Resilient Design Techniques for Improving Cache Energy Efficiency

Brian Zimmer

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-182
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-182.html

December 1, 2016

Acknowledgement

This work was funded by ST Microelectronics, TSMC, DARPA PERFECT Award Number HR0011-12-2-0016, Berkeley Wireless Research Center sponsors, ASPIRE sponsors, Intel ARO, AMD, and the NVIDIA fellowship.

Resilient Design Techniques for Improving Cache Energy Efficiency

by

Brian Matthew Zimmer

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Borivoje Nikolić, Co-chair
Professor Krste Asanović, Co-chair
Professor Daniel Tataru

Summer 2015

Resilient Design Techniques for Improving Cache Energy Efficiency

Abstract

Resilient Design Techniques for Improving Cache Energy Efficiency

by

Brian Matthew Zimmer

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Borivoje Nikolić, Co-chair

Professor Krste Asanović, Co-chair

Improving energy efficiency is critical to increasing computing capability, from mobile devices operating with limited battery capacity to servers operating under thermal constraints. The widely accepted solution to improving energy efficiency is dynamic voltage and frequency scaling (DVFS), where each block in a design operates at the minimum voltage required to meet performance constraints at a given time. However, variation-induced SRAM bitcell failures in caches at low voltage limit voltage scaling—and therefore energy-efficiency improvements—in advanced process nodes. Analyzing and modeling bitcell failures is necessary to develop resiliency techniques that prevent or tolerate SRAM failure to improve the minimum operating voltage of caches.

This work demonstrates a holistic approach that uses both circuit-level and architecture-level design techniques to improve low-voltage operation of SRAM. A simulation framework and experimental measurements from a 28nm testchip explore failure mechanisms of SRAM bitcells. The simulation framework is further utilized to evaluate the effectiveness of circuit-level SRAM assist techniques using dynamic failure metrics. New circuit-level techniques that use replica timing are developed to make SRAM macros more resilient to process variation. An architecture-level error model is developed to translate bitcell failure probability to yield, and to evaluate a variety of error-correcting code (ECC) and redundancy-based resiliency techniques. New resiliency schemes, named dynamic column redundancy (DCR) and bit bypass (BB), are proposed to tolerate a high bitcell failure rate with low overhead.

The methodology and proposed schemes were validated with five different 28nm chips. The RAVEN1 testchip measured in-situ threshold voltage variation of 30,000 bitcells and was used to analyze the effect of random telegraph noise on failures. The RAVEN2 testchip explored a single-p-well bitcell design that can compensate for global process variation. The RAVEN3 and RAVEN3.5 chips included processors with on-chip switched-capacitor voltage conversion, and resilient SRAM macros with circuit-level techniques that enable operation down to 0.45V. The SWERVE processor included architecture-level techniques to avoid failing cells, and decreases energy by over 30% with only 2% area overhead, and includes in-situ

pipelined ECC to measure the contribution of intermittent SRAM error sources such as random telegraph noise and aging.

Overall, this dissertation describes a general methodology that is used to evaluate resilient design technique effectiveness for different process and architecture assumptions, and proposes a new set of resilient design techniques that lower the minimum operating voltage of caches with low overhead.

To my wife, Anjali.

Contents

List of Figures

List of Tables

Acknowledgments

While writing this dissertation, I was constantly reminded that none of this work would have been possible without the support of so many friends and colleagues. I have had a wonderful time at Berkeley over the last five years because I have been fortunate enough to work with so many great people.

First, I would like to thank my co-advisors: Borivoje Nikolić and Krste Asanović. I couldn't have asked for better advisors. I never imagined that I would work for advisors who would spend so much time and energy to make sure that I succeed. Their combined technical expertise from circuits to software was vital for this work. Both are incredibly loyal, care deeply about the personal and professional development of their students, and have created an excellent culture at the BWRC and ASPIRE lab—I can't thank them enough.

Bora's group has many great students working in diverse research areas. Seng's dissertation formed a great foundation for my work, and his continuing advice has helped guide my research decisions; I wish we had overlapped for a longer period of time. Olivier Thomas helped teach me circuit design and collaborated on two tape-outs when I was a brand new student. The RAVEN team was a joy to work with. I credit many of the skills I learned to being able to work in such a supportive, friendly, collaborative, and creative environment—thank you so much Ruzica Jevtić, Ben Keller, Stevo Bailey, Martin Cochet, Jaehwa Kwak, Milovan Blagojević, Alberto Puggelli, and Pi-Feng Chiu. Building chips takes a lot of time and energy, and the huge contributions from this team made RAVEN3 and RAVEN3.5 possible. And to all of the other members of Bora's group over the years that provided moral support and friendship—Katerina Papadopoulou, Matt Weiner, Rachel Hochman, Amanda Pratt, Luis Esteban Hernandez, Milos Jorgovanovic, Dusan Stepanovic, Vinayak Nagpal, Ji-Hoon Park, Nicolas Le Dortz, Dajana Danilovic, Sameet Ramakrishnan, Sharon Xiao, Angie Wang, Amy Whitcombe, Antonio Puglielli, Vladimir Milovanovic, Charles Wu, Nicholas Sutardja, and John Wright—thank you.

The work from Krste's group has been incredible, and the development of RISC-V and processor IP has been invaluable for my research. Andrew Waterman's advice and feedback has been very helpful. Rimas Avižienis helped tape-out RAVEN1, and taught me a lot about the CAD toolflow. I would like to thank Yunsup Lee for being my inspiration and role model throughout my graduate career—his great attitude, effectiveness, vision, and helpfulness are unparalleled, and I feel incredibly fortunate that we were able to work together so closely. Thank you to the rest of the group helped build and support the RTL used in all five of my tape-outs Henry Cook, Huy Vo, Scott Beamer, Christopher Celio, Donggyu Kim, Palmer Dabbelt, Eric Love, Martin Maas, Albert Ou, and Colin Schmidt.

Outside of my advisors' groups, I would like to thank the other students that helped along the way. Nathan Narevsky always seemed to know the answers to all of my questions, and I had a great time developing curriculum and teaching EE141 with him. Stephen Twigg was always there to help me with my Chisel coding. Thanks to Nattapol Damrongplasit and Ying Qiao for teaching me about transistor measurement.

I had a great team of undergraduate researchers help me at various steps in my graduate career. Behzad Boroujerdian helped explore ECC codes; Brian Jenkins investigated BIST algorithms to diagnose error causes; Joey Talia designed an early version of a replica timing script; and Taylor Vincent found the first RTN in RAVEN1.

The Berkeley Wireless Research Center has been an awesome working environment, made possible by awesome staff. Brian Richards has been a joy to work with, patiently sharing his vast expertise throughout my entire graduate career. Thank you to Leslie Nishiyama, Deirdre McAuliffe-Bauer, Bira Coelho, Olivia Nolan, and Sarah Jordan for taking care of all of the administrative work so that I could focus on research. James Dunn is dependable and courageous in tackling new tasks, and has been a great help with IT support and PCB design. Daniel Burke and Fred Burghardt provided much-appreciated lab support.

If having two advisors wasn't enough, many other professors at Berkeley graciously provided help and advice whenever I asked. Elad Alon is extremely helpful and always has answers to our tough technical questions. I would like to thank Cari Kaufman and David Patterson for serving on my qualification committee. Tsu-Jae King Liu provided advice for the RAVEN1 design and measurements. John Wawrzynek and Vladimir Stojanovic were great instructors that I was fortunate to be a teaching assistant for. And thanks to Daniel Tataru for taking the time to be on my dissertation committee.

Industry feedback has been critical to making sure my work was relevant; thank you to Tom Burd, Mark Rowland, and Stephen Kosonocky, for taking the time to provide feedback on my research over the years. I would also like to thank my mentors at NVIDIA—Tom Gray, Bill Dally, Ethan Frazier, Mahmut Sinangil, Brucek Khailany, and John Poulton—for providing an awesome internship experience that taught me many useful skills and renewed my motivation for circuits research halfway through my PhD.

My research was funded by DARPA PERFECT Award Number HR0011-12-2-0016, Berkeley Wireless Research Center sponsors, ASPIRE sponsors, Intel ARO, AMD, and the NVIDIA fellowship. ST Microelectronics has donated four tape-outs, and Philippe Flatresse and Andreia Cathelin provided support that was crucial to the success of my research. TSMC donated a 28nm tape-out for the SWERVE project.

Outside of school, I was fortunate to have a great group of friends and family. Thank you Samarth Bhargava, Siddharth Dangi, Suraj Gowda, Kevin Messer, Ryan Going, Cameron Rose, and Dan Calderone for all of the great times! My twin brother, Michael Zimmer, ended up in the same PhD program as I did at the same time at the same school, which turned out to be one of the luckiest things that has happened to me. Spending time with him and his wife Sarah while sharing the ups and downs of our voyage through graduate school has been an incredible gift. My sister Julie has always given me endless love and encouragement. My parents, Paul and Jean, provided the best childhood anyone could ask for—encouraging me to develop and follow my passions and supporting all of my decisions.

And last I would like to thank my wife, Anjali. I would never have completed this dissertation without her constant love and support, and I am incredibly grateful to have her by my side.

I will cherish my memories at Berkeley because of all of the people who helped along the way—thank you all so, so much.

Chapter 1

Introduction

This chapter introduces the overall topic of the dissertation—how resilient design techniques can be used to improve cache energy efficiency. First, the motivation for this work is introduced by describing the importance of SRAM to modern digital systems and summarizing obstacles to further improvements. Next, background information about SRAM and its usage is reviewed, and related work is surveyed to provide information about other potential approaches and solutions. Last, the overall organization of this dissertation is provided.

1.1 Motivation

SRAM is the most popular dense memory structure in CMOS, due to the combined attributes of small size and high speed. While the memory element of an SRAM bitcell, a cross-coupled inverter pair, is also used in latches and flip-flops, the SRAM architecture assumes multiple bits are accessed at the same time, and thereby exploits a dense array organization and shared peripheral circuitry to amortize area overhead and enable more aggressive design rules. Embedded DRAM achieves higher area density due to the single transistor storage element, but requires extra process steps, needs refresh operations, and operates more slowly.

Processors are some of the most common digital systems, and SRAM makes up a large proportion of their chip area. Large on-chip caches in processors greatly improve performance and energy efficiency by avoiding unnecessary communication with off-chip memory, so a large portion of die area is necessarily devoted to SRAM bitcells. Processors will remain a workhorse of future integrated circuits—in devices such as laptops, smartphones, or the emerging markets of wearable electronics, internet of things devices, and automotive—so SRAM will endure as a vital system building block that requires constant improvements to performance, energy efficiency, and area. As parallelism increases and communication becomes more expensive, local memory that is dense and fast becomes even more critical to improve performance and save energy.

To minimize energy in digital systems, either the effective switched capacitance or voltage needs to be reduced, as switching energy is proportional to CV^2. The effective switched capacitance (C) is reduced with technology scaling and improved architectures that minimize the amount of capacitance that needs to be switched to finish a given task. The advantage of changing the operating voltage (V) is that it can be done during operation. When performance requirements are high, the voltage is increased to reduce transistor delay, but when lower performance is sufficient, the voltage is reduced to save energy in a scheme called dynamic voltage and frequency scaling (DVFS). The quadratic relationship between voltage and active energy makes voltage scaling an especially powerful means of reducing energy—a voltage decrease of only 200mV from 0.9V to 0.7V corresponds to a 40% reduction in energy.

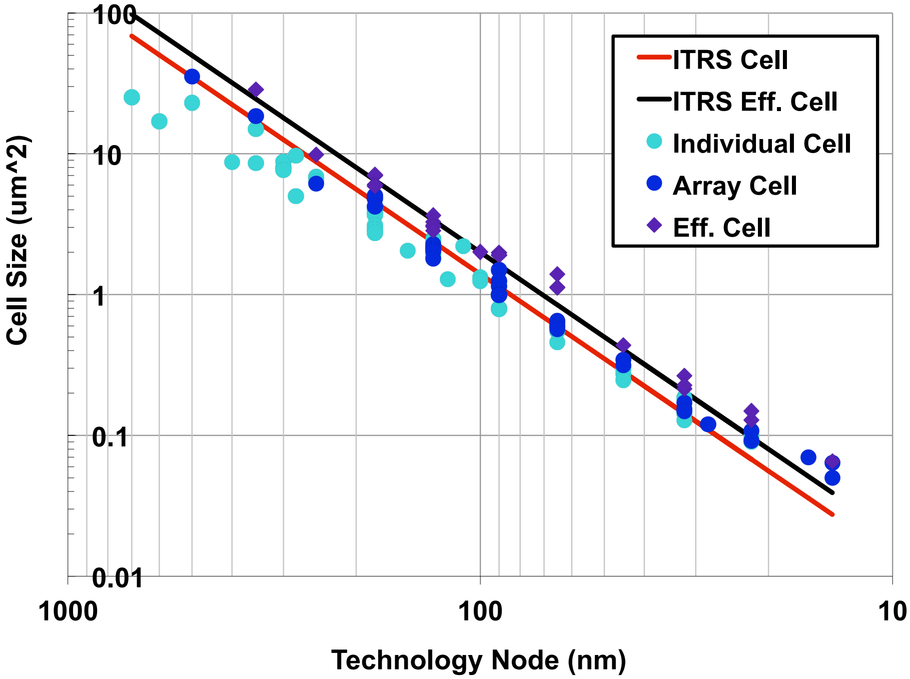


Figure 1.1: Measured bitcell area scaling (for individual pre-production cells, cells within arrays, and the effective cell sizes including overhead of peripheral circuitry) versus ITRS scaling predictions for decreasing technology nodes.

SRAM sets the minimum operating voltage of a system (V_{min}), due to the billions of extremely small transistors inside SRAM bitcells, and reliance on strength ratios to preserve functionality. To support large SRAM-based caches on-chip, transistor dimensions have been pushed to the extreme limit within each bitcell. SRAM bitcell size has maintained Moore's law scaling over many technology nodes, as shown in Figure 1.1. However, aggressive bitcell sizing yields transistors that have increasing process variation, and the dramatically increasing overall count of bitcells increases the probability of process variation sufficient to cause SRAM failure at low voltage. The read and write mechanisms of SRAM bitcells require specific strength ratios between devices. Figure 1.2 plots a simplified illustration of why SRAM limits V_{min}. A PMOS and NMOS are connected in series between V_{DD} and ground, forming a voltage divider, and the middle voltage as a percentage of V_{DD} is plotted versus voltage. A bitcell cannot be written when the middle voltage surpasses a threshold. Two variation cases are enumerated: no variation at all, and a shift in transistor threshold voltage representing the strongest PMOS and weakest NMOS out of 1 million cells. At high voltages, process variation has little effect, because the gate voltage is far from the threshold voltage. However at lower voltages, the shift in threshold voltage has a significant impact on transistor strength, and the intermediate voltage surpasses the success threshold—representing a failure event. Therefore, making SRAM cells resilient to process variation is necessary to further reduce V_{min} and improve energy efficiency.

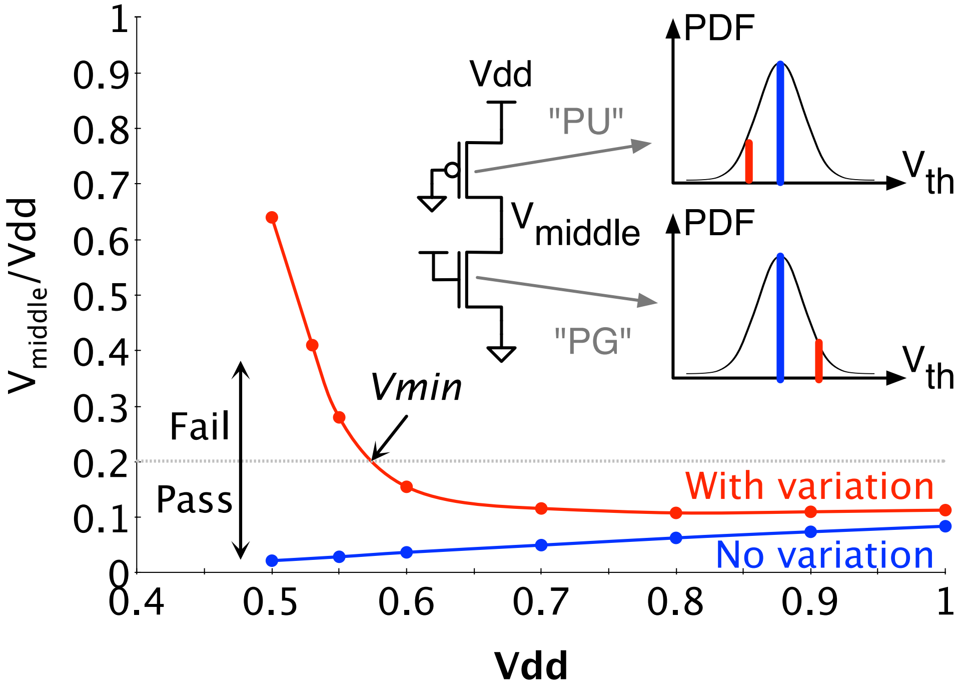


Figure 1.2: Variation only has a major effect on the strength ratio between devices at lower voltages.

One simple solution to avoiding SRAM failures at low voltage is to isolate the SRAM supply from the logic supply in order to keep SRAM at high voltage. However, this approach, sometimes referred to as "dual rail" [2], has a few disadvantages. First, power delivery complexity increases because SRAM arrays are mixed into logic areas. Second, timing verification between two separate voltage domains is complicated and requires additional margining. Third, overall energy efficiency improvements from voltage scaling are limited by the fixed energy consumption of SRAM, especially in L2 or L3 caches where SRAM contributes almost all of the power consumption. For these reasons, this work dismisses dual rail and focuses on the case where SRAM shares the same power domain as surrounding logic.

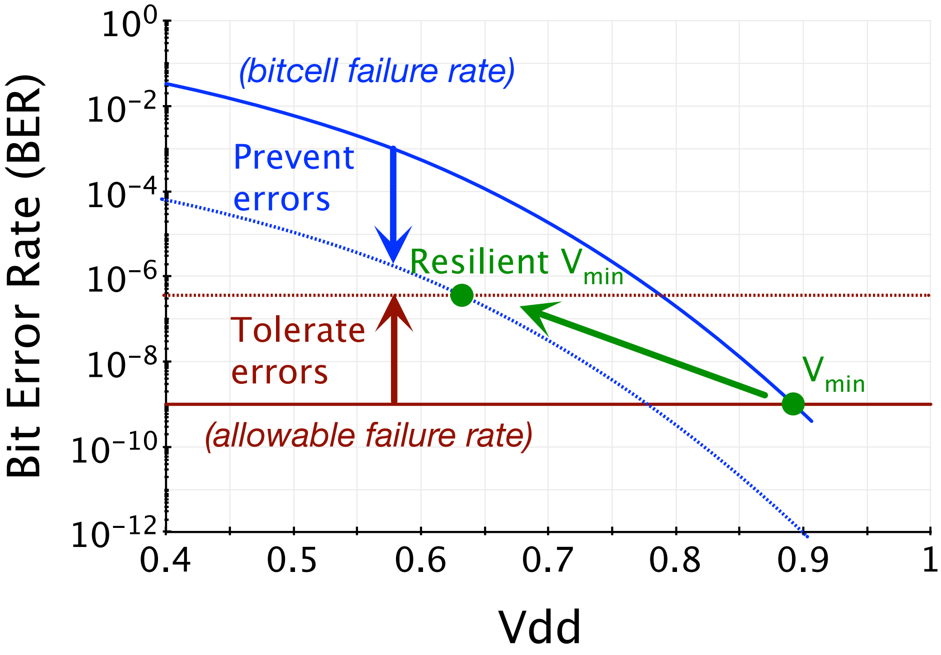


Figure 1.3: Resilient techniques decrease V_{min} by either reducing the bitcell error rate or tolerating failures to increase the acceptable failure rate.

The other solution, and the focus of this work, is to use resilient design techniques to lower the minimum operating voltage of SRAM. Resiliency refers to the ability to tolerate process variation and prevent device non-idealities from causing system failure. Figure 1.3 shows the general idea of resilient design. Bitcell failure rate is plotted versus operating voltage. The number of cells in a design will set a maximum allowable bitcell failure rate, which is represented as a horizontal line on the plot. The intercept of the failure curve and the allowable failure rate represents V_{min} of the design. Two general strategies can be used to reduce V_{min}—either prevent or tolerate bitcell failures. Resiliency techniques that try to *prevent* process variation from causing failure decrease the probability that a bitcell fails at a given voltage, shifting the curve down and moving the intercept to the left, representing a reduction in V_{min}. Resiliency techniques that *tolerate* bitcell failures at a system level increase the maximum allowable bitcell failure rate, shifting the horizontal line up and also moving the intercept to the left, representing another method of reducing V_{min}. Both strategies can be used together for maximum V_{min} reduction.

Each resiliency technique reduces V_{min}, but at the cost of additional overhead in terms of area, energy, delay, and design complexity. Evaluating both effectiveness and overhead of different schemes at a system level is required to choose the appropriate techniques for a particular design.

To summarize, processors need huge SRAM-based on-chip caches that operate at low voltages to maximize energy efficiency and performance in digital systems. Achieving this goal requires using extremely small transistors inside SRAM bitcells that experience large random variations, which are further exacerbated at low voltages. Resiliency techniques that prevent or tolerate errors are necessary to continue innovation in digital systems by improving cache energy efficiency through increased voltage scaling.

1.2 Background

Background knowledge about SRAM and its usage is necessary to understand the motivation and context of various solutions in the following chapters. Due to the holistic approach required to improve energy efficiency in caches, a review of both circuit-level and architecture-level topics is necessary.

1.2.1 Circuit-level Topics

Circuit-level bitcell topics include describing how SRAM cells are designed, explaining how peripheral circuit reads and writes individual bitcells, and demonstrating how transistor variation can cause failures.

Bitcells

Static random-access memory (SRAM) is a form of on-chip memory that requires a power supply to maintain its contents. Most SRAM uses six transistors (6T) to store one bit of information, while other versions such as the eight-transistor (8T) cell use extra transistors to enable multiple reads or writes per cycle. Figure 1.4 shows both cells and some naming conventions: PU for pull-up, PD for pull-down, PG for pass-gate in the 6T and 8T cell, and additionally RPD for read-pull-down, RPG for read-pass-gate in the 8T cell. The PD and PU devices form the storage element of the SRAM cell using positive feedback to hold differential data on the left and right side of the cell. The PG devices enable access to this bit of memory for either reading or writing. Unlike DRAM, no refresh is needed to maintain the state of the cells. If the voltage falls below the retention voltage (V_{ret}), the transistors in the storage element effectively turn off, and the contents of the cell are lost. Therefore in sleep states, the state stored in SRAM must either be moved to another location in order to turn off power to the SRAM, or the voltage must be maintained at a specific V_{ret}.

Figure 1.5 depicts how bitcells are read and written with example waveforms showing successful and unsuccessful operations. For a write operation, the desired cell contents are placed differentially on the bitline, so one side will be high while the other will be low. Writeability failures occur when either the high node cannot be pulled low through the pass gate (because the PG is not stronger than the PU) or the low node cannot be pulled high through the PU (because the PD is stronger than the PU and PG). For a read operation,

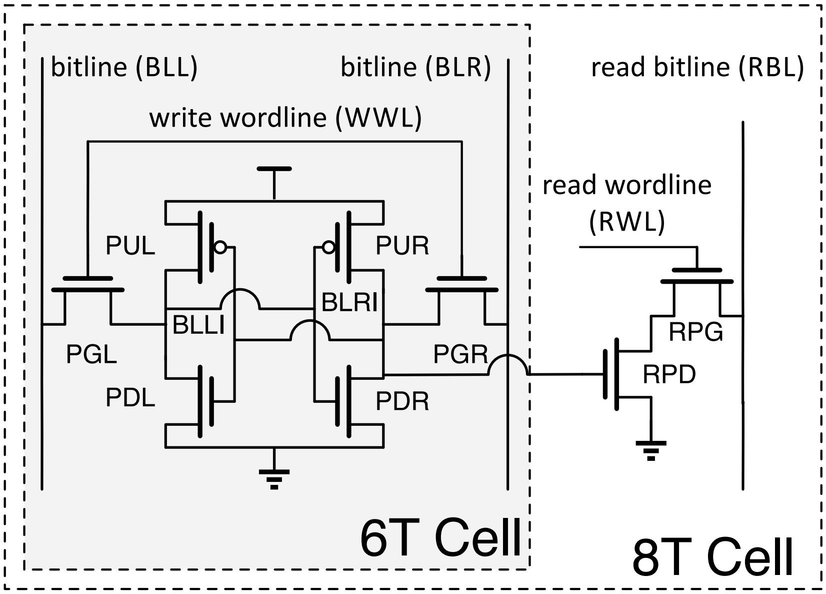
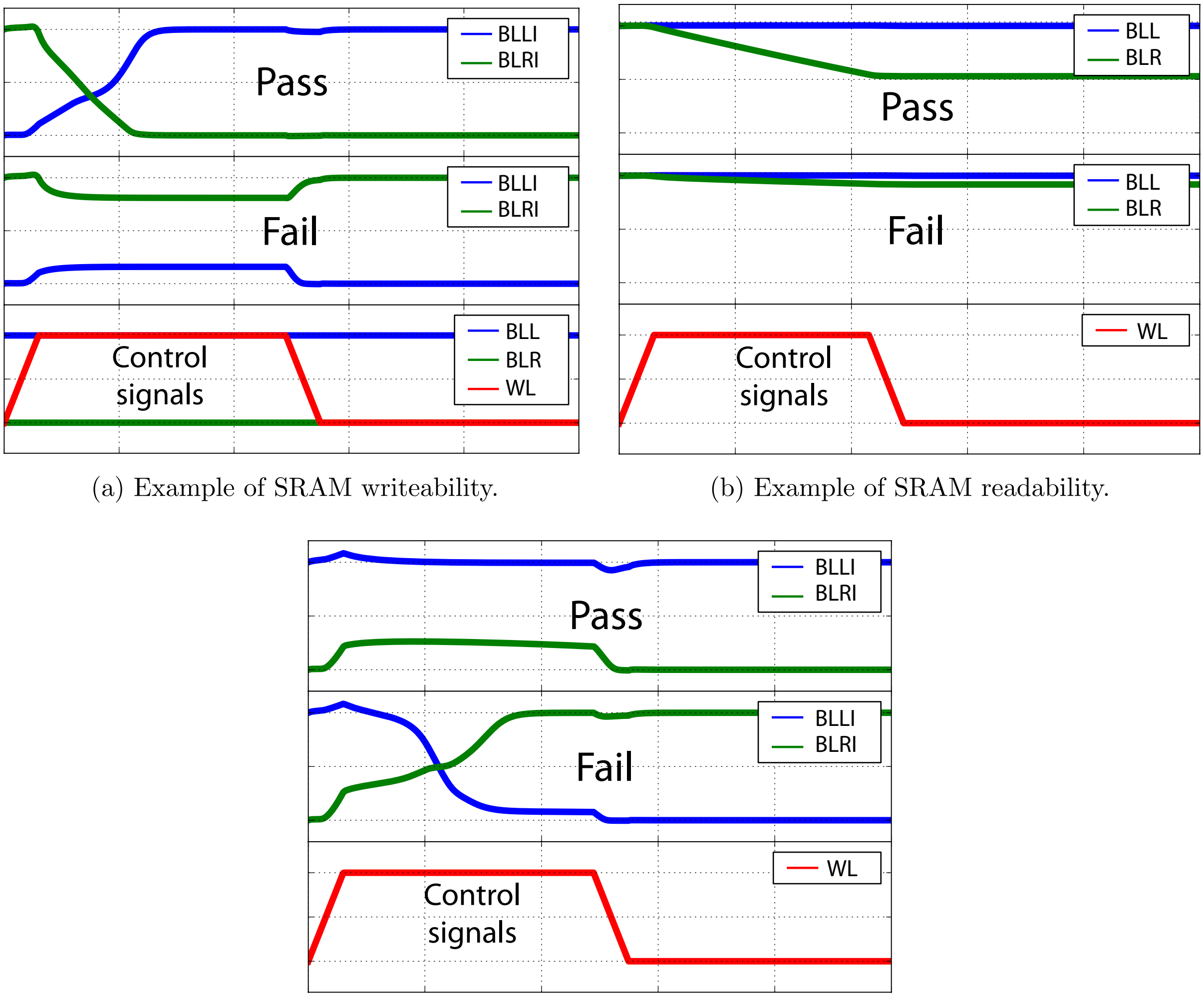


Figure 1.4: 6T and 8T SRAM bitcell schematics and naming convention.

both bitlines will be precharged and left floating at a high value. Readability failures occur when the voltage differential generated on the bitlines does not surpass the sensing offset voltage of the sense amplifiers. Because many bitcells share a single bitline to reduce area overhead, the capacitive loading of the bitline can be quite high, and weak bitcells cannot sink enough read current to discharge the bitline in a given amount of time. Read stability failures occur during either a read access, when the side of the cell holding the low value will experience a voltage bump due to the voltage divider between the PD and PG. If the PD is not sufficiently strong, the voltage can rise above the trip point of the cell and accidentally flip the contents of the cell.

There is an intrinsic conflict between stability and writeability in a bitcell. Large transistors in the storage element (PU and PD) with weak access transistors (PG) would favor stability, while large access transistors and small storage element transistors will favor writeability. In particular, for a cell to be both stable and writable, the PD must be sufficiently stronger than the PG, and the PG must be sufficiently stronger than the PU. Therefore the PG has both a lower and upper bound on strength, and changing PG size will trade-off stability and writeability.

Bitcell failures can be measured with either static or dynamic metrics. Traditionally, static metrics, such as the static noise margin (SNM), were used to estimate V_{min}. However, static metrics assume infinite access time, and therefore are pessimistic measures of read stability, and optimistic measures of writeability [3]. Dynamic metrics, which account for finite access time of bitcells, closely predict actual V_{min} [4], and will be used throughout this work.



(a) Example of SRAM writeability.

(b) Example of SRAM readability.

(c) Example of SRAM read stability.

Figure 1.5: SRAM operation waveforms.

Sources of variation

There are many sources of device parameter variations caused by an imperfect manufacturing process, such as random dopant fluctuations (RDF) or line-edge roughness (LER) [5]. Random (or local) variations such as RDF exist between each transistor in the design, reducing matching in analog differential pairs or ratioed devices in SRAM cells. Additionally, there are systematic variations that affect all devices on the same die, wafer, or lot—causing all devices to have different drive current versus gate voltage characteristics.

While process parameter variations have dozens of physical sources that change many components of transistor performance, variation can be modeled with a first-order approximation by assuming all variation only affects the threshold voltage of the device. Using 3D simulation, the distribution of the threshold voltage was predicted to be Gaussian to about 3 sigma, then is positively skewed [6]. Random or local variation is represented as the variance of the distribution, while systematic variation corresponds to shifting the mean of the distribution. As technology features decrease in size, the contribution of different physical sources of variation changes. In general, smaller technology nodes have increasing variation, because the fewer dopant ions in the channel increases variation from random dopant fluctuations [5].

Array organization

To amortize the area overhead of peripheral circuits used to read and write the SRAM, bitcells are organized into arrays, as shown in Figure 1.6. When reading a word, the entire wordline is turned on along the row direction, and the data in each cell is transferred onto the differential bitlines along the column direction by discharging one of the bitlines on the side of the cell holding a zero value. Sense amplifiers that detect the voltage difference on the bitlines are generally larger than the width of a cell, so bitcells are physically interleaved to only read 1 out of every N bitcells, where N is a power of two. After the read operation is completed, the bitlines are precharged again to prepare for the next operation. During a write operation, one of the bitlines is discharged low, while the other is held high (or left floating high). When the pass gate transistors are turned on by the wordline, the feedback of the internal inverters is overcome by the pass gates to write a new value into the cell.

From a system perspective, an SRAM appears with a logical width w and depth d, where an access reads or writes w bits (a word), there are d unique addresses to store w bits, and there are $w \cdot d$ bits total in an array. In a non-interleaved design, there are d wordlines (rows) for each address, and there are w bitline pairs (columns). The shape of the array can be changed by physically interleaving multiple words on a row and using multiplexers to access the desired set of columns. Arranging more cells on each wordline and bitline improves area density, but also increases energy and delay.

In the simplest implementation, the row decoder is placed on either the left or the right side of the array, the column logic is placed above or below the array, and the control logic is placed in one corner. Performance can be increased at the expense of area and additional metal layers by segmenting the wordline, the bitline, or both. In a segmented

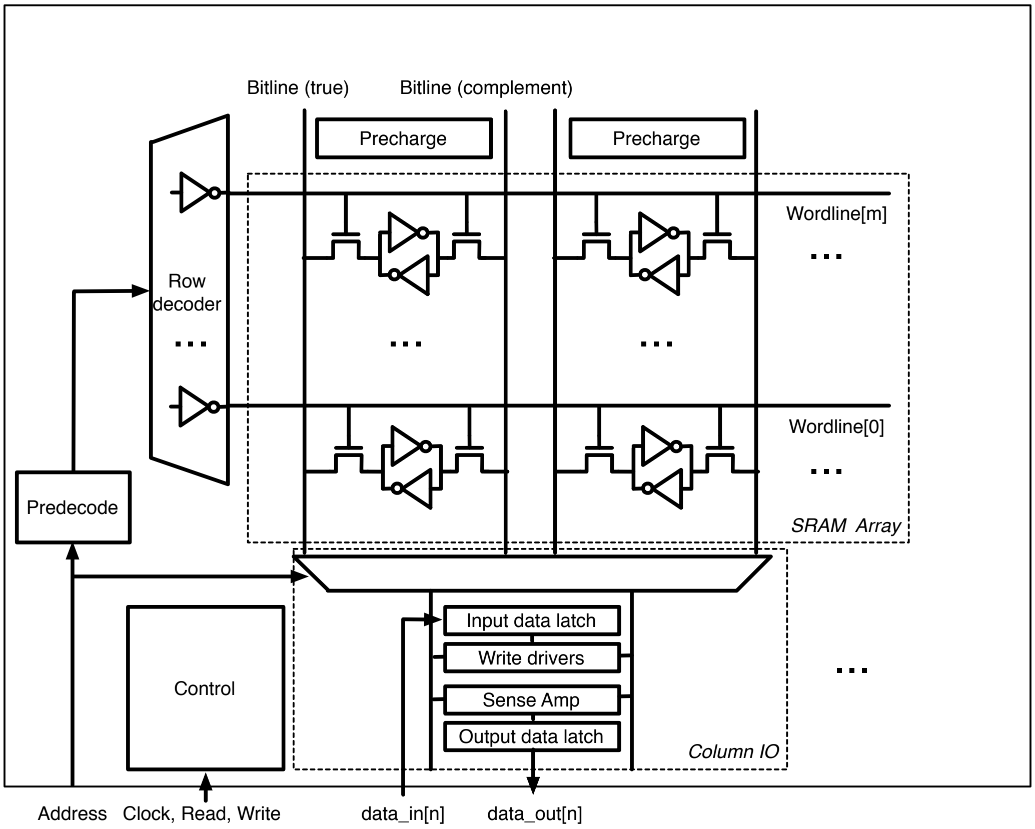


Figure 1.6: Organization of SRAM bitcells into an array.

design, the row decoders can be placed in the middle of the array horizontally to reduce wordline length, and the column IO can be placed in the middle of the array vertically to reduce the bitline length by half [7]. To further increase performance, the bitline can be segmented into multiple hierarchical local bitlines that drive larger global bitlines at the expense of area overhead [8],[9].

1.2.2 Architecture-level Topics

A level of abstraction separates circuit-level and architecture-level topics. Once an SRAM array is designed and verified at the circuit-level, it is abstracted as a memory that holds a specific number of addressable words of data with a maximum cycle time. These SRAM macros are used at the architecture-level to build larger structures, such as caches, that don't depend on specific circuit issues within each SRAM macro.

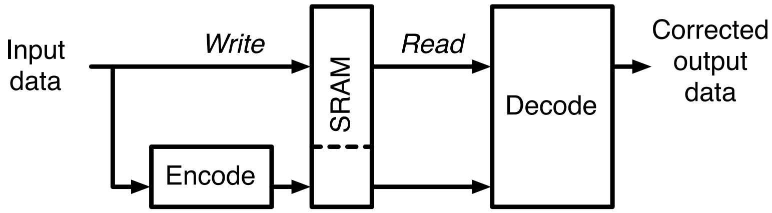


Figure 1.7: Generic error-correcting codes (ECC) stores extra bits to repair faults during SRAM read or write.

Cache organization

SRAMs have many different uses in digital system, and are generally used for memory elements with thousands of bits or more. On-chip caches in processors contain many millions of SRAM cells; there are up to 300 million bits in modern 22nm Intel Xeon processors [10].

A cache attempts to hide memory latency by appearing as a much larger memory than it actually is by storing frequently accessed data [11]. Only the lower bits of the address map into each row in the memory, so different data with differing upper address bits could conflict and map to the same constrained resource. Caches generally use SRAM for both the tag and data portion of the cache, where the tag array is a small portion of the overall area.

The caching technique is repeated hierarchically, with usually two to three levels of caching, where the smallest and fastest (level-one or L1 cache) use large bitcells and smaller bitlines to reduce delay, and the last level of the cache (L2 or L3 caches) use the smallest bitcells possible to increase capacity. These multiple levels make caching a particularly interesting design target because each level has very different SRAM design constraints.

Error-correcting codes (ECC)

Error-correcting codes, commonly referred to as ECC, detect and correct bit flips during every read operation by decoding additional bits stored with the words. Figure 1.7 explains how a generic error-correcting code protects an SRAM array from failure. ECC detects bit flips from all failure sources including intermittent and soft errors. Numerous ECC codes offer differing correction and detection capabilities. Most commonly, SECDED corrects a single bit per word, white DECTED corrects up to two bits per word. Upgrading from a SECDED code to DECTED code increases the complexity and overhead of the encoder and decoder [12], potentially precluding usage in L1 caches.

1.2.3 Relationship between Voltage, Energy and Delay

Improving energy efficiency is crucial for all systems and workloads, as Equation 1.1 illustrates with a high-level abstraction.

$$\text{Performance}\left(\frac{\text{ops}}{\text{s}}\right) = \text{Power}\left(\frac{\text{J}}{\text{s}}\right) \cdot \text{Energy Efficiency}\left(\frac{\text{ops}}{\text{J}}\right) \tag{1.1}$$

If performance is constrained, the only way to reduce power is to improve energy efficiency, while if power is constrained (either for thermal reasons or limited battery capacity), the only way to improve performance is to improve energy efficiency. Therefore while mobile and server applications have power budgets two orders of magnitude apart, perform different workloads, and have widely varying performance constraints, using voltage scaling to improve energy-efficiency is beneficial in both systems.

Energy consumption in digital systems comes from two sources: switching (or active) and leakage. Additionally, to understand energy-efficiency, the distinction between energy and power (energy per unit time) is important. Switching energy per clock cycle is equal to the total capacitance switched times the voltage squared ($E_{switch} = C_{\mathit{effective}}V^2$), and can be converted to power by estimating the frequency ($P_{switch} = C_{\mathit{effective}}V^2 \cdot f$). The effective switched capacitance changes each cycle based on the activity of the design. Leakage is generally measured as power, but can be converted into energy using the frequency of operation. Leakage power is dependent on process parameters (current versus gate voltage), the voltage, logic gate mixture, and logic state.

Decreasing the voltage will decrease both the switching and leakage power, but the transistors will have less drive current and the frequency of operation will be reduced. Energy-efficiency will improve as voltage decreases, until the point at which the dramatic increase in transistor delay near the threshold voltage causes more leakage to be integrated per cycle than saved switching energy, creating an optimal energy-efficiency point shown in Figure 1.8. The optimal point is usually well below the V_{min} of SRAM, so using resiliency to reduce V_{min} to the optimal point is the goal of this work. The minimum energy point exists near the threshold voltage, but also is a function of the ratio between switching and leakage energy—if switching energy is a higher proportion of total energy either implicitly from the design or due to higher activity, the minimum energy point will be at a lower voltage.

1.3 Related Work

Many resilient design techniques have been developed to improve SRAM V_{min}. Understanding the effect of process variation is vital both for yield improvement and V_{min} reduction [5]. SRAM bitcell failures due to persistent process variations (such as random dopant fluctuation) have been studied using both simulation and experimental measurements. Simulation-based analysis relies on either static metrics [13] or accelerated-Monte-Carlo simulation to predict rare failure events [14], [15], [16]. Measurement-based analysis uses a combination of individual transistor measurements and at-speed test generators to locate bitcell failures [17], [4]. In addition to persistent process variations, soft error strikes can cause bitcell failures [18], and random telegraph noise and aging change transistor strength over time [19].

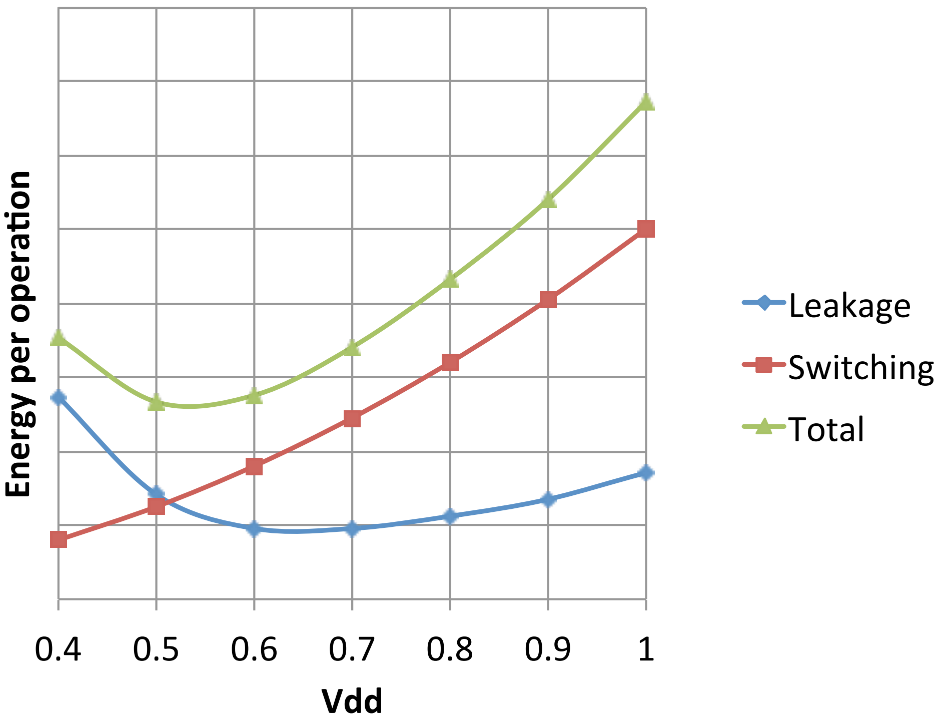


Figure 1.8: Energy efficiency is optimal when the sum of energy from leakage and switching energy is minimized.

Circuit-level resiliency techniques have been proposed to prevent extremely weak or strong transistors from causing bitcell failures. Adding two extra transistors to each bitcell to create the single-ended 8T cell decouples the read and write operation to improve low voltage operation [20], [21], [22]. Circuit assist techniques change wordline [8], bitline [23] [24], or cell supply voltages [7] on a cycle-by-cycle basis to strengthen or weaken particular devices during each operation, and have been shown to significantly reduce V_{min}.

Architecture-level resiliency use redundancy or error correction to repair or avoid bitcell failures. Redundancy-based techniques guarantee working memory cells to compensate for failing cells [25], [26], [27]. Error-correction-based techniques encode data with extra bits that are used to detect and correct bit flips when they occur [28], [29], [30].

In general, existing studies have focused on a single layer of abstraction, comparing circuit solutions to other circuit solutions and architecture techniques to other architecture techniques (with a few exceptions, such as [31]). A comprehensive analysis with a common evaluation framework can determine the optimal set of resiliency techniques to lower V_{min} for a given set of process and implementation assumptions.

