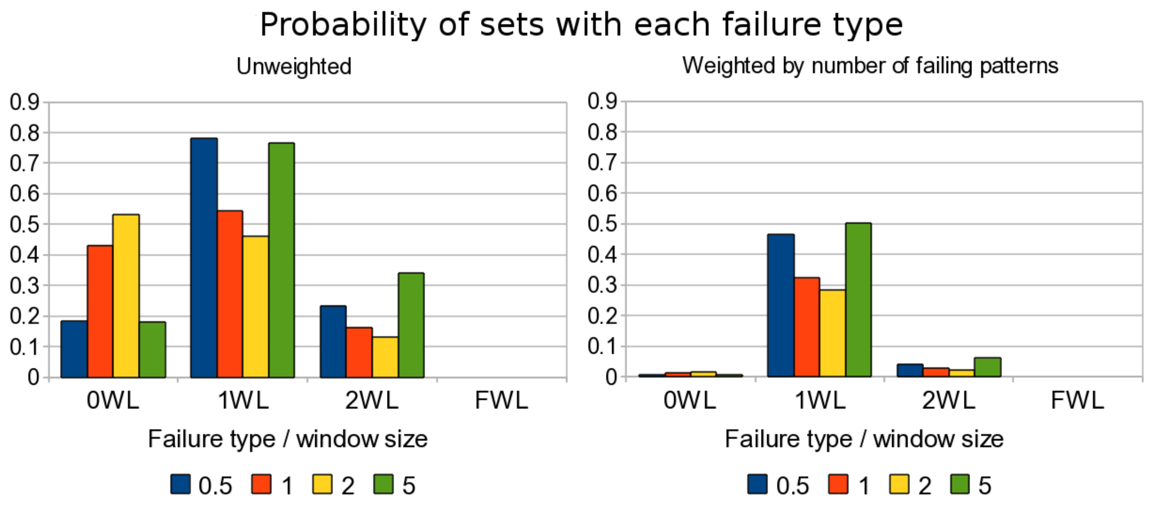


Figure 5.5: Results of an examination of layout proximity in an industrial SRAM row decoder. 0.7 on y-axis of unweighted graph indicates 70% of sets manifested the failure type on the x-axis. The weighted graph multiplies this number by the proportion of failing patterns to total tested patterns, e.g., if 70% of sets manifested a 1WL failure type and these sets on average demonstrated failure in 248/256 patterns, the weighted probability would be 0.68.

disturbance to one or more bits of a word as opposed to the entire word. These failures can be handled by the same techniques which handle failures in the array itself, such as ECC.

5.2 Peripheral logic fault detection

We place the emphasis of our investigation on the address decode path for the reasons stated in Section 5.1: a particle strike on a portion of the data path (including the column I/O circuitry) would result in a disturbance to one or more bits of a word as opposed to the entire word, and thus would introduce a similar failure pattern as that of a particle strike on the cell array. These failures can be handled by the same techniques which handle failures in the array itself, such as ECC. The exception to this is the column multiplexer decoder, which is technically part of the address decoder, and thus its faults can be covered by the same method (re-encoding the address from the decoded horizontal column mux lines) as those used to protect the wordline decoder in the literature [82]. We will not discuss it further here.

Additionally, there has been significant work in the literature on protecting the sensitive storage nodes of latches, such as those that might hold the address [12, 47, 81]. Since latches and other sequential elements generally represent a small portion of the overall memory area, an proportionate increase in size in exchange for hardening against particle strikes is not a significant overall cost as compared to the cost of hardening the SRAM cells or address decode logic.

We evaluate the differences between an existing technique and our novel technique, generating enough information to allow designers to make a decision as to which scheme is suitable for a memory with given array size and specifications.

The wordline encoder discussed in Section 3.6 carries certain requirements: the ROM bitlines must be full-swing and single-ended, and a column's pitch is dependent on the column output devices rather than the ROM cell itself. In a design whose data bitlines are single-ended static sense [82], this may be suitable. But for last-level cache, where density is important, a large number of cells on a long bitline may be desirable to maximize array efficiency (the ratio of cell area to total area including peripheral logic). In this case, the number of address bits and thus the area overhead of the ROM bitlines increases.

The local area overhead of the wordline encoder technique can be calculated using the assumptions shown in Figure 5.6: a 32-row block would carry an overhead of 3.5% while a 512-row block would carry an overhead of 6.2%. The total area overhead for all EC circuitry is reported as $< 15\%$ of cache area [82].

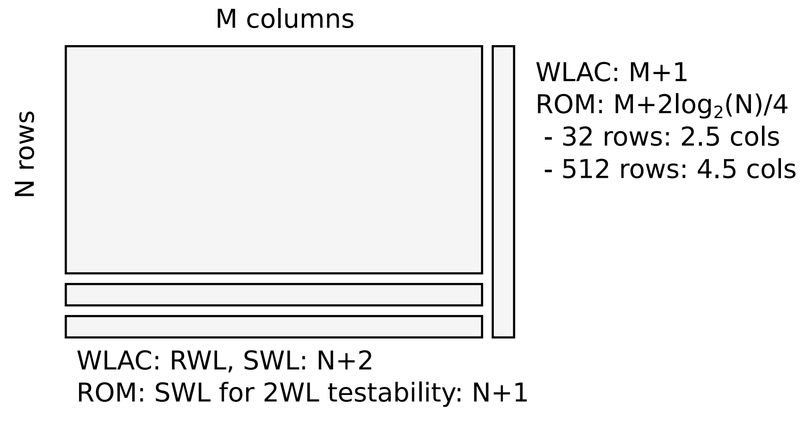
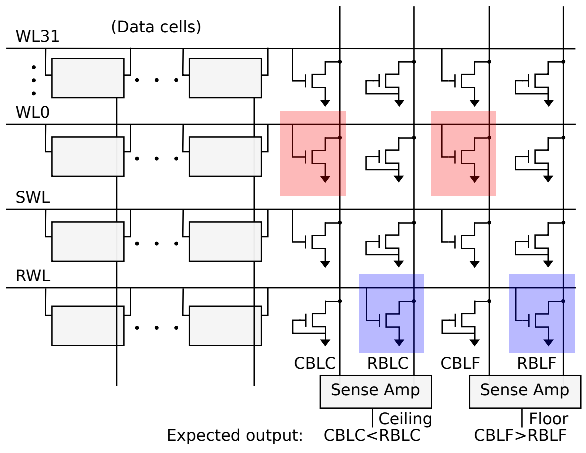


Figure 5.6: Local area overhead of both ROM address storage and wordline assertion comparator

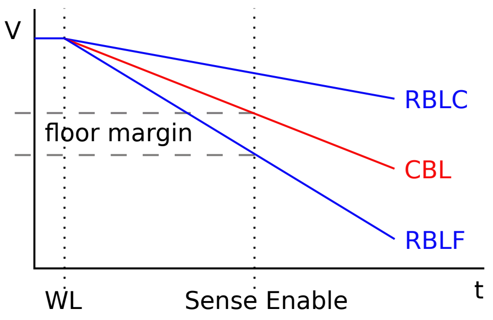
An additional caveat of this technique is that it does not provide manufacture-time testability (demonstration of caught failures) without augmentation. We discuss additions to allow manufacture-time testability in Section 5.2.1.

5.2.1 Wordline assertion comparator

In response to the stated limitations of the ROM-ADDR technique we have designed a **wordline assertion comparator (WLAC)** (Figure 5.7), which compares the combined assertion time of the wordlines in the array against the assertion time of a reference wordline (RWL). In the ideal case, the assertion time should be the same, which would indicate that exactly one wordline has fired.



(a) Circuit diagram



(b) Waveform

Figure 5.7: WL assertion comparator enables detection of "no-row" or "multi-row" address decoder faults.

We propose an alternative scheme using a pair of sense amplifiers along with two pairs of differential bitlines to ensure that only a single wordline has been asserted. We compare the wordline assertion times by measuring the voltage of two pairs of bitlines which we designate the check bitline (CBL) and the reference bitline (RBL). The check bitline is pulled down by an NFET attached to the requested wordline (or whichever wordlines are asserted in the case of an address decode failure), while the reference bitline is pulled down by an NFET attached to a reference wordline.

The ceiling bitline pair are compared to verify that a 0WL failure has not occurred, while the floor bitline pair are compared to verify that a 2WL failure has not occurred. Correct behavior is indicated when the check bitlines are below the reference ceiling and above the reference floor, i.e., CBLC $<$ RBLC and CBLF $>$ RBLF. In other words, in the event of a no-row failure, the voltage on CBLC will be higher than that of RBLC; in the event of a multi-row failure, the voltage on CBLF will be lower than that of RBLF. Thus, this scheme addresses both the no-row and multi-row failure modes.

Adapting the ceiling bitline pair to its purpose means sizing the reference bitline pull-down device for a lower on-current than the lowest on-current (through mismatch variability simulation) of a check bitline pull-down device. Adapting the floor bitline pair to its purpose means sizing the reference pull-down device for a higher on-current than the highest on-current (through mismatch variability simulation) of a check bitline pull-down device. Additionally, in both the ceiling and floor pair, the design sizing of the check bitline pull-downs should be chosen such that the nominal device brings the voltage of the check bitline to $V_{DD}/2$ at the same time the sense enable signal is asserted for the block or the bitline is otherwise read out. This is chosen so as to balance the competing interests of maximum differential shown at the sense amplifiers and ensuring that both check and reference bitline voltages do not reach 0 V, which would reduce the reliability of the checking.

The sense amps which compare check bitline (CBL) to reference bitline (RBL) are asserted by a control signal with timing similar to the sense enable signal but which asserts on both read and write cycles.

For testability purposes, two new input pins (t_0wl and t_2wl) are added to the memory to exercise the 0WL and 2WL failure modes. The 0WL failure mode is induced by adding an input to an early stage of the per-array decode structure which is controlled by t_0wl, allowing the decoders to effectively be turned off. The 2WL failure mode is induced by adding a "second wordline" (SWL) which follows a replica path through the row decoder logic whose activation is controlled by t_2wl. In the event of a false positive or false negative response during manufacture-time test of the 0WL and 2WL failure modes, the WLAC is determined to be faulty, which would increase yield loss. Thus it is of paramount importance that the WLAC design be robust against variation.

The local area overhead of the wordline assertion comparator block can be calculated using the assumptions shown in Figure 5.6: a 32-row block would carry an overhead of 9.2% while a 512-row block would carry an overhead of 3.2%. Note that this includes the testability enhancement of the SWL; if this enhancement is also included in the WL encoder/ROM address storage scheme, its local area overhead increases to 6.7% for a 32-row block or 6.4% for a 512-row block. The compact 32-row block would be better suited for L1 cache, while the dense 512-row block would be better suited for last-level cache which emphasizes density over performance. Thus, this technique is more beneficial at higher levels of cache with longer bitlines, but may not be preferable to ROM-ADDR in all cases. The WLAC technique achieves 3% area improvement over ROM address bits for a 512-row array.