1.4 Thesis Outline

This work takes a holistic approach that explores resiliency solutions which enable V_{min} reduction of SRAM at different levels of abstraction, verifies the models and techniques with fabricated and measured designs in a modern 28nm process, and evaluates each solution within the context of a full system.

Chapter 2 proposes an error model that translates basic sources of variation and soft error strike models to system yield and FIT as a function of resiliency techniques at both the circuit and architecture level. This model is used in later chapters to gain insight into the effectiveness of different resiliency techniques and provide in-depth analysis and comparison of different ideas. While targeted at SRAM inside cache-like structures, the methodology is easily generalized to any SRAM organization in digital systems.

Chapter 3 explores SRAM failure mechanisms using both simulation and measurement based approaches. Simulation approaches enable a design-time study of the effectiveness of various circuit techniques to improve low voltage SRAM operation. Measurement results from a 28nm testchip validate assumptions about process variation and explore the impact of intermittent sources of variation. These results are crucial for correctly anticipating the behavior of SRAM failures in real silicon for effective design of complete resilient systems.

Chapter 4 proposes a few circuit-level resiliency techniques. A new bitcell, measured in a 28nm design, uses back bias to compensate for systematic process variation. A new macro, part of two more 28nm chips, enables a wide operating range of 0.45V to 1V using a write assist technique and replica timing.

Chapter 5 analyzes the effectiveness of a variety of previously proposed architecture-level resiliency techniques, and proposes new techniques called bit bypass and dynamic column redundancy that significantly increase the maximum allowable bitcell failure rate with very little overhead. Also, a discussion of the analysis summarizes a set of guidelines that provide intuition about different resiliency approaches. Last, redundancy and error-correcting code based techniques are compared using the joint hard and soft error model.

Chapter 6 compiles the results from previous chapters to build a complete processor with resilient cache that avoids bitcell errors to reduce V_{min} and improve energy efficiency. A memory test identifies failing bitcells at different voltages, and error-correcting codes detect both soft and intermittent errors during operation.

Chapter 7 concludes by summarizing the key contributions and suggesting directions of future research.

Chapter 2

Error Models

This chapter describes the error model used throughout this thesis to evaluate different resiliency techniques. The model is holstic, and translates basic physical sources of error to overall cache yield and failures in time while accounting for both circuit and architecture-level resiliency techniques.

Error models are necessary to evaluate the effectiveness of different resiliency schemes. A memory error model was developed to calculate yield and failures-in-time as a function of cache design, resiliency techniques, expected transistor variation, and operating voltage. As shown in Figure 2.1, the model uses both a circuit-level model and a microarchitecture-level model to translate bitcell failure probability at different operating conditions to cache failure probability. The circuit-level model determines the probability that a single bitcell will fail, based on transistor-level simulations of the bitcell affected by random variation under different operating conditions. The architecture-level model assumes bitcells are organized as a cache, and translates the probability of bitcell failure to cache yield. A similar methodology could be used for any organization of bitcells in a digital system, but most SRAM in digital systems are used in caches, so assuming organization into a cache can provide valuable insight and immediately useful results.

The model focuses on the probability that a cache fails, but *not* the probability that a complete system fails; by avoiding assumptions about architecture and activity, the results are general and widely applicable. Higher-level architecture and application-level error-vulnerability models can translate the probability that a cache will fail to the probability that a system will fail [32], but this higher-level analysis is orthogonal to the evaluation of alternative cache designs. Note that the higher levels can only mask soft errors and not hard faults, under the assumption that a system will rely on all parts of the available cache at some point, and so can only decrease failures-in-time (FIT) and not yield.

This model makes two main contributions. First, the model is holistic—modeling failure rates at both the circuit and architecture level, and accounting for interactions between the two different levels of abstraction. Chapter 5 will show that an architecture-level resiliency scheme's effectiveness is very sensitive to circuit-level assumptions, and that accurately modeling bitcell failure probability as a function of voltage is critical.

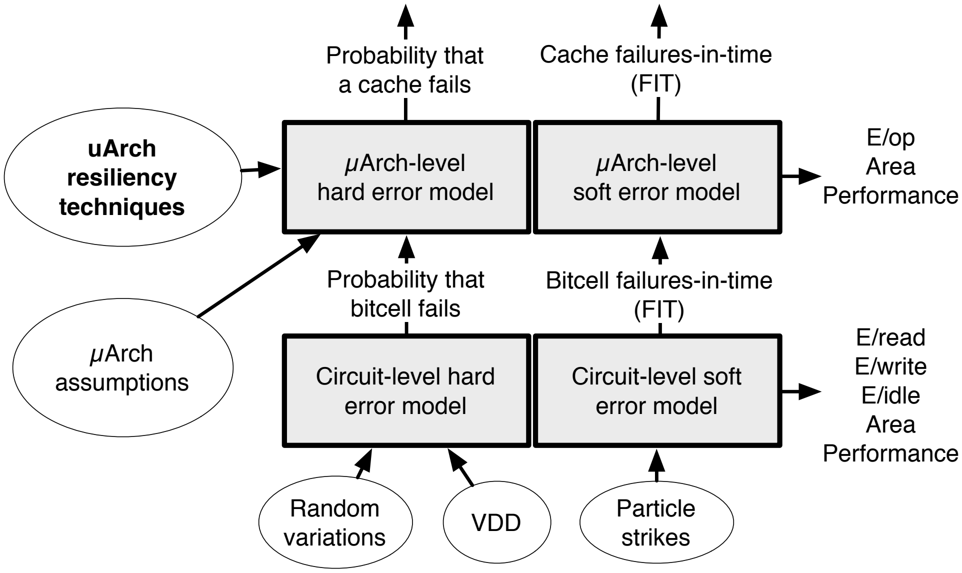


Figure 2.1: Memory error model for hard and soft faults.

Second, the microarchitecture-level error model is general, and not specialized to the particular scheme under evaluation. This flexible approach to modeling resiliency schemes enables easy comparison between many different schemes without sacrificing any accuracy.

2.1 Modeling Sources of Error

It is important to differentiate between different sources of error, as some resiliency techniques can only protect against specific kinds of errors. Even though there are many physical sources of errors, errors are categorized based only on their characteristics.

2.1.1 Hard Faults

Hard faults (also known as persistent errors) are generally caused by an imperfect manufacturing process and persist for the lifetime of a chip. These errors are localized to particular devices. The hard faults analyzed here are *not* yield faults in the bitcells or periphery that cause failures at nominal voltage, but rather bitcells that work well at high voltage and only begin to fail at lower voltages due to variation. The most common sources of variation are line-edge roughness (LER) and random dopant fluctuation (RDF). These effects are modeled as a shift in the threshold voltage of a device, using a Gaussian distribution with a different standard deviation for each process technology. The circuit-level model translates LER and RDF effects into bitcell failure probability as a function of voltage. A built-in-self-test (BIST) that tests the SRAM can identify the location of these failures.

2.1.2 Soft Errors

Soft errors (also known as transient errors) are rare events that cause an error, but do not cause long-term damage. A high-energy particle striking a circuit node results in a voltage pulse that can flip the value of a bitcell. In the model, the case where two or more separate soft error strikes occur in the same cache line is ignored. Detailed analysis showed that the high frequency of cache accesses prevents multiple soft errors from building up in single word, and that physical bitcell interleaving can easily prevent a high-energy multi-bit particle strike from affecting multiple bitcells in the same word.

In a few cases, soft errors can affect the proposed hard-fault model. In some proposed schemes, SECDED is used to correct hard faults without upgrading ECC protection to DECTED, leaving words with hard faults vulnerable to soft errors [31]. An in-depth analysis provided later shows that this design choice can only be justified with particular combinations of assumptions about bitcell failure rates, bitcell FIT, system architecture, system activity, and required system FIT. In general, sharing single-bit ECC between hard faults and soft errors is not acceptable, and so we do not consider this option further in this thesis.

2.1.3 Intermittent Errors

Intermittent errors, such as random telegraph noise (RTN) and aging-related errors, share characteristics of both hard and soft errors. Like soft errors, the probability of occurrence is not 100%, but like hard faults, these defects are bound to specific devices. Because BIST might not detect these faults, repair-based resiliency techniques cannot protect against intermittent errors, and these errors need to be treated as soft errors for failure analysis.

2.2 Circuit-level Hard-fault Model[1]

The circuit-level hard fault model translates random variations to memory bitcell probability of failure for different bitcells, voltages, and circuit-level resiliency techniques. This model is essential to correctly compare resiliency techniques, because the slope of voltage versus bitcell failure probability determines the amount of voltage reduction, and therefore energy-efficiency gains, enabled by different resiliency techniques. A steeper failure slope means that resiliency techniques provide less energy-efficiency improvements than a flatter slope, making it more difficult to justify the additional overhead or complexity of a technique.

Transient simulations are required to quantify the effect of various design decisions on failure rates, because static metrics poorly match reality [4]. A Monte Carlo simulation can be used to measure the effect of variability with transient simulations, but an infeasible number of simulations would be required to find a rare failure event. Importance sampling (IS) enables enormous speedup for Monte Carlo analysis of rare events [14]. The proposed

[1] The content in this section was derived from a publication in TCAS-II 2012 [33].

error modeling methodology uses importance sampling of transient simulations to predict SRAM failure rates at different voltages.

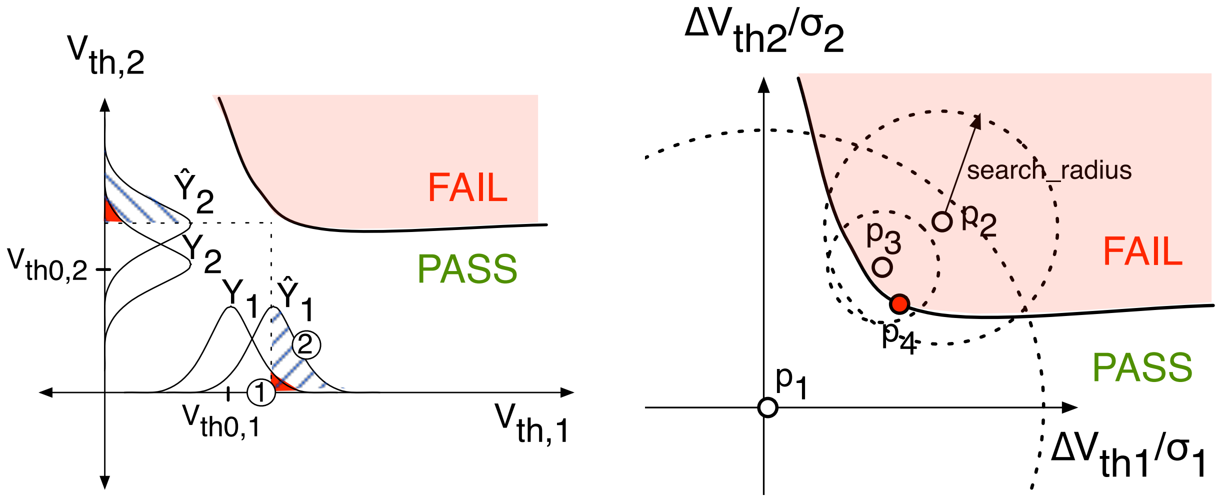
To fairly compare between different voltages, the frequency of the simulations is defined in terms of the fanout-of-4 delay (FO4), which remains constant over all voltages while appropriately adjusting actual operating frequency. For a given technology, this model translates assumptions about random variations and voltage to bitcell failure probability, as shown in Figure 2.13b.

Analysis Methodology

Variation in transistors is modeled in the transistor-level simulation models with distributions that describe variations of various device parameters. Monte Carlo simulation samples from these distributions, and then runs the desired operating point or transient analysis. Analyzing most digital structures only requires a few thousand simulations to model the desired variable as a distribution and extract the mean and variance; for example, leakage is commonly modeled as a log-normal distribution [34]. However, for SRAM cells, the limiting device will exhibit variations in the extreme tail of the distribution at over six standard deviations from the mean, where behavior deviates from an extrapolated value and numerous Monte Carlo simulations would be needed to find a single error. The proposed analysis methodology uses importance sampling to solve this problem by only sampling from the important region of the distribution where failures are more likely, and avoiding unnecessary simulations at points which will not fail.

Importance Sampling

Figure 2.2a shows a graphical representation of the main idea behind importance sampling for a simplified case of two devices, where the two axes represent device threshold voltages. Under the assumption that threshold-voltage deviations due to random dopant fluctuation can be modeled with a Gaussian distribution, Monte Carlo will simulate the two-device circuit with threshold probability density functions (PDF) given by Y_1 and Y_2, where the mean is the device's nominal value, and the σ is given by $\frac{A_{Vt}}{\sqrt{W*L}}$. This enables investigation of a technology before full statistical models are available as long as an approximate $\sigma_{V_{th}}$ is known. In general, this method can be applied to any statistically modeled technology parameter. Ordinary Monte Carlo will only sample failure events with the very small probability shown by region ①. For importance sampling, the mean of Y_1 and Y_2 is changed to create a new PDF, labeled $\hat{Y}_1$ and $\hat{Y}_2$, so Monte Carlo samples failure events with the probability given by region ②. To determine how often these failures would occur without the artificial shift, these samples are unbiased [14].



(a) Importance sampling provides more information about the failure region by shifting the mean of each device's threshold voltage.

(b) Graphical example of the variable-radius algorithm for hypothetical two-device circuit.

Figure 2.2: Two step algorithm to determine p_{bit}.

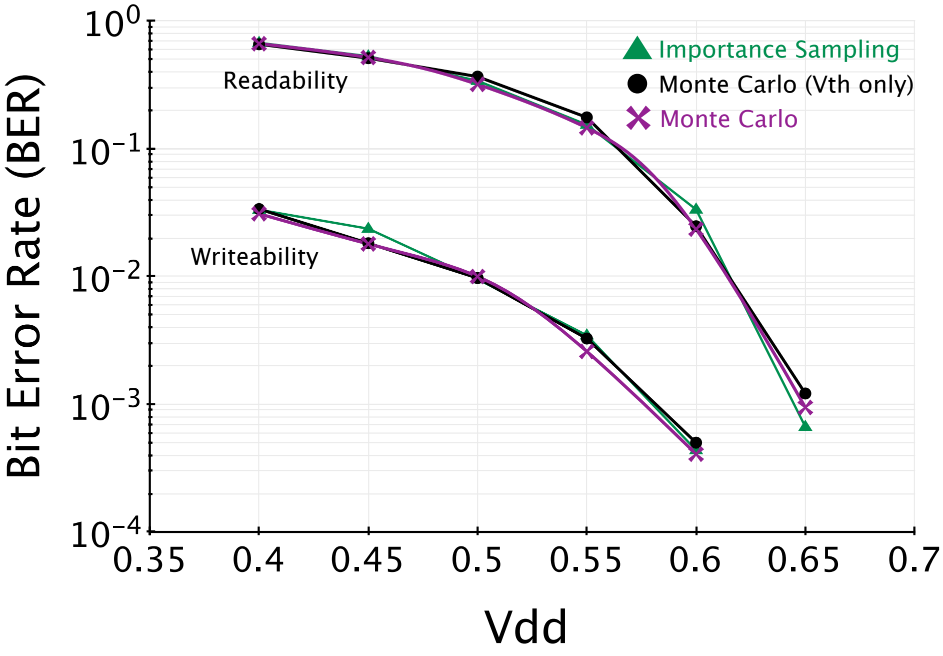


Figure 2.3: Comparison of Monte Carlo and Importance Sampling p_{bit}-estimates for 90% accuracy and confidence.

Variable-radius Most-Probable Failure Point Search

Finding the optimal sampling distribution is complex for a multi-dimensional design space. For quick convergence, the set of mean shifts must be the multi-dimensional most-probable failure point (MPFP), which is the point closest in distance to the origin. This methodology uses a method that performs uniform sampling of a variable radius n-dimensional sphere around points, a similar idea to [15].

Figure 2.2b illustrates a simple case of a two-device circuit. The space being searched is defined as shifts in thresholds for each device (normalized by their standard deviation), and the desired point is the closest point to the origin. Initially, a large search of 5σ in all directions is uniformly sampled from the origin (p_1) and the closest failure point p_2 is found. Larger initial searches are used if no failures are found after the first search. Sampling continues with a decreased search space until no closer points are found. Last, importance sampling is run to determine the final bit error rate.

Most importantly, this algorithm can be readily adapted to different circuits with different numbers of variables. All that is needed to run this algorithm and return a probability of failure is a netlist describing which thresholds can be shifted and failure metrics which return a pass or fail signal for a given threshold shift. This methodology finds the p_{bit} for any dynamic metric in approximately five minutes of simulation time on a modern computer, where most time is spent running small Monte Carlo transient simulations. Designer usability, rather than absolute performance operation, was the goal of this implementation.

The MPFP approach will not only determine failure probability, it will also determine the relative strengths of particular devices that cause failure, and therefore explains why a cell fails. The intuitive explanation of the MPFP is that if Monte Carlo was run for a very long time, most failures will have devices with thresholds shifted by approximately these amounts. Assuming devices models are correct, real silicon cells would also fail for similar device characteristics.

This simulation methodology allows for rapid design space exploration without sacrificing accuracy, and provides intuition about the cause of failures.

Verification

These results closely match Monte Carlo simulation, as shown in Figure 2.3. Note that this IS implementation assumes that the only source of variation is V_{th} variation, yet can be seen to track full MC well. p_{bit} smaller than 10^{-4} cause an excessive MC runtime. Other studies have shown that IS matches MC for longer simulations [14].

2.3 Microarchitecture-level Hard-fault Model

The microarchitecture-level hard-fault model translates probability of bitcell failure to system yield. Microarchitecture-level resiliency techniques tolerate failures by either using ECC to detect and correct errors, programming redundancy to remap failing bits to working bits, or disabling pieces of memory (such as a line in a cache) so that failing bits are never accessed. Schemes are compared by their cache bit error rate.

Generic Framework

The proposed generic framework uses a hierarchy of binomial distributions shown in Figure 2.4 and defined in Equations 2.1-2.8 to translate bitcell failure probability to system failure probability. The distribution of failures in each level of the hierarchy can be represented by a binomial sample of the level below. For example, the number of failing lines in a set is a binomial sample of n total lines in a set with a given probability of line failure. The probability that a given level fails can be determined by evaluating the cumulative density function (CDF) of the level. For example, the probability that a set fails is the CDF evaluated at 0 (assuming no lines can fail in a set). Resiliency techniques simply change the evaluation of each cumulative density function.

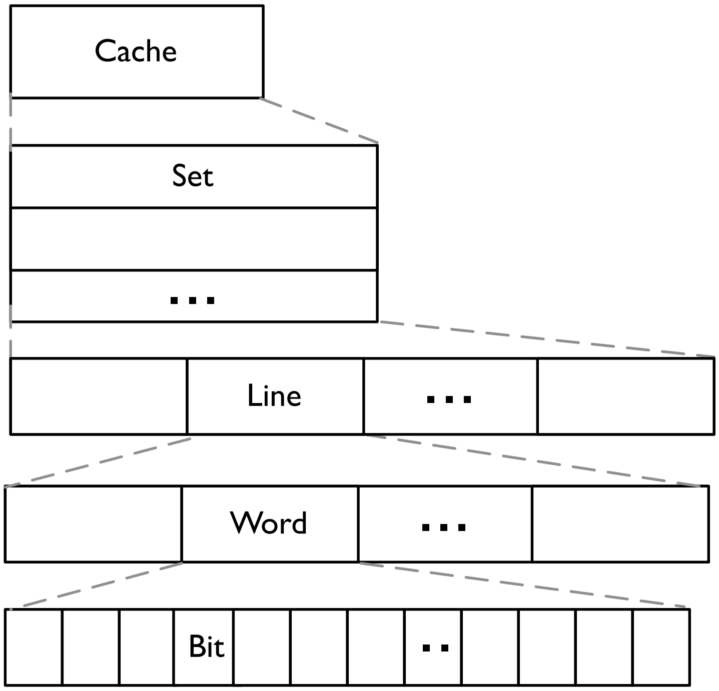


Figure 2.4: Theoretical translation of bitcell failure rate to system failure rate.

$$p_{bit\_fails} = \text{Defined for given } V_{DD}$$

$$X_{word} \sim Binomial(n_{b-w}, p_{bit\_fails}) \tag{2.1}$$

$$p_{word\_fails} = 1 - \mathbb{P}(X_{word} \leq a_{b-w}) \tag{2.2}$$

$$X_{line} \sim Binomial(n_{w-l}, p_{word\_fails}) \tag{2.3}$$

$$p_{line\_fails} = 1 - \mathbb{P}(X_{line} \leq a_{w-l}) \tag{2.4}$$

$$X_{set} \sim Binomial(n_{l-s}, p_{line\_fails}) \tag{2.5}$$

$$p_{set\_fails} = 1 - \mathbb{P}(X_{set} \leq a_{l-s}) \tag{2.6}$$

$$X_{cache} \sim Binomial(n_{s-c}, p_{set\_fails}) \tag{2.7}$$

$$p_{cache\_fails} = 1 - \mathbb{P}(X_{cache} \leq a_{s-c}) \tag{2.8}$$

where a_{b-w} = Allowable number of bit failures in word
a_{w-l} = Allowable number of word failures in line
a_{l-s} = Allowable number of line failures in set
a_{s-c} = Allowable number of set failures in cache
n_{b-w} = Number of bits in word
n_{w-l} = Number of words in line
n_{l-s} = Number of lines in set
n_{s-c} = Number of sets in cache

This analytical approach is superior to a numerical approach because it enables analysis for a large range of p_{bit_fails}. Generating thousands of random fault maps (a Monte Carlo numerical approach) can only translate bitcell failure probability to yield for a very limited range at very low voltage, because tens of thousands of fault maps would need to be generated to make low probability faults appear at high voltage.

While using a binomial distribution to translate failure probabilities is common practice, the new error model introduces a hierarchical combination of explicit binomials in a configuration that enables the evaluation of many different schemes by simply changing a few parameters. At each voltage, the corresponding probability of bitcell failure p_{bit_fails} is inserted, and all of the other probabilities and final yield can be directly evaluated. Most schemes can be evaluated directly by determining the $a_{\_}$ and $n_{\_}$ parameters. Even though the model is generic, it is not an approximation. For more complex schemes, one of the binomial distributions can be replaced by a multinomial distribution for increased flexibility at the cost of decreased intuitiveness.

Example

SECDED is a common resiliency scheme using single-error-correction, double-error-detection ECC on every word to allow correct operation even if one cell in every word is non-functional. The probability that a cache fails is calculated with Equation 2.9 where n is the number of bits in a word, p_{bit} is the probability that a bit fails, and w is the number words in the cache.

$$\begin{aligned} p_{word} &= 1 - [(1-p_{bit})^n + n \cdot p_{bit} \cdot (1-p_{bit})^{n-1}] \\ p_{cache} &= 1 - (1-p_{word})^w \end{aligned} \tag{2.9}$$

A yield specification sets the allowable probability of cache failure, and therefore sets the required bitcell failure probability and required operating voltage to meet this bitcell failure requirement.

This scheme can now be used to quickly reproduce the previous example. For SECDED, the parameters $a_{b-w} = 1, a_{w-l} = 0, a_{l-s} = 0, a_{s-c} = 0$ represent the idea that one bit in every word can fail, no words can fail in a line, no lines can fail in a set, and no sets can fail in a cache. If a line can be disabled, a_{l-s} is changed from 0 to equal the number of lines with two or more errors that can be disabled in a set.

2.4 Circuit-level Soft-fault Model

The circuit-level transient error model translates particle strikes to memory bitcell failures-in-time (FIT). There are two ways for a particle strike to cause a bitcell value flip, as shown in Figure 2.5. A strike to an NMOS drain on the side holding a high value will inject negative current and cause a voltage droop, potentially flipping the cell as shown in Figure 2.6. A strike to a PMOS drain on the side holding a low value will inject positive current and cause a voltage bump, potentially flipping the cell. Note that a strike to an NMOS on the low value side or PMOS high value side will merely reinforce the stored value and will not cause a bit flip. Generally, NMOS strikes are more likely to cause a bit flip, because the combined area of the pass-gate and pull-down devices dwarfs the pull-up size. Using a double exponential equation to model the fast spike and slow fall of the current pulse [35], a binary search determines the critical charge (Q_{crit}) needed to cause a bitcell flip. Figure 2.7a and 2.7b show the critical charge for PMOS and NMOS strikes respectively.

The transient error model intentionally does not account for persistent variation. Random variation in a cell could increase or decrease the susceptibility of a cell to a strike-induced bit flip. However, importance-sampling-based simulations that jointly account for persistent failures and transient failures confirm that ignoring random variation barely affect the results, as shown in Figure 2.8.

In order to translate critical charge to failures-in-time, assumptions need to be made about the frequency and amplitude of particle strikes. The probability of a particle striking a given area per second of at least charge Q_{crit} can be modeled with Equation 2.10 [36].

$$R(q) = F \times A \times K \times exp(-\frac{q}{Q_s}) \tag{2.10}$$

where

- F = neutron flux, evaluated $56.5/m^2$

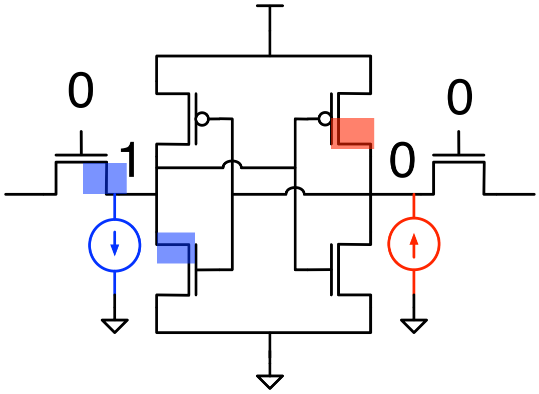


Figure 2.5: Particle strikes can inject current into sensitive NMOS drain nodes (blue) and sensitive PMOS drain nodes (red).

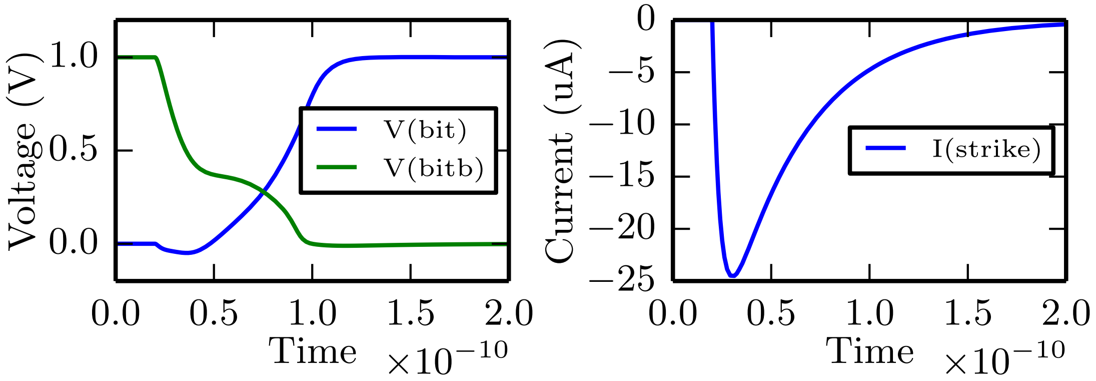
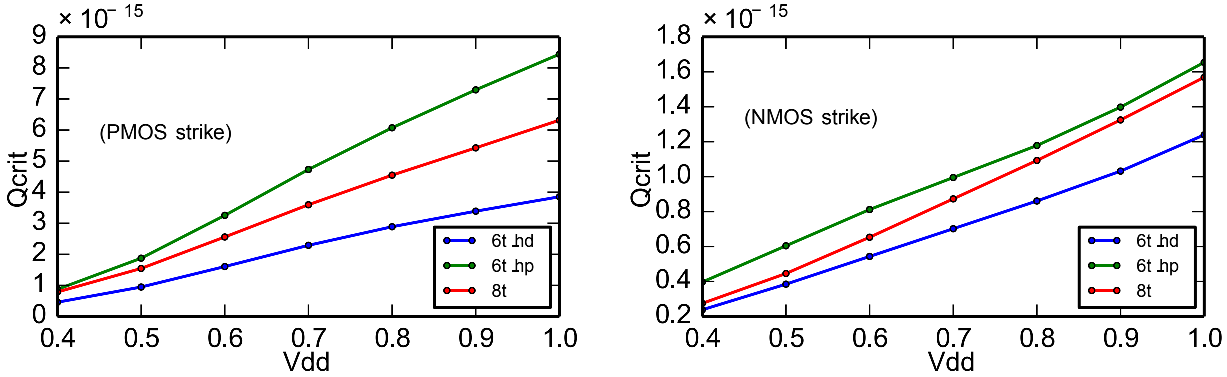


Figure 2.6: Transistor-level simulation of particle strike for the high storage node (NMOS drain strike) sinking current.



(a) PMOS drain strike.

(b) NMOS drain strike.

Figure 2.7: Q_{crit} versus supply voltage for different bitcells.

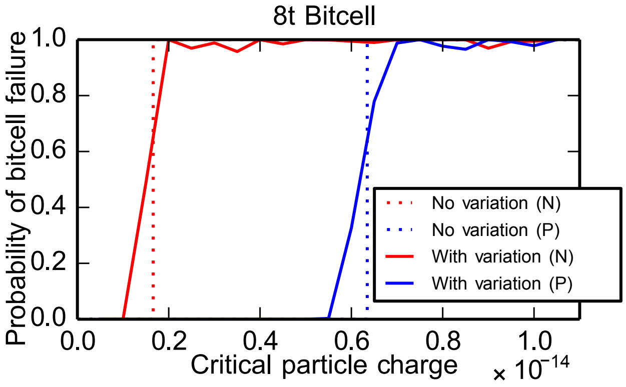


Figure 2.8: Transistor-level simulation of particle strike Q_{crit} with and without accounting for random variation.

- A = sensitive diffusion area of a given node for a given node value
- K = technology-independent fitting parameter (see [36]), assumed to be 2.2×10^{-5}
- Q_s = technology-dependent charge collection parameter, different for NMOS and PMOS and estimated from [36], assumed to be 17fF for NMOS and 6.5fC for PMOS

To calculate FIT, R is multiplied by Q_{crit} for the bitcell and operating condition, then multiplied by 3.6×10^{12} to translate to failures per 10^9 hours (definition of FIT). FIT for particle strikes to the high and low sides are calculated separately, as they have different Q_{crit}, sensitive area, and charge-collection efficiency. The total FIT per cell is the sum of both methods of failure. Note that the stored value doesn't affect FIT for 6T bitcells because they are symmetric. Circuit-level transient-error resiliency techniques, such as increasing bitcell size, can be evaluated using this model. Technology changes can dramatically affect soft error rate—for example, transitioning from BULK to FDSOI dramatically reduces the sensitive area and decreases the error rate by 110× based on experimental data [18].

2.5 Microarchitecture-level Soft-fault Model

The microarchitecture-level transient-error model translates bitcell FIT to system FIT. Given a bitcell FIT, a simple estimate of system FIT can be made by summing every bitcell's individual FIT. A required system FIT can be determined by translating FIT to mean-time-to-failure. A typical FIT target of 4000 corresponds to mean-time-to-failure (MTTF) of 30 years [37]. However, there are three effects in memory that complicate the calculation of system FIT:

1. Persistent and intermittent errors: Intermittent errors that are not detected by BIST, or persistent errors that are only repaired by ECC, can use up all ECC correction capacity for some words and leave these particular words more vulnerable to soft errors.

2. Multi-bit upsets: As process geometries decrease, the proportion of soft errors that affect more than one bit in a word increases [38]. Multiple bit upsets in a single word can exceed error-correction capabilities; however, interleaved bitcells ensure that physically adjacent bitcells map to different logical words and reduce multi-bit upsets.

3. Error accumulation: During a given period T, it is possible for a word to have multiple transient errors occur between corrections. If transient errors are rare events, error accumulation is a negligible effect. This accumulation of errors can be prevented by "scrubbing", where the system reads and rewrites every word periodically to fix soft errors before a second error occurs on the same word, reducing the accumulation period T from system lifetime to the scrubbing interval. For SRAM caches, the period T that elements remain in the cache can be very small.

The proposed model accounts for all three of these effects to accurately translate bitcell FIT to system FIT, assumes that each strike can upset k bits, and the possibility of upsetting k bits is estimated from multi-bit upset distances reported in [38]. The number of cells in a row is the product of the interleaving factor and number of bits in a word, and it is assumed that multi-bit strikes happen to adjacent cells along a row only, as shown in Figure 2.9. While multi-bit strikes can affect multiple rows, the analysis is simplified by assuming that strikes only affect one row because strikes in different rows can never affect the same word.

Published data about multi-bit errors that differentiate between errors along a row and along a column is scarce. The distinction is critical, because multi-bit errors along a column will affect different words and therefore be correctable by single-bit ECC, while multi-bit errors along a row would require double or higher ECC if the interleaving distance isn't large enough. Equation 2.13 uses a geometric distribution model to match published data from the

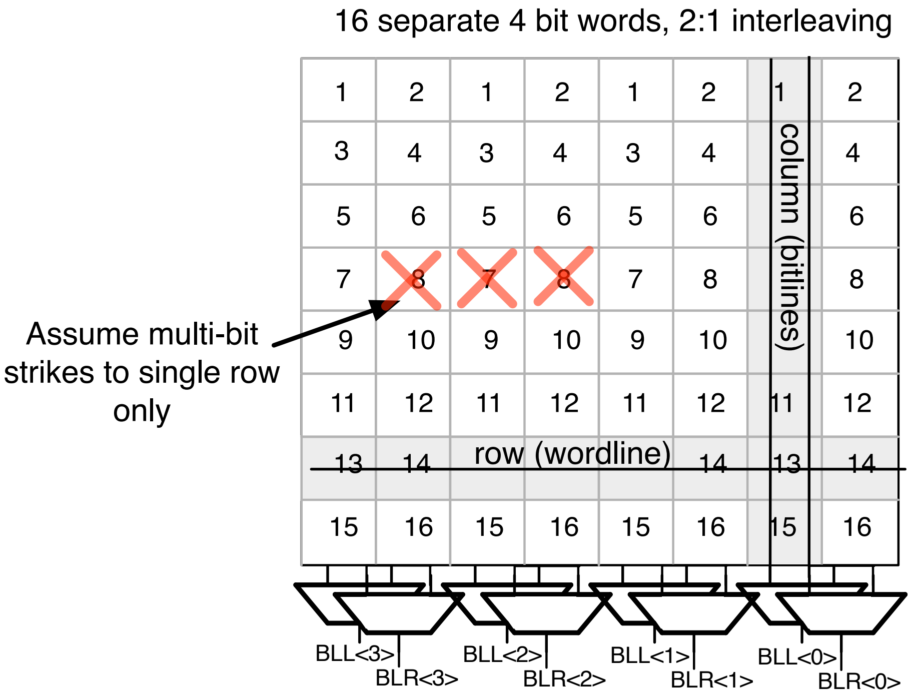


Figure 2.9: Example physical organization of SRAM arrays for n_{bw}=4 and interleaving of 2.

most convincing report on multi-bit strikes [38]. Figure 2.10 plots the expected proportion of strike sizes using this model.

$$N \sim Geom(r_{mbu}) \tag{2.11}$$
$$r_{mbu} = 0.3 + 3.2 \times 10^5 \cdot w_{cell} \tag{2.12}$$
$$\mathbb{P}(N = n) = (1 - r_{mbu})^{(k-1)} r_{mbu} \tag{2.13}$$

Arrivals of strikes are a Poisson process with a rate set by the bitcell FIT rate calculated by the circuit-level error model (E_{bit}) and the number of bitcells in a single row defined by the interleaving factor (I) times number of bits in word (n_{bw}). The size of the strike N is geometric, so the random variable M that represents the number of upsets in a row during period T is a Geometric Poisson distribution with PDF given by Equation 2.16.

$$\lambda = E_{bit} * n_{bw} * I \tag{2.14}$$
$$\mathbb{P}(M = 0) = e^{-\lambda} \tag{2.15}$$
$$\mathbb{P}(M = m) = \sum_{k=1}^{m} e^{-\lambda} \frac{\lambda^k}{k!} (1 - r_{mbu})^{m-k} r_{mbu}^k \binom{m-1}{k-1} \tag{2.16}$$

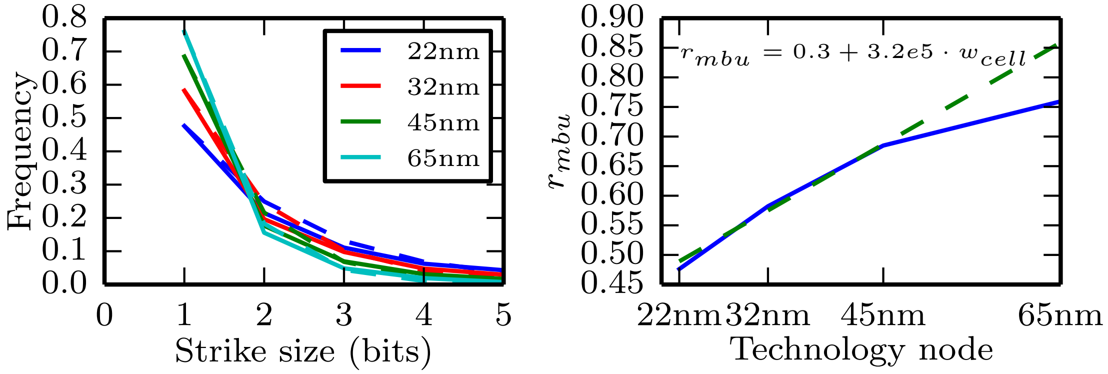


Figure 2.10: Distribution of multi-bit upsets from a particle strike.

This distribution includes accumulation of errors. For example, the probability that there are n=2 errors is given by both the probability that there is a 2-bit strike (k=1 term) and two separate 1-bit strikes in the same period (k=2 term). However, accumulation is very rare unless the failure rate is extremely high, so N_{bit} can optionally be simplified to just equal the probability of a single strike of different sizes in Equation 2.17.

$$\mathbb{P}(M = m) = e^{-\lambda}\lambda(1 - r_{mbu})^{m-1} r_{mbu} \tag{2.17}$$

Table 2.1 shows the number of upsets over a 1-year period for Equation 2.16 and 2.17 for the base design with SECDED, as well as design points with more interleaving and much higher failure rates. This table shows that probability of bit upsets larger than the interleaving factor are much more likely than accumulating multiple strikes to the same word, even when errors are allowed to accumulate for an entire year. Accumulation only becomes an issue when the FIT rate is 10,000 times higher than current rates and multi-bit upsets are ignored. Additionally, typical on-chip caches will accumulate errors for less than 1 second, and certainly not 1 year, so accumulation will be ignored in this study.

A larger cell, or more interleaving, decreases the probability that a strike upsets multiple bits in a single word. Larger cells are easily modeled because Equation 2.13 is a function of bitcell size. To model interleaving, the number of upsets in a row from Equation 2.17 is transformed into to the number of upsets in a word, which will be represented by the random variable Y. Due to interleaving, a multi-bit strike will cause multiple upsets of different sizes in different words, as shown in Figure 2.11, and this pattern can be generalized to different interleaving distances by Equation 2.19.

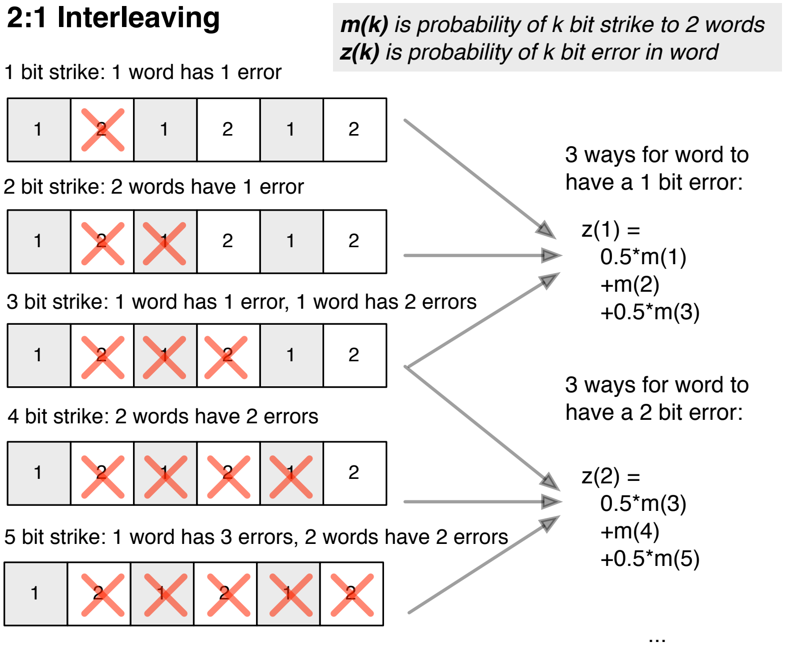

2:1 Interleaving
m(k) is probability of k bit strike to 2 words
z(k) is probability of k bit error in word
1 bit strike: 1 word has 1 error
1 2 1 2 1 2
2 bit strike: 2 words have 1 error
1 2 1 2 1 2
3 bit strike: 1 word has 1 error, 1 word has 2 errors
1 2 1 2 1 2
4 bit strike: 2 words have 2 errors
1 2 1 2 1 2
5 bit strike: 1 word has 3 errors, 2 words have 2 errors
1 2 1 2 1 2
3 ways for word to have a 1 bit error:
z(1) = 0.5*m(1) +m(2) +0.5*m(3)
3 ways for word to have a 2 bit error:
z(2) = 0.5*m(3) +m(4) +0.5*m(5)
...

Figure 2.11: Visual representation of Equation 2.19 for an interleaving factor of 2.

$$\mathbb{P}(Y=0)=\mathbb{P}(M=0) \tag{2.18}$$

$$\mathbb{P}(Y=y)=\sum_{k=(y-1)*I+1}^{y*I-1}\mathbb{P}(M=k)\cdot\frac{k-(n-1)\cdot I}{I}+\sum_{k=y*I}^{(y+1)*I-1}\mathbb{P}(M=k)\cdot\frac{I-(k-n\cdot I)}{I} \tag{2.19}$$

Table 2.1: Comparison of including accumulation effects for a 32KB L1 cache with SECDED for a typical FIT rate (1×10^{-4}) and extreme FIT rates during a 1 year-accumulation period.

FIT of bitcell	Multi-bit upsets	Cache FIT (Eq. 2.17)	Cache FIT (Eq. 2.16)
1e-04	Yes	3.2×10^{0}	3.2×10^{0}
0.1	Yes	3.2×10^{3}	3.2×10^{3}
1	Yes	3.2×10^{4}	3.2×10^{4}
10	Yes	3.1×10^{5}	3.3×10^{5}
1.0e-04	No	0	5.4×10^{-14}
0.1	No	0	5.4×10^{-5}
1	No	0	5.4×10^{-2}
10	No	0	5.4×10^{1}

To account for persistent errors, the transient error model extends the existing persistent error model. For a given operating condition, the probability that a bitcell fails due to a persistent error is known from Equation 2.3, so the distribution of the number of persistent errors per word is known. Error-correction codes set the allowable number of faults per word due to both persistent and transient sources of error, so the probability of a fault will be the probability that the sum of persistent and transient errors within a word surpasses the allowable number of faults.

$$\mathbb{P}(Z = z) = \sum_{k=0}^{z} \mathbb{P}(Y = k) \cdot \mathbb{P}(X_{bit} = z - k) \tag{2.20}$$

If persistent errors are left uncorrected, then the FIT calculation will not be correct, so the range of X is limited to $(0,a_{bw})$. Z in Equation 2.20 gives the number of upsets during period T per word from both persistent and transient errors by adding M (probability of m transient errors) and X (probability of x persistent errors), which corresponds to a convolution of the probability-density functions. To translate the resulting probability of cache failure due to all error sources to cache FIT, we need sum the probability of failures beyond the correctable number of bits (a_{bw}), multiply Z by the number of words in the cache, and scale the rate to be in upsets per 10^9 hours $(3.6 \times 10^{12}$ seconds).

$$FIT = (1 - \sum_{k=0}^{a_{bw}} \mathbb{P}(Z = k)) \cdot \frac{3.6 \times 10^{12}}{T} \cdot n_{wl} \cdot n_{ls} \cdot n_{sc} \tag{2.21}$$

This model has been validated with a brute-force Monte Carlo simulation that uses a Binomial distribution to inject persistent errors for a given supply voltage, and a Poisson distribution to inject transient errors for a given bitcell FIT and period T, with size of strike given by M.

This model can be used to evaluate resiliency schemes. For example, Figure 2.12 shows system FIT for a cache with no ECC, SECDED ECC, and DEC-TED ECC for a 32KB cache of different sized bitcells, and an interleaving of 2.

2.6 Modeling Energy, Area, Delay, and CPI

Published measurement data showing the effect of voltage scaling on energy, delay, and bitcell failure probability is scarce for modern processes. Dynamic energy and leakage power is usually approximated as scaling with V^2 for dynamic and V^3 for leakage. Total energy at a given voltage is strongly dependent on delay scaling, because dramatic increases in delay at low voltages cause leakage to be integrated over longer cycles and prevent further voltage scaling from actually saving energy. Additionally, the relationship between bitcell failure probability and voltage is critical, because the bitcell failure rate at the voltage that

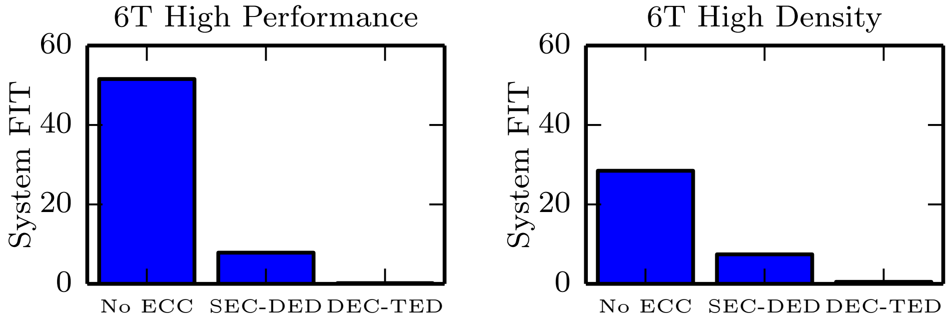
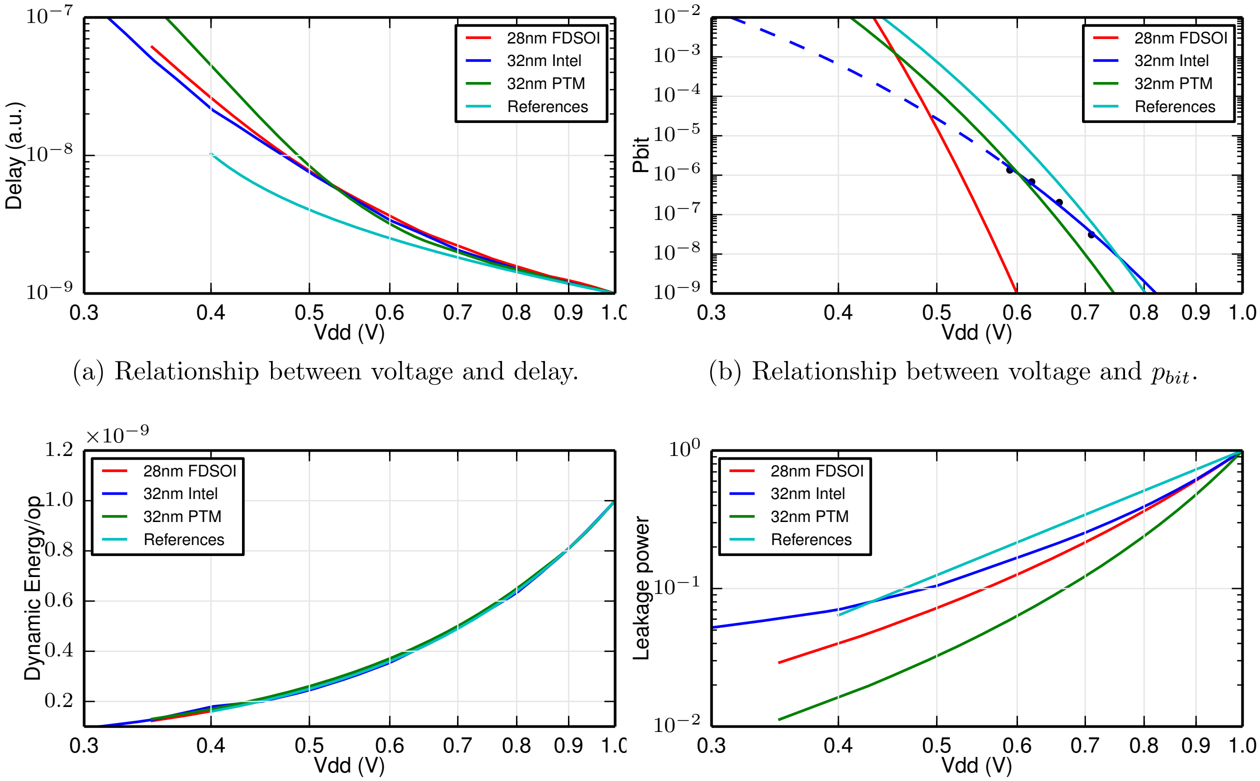


Figure 2.12: System FIT for a smaller, high-density (HD) bitcell and larger, high-performance (HP) bitcell, and different ECC schemes.

corresponds to minimum energy will set the amount of resiliency needed. Most earlier work assumes that further resiliency and voltage scaling will always result in an energy decrease, but this assumption is not always true.

Four datasets are used to explore the interaction of assumptions about energy, delay, and bitcell failure probability with voltage scaling. The first dataset, "28nm FDSOI", determines power and delay from transistor-level simulation of a commercial 28 nm FDSOI process and calculates bitcell error probabilities using the hard-fault modeling scheme. The second dataset, "References", contains a standard set of assumptions used by many different architecture resiliency papers [29], [28], [39], [25], and includes frequency, power, EPI, and bitcell failure versus supply voltage assumptions taken from measurements and simulations in a 65 nm process [40]. The third dataset, "Intel 32nm", comes from reported measurement results of a commercial 32 nm process with a 0.199 μm^2 cell [41] [42]. The last dataset, "PTM", comes from simulations of the predictive technology model [43]. For comparison purposes, the delay is normalized to be 1 ns at 1 V, the dynamic energy is normalized to be 1 nJ at 1 V, and the leakage power is normalized to be 1 W at 1 V, which results in equal dynamic and leakage contributions at 1 V. Figure 2.13a shows how delay scales with voltage. The "References" dataset uses a linear fit to model frequency (reported values down to 0.5 V, extrapolated further for comparison purposes), which predicts a much faster operating frequency at low voltage than any other dataset and neglects leakage increases at low voltages.

Figure 2.13b shows how bitcell failure probability (p_{bit}) scales with voltage. Both the slope of the failure probability, and the absolute V_{min} vary dramatically between the datasets. The steeper the slope, the less resiliency techniques can help voltage scaling, because the same increase in allowable p_{bit} corresponds to a smaller voltage reduction. The "Intel 32nm" dataset will predict that small increases in resiliency produces much larger V_{min} reduction than other datasets.



(a) Relationship between voltage and delay. (b) Relationship between voltage and p_{bit}.

(c) Relationship between voltage and dynamic energy. (d) Relationship between voltage and leakage power.

Figure 2.13: Survey of assumptions used for architecture-level resiliency analysis.

Figure 2.13c show how dynamic energy scales with voltage. As expected, all datasets closely match theory because capacitance is constant at different voltages. This figure plots dynamic energy, not power, to avoid a dependence on operating frequency.

Figure 2.13d shows how leakage power scales with voltage. In this case, PTM assumes that leakage decreases much more than the other data sets, which will optimistically assume that aggressive voltage scaling is worthwhile. This figure plots leakage power, not energy, to prevent a dependence on operating frequency with this metric.

These four metrics—delay, p_{bit}, dynamic energy, and leakage power—as a function of V_{DD} can dramatically affect the results of architecture studies, but are well-defined metrics if the process has been characterized. However, modeling the translation of dynamic energy and leakage power into total energy is subjective, and requires knowing both the architecture and activity of the system. For compatibility with existing analysis, this model assumes the core, L1, and L2 are all on the same voltage rail. To calculate total energy, this model assumes the system is optimized such that at max performance, 2/3 of energy is active and 1/3 is leakage, which matches analysis of various systems by McPAT [44]. This is an upper bound on the

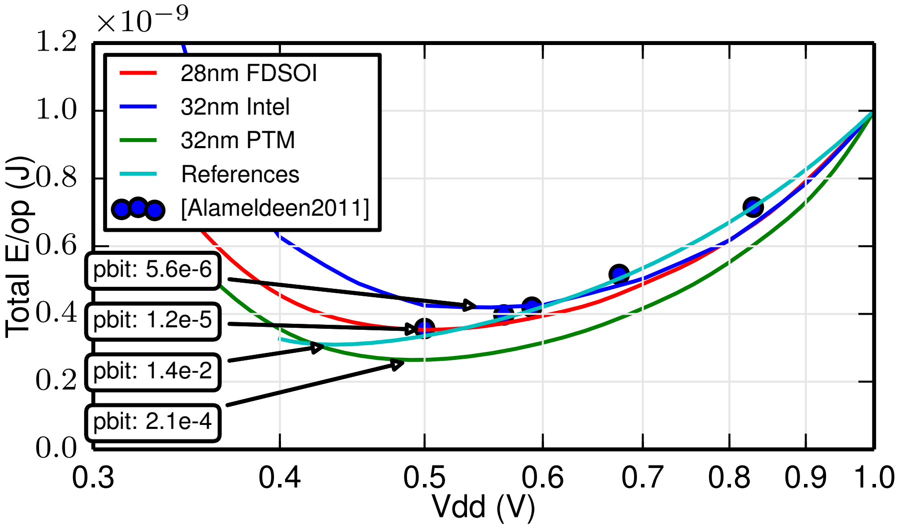


Figure 2.14: Energy per operation for each dataset assuming 2/3 dynamic energy and 1/3 leakage energy at 1V.

energy reduction achievable with resilient design, because it does not account for resiliency technique overhead due to extra circuitry or reduced cache capacity. Figure 2.14 shows the energy scaling potential of all four datasets using the theoretical method of scaling energy. The bitcell failure probability at the minimum energy (E_{min}) is annotated. Every dataset except the “References” data set shows diminishing returns for voltage scaling, because the large increase in delay near the threshold voltage integrates more leakage per cycle.

Most prior cache resiliency technique evaluations [29], [28], [39], [25] use circuit-level assumptions from an experimental bitcell design in 65 nm (“References” dataset [40]), which has very different characteristics than datasets from a variety of modern 28 nm processes with industry-standard 6T bitcells. Figure 2.14 shows that when targeting optimal energy efficiency, the “References” dataset predicts that resiliency schemes need to tolerate a 100× to 10,000× higher bitcell failure probability than predicted by the other three datasets, leading to unnecessarily complex resiliency techniques. Assumptions about energy, area, and delay scaling versus voltage have major implications for resilient design, because these assumptions affect the trade-off between the benefits of voltage scaling and the costs of improving resiliency.

Chapter 3

SRAM Failure Mechanisms

To enable the highest energy efficiency, a circuit should be operated at the lowest voltage that satisfies a given performance requirement, yet increased variability in deeply-scaled technologies prevents SRAM from achieving as low a V_{min} as the surrounding logic.

Developing circuits that improve SRAM operation at low voltages requires understanding the various failure mechanisms of SRAM cells. Transistor characteristics change for every process node, so techniques effective in one node might be useless in another node. Therefore a methodology is required to evaluate different techniques for every new process node.

This chapter is split into two sections: simulation of SRAM failure mechanisms, and measurement of SRAM failure mechanisms. Simulation of SRAM failure enables experimentation and evaluation at design-time, but lacks accuracy. Even after developing techniques to speed up rare event simulations, analysis results rely on simulation models of transistors being accurate out to six sigma from the average. Silicon measurement of SRAM failures requires a long feedback cycle: many months are required to design the circuits, fabricate, and test. In addition, insight into the design is difficult because observability is limited in real hardware. However, simulation results that have been calibrated and validated against real silicon can be trusted for products. Both methods are necessary and interdependent: simulations need measurements for improved calibration, and measurements need simulations to decrease the possible design space size.

3.1 Simulating Failure Mechanisms [1]

This section describes the methods used in this thesis to evaluate circuit-level techniques at design-time by using the simulation model described in Section 2.2 to analyze the effect of various design decisions on predicted failure rates. Establishing meaningful failure metrics is critical to ensure that predictions eventually match measured behavior. The most-probable-failure-point is analyzed to provide insight into the effectiveness of various resiliency tech-

[1] The content in this section was derived from a publication in TCAS-II 2012 [33].

niques. Last, the effectiveness of various techniques are compared for a 28nm high-density bitcell.

3.1.1 Failure Metrics

Several metrics based on transient simulations can be used to take dynamic effects into account by essentially imitating typical SRAM read and write operations. The methodology proposed in this thesis defines three modes of failure: readability, writeability, and read stability. Readability failures occur when the read bitline discharge in a specified time is less than the offset of the sense amplifier. This study assumes a differential read using a sense amplifier for 6-transistor (6T) cells, and a domino-style read with a skewed inverter for 8-transistor (8T) cells. Unlike I_{read}, this readability metric takes into account the reduced V_{DS} on the pass-gate as the bitline discharges. Writeability failures occur when the internal node voltage does not reach the desired write value. There are two forms of stability failures. Read stability failures occur when bitcell contents flip accidentally during a read condition. Half-select stability failures (read stability failures on non-accessed columns in interleaved arrays) occur when bitcells on non-accessed columns of an interleaved array flip accidentally during a write operation. For readability and writeability, the result is checked for any errors at the end of the clock period to emulate typical SRAM operation where back-to-back access is supported. This pessimistic measure, which accounts for successive reads, results in an average V_{min} increase of 30mV. Each mode of failure is measured by transient simulations of netlists that accurately model read or write signal timing and capacitances.

For 90% yield on a 1MB array, the probability of a single bit failure, or the probability of bitcell failure, p_{bit}, must be less than 10^{-9}. V_{min} is defined as the minimum operating voltage for which the p_{bit} of all three types of failures is $< 10^{-9}$. This analysis focuses on the 6T cell, then extends the results to the 8T cell.

3.1.2 Failure Analysis

Using the methodology described, it is possible to analyze how different design decisions affect p_{bit} and V_{min}, and use the MPFP to provide intuitive explanations for these quantitative findings.

For example, consider the MPFP for writeability failure determination at 0.8V in Table 3.1, with a predicted p_{bit} of $2 \cdot 10^{-11}$. The standard deviation of the threshold voltage (σ) is around 25mV, so the MPFP has a right PG with a threshold that is about 140mV larger than normal. In this example, the goal is to write a 0 to the right side (BLRI); the weak pass gate and strong pull up prevent the high node on the right side from being pulled low—suggesting that assist techniques that either weaken the pull up or strengthen the pass gate could reduce p_{bit}. These failure mechanisms change with cell size, process corners, and voltages, making the intuition given by this approach invaluable to designing effective assists.

Table 3.1: Most probable failure point for writeability at 0.8V.

PUL ΔV_{th}	PDL ΔV_{th}	PGL ΔV_{th}	PUR ΔV_{th}	PDR ΔV_{th}	PGR ΔV_{th}
-.013σ	-1.39σ	1.62σ	-2.97σ	0.0292σ	5.64σ

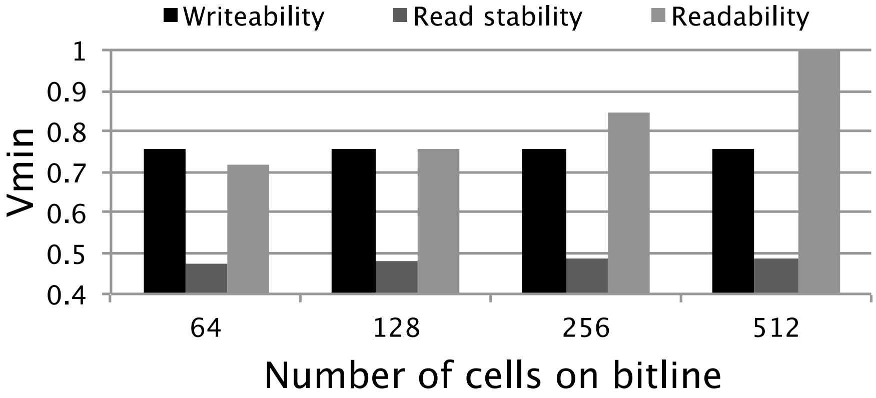


Figure 3.1: Effect of bitline capacitance on p_{bit}.

Effect of array organization and timing

Bitline capacitance, determined by the number of cells on a bitline, has a large effect on readability only, shown in Figure 3.1. Writeability is unaffected as the bitlines are assumed to be held by a device, and stability shows negligible improvement for shorter bitlines because the low-node bitline can easily decrease voltage to match the internal value. Also, changing the frequency of SRAM operation will change the error rate for all three modes of failure, as shown in Figure 3.2. Most importantly, this figure shows the large V_{min} discrepancy between conventional static metrics and this methodology's read stability. At 2GHz, V_{min} for read stability is about 450mV, while a static test would predict a V_{min} of 650mV. Furthermore, fitting a Gaussian distribution to the static noise margin (SNM) instead of using importance sampling would predict a V_{min} of 780mV. Designing using pessimistic metrics such as SNM and distribution approximations results in a drastic overdesign of the bitcell for stability.

Effect of corners

A summary of the effects of corners is shown in Figure 3.3. Readability and writeability are anti-correlated with stability, and process corners have the largest effect on stability. This behavior suggests that design-time optimization of the stability versus writeability trade-off would be dangerous.

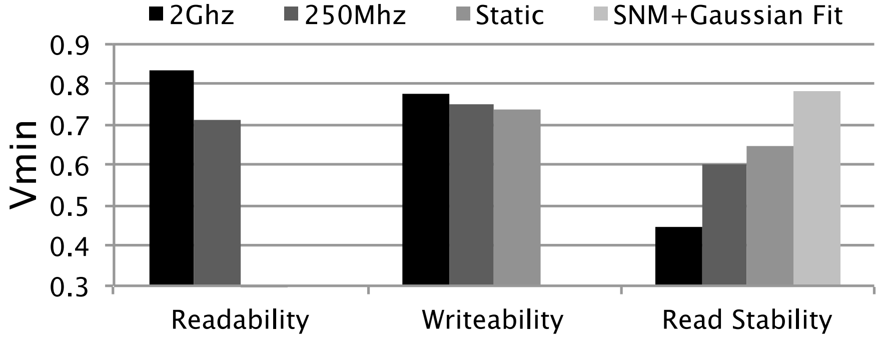


Figure 3.2: Effect of clock period on V_{min}.

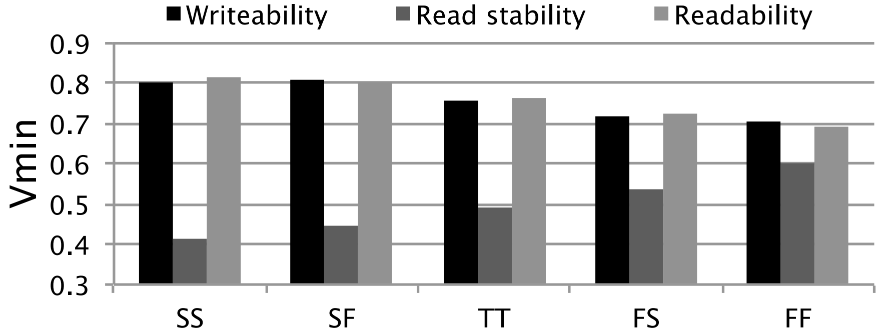


Figure 3.3: Effect of process corners on V_{min}.

Effect of assist techniques

Assist techniques that dynamically change the operating characteristics of bitcells, such as boosting the wordline voltage above the cell voltage, can lower V_{min} for SRAM. A comprehensive analysis of the effectiveness of assist techniques was performed in [13] using static metrics, which have been shown to be a poor match to silicon failures [4]. Importance sampling has been used to analyze how cell sizing, doping, and assists can minimize SRAM energy, but limited focus was given to assist techniques [16]. An alternative to importance sampling based on statistical classifiers was used to analyze V_{min}, but focused on dynamic writeability only and provided little investigation of assists [45]. This section provides an in-depth analysis of assist techniques' potential to reduce V_{min} using importance sampling for dynamic metrics.

For the investigation of each assist technique, the following assumptions were made: corner: TT, design: HD 28nm 6T $0.120\mu m^2$ cell [46], nominal voltage: 1V, period: 50 FO4 (≈1ns at 1V), wordline pulse width: 25 FO4 (1/2 of period), sense-amplifier offset: 0.1V, bitline capacitance: 15fF (128 cells). The effect of changing all of the above assumptions have been investigated in earlier sections. Note that the period will track the process FO4 as the supply is reduced to match the assumption that the SRAM operating frequency will

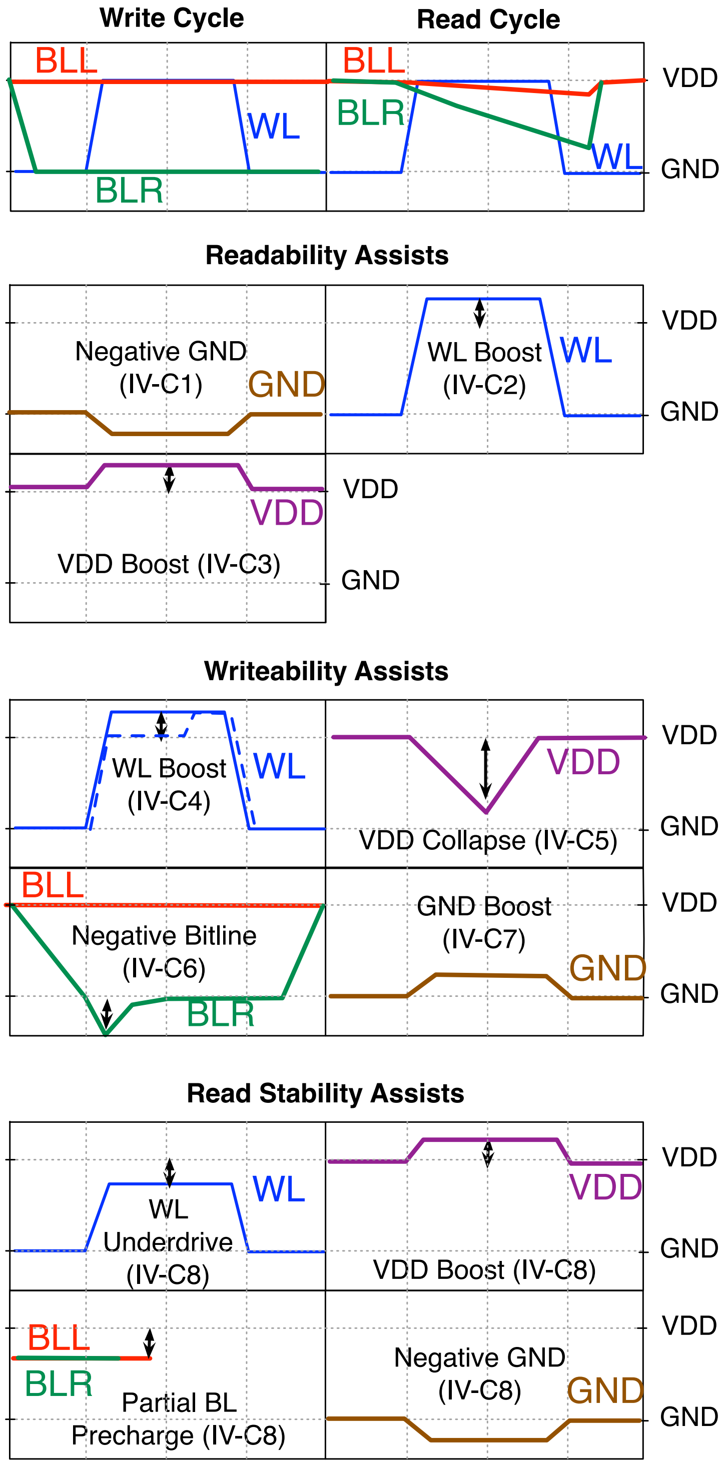


Figure 3.4: Summary of assist techniques: negative GND, WL boost, V_{DD} boost, V_{DD} collapse, negative BL, GND boost, WL underdrive, partial BL precharge

be set by the critical path of a processor.

Different degrees of assist are defined as a percentage of the supply voltage (as opposed to the absolute voltage) because assist voltages are generally set by voltage dividers or charge redistribution and therefore are proportional to V_{DD}. Discussion of side effects assumes that V_{DD} runs vertically (parallel with bitlines), and GND runs horizontally (parallel with wordlines). Energy and area overhead are implementation dependent so are not quantified.

The voltage waveforms for a variety of assist techniques that target each mode of failure—readability, writeability, and read stability—are summarized in Figure 3.4. The results of a thorough design-space exploration of effective assist techniques results are summarized in Figure 3.5 for readability and writeability assists. Each V_{min} measure also includes any stability consequences caused by the assist. Because all assists can be applied on a per-operation basis, the results for write V_{min} and read V_{min} are independent.

Effect of negative cell GND as a readability assist

Reducing the voltage of GND has been shown to improve readability [23]. Negative GND is the most effective of all readability assist techniques as it increases the gate over-drive on both the PD and PG by pulling the internal node holding 0 below ground. Unfortunately, this technique has a very high energy cost for 6T arrays, because each cell has two GND lines running horizontally, which have a large capacitance.

Effect of wordline boost as a readability assist

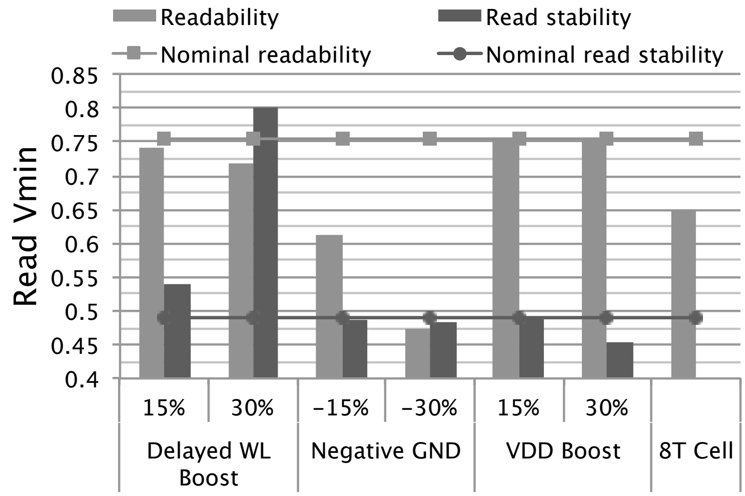
Using a wordline boost can improve both the readability and writeability of cells, but will drastically diminish the read stability of all half-selected cells in interleaved arrays, as shown in Figure 3.6. To avoid the half-select issue, the boost can be delayed until part-way through the wordline pulse, so that half-selected cells have already started reading and the bitline voltage matches the internal voltage more closely [8]. Figure 3.6 shows that while delayed boost helps, the trade-off between writeability and read stability remains very sensitive for wordline boosting. Adaptive circuitry must be used in order to tune this assist [7].

Effect of cell V_{DD} boost as a readability assist

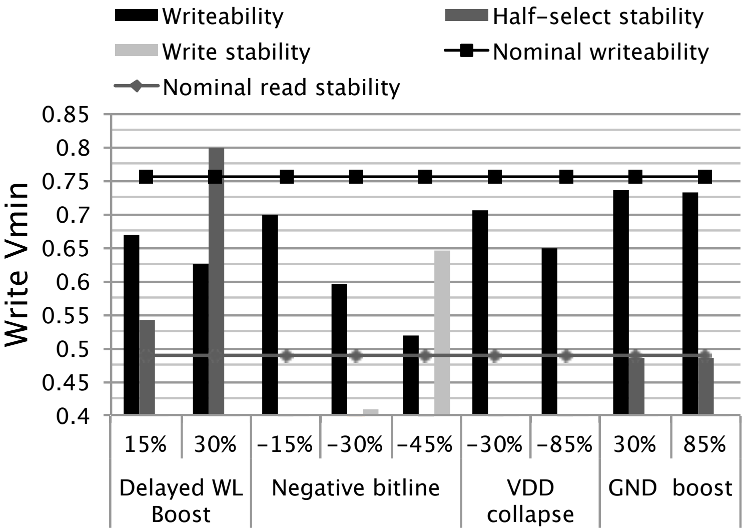
For readability, both the PG and PD devices need to be strong. A V_{DD} boost improves the strength of the PD, but not the PG. Because the PD is sized larger than the PG for stability reasons, the probability that the PD limits read current more than the PG is much lower, suggesting that using wordline boost to improve the PG strength would be more effective.

Effect of wordline boost as a writeability assist

Wordline boost improves writeability by strengthening the PG, but hurts stability as discussed above.



(a) Readability



(b) Writeability

Figure 3.5: Impact of assist techniques on V_{min}.

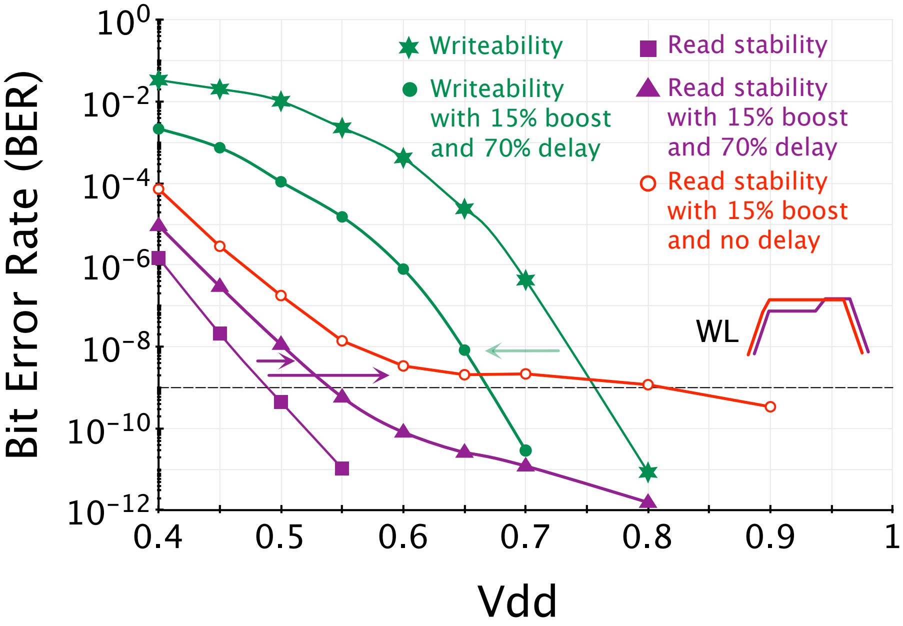


Figure 3.6: Wordline boost improves writeability while reducing read stability.

Effect of cell V_{DD} collapse as a writeability assist

V_{DD} collapse decreases write V_{min} by decreasing the strength of the cross-coupled inverters [7]. However, for the analyzed cell, this assist is much less effective than negative bitline, because while V_{DD} collapse helps the PG pull the high-node low by weakening the high-node PU, a writeability failure can still occur if the PU on the low-node side is weak.

The dangerous consequence of V_{DD} collapse is potential violation of the data retention voltage of non-accessed SRAM in the same column. However, IS tests of this condition show a $p_{bit} < 10^{-9}$ for all cases of interest.

Effect of negative bitline as a writeability assist

Negative bitline improves writeability by increasing the V_{GS} on the PG [23]. Simulation results show this as the most effective write assist, because it both strengthens the PG and helps pull the high-node low while also strengthening the low-node PU to complete the write operation. The IS simulation testbench uses a flying capacitor as opposed to a negative regulated voltage to accurately match typical implementations. However, by decreasing one of the bitlines below GND, a non-zero gate over-drive will appear across the PG of non-accessed rows. If the internal node of an unaccessed bitcell on this side is high, then the value of the cell could flip, causing a write stability error. An IS test found that the p_{bit} for this case was $< 10^{-9}$ for boost amounts $\leq 30\%$.

Effect of cell GND boost as a writeability assist

GND boost weakens the cross-coupled inverters, improving writeability [47]. However, the effect of a large GND boost saturates after 30%, because the NMOS PG must pull the low internal node high for this assist to work, but can only pull up to around V_{DD}-V_{th}. This limitation does not exist for V_{DD} collapse, as NMOS can pass low voltages without limitation.

Effect of stability assists

Stability assist are not useful for the cell under investigation as readability and writeability V_{min} dominate. However, for a different cell and process, stability could be a major concern.

Investigations at very low voltage show WL underdrive to be the most effective stability assist. Both V_{DD} boost and negative GND attempt to limit the voltage bump on the low side, but this effect is countered by shifting the switching threshold of the high-side inverter, canceling most gains and making these techniques ineffective.

A regulator can be used to precharge bitlines to around 70% of V_{DD} to improve yield from 5 to 5.7 sigma, or equivalently, from a p_{bit} of 2.9×10^{-7} to 6×10^{-9} [24]. Importance sampling analysis of read stability confirmed these results, with a p_{bit} improvement of around 1.5 orders of magnitude at 0.7V. But the assist becomes less helpful at low supplies and only achieves a V_{min} reduction of 25mV. Note that readability is slightly diminished due to the decreased V_{DS} on the PG.

Other techniques to improve readability

Readability has conventionally not been an SRAM design metric, as it depends on peripheral circuitry, while read stability and writeability do not. Variations cause a wide variability in read current. Shorter bitlines, as shown in Figure 3.1, allow smaller read currents to provide the same required voltage difference but increases area overhead as sensing circuitry is amortized over fewer cells.

Leakage

The technology under investigation is a low-power process, so leakage was negligible even for a worst-case column of 512 cells. However, leakage can easily be taken into account with this methodology by using Monte Carlo to characterize the leakage current of N worst-case cells as a log-normal and including it into IS as an additional variable described by the fit distribution.

Assist methods for 8T cells

8T bitcells have the same writeability assists as 6T bitcells. If 8T cells are interleaved, write operations will cause read stress on half-selected cells, producing the same read stability trade-offs as the 6T cell. Readability assists are generally no longer needed, due to the much improved read path. Figure 3.5a compares readability of the reference 6T array and bitcell to a domino-style read for an 8T cell with the same number of bitcells on a column.

3.1.3 Summary

The bitcell analyzed here is limited by both readability and writeability, but not stability. Using a combination of negative GND line to improve readability and a negative BL to improve writeability would lower V_{min} by 175mV. However, implementing negative GND involves the very difficult task of regulating a negative voltage that needs to sink every column's read current. So to improve readability, modifications of the read path such as using shorter bitlines, implementing hierarchical bitlines to minimize local bitline capacitance, using lower threshold devices, or lowering sense amplifier offset, are needed to lower V_{min}. Assists that improve stability have a very detrimental effect on readability and writeability, so cell sizing (sizing the PD stronger than the PG) remains the best option for maintaining stability.

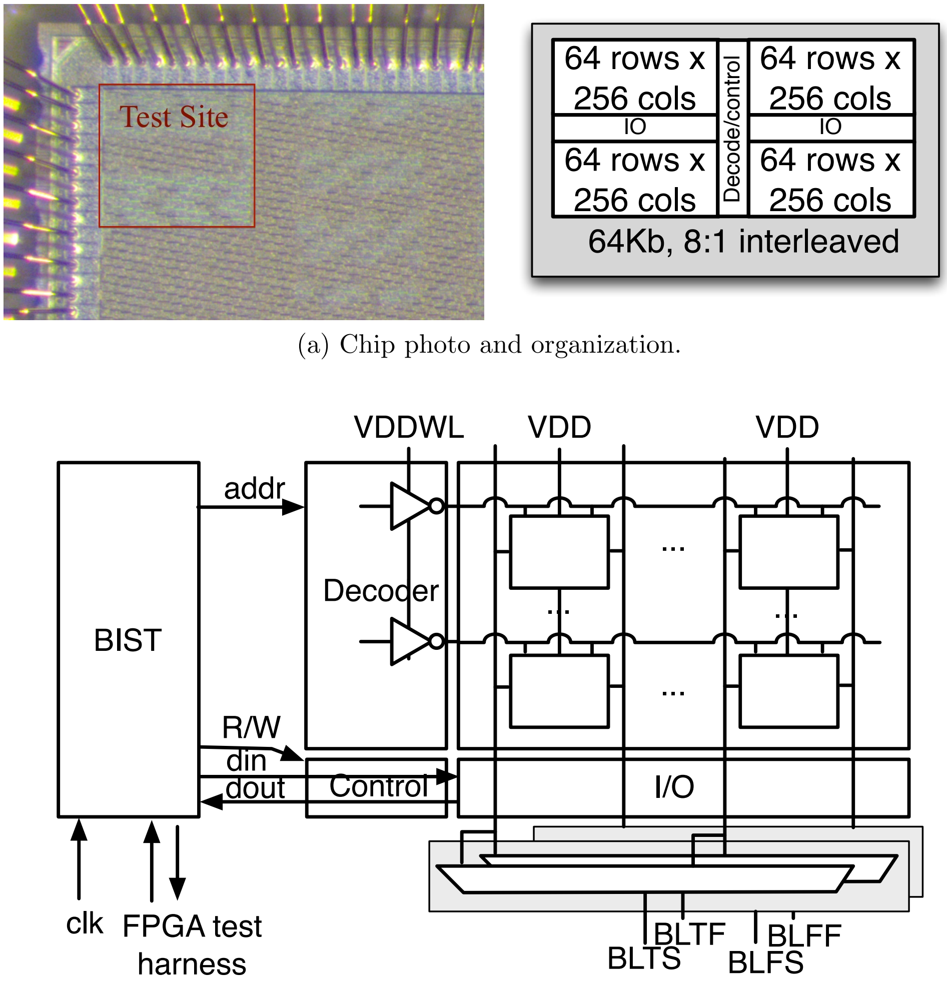
3.2 Measuring Failure Mechanisms[2]

Measurements of bitcell failures are necessary to validate the assumptions that simulation models make. In particular, simulation results are extremely dependent on assumptions about the distribution of transistor threshold voltage. The testchip described in the following sections was used to measure the variation in threshold voltage for 32k bitcells.

3.2.1 Introduction

Accurate characterization of cell failures at SRAM V_{min} requires Monte Carlo simulation of devices with calibrated variation models, and simulation predicts that cells with a parametric shift beyond an Nx6-dimensional failure contour will have persistent failures, where each of the N dimensions represents a varying quantity in the cell. Additionally, random telegraph noise (RTN) varies the threshold voltage of transistors over time, and can cause intermittent failures of usually functional cells [49]. It has been predicted that scaling will make RTN a more significant source of error [19]. However, the joint effect of RTN and random variation on SRAM failure is still not well understood. This work investigates the validity of the failure contour, and explores the interaction between persistent and intermittent causes of SRAM bitcell failures. Measurements are taken from cells within an SRAM array, not padded-out cells, which enables more sample points and increases certainty that the results will be applicable for production SRAM arrays.

[2] The content in this section was derived from a publication in ESSDERC 2014 [48].



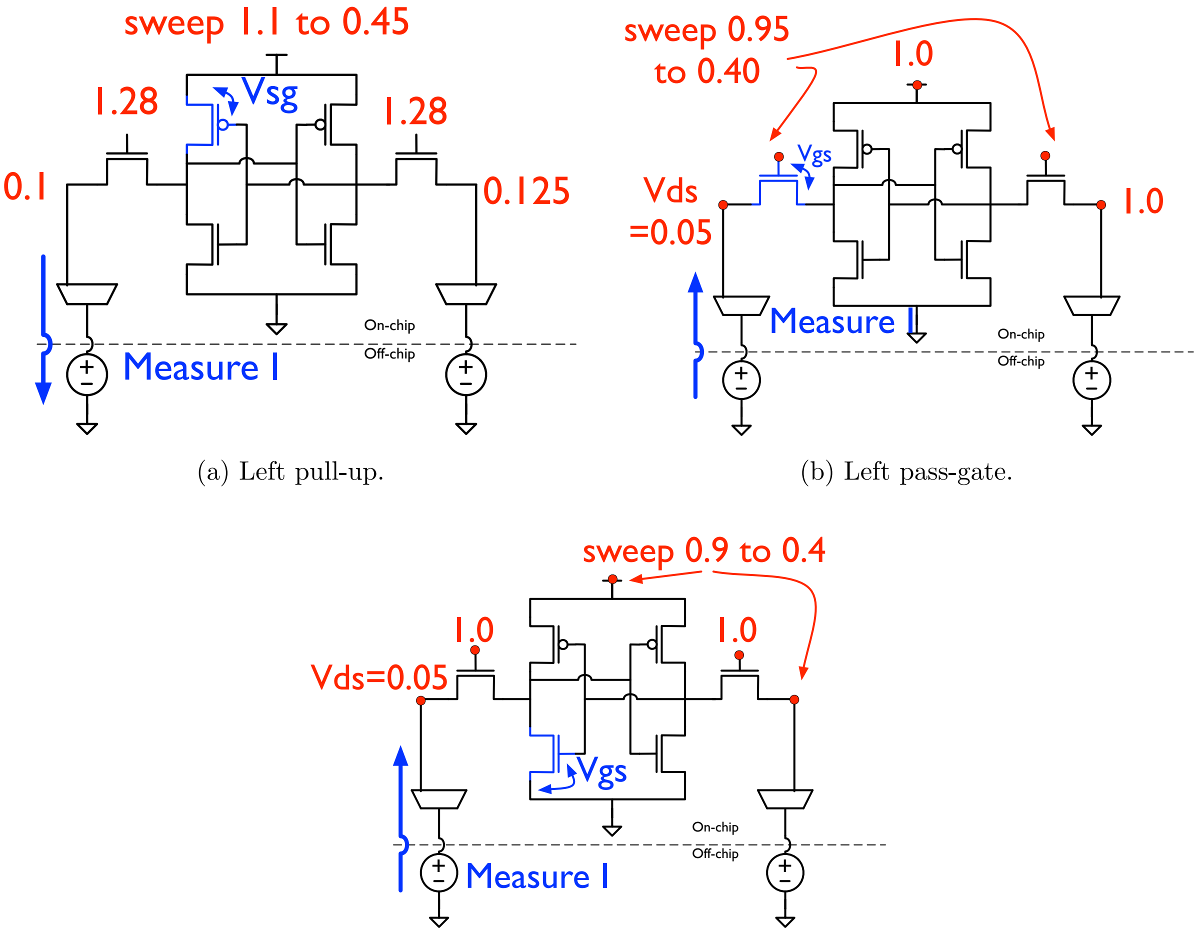
(a) Chip photo and organization.

(b) Characterization system architecture.

Figure 3.7: 28nm characterization testchip details.

3.2.2 Characterization Architecture

The study is applied to an array of 32k $0.120\mu m^2$ bitcells in a pre-production HKMG 28nm process [50]. Both bulk and FDSOI wafers were manufactured using the same mask set to provide a unique opportunity for comparison between FDSOI and bulk. Bulk and FDSOI have different gate stacks to adjust V_{th}. Figure 3.7 shows the chip photograph, physical organization, and characterization architecture. A programmable BIST enables dynamic measurements, and separated supplies, combined with a bitline multiplexer, enable static IV measurements similar to [17].



(a) Left pull-up.

(b) Left pass-gate.

(c) Left pull-down.