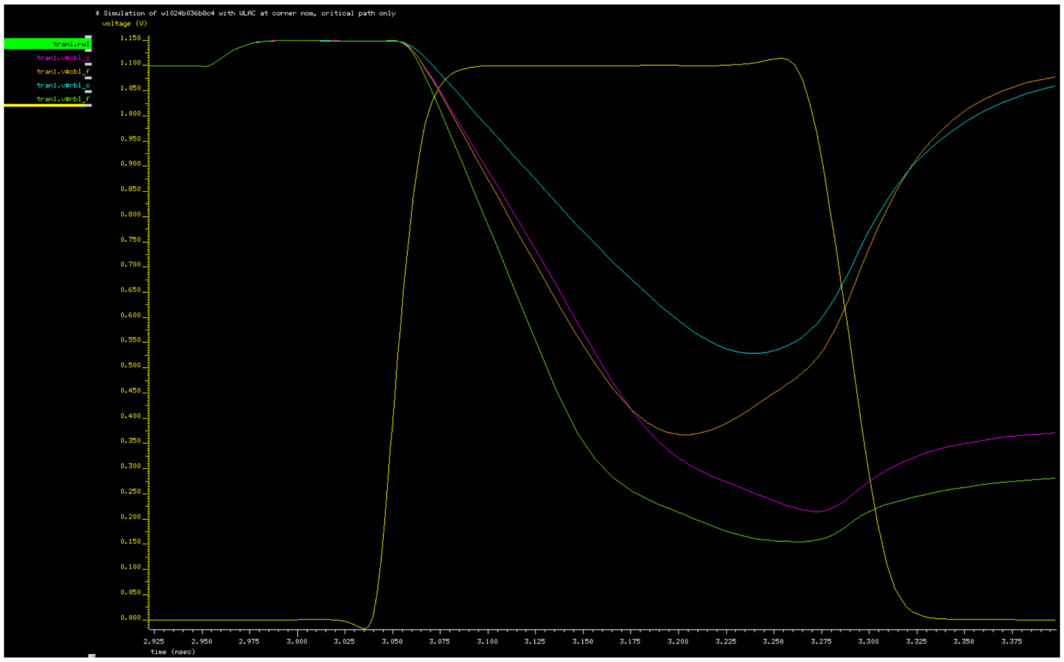


Figure 5.8: Waveform showing correct operation detected by the WLAC. Correct behavior is indicated when the check bitlines are below the reference ceiling and above the reference floor, i.e., CBLC < RBLC and CBLF > RBLF. Legend: yellow = RWL, pink = CBLC, orange = RBLC, blue = CBLF, green = RBLF.

The wrong-row failure mode is addressed by the ROM address storage scheme but not by WLAC. We showed in Section 5.1 that this failure mode is far less likely than 0WL or 2WL failure modes. Furthermore, since both of these schemes are mainly of interest for high-reliability

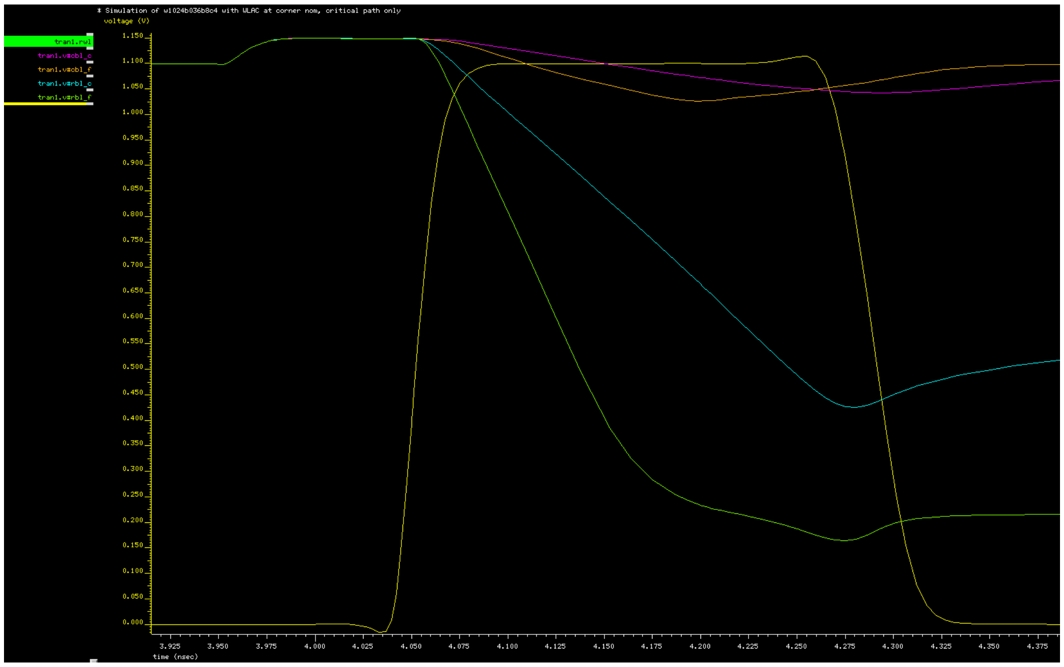


Figure 5.9: Waveform showing 0WL failure mode detected by the WLAC. The incorrect behavior is detected, as the check bitlines are above the reference ceiling, i.e., CBLC $>$ RBLC. Legend: yellow = RWL, pink = CBLC, orange = RBLC, blue = CBLF, green = RBLF.

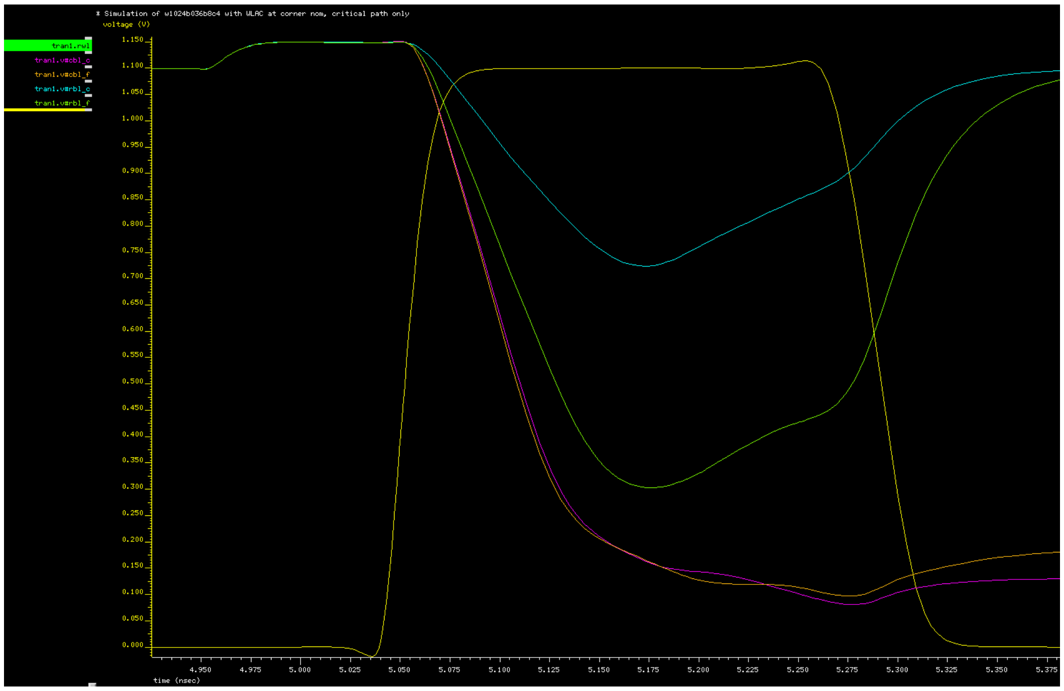


Figure 5.10: Waveform showing 2WL failure mode detected by the WLAC. The incorrect behavior is detected, as the check bitlines are below the reference floor, i.e., CBLF $<$ RBLF. Legend: yellow = RWL, pink = CBLC, orange = RBLC, blue = CBLF, green = RBLF.

memories, it is a fair assumption that some type of ECC would be used as well. For SECDED ECC, codes are generally truncated to match the data word width rather than being used at the optimal code length. Coding theory shows that for a code with m parity bits, the code word is $2^m - 1$ bits, and the data word is $2^m - m - 1$ bits. This corresponds to distance 3 codes of (31, 26), (62, 56), or (127, 120). An extra parity bit enables distance 4 codes of (32, 26), (63, 56), or (128, 120); these codes can be used for SECDED. However, data words are frequently powers of 2, so these codes would be truncated to (22, 16), (39, 32), or (72, 64). Thus, there is slack in the word which can be used to encode protection for additional bits without adding overhead of increased parity bits. These additional bits can be used to encode the address along with the data, thus providing the capability of detection of a wrong-row access with only a slight increase in ECC encoder/decoder area but no local array overhead.

To test the concept, the WLAC unit was designed in a 40nm bulk complementary metal-oxide-semiconductor (CMOS) process on an industrial memory design 36 kb in size with 1024 words, 36-bit words, and 4-way column multiplexing. Simulations of the design across corners and mismatch demonstrate that the technique can be employed with robust safety margins for detecting SETs.

5.2.2 Results

The design was simulated at nominal corner (Figures 5.8, 5.9, & 5.10), across corners (Figure 5.11), and through MC analysis with process and mismatch variation (Figures 5.12 & 5.13). In each of these analyses, negative margins were shown in simulations with failures in the address decoder, while positive margins were shown in simulations with correct functionality of the address decoder. The worst-case margins out to 4σ process/mismatch variation were -197 mV for the 0WL case, 92 mV and 35 mV for the ceiling and floor margins of the 1WL case, and -43 mV for the 2WL case. These results demonstrate the robustness of the technique.

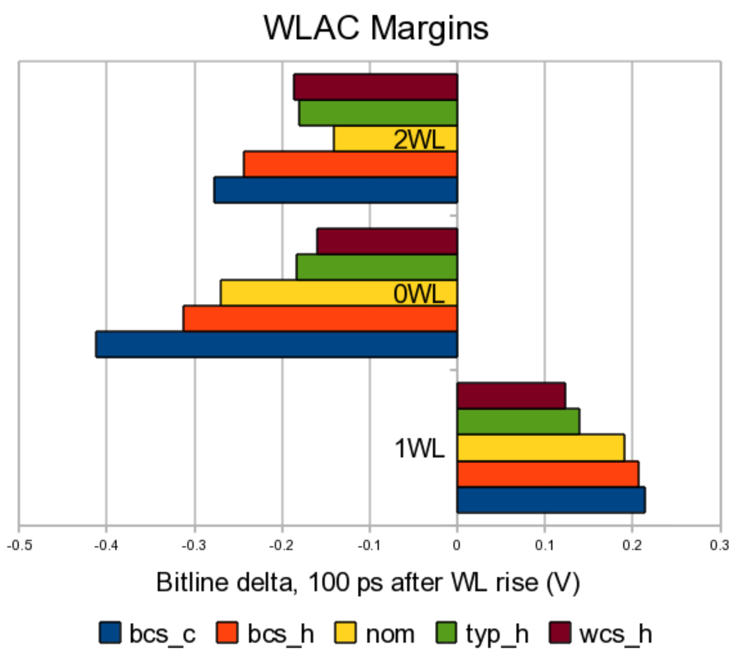
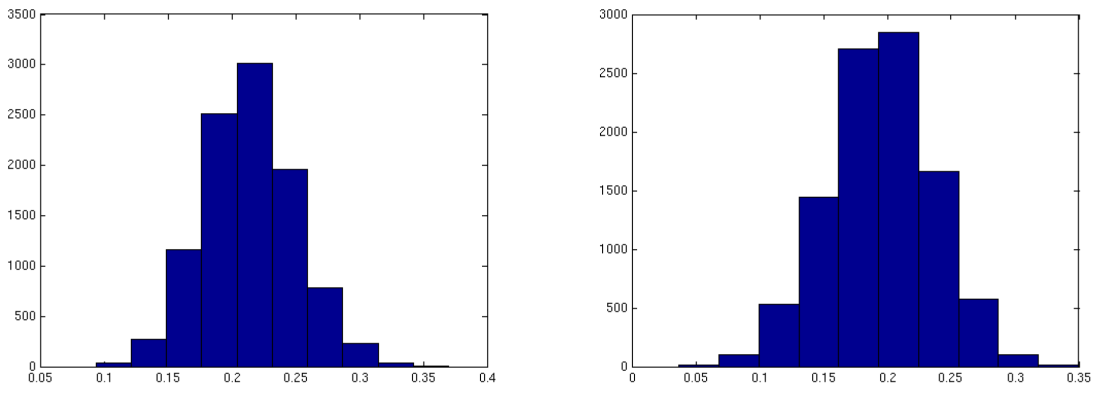


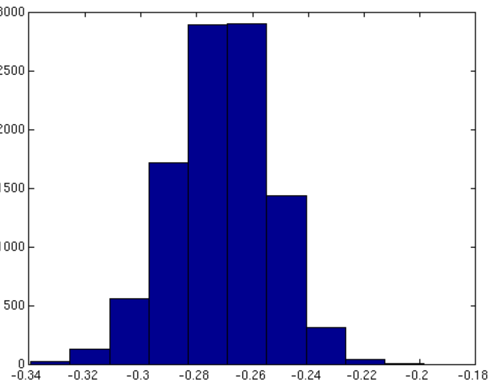
Figure 5.11: Comparison of margins across corners



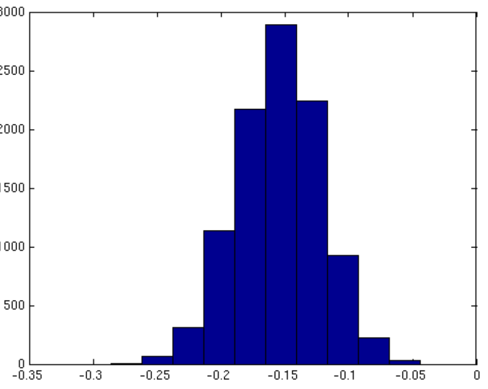
(a) Ceiling margin, 1WL

(b) Floor margin, 1WL

Figure 5.12: WLAC margins: correct operation. Ceiling pair: $\mu = 213mV; \mu - 4\sigma = 92mV$ Floor pair: $\mu = 195mV; \mu - 4\sigma = 35mV$



(a) Ceiling margin, 0WL



(b) Floor margin, 2WL

Figure 5.13: WLAC margins: failing operation. Ceiling pair: $\mu = -270mV; \mu + 4\sigma = -197mV$ Floor pair: $\mu = -154mV; \mu + 4\sigma = -43mV$

5.3 Summary

In this chapter we evaluated a novel technique for detecting address decoder failures which has benefits for area overhead versus a competing technique in large cell array designs. Our study of particle strike locality in an industrial address decoder design demonstrates that the risk of 0WL, 2WL, or WBWL failures is much higher than the risk of FWL failures; thus, the structure of the WLAC is amenable to addressing the most common error manifestations of particle strikes. Additionally, through the augmentation of ECC by encoding both the address and the data, these rare failure events may be detected as well. An illustration of the technique's differences from ROM-ADDR for certain memory sizes and specifications is shown in Section 5.2.1, with improved area efficiency when used with large, dense memories. The WLAC technique achieves 3% area improvement over ROM address bits for a 512-row array.

In Chapter 6 we will explore the design and implementation of a testchip in a 55nm bulk CMOS process containing multiple reliability-improvement techniques including vertical parity, erasure coding using runtime read latency failure detection circuits and offline erasure storage in CAM, and ROM address bits to detect address decoder failure.

Chapter 6

Prototype testchip with erasure-based ECC

To address failures occurring in the cell array, we have implemented a novel multi-bit ECC scheme based on erasure coding that can extend conventional ECC already used in memory to correct and detect multiple erroneous bits with low latency, area, and power overheads. Our EB-ECC scheme can correct two hard errors and detect one soft error per word without significantly increasing memory access latency. When EB-ECC is combined with the existing low-overhead soft error correction mechanism of vertical parity [35, 44], our scheme can correct two hard errors and one soft error per word. This design is the maximum number of erasures correctable while maintaining an extra bit of random error detection given a Hamming code of minimum distance 4 (see Table 4.1), though erasure coding could provide even greater multi-bit error correction given a Hamming code with a higher minimum distance. Since correction would no longer be a single-cycle low-area operation in this case, it was not implemented at this time.

The testchip, which was designed and fabricated on a 55nm bulk CMOS process, viewed from a high level, contains a core SRAM, the novel EB-ECC modules, and BIST circuitry necessary to test the functionality of the design (Figure 6.1) [45]. This 55nm technology was an optical shrink from 65nm; the design and simulation was performed using 65nm models and all cited measurements are taken from 65nm layout, however the testchip returned from fabrication contained devices optically shrunk from 65nm to 55nm minimum feature size. Thus, simulation and area data can be compared to other designs in the 65nm node while silicon testchip results should be compared to designs in the 55nm node.

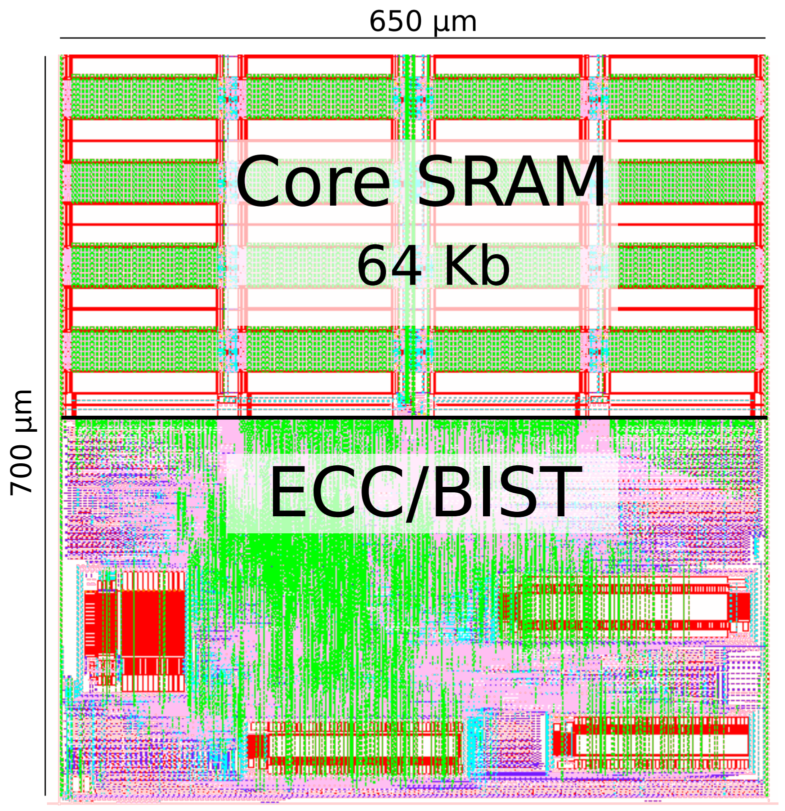


Figure 6.1: Top-level layout of testchip on 55nm bulk CMOS process, containing a core SRAM, the novel EB-ECC modules, and BIST circuitry necessary to test the functionality of the design.

6.1 Core memory

The core SRAM is a 64 kb array consisting of 16 sub-blocks, each 64 rows and 72 columns of 6T cells in size. The memory is designed for single-cycle access with a frequency of 800MHz at 1.2V and completes an atomic read-before-write operation on the event of a write request, which is required for updating the vertical parity bits in 2-D coding [35, 44]. The memory has programmable self-timed circuitry for both the read and write paths. The V_{DD} supply to the core is decoupled from the rest of the logic. Independent control of the supplies can provide a better understanding of memory errors as a function of supply voltages. This also creates various conditions to assess the error correction capabilities of our design.

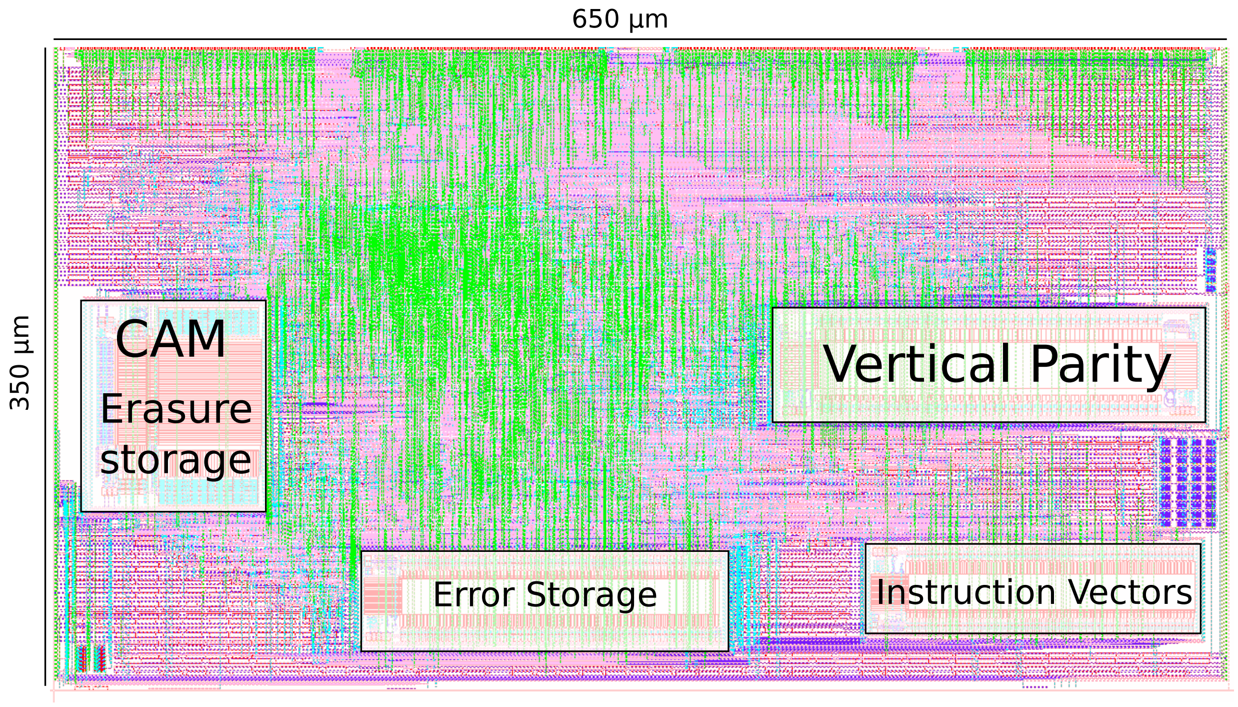


Figure 6.2: Enlarged view of synthesized ECC, BIST logic, and supplemental memories.

Technology	55nm bulk CMOS
Nominal supply	1.2 V
Routing	4 metal layers
Area	0.46 mm^2
Core array	0.23 mm^2
ECC/BIST	0.23 mm^2

Core array
42% area efficiency
1.06 μm^2 6T cell
1.2 ns read latency
16 (64x72b) blocks

Table 6.1: Testchip detail

The memory has a hierarchical bitline architecture with every local bitline shared across 32 rows. The memory word size is 64 bits, with 8 extra parity bits per word to store the SECDED encoded bits. The vertical parity also requires marking each word as "valid" or "invalid" and this is done using an extra specialized bit per word that marks the validity of the word, which is always set to "valid" when a write cycle occurs on that particular word. Global invalidation is an efficient 1-cycle operation that invalidates the entire memory. This per-word valid bit is used during correction: only words which have deterministic values set after power-up are used in the parity reconstruction of a corrupted data word. If there were no valid indicator, random data in words from their power-up state would corrupt the vertical parity reconstruction. Another way of handling this at an architectural level is to require a full array clear for both the vertical parity and core arrays, to ensure consistency of the vertical parity with the contents of the memory array. The valid bit structure eliminates this boot-up delay at the cost of one extra cell/word overhead in area.

To enable runtime erasure detection of read latency failures, both bitlines are read as shown in Figure 4.5.

During the read-before-write operation, the write starts after a delay that is controlled by a global signal and programmed to be sufficient for a read. Each column has localized circuitry to detect read completion and this extra time can be used to start the write operation. This slightly increased time for write can prove crucial for cells that suffer from imbalances resulting in fast read but slow write operation. Additionally, 8 configurations for varying read and write latency are available.