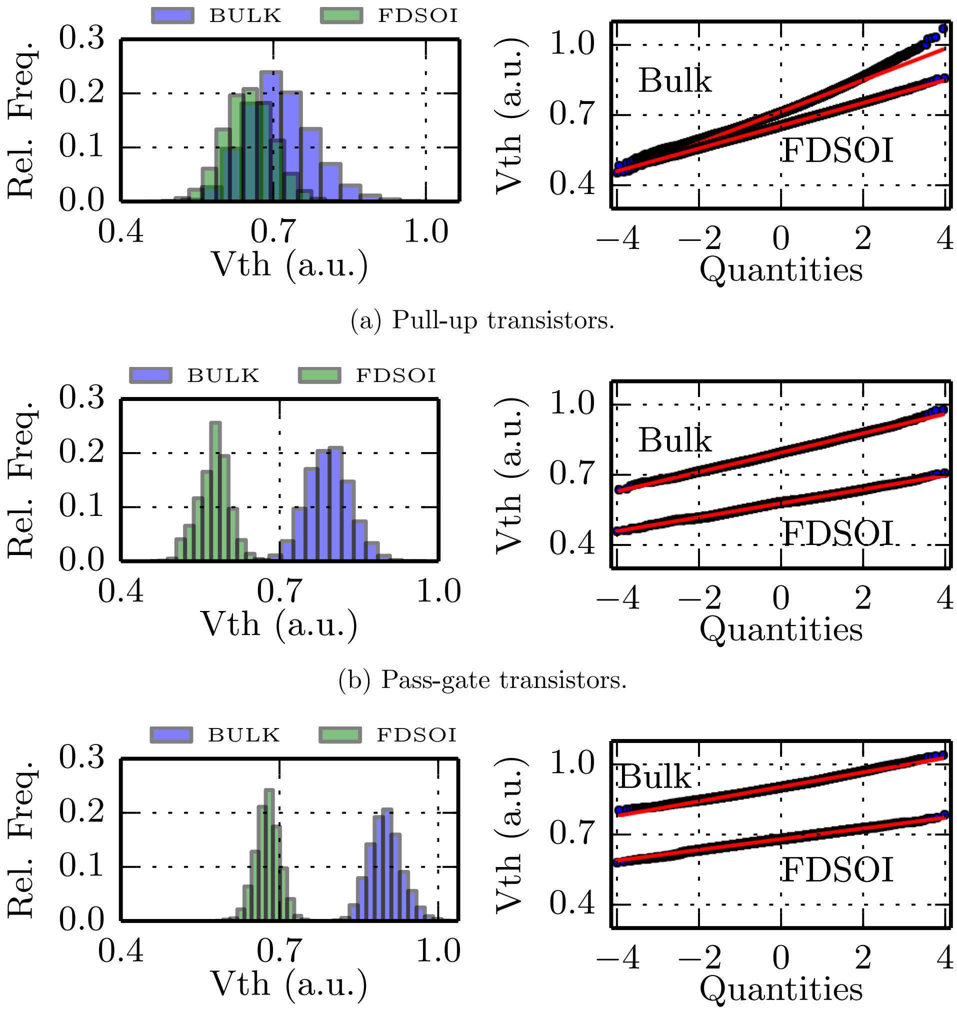
Figure 3.8: Scheme used to measure V_{th} of each transistor in the SRAM array.

3.2.3 Random Variation Measurement

Threshold-voltage variation is measured for every transistor in the SRAM array by using a modification of the direct bit transistor-access scheme (DBTA) [17]. The exact measurement setup is depicted in Figure 3.8. Multiplexed bitlines and separated wordline and cell voltages enable IV measurements of every cell in the array.

Figure 3.9 shows the Gaussian distribution of V_{th} for the pull-up, pass-gate, and pull-down transistors of 32,000 cells. FDSOI devices have approximately 27%, 26%, and 28% lower standard deviation of V_{th} than bulk devices for the pull-up, pass-gate, and pull-down respectively.

Because IV measurements are not performed on isolated devices, a transient simulation



(a) Pull-up transistors.

(b) Pass-gate transistors.

(c) Pull-down transistors.

Figure 3.9: Histogram and normal QQ plots of measured V_{th} distribution for 32k cells from both FDSOI and bulk chips.

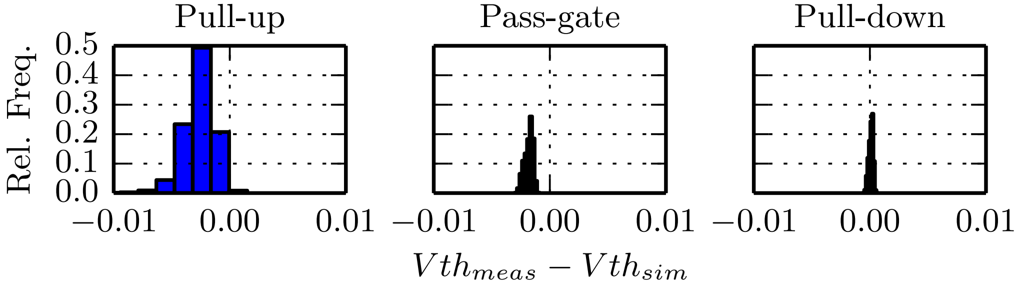


Figure 3.10: V_{th} measurement difference between measured V_{th} and simulated scheme using transistor V_{th} shifts from measurement.

is performed to provide confidence that the V_{th} measurement of a single transistor is not affected by other transistors in the cell. The simulation netlist represents the actual V_{th} measurement scheme and uses cell V_{th} shifts annotated from actual measured data, and Figure 3.10 plots the distribution of simulated measurement error of the DBTA scheme.

3.2.4 Random Telegraph Noise Measurement

Random Telegraph Noise (RTN) is caused by the trapping or de-trapping of a carrier in a transistor, and manifests itself as a temporal shift in threshold voltage. At smaller device geometries, the impact of RTN approaches that of RDF [51]. Because SRAM is not guaranteed to be tested while the bitcells are in their worst-case state, the impact of RTN must be well understood to appropriately margin against failures during operation.

RTN was measured using the alternating-bias technique [52]. Figure 3.11 provides procedure details, and shows example waveforms of the alternating-bias technique versus a conventional RTN measurement. Traditional schemes can find trap emission and capture time constants as a function of gate voltage, but require sweeping the gate voltage and long measurement times. The alternating-bias technique speeds up the procedure by alternately turning on and off the gate before measurement, which attempts to force either emission or capture before the testing period. While this technique emphasizes worst-case RTN effects, it closely emulates the real operating conditions of a cell, where devices are either turned on or off immediately before an access.

The measured amplitude distribution in Figure 3.12 shows RTN-induced changes in drain current around the threshold voltage follow a log-normal distribution with a long tail. Both pull-down and pass-gate NMOS devices show similar RTN amplitudes, but for the pull-up PMOS devices, RTN amplitude is slightly higher in FDSOI than in bulk.

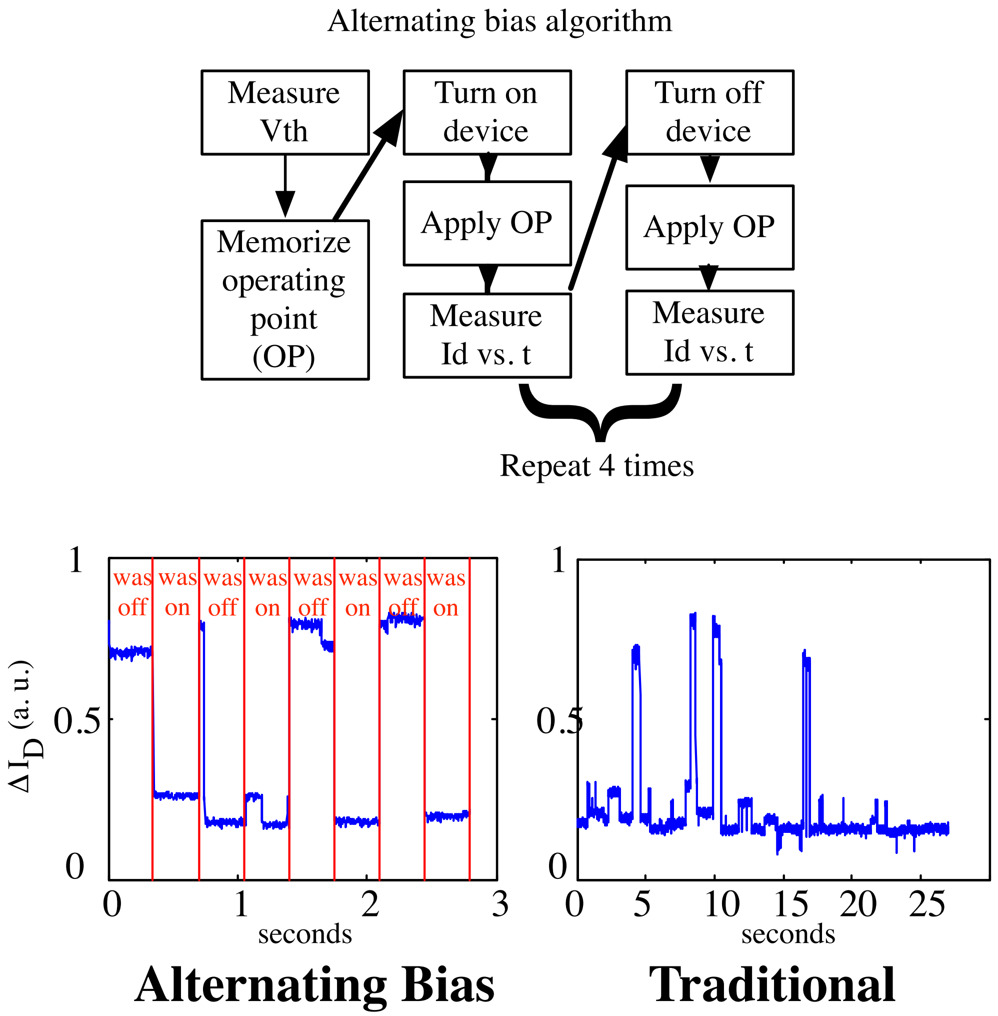


Figure 3.11: Alternating bias versus conventional RTN measurement scheme.

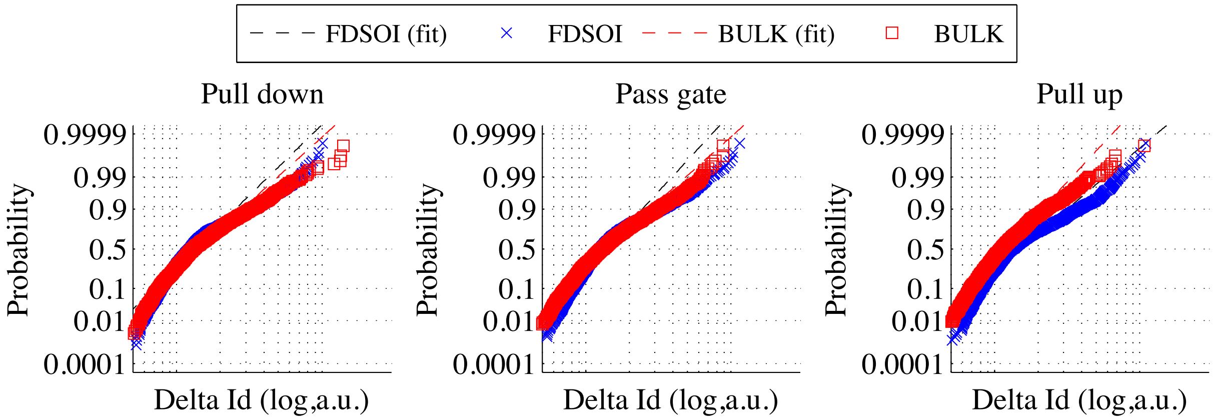


Figure 3.12: Log-normal probability plot of RTN-induced current differences at cell V_{th} using the alternating-bias technique.

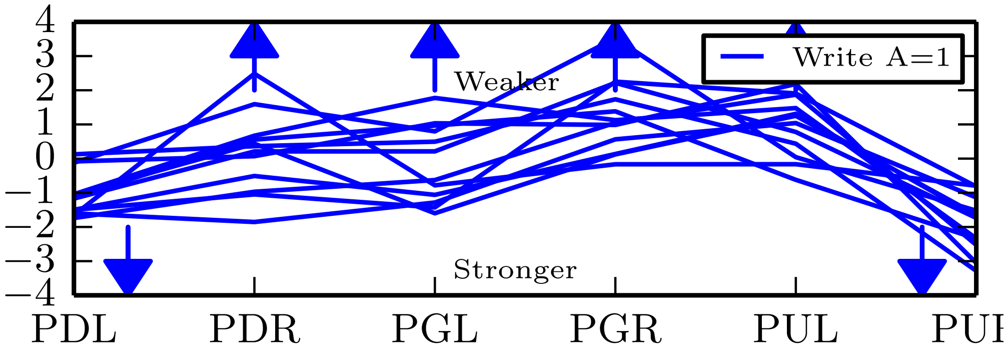


Figure 3.13: V_{th} shift vector for cells failing to write 1. Arrows indicate worsening ability to write A=1.

3.2.5 Joint Effect of RTN and Variation on Writeability

Measurement results confirm that writeability failures are caused by V_{th} shifts induced by both random variations and RTN. Writeability is measured by initializing a value in a cell at a safe V_{DD}, writing the opposite value at the test V_{DD}, then checking for a correct write at a safe V_{DD}. The ability to write both values is measured for the entire array at decreasing voltages. Figure 3.13 shows the V_{th} shifts of the first 12 failing cells with arrows indicating the V_{th} shift direction that worsens writeability for each device. Only writeability failures are analyzed, and cells that are unstable at target V_{DD} are filtered (to avoid simultaneous measurement of a cell's ability to write A without flipping back to $\bar{A}$ during a half-select).

The effect of RTN on dynamic writeability can be measured by applying two different write schemes with opposite resting values that expose different RTN behavior, as shown

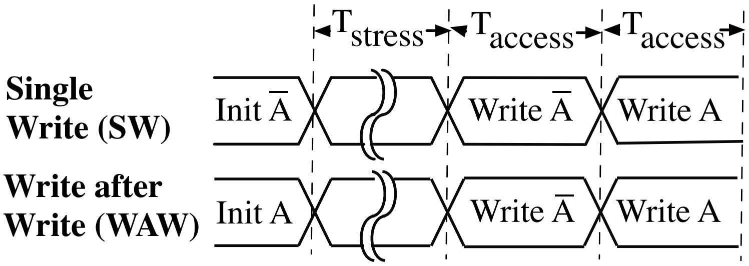


Figure 3.14: BIST waveforms of two different writeability tests that expose RTN, where $T_{stress} = 1s$ and $T_{access} = 300ns$ with a 50% duty cycle.

in Figure 3.14 [4]. The dynamic write patterns are exactly the same, so without RTN V_{min} should be exactly the same. However, different resting values force different trap occupancy immediately before the writeability test, which exposes the effect of RTN on writeability. The difference in V_{min} between the single write (SW) mode and write-after-write (WAW) mode will be referred to as V_{diff}. This effect can cause problems for production BIST tests at low voltage, because failures at a specific voltage depend on trap occupancy. Therefore, understanding V_{diff} is required to accurately margin BIST measurements of V_{min}.

Figure 3.16 shows how RTN causes a V_{min} difference between the two write modes for writing a 1 to a specific bitcell. Random-variation-induced V_{th} shift weakens the cell. Alternating-bias RTN measurements show that when the PDL is left off, it will have a higher current than when it is left on. For the WAW mode, Figure 3.15 shows that the bitcell holds a 1 for a long period and the PDL is off, so RTN will cause higher current and therefore lower V_{th}, which makes failure in the WAW mode more likely than the SW mode.

The distribution of V_{diff} values for the first 25 failing cells in bulk and FDSOI is shown in Figure 3.17. Larger V_{min} differences in FDSOI suggest that while RTN ΔI_D amplitude is similar between FDSOI and bulk, the lower standard deviation of the random variation in FDSOI increases RTN's impact on cell failures.

The effect of V_{diff} on overall chip V_{min} can be evaluated by counting the number of cells that fail at each voltage for both SW and WAW (no RTN effect) versus the number that fail for either SW or WAW (including RTN). Figure 3.18 plots the V_{min} difference due to RTN for six different chips normalized to the same voltage. Note that the effect of RTN is suppressed for the entire array because the cells with the largest RTN effect are not necessarily the cells that limit array V_{min}.

3.2.6 Summary

Measurement of transistor threshold voltage in 32,000 bitcells shows that FDSOI technology reduces threshold variation due to random variations by 27% compared to bulk. The alternating-bias technique reveals similar threshold variation due to RTN in bulk and FDSOI for NMOS devices, and slightly increased RTN amplitude in FDSOI for PMOS devices.

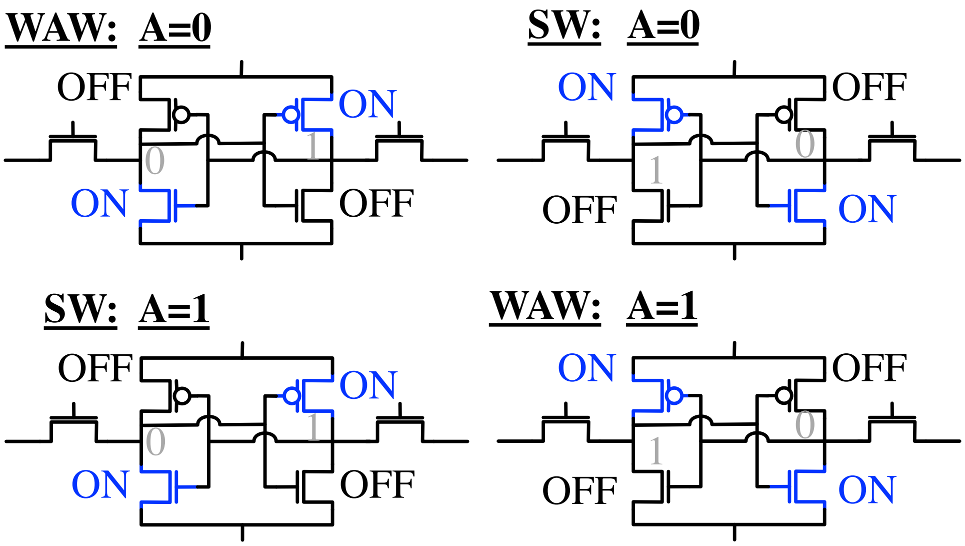


Figure 3.15: Transistor stress state for different write modes and values.

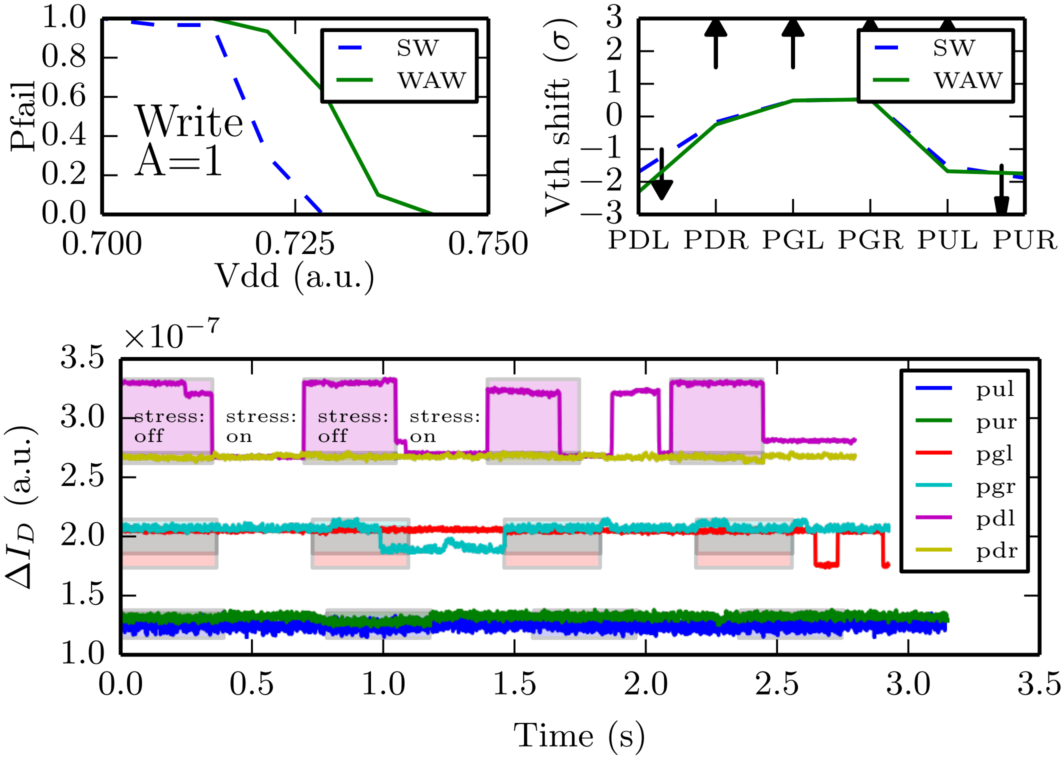


Figure 3.16: Example measured effect of RTN on V_{diff} for a specific bitcell.

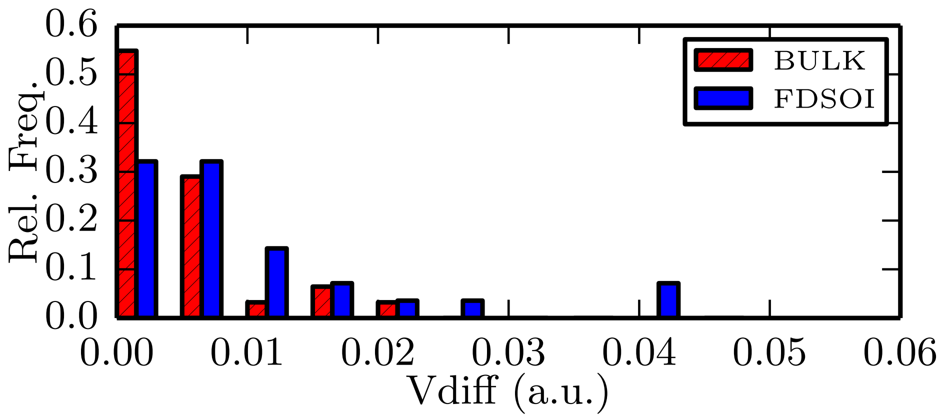


Figure 3.17: Distribution of V_{min} difference between write modes.

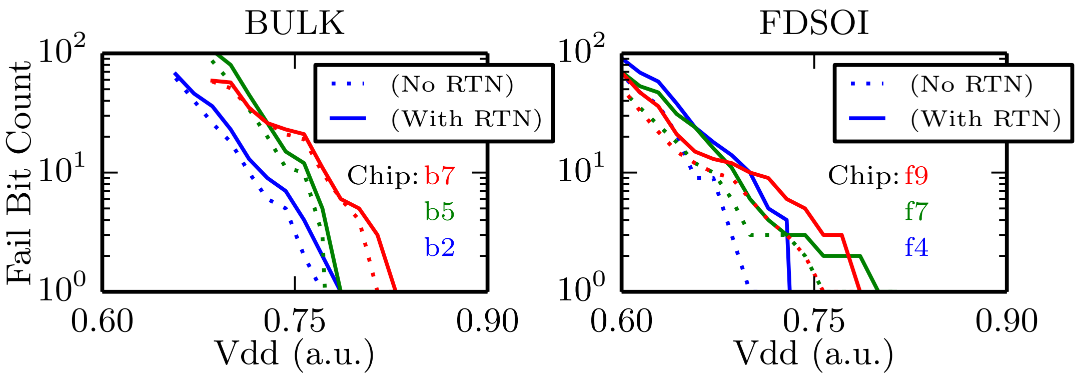


Figure 3.18: Difference in V_{min} for six different chips.

Dynamic measurement of writeability in both FDSOI and bulk shows that the decreased random variation in FDSOI enables an approximately 7% reduction in V_{min}, but also exacerbates the effect of RTN on minimum operating voltage failures, suggesting that even as intrinsic channel devices reduce the relative amount of random variation, RTN will emerge as an obstacle to future voltage scaling and increase voltage margins for BIST results.

Chapter 4

Circuit-level Resiliency Techniques

The analysis methodology for simulating bitcell failures described in Section 2.2, an investigation of assist technique effectiveness in Section 3.1, and analysis of bitcell failure causes based on experimental measurement in Section 3.2 all provide insight into using circuit-level resiliency techniques to lower SRAM V_{min}. In this chapter, two circuit-level techniques are proposed. The first, a single-p-well bitcell, exploits FDSOI technology to maximize bitcell writeability for different process corners. The second, a wide-voltage-range 8T macro, uses assist techniques and pseudo-differential sensing to improve speed and reduce energy consumption of SRAM macros.

4.1 Introduction

At the circuit level, increasing cell size to reduce variation can reduce the minimum operating voltage (V_{min}) by over 200 mV [42]. Additionally, different cell topologies, such as the single-ended 8T cell, can decouple conflicting sizing requirements to increase resiliency and performance at the expense of additional area [20], [53], [54]. Assist techniques, analyzed in Section 3.1, improve resiliency by temporarily overdriving or under-driving the voltages on critical transistors in the cell, and can improve V_{min} at the cost of area and energy overhead. For example, the cell V_{DD} can be temporarily collapsed during a write operation to lower V_{min}, but extra transistors are required and energy overhead is increased by discharging cell V_{DD}during every write operation [55]. These assist techniques have grown increasingly necessary as new FinFET-based technologies prevent using conventional transistor sizing to balance read and write failure mechanisms within a cell. The strength of assist techniques can be tuned based on the process corner to correctly balance the ratio between NMOS and PMOS to optimize the read stability versus writeability conflict [56].

Finally, a solution called “dual-rail” places the SRAM on a separate and higher supply voltage than the rest of the logic [57]. However this involves routing two supplies over the core, increases latency due to level shifting, increases verification complexity, and prevents SRAM energy scaling. Even though SRAM account for only 1/4 to 1/2 of chip energy, avoiding SRAM voltage scaling quickly causes SRAM energy to dominate system energy as the system supply voltage is lowered.

4.2 Single-p-well Bitcell[1]

Generally, circuit-level resiliency techniques focus on periphery circuit changes, such as assist techniques, to improve low voltage operation. With the exception of the 1R1W 8T cell, the industry has standardized on the 6T bitcell. Adding more than six transistors dramatically increases cell area, and therefore very few bitcell-modification ideas have been considered worthwhile enough to be used in products.

A 28nm ultra-thin body and buried oxide (UTBB) FDSOI technology implements a thin silicon layer on top of a buried-oxide (BOX) [50]. FDSOI eliminates channel doping to lower intrinsic transistor variability, and the isolation between the channel and oxide enable both n-type and p-type wells below both NMOS and PMOS transistors. Additionally, the thin BOX isolation enables extended body biasing beyond the forward-bias diode voltage to change the speed versus leakage trade-off during runtime.

The proposed single-p-well (SPW) design, shown in Figure 4.1, places both PMOS and NMOS devices over a common p-well, leading to LVT PMOS pull-up (PU) devices and RVT pull-down (PD) and pass-gate (PG) NMOS devices. By changing the back-bias voltage, V_{PW}, the NMOS-to-PMOS strength ratio can be varied—enabling optimization of bitcell writeability, retention voltage, access time, and leakage power consumption. Positive V_{PW} reduces the V_{th} of the NMOS PD and PG (FBB) while increasing the V_{th} of the PMOS PU (RBB), while negative V_{PW} yields NMOS RBB and PMOS FBB. Derived from the standard high-density ($0.120\mu m^2$) SRAM bitcell architecture, SPW design does not require process and footprint modifications [59] and reduces scaling limitations caused by well-proximity and diffusion issues.

The testchip, summarized in Figure 4.2, provides a BIST that evaluates all SRAM failure mechanisms—writeability (WA), readability (RA), read stability (RS) and retention stability (RET)—in a 140kb SRAM macro. The BIST includes an on-chip pulse generator to accurately characterize the critical read and write pulse widths of dynamic failure metrics. A programmable finite-state machine orchestrates a wide variety of SRAM access patterns. The input and output from the chip is performed through a low-frequency scan chain.

Figure 4.3 summarizes the physical design of the SRAM macro. The wordline drivers and column IO circuits are placed in the middle of the array to reduce the resistance contributions of the long metal wordlines and bitlines, and the control circuitry is in the middle of the design. The bitcells are interleaved 8-to-1 to ease the layout of the column IO. Before the clock edge, the address predecode and decoder enables a single row driver. The rising edge of the clock turns on the wordline driver to access a row and generate a voltage differential

[1] The content in this section was derived from a publication in IEDM 2014 [58]. Brian Zimmer designed the SRAM periphery circuitry, Olivier Thomas designed the BIST with pulse generator and measured the chip.

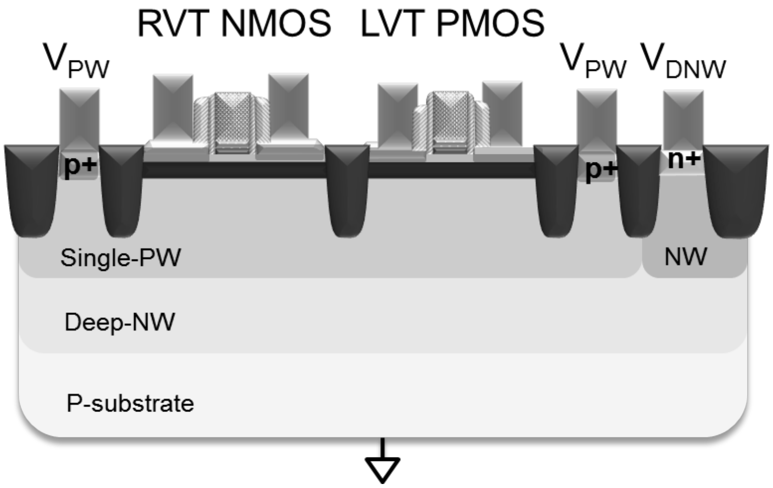


Figure 4.1: SPW bitcell device cross section view. DNW isolates the PW from the p-substrate enabling wide voltage range PW back biasing.

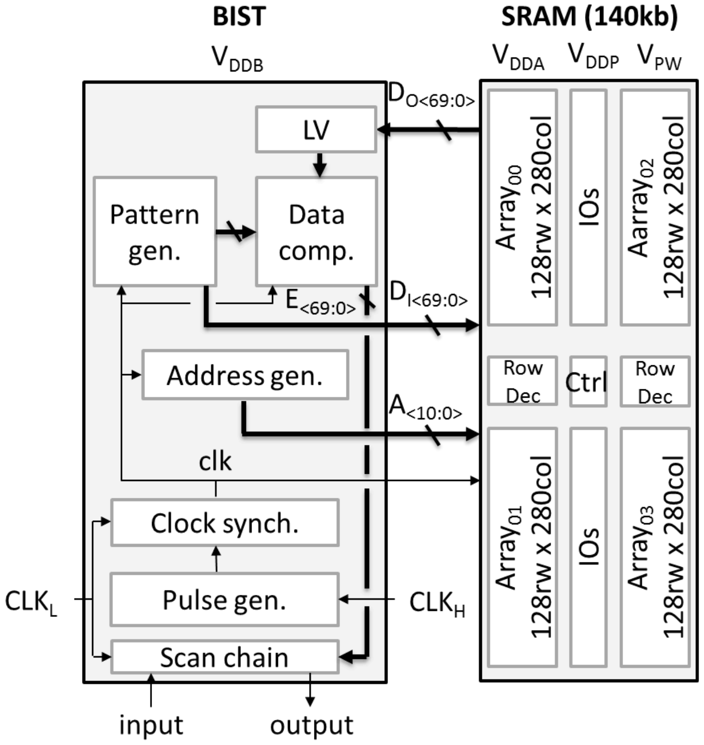


Figure 4.2: Dynamic characterization module architecture, including a 140kb SPW SRAM macro clocked by an on-chip pulse generator and controlled by a programmable BIST.

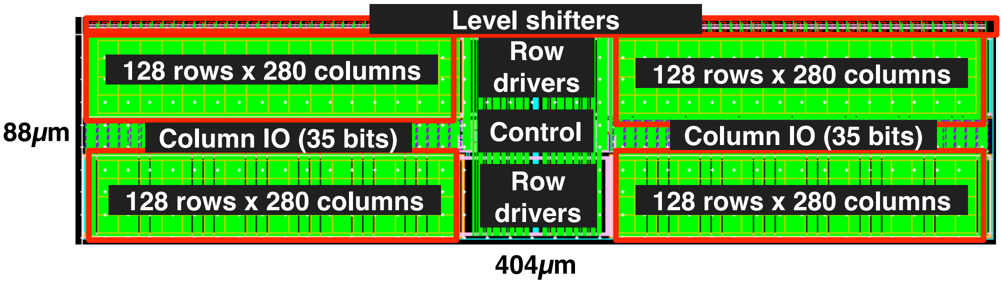


Figure 4.3: 6T based high density 32KB SRAM array for failure characterization.

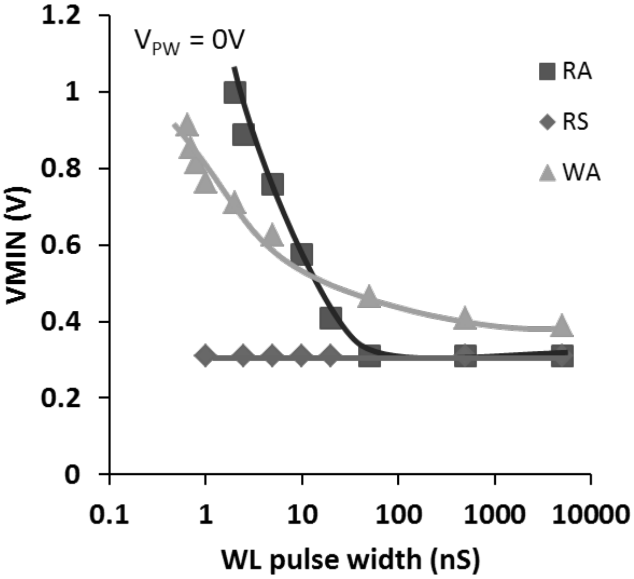


Figure 4.4: RS, WA, RA V_{min} versus WL pulse width.

on the bitline. The bitlines are multiplexed through PMOS pass gates to a StrongARM style sense amplifier, and an SR latch holds the output data until the next operation. The pass gate pulse width matches the high voltage period of the clock, and therefore is controlled by changing the duty cycle and frequency of the clock.

Unlike conventional March tests, the BIST characterization structure identifies the actual cause of failure by using high-voltage operations to guarantee successful reads or writes. Figure 4.4 plots failures due to RA, RS, and WA for different voltages and operating frequencies. Because readability is directly related to the amount of read current supplied by a cell multiplied by the access time, readability V_{min} is very sensitive to access time. Longer access times slightly improve writeability V_{min}, and read stability is not a problem for the proposed bitcell. For most voltages, the minimum operating voltage is limited by bitcell writeability.

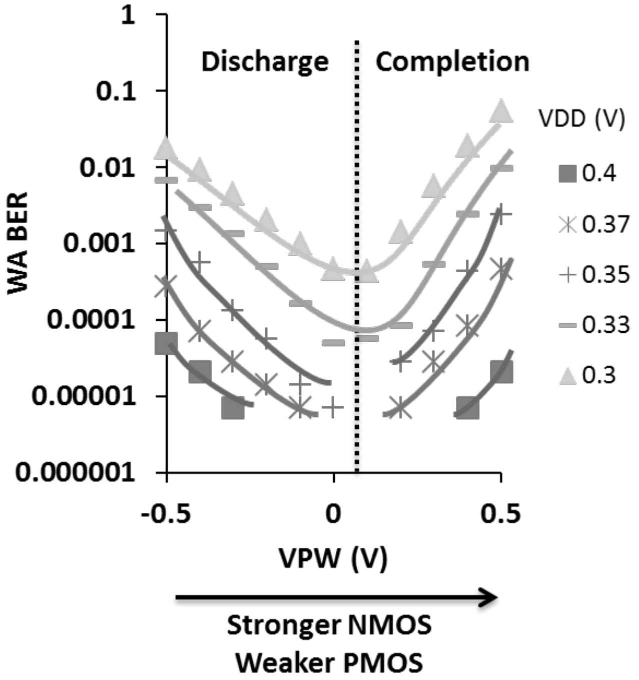


Figure 4.5: Writeability bit-error rate (BER) versus V_{PW} for different voltages.

Figure 4.5 shows that optimization of V_{PW} can minimize the writeability failure rate. When V_{PW} is negative, NMOS are weaker, and PMOS are stronger, and in the weakest bitcells it is difficult for the pass gate to pull the high node low, and therefore inadequate discharge of the high node limits writeability. When V_{PW} is positive, NMOS are stronger, and PMOS are weaker, and while the high node is easily discharged low, the low node cannot be pulled high through a weak PMOS by the end of the cycle. Therefore, V_{PW} can be adjusted to compensate for systematic process skew between NMOS and PMOS strength.

The SPW bitcell has a retention voltage of 305mV, a minimum leakage of 100fA per cell, and body bias can improve readability and writeability to compensate for process variations for a less than 5% increase in leakage current. Understanding of SRAM failure mechanisms and exploitation of a new FDSOI process technology improves the minimum operating voltage of a standard 6T bitcell with no additional required masks.

4.3 Wide-voltage-range 8T macro

A custom SRAM macro was designed for the RAVEN project to operate at extremely low voltages (down to 0.45V) and tolerate significant voltage noise on the power rail (up to 100mV ripples at high voltage). The RAVEN project using on-chip voltage conversion with switched-capacitor converters to enable finer-grain DVFS algorithms and simplify package

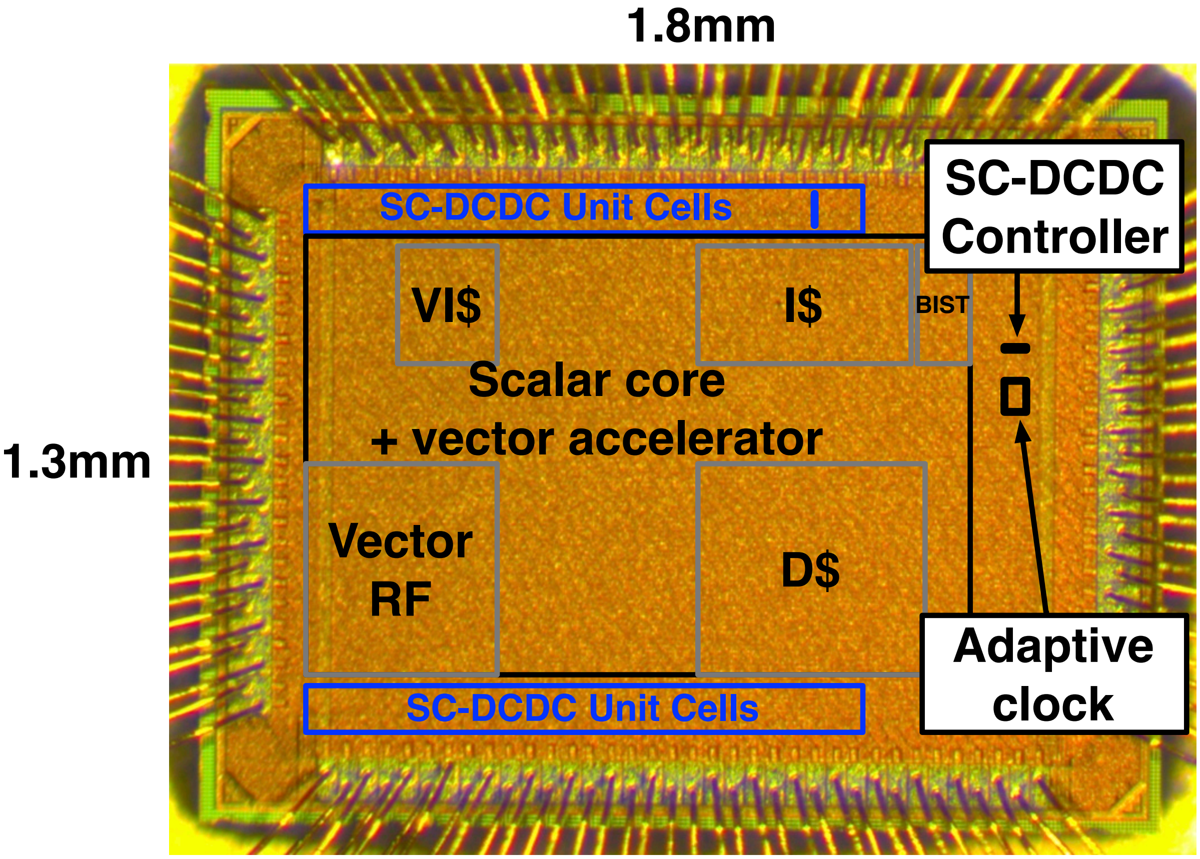


Figure 4.6: A RISC-V processor with on-chip voltage conversion fabricated in 28nm FD-SOI [1].

design. Previous approaches to switched-capacitor converters interleave converters in order to stabilize the output voltage, but incur losses due to charge sharing between the phases. The RAVEN project proposes using non-interleaved operation to switch all converter unit cells at the same time to avoid these losses, but generates a voltage ripple at the output. An adaptive clock exploits the voltage ripple by translating higher instantaneous voltage into higher operating frequency. Reconfigurable switches enable conversion of standard 1V core and 1.8V IO input voltages to four voltages between 0.5V and 1V. To prove the concept, a $2.4mm^2$ chip shown in Figure 4.6 contains a 64-bit RISC-V in-order scalar core and 64-bit vector accelerator and is implemented in 28nm FDSOI technology [1]. The processor boots Linux and runs Python, operates at a maximum frequency of 960MHz, can operate down to 0.45V, and has a measured system conversion efficiency of 80-86%.

Two versions of the SRAM have been designed with circuit-level resiliency techniques to improve low voltage operation. The first version, shown in Figure 4.7, was used in the

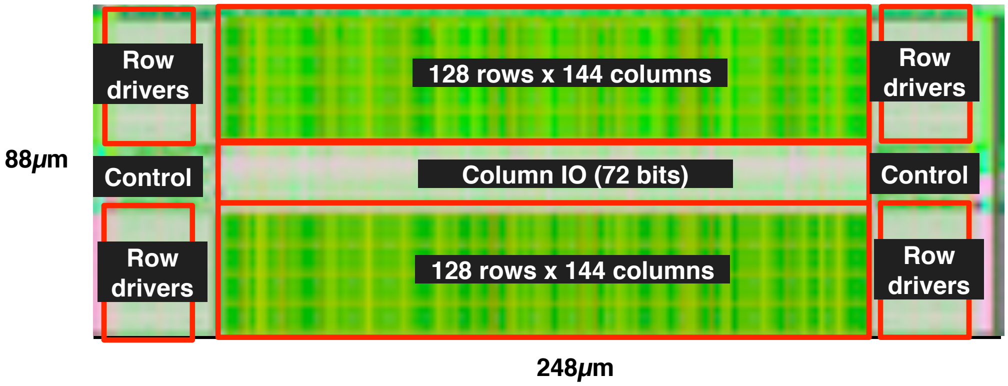


Figure 4.7: Custom low-voltage SRAM designed for the RAVEN3 project.

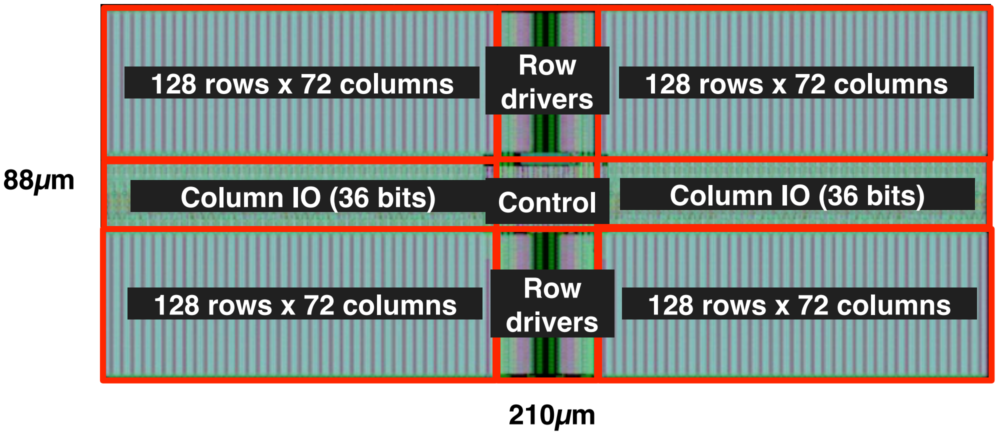


Figure 4.8: Custom low-voltage SRAM designed for the RAVEN3.5 project.

L1 cache and vector register file in the RAVEN3 chip. The SRAM was functional down to 0.45V, but the BIST control logic failed at 0.45V, preventing SRAM characterization below this voltage.

The second version, shown in Figure 4.8, has a variety of performance improvements, and was implemented as a test site in the RAVEN3.5 chip. The wordline drivers for both the read and write port were moved to the middle to reduce resistance of the wordlines, and thereby reduce area by 15%. Delay buffers were replaced with inverters to balance rise and fall time and reduce the possibility of variation causing a timing pulse to be stretched or compressed. Additionally, the minimum pulse width was reduced for further potential energy savings. The BIST runs at high voltage, while the SRAM runs on the lower testing voltage, to ensure that the true minimum operating voltage of the SRAM can be measured.

4.3.1 Design Overview

There are two major goals of the custom SRAM macros in the L1 cache. The first goal is to build an SRAM that can operate from a rippling on-chip DC-DC supply down to 450mV. The second goal is to reduce swing on the bitlines in order to improve delay at low voltage and minimize switching energy.

To achieve both goals, the SRAM uses a timing-replica circuit to turn off the wordline after the minimum required bitline voltage is developed by the worst cell in the array, and uses a sense amplifier to correctly read values from the resulting low-swing bitline. At low voltages, this scheme improves read speed as the added delay of the sense amplifier is less than the delay of waiting for a full-rail swing. At high voltages, energy is reduced due to the lower required voltage swing on the bitlines.

The SRAM macro has 256 rows and 144 physical columns which are logically organized as 512 entries of 72 bits (64+8 for ECC). Figure 4.9 illustrates how the 144 physical columns are multiplexed into 72 input and output bits. Only metal up to M4 is used, and the power grid is exposed in M4. The cells in the array are physically interleaved 2:1 (every other column belongs to the same logical word). Interleaving reduces layout complexity for the column circuitry, but causes a couple of problems. First, when writing a word, the other word in that row will see a read operation, which requires the bitcell to be sized for stability. Second, write accesses to a different address, but on the same row, cause a voltage bump or drop on the internal node that either slows down reading a 1, or misreads a 0 as a 1 [60]. However, the bitcell provided by the foundry was read stable based on importance-sampling simulations, and the voltage bump was small enough to not affect read operations. These disadvantages can be removed by not interleaving cells [21].

The cell itself is an 8T cell (6T cell + 2T decoupled read port) with dedicated read wordline and write wordline allowing for a single read operation and a single write operation per cycle. Instead of conventional single-ended reads, this design uses an unaccessed bitline as a reference, then uses a sense amplifier to read data with reduced swing on the bitlines. Writes can be masked at a 9 bit granularity to support a parity bit on every byte, and a negative bitline assist improves writeability at low voltages. A concurrent read and write access is allowed, but conflict detection and resolution must be implemented outside the array. There cannot be simultaneous accesses to the same word because the read output value will be unknown.

The read and write timing control blocks were designed by using standard cells at the schematic level, which were then automatically placed-and-routed with Synopsys IC Compiler$^{\text{TM}}$ and inserted into the design. The pre-decoding logic was synthesized from a Verilog description. The rest of the layout was full-custom. The wordline drivers are interleaved 2:1 and therefore allow 4 pitches of poly for each driver gate. The column IO is also interleaved 2:1 to provide room for a common-centroid sense-amplifier layout that minimizes offset from local mismatch.

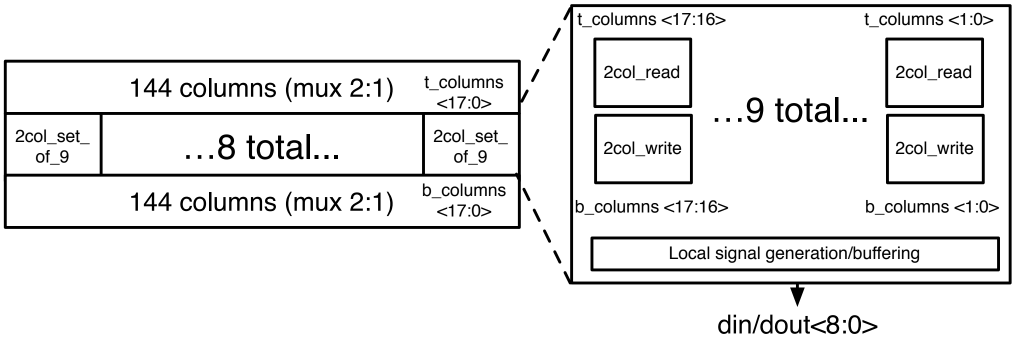


Figure 4.9: Column IO organization

Control signals

The array has many configuration options that allow the array to adapt to different process corners and operating conditions. Timing can be configured to use only the rising edge of the clock, or both the rising edge and the falling edge. When the rising edge only is used, internal delay chains (built from either inverters or replica bitcells) generate the wordline and precharge signals. When the falling edge is also used, the falling edge turns off the wordline and turns on the precharge signal.

The SRAM operates behaviorally as a synchronous element. Every signal to the SRAM has a setup and hold time around the rising edge of the clock. The SRAM latches the data three different ways internally: with a flip-flop, with a latch, and with a pulsed S-R NAND latch. All of the control signals are assumed to remain stable, but still use a flip-flop for safety. The flip-flop is triggered by the rising edge of either the read clock or write clock. The address signals are latched, so their values can propagate through the address decoder before the edge arrives. These latch signals are controlled by an internally generated signal based on the precharge signal, so whenever the bitlines are precharging the input latches will be transparent, and during an operation, the latches will hold state.

Programmable wordline and precharge timing is built around an SR latch, shown in Figure 4.10. One input is connected to the clock input, which turns on the wordline and turns off precharge when the clock edge arrives. The other input is sent a pulse to turn off the wordline, and the pulse can be generated from the falling edge of the clock, a delay path, or the replica circuit. A similar mechanism is used to control the precharge signal.

Library Characterization

Correctly measuring timing information on SRAM input and outputs is important because the SRAM will generally be on the critical path. Additionally, long internal clock paths (specifically to the data input flops) generate a substantial hold time that must be carefully measured after extracted simulations. The macro was characterized using Synopsys

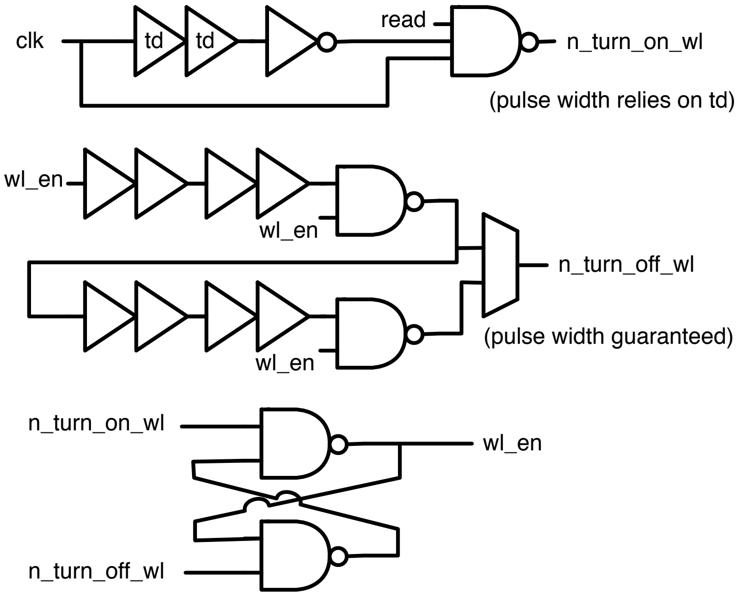


Figure 4.10: Read wordline timing.

Siliconsmart^{TM}, which runs transient simulations on the extracted layout to calculate timing constraints for different voltages and process corners.

4.3.2 Writeability Assist

Simulation predicted that V_{min} of the proposed array would be limited by writeability. Based on results from Section 3.1, the negative-bitline-boost assist technique was found to be most effective for our technology. This assist has gained rapid popularity in recent years, with Samsung [61] and TSMC [62] reporting macros with negative-bitline assist in 14nm and 16nm respectively.

The negative bitline assist has two inherent problems. First, tuning the amount of bitline boost over a wide voltage is necessary to avoid accelerated bitcell aging from an excessive over-drive at nominal voltages. Digital configuration bits that either change the amount of capacitance switched [63] or change the voltage swing across the boost capacitor [62] can be used to tune the amount of assist. Second, boosting incurs an energy overhead, so energy can be optimized by only using the minimum amount of boost for each array [63].

The proposed negative-bitline assist scheme is described in Figure 4.11. First, the bitline that will write low is pulled to zero and the bitline that will write high is pulled to V_{DD} with appropriate control signals based on the address and data input. Then, for a regular write operation, the wordline is turned on to transfer the contents of the bitlines into the cell. For an assisted write operation, the NMOS that pulled the bitline to zero is turned off to leave the low node floating, a programmable bank of capacitors, shown in Figure 4.12, pull the low node below zero. The NOR gates driving the bitlines use the boosted node as the ground node to pull the gate voltage below ground as well and prevent the off-NMOS transistor connected to the boosted node from leaking due through a non-zero V_{GS}. The programmable capacitor bank uses NMOS capacitors to achieve high area density, and the entire scheme adds an additional 3.9% area overhead to the macro. Due to the 2:1 physical interleaving, the boosting capacitor area overhead is minimized by sharing the capacitor between four pairs of bitlines.

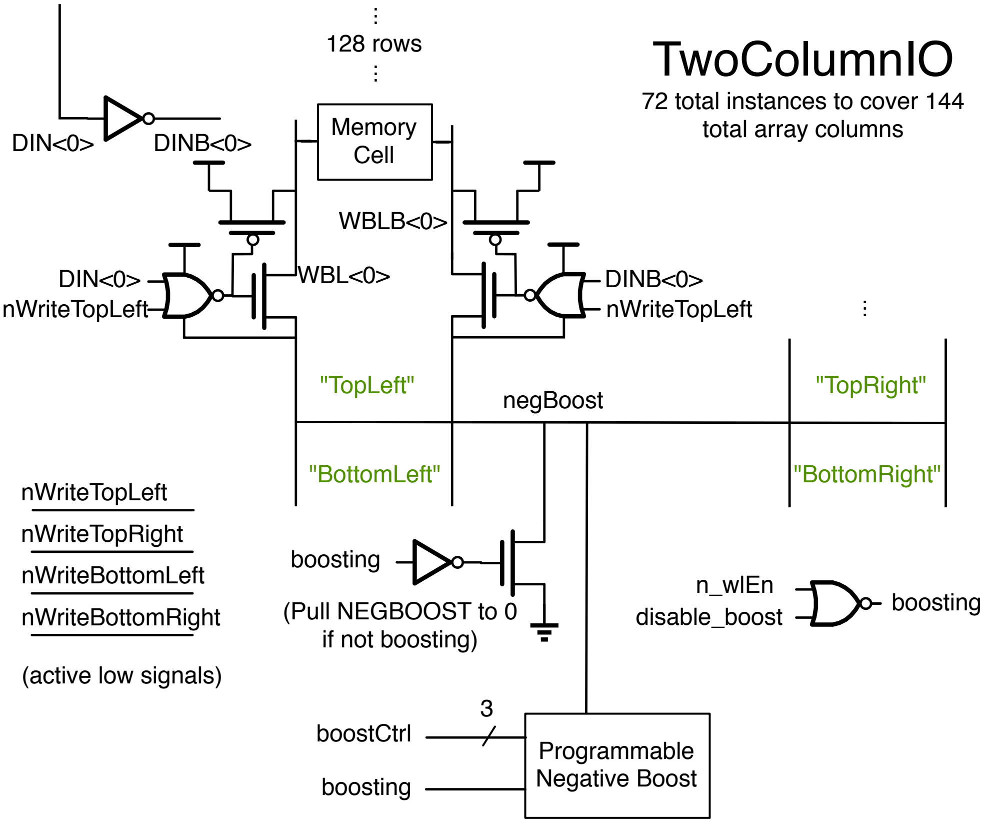


Figure 4.11: The proposed writeability assist scheme generates a negative voltage on the bitlines to lower V_{min} for write operations.

Figure 4.13 shows peak programmable boost amount over different voltages. The design achieves approximately 15-20% negative boost across the entire operating range. Calibration based on measurement will determine the optimal tuning code to use for different voltage regions.

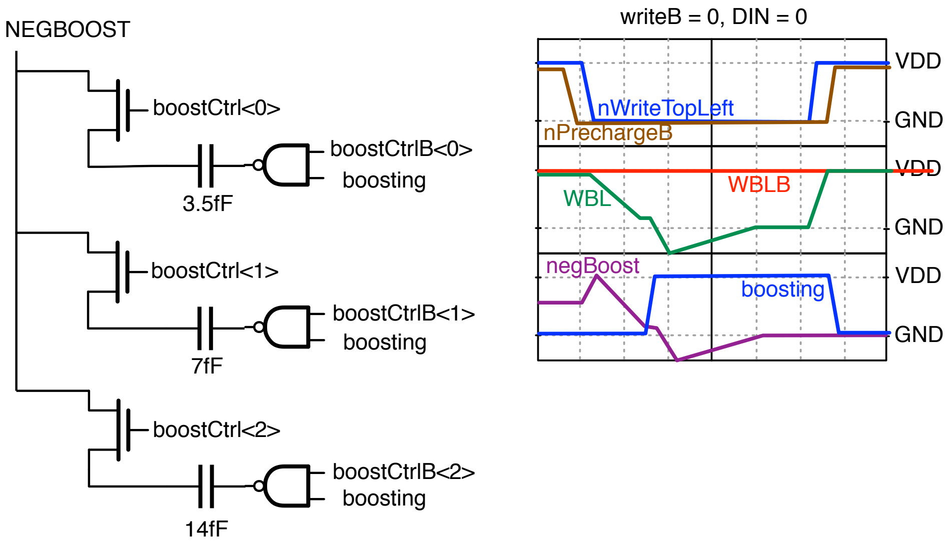


Figure 4.12: A bank of programmable capacitance generates different strengths of negative assist to optimize energy at different voltages and process corners.

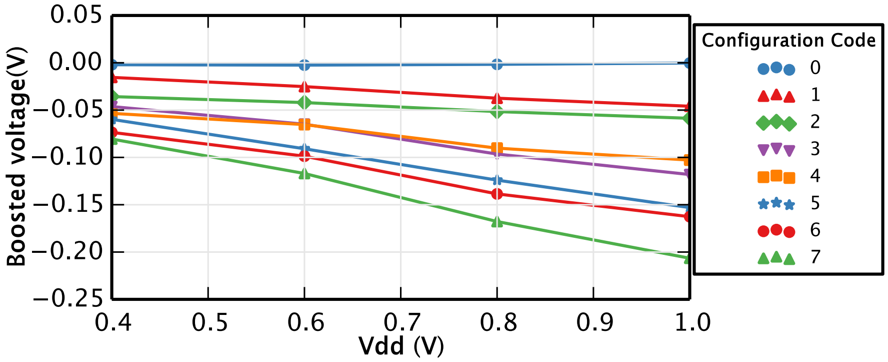


Figure 4.13: Resulting negative boost amount for different configurations.

4.3.3 Readability Assist

Traditional 8T-based designs use single-ended full-swing bitline reads. To improve read speed and decrease energy, local bitlines of 8 to 32 cells use full-swing reads, and transfer the data onto lower swing global bitlines, but this breaks up the periodic layout of the array and increases area overhead [22]. Another approach that maintains periodic layout would remove local bitlines, and instead only require low-swing on longer bitlines of 64 to 256 cells, to both improve read speed (because less time is required to wait before the cell discharges enough of the bitline capacitance) and decrease read energy (because the cell is turned on for less time). However, a low-swing single-ended read needs two problems to be solved—first, a new signal is required to trigger the sense amplifiers at the perfect time after the weakest cell in the array has finished reading, and second, a reference signal is required for the sense amplifier to distinguish between a read-0 and read-1.

Proposed low-swing single-ended read scheme

Figure 4.14 shows the proposed single-ended replica-timing-based read scheme, and Figure 4.15 shows the operational waveforms. One input to the sense amplifier is provided by the appropriately multiplexed bitline. The other input to the sense amplifier needs to be a reference that differentiates between low and high values. Ideally, this reference is equal to V_{DD} minus the input voltage offset of the sense amplifier. To generate the reference, a pre-discharged capacitor is connected to unaccessed bitlines to reduce the voltage on the reference bitline to a programmed level. Even though this method requires extra area for capacitors, it functions even with extreme supply noise, as both the target and reference bitline will be precharged to the same voltage. The reference bitline is guaranteed not to be accessed, as only a single read operation is supported per cycle, and a write operation will not affect the read bitline. The 2:1 interleaving of the bitcells is possible because the cell was determined to be read stable through simulation.

Replica timing

The proposed low-swing approach requires a carefully timed replica signal to turn off the wordlines and fire the sense amplifier at the perfect time. A conventional full-swing design still requires a signal to begin the precharge operation, but the overall delay and energy consumption will be much less sensitive to this signal. Recent research has sought to improve the sense-amplifier enable signal (SAE) in the context of a conventional 6T-based array with differential read, but is also relevant for the proposed 8T low-swing design. SAE needs to turn on when the weakest cell in the array has overcome the offset of the sense amplifier. A simple delay chain of inverters poorly tracks the weakest cell over different voltages and process variation, so replica bitlines based on actual SRAM cells have been proposed to improve tracking [64]. The sense-amplifier enable signal can be generated from the statistics of signal arrival times, but at high area and energy cost [65]. The number of replica bitcells should be high to average out variation within the replicas themselves, but too many bitcells generates the SAE signal too quickly, so the delay can be increased with a multiplier circuit [66]. Suppressing the wordline voltage of conventional replica cells can emulate the threshold shift of the weakest cell in an array and improve tracking [67].

The proposed design uses replica bitcells with weakened supply voltages to generate the timing signal, as shown in Figure 4.16. A switched capacitor circuit accurately generates a degraded V_{GS} to emulate the weakest cell while tolerating large voltage noise. In order to generate the falling edge of the wordline, a replica bitline of 7 cells in parallel (to average together random variations) is supplied by a cell voltage that is set to be less than V_{DD} with the replica voltage generation circuit shown in Figure 4.17. In the first phase, the capacitor is connected between V_{DD} and 0.1V. In the 2nd phase, the top plate is disconnected from

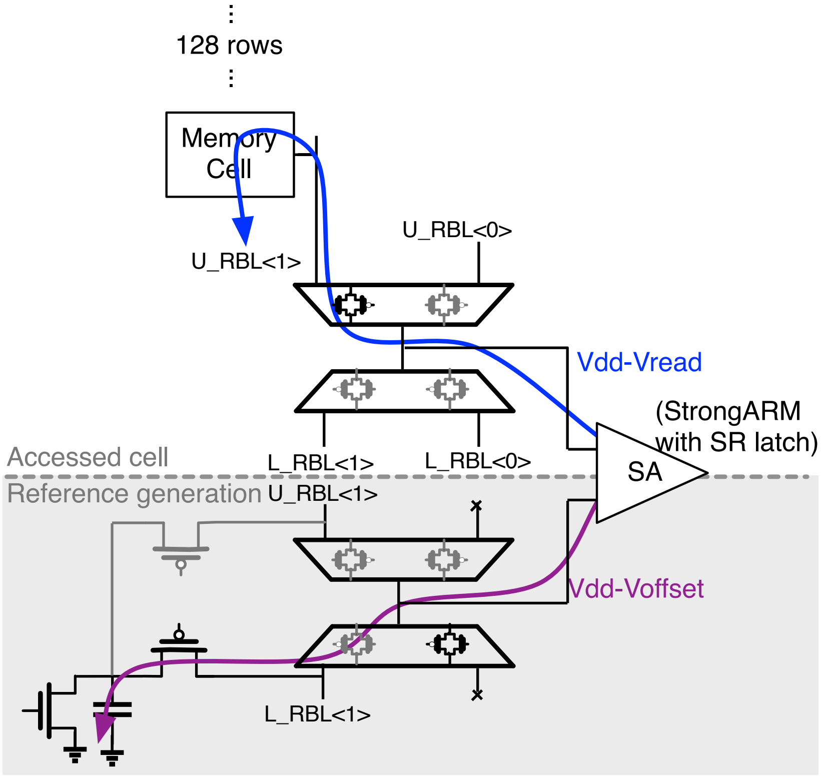


Figure 4.14: Proposed low-swing single-ended read scheme generates a reference using an unaccessed bitline.

V_{DD} and the bottom plate is pulled to 0, ideally setting the top to be V_{DD}-0.1. However, charge sharing diminishes the droop, so the reference needs to be calibrated appropriately. The 22fF capacitance is generated using a MOM cap, instead of a MOS cap, to maintain capacitance value with low voltages.

Within each column, the sense amplifier is active between the time in which the wordline turns off (to avoid excessive bitline discharge that doesn't provide further help to the read operation) and the precharge turns on. Figure 4.18 shows the circuits used to generate the sense amplifier enable signal. The SAE signal is not sent from the control, but rather derived locally (shared between 18 columns) from the wordline and precharge signal to ensure that there is no short circuit path between the sense-amplifier tail source and precharging the cell, and also to make sure the sense amplifier triggers as early as possible after the wordline turns off.

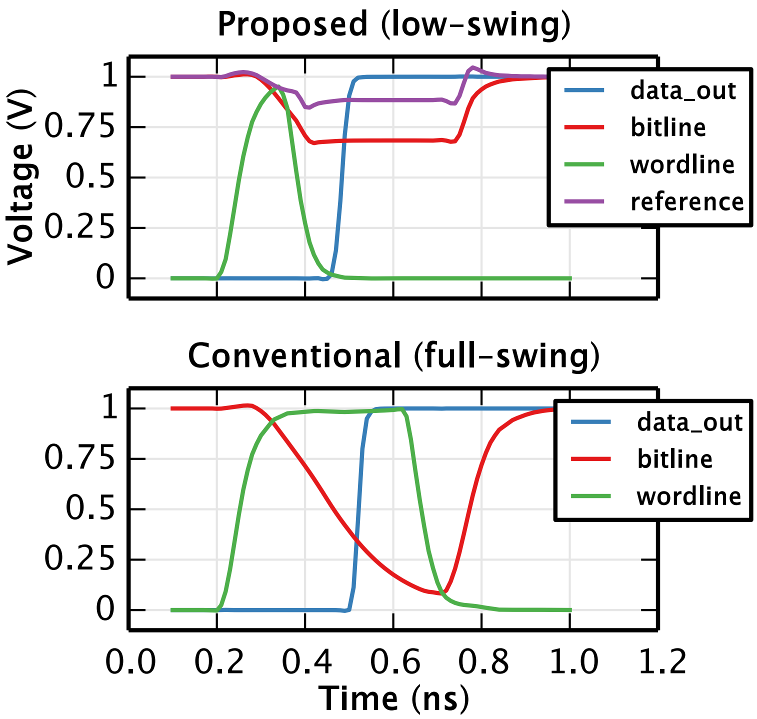


Figure 4.15: Simulation-based operational waveforms comparing the proposed low-swing scheme with the conventional full-swing scheme.

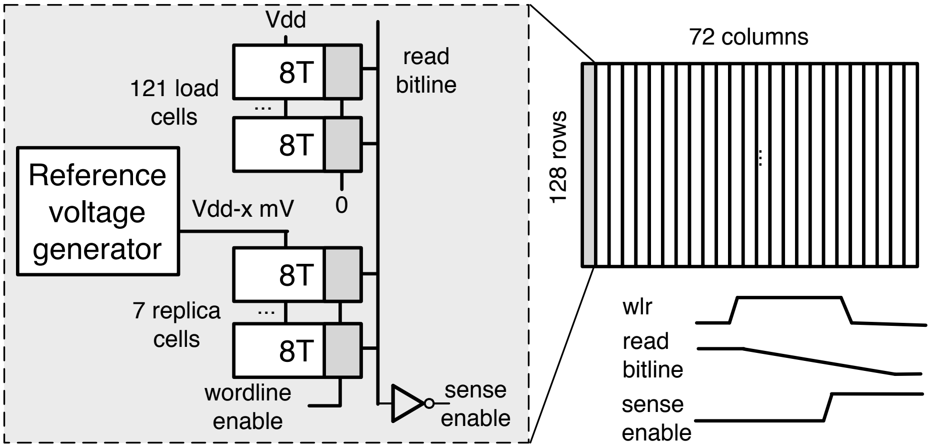


Figure 4.16: A replica bitline emulates the weakest cell in the array to turn off the wordline and turn on the sense amplifier at the optimal time.

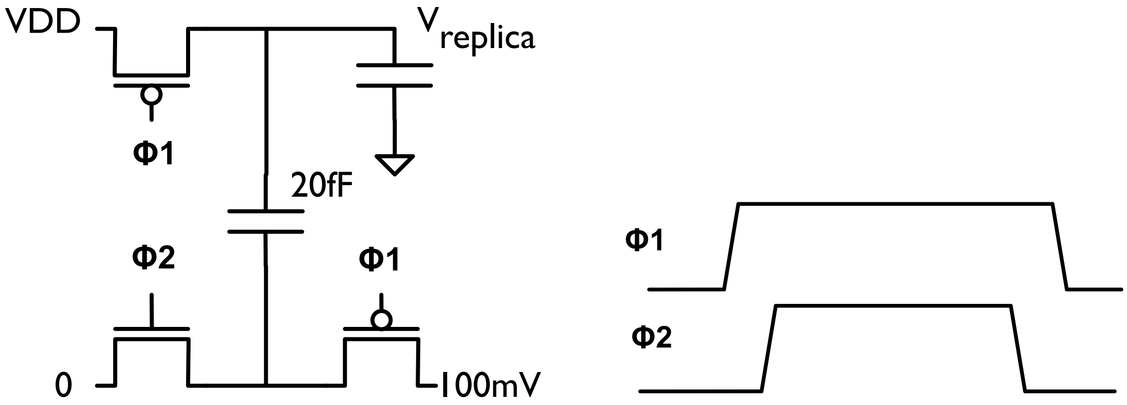


Figure 4.17: Replica reference voltage generator subtracts a constant V_{th} offset at the beginning of every cycle to respond to voltage noise.

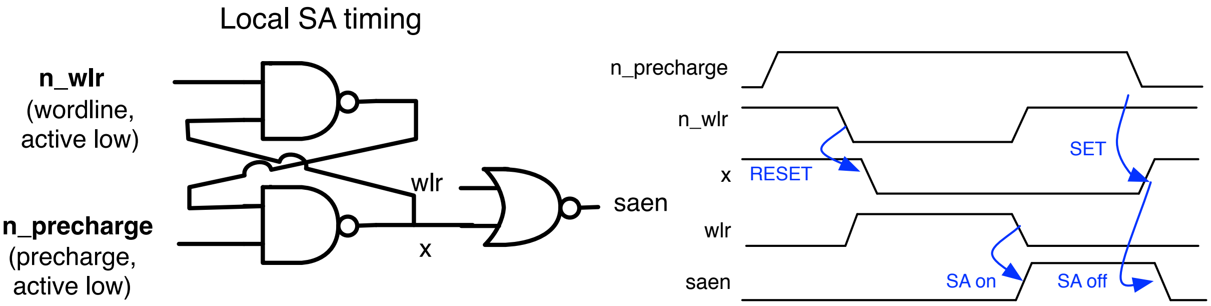


Figure 4.18: Local sense amplifier timing generation.

Backup read scheme

In order to compare the proposed single-ended read with the conventional case, a backup read scheme is implemented, as shown in Figure 4.19. The multiplexed read node is connected to a skewed inverter (700μm PMOS, 200μm NMOS), and followed by a latch, to directly read the single ended bitline.

4.3.4 Energy and Delay Simulation Results

Figure 4.20 shows energy per read operation for different voltages under the TT corner. At low voltages, energy savings disappear as the increasingly long delay required to read the weakest cell in the array causes the majority of the bitlines to discharge completely and approach the energy of the conventional full-swing bitline read. Energy savings are reported for reading 1's; reading 0's will favor full-swing reads because no bitline is discharged at all, and decrease the energy savings of the proposed scheme.

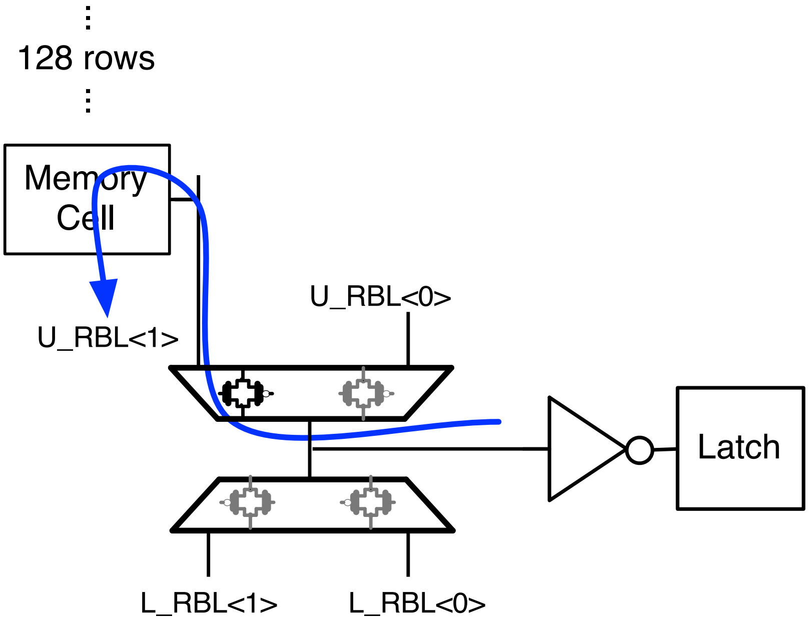


Figure 4.19: Backup read scheme

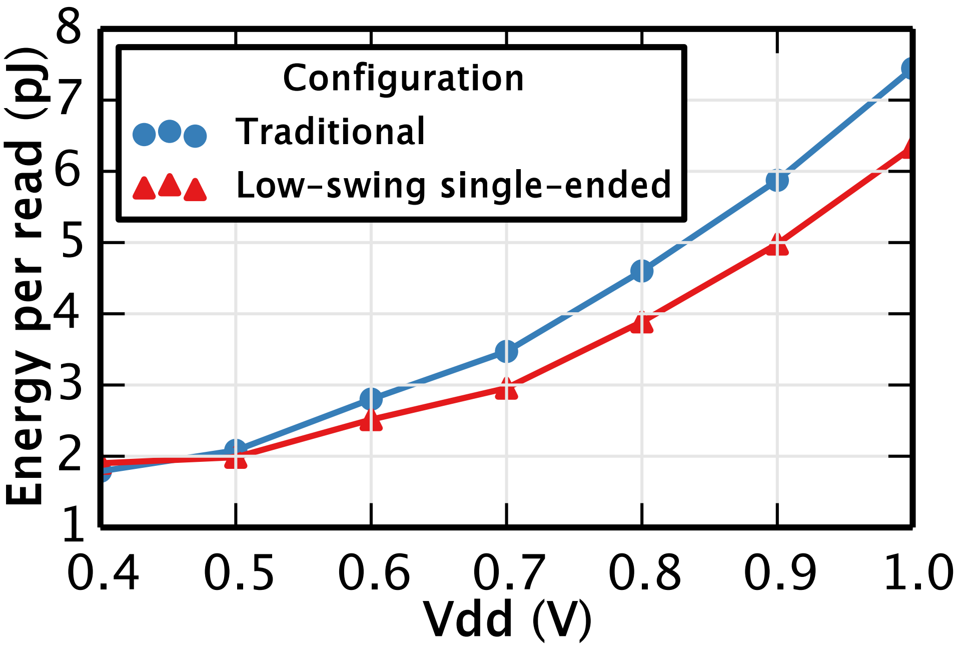


Figure 4.20: Energy comparison between a conventional full-swing bitline and the proposed low-swing scheme.

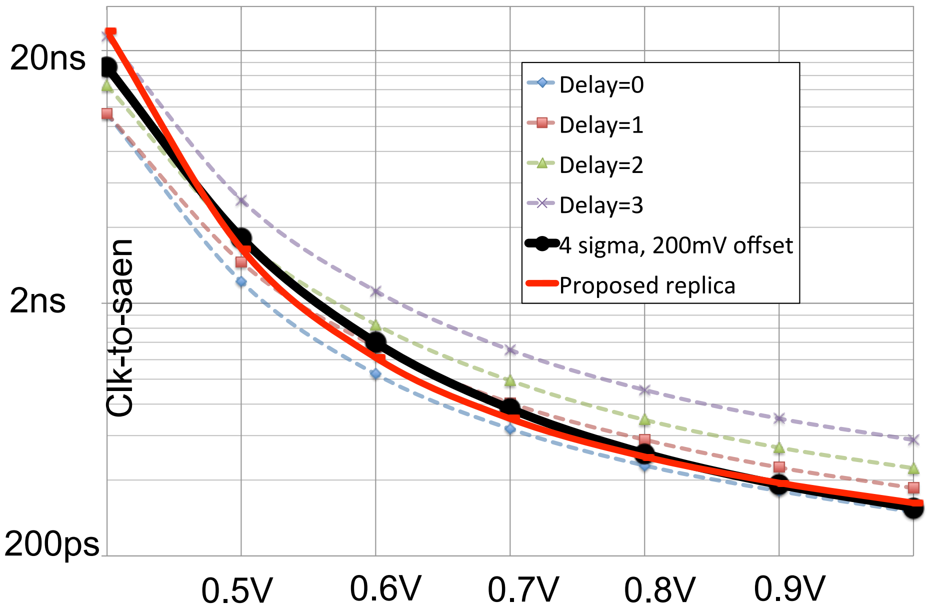


Figure 4.21: Replica timing scheme tracks the weakest cell in the array more closely than an inverter-based delay chain.

Figure 4.21 shows the optimal SAE delay versus SAE generation options over different voltages for a TT design. A simple delay chain does not slow down the SAE signal enough at low voltages, because the threshold voltage is different between the inverters in the delay chain and the bitcell. The proposed replica-timing approach closely tracks the ideal wordline pulse width over a wide range of voltages, enabling speed improvements through earlier sense amplifier triggering, and energy efficiency improvements by reducing the voltage swing on the bitlines.

4.4 Summary

Circuit-level resiliency techniques improve resiliency and lower V_{min} by preventing process variations from causing bitcell failures. Increasing the size of the bitcell, using body bias to improve NMOS-to-PMOS strength ratios, and using periphery assist techniques to under-drive or over-drive particular transistors can all be used to improve SRAM operation at low voltage. A single-p-well bitcell was described that uses body bias in FDSOI to trade-off discharge and completion failure modes during write operations. Also, an 8T-based SRAM design used in a low-voltage processor with a rippling supply voltage implements a negative bitline assist and replica read scheme to optimize the wordline pulse width at different voltages to increase speed and decrease energy consumption.

Chapter 5

Architecture-level Resiliency Techniques

In this chapter, two new resiliency techniques are proposed to tolerate bitcell failures at the architecture level. The generic error-modeling framework from Section 2.3 provides the means to evaluate a variety of exiting architecture-level resiliency techniques that tolerate bitcell failures. Using the intuition provided by this analysis, two new resiliency techniques are proposed to protect against hard faults. *Dynamic column redundancy (DCR)* with line disable can protect the data macros in L1 and L2 caches against high failure rates with low overhead, while *bit bypass (BB)* protects the tag macros.

5.1 Introduction

Architecture-level resiliency techniques assume that bitcells will fail, and use either error correction or redundancy to prevent bitcell failures from causing system failures. In general, circuit-level techniques move the entire failure distribution away from the failure region, while architecture-level techniques target the tail of the distribution—suggesting that architecture techniques can be more efficient and effective. Also, architecture-level techniques remain relevant for multiple process nodes, while circuit-level techniques need to be constantly redeveloped for new processes.

A large number of architecture-level techniques have been proposed to work around failing cells. One family of techniques uses redundancy to avoid failures. The Intel Pellston technique disables lines based on ECC feedback [68]. The non-disabled ways in the cache can be thought of as redundant lines. However, as error rates increase, disabling entire lines (typically 512 bits) to repair a single-bit failure causes the capacity to decrease dramatically. To increase the number of available bits in an SRAM at low voltages, functioning bits from multiple words can be merged together to form a single operational word [26], but this technique dramatically reduces cache capacity and increases switching activity. Exploiting the cache organization, words within a line can be disabled to trade off capacity for low-voltage operation with very little overhead [27]. Failing cells can also be substituted at the word level within caches, and extreme failure rates can be tolerated through aggressive offline graph algorithms to optimally match operational words [25], [69].

For all schemes with reconfiguration-based redundancy for hard faults, built-in-self-test (BIST) is required to identify error locations. Instead of reconfiguration with BIST, another family of techniques use error-correcting codes (ECC) to dynamically detect and repair errors at the cost of increased delay due to encoding, decoding, and correction [70]. At low voltage, data ways can be traded for checkbits to implement orthogonal Latin square codes (OLSC) that can correct many bit failures, but area and complexity of ECC is high [28]. ECC overhead can be reduced by using minimum-strength ECC codes for the number of faults in each word, using a stronger 4-bit correction code on the few ways that have more bit errors, but this scheme only makes sense for L2 caches where access to the ECC checkbits can be amortized over whole cache line accesses [29]. A combination of word disable to protect against multi-bit errors and ECC codes to protect against single-bit errors can significantly reduce V_{min}, but would still require a BIST to program the disabled words [71].

5.2 Protecting Data Arrays with DCR+LD

In general, previously proposed architecture-level techniques have targeted solutions that can tolerate extremely high failure rates where up to one out of 100 bitcells fail—requiring solutions that have considerable complexity and overhead. Holistic analysis in this section will show that less aggressive resiliency solutions that tolerate lower failure rates around 1×10^{-4} obtain practically the same energy efficiency gains with much less complexity and overhead. Additionally, ECC-based schemes require very complex codes to provide correction capability for both soft and hard faults. Overhead can be reduced further by using BIST characterization to identify fault location and reserving ECC for soft and intermittent faults only. The proposed DCR+LD technique stores fault location information in the tag array, and two separate mechanisms protect against single-bit and multiple-bit failures by using this fault location information to avoid failing bits.

5.2.1 DCR+LD Microarchitecture

Column-redundancy schemes handle hard faults by adding a spare column (SC) to an SRAM array, with a two-way multiplexer at the bottom of each column to shift columns over to map out a failing column. Traditional static column-redundancy schemes are configured with fuses and only correct one failing column per array. The proposed dynamic column-redundancy (DCR) scheme, shown in Figure 5.1, uses the same column multiplexer structure but associates a redundancy address (RA) with each row of the SRAM to dynamically select a different multiplexer shift position on each access, correcting one bit per row instead of one bit per array.

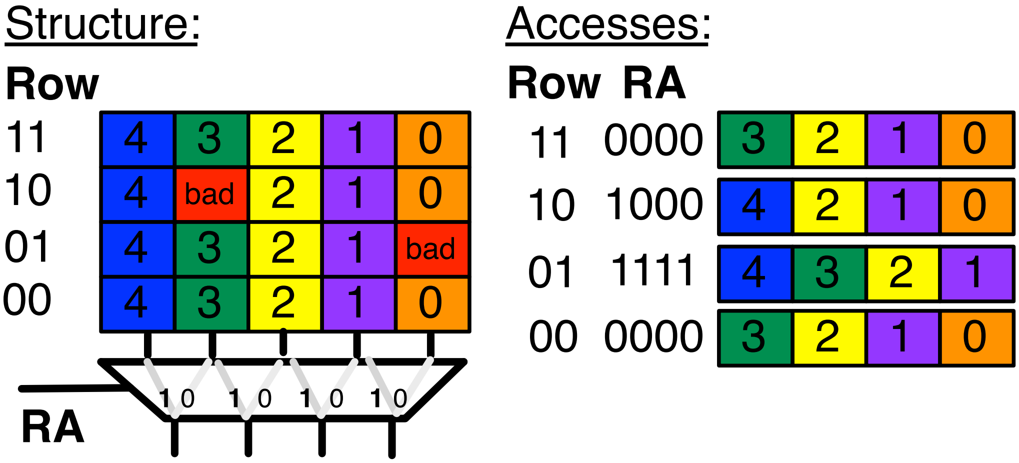


Figure 5.1: Overview of the proposed DCR scheme, which reprograms a multiplexer to avoid failing columns.

DCR offers the same capability of masking a single bit error per line as a SECDED code used to correct hard faults, but without requiring the latency and overhead of encoding or decoding logic. The extra bits required to store the RA per way are stored in the tag array, and require a similar area overhead as a per-cache-line SECDED code. The RA in DCR can be unique *Per-Set* (DCRPS) or *Per-Word* (DCRPW). For set-associative caches, the area overhead of DCRPW will be much larger than DCRPS, but more bits can be corrected.

A scheme similar to baseline DRC was proposed for a different application in resistive memories, where stored pointers indicate the location of failing bits [72]. But this scheme had the requirement of correcting multiple bits in a row, while the proposed scheme disables ways to handle multiple bits failing in a row. This DCR+LD scheme dramatically decreases the minimum operating voltage V_{min} beyond what is possible with only single-bit fault correction or only disabling lines.

In addition to hard faults, many large caches need to cope with soft and intermittent errors (Section 2.1.2) using some form of ECC. The proposed DCR+LD scheme is particularly attractive because it can be used in conjunction with a simple SECDED codes for soft errors, while DCR and LD repair hard faults to reduce V_{min}. Note that if ECC is used to correct hard faults, an expensive DECTED scheme is needed to also survive a soft error as well as a hard fault.

Like other redundancy schemes, DCR+LD requires BIST to locate failed cells and program the RA values. This can either be performed offline and stored in non-volatile memory to be reloaded on boot, or the BIST can be rerun during system reboot or as needed when aging errors are detected.

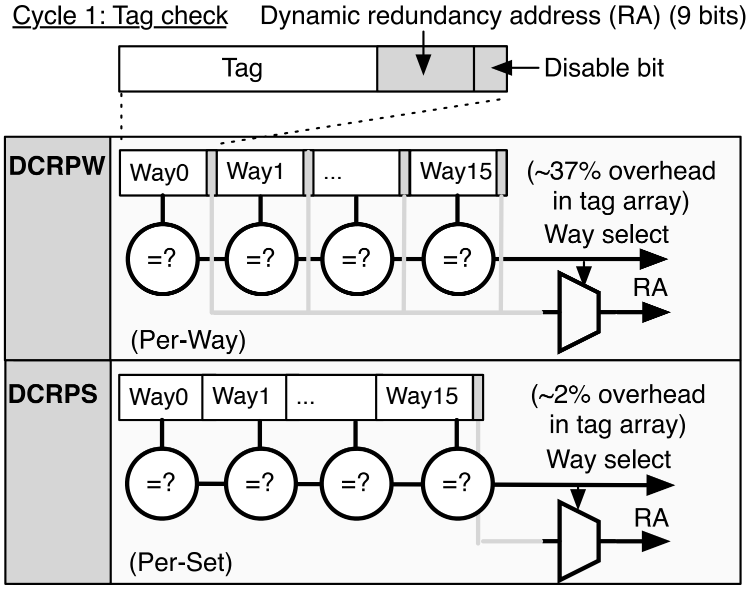


Figure 5.2: An extra redundancy address is added to each way in the tag array for DCRPW, or to each set for DCRPS.