6.2 ROM address bits

An additional fast check for the address decoding path is done by an array of ROM that has a 10-bit address-check word and shares the wordline with each 72-bit SRAM word. Each word stores the 10-bit address that should activate that particular wordline. Address decoding errors can be easily caught by comparing the ROM data out to the input address of each cycle. Our implementation of ROM address bits was done independently and without knowledge of a recent publication of a design which provides the same functionality [82]. In the layout of our design, each DRC-compliant ROM cell was 0.45 μm wide while the DRC-compliant 6T cell was 1.85 μm wide, leading to a 3.4% local array area overhead for the 10-bit ROM address bits. The ROM column pitch is primarily limited by two factors: the column output buffers and the metal width/spacing. One advantage of the previously-published design [82] versus our implementation was its dual complementary structure (see Section 3.6). While this dual complementary structure does not require any additional area for the pulldown transistor, it will require one extra bitline per address bit, which increases the area overhead. Preserving the same bitline pitch of 0.3 μm (0.15 μm metal width, 0.15 μm spacing), transitioning to a dual complementary structure would increase the ROM cell width from 0.45 μm to 0.75 μm, and increase the local array area overhead to 5.7%.

6.3 Supplemental memories

Four supplemental memories are implemented in this testchip. Two (vertical parity and CAM for erasure storage) are needed for the error control coding techniques employed while two (log storage and instruction vectors) are used in conjunction with the BIST logic to allow the test modes described in Section 6.4. All memories with the exception of the CAM use the same 6T cell as is used in the core array. The CAM uses a modified 9T cell for the address and the same 6T cell for the data. The 32-word size of the vertical parity memory was chosen to match the number of cells on the local bitline; in the case of a failure causing corruption of one or more entire columns of 32 bits of data in a single sub-block, the vertical interleaving of the vertical parity array maps every cell of that column to a separate vertical parity word, and thus they all can be reconstructed if the error is detected by the horizontal SECDED ECC.

- Vertical parity memory - A writethrough array with 32 words of 72 bits each.
- Erasure storage memory - A CAM with 32 words; the addressable content is an 10-bit

address uniquely identifying a word in the core SRAM and a single valid bit, while the retrieved data is a 72-bit offline erasure vector for the word identified by the address.

- Log storage memory - A single-port memory with 32 words of 93 bits: each word contains a valid bit, two 10-bit addresses, and a 72-bit data word. The two addresses indicate the requested core memory address of the command being logged and the core memory address returned by the ROM address bits.
- Instruction vector memory - A single-port memory with 32 words of 85 bits: each word contains a valid bit, read and write request bits, a 10-bit address, and a 72-bit data word.

6.4 Synthesized logic and test modes

The error control coding logic on this testchip includes two SECDED ECC decoders and one SECDED encoder. These decoders are used along with the erasure locations as shown in Figures 4.2 & 6.4. The erasure locations are taken from either the runtime erasure detection or the CAM, which stores erasures detected through external chip testing.

From simulation results, the core array latency is 1.2 ns, while the erasure storage CAM is 1.5 ns, and the ECC datapath is 0.6 ns. Thus, the latency overhead for single-cycle operation using runtime erasure correction is 0.6 ns, while the latency overhead for single-cycle operation using offline erasure correction (with erasure locations and vectors read out from the CAM) is 0.9 ns. For a larger core memory design than our 64 kb design, it would be reasonable to assume pipelined operation. If this protection system were implemented in such a pipelined operation, these overhead latencies would correspond to roughly 2 cycles at our target frequency for 65nm of 2 GHz, which we defined in Section 4.5.1. Information on the methodology of our simulation infrastructure is provided in Appendix A.

Built-in-self-test logic is incorporated into the testchip to allow scanned patterns to be applied to the ECC logic and core SRAM (see Figure 6.3). Instructions scanned into the chip can be run singly, repeated, or several instructions can be run in sequence from the supplied 32-word vector

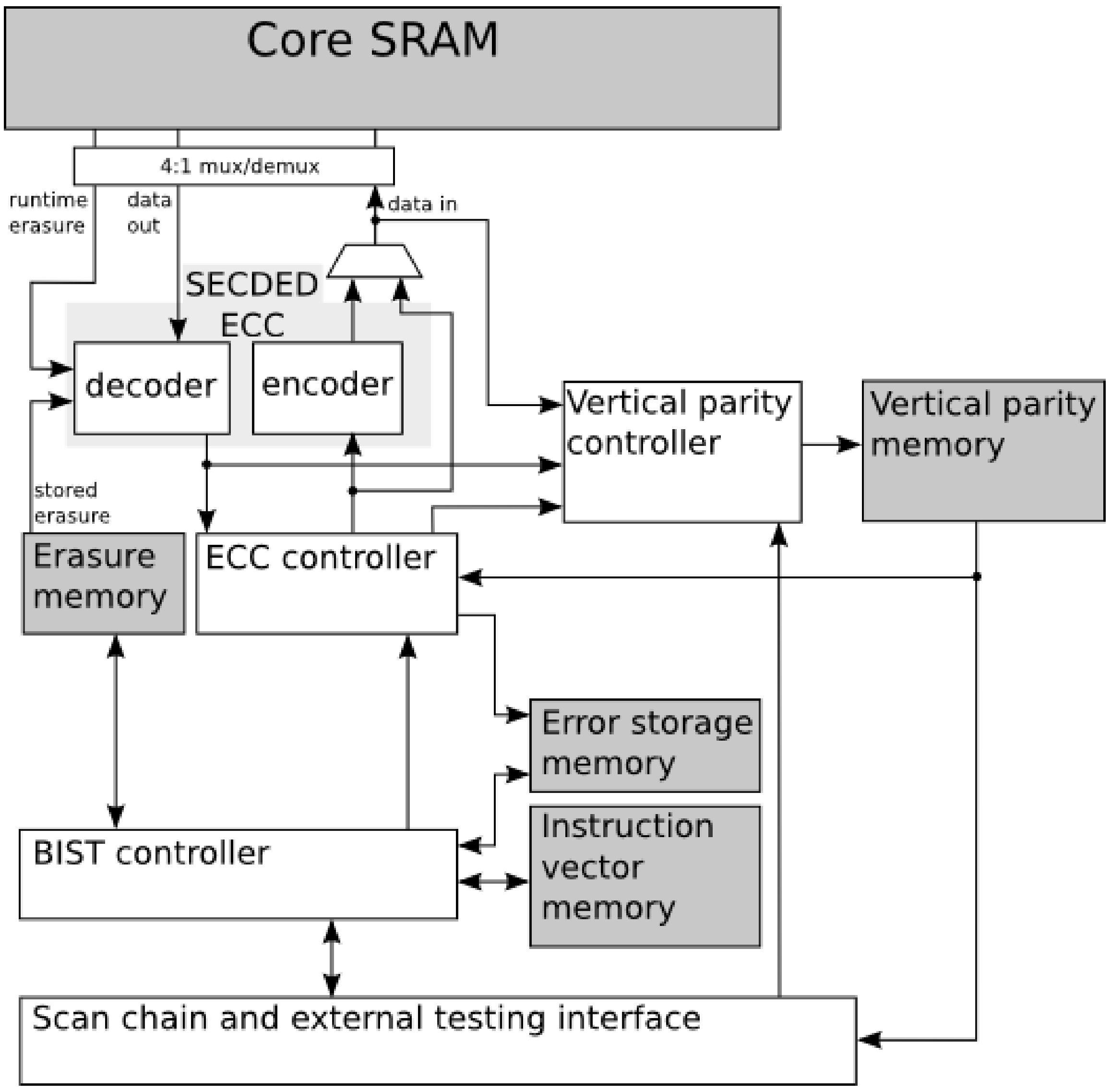


Figure 6.3: Testchip block diagram showing testing support logic and memories. The SECDED ECC decoder in the shaded region is shown in greater detail in Figure 6.4.

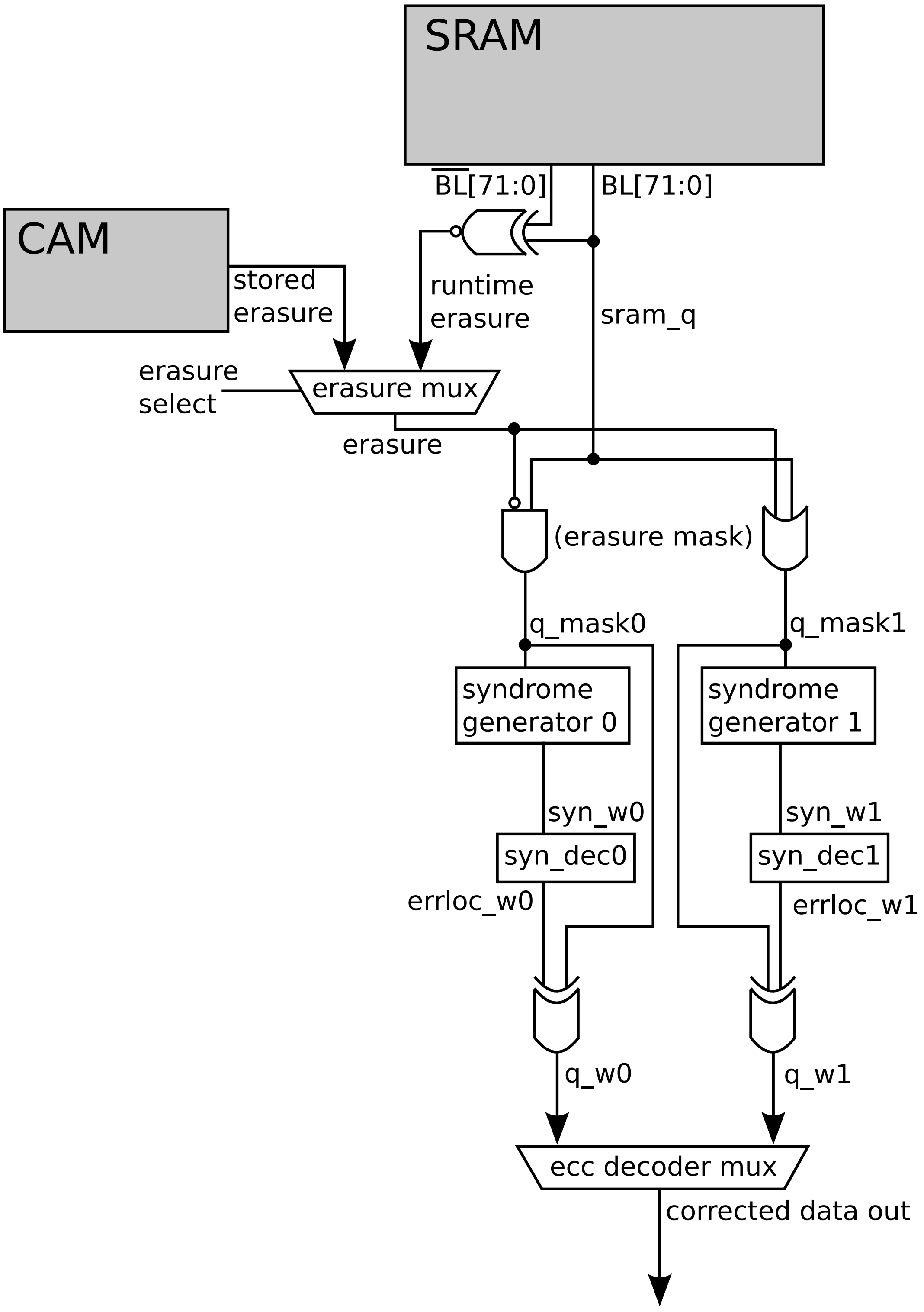


Figure 6.4: ECC datapath showing erasure masking and SECDED decoding circuits.

memory. The intent in provisioning a vector and log storage memory was to be able to run up to 32 operations in sequence at-speed using the on-die clock generator, allowing for test of memory failure patterns which we may not have anticipated during the design process.