DCR Implementation for L2 Caches

Figure 5.2 illustrates modifications to the tag array for a 16-way set-associative L2 cache for both the DCRPW and DCRPS scheme. For DCRPW, each entry in the tag array now includes an extra RA as well as a disable bit. These extra 10 bits add a 37% penalty on tag array size assuming 27-bit tags and a 16-way cache. Although 37% overhead may appear large, the 10 extra bits per line in the tag is similar to the 11 extra bits that would be required in the data array for SECDED used to catch hard faults on 512-bit lines, although redundancy address cells would need to be slightly larger for resiliency. Another variant of the proposed scheme, DCRPS+LD, trades off slightly diminished correction capability for substantial area reduction by storing a single RA per set instead of per way.

After a cycle for L2 tag access, the correct way is identified and RA for this way is retrieved. During the subsequent data access, the RA is used to avoid the failing bit. Figure 5.3 describes how shift redundancy is implemented for L2 caches. The SRAM macros have 64 bits of IO arranged physically as 256 columns with four-to-one interleaving [73], and 8 macros are accessed in parallel to read or write an entire 64-byte cache line. One of these 8 macros has a redundant IO column (four physical bit columns) to protect against sense-amplifier failure as well as bit failure. Therefore the main overhead of this scheme is in the tag arrays, not the data arrays, as this redundant IO adds only 0.2% overhead to the data while adding 2-37% overhead to the tags. The shift multiplexer for reads and demultiplexer for writes can be implemented with pass transistors and therefore adds very little extra latency to the critical path. Computing the thermometer code for the shift multiplexers can be done dur-

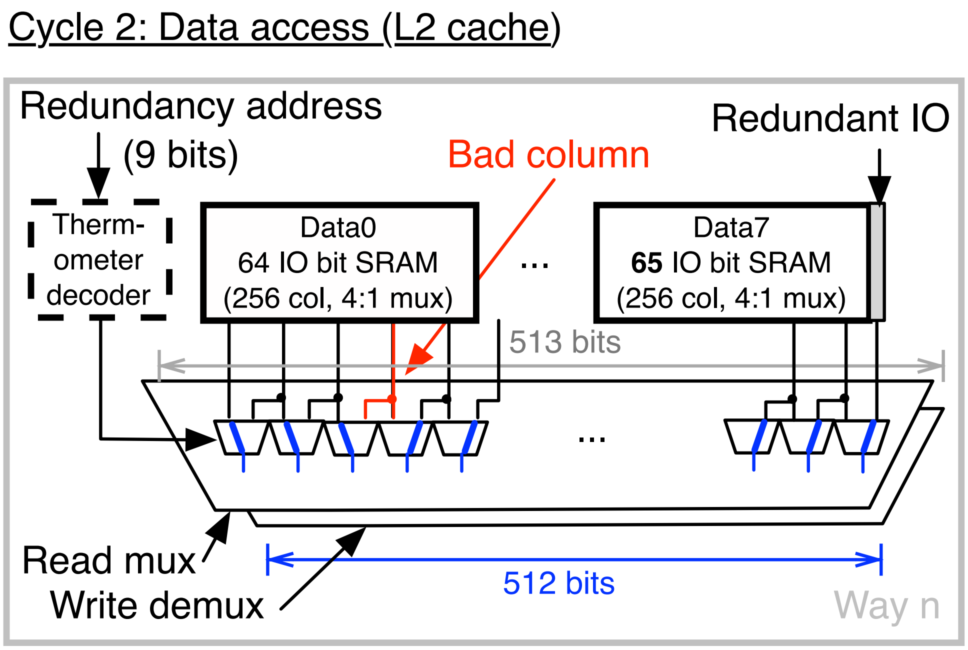


Figure 5.3: For L2 caches, the proposed scheme uses the RA to multiplex/demultiplex around failing columns.

ing the SRAM access and does not affect latency. Synthesis results summarized in Table 5.1 show that the area and energy of the added decoder and multiplexers is much lower than a SECDED decoder.

DCR Implementation for L1 Caches

Many existing schemes can only be used in large L2 or L3 caches where the increased latency has a smaller effect and operations always occur on entire lines instead of words. DCR+LD can be modified to perform well on small granularity and low-latency accesses in L1 caches.

Table 5.1: Delay and area comparison between SECDED and DCR from synthesized implementations in 28nm.

	Write (encode)		Read (detect/correct)	
Scheme	Area	Delay	Area	Delay
SECDED	290 μm^2	190ps	610 μm^2	280/400 ps
DCR (proposed)	150 μm^2	20ps †	150 μm^2	20ps †

† Thermometer decoder takes 90ps, but operates in parallel with array sensing/latching

L1 caches perform accesses in a single cycle on a single word (64 bits) instead of an entire line. In order to avoid read-modify-write of an entire line during writes to a single word, ECC-based schemes on L1 caches need to perform ECC on words instead of lines (eg. checkbits for every 64 bits instead of 512 bits), increasing the checkbit overhead from 2% to 12.5% for the SECDED case. Even more checkbits would be required for a DECTED code that can protect against both hard and soft faults. The main advantage of the DCR scheme for L1 caches is that it is still possible to use a single RA per line, instead of requiring a redundancy address per word, which dramatically reduces the area overhead of the DCR scheme compared to ECC schemes for the L1 cache.

Figure 5.4 illustrates the dynamic redundancy scheme for L1 caches, which can still repair one bit in every 512-bit line. In the L1 data arrays, ways are often physically interleaved, so accessing a 64-bit word would access all four ways in parallel, then multiplex the correct way at the end of the cycle after tag comparison is complete. To add DCR to L1 data caches, each SRAM macro is extended from 64 input-output columns to 65 input-output columns and a shift multiplexer and demultiplexer are added to avoid bad columns. The RA is shared by all 8 words in a cache line, yet all 8 arrays have a redundant bit, so 7 of the redundant bits will not be used on each access. However, the only alternative is to store the redundant bit as one flip-flop per set, which actually would cause a larger overhead than extra columns. For the L1 cache, the thermometer decoder latency must be less than the sense amplifier and latch latency to avoid extending the critical path.

5.2.2 Evaluation

In this section, a variety of proposed architecture-level resiliency techniques are evaluated with the proposed generic framework (Section 2.3) using the same bitcell failure probability, target cache size, and soft-error assumptions. The target cache designs are a 4-way 32 KB L1-style cache and a 16-way 2 MB L2-style cache with 64-bit words and 64-byte lines. This analysis focuses only on data portions of caches. Tags need to be correct, but because they are much smaller than the data portion of caches they can be protected with circuit techniques, or a separate redundancy technique described in Section 5.3.

All schemes are required to maintain at least 99% capacity. This requirement keeps the results generic and independent of activity or architecture-level assumptions. Therefore no architecture-level simulation-based studies are required to compare energy and execution time of the different schemes. Section 5.2.4 discusses why ignoring operating points at lower capacities is appropriate. This evaluation assumes that a single bit of soft-error correction is required. For schemes that do not protect against soft errors, one bit of ECC is added to ensure a fair comparison.

The evaluation was performed using the architecture-level error model described in Section 2.3. Table 5.2 shows the parameters used in the generic framework to evaluate each technique. Every technique can be compared with the single framework by just changing a few parameters, enabling easy design-space exploration of a wide range of input parameters. The DEC, DEC+LD [71], and VS-ECC [29] schemes rely on BCH codes, and therefore are

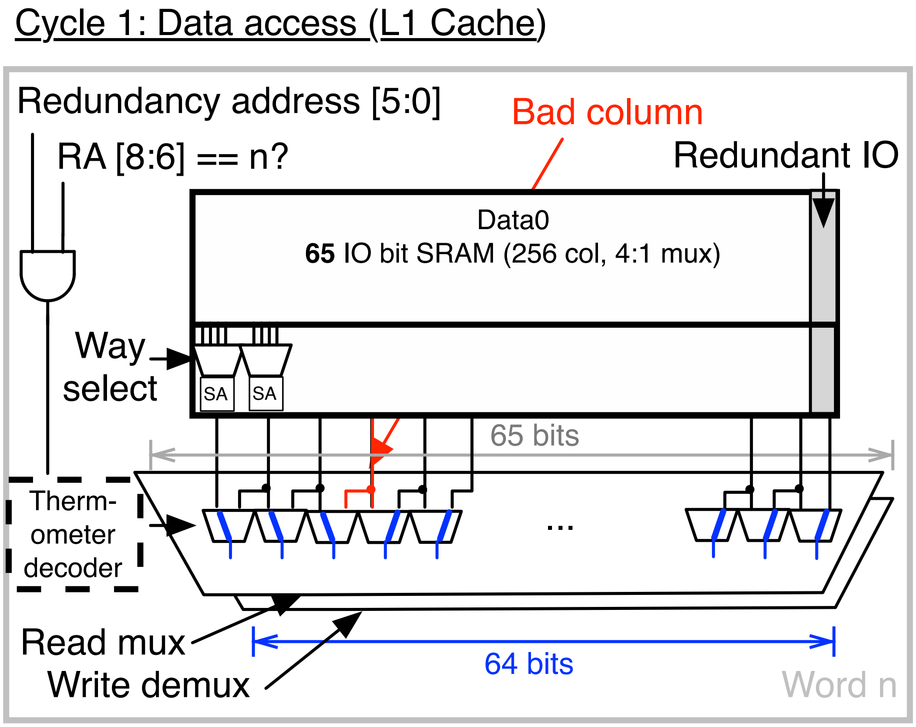


Figure 5.4: For L1 caches, the proposed scheme uses a redundant column per array.

not included in the L1 comparison because small access granularities increase checkbit overhead from 4% to 22% [30], and high encoding and decoding latency makes them ill-suited for L1 caches. The Mixed-cell [39] scheme is excluded from L1 comparison because it would require double the dynamic energy when readings tags and data in parallel.

Table 5.3 includes all of the equations used to calculate capacity, which is important because capacity of a cache at a given voltage can also determine V_{min} due to the 99% capacity requirement. The sections below explain how Tables 5.2 and 5.3 were derived. Each evaluation is validated with a numerical simulation of each technique using fault maps. The results of the evaluation using the proposed generic framework are exact and no approximations are made.

Line Disable (Pellston) [68]

Disabling failing ways in a set-associative cache is a simple and effective way to improve resiliency. V_{min} is limited by requiring at least one working way ($a_{l-s} = n_{l-s} - 1$) and requiring 99% capacity, because every bit failure requires a line to be disabled. This technique has no latency overhead and a negligible tag-array overhead due to the disable bit, so is applicable for both L1 and L2 caches. Line disable would need to be supplemented with a SECDED code to provide soft-error resiliency.

DECTED (DEC)

SECDED codes such as the Hsiao code allow correction of a single bit in a word [74] at the cost of extra checkbits, encoding and decoding. However, in order to protect against soft and intermittent errors, two-bit correction is required (1 bit for hard fault and 1 bit for soft errors), which requires a substantially higher overhead DECTED code such as the BCH code [30]. After margining for soft errors, one bit is allowed to fail in every word ($a_{b-w} = 1$). V_{min} is limited by multiple-bit failures at voltages where two or more hard faults per word become common.

DECTED + line disable (DEC+LD) [71]

For line disable alone, voltage scaling is limited by single-bit hard faults diminishing capacity, while for DECTED alone, voltage scaling is limited by multiple-bit faults beyond the capabilities of ECC. Combining these two techniques enables substantially more voltage scaling. Capacity is determined by the probability that a line has two or more errors.

Macho [25]

Macho sacrifices cache lines to provide additional redundancy for other lines within the same set (or group of a couple of sets) at a word granularity. To model this with the generic model, n_{b-w} is treated as the number of bits in a word as usual, n_{w-l} represents the number of lines in a set and n_{l-s} represents the number of words in a line (n_{w-l} and n_{l-s} are switched). Therefore V_{min} will be limited by needing one word to work at every word offset, and no other errors are allowed. Capacity is determined by calculating the expected value of the maximum of n_{l-s} samples from X_{line} using order statistics [75], which corresponds to the number of working words in the worst word offset location. This technique requires complex

Table 5.2: Inputs to evaluate the minimum operating voltage (V_{min}) using the proposed generic model in Equation 2.8 for different architecture-level resiliency techniques.

	Resiliency Mechanism		Inputs to Equations 2.1-2.8											
			Scheme parameters				L1 cache (32KB)				L2 cache (2MB)			
Technique	Hard	Soft	a_{b-w}	a_{w-l}	a_{l-s}	a_{s-c}	n_{b-w}	n_{w-l}	n_{l-s}	n_{s-c}	n_{b-w}	n_{w-l}	n_{l-s}	n_{s-c}
Nominal	-	SEC	0	0	0	0	64	8	4	128	512	1	16	2048
Pellston [68]	LD	SEC	0	0	$n_{l-s} - 1$	0	64	8	4	128	512	1	16	2048
DEC	DEC		1	0	0	0	-	-	-	-	512	1	16	2048
DEC+LD [71]	LD+DEC		1	0	$n_{l-s} - 1$	0	-	-	-	-	512	1	16	2048
Macho [25]	Redund.	SEC	0	$n_{w-l} - 1$	0	0	-	-	-	-	32	16	16	2048
Mixed-cell [39] ◇	Upsize	SEC	0	0	0	0	64	8	1	128	512	1	4	2048
VS-ECC [29]	4EC	SEC	2	0	$\frac{n_{l-s}}{4} - 1$	0	-	-	-	-	512	1	16	2048
DCRPS+LD (prop.)	DCRPS+LD	SEC	0	0	$n_{l-s} - 1$	0	512	1	4	128	512	1	16	2048
DCRPW+LD (prop.)	DCRPW+LD	SEC	1	0	$n_{l-s} - 1$	0	512	1	4	128	512	1	16	2048

◇ Analyze robust portion (1/4 array) only, 65mV lower V_{min} for L2 bitcell, 130mV lower V_{min} for L1 bitcell (larger cells) offline graph coloring algorithms to map lines together in order to maximize capacity. The main overhead required is the storage of the fault map. Macho would need to be supplemented with a SECDED code to provide soft-error resiliency.

VS-ECC [29]

The VS-ECC-Disable scheme uses SECDED on 3/4 of cache ways and 4EC5ED on 1/4 of cache ways, which can tolerate both soft errors and hard faults. Online testing disables ways with 3 to 5 errors, assigns 4EC5ED to up to 1/4 of ways with between 1 and 2 errors, and assigns SECDED to ways with 0 errors (so that soft errors can be tolerated). The reported results match the case where voltage can be reduced until a maximum of one quarter of the ways ($a_{l-s} = \frac{n_{l-s}}{4} - 1$) are disabled due to three or more errors ($a_{b-w} = 2$). To compute capacity, this is one of the few schemes that requires replacing one level in the binomial hierarchy with a multinomial distribution. Capacity is determined by categorizing lines as either having 0 failures, 1 or 2 failures (use 4EC), or greater than 2 failures (disable), computing the probability of all possible permutations of line categories within a set, and multiplying the probability by the number of resulting error-free or corrected lines. For example in a 4-way cache, one term of this sum represents the probability that 2 lines have 0 failures, 1 line has 2 failure, and 1 line has 3 or more failures.

Table 5.3: Calculation of cache capacity for each scheme.

Technique	Capacity
Nominal	100%
Pellston [68]	$(1 - p_{line\_fails}) \cdot 100\%$
DEC	100%
DEC + LD [71]	$(1 - p_{line\_fails}) \cdot 100\%$
Macho [25]	$\frac{n_{w-l}-z}{n_{w-l}} \cdot 100\%$ $z = \sum_{x=0}^{n_{w-l}-1} [1 - [\mathbb{P}(X_{line} \leq x)]^{n_{l-s}}]$
Mixed-cell [39]	$(0.25 + 0.75 \cdot \mathbb{P}(X_{word} = 0)) \cdot 100\%$
VS-ECC [29]	$z \cdot 100\%$ $X_{set} = Multinom(\mathbb{P}(X_{line}=0), \mathbb{P}(X_{line}=1) + \mathbb{P}(X_{line}=2), \mathbb{P}(X_{line}>2)))$ $z = \sum_{x=0}^{n_{l-s}} \sum_{y=0}^{n_{l-s}-x} \mathbb{P}(X_{set}=(x, y, n_{ls}-x-y)) \cdot \frac{x+\min(\frac{n_{ls}}{4},y)}{n_{l-s}}$
DCRPS+LD (prop.)	$z \cdot 100\%$ $X_{set} = Multinom(\mathbb{P}(X_{line}=0), \mathbb{P}(X_{line}=1), \mathbb{P}(X_{line}>1)))$ $z = \sum_{x=0}^{n_{l-s}} \sum_{y=0}^{n_{l-s}-x} \mathbb{P}(X_{set}=(x, y, n_{ls}-x-y)) \cdot \frac{x+\min(1,y)}{n_{l-s}}$
DCRPW+LD (prop.)	$(1 - p_{line\_fails}) \cdot 100\%$

Mixed-cell [39]

The mixed-cell scheme revises the VS-ECC scheme by using larger cells instead of extra-ECC on the robust portion of the cache. V_{min} is limited by the robust portion of the array, because in the worst case, every non-robust cell would be disabled, and therefore errors of non-robust cells would not affect V_{min}. Analysis of the robust portion is achieved by reducing the analyzed cache size by 75% (n_{l-s} decrease), and shifting the p_{bit} vs. V_{DD} curve. Analysis of capacity depends on the non-robust cells and is performed on normal cells. The main problem is that this scheme incurs an overhead at all voltages because the resulting cache is 25% smaller in all operating modes and robust cells use more energy. Mixed-cell would need to be supplemented with a SECDED code to provide soft-error resiliency. The V_{min} results from [39] assume SECDEC can be used for hard faults, and compare to a baseline that already has 2× sized transistors, and therefore has a lower estimated V_{min}.

Dynamic Column Redundancy Per-Way with Line Disable (DCRPW+LD)

DCRPW+LD allows one bit to fail in each word ($a_{b-w} = 1$) and all but one way to fail in a set by disabling words with multi-bit failures ($a_{l-s} = n_{l-s} - 1$). Capacity is determined by the remaining number of working or repairable words. ECC protection is necessary to protect against soft errors, and so a SECDED code is implemented for both the L1 and L2 cache.

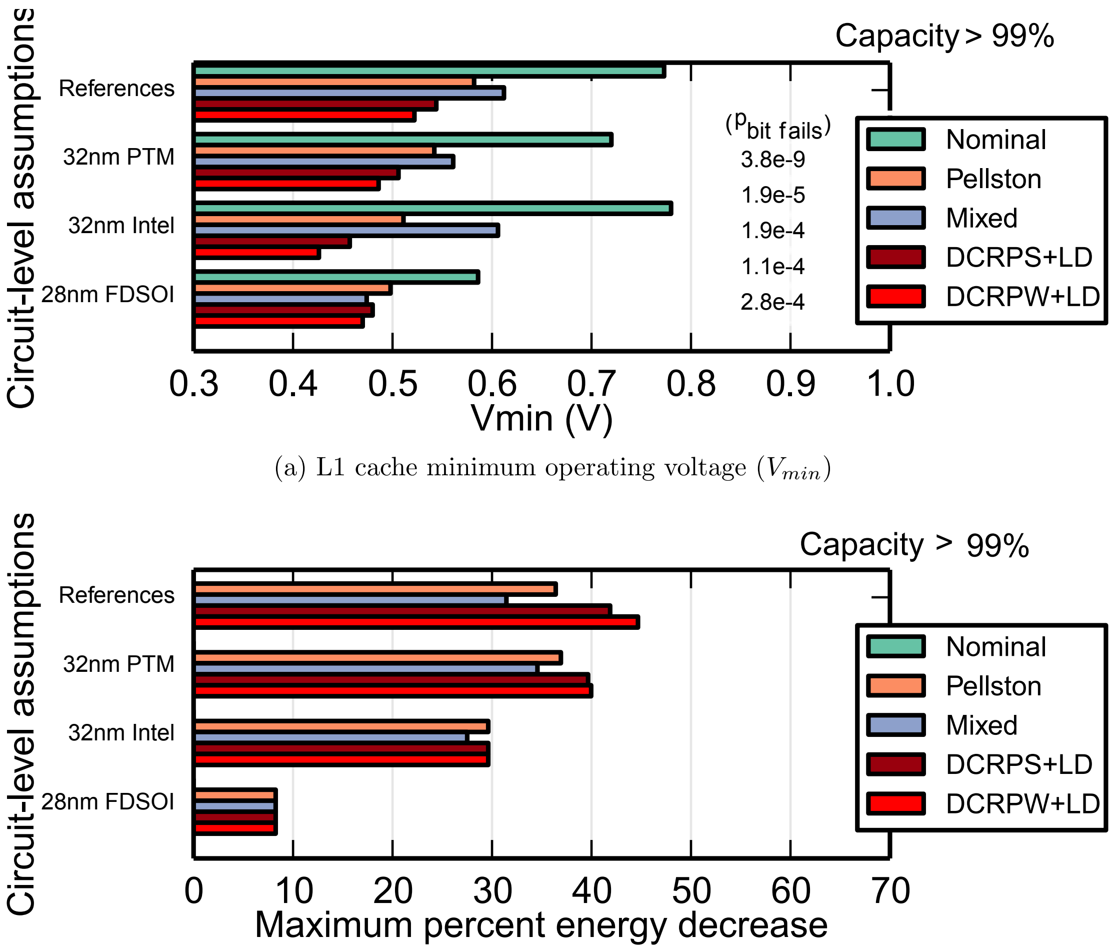
Dynamic Column Redundancy Per-Set with Line Disable (DCRPS+LD)

DCRPS+LD only repairs one bit per set instead of per way, which greatly reduces the checkbit overhead at the cost of a reduced repair capability. Like VS-ECC, because there are two components of the scheme in use at once, a multinomial distribution is used to calculate the capacity in the same manner as before. SECDED is added to protect against soft errors.

5.2.3 Results

By setting the probability of cache yield failure equal to 1×10^{-3} (99.9% yield) Figures 5.5a and 5.6a plot the minimum operating voltage for each technique as a function of each dataset for the L1 and L2 cache, and annotates the maximum p_{bit}, which is independent of assumptions about bitcell voltage scaling. For L1 caches, the proposed DRC schemes achieve the lowest possible V_{min}. For L2 caches, DCRPW+LD achieves a V_{min} only 10–25 mV higher (depending on the dataset) than the lowest V_{min} achieved by the VS-ECC scheme, and DCRPS+LD achieves a V_{min} only 20–70 mV higher.

Figures 5.5b and 5.6b plot the maximum energy savings provided by each scheme for the energy scaling assumptions in Figure 2.14 and show that for all but one dataset, the difference in V_{min} between the proposed DCR schemes and VS-ECC results in no difference in E_{min}. Therefore the proposed scheme achieves comparable energy-efficiency improvements with much lower complexity and overhead than previously proposed schemes.



(a) L1 cache minimum operating voltage (V_{min})

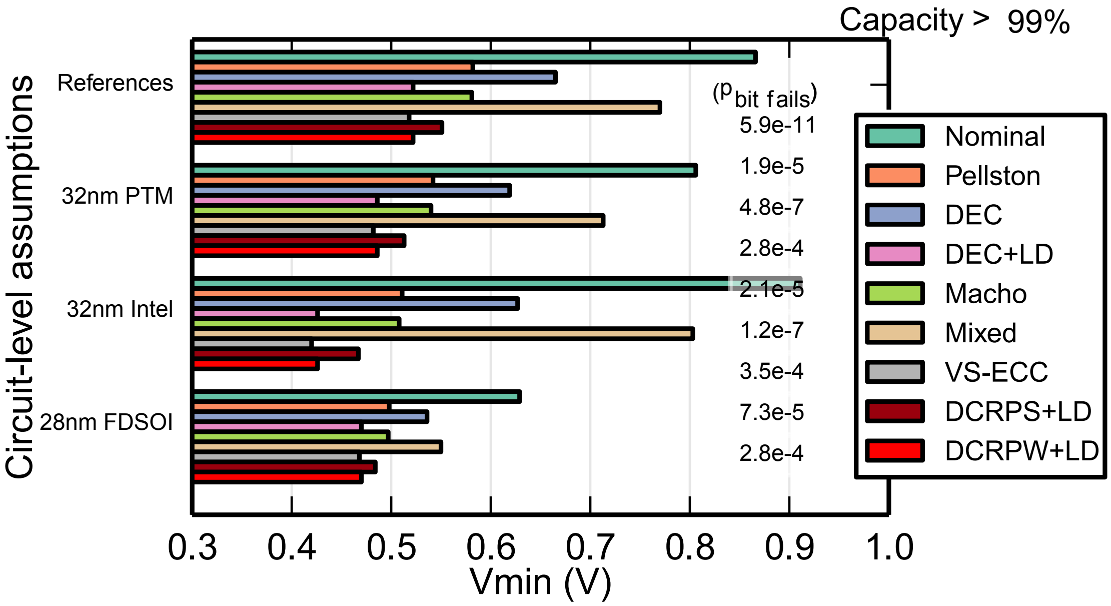
(b) L1 cache maximum energy reduction (E_{min})

Figure 5.5: Evaluation of the lower bound on voltage and energy reduction for L1 caches.

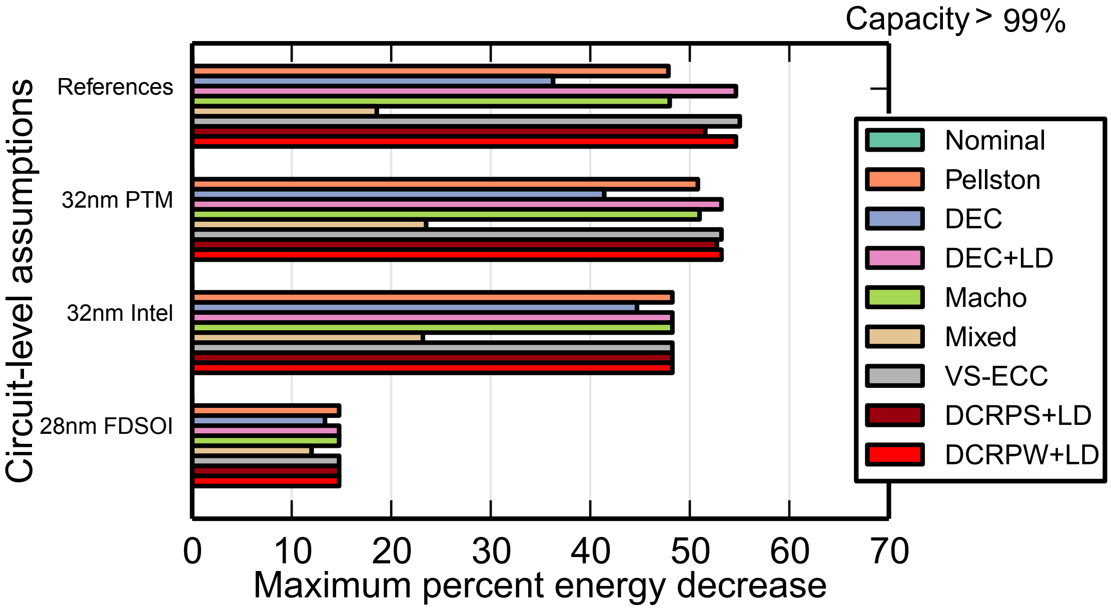
If less than 99% capacity is allowed, Macho can achieve a slightly lower V_{min}, as shown in Figure 5.7. However, because the p_{bit} increases so dramatically, only approximately 50mV of further scaling is possible in exchange for a cache capacity decrease of 40%. Operating in this low-capacity region is avoided in this analysis, because energy-scaling benefits have already saturated and cycles-per-instruction (CPI) will increase.

5.2.4 Discussion

Beyond showing the usefulness of the proposed generic model and comparing the effectiveness of many different published schemes, the evaluation results can be extended to propose a series of design guidelines that explain why certain ideas work better than others and predict how resilient design techniques will need to change when technology and usage assumptions change.



(a) L2 cache minimum operating voltage (V_{min})



(b) L2 cache maximum energy reduction (E_{min})

Figure 5.6: Evaluation of the lower bound on voltage and energy reduction for L2 caches.

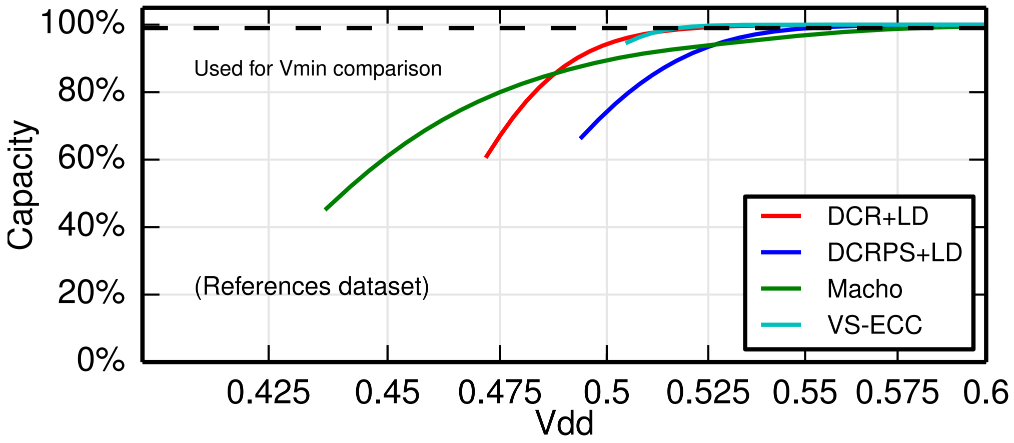


Figure 5.7: Further V_{min} reduction ($<$ 50mV) is possible with capacity reduction.

Guideline #1: The possibility of undetectable errors limits voltage scaling

An ECC scheme such as DECTED can be dramatically improved by adding line disable capability (DECTED+LD), as shown in Figure 5.6a. At V_{min} of DECTED for the L2 cache (where one bit of correction is saved for soft errors), the probability that DECTED is needed to correct one failing bit in a word is only 0.024%. V_{min} is limited because the probability that a word has two or more errors (3.0×10^{-8}), and with 32,767 other words in the cache, 0.1% of caches will have a word with two or more errors (99.9% yield specification). However, by disabling lines in the cache, V_{min} is dramatically reduced, and now DECTED replaces a failing bit in 12.4% of cache lines, and only 1% of the lines are even disabled. Therefore the key to resiliency schemes is to make the common case (no failures) fast while inexpensively preventing multiple failing bit words from harming system operation.

Guideline #2: Each process requires different resiliency strength to achieve E_{min}

Figure 2.14 annotates the probability that a bit fails at the minimum energy point. For all four datasets, p_{bit} at the minimum energy point ranges from between approximately 1×10^{-2} and 1×10^{-6}. E_{min} depends both on assumptions about bitcell failures versus voltage as shown in Figure 2.13b, as well as energy calculations that account for increased leakage at low voltages. In some cases E_{min} isn't the most appropriate metric, because a slightly higher voltage and energy could dramatically increase performance. Figure 5.8 shows the energy versus delay for all four datasets, with annotations marking the minimum E^2D (energy squared times delay) product. To achieve this metric, a p_{bit} of 1×10^{-7} is required, so even weak resiliency schemes would work. Focusing on V_{min} reduction alone encourages complex schemes that have more resiliency than needed in system designs.

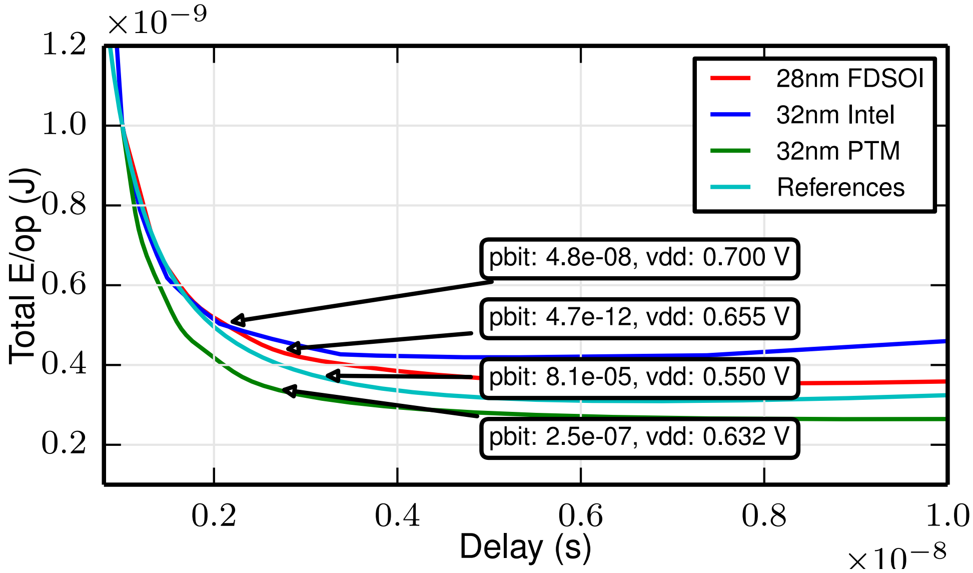


Figure 5.8: Energy per operation versus delay with annotated points for minimum E^{2}D.

Guideline #3: Benefits of voltage scaling saturate at p_{bit}=1×10^{-3} to p_{bit}=1×10^{-4}

Figure 5.9 plots the probability that a word fails versus the probability that a bitcell fails for both the L1 and L2 cache. This analysis is completely decoupled from voltage and energy. One order of magnitude difference in p_{bit} around 1×10^{-3} causes a cache to go from almost completely working to almost completely failing. To put this slope in perspective, the figure annotates the voltage change required to go from 4×10^{-4} to 4×10^{-3} for different datasets. A 14 mV to 72 mV change in voltage will cause cache capacity to decrease dramatically from 90% to 20%. Additionally, energy at these voltages will likely be dominated by leakage, which means that this voltage decrease will translate into a very small decrease in energy and large increase in delay. Therefore, schemes that attempt to handle high failure rates not only add substantial complexity, but also trade a dramatic reduction in capacity for a very small decrease in V_{min} and potentially no additional energy savings.

Guideline #4: Energy overheads at all voltages matter

Designs that incur energy overheads at all voltages could harm overall energy efficiency even if they do enable a significant improvement of V_{min}. For example, a hypothetical scheme that trades a 10% increase of energy at V_{DD} of 1 V (for example, using slightly larger cells) to decrease V_{min} by 100 mV will be offset by decreasing the supply voltage from 600 mV to 500 mV (when optimistically assuming energy scales with CV^2). Unless more energy is expended in low-power mode than in high-performance mode, this hypothetical technique would actually harm energy efficiency. Additionally, techniques that require cache flushes when transitioning between modes incur an energy cost to switch modes, which can diminish

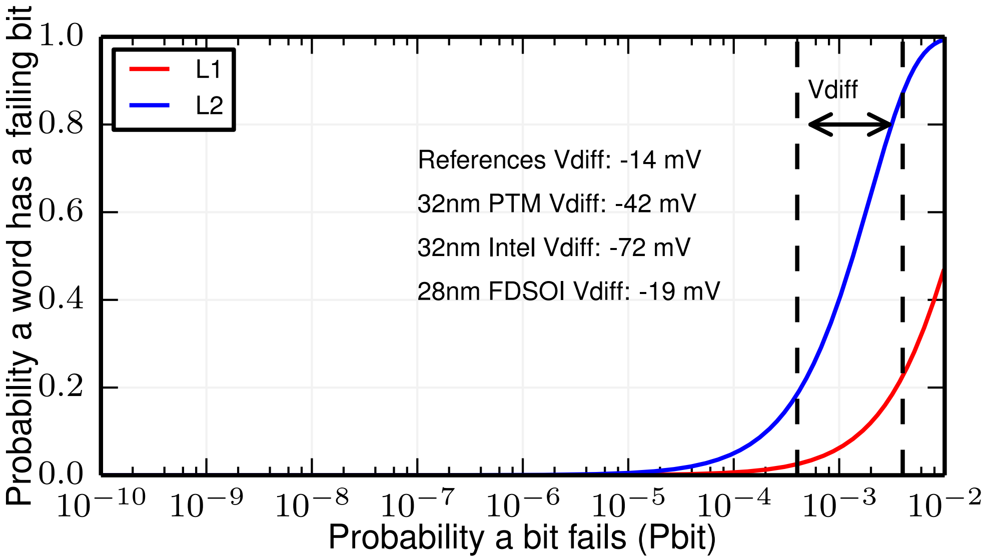


Figure 5.9: Probability that a 64 bit (L1) and 512 bit (L2) word has failing bits versus bitcell failure probability.

energy-efficiency savings.

5.3 Protecting Tag Arrays

The proposed DCR+LD technique only works for the data arrays inside caches, because the redundancy information stored in the tags is guaranteed to be available prior to data array access. For the tag array, or other stand-alone SRAM macros in the design, a different technique is required to avoid failing bitcells.

Circuit-level techniques, such as using a larger cell size or implementing assist, could improve the redundancy of the tags. However, predicting the effectiveness such that it matches the resiliency provided by an architecture-level technique is difficult. Also, an architecture-level technique to protect the tags would allow the simultaneous usage of both a circuit and architecture technique to further reduce V_{min}. The proposed architecture-level technique, named bit bypass, stores the row and column address of failing bitcells in flip-flops to bypass failing bits.

5.3.1 Bit Bypass (BB)

Figure 5.10 explains the general principle of bit bypass. Each row can have a redundant set, and within the row, each failing bitcell can have a redundant entry. The number of entries within each set is determined by the target failure rate, through evaluation of the architecture-level error model with a_{bw} set to the number of entries in each set with all other

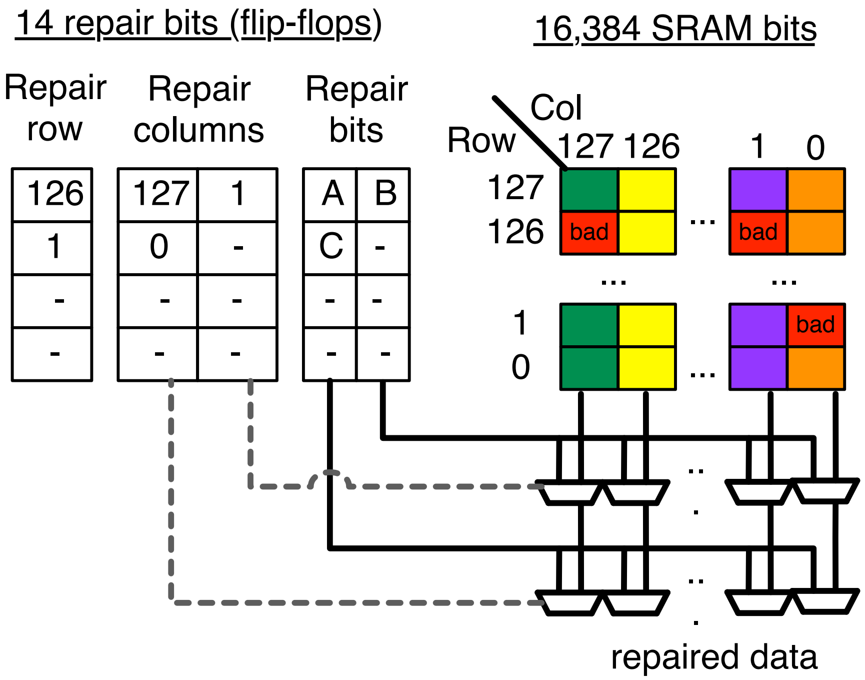


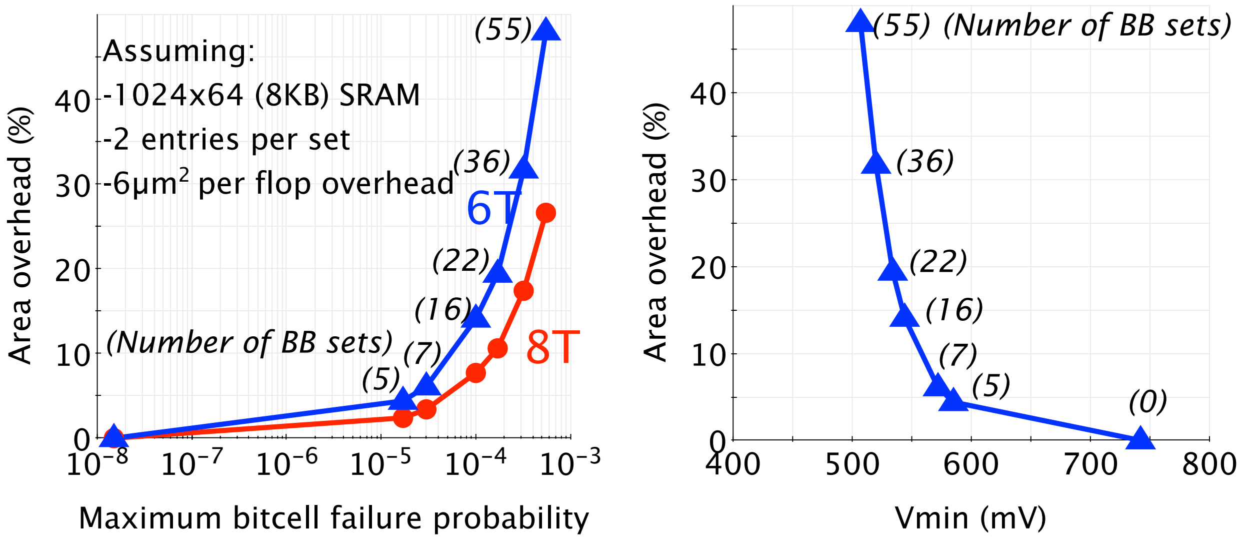
Figure 5.10: Overview of the proposed bit bypass scheme.

parameters set to 0. For a failure rate of 10^{-4}, two entries per set would be sufficient, but as target failure rate is increased, more entries will be needed to tolerate multi-bit errors. Adding extra entries increases the area linearly, and adds an extra multiplexer to the critical path.

Figure 5.11 plots area overhead versus maximum probability of failure and approximate V_{min}. For a typical SRAM macro size (1024 entries of 64 bits for a total 8KB macro), each entry requires 10 flip-flops for the row address, 6 flip-flops for each of the two column addresses, plus 2 flip-flops for valid bits. In addition to flip-flops, the combinational logic was approximated as 2× the total size of the flip-flops, based on calibrated data from place-and-route results. In the figure, the total area overhead of bit bypass for an 8T array is smaller because bit bypass requires the same area while the macro it protects is larger than a 6T macro. Increasing the number of BB sets from the nominal case with no BB initially improves resiliency and reduces V_{min} for small area overhead, but saturates quickly at higher acceptable failure rates and lower voltages.

The simplest implementation uses flip-flops to store all state, and uses synthesis and place-and-route to implement the actual design. The area overhead can be improved by using latches instead of flip-flops, or through use of custom design to fold the redundancy inside the SRAM macro to allow for denser layout and more logic sharing.

Another possible technique to protect against failures in the tag array is to use line disable within the tags themselves, because only a small number of failures will need to be fixed.



(a) Bit bypass overhead versus maximum bitcell failure probability.

(b) Bit bypass overhead versus V_{min} for the “References” dataset.

Figure 5.11: Bit bypass trades off area overhead for increased resiliency.

However, this would require storing valid bits for each way in flip-flops outside the array, which would require more flip-flops than bit bypass. Also, line disable for tags would rely on spare lines, so would not work for stand-alone SRAM macros—a limitation that bit bypass avoids.

5.4 Protecting Against Intermittent Errors

The proposed DCR, BB, and LD schemes require knowledge of fault location to correctly program the redundancy. SRAM errors that appear after BIST, such as intermittent errors or soft errors, cannot be detected or fixed with DCR, BB, and LD. However, ECC can easily detect intermittent errors, because any bit flip that happens between a write operation and the final read operation can be detected.

5.4.1 Using ECC for Hard Faults

SECDED correction capability could potentially be shared between repairing hard and soft faults. Using the architecture-level soft error model from Section 2.5, Figure 5.12 shows FIT (fixed to 1×10^{-3}/bit) versus voltage with one day of accumulation for an L2 cache that uses SECDED to repair hard errors, and does not save any bits for ECC. The minimum FIT at 1×10^{-4} reflects the probability of accumulating multiple strikes to the same word during the 1 day period. As hard errors start occurring, SECDED is used to correct increasingly frequent

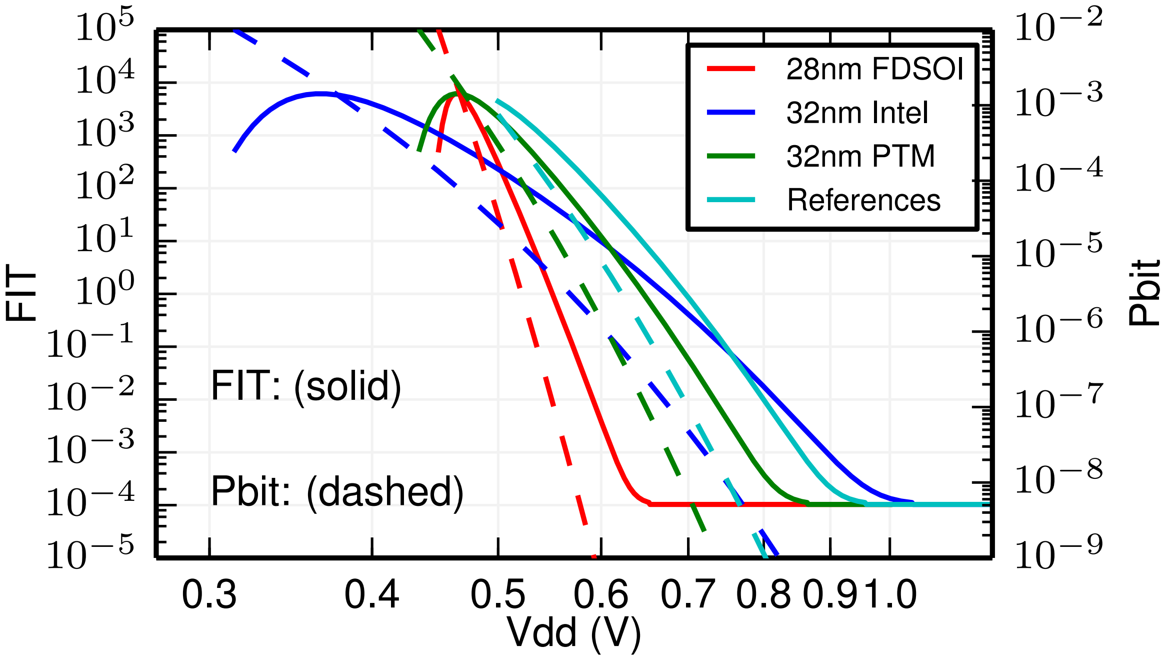


Figure 5.12: System FIT (with bitcell FIT=1×10^{-3}) with SECDED correction and 1 day of accumulation.

hard errors and leaves words vulnerable to soft errors, so the FIT increases dramatically. At extremely low voltages, many words with multi-bit hard errors are disabled, so the number of vulnerable words actually decreases and causes FIT to decrease. A FIT of 4000 corresponds to a MTTF of 30 years, so a system could meet FIT specifications when using SECDED to repair hard errors. Additionally, errors can be masked at the architecture level, such as the case where data is evicted before it is used, so this FIT value is an upper bound on FIT. These results also suggest that using ECC to program redundancy is possible, because the probability that a soft error occurs before the first error detection is low.

5.5 Summary

Architecture-level techniques that tolerate bitcell failures are particularly attractive because they are independent of technology and remain relevant for multiple process nodes. Existing work focuses on tolerating extremely high failure rates with either redundancy or ECC. However, decreasing V_{min} quickly faces diminishing returns, as bitcell failure rate increases exponentially with decreasing voltage. The holistic error model from Chapter 2 was used to compare the effectiveness of a variety of existing resiliency schemes and revealed general design guidelines about architecture-level resiliency.

Two proposed techniques, dynamic column redundancy with line disable and bit bypass target less ambitious V_{min} reduction and therefore are able to achieve substantial energy savings with very low area, energy, delay, and complexity overhead. The proposed schemes require a BIST to identify error location, which requires either non-volatile storage or a short test for every power-up. These schemes can be supplemented with simple SECDED ECC codes to protect against soft and intermittent errors.

Chapter 6

Resilient Processor Design

This chapter describes how the architecture-level techniques proposed in Chapter 5 were implemented in a 28nm resilient microprocessor to explore implementation issues and prove the concept. The resilient microprocessor applies these new resiliency techniques to the L1 and L2 cache of a RISC-V processor, and uses a custom built-in-self-test to identify fault locations and program the resiliency techniques. Additionally, ECC protection on the caches in the processor can be used to investigate the impact of intermittent faults.

6.1 Introduction

Persuasive proposals of resiliency techniques require building complete digital systems, because solutions developed in isolation at either the circuit or architecture level, without context within a full system, can lead to non-optimal designs. For example, at the circuit-level, switching from a 6T to 8T transistor improves speed and V_{min} while doubling area —a trade-off that is only worthwhile for arrays that are a small proportion of overall chip area. Or at the architecture-level, bit bypass lowers V_{min} while increasing area overhead—a trade-off that is worthwhile only if energy savings of a lower V_{min} offset the opportunity cost of the area increase.

Processors are an excellent system design target for two main reasons. First, processors are ubiquitous; design specifications and trade-offs are well understood, and due to the widespread usage of processors in consumer products such as smartphones, promising techniques have much improved chances of adoption and impact. Second, processors have a wide range of SRAM requirements. The level-1 (L1) cache directly determines the critical path of the processor and has high activity levels, so latency and V_{min} are important, while area is less critical. The level-2 (L2) cache consumes a large proportion of chip area, but latency has a smaller impact on performance, so area and V_{min} are important, while latency is less critical.

Circuit-level techniques in processors can reduce V_{min} and improve energy efficiency with low overhead, as described in Chapter 4, but as process technology changes, circuit-level techniques need to be redeveloped. For example, when foundries transitioned from planar transistors to FinFETs, transistor widths within a cell became quantized, and the wordline voltage needed to be reduced to maintain cell stability [76]. Or when new technology allowed PMOS strength to approach NMOS strength, it became helpful to replace the pass gates of 8T cells with PMOS devices to help single-ended read operations [22]. Even with methodologies that predict bitcell failure at design time, detailed in Section 3.1, the design schedule must allocate extra time for early tape-outs to verify SRAM operation. A technique that divorces bitcell failure tolerance from process technology would simplify SRAM array design and shorten product launch cycles. Therefore, architecture-level techniques are a very promising approach to reducing V_{min} because they are mostly decoupled from process technology,

Most previous work in the architecture-level resiliency area is theoretical, with no RTL or silicon implementation, because actual implementations are deemed unnecessary: by assuming a bitcell failure curve, it is possible to evaluate effectiveness without an actual implementation. Section 5.1 includes an extensive list of simulation-based studies of architecture-level techniques. However, skipping silicon verification, or even RTL implementation, obscures potential flaws. First, optimistic failure curve assumptions could unfairly overestimate scheme effectiveness. Second, many techniques target the data portion of caches only, but there are SRAM macros elsewhere in the design—in the branch predictor or tags for example—that need separate methods of protection. Generally, researchers propose protecting these independent macros with circuit-level techniques; but without silicon verification, the effectiveness of protecting independent macros is unknown. Third, abstract implementations hide true technique complexity. Schemes that look simple could have many complex corner cases or add logic that increases the critical path, and therefore scheme viability is unknown until the implementation is completely finished.

Relatively few architecture-level techniques have been adopted into products. Architecture-level techniques can be categorized into ECC-based, where errors are corrected during operation using error-correcting codes, and redundancy-based, where error locations are determined ahead of time and redundancy is used to avoid failing locations. In general, ECC-based schemes have a lower barrier to adoption, because ECC has been industry-standard practice to protect against soft errors, and extending correction capability to fix hard faults is considered a low risk approach. Intel has reported using ECC in the L3 cache to reduce V_{min} [77]. However, redundancy-based schemes that require BIST have not been so readily adopted, because they require a non-volatile location to store fault locations. Section 6.4.5 will describe some possible solutions to store fault locations to promote adoption of redundancy-based schemes. Intel announced that the L3 cache in the Ivy Bridge processors can be shrunk dynamically, but this is not a redundancy technique—it avoids storing fault location by simply disabling the same ways every time, and therefore only achieves a 30mV V_{min} reduction [78], [79].

The SWERVE (SRAM With ECC and Re-programmable redundancy to aVoid Errors) project aims to lower the minimum operating voltage of a processor using the architecture-level redundancy-based schemes proposed in Chapter 5, and will use measurements from a full silicon implementation to validate failure rate assumptions and investigate the impact of intermittent errors. The cache tag SRAMs are protected by bit bypass (BB), and the cache data SRAMs are protected by line disable (LD) and per-set dynamic column redundancy (DCR). An at-speed SRAM built-in-self-test (BIST) detects SRAM errors before operation, and uses the error information to program BB, DCR, and LD. Additionally, error correction on every SRAM allows for in-situ logging of SRAM errors during runtime to determine the contribution of intermittent errors not detected by BIST.

6.2 System Architecture

Figure 6.1 shows the system diagram of the proposed processor, referred to as SWERVE. The processor is based on a 64-bit RISC-V 6-stage single-issue in-order processor (Rocket), which supports page-based virtual memory, boots modern operating systems [80], and is publicly available [81]. Many modifications, described in the following sections, were made to the design to support DVFS, SRAM BIST, error correction and logging, and dynamic redundancy.

The processor has L1 and L2 caches to hide memory latency. To support DVFS, the processor is split into three independent voltage and frequency domains with the pipeline and L1 in one domain, the L2 cache in the second domain, and the uncore with digital IO pads to connect to off-chip in the third domain. Because the L1 cache and L2 cache use different sizes of bitcells and have very different activity rates, separate voltage domains will enable a more energy-efficient design. The uncore communicates between the L2 cache and the main memory off-chip, and is supplied by a fixed 1V supply. Asynchronous FIFOs and level shifters allow communication between the voltage and frequency islands.

The VDDCORE domain holds the Rocket pipeline and L1 cache. The L1 cache uses an 8T-based SRAM array to support an independent read and write every cycle. Because these caches are small, the 2× area overhead of the 8T cell is worth the increased number of ports and improved speed. A BIST controller searches for SRAM failures, and records the errors in a local error buffer. The core has control status registers (CSR), that allow the processor and off-chip host to communicate by reading and writing to the same set of 32-bit registers.

The VDDSRAM domain contains the 1MB L2 cache. The L2 cache uses a high density 6T-based SRAM array to maximize area efficiency. Like the VDDCORE domain, the VDDSRAM domain has an independent BIST and CSR. The BIST is not shared between the core and L2 to prevent timing paths traversing multiple voltage domains. The L2 cache is split into four interleaved parallel banks because the L1 cache supports multiple outstanding misses, and to support future designs with multiple processor cores.

The uncore domain supports communication between on-chip and off-chip. Three independent clock sources from off-chip clocks pass through clock receivers. Multiplexers enable selection of different clock sources for every clock tree in the system. Off-chip communication takes place through digital IO pads at 100MHz with a 16-bit input and 16-bit output data bus, and the host is clocked by recovering a divided version of the uncore clock.

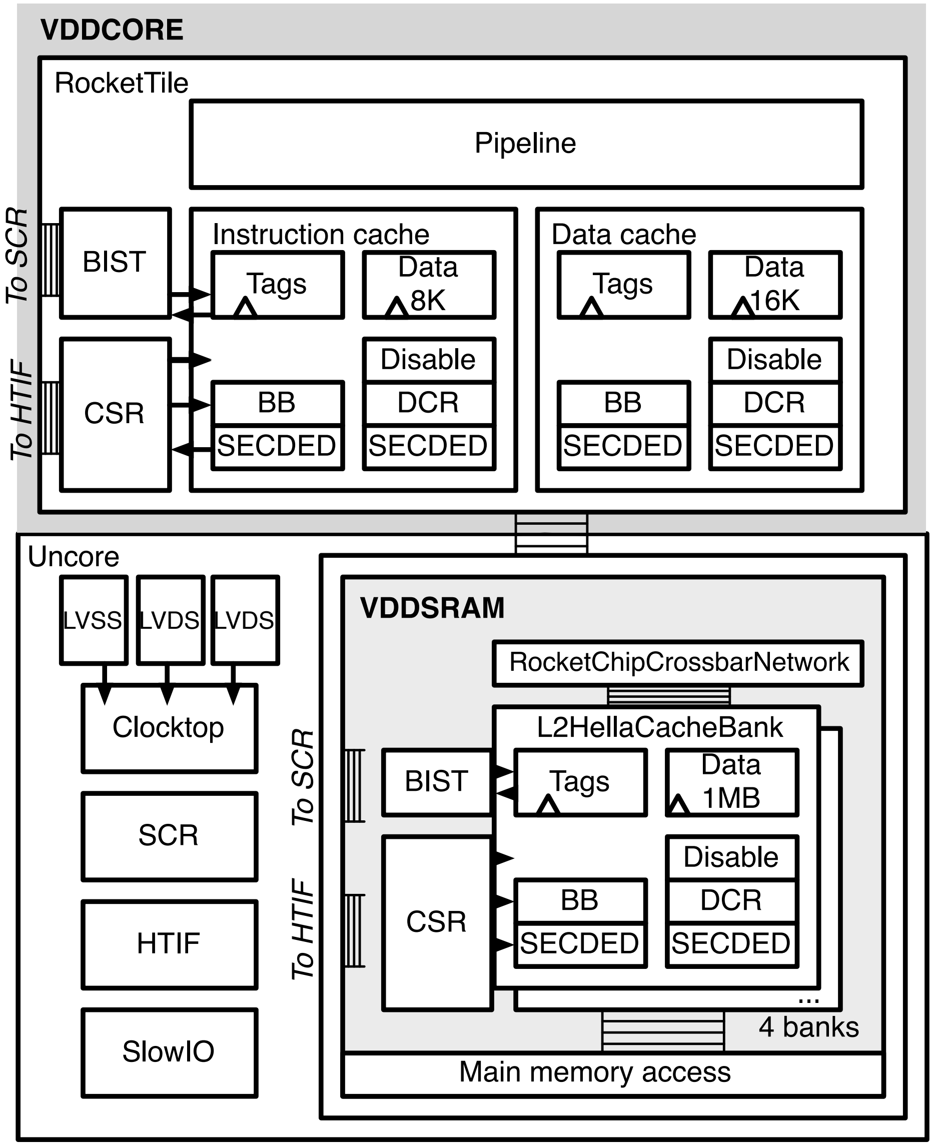


Figure 6.1: High-level overview of the SWERVE system architecture.

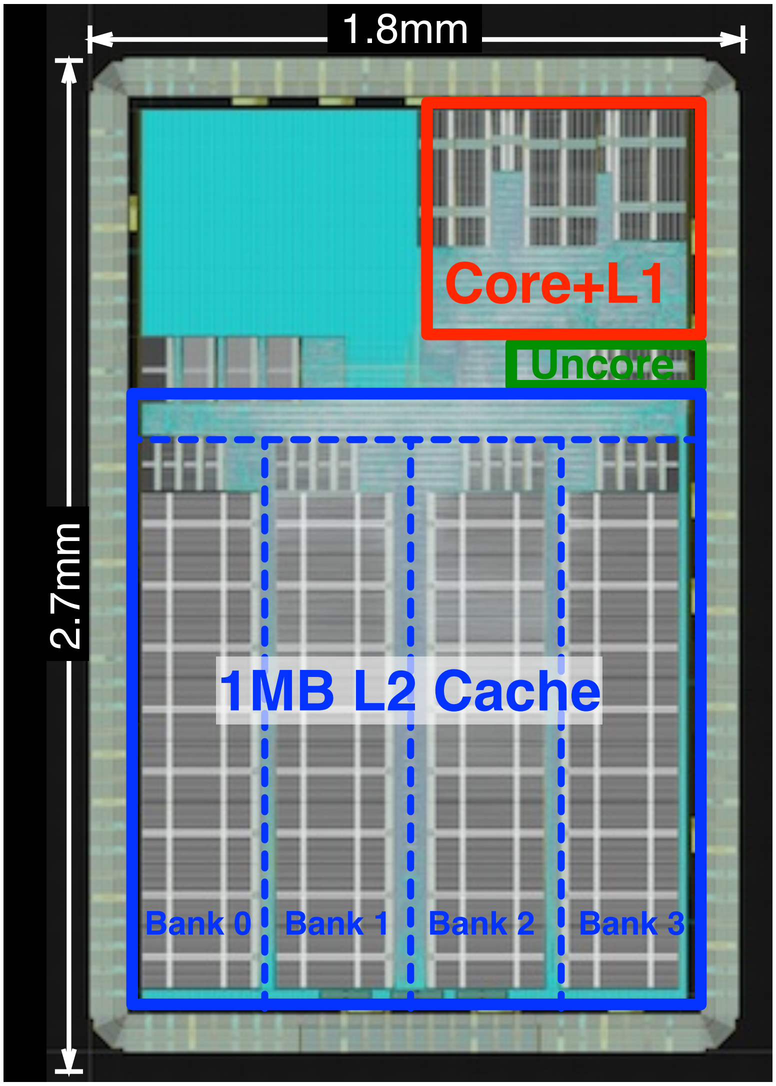


Figure 6.2: SWERVE floorplan showing the physical layout of the core, L1 cache, and L2 cache.

Figure 6.2 shows the chip implementation floorplan. The single-core processor is fabricated in a TSMC 28nm HPM process with a 2mm by 3mm die area. A large proportion of the die area is filled by a 1MB L2 cache. In addition to the processor and cache, four SRAM macros with sense amplifier studies are embedded into the L2 cache domain. The chip is surrounded by 124 wire-bonded IO pads designed for chip-on-board packaging. The processor was designed in Chisel [82], which generated Verilog for implementation, and the BIST engine was programmed directly in Verilog. The design was implemented with a Synopsys toolchain of Design Compiler for synthesis, IC Compiler for place-and-route, StarRCXT for parasitic extraction, and Primetime for timing signoff.

6.3 Programmable Built-In-Self-Test (BIST)

This section describes the on-chip BIST engines that identify fault locations. An effective BIST is critical because the proposed redundancy methods, DCR and BB, rely on perfect knowledge of fault location. For decades, industry has been using Memory Built-In-Self-Test (MBIST or simply BIST) to prevent shipping non-functional products to customers. For the DCR and BB scheme, the same BIST circuitry will be used, but instead of rejecting parts with bitcell failures, an algorithm will program the redundancy to bypass the failure cells. The programming operation is performed at the lowest operating voltage, which assumes that all failures are monotonic with voltage (a cell that works at a certain voltage will not fail at higher voltages).

6.3.1 BIST Architecture Overview

Various companies offer CAD tools that automatically insert MBIST [83]. However, a custom BIST architecture was developed to avoid dependencies on a complex product and to ease customization. The BIST is highly programmable and supports a large number of possible March tests with the same hardware. The proposed BIST implementation is split into two parts: control and datapath, shown in Figure 6.3. The datapath contains all of the SRAMs under test, along with pipeline registers for the inputs and outputs to allow all of the test logic to operate at a higher frequency than the SRAM—ensuring that SRAM reads and writes are on the critical path. The control portion is a finite state machine that provides at-speed input vectors of address and data to the SRAM. The entire BIST, including control, datapath, and SRAMs, all operate on a single voltage domain to avoid multi-voltage timing paths. A separate BIST control and data path exists for each voltage domain. The BIST is programmed through system control registers (SCR) from off-chip.

6.3.2 BIST Control

A style of memory test patterns known as March tests provide high fault detection coverage in SRAM [84]. The time required to run March tests scales with `n`, which makes March tests suitable for ever-increasing SRAM sizes. March tests can detect a wide range of faults from simple stuck-at faults within bitcell to faults in the address decoder. Table 6.1 lists some of the March tests supported by this BIST implementation, derived from [85].

Tests are orchestrated by a programmable state machine shown in Figure 6.4. In addition to six programmable March elements (TEST0 through TEST5), there are extra SAFEWRITE and SAFEREAD states, which pause when entering the state to raise the domain to a high voltage and pause again when leaving the state to lower the domain to a lower testing voltage. Traditional March tests cannot diagnose the cause of an error—errors are detected during read operations, and it isn't known whether the error was caused by an incorrect write, a stability upset during a different access, or an incorrect read. By reading

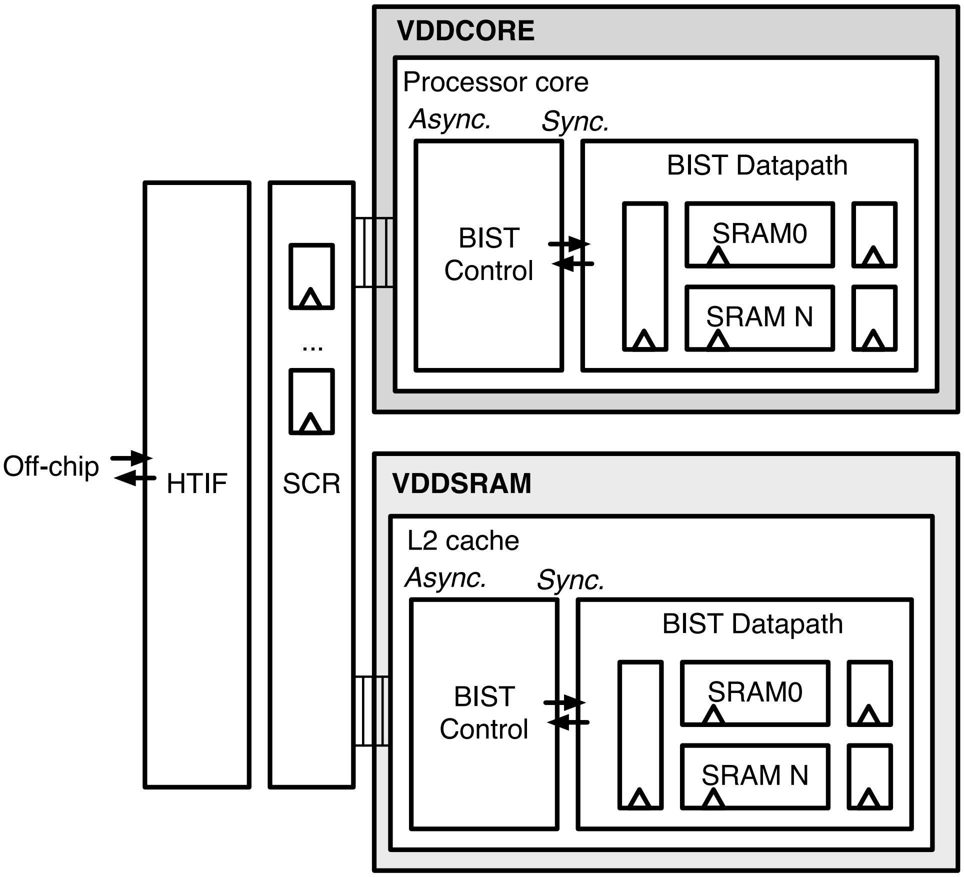


Figure 6.3: BIST is organized as separated control and datapath in each voltage domain, and communicates with off-chip through system control registers (SCR) and the host-target interface (HTIF).

Table 6.1: List showing a subset of supported March tests by on-chip BIST controller.

- MATS
- MATSP
- MATSPP
- March CM
- March A
- March B
- Algorithm B
- March CP
- MOVI
- March 1_0
- March TP
- March U
- March X
- March Y
- March LR
- March LA
- March RAW
- March RAW1
- March AB
- March AB1
- March BDN
- March SR
- March SS

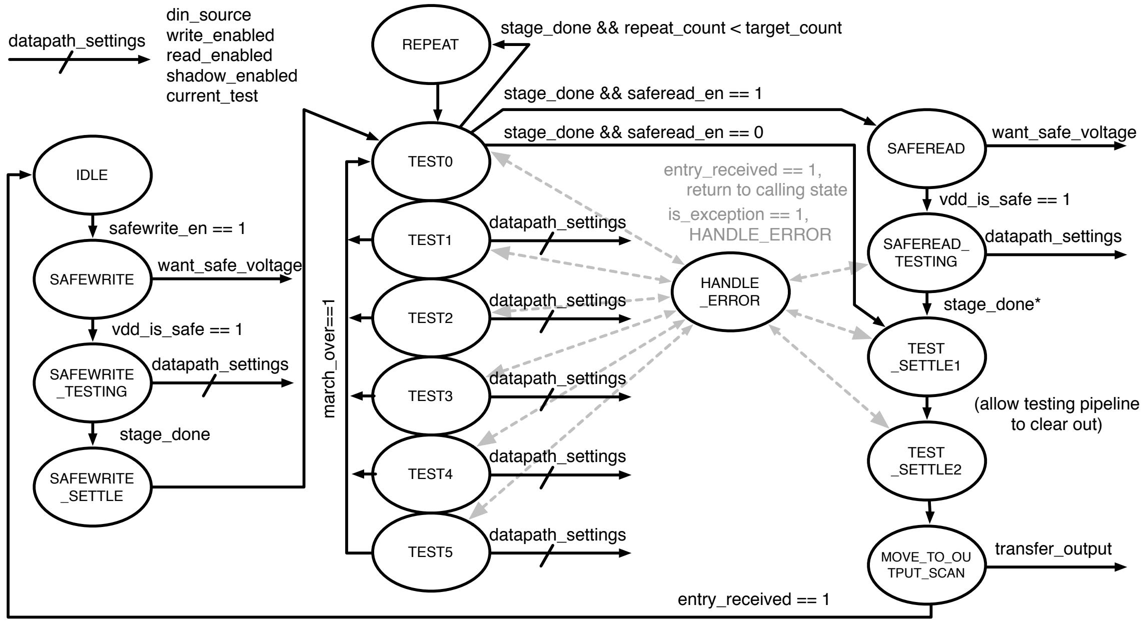


Figure 6.4: BIST control state machine allows a wide variety of programmable March tests.

and writing the SRAM at high voltage, failure causes can be ruled out to isolate the actual failure cause.

The number of active test states can vary from one to six, and the address counter start address, stop address, and address increment value are all programmable. When an error is detected, the machine moves to the HANDLE_ERROR state, which allows the off-chip host to retrieve the error entry from every SRAM. Each error entry includes the entire output data word, the address of the access that failed, and the current test number. To save area, only a single error can be stored at once, requiring the state machine to constantly stop and restart when downloading errors. After the errors have been recorded, the test continues at the state that caused the error. Every active test state is visited before updating the address. After the test is completed, two settling states allow the 3-stage BIST datapath pipeline to clear out any remaining errors.

Each test state (eg. TEST1) can be programmed to either be a write or read operation. For write operations, unique input data can be programmed. For read operations, the expected output data can be programmed. Additionally, for 8T designs, a shadow access can be defined for read operations to write an adjacent cell, causing a half-select condition during the actual read operation to force a worse case read.

6.3.3 BIST Datapath

The goal of the BIST datapath is to correctly exercise the SRAM inputs, and compare SRAM outputs to expected results, at the maximum speed of every SRAM in the design. Figure 6.5 shows the BIST datapath that tests all of the SRAMs in the L1 cache. To ensure that the SRAM itself is on the critical path, and not the control, the output is first latched by transparent-high latches before being sent to a normal positive-edge flip flop. Adding latches provides two advantages: first, the control logic can be two times slower than the SRAM, and second, duty-cycle control of the clock can further stress SRAM timing failures.

A clock multiplexer chooses whether the SRAMs are controlled by the core clock during normal operation or the BIST clock during testing. The core and BIST are both in reset during transitions between operating modes, so no special timing is needed for the BIST enable signal on the select input of the multiplexer.

The datapath width of the BIST is 73 bits, which corresponds to the maximum word width of individual SRAM macros. Table 6.2 summarizes the mapping between SRAM macros and macros visible to BIST. Memories that are logically wider than 73 bits (such as 128-bit entries in the data arrays), are split into separate physical SRAMs and appear to be separate macros to BIST. When programming DCR, column failures are shifted appropriately. For example, an error in column 6 of the upper macro with 64-bit width would correspond to column 70.

Different SRAMs have different depths as well. The address range of BIST algorithms is usually programmed to equal that of the deepest array. Because all macros are tested in parallel, macros with shallower depths have an extra bit that turns off read/write signals and disables comparisons for the non-existent entries.

When an error is detected, the address, current test, and output data vector is inserted into an error buffer. A separate register remembers which error buffers received valid entries. The error buffers are only one entry deep, so the BIST must pause every time an error is detected and wait for the host to retrieve the error information.

6.3.4 BIST Interface

The interface between the BIST datapath and control is synchronous, while the interface between the BIST control and the SCR is asynchronous. For the asynchronous interface, only a few handshake signals are synchronized, while the rest are assumed to remain constant during operation. The synchronized signals will be referred to as handshake signals, while the unsynchronized signals will be referred to as settings signals (for inputs) or response signals (for outputs). Communication between the BIST and uncore is asynchronous, so all handshake signals are synchronized as shown in Figure 6.6. The SCR registers only have a 32-bit width for compatibility with the 32-bit small co-processor. To avoid bugs, separate addresses are used for different settings (even though they could be compacted if the address space gets crowded).

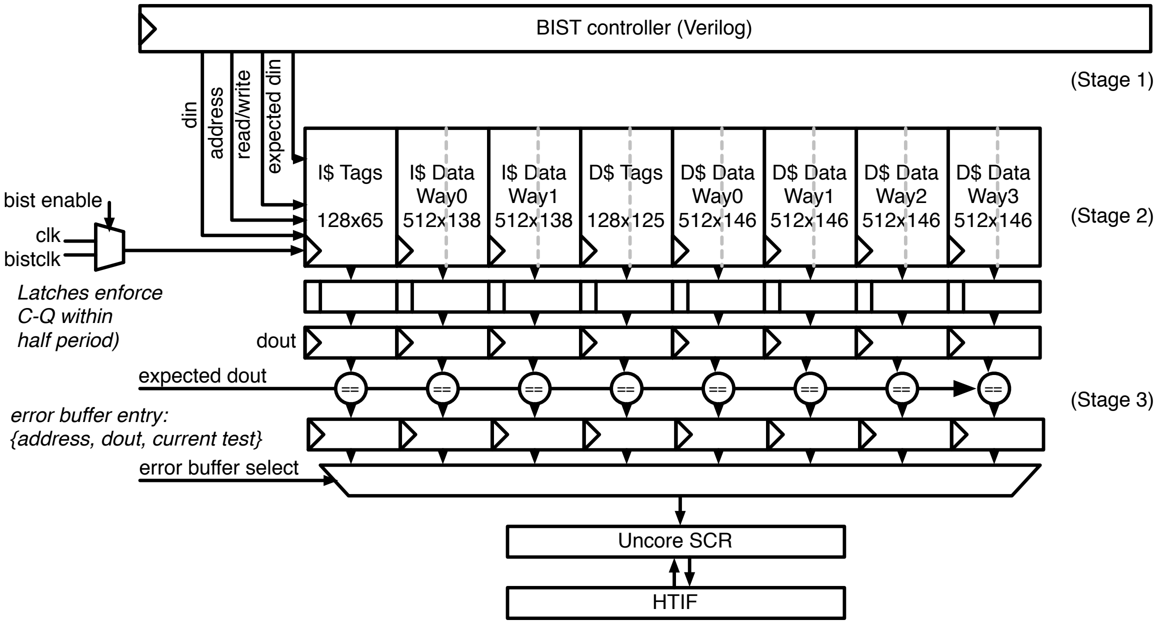


Figure 6.5: BIST datapath reads and writes every SRAM in parallel at the maximum SRAM frequency.

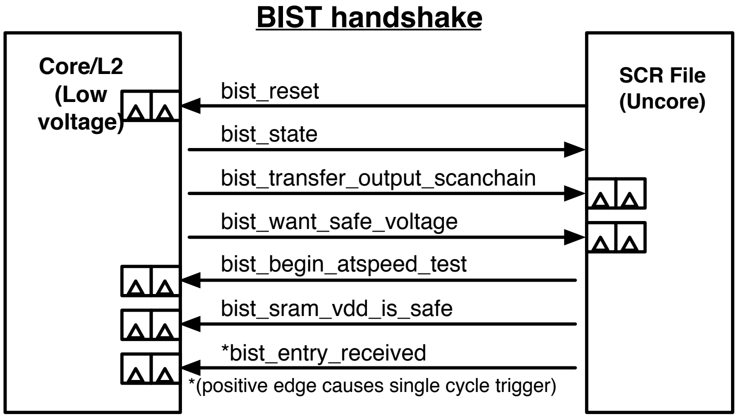


Figure 6.6: Signals to and from BIST are synchronized between differing voltage and frequency domains.

Table 6.2: Summary of every testable SRAM in the chip, with corresponding size and BIST macro address.

Structure	Size	SRAM Name	SRAM #	SRAM Size
L1 Instruction Cache Data (2 way)	16KB (17,644B)	RocketTile.icache.icache.icache_data_0.sram1	0	512x69
		RocketTile.icache.icache.icache_data_0.sram0	1	512x69
		RocketTile.icache.icache.icache_data_1.sram1	2	512x69
		RocketTile.icache.icache.icache_data_1.sram0	3	512x69
L1 Instruction Cache Tags	128x65 (1,040B)	RocketTile.icache.icache.tag_array.mem.sram0	4	128x65
L1 Data Cache Tags	128x125 (2,000B)	RocketTile.dcache.meta.tag_arr.mem.sram1	5	128x62
		RocketTile.dcache.meta.tag_arr.mem.sram0	6	128x63
L1 Data Cache Data (4 ways)	32KB (37,376B)	RocketTile.dcache.data.dcache_data_2_1.sram1	7	512x73
		RocketTile.dcache.data.dcache_data_2_1.sram0	8	512x73
		RocketTile.dcache.data.dcache_data_0_1.sram1	9	512x73
		RocketTile.dcache.data.dcache_data_0_1.sram0	10	512x73
		RocketTile.dcache.data.dcache_data_2_0.sram1	11	512x73
		RocketTile.dcache.data.dcache_data_2_0.sram0	12	512x73
		RocketTile.dcache.data.dcache_data_0_0.sram1	13	512x73
		RocketTile.dcache.data.dcache_data_0_0.sram0	14	512x73
L2 Bank X Tags (X=0,1,2,3)	512x229 (14,656B)	uncore.outmemsys.L2HellaCacheBank_X_.meta.meta.tag_arr.mem.sram3	15,21,27,33	512x57
		uncore.outmemsys.L2HellaCacheBank_X_.meta.meta.tag_arr.mem.sram2	16,22,28,34	512x57
		uncore.outmemsys.L2HellaCacheBank_X_.meta.meta.tag_arr.mem.sram1	17,23,29,35	512x57
		uncore.outmemsys.L2HellaCacheBank_X_.meta.meta.tag_arr.mem.sram0	18,24,30,36	512x58
L2 Bank X Data (8 ways) (X=0,1,2,3)	Bank: 256KB (282,624B) Total: 1MB (1,104KB)	uncore.outmemsys.L2HellaCacheBank_X_.data.array.sram1_0	19,25,31,37	4096x69
		uncore.outmemsys.L2HellaCacheBank_X_.data.array.sram1_1	19,25,31,37	4096x69
		uncore.outmemsys.L2HellaCacheBank_X_.data.array.sram1_2	19,25,31,37	4096x69
		uncore.outmemsys.L2HellaCacheBank_X_.data.array.sram1_3	19,25,31,37	4096x69
		uncore.outmemsys.L2HellaCacheBank_X_.data.array.sram0_0	20,26,32,38	4096x69
		uncore.outmemsys.L2HellaCacheBank_X_.data.array.sram0_1	20,26,32,38	4096x69
		uncore.outmemsys.L2HellaCacheBank_X_.data.array.sram0_2	20,26,32,38	4096x69
		uncore.outmemsys.L2HellaCacheBank_X_.data.array.sram0_3	20,26,32,38	4096x69
DUT2		uncore.outmemsys.pifeng_srams.sram_dut2.sram0	40	1024x72
DUT1		uncore.outmemsys.pifeng_srams.sram_dut1.sram0	41	1024x72
DUT0		uncore.outmemsys.pifeng_srams.sram_dut0.sram0	42	1024x72

A BIST algorithm is run by programming the BIST settings registers, then providing the correct handshake signals to complete the test. The test protocol for every test is stored in a textual control file. A pre-compiled C program running on the host (or an on-chip co-processor in future designs) parses the file. The commands are separated into a text file to avoid requiring a recompile for different tests.

The example control file for the MATS++ test is shown in Figure 6.7, and is automatically generated with a script from its March description: `a(w0);u(r0,w1);d(r1,w0,r0)`. Comments within the figure describe the effect of each command. The settings programmed by this file control both the datapath (such as the data values and address steps) and the control (to determine which March elements are enabled).

```
* MATSPP
 bist_reset =1 // toggle reset
 bist_reset =0
 bist_safe_write_enable =0
 bist_safe_read_enable =0
 bist_test0_enable =1
bist_test0_mode=1 // write operation
bist_test0_compare_to=0
 bist_test0_din =0x000000000000000000 // write a zero
 bist_test0_start_address =0 // next 3 lines  control  address counter
 bist_test0_end_address =1023
 bist_test0_step_bit =1
 bist_test1_enable =0 // only use single  march element (w0 only)
 bist_test2_enable =0
 bist_test3_enable =0
 bist_test4_enable =0
 bist_test5_enable =0
 bist_begin_atspeed_test =1 // toggle handshake to begin test
 bist_begin_atspeed_test =0
@bist_transfer_output_scanchain==1 // poll until BIST is complete
#echo // report failure  data
 bist_entry_received =1 // return on−chip state machine to idle
 bist_entry_received =0
************
 bist_safe_write_enable =0
 bist_safe_read_enable =0
 bist_test0_enable =1
bist_test0_mode=0 // read operation
bist_test0_compare_to=2 // compare to test0_din=0x0
 bist_test0_din =0x000000000000000000
 bist_test0_start_address =0
 bist_test0_end_address =1023
 bist_test0_step_bit =1
 bist_test1_enable =1
bist_test1_mode=1 // write operation
bist_test1_compare_to=0
 bist_test1_din =0    xffffffffffffffffff      // write  all  ones
 bist_begin_atspeed_test =1
 bist_begin_atspeed_test =0
@bist_transfer_output_scanchain==1
#echo
 bist_entry_received =1
 bist_entry_received =0
 ...
```

Figure 6.7: Excerpt from example control file implementing the MATS++ SRAM test.

6.4 Architecture-level Resiliency

Once BIST has determined the locations of all SRAM faults, the proposed architecture-level resiliency techniques, BB, DCR, and line disable, can be programmed to allow the system to avoid failing bits. These techniques were designed to have minimal overhead in terms of area, energy, and delay while tolerating SRAM failure rates 1,000,000 times higher than normal. The implementations described in the following section are optimized for the specific Rocket pipeline, but similar implementations are possible for a wide variety of processor architectures.

6.4.1 BB Implementation

Section 5.3.1 summarizes the general principle of bit bypass (BB)—writes and reads to rows with known faults use a small number of redundant flip flops to bypass the failing bits. Figure 6.8 describes the specific implementation of bit bypass in this chip. Bit bypass is instantiated in the RTL code as a wrapper around any array, with additional input signals used to program the bit bypass sets. A specific number of *replacement sets* match different access addresses in the macro. For each each replacement set, there are a particular number of *replacement entries*. Bit bypass operates on the logical organization of the array, not the physical organization, so two separate replacement sets that match different address could repair bits in the same physical row for interleaved macros. Therefore bit bypass can be conceptualized as row redundancy on the logical organization of a macro.

Bit bypass uses a separate reset signal called `redundancy_reset`. Two reset signals are needed because BB needs to be programmed while the processor remains in reset. Each replacement entry has a valid bit that is reset to zero (not valid). The programming operation inserts the row address of the set, and column address and valid bit for each replacement entry.

The targeted allowable bitcell failure rate of a macro will determine the number of bit bypass sets and entries required. For the L1 instruction cache tag array, there are 5 sets of 2 entries. For the L1 data cache tag array, there are 7 sets of 2 entries. For each bank, the L2 cache tag array has 22 sets of 2 entries. Table 6.3 shows the improved allowable failure rate and potential energy savings for all of the tag arrays after adding bit bypass. The maximum failure rate was targeted at 10^{-4} to correspond with the maximum effectiveness of DCR and LD.

The total area overhead of this scheme is summarized in Table 6.4. To achieve the target resiliency, only 10 total bits need to be repaired in the L1 instruction cache, but 134 total flip-flops are required to store the replacement set row and column address for each redundant bit. While the overhead compared to the macro is somewhat large (16%), because the tag arrays contribute to a small proportion of cache area (and an even smaller proportion of overall chip area), the area overhead of this technique is only around 1%. This calculation assumes 100% standard cell utilization, so the area overhead can be multiplied by the appropriate factor to estimate the final implementation overhead. Due to ungrouping during optimization, the L1

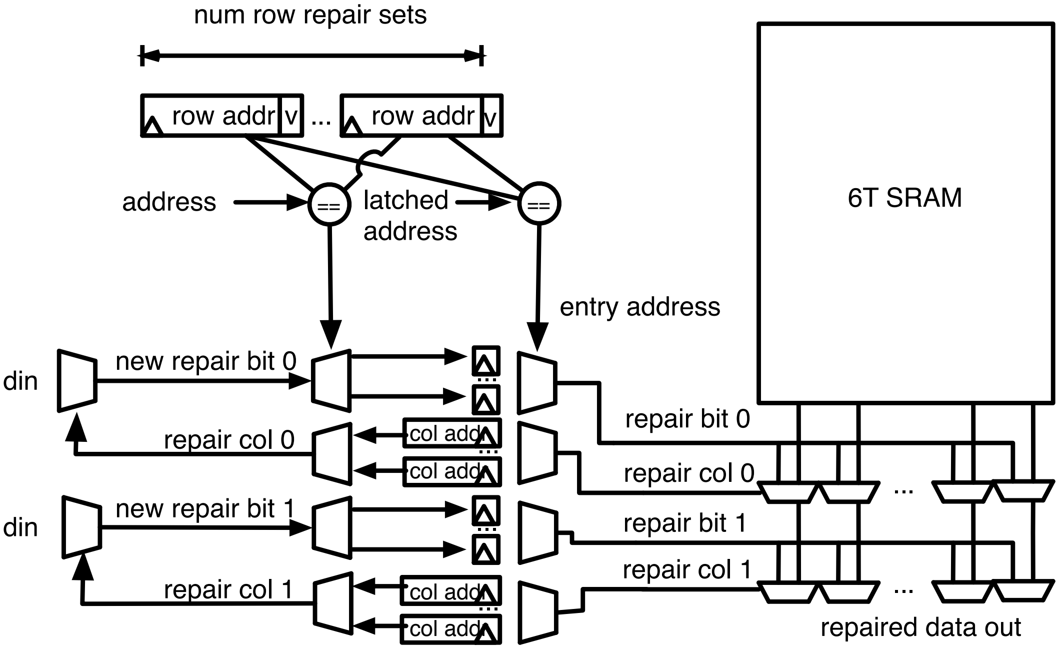


Figure 6.8: Extra combinational and sequential logic required to implement bit bypass.

Table 6.3: Effectiveness of bit bypass at increasing the acceptable failure rate and reducing the minimum operating voltage.

Cache	Technique	Max. p_{bit}	V_{min}	Energy Reduction
L1 Instruction Tags	Nominal	1.2×10^{-7}	700mV	38%
	Bit bypass (5 sets)	1.3×10^{-4}	540mV	
L1 Data Tags	Nominal	6.3×10^{-8}	710mV	52%
	Bit bypass (7 sets)	1.3×10^{-4}	540mV	
L2 Tags	Nominal	8.5×10^{-9}	755mV	48%
	Bit bypass (22 sets)	9.8×10^{-5}	545mV	

data cache tag overhead could not be calculated, but will be very similar to the L1 instruction cache. If the L1 cache used 6T cells, the area overhead would increase to around 2%.

Bit bypass adds delay to various timing paths. For write operations, the address setup time is increased, because a comparison operation needs to be completed in order to determine which data input bits will be stored in the correction entries. For read operations, the clock-to-Q is slightly increased by the levels of multiplexing required to substitute redundant bits for failing bits. Table 6.5 compares the critical path of bit bypass to a stand-alone SRAM macro for both read and write operations. Bit bypass did not appear on any of the critical paths of the final design and therefore has no timing overhead for this particular system. If the data address setup time is long, the comparison can be pipelined by using a write buffer to always write in the following cycle at the cost of increased overhead.