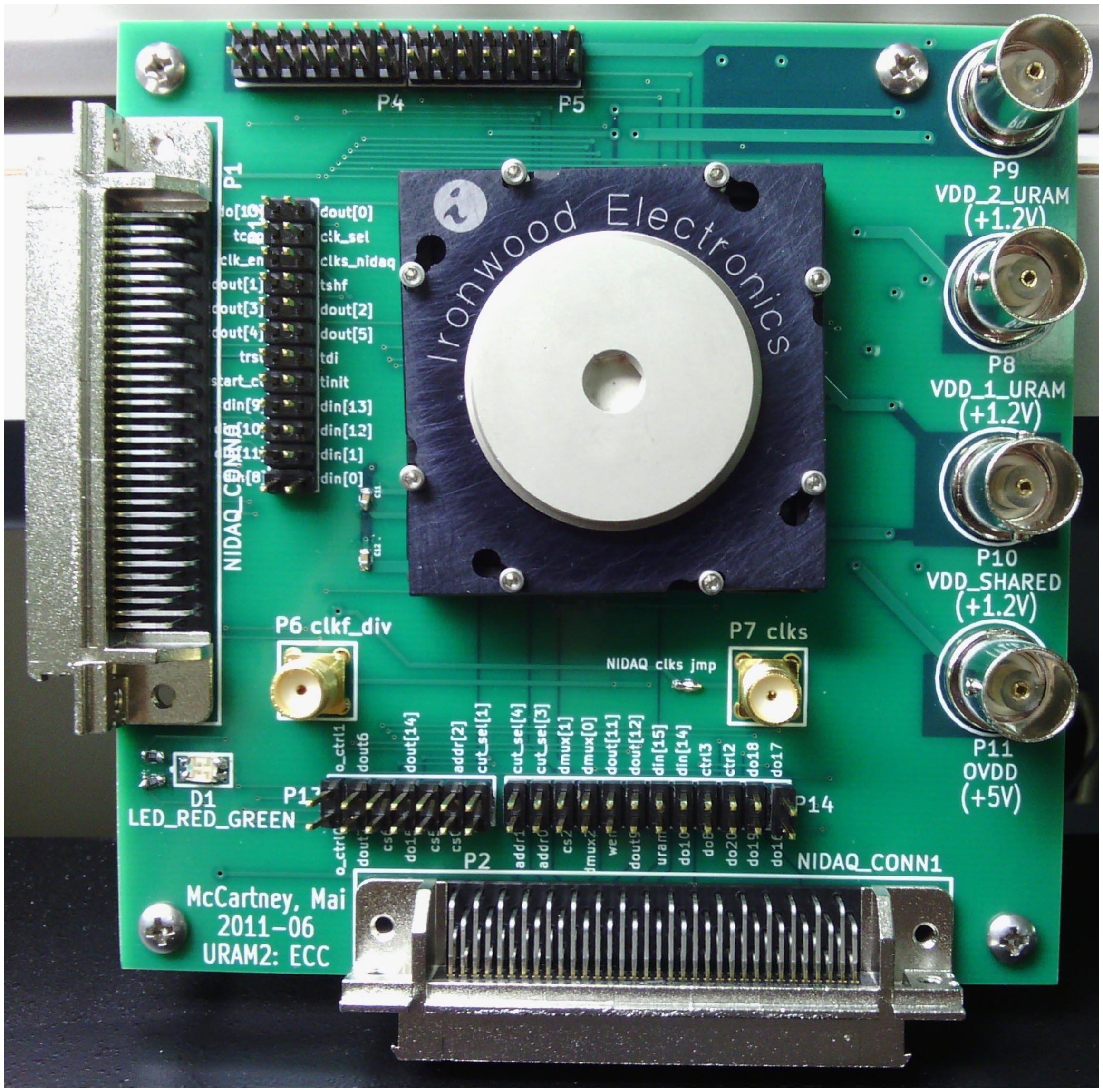


Figure 6.5: PCB with socket and all components populated, disconnected from the DAQ board and power supplies.

	Area	% of core array
Core array	215k μm^2	100
Vert. parity	14k μm^2	6.5
Erasure	11k μm^2	5.1
Logic	182k μm^2	85
Test setup	Area	% of core array
Log storage	10k μm^2	4.7
Inst. vectors	9k μm^2	4.3

Table 6.2: Area breakdown of testchip

6.5 System overheads

6.5.1 Area overhead of EB-ECC

The area overheads of various components on the testchip are shown in Table 6.2; to put these overheads in perspective, we take as a baseline a design from industry of that generation, the Intel Xeon 7100 dual-core (65nm) [13]. This design features a shared L2/L3 bitcell with area 0.624 μm^2. Thus, for the L2 cache of size 8 Mb, we can estimate its area to be 7.5 mm^2 assuming 70% array efficiency. Similarly, for the L3 cache of size 128 Mb, we can estimate its area to be 120 mm^2 assuming 70% array efficiency. Using these assumptions, a practical illustration of area overheads is shown in Figure 6.6.

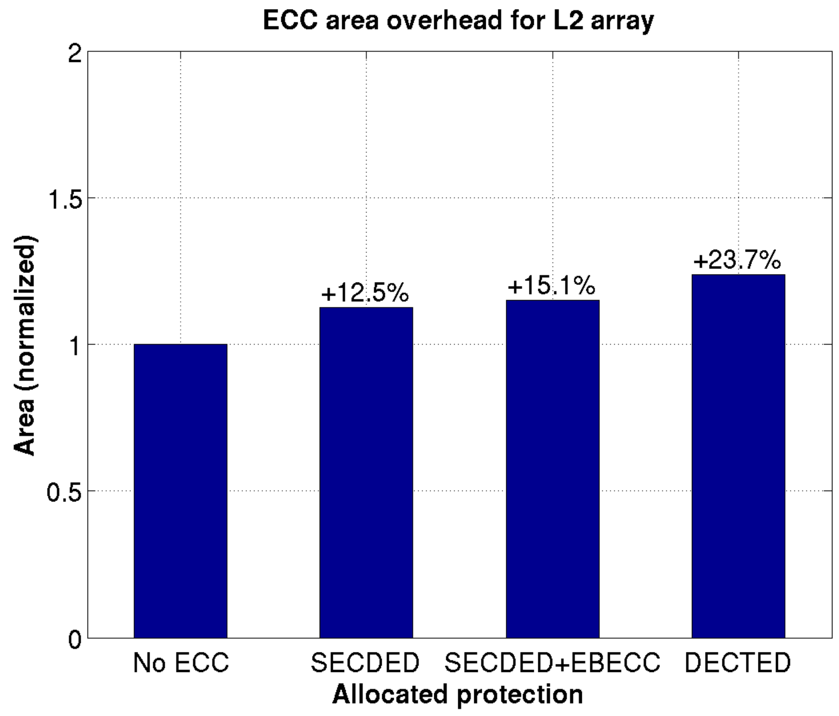
For the L2 cache, the EB-ECC scheme incurs a 2.4% area increase over conventional SECDED-protected L2 array and 6.7% area savings when compared with a DECTED-protected L2 array.

For the L3 cache, the EB-ECC scheme incurs only a 0.15% area increase over conventional SECDED-protected L3 array and 8.8% area savings when compared with a DECTED-protected L3 array.

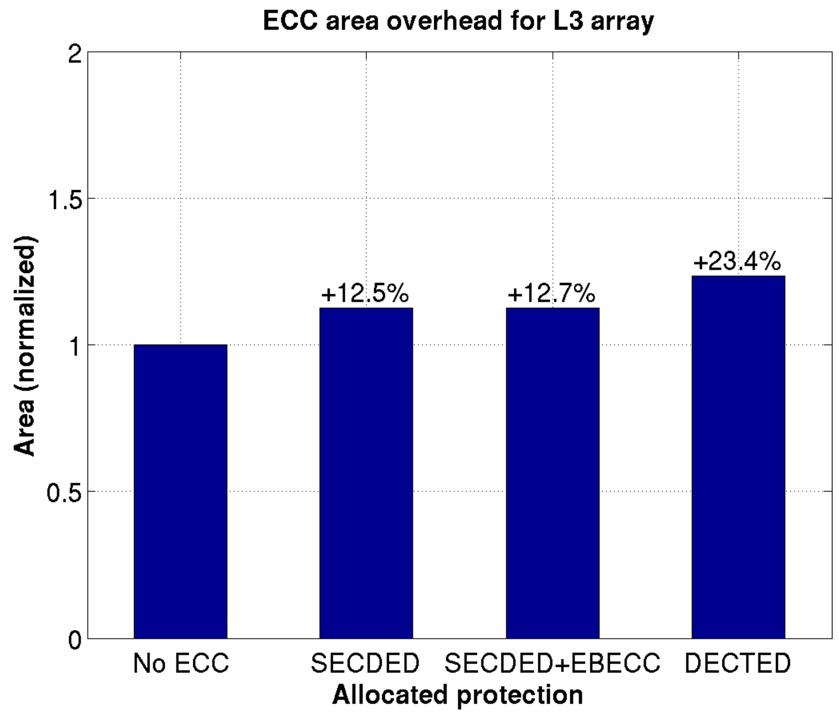
Certain components of the design have a fixed area overhead independent of the array size:

- ECC encoder, decoder
- Vertical parity, erasure storage arrays
- BIST logic for vertical parity, erasure

However, the parity bits' area overhead is always proportional to the array size (see Fig-



(a) L2 cache array: 8 Mb



(b) L3 cache array: 100 Mb

Figure 6.6: Practical area overhead for a typical L2 and L3 array. Sizes selected based on Intel Xeon 7100 dual-core (65nm) [13].

ure 3.6). Thus, the (72, 64) SECDED code incurs 12.5% overhead while the (79, 64) DECTED code incurs 23.4% overhead, a wide gap which EB-ECC reduces.

It is important to note here that the prototype testchip contained early versions of EB-ECC (including vertical parity protection [35, 45]) and address decoder failure detection; accordingly, with the development of TOSA (see Section 4.5.1), it may be more appropriate to use TOSA for runtime read latency erasure detection rather than the bitline XNOR for large, dense arrays which may benefit from small-swing sensing. However, the area overhead of TOSA is higher and partially proportional to the array size in a way that the bitline XNOR is not. Amortized over a 128-cell column, the TOSA incurs approximately 6% increased local array area overhead. Larger columns would incur lower local array area overhead. Even given these caveats, the EB-ECC scheme is estimated to have area savings over DECTED protection (for columns of 128 cells or more).

6.5.2 Local array overhead

As implemented, the runtime read latency erasure detection scheme is accomplished by outputting the inverted versions of both bitlines and taking the XNOR to form the erasure globally; this is done to reduce local array area overhead. The technique mainly incurs its overhead in local routing; by increasing the number of vertically routed signals in M4 from 2 (DIN, DOUT) to 3 (DIN, DOUT, DOUTB). However, this increase is transparent since the routing in M2 is more congested (VDD, VSS, BL, BLB).

6.5.3 Architectural latency overhead

Increasing the memory's read path latency by the use of erasure coding may or may not incur system-level overhead. Whether this increase is transparent or not depends on the critical path latency of the memory which is being protected. At 0.6 ns (19 FO4 in this 65nm technology), the SECDED and EB-ECC overhead may fit within a single cycle's latency, in which case its system-level overhead would be comparable to SECDED ECC alone. Otherwise, the added latency overhead will be on the critical path and thus not transparent to the architecture.

6.6 Testing

Tests have been run at an external clock frequency of 10 MHz through a data acquisition PCI card. The experimental design called for the following tests:

1. Access main memory and verify read of 10-bit ROM address as well as 72-bit written data word.
2. Access supplemental memories.
3. Enable SECDED ECC and demonstrate correction of single-bit errors.
4. Adjust main memory self-timing to reduce column I/O delays and drive read and write operations past the point of latency failure, demonstrating failures (with ECC disabled) and correction (with ECC enabled).
5. Enable vertical parity correction and demonstrate correction of two-bit errors.
6. Enable erasure protection and demonstrate correction of two-bit errors with error vector mask in CAM.
7. Enable erasure protection and demonstrate correction of two-bit errors on-the-fly using runtime erasure detection through XNOR of bitlines sampled at the end of the read-before-write phase.
8. Enable on-chip/internal clock generator and determine maximum frequency of BIST controller, using vector and log storage memories.

During testing of packaged chips, we have gained access to the main ECC-protected memory (1) and each of the supplemental memories used for vertical parity (2). We verified correction of single-bit errors in the main memory (3) and adjusted self-timing (4). However, no configuration of self-timing has caused read latency failures to occur on any of the packaged chips, which limits the ability to exercise the run-time erasure detection circuitry. Thus it is impossible to verify the functionality of the runtime read latency erasure detection circuits on the fabricated design.

The vertical parity control (5) operates in two phases: during normal operation with vertical parity enabled, every write to the protected SRAM is intercepted by the vertical parity controller and the vertical parity memory is updated with the XOR of the written data, the read-before-write value of the word that was written, and the current word in the vertical parity array[1]. This vertical parity update has been observed to be functioning properly. On the event of an uncorrectable error detected by the horizontal SECDED ECC, the vertical parity correction procedure is to begin. The vertical parity controller is to read through each row with the same 5-bit low-order address in the array, reconstructing the proper value of the word in error from the vertical parity array and the protected SRAM array. At the conclusion of this sequence, the vertical parity controller writes back the corrected data to the array. However, due to a high capacitance data input node in the array which was improperly driven with a small buffer by the BIST logic, the delay to properly charge the input nodes is high, and when the writeback cycle is reached, the corrected data from the vertical parity controller is not written back correctly. This can be fixed with a small adjustment to the driving gate of the data input nodes but was not noticed during verification.

Furthermore, the proprietary internal clock generator supplied by Freescale has not responded to configuration and thus we are limited to an external clock frequency of 10 MHz and have limited ability to exercise the design at-speed. However, the efficacy and performance benefits of the included ECC units are shown with overheads determined through design and simulation.

6.7 Summary

In this chapter we discussed a testchip implementation of multiple reliability improvement techniques, including ROM address bits to detect address decoder failures, vertical parity, and erasure

[1]If the valid bit of the word written to is 0, then the XOR omits the previous read-before-write value in the protected SRAM as its contents are assumed to be the zero vector, which is the initialized value of the vertical parity array.

coding with two sources of erasures (offline erasures from CAM and runtime erasures from bit-line XNOR).

Chapter 7

Conclusions and future work

Errors that occur at run-time are pernicious and there are fewer options available to address them than those that are caught at time of manufacture. BIST and BISR mechanisms, or else some means of prevention, must be implemented to handle these run-time failures. Techniques to prevent failures, or to detect and correct them, provide a much-needed capability to the designer of reliable memory systems.

7.1 Contributions

This thesis puts forward the following novel contributions.

- Extension of the widely-used communications technique of erasure coding to memory, and a discussion of the overheads and benefits that can be realized.
- Design and test of prototype testchip in 55nm bulk CMOS containing an erasure-based ECC system (EB-ECC) for reliable SRAM operation. To the best of our knowledge this testchip represents the first implementation of vertical parity correction or erasure coding for SRAM.
- A method of runtime erasure detection for small-swing bitlines using ternary-output sense amplifiers which improves upon the bitline XNOR technique which is only suitable for

paired full-swing bitlines.

- A robust method of address decoder failure detection using a wordline comparator, which achieves 3% area improvement over ROM address bits for a 512-row array. Justification for this method of address decoder failure detection is provided by a study of particle strike locality in an industrial row decoder layout.

7.2 Future research directions

In this section, we discuss new research directions revealed by the work presented in this thesis, which may be useful for future researchers.

7.2.1 Implications of erasure coding with multiple error correcting codes

Both the benefits and area/power overheads of erasure coding increase linearly with the use of higher minimum Hamming distance codes. Latency overhead is only slightly increased versus the code itself. For those cache designs which already pay the overheads necessary for a multiple error correction code, either multiple cycles latency or large area overhead, employing erasure coding in addition to that multiple error correction code requires roughly a doubling in area/power and a slight latency increase for the masking and erasure detection circuitry. In return for these overheads, the number of correctable erasures is double the number of random bit errors correctable with the code itself. For example, while a DECTED code with d_{min}=6 can correct two random bit errors, with the addition of erasure coding, the same code can correct four erasures. The erasure locations are masked to all-zeros and all-ones and run through two DECTED decoders in parallel. The output with fewer bits of error is correctable and is used as the corrected output.

7.2.2 Peripheral logic protection

A fuller examination of the potential effects of particle strikes on the peripheral logic would involve fault simulation not using the pessimistic assumption that any particle strike causes a stuck-at for a full cycle, but rather to perform transient simulations using a switched capacitor holding a charge equivalent to the charge injected during the particle strike event. A flowchart for this more complete study of particle strike locality is shown in Figure 7.1.

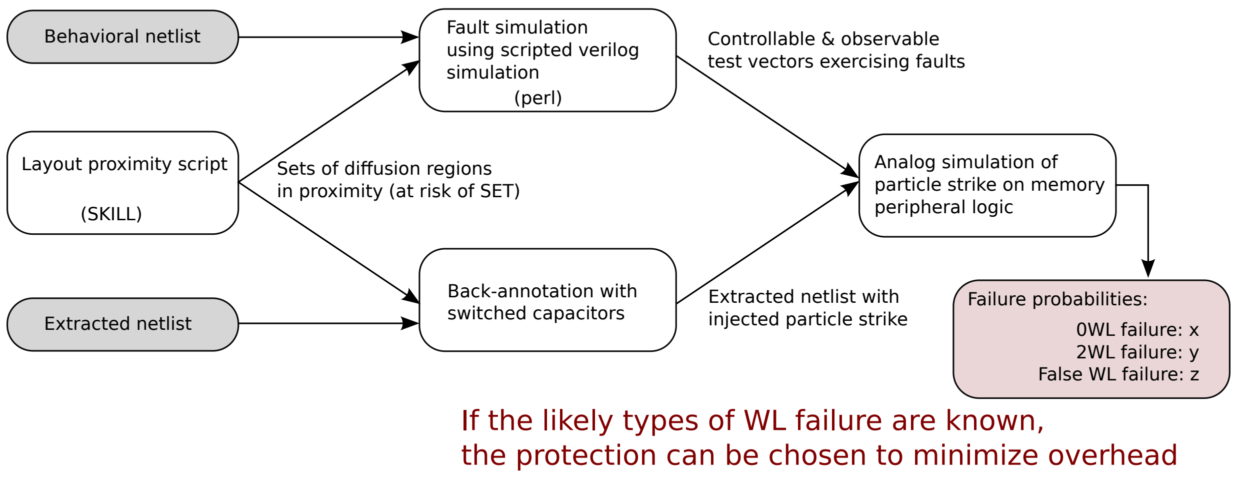

Behavioral netlist
Layout proximity script (SKILL)
Extracted netlist
Fault simulation using scripted verilog simulation (perl)
Sets of diffusion regions in proximity (at risk of SET)
Back-annotation with switched capacitors
Controllable & observable test vectors exercising faults
Analog simulation of particle strike on memory peripheral logic
Extracted netlist with injected particle strike
Failure probabilities:
0WL failure: x
2WL failure: y
False WL failure: z
If the likely types of WL failure are known, the protection can be chosen to minimize overhead

Figure 7.1: Flowchart of full particle strike study

7.2.3 Soft-decision decoding beyond read latency erasures

The concept of erasures as discussed in Section 4.1 was explored in this thesis in the context of read latency failures, since they are detectable at the moment they occur on the bitlines. The other failure modes of read disturb failure, write failure, and hold disturb failure may occur silently in existing cell designs, since they do not have an external manifestation and most SRAM cells are inherently binary in nature. In the non-volatile memory world, there exist multi-level cellss (MLCs) capable of storing more than a single bit of information; these cells fail in a deterministic way and thus soft information regarding the possible failure is available for use with certain decoders, such as those used for low-density parity check (LDPC) codes [9, 10]. It may be possible to construct an SRAM cell which provides similar soft information in the form of erasures in the event of one or more of the three other failure modes which are not currently detectable by the work of this thesis.

Appendix A

Methodology

In the 65nm model files used for simulation of the designs in Chapters 4 & 6, the following global and local parameters are defined for variation:

Process parameters affected by global variations (and modeled under the "process" category):

- Poly and active critical dimension (CD) for transistors and resistors (etching and lithography variation)
- Oxide thickness for transistors and capacitors
- Threshold voltage for transistors and sheet resistances for resistors (implant variation)
- Junction capacitance (area, perimeter)
- Diode current

Local instance parameters affected by local variations (and modeled under the "mismatch" category):

- Threshold voltage and mobility for transistors
- Sheet resistance for resistors

Bibliography

[1] Mohamed H Abu-Rahma and Mohab Anis. Variability in nanometer technologies and impact on SRAM. In *Nanometer Variation-Tolerant SRAM*, pages 5–47. Springer, 2013. URL `http://link.springer.com/chapter/10.1007/978-1-4614-1749-1_2`. 1

[2] R. Dean Adams. *High Performance Memory Testing: Design Principles, Fault Modeling and Self-Test*. Kluwer, 2003. 4.4

[3] B.S. Amrutur and M.A. Horowitz. A replica technique for wordline and sense control in low-power SRAM's. *IEEE J. Solid-State Circuits*, 33(8):1208–1219, August 1998. doi: 10.1109/4.705359. 3

[4] Umut Arslan, Mark P. McCartney, Mudit Bhargava, Xin Li, Ken Mai, and Lawrence T. Pileggi. Variation-Tolerant SRAM Sense-Amp Timing Using Configurable Replica Bitlines. *Proceedings of IEEE Custom Integrated Circuits Conference*, 30:415–418, 21–24 September 2008. doi: 10.1109/CICC.2008.4672108. 2.2.3, 2.3.1, 3, 4.5.4

[5] M. Bhargava, M.P. McCartney, A. Hoefler, and Ken Mai. Low-overhead, digital offset compensated, SRAM sense amplifiers. In *Custom Integrated Circuits Conference, 2009. CICC '09. IEEE*, pages 705–708, 2009. doi: 10.1109/CICC.2009.5280732. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5280732`. (document), 2.2.3, 4.5.1, 4.6

[6] Mudit Bhargava, YK Chong, Vincent Schuppe, Bikas Maiti, Martin Kinkade, Hsin-Yu Chen, Andy W Chen, Sanjay Mangal, Jacek Wiatrowski, Gerald Gouya, Abhishek Bara-

dia, Sriram Thyagarajan, and Gus Yeung. Low VMIN 20nm embedded SRAM with multi-voltage wordline control based read and write assist techniques. In *VLSI Circuits Digest of Technical Papers, 2014 Symposium on*, pages 1–2, 2014. doi: 10.1109/VLSIC.2014.6858412. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6858412`. 3.1

[7] D. Blaauw, S. Kalaiselvan, K. Lai, Wei-Hsiang Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull. Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 400–622, 2008. doi: 10.1109/ISSCC.2008.4523226. 4.5.4

[8] Michael Bushnell and Vishwani D Agrawal. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, volume 17. Springer, 2000. 2.2

[9] Yu Cai. *NAND flash: characterization, analysis, modelling, and mechanisms*. PhD thesis, Carnegie Mellon University, 2013. 7.2.3

[10] Yu Cai, E.F. Haratsch, M. McCartney, and Ken Mai. FPGA-based solid-state drive prototyping platform. In *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, pages 101–104, 2011. doi: 10.1109/FCCM.2011.28. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5771258`. 7.2.3

[11] B. H. Calhoun, Yu Cao, Xin Li, Ken Mai, L. T. Pileggi, R. A. Rutenbar, and K. L. Shepard. Digital Circuit Design Challenges and Opportunities in the Era of Nanoscale CMOS. *Proc. IEEE*, 96(2):343–365, 2008. doi: 10.1109/JPROC.2007.911072. (document)

[12] T. Calin, M. Nicolaidis, and R. Velazco. Upset hardened memory design for submicron CMOS technology. *IEEE Trans. Nucl. Sci.*, 43(6):2874–2878, 1996. doi: 10.1109/23.556880. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=556880`. 5.1, 5.2

[13] J. Chang, Ming Huang, J. Shoemaker, J. Benoit, Szu-Liang Chen, Wei Chen, Siufu Chiu,

R. Ganesan, G. Leong, V. Lukka, S. Rusu, and D. Srivastava. The 65-nm 16-MB Shared On-Die L3 Cache for the Dual-Core Intel Xeon Processor 7100 Series. *IEEE J. Solid-State Circuits*, 42(4):846–852, 2007. ISSN 0018-9200. doi: 10.1109/JSSC.2007.892185. (document), 1, 4.4, 6.5.1, 6.6

[14] L. Chang, R.K. Montoye, Y. Nakamura, K.A. Batson, R.J. Eickemeyer, R.H. Dennard, W. Haensch, and D. Jamsek. An 8T-SRAM for variability tolerance and low-voltage operation in high-performance caches. *IEEE J. Solid-State Circuits*, 43(4):956–963, 2008. doi: 10.1109/JSSC.2007.917509. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4476480`. 2.1.1, 3.2

[15] Ihao Chen and A.J. Strojwas. A methodology for optimal test structure design for statistical process characterization and diagnosis. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 6(4):592–600, 1987. doi: 10.1109/TCAD.1987.1270307. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1270307`. 2.3.1

[16] B. Cheng, S. Roy, and A. Asenov. The impact of random doping effects on CMOS SRAM cell. In *Solid-State Circuits Conference, 2004. ESSCIRC 2004. Proceeding of the 30th European*, pages 219–222, 2004. doi: 10.1109/ESSCIR.2004.1356657. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1356657`. 2.2.3

[17] Y.T. Chiu, Y.F. Wang, Y.-H. Lee, Y.C. Liang, T.C. Wang, S.Y. Wu, C.C. Hsieh, and K. Wu. Analysis of the reliability impact on high-k metal gate SRAM with assist-circuit. In *Reliability Physics Symposium, 2014 IEEE International*, 2014. doi: 10.1109/IRPS.2014.6861171. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6861171`. 3.1

[18] P. E. Dodd, M. R. Shaneyfelt, J. R. Schwank, and G. L. Hash. Neutron-induced latchup in SRAMs at ground level. In *Proceedings of IEEE Int. 41st Annual Reliability Physics Symp*, pages 51–55, 2003. doi: 10.1109/RELPHY.2003.1197720. IRPS Neutron Induced Latchup Sandia Reference received from David Burnett at Freescale 2011-02-28. 2.2.2

[19] P.E. Dodd, M.R. Shaneyfelt, J.R. Schwank, and G.L. Hash. Neutron-induced soft errors, latchup, and comparison of SER test methods for SRAM technologies. In *Proceedings of International Electron Devices Meeting*, pages 333–336, 2002. doi: 10.1109/IEDM.2002.1175846. 2.2.2

[20] Varghese George, Sanjeev Jahagirdar, Chao Tong, K. Smits, Satish Damaraju, Scott Siers, Ves Naydenov, Tanveer Khondker, Sanjib Sarkar, and Puneet Singh. Penryn: 45-nm Next Generation Intel® Core™ 2 Processor. In *Solid-State Circuits Conference, 2007. ASSCC '07. IEEE Asian*, pages 14–17, 2007. doi: 10.1109/ASSCC.2007.4425784. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4425784`. 1

[21] N. Gierczynski, B. Borot, N. Planes, and H. Brut. A new combined methodology for write-margin extraction of advanced SRAM. In *Microelectronic Test Structures, 2007. ICMTS '07. IEEE International Conference on*, pages 97–100, 2007. doi: 10.1109/ICMTS.2007.374463. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4252413`. 2.3.1

[22] F. Hamzaoglu, U. Arslan, N. Bisnik, S. Ghosh, M.B. Lal, N. Lindert, M. Meterelliyoz, R.B. Osborne, Joodong Park, S. Tomishima, Yih Wang, and K. Zhang. A 1Gb 2GHz embedded DRAM in 22nm tri-gate CMOS technology. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pages 230–231, 2014. doi: 10.1109/ISSCC.2014.6757412. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6757412`. 2.1

[23] R. Heald and P. Wang. Variability in sub-100nm SRAM designs. In *IEEE/ACM International Conference on Computer Aided Design*, pages 347–352, 7–11 November 2004. 2.3.1

[24] V.G. Hemmady and S.M. Reddy. On the repair of redundant RAMs. In *Design Automation, 1989. 26th Conference on*, pages 710–713, 1989. doi: 10.1109/DAC.1989.203492. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1586476`. 2.2, 3.5

[25] W Daniel Hillis. New computer architectures and their relationship to physics or why computer science is no good. *International Journal of Theoretical Physics*, 21(3-4):255–262, 1982. URL `http://link.springer.com/article/10.1007/BF01857728`. 4

[26] K. Honda, K. Miyaji, S. Tanakamaru, S. Miyano, and K. Takeuchi. Elimination of half select disturb in 8T-SRAM by local injected electron asymmetric pass gate transistor. In *Custom Integrated Circuits Conference (CICC), 2010 IEEE*, pages 1 –4, Sept. 2010. doi: 10.1109/CICC.2010.5617440. 2.1.1, 3.2

[27] M. Y. Hsiao. A Class of Optimal Minimum Odd-weight-column SEC-DED Codes. *IBM Journal of Research and Development*, 14(4):395–401, 1970. doi: 10.1147/rd.144.0395. 3.4.1

[28] S. Ishikura, M. Kurumada, T. Terano, Y. Yamagami, N. Kotani, K. Satomi, K. Nii, M. Yabuuchi, Y. Tsukamoto, S. Ohbayashi, T. Oashi, H. Makino, H. Shinohara, and H. Akamatsu. A 45 nm 2-port 8T-SRAM Using Hierarchical Replica Bitline Technique With Immunity From Simultaneous R/W Access Issues. *IEEE J. Solid-State Circuits*, 43(4):938–945, 2008. doi: 10.1109/JSSC.2008.917568. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4476491`. 2.1.1, 3.2

[29] T. Jhaveri, V. Rovner, Lars Liebmann, L. Pileggi, A.J. Strojwas, and J.D. Hibbeler. Co-optimization of circuits, layout and lithography for predictive technology scaling beyond gratings. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 29(4):509–527, 2010. doi: 10.1109/TCAD.2010.2042882. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5433742`. 2.2.1

[30] J. Keinonen, K. Arstila, and P. Tikkanen. Electronic stopping power of Si and Ge for MeV-energy Si and P ions. *Appl Phys Lett*, 60(2):228–230, 1992. doi: 10.1063/1.106972. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4878290`. 2.2.2

[31] G. Keskin, J. Proesel, and L. Pileggi. Statistical modeling and post manufacturing configuration for scaled analog CMOS. In *Custom Integrated Circuits Conference (CICC), 2010 IEEE*, pages 1–4, 2010. doi: 10.1109/CICC.2010.5617625. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5617625`. 2.2.3

[32] M. Khellah, Nam Sung Kim, Yibin Ye, D. Somasekhar, T. Karnik, N. Borkar, G. Pandya, F. Hamzaoglu, T. Coan, Yih Wang, K. Zhang, C. Webb, and V. De. Process, Temperature, and Supply-Noise Tolerant 45nm Dense Cache Arrays With Diffusion-Notch-Free (DNF) 6T SRAM Cells and Dynamic Multi-Vcc Circuits. *IEEE J. Solid-State Circuits*, 44(4): 1199–1208, 2009. doi: 10.1109/JSSC.2009.2014015. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4804978`. 3.1

[33] Daeyeon Kim, V. Chandra, R. Aitken, D. Blaauw, and D. Sylvester. Variation-aware static and dynamic writability analysis for voltage-scaled bit-interleaved 8-T SRAMs. In *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, pages 145–150, 2011. doi: 10.1109/ISLPED.2011.5993627. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5993627`. 3.3

[34] Ilyoung Kim, Y. Zorian, G. Komoriya, H. Pham, F.P. Higgins, and J.L. Lewandowski. Built in self repair for embedded high density SRAM. In *Proceedings of International Test Conference*, pages 1112–1119, 18–23 October 1998. doi: 10.1109/TEST.1998.743312. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=743312`. 2.2, 3.5

[35] Jangwoo Kim, Nikos Hardavellas, Ken Mai, Babak Falsafi, and James Hoe. Multi-bit error tolerant caches using two-dimensional error coding. In *IEEE/ACM International Symposium on Microarchitecture*, pages 197–209, 2007. doi: 10.1109/MICRO.2007.4408256. (document), 1, 3.4.2, 3.4.3, 3.7, 6, 6.1, 6.5.1

[36] Jangwoo Kim, Mark McCartney, Ken Mai, and Babak Falsafi. Modeling SRAM Failure Rates to Enable Fast, Dense, Low-Power Caches. *IEEE Workshop on Silicon Errors in Logic - System Effects*, 24–25 March 2009. (document), 2.3.1

[37] Jangwoo Kim, Hyunggyun Yang, Mark P. McCartney, Mudit Bhargava, Ken Mai, and Babak Falsafi. Building fast, dense, low-power caches using erasure-based inline multi-bit ECC. In *Dependable Computing (PRDC), 2013 IEEE 19th Pacific Rim International Symposium on*, pages 98–107, 2013. doi: 10.1109/PRDC.2013.19. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6820845`. (document), 2.3.1, 4.2, 4.5

[38] T. Kobayashi, K. Nogami, T. Shirotori, and Y. Fujimoto. A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architecture. *IEEE J. Solid-State Circuits*, 28(4):523–527, 1993. doi: 10.1109/4.210039. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=210039`. 4.5.1

[39] S. Komatsu, M. Yamaoka, M. Morimoto, N. Maeda, Y. Shimazaki, and K. Osada. A 40-nm low-power SRAM with multi-stage replica-bitline technique for reducing timing variation. In *Custom Integrated Circuits Conference, 2009. CICC '09. IEEE*, pages 701–704, 2009. doi: 10.1109/CICC.2009.5280731. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5280731`. 3

[40] N. Kurd, M. Chowdhury, E. Burton, T.P. Thomas, C. Mozak, B. Boswell, M. Lal, A. Deval, J. Douglas, M. Elassal, A. Nalamalpu, T.M. Wilson, M. Merten, S. Chennupaty, W. Gomes, and R. Kumar. Haswell: A family of IA 22nm processors. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pages 112–113, 2014. doi: 10.1109/ISSCC.2014.6757361. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6757361`. 1, 2.1

[41] N.A. Kurd, S. Bhamidipati, C. Mozak, J.L. Miller, T.M. Wilson, M. Nemani, and M. Chowdhury. Westmere: A family of 32nm IA processors. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 96–97, 2010. doi: 10.1109/ISSCC.2010.5434033. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5434033`. 3.4.1

[42] K. Lilja, M. Bounasser, S.-J. Wen, R. Wong, J. Holst, N. Gaspard, S. Jagannathan,

D. Loveless, and B. Bhuva. Single-event performance and layout optimization of flip-flops in a 28-nm bulk technology. *IEEE Trans. Nucl. Sci.*, 60(4):2782–2788, 2013. doi: 10.1109/TNS.2013.2273437. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6575154`. 5.1

[43] S. Lin and D.J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983. (document), 3.4, 3.4.2

[44] Mark McCartney, Jangwoo Kim, Mudit Bhargava, Babak Falsafi, James C. Hoe, and Kenneth Mai. Two-Dimensional Coding for SRAM Yield Improvement and Soft Error Resilience. *SRC Techcon*, 2008. URL `https://www.src.org/library/publication/p023647/`. 6, 6.1

[45] Mark McCartney, Mudit Bhargava, Craig Teegarden, Cagla Cakir, and Kenneth Mai. Erasure Coding Techniques for SRAM Yield Improvement. *SRC Techcon*, 2011. URL `https://www.src.org/library/publication/p060325/`. 1, 6, 6.5.1

[46] A. Meixner and J. Banik. Weak Write Test Mode: an SRAM cell stability design for test technique. In *Proceedings of International Test Conference*, pages 309–318, 20–25 October 1996. doi: 10.1109/TEST.1996.556976. 4.4

[47] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and Kee Sup Kim. Robust system design with built-in soft-error resilience. *Computer*, 38(2):43–52, 2005. doi: 10.1109/MC.2005.70. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1401773`. 5.2

[48] Todd K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005. 4.1, 4.3

[49] G Moore. The MOS transistor as an individual device and in integrated arrays. In *IRE International Convention Record*, volume 13, pages 44–52. IEEE, 1966. 1

[50] Gordon E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, April 19, 1965, pp.114 ff. *IEEE Solid-State Circuits*

Society Newsletter, 11(5):33–35, 2006. doi: 10.1109/N-SSC.2006.4785860. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4785860`. 1

[51] D. Morris, V. Rovner, L. Pileggi, A. Strojwas, and Kaushik Vaidyanathan. Enabling application-specific integrated circuits on limited pattern constructs. In *VLSI Technology (VLSIT), 2010 Symposium on*, pages 139–140, 2010. doi: 10.1109/VLSIT.2010.5556202. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5556202`. 2.2.1

[52] S. Mukhopadhyay, H. Mahmoodi, and K. Roy. Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(12):1859–1880, December 2005. doi: 10.1109/TCAD.2005.852295. 2.3.1, 2.3.1

[53] R. Naseer and J. Draper. Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs. In *Proceedings of 34th European Solid-State Circuits Conference ESSCIRC 2008*, pages 222–225, 2008. doi: 10.1109/ESSCIRC.2008.4681832. (document), 3.1, 3.2, 3.5

[54] National Science Foundation. Cyber-Physical Systems (CPS) Program Solicitation, 2014. URL `http://www.nsf.gov/pubs/2014/nsf14542/nsf14542.htm`. 1

[55] Y. Niki, A. Kawasumi, A. Suzuki, Y. Takeyama, O. Hirabayashi, K. Kushida, F. Tachibana, Y. Fujimura, and T. Yabe. A digitized replica bitline delay technique for random-variation-tolerant timing generation of SRAM sense amplifiers. *IEEE J. Solid-State Circuits*, 46(11): 2545–2551, 2011. doi: 10.1109/JSSC.2011.2164294. 3, 4.5.4

[56] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou. Yield-aware cache architectures. In *IEEE/ACM International Symposium on Microarchitecture*, pages 15–25, December 2006. doi: 10.1109/MICRO.2006.52. (document)

[57] M. J. M. Pelgrom, H. P. Tuinhout, and M. Vertregt. Transistor matching in analog CMOS applications. In *Proceedings of International Electron Devices Meeting IEDM '98 Techni-*

cal Digest, pages 915–918, December 6–9, 1998. doi: 10.1109/IEDM.1998.746503. 2.2.3

[58] M.J.M. Pelgrom, A.C.J. Duinmaijer, and A.P.G. Welbers. Matching properties of MOS transistors. *IEEE J. Solid-State Circuits*, 24(5):1433–1439, October 1989. doi: 10.1109/JSSC.1989.572629. 2.2.3, 3.3

[59] L. Peters. ITRS: Yield Enhancement: The End Justifies the Means. Technical report, Semiconductor Research Corporation, Jan 2008. 2.2.1

[60] H. Pilo, C. Barwin, G. Braceras, C. Browning, S. Lamphier, and F. Towler. An SRAM Design in 65-nm Technology Node Featuring Read and Write-Assist Circuits to Expand Operating Voltage. *IEEE J. Solid-State Circuits*, 42(4):813–819, April 2007. doi: 10.1109/JSSC.2007.892153. 3.1

[61] D. W. Plass and Y. H. Chan. IBM POWER6 SRAM arrays. *IBM Journal of Research & Development*, 51(6):747–756, November 2007. 1

[62] Jan Rabaey. *Low power design essentials*. Springer, 2009. 1

[63] R.J. Riedlinger, R. Bhatia, L. Biro, B. Bowhill, E. Fetzer, P. Gronowski, and T. Grutkowski. A 32nm 3.1 billion transistor 12-wide-issue Itanium® processor for mission-critical servers. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pages 84–86, 2011. doi: 10.1109/ISSCC.2011.5746230. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5746230`. 3.4.1

[64] M. L. Rieger, J. P. Mayhew, and S. Panchapakesan. Layout design methodologies for sub-wavelength manufacturing. In *Proceedings of Design Automation Conference*, pages 85–88, 2001. 2.2.1

[65] A. Romano. *Applied Statistics for Science and Industry*. Allyn & Bacon, 1977. 2.3.1

[66] D. Rossi, N. Timoncini, M. Spica, and C. Metra. Error correcting code analysis for cache memory high reliability and performance. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6, 2011. doi: 10.1109/DATE.2011.5763257. URL `http://ieeexplore.ieee.org/stamp/stamp.`

jsp?arnumber=5763257. 3.4.1

[67] S. Rusu, Simon Tam, H. Muljono, J. Stinson, D. Ayers, Jonathan Chang, R. Varada, M. Ratta, and S. Kottapalli. A 45nm 8-core enterprise Xeon® processor. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 56–57, 2009. doi: 10.1109/ISSCC.2009.4977305. 3.4.1

[68] S. Rusu, H. Muljono, D. Ayers, S. Tam, Wei Chen, A. Martin, Shenggao Li, S. Vora, R. Varada, and E. Wang. Ivytown: A 22nm 15-core enterprise Xeon® processor family. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pages 102–103, 2014. doi: 10.1109/ISSCC.2014.6757356. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6757356. 3.4.1

[69] E. Seevinck, F. J. List, and J. Lohstroh. Static-noise margin analysis of MOS SRAM cells. *IEEE J. Solid-State Circuits*, 22(5):748–754, 1987. doi: 10.1109/JSSC.1987.1052809. (document), 2.3.1, 2.8

[70] M. Spica and T.M. Mak. Do we need anything more than single bit error correction (ECC)? In *Proceedings of Records of the 2004 International Workshop on Memory Technology, Design and Testing*, pages 111–116, 9–10 August 2004. doi: 10.1109/MTDT.2004.1327993. (document)

[71] B. Stackhouse, B. Cherkauer, M. Gowan, P. Gronowski, and C. Lyles. A 65nm 2-Billion-Transistor Quad-Core Itanium Processor. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 92–598, 2008. doi: 10.1109/ISSCC.2008.4523072. 1, 4.5.3

[72] K. Takeda, H. Ikeda, Y. Hagihara, M. Nomura, and H. Kobatake. Redefinition of write margin for next-generation SRAM and write-margin monitoring circuit. In *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, pages 2602–2611, 2006. doi: 10.1109/ISSCC.2006.1696326. 2.3.1

[73] N. Verma and A.P. Chandrakasan. A 256 kb 65 nm 8T subthreshold SRAM employing

sense-amplifier redundancy. *IEEE J. Solid-State Circuits*, 43(1):141–149, 2008. doi: 10.1109/JSSC.2007.908005. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4443213`. 4.5.1

[74] Jiajing Wang, S. Nalam, and B.H. Calhoun. Analyzing static and dynamic write margin for nanometer SRAMs. In *Low Power Electronics and Design (ISLPED), 2008 ACM/IEEE International Symposium on*, pages 129–134, 2008. doi: 10.1145/1393921.1393954. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5529055`. 2.3.1

[75] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010. ISBN 0321547748, 9780321547743. 3.4.2

[76] C. Wilkerson, Hongliang Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and Shih-Lien Lu. Trading off Cache Capacity for Reliability to Enable Low Voltage Operation. In *Proceedings of 35th International Symposium on Computer Architecture*, pages 203–214, 21–25 June 2008. doi: 10.1109/ISCA.2008.22. 2.3.1, 3.4.2

[77] Chris Wilkerson, Alaa R Alameldeen, Zeshan Chishti, Wei Wu, Dinesh Somasekhar, and Shih-Lien Lu. Reducing cache power with low-cost, multi-bit error-correcting codes. *ACM SIGARCH Computer Architecture News*, 38(3):83–93, 2010. URL `http://dl.acm.org/citation.cfm?id=1815973`. 3.4.2

[78] Chung-Yu Wu and Liow Yu-Yee. A new dynamic ternary sense amplifier for 1.5-bit/cell multi-level low-voltage CMOS DRAMs. In *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, volume 1, pages 47–50, 1999. doi: 10.1109/ISCAS.1999.777802. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=777802`. 4.5.1

[79] M. Yamaoka, N. Maeda, Y. Shinozaki, Y. Shimazaki, K. Nii, S. Shimada, K. Yanagisawa, and T. Kawahara. Low-power embedded SRAM modules with expanded margins for writ-

ing. In *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, pages 480–611, 2005. doi: 10.1109/ISSCC.2005.1494078. 3.1

[80] Josh Yang, Baosheng Wang, Yuejian Wu, and A. Ivanov. Fast detection of data retention faults and other SRAM cell open defects. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 25(1):167–180, 2006. doi: 10.1109/TCAD.2005.852680. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1564312`. 4.5

[81] K. Yano, T. Hayashida, and T. Sato. Analysis of SER Improvement by Radiation Hardened Latches. In *Dependable Computing (PRDC), 2012 IEEE 18th Pacific Rim International Symposium on*, pages 89–95, 2012. doi: 10.1109/PRDC.2012.9. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6385074`. 5.2

[82] Xiaoyin Yao, L. T. Clark, D. W. Patterson, and K. E. Holbert. Single event transient mitigation in cache memory using transient error checking circuits. In *Proceedings of IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, 2010. doi: 10.1109/CICC.2010.5617439. (document), 3.6, 3.8, 5.2, 6.2

[83] J. F. Zeigler, H. P. Muhlfeld, C. J. Montrose, H. W. Curtis, T. J. OGorman, and J. M. Ross. Accelerated testing for cosmic soft-error rate. *IBM Journal of Research and Development*, 40(1):19–39, January 1996. 2.2.2

[84] K. Zhang, U. Bhattacharya, Zhanping Chen, F. Hamzaoglu, D. Murray, N. Vallepalli, Yih Wang, Bo Zheng, and M. Bohr. A 3-GHz 70-Mb SRAM in 65-nm CMOS technology with integrated column-based dynamic power supply. *IEEE J. Solid-State Circuits*, 41(1): 146–151, 2006. doi: 10.1109/JSSC.2005.859025. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1564355`. 2.3.1, 3.1

[85] B. Zimmer, Seng Oon Toh, Huy Vo, Yunsup Lee, O. Thomas, K. Asanovic, and B. Nikolic. SRAM assist techniques for operation in a wide voltage range in 28-nm CMOS. *IEEE Trans. Circuits Syst. II*, 59(12):853–857, 2012. doi: 10.1109/TCSII.2012.2231015. URL `http://ieeexplore.ieee.org/stamp/stamp.`

`jsp?arnumber=6424019`. 3.1