Table 6.4: Area overhead for bit bypass after synthesis.

Macro	SRAM area	Extra sequential area	Extra combinational area	BB overhead	Cache area	Cache overhead
L1 Inst. Cache Tags	6933 μm^2	470 μm^2(134 flops)	672 μm^2	16%	89,537 μm^2	1.3%
L2 Tags (per bank)	33,103 μm^2	1,205 μm^2(649 flops)	2,129 μm^2	10%	544,378 μm^2	0.6%

Table 6.5: Timing overhead for bit bypass during read and write operations after synthesis, for 70 FO4 processor.

Operation	Nominal path (FO4)	BB path (FO4)	Overhead (FO4)
Write address setup	24	42	18
Read clock-to-Q	26	34	8

6.4.2 DCR Implementation

Section 5.2.1 explains the general concept of dynamic column redundancy (DCR) and compares it to other proposed resiliency schemes, while this section explains a specific implementation in the L1 and L2 cache of the SWERVE processor.

DCR uses one extra redundant column to avoid up to one failing bit per set by steering around the failing bit. The DCR encoder creates an `n+1` bit word from an n bit input, with a redundant entry next to the failing column, and the decoder converts the `n+1` bit word back to n bits without the failing column, as shown in Figure 6.9. For every row that is accessed, a different redundancy address (RA) changes the thermometer code to avoid different columns in different rows.

Figure 6.10 shows the Chisel code used to implement this circuitry. Much of the complexity required to implement this scheme comes from programming the redundancy address into the appropriate tag sets based on failure locations identified by BIST, rather than the simple encoder and decoder added to the datapath.

The RA is stored inside the tag array, is shared between all lines in a set, and is read either prior to data array accesses (in the case of writes) or in parallel with data array accesses (in the case of reads), as shown in Figure 6.11. Most of the overhead of the DCR scheme is caused by storing and reading this redundancy address.

Table 6.6 summarizes the area overhead of the DCR scheme in terms of increased tag array area to store the redundancy address. The encoder and decoder are small, $\approx 300\mu m^2$, and considered negligible in comparison to the data array area of over $150,000\mu m^2$.

DCR has a small timing overhead. For write operations, the data needs to be encoded around the failing column, and for read operations, the data needs to be decoded around the failing column. This adds around 4 FO4 of delay through the single stage of 2-to-1 multiplexers.

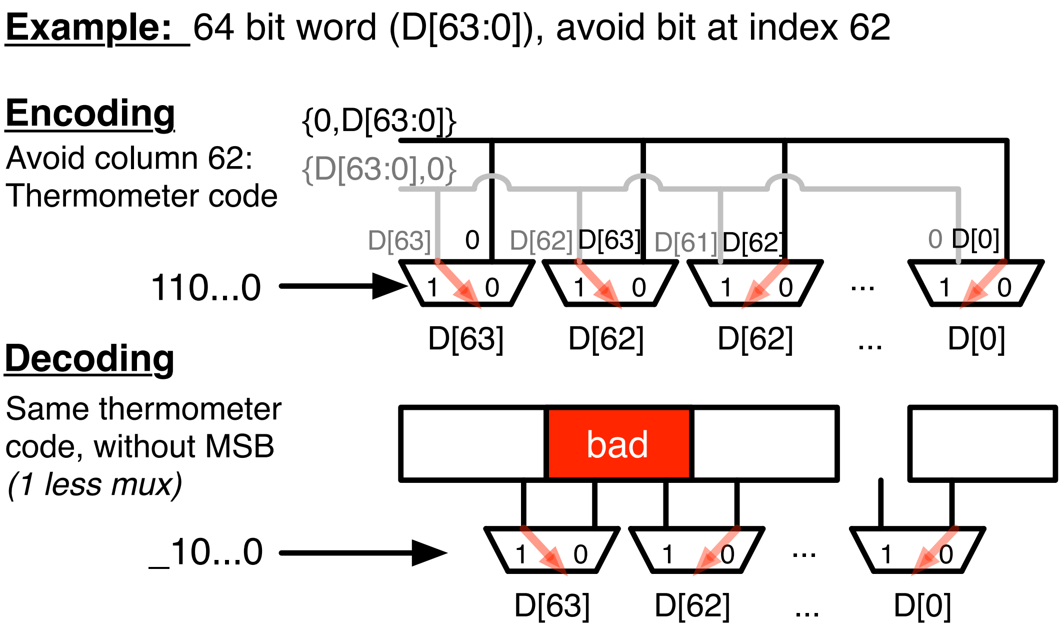


Figure 6.9: Encoding and decoding uses 2:1 multiplexers and a thermometer code to avoid one column.

Table 6.6: Area overhead for DCR in the data and tag arrays.

Cache	Nominal width (data+ECC)	Nominal width ways*(tags+ECC)	DCR overhead (data+ECC)	DCR overhead (tags)	Cache overhead
L1 Inst. Cache	128+9=137	2*(19+7)=52	1 (0.7%)	8+5=13 (25%)	2.9%
L1 Data Cache	64+8=72	4*(21+6)=108	1 (1.4%)	8+5=13 (12%)	2.2%
L2 Cache	128+9=137	8*(21+6)=216	1 (0.7%)	8+5=13 (6%)	1.1%

Table 6.7 summarizes the effectiveness of DCR at lowering V_{min}. Section 5.2.1 shows that having a redundancy address per way instead of per set would correct more bits, and lower the minimum operating voltage. However, for this implementation, it was determined that the large increase in tag array size due to storing dramatically more redundancy addresses wasn't worth the additional voltage reduction.

6.4.3 Line Disable Implementation

Section 5.2.4 concluded that multi-bit failures limit the minimum operating voltage. While BB can repair multi-bit failures, DCR can only repair single bit failures. However, because caches are usually set associative, disabling a single way in a set can prevent accesses to lines with multi-bit failures without harming functionality. The minimum operating voltage is set

```
object RedundancyEncode
{
  def apply(narrow_input: Bits, avoid_index: UInt): Bits = {
    val therm_code = SInt(-1,narrow_input.getWidth+1) << avoid_index
    val right_choice = Cat(UInt(0,width=1),narrow_input)
    val left_choice = right_choice << 1
    val muxes = Vec.tabulate(narrow_input.getWidth+1){ i =>
        Mux(therm_code(i),left_choice(i),right_choice(i)) }
    muxes.toBits
  }
}

object RedundancyDecode
{
  def apply(wide_input: Bits, avoid_index: UInt): Bits = {
    val therm_code = SInt(-1,wide_input.getWidth) << avoid_index
    val muxes = Vec.tabulate(wide_input.getWidth-1){ i =>
        Mux(therm_code(i),wide_input(i+1),wide_input(i)) }
    muxes.toBits
  }
}
```

Figure 6.10: Chisel code implementing the encoding and decoding procedure to avoid a failing column.

Table 6.7: Effectiveness of DCR at increasing the acceptable failure rate and reducing the minimum operating voltage.

Cache	Technique	Max. p_{bit}	V_{min}	Energy Reduction
L1 Inst. Cache Data	Nominal	7.1×10^{-9}	760mV	36%
	DCR	5.1×10^{-6}	610mV	
L1 Data Cache Data	Nominal	3.4×10^{-9}	775mV	34%
	DCR	2.0×10^{-6}	630mV	
L2 Data	Nominal	1.1×10^{-10}	850mV	36%
	DCR	2.4×10^{-7}	680mV	

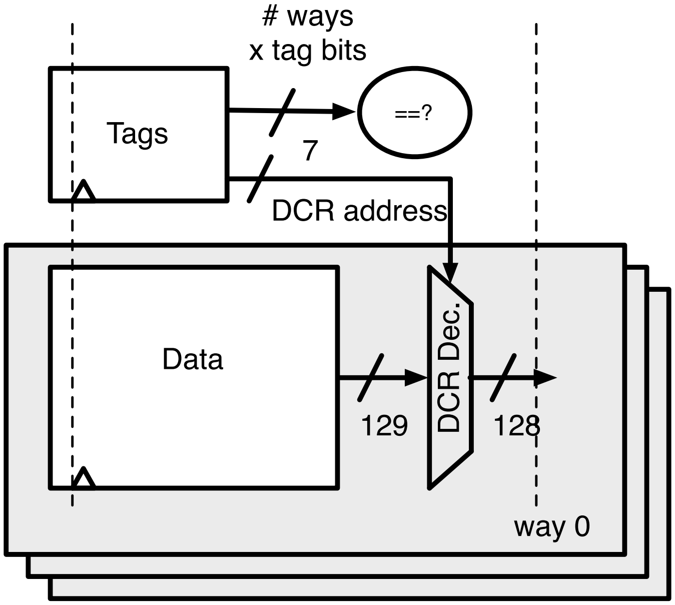


Figure 6.11: The redundancy address is accessed in parallel with the data array to identify the failing column corresponding to the accessed row.

at a target where only a few lines are disabled and the cache still has 99% capacity, which removes the need for analysis of the cost of increased misses due to decreased capacity.

Disable bits are stored with each tag, and are always read and written with the rest of the tag. When reseting the redundancy information, all of the tags and valid bits are cleared, and all of the disable bits are set to zero, indicating that all ways are enabled. After BIST is complete, the lines that need to be disabled have the appropriate disable bit set high. When disabling the way, the tag value no longer matters, so tags do not need to be written separately from the disable bit.

The only time disable bits need to be checked is during refills, as the processor is not allowed to allocate data into a disabled way. A pseudo-random algorithm shown in Figure 6.12 selects the replacement way from among the enabled ways. During tag reads, the disable bit is not necessary, because the valid bit cannot be high when the line is disabled, so tag comparison will never match a disabled way. In future implementations with online programming, the line needs to be released to write back dirty data, which would also prevent the valid bit and disable bit from both being high.

By disabling lines, the minimum operating voltage can be decreased further. From Table 6.7, DCR alone has limited effectiveness—only tolerating a failure rate of 2×10^{-6}. At this voltage and failure rate, multi-bit failures are limiting the voltage—99.97% of words have 0 errors, and 0.02% of words are repaired by DCR. If lines with multi-bit failures are disabled, the maximum failure rate can be increased by 50×, as shown in Table 6.8. At a failure rate of 10^{-4}, 1.3% of 128-bit words (or 5% of 64-byte lines, or 27% of sets) are repaired with

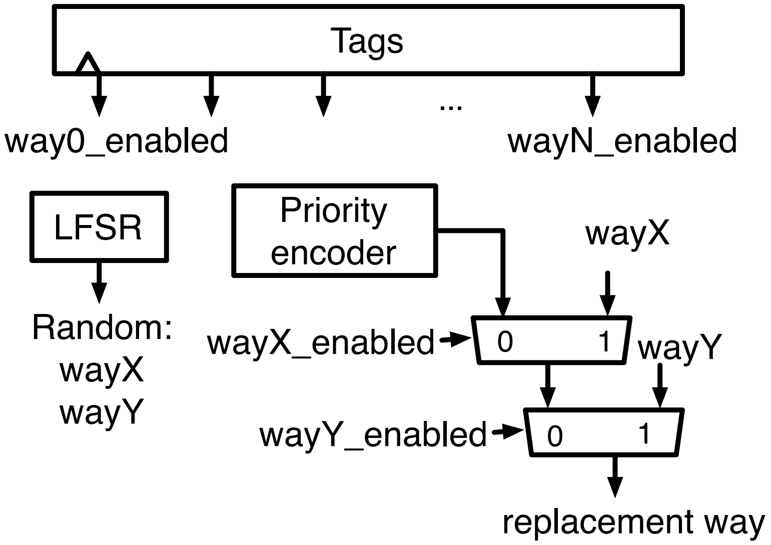


Figure 6.12: Pseudo-random algorithm to choose replacement way among remaining enabled ways.

Table 6.8: Effectiveness of DCR+LD at increasing the acceptable failure rate and reducing the minimum operating voltage.

Cache	Technique	Max. p_{bit}	V_{min}	Energy Reduction
L1 Inst. Cache Data	Nominal	7.1×10^{-9}	760mV	50%
	DCR+LD (proposed)	1.3×10^{-4}	540mV	
L1 Data Cache Data	Nominal	3.4×10^{-9}	775mV	51%
	DCR+LD (proposed)	1.3×10^{-4}	540mV	
L2 Data	Nominal	1.1×10^{-10}	850mV	59%
	DCR+LD (proposed)	9.8×10^{-5}	545mV	

DCR, and only 1% of lines are disabled. By allowing greater than 1% of lines to be disabled, the minimum operating voltage can be further reduced, as shown in Figure 5.7.

The area overhead is very small (one bit per way in the tag array), and the only timing overhead comes from increased complexity in the way-replacement algorithm of Figure 6.12.

6.4.4 Redundancy Programming Algorithm

The previously described techniques—BB, DCR, and LD—all rely on failure locations determined from BIST being appropriately programmed into redundancy entries before the processor starts. Figure 6.13 describes the redundancy programming algorithm. The algorithm

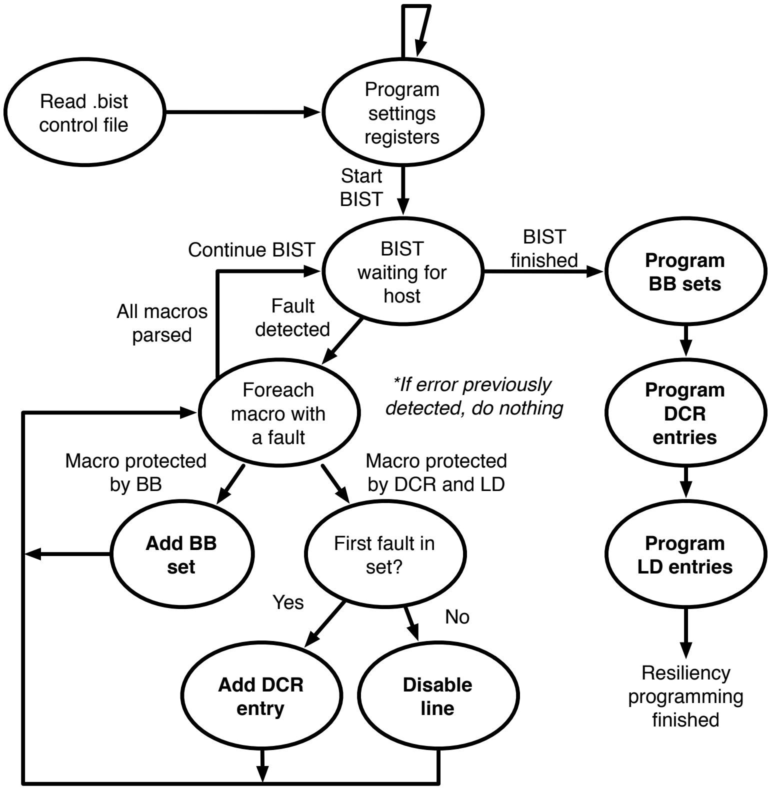


Figure 6.13: BIST failure locations are used to program dynamic redundancy.

is implemented in C and runs on the host machine. In future versions of this architecture, a small co-processor can run this algorithm from on-chip to improve testing speed and remove the need for an off-chip host.

The programming algorithm must be able to read existing redundancy data because decisions depend on previously detected errors. For example, when programming BB, a second error in the same word needs to be added to the existing repair set. When programming DCR, the line needs to be disabled if a second error occurs in the same line, and the first error could have come from a previous March step or an entirely different word address that happens to be in the same line. Because March tests are destructive, and the redundancy information itself is stored in SRAMs, redundancy information must be stored elsewhere during BIST and programming. Data structures in C hold all of the redundancy information. Checking previous errors only requires accessing local data structures. Therefore the entire BIST and programming algorithm is completed before programming all of the redundancy bits at once to the chip. Because the tag arrays are used to store DCR addresses and line disable bits, BB must be programmed first to ensure tag array functionality.

6.4.5 Storing Redundancy State

The programmed redundancy information stored for both DCR and BB exists in volatile memory (either SRAM or flip-flops). After the device is powered down, the information is lost. There are two main ways to ensure that the information persists between power losses—either store the bits in non-volatile memory, or reprogram on every power-up.

Reprogramming on every power-up adds time to the boot process. By testing every array in parallel, the testing time can be reduced to the time to run a March test on the deepest SRAM macro (based on the number of address bits). Even a large cache (4096 entries) and sophisticated test (March AB requires $22 \cdot n$ accesses) would only require approximately 90,000 cycles at 1GHz, or $90\mu s$. Additionally, re-testing can decrease margin against errors due to aging.

6.5 In-situ Error Correction and Detection

SWERVE enables further voltage scaling with minimal overhead by using dynamic redundancy (DCR and BB) to bypass cells with hard faults. Yet, intermittent SRAM errors, undetected by the BIST and therefore unprotected by dynamic redundancy, can cause system failure during operation.

To investigate the impact of intermittent errors, SECDED is added to every SRAM macro. This design uses a common SECDED code that can correct a single bit and detect double-bit failures with a reasonable area and timing overhead [86]. The main challenge of adding ECC is preventing the long-latency encoding and decoding operations from adding to the critical path of the design and reducing the maximum operating frequency. In general, this problem is solved by pipelining the decoding operation into the following cycle, while ensuring consumers of the data can occasionally handle longer latency accesses. Both ECC correction and dynamic redundancy can be disabled to allow independent evaluation of either technique. Implementation details of ECC differ between the tag and data arrays, and differ for the L1 instruction cache, the L1 data cache, and the L2 cache.

6.5.1 L1 Instruction Cache ECC

The pipeline diagram of the resiliency additions in the L1 instruction cache is shown in Figure 6.14. A SECDED code is used on both the tag and data arrays in the instruction cache. For the tag arrays, a separate code is computed for the DCR redundancy address and for each tag to avoid requiring a read-modify-write operating when updating a single tag. For the data arrays, a code is computed on the 128-bit access width.

Errors in either the tag or data array will result in an incorrect way comparison (either missing when there should be a hit, or hitting when there should be a miss). For any error, a refill is triggered. The correction capability of the SECDED code is not used, because there is no dirty data and therefore no need for in-situ correction latency to be on the critical path. If an error is persistent, refills will happen to random ways until the error is avoided. An error

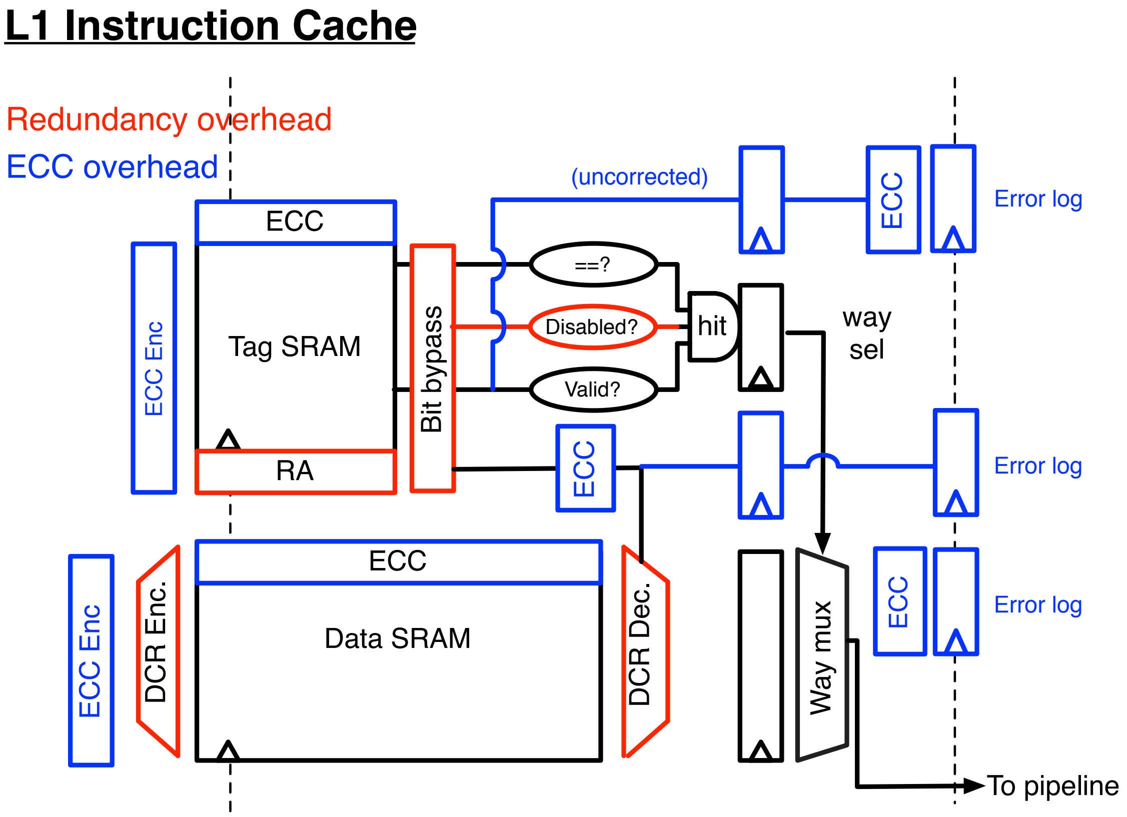


Figure 6.14: Implementation of DCR, BB, LD, and SECDED in the L1 instruction cache.

in every way will lock the design until the dynamic redundancy is reprogrammed to avoid the error. Future designs can forward directly from the L2 to the pipeline to avoid writing to a bad L1 word. However, a full SECDED code is still used for error logging purposes. If only parity was used, there would be no way to identify the error location.

Errors in the redundancy address of the tag array use the correction capability of the SECDED code, because correcting the RA doesn't appear on the critical path.

Writes only occur during refills. Before the data is written into the data arrays, it needs to be shifted based on the redundancy address in the tag array. During the memory request, the tags are read to find the disable bits and shift address for use during a refill. The redundancy shift is applied after ECC has been computed.

6.5.2 L1 Data Cache ECC

Figure 6.15 shows how resiliency is added to the L1 data cache. ECC encoding does not need to be pipelined, but ECC decoding is a long-latency operation and needs to be pipelined to prevent adding to the critical path. Because errors are very rare occurrences, correction is performed by recycling the operation through the same datapath by multiplexing in the corrected data at the output of the SRAM, to avoid adding one cycle of latency to every access. When an error occurs, the pipeline stalls, and the corrected data will be returned two cycles later.

ECC is performed on every tag independently. For the data array, ECC is performed after way selection to prevent requiring registers to store information from every way. In the case where there is an ECC error on the tags, way selection could be incorrect and hide an error in the correctly selected way. Therefore, tags are also recycled during data errors, and two recycle operations will be needed when errors exist in both the tag and data at the same time—the first recycle to repair the tags, and the second recycle to repair the data. The redundancy address is corrected immediately because it is not on the critical path. Extra registers are needed to ensure that back-to-back recycled instructions retire in their original order.

For soft errors, the corrected data can be written back to the cache when a failure is detected, but for hard faults, the failing bit will persist, so the corrected data needs to be stored in registers during the replay operation. Most of the implementation complexity is due to allowing for variable latency accesses to the data. There are many consumers of the SRAMs in the non-blocking data cache—the pipeline, MSHR, the probe unit, and the writeback unit. For the pipeline access, a nack is sent back to the pipeline to prevent incorrect forwarding operations and cause the pipeline to reissue the memory request. However, the reissued request is not guaranteed to be the next request through the cache pipeline, so additional registers flush the corrected information if an error occurs before the reissue.

6.5.3 L2 Cache ECC

Figure 6.16 shows the addition of ECC to the L2 cache tags and data. Because latency is less critical in the L2 cache, tag and data access is performed sequentially to avoid unnecessary reads of unselected ways. For tags, SRAM access is provided an entire cycle, and ECC decoding and way comparison occurs in a second cycle. For data, SRAM access is given an entire cycle, and DCR decoding and ECC decoding occurs in the next cycle. Serializing ECC avoids the extra registers required to recycle accesses as used in the latency-sensitive L1 caches, but the response always takes two cycles now instead of a single cycle.

6.5.4 Error Logging

Figures 6.17 and 6.18 describe the error logging architecture for the L1 and L2 cache respectively. When an error is detected, the syndrome (storing the column of the error), row address, way, and correct bit value is inserted into a 32-bit, 2-entry queue in parallel for every ECC decoding block in the design. An arbiter exposes a single error entry at a time to a mapped control status register (CSR) address. The host reads the error information from the CSR address, then sets the valid bit low to empty the entry from the buffer and expose the next error.

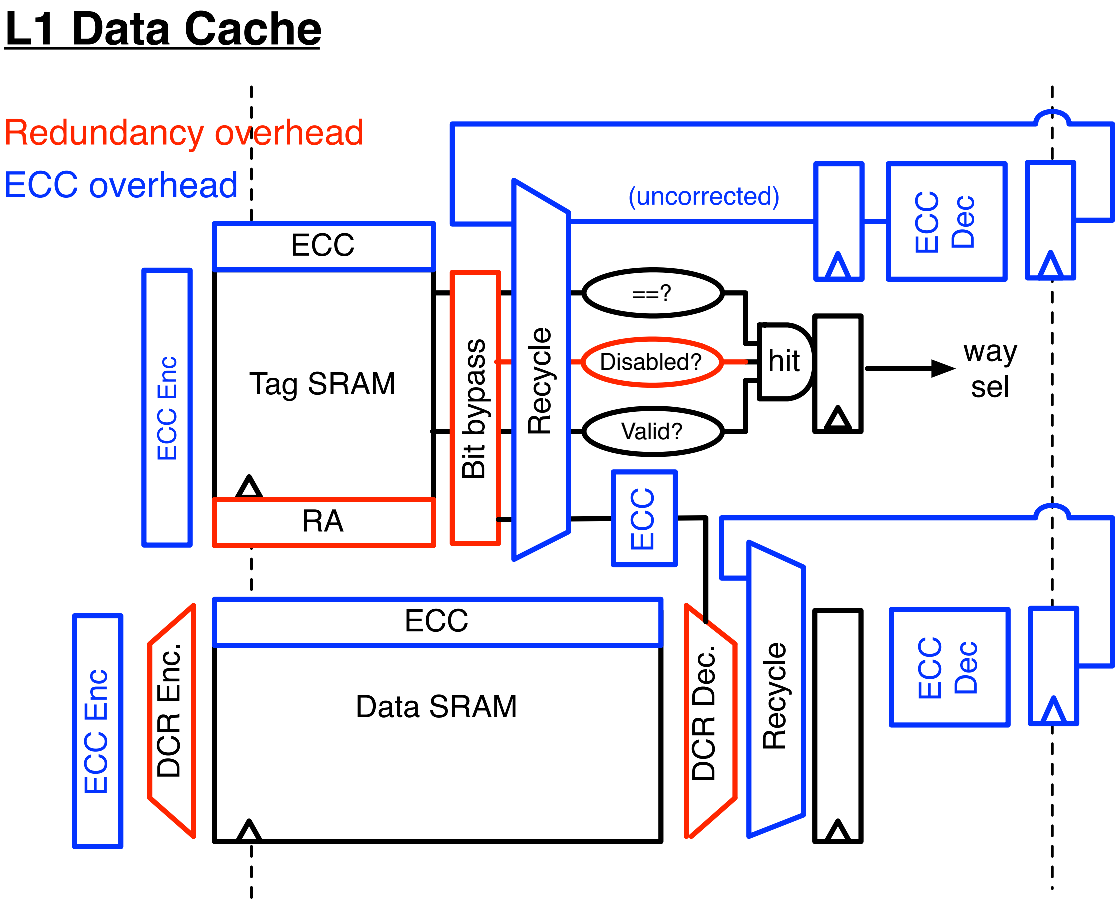


Figure 6.15: Implementation of DCR, BB, LD, and SECDED in the L1 data cache.

No feedback path from the error buffer to the processor exists; when the buffer is full, further errors will not be logged. Larger queues would allow more error history at the expense of area overhead. Future designs could use an on-chip co-processor and scratchpad memory to increase the maximum error logging rate.

Online programming

Online error programming refers to reprogramming dynamic redundancy during operation to avoid bit failures detected by ECC-based error logging. Due to the project deadline, online error programming was not implemented in the SWERVE chip. However, the following section describes a future possible implementation.

The first time a fault occurs at a specific bit, the fault is assumed to be a soft error, so the error location is logged and ECC is used to correct the error and continue operation. When multiple errors are detected as the same address, either BB, DCR, or LD need to be

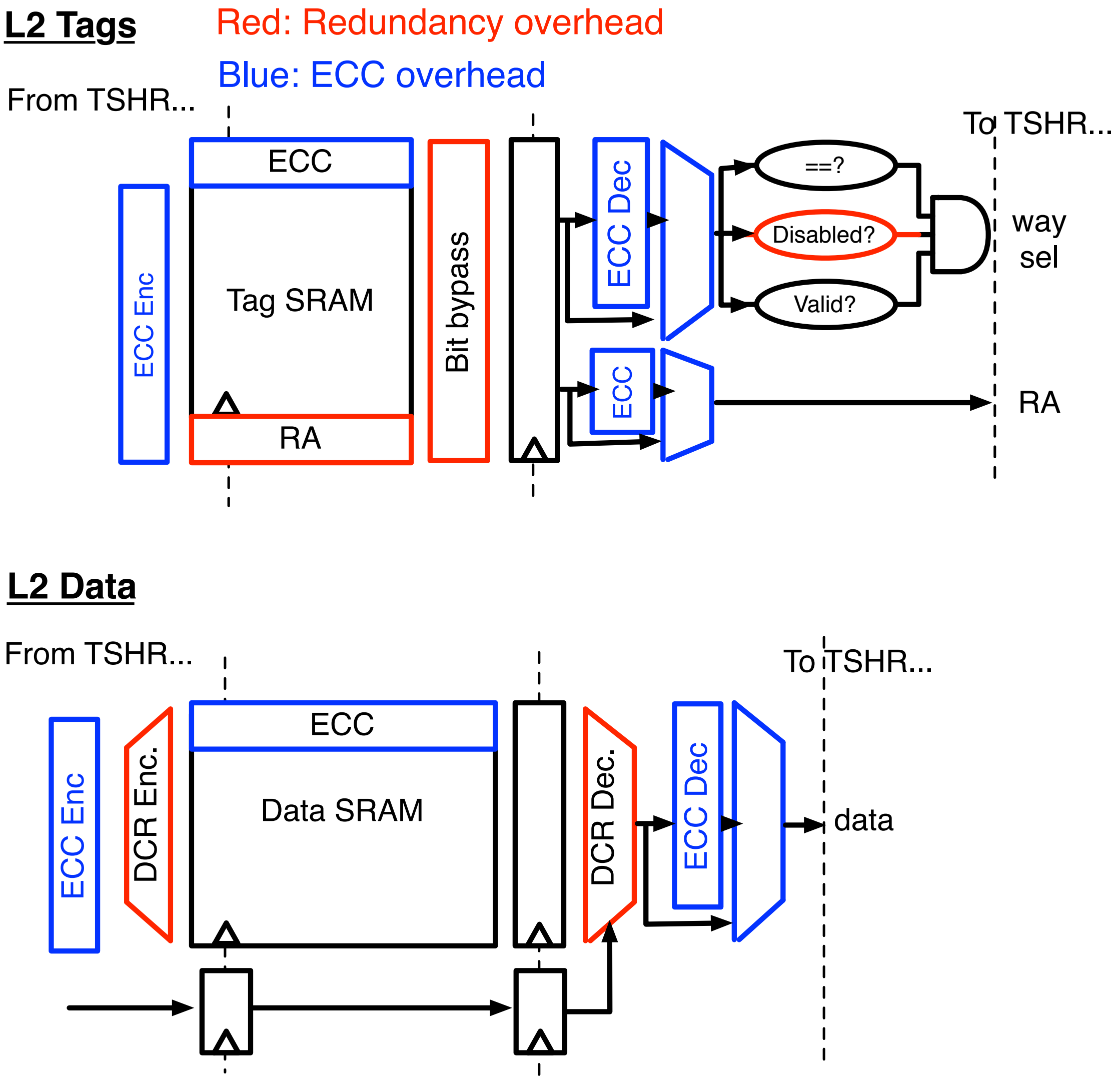


Figure 6.16: Implementation of DCR, BB, LD, and SECDED in the L2 cache.

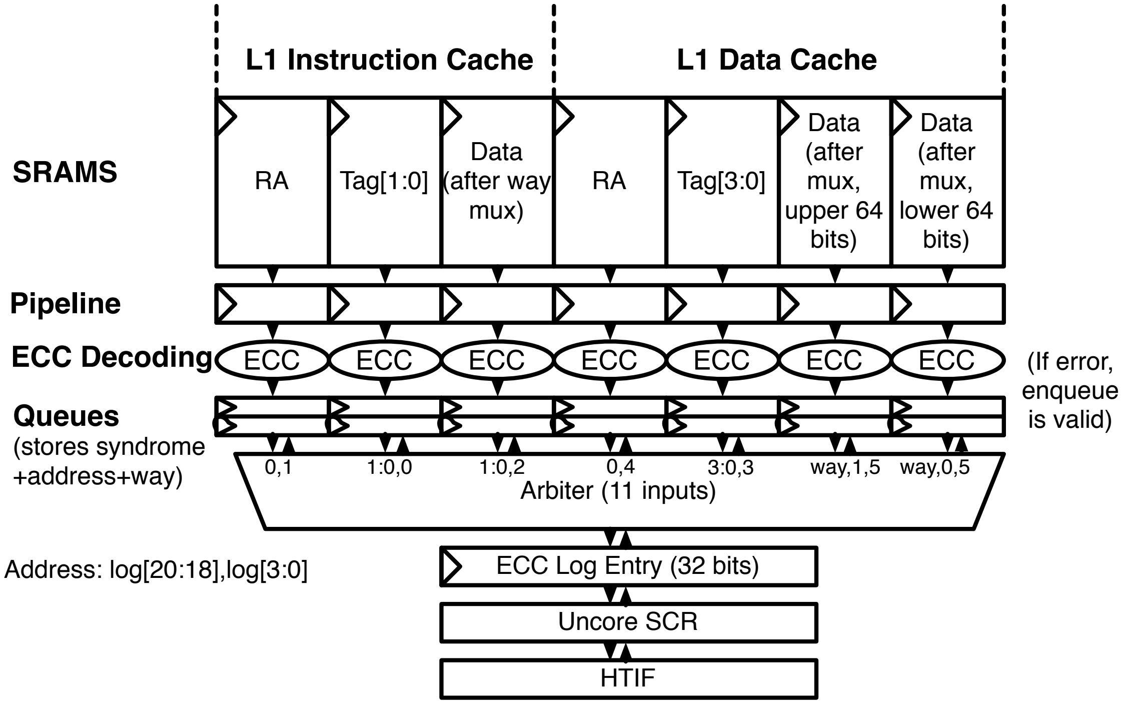


Figure 6.17: L1 cache error logging.

reprogrammed to avoid further accesses to that bitcell. The processor is stalled by holding the next instruction at instruction fetch stage until programming is finished.

For bit bypass, the correct value of the bit stored in the error log can be inserted into the reprogrammed redundancy set. However, online programming of DCR and line disable requires flushing the entire set and line respectively, because the physical location of the logic bits will change, and the data in the cache could be dirty. Once the entry is flushed through the memory system, DCR and LD can be safely programmed as that entry will be empty.

6.6 Simulation Results

The chip was taped out in April 2015, and the following section evaluates the design through simulation. Future measurements should confirm the simulation results.

Table 6.9 reports the critical path for the three major clock domains in the chip both in terms of frequency at the nominal operating corner and the technology-independent fanout-of-four (FO4) parameter. Design effort was focused on the core critical path, and there are further frequency improvements available in the L2 cache. The critical path in the core is caused by ECC logging, and the dynamic redundancy techniques are not on the critical path.

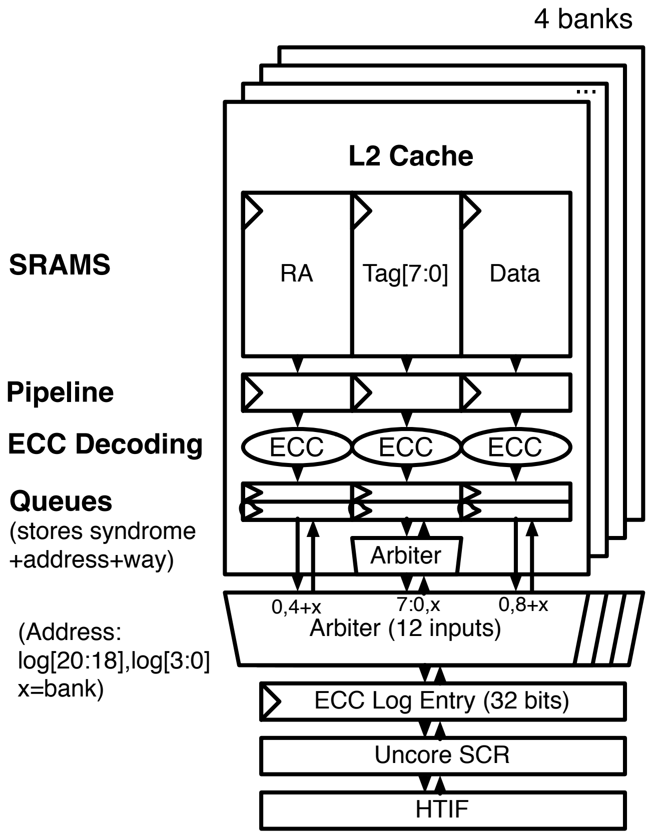


Figure 6.18: L2 cache error logging.

Table 6.9: Critical path in terms of time and FO4 delay in each clock domain.

Domain	FO4
Core	≈ 70
L2 Cache	≈ 110
Uncore	≈ 70

Table 6.10: Area of each voltage domain and contribution of SRAM arrays to overall area.

Domain	Dimensions	Area	SRAM Area	SRAM %
Core	775μm×650μm	525,000μm^2	280,000μm^2	53%
L2 Cache	1700μm×1550μm	2,635,000μm^2	Data: 1,985,000μm^2 Tags: 400,000μm^2	91%
Uncore	650μm×175μm	113,750μm^2	49,000μm	43%

Table 6.11: Cell statistics

Combination Cells	Sequential Cells	Total
324,341	58,423	382,764

Table 6.10 summarizes the area of each voltage domain in the design, as well as the contribution of SRAM to overall area. The L1 cache consumes half of the core area, while SRAMs in the L2 cache consume almost all of the area. Resiliency overhead calculated in previous sections report overhead based on only SRAM area, so overall area increase will be even smaller. Table 6.11 reports the number of standard cells used in the design.

Table 6.12 summarizes the QOR of the clock tree. The L2 cache covers a large area with many large macros restricting clock buffer placement, which explains the increased skew and insertion delay in this domain.

6.7 Summary

By adding DCR, BB, and LD, the SWERVE chip can reduce energy by around 50% through improved supply voltage scaling with less than 2% area overhead and minimal timing overhead. In-situ SECDED protection for every SRAM in the design enables analysis of intermittent and soft errors during runtime. BIST of each SRAM can report the slow of bitcell failure probability versus voltage, will increase the accuracy of resiliency scheme assumptions, and validate the architecture-level error model.

Table 6.12: Clock tree metrics for TT 0.9V 25C corner.

Clock domain	Sinks	Skew	Insertion delay
Core	19,522	129ps	338ps
Core BIST	3,507	66ps	297ps
L2 Cache	21,027	450ps	1220ps
L2 BIST	5,985	265ps	811ps
Uncore	5,831	70ps	215ps

Chapter 7

Conclusion

This work takes a holistic approach to improving energy efficiency through reducing the minimum operating voltage of SRAM with fault analysis and resiliency design techniques at both the circuit and architecture level. Assumptions are validated and proposed techniques are evaluated with implementations in a variety of 28nm testchips.

7.1 Summary of Contributions

The main contributions of this work are:

- A holistic analysis methodology and vocabulary that enables evaluation and comparison of resilient design techniques at different levels of abstraction (Chapter 2).
- A circuit-level error model that translates operating conditions and transistor variation to bitcell failure probability and failures-in-time (FIT), with short runtime and insight into failure mechanisms (Section 2.2 and 2.4).
- An architecture-level error model that translates bitcell failure probability to cache failure probability without relying on fixed circuit-level assumptions (Section 2.3 and 2.5).
- A comprehensive analysis of different SRAM assist techniques using dynamic failure metrics (Section 3.1).
- In-situ measurement of threshold shift and RTN to analyze of the joint effect of random variations and RTN on bitcell writeability failure (Section 3.2).
- A wide-operating-range custom SRAM macro that operates down to 0.45V (Section 4.3).
- An extensive study of different architecture-level resiliency techniques, and a set of design guidelines that offer insight into scheme effectiveness (Chapter 5).

- A processor with novel architecture-level resiliency features (DCR+LD and BB) to lower energy by around 50% with around 2% area overhead, and in-situ ECC that can quantify the effect of intermittent errors during program runtime (Chapter 6).

7.2 Future Work

While this work strove to develop a methodology, not stand-alone solutions, new research will need to continue as process nodes continue to advance to 14nm, 7nm, and further.

- While this work focuses on industry-standard 6T and 8T bitcells, new bitcells that use 7, 8, 9, or 10 transistors are frequently proposed in conferences to improve energy, delay, or V_{min}. While in general the area overhead of the extra transistors is not worth the benefit, a holistic approach to design could uncover a use-case for these bitcells.

- The circuit-level hard fault model relies on threshold shift as the only source of variability, and assumes transistor models are accurate under extreme threshold variation. More accurate design-time analysis would model each physical source of variability, and ensure transistors models were well calibrated for cells with large variation. As the number of variables increase, importance sampling may no longer be the best solution to speed up Monte Carlo analysis.

- The architecture-level soft error model should be extended to include a complete system, not just a cache. While cache-level analysis is general and useful, opportunities for masking soft errors at the architecture level will have a large impact on FIT.

- More realistic models of multi-bit soft error strikes are needed. Multi-bit soft error strikes have an enormous impact on system FIT, but all current publications don't differentiate between multi-bit strikes along the column (acceptable, because strikes occur to different words) and the row (bad, because a multi-bit fault could be caused).

- If more assumptions are made about the overall design architecture and activity, additional optimizations are possible. For example, the L1 cache could be made write-through as a means of error correction, but the effectiveness of this idea would be heavily dependent on workload assumptions. As long as workloads are well understood, and a good power and energy measurement methodology exists at design time, an entire new field of resiliency or energy-saving techniques can be explored.

- Most of this work has focused on reducing V_{min} to achieve the minimum operating point, however a different approach to saving energy, colloquially referred to as "race-to-halt" operates systems at higher voltages and finishes the workload quickly so that they entire system can go to sleep. The retention voltage of SRAM, a variant of read stability, can be analyzed with the proposed circuit-level hard fault model. A methodology that determines that cost of mode switching, and the proportion of time spent in sleep and high performance modes, is necessary to compare this to simply operating at the most energy efficient point.

Variation-induced SRAM failure will continue to be a critical problem limiting voltage scaling, and developing holistic solutions that leverage techniques at every level of abstraction is vital to improving energy efficiency in future digital systems.

Bibliography

[1] B. Zimmer, Y. Lee, A. Puggelli, J. Kwak, R. Jevtic, B. Keller, S. Bailey, M. Blagojevic, P.-F. Chiu, H.-P. Le, P.-H. Chen, N. Sutardja, R. Avizienis, A. Waterman, B. Richards, P. Flatresse, E. Alon, K. Asanovic, and B. Nikolic, “A RISC-V Vector Processor with Tightly-Integrated Switched-Capacitor DC-DC Converters in 28nm FDSOI,” in *VLSI Circuits (VLSIC), 2015 Symposium on*, 2015.

[2] O. Hirabayashi, A. Kawasumi, A. Suzuki, Y. Takeyama, K. Kushida, T. Sasaki, A. Katayama, G. Fukano, Y. Fujimura, T. Nakazato, *et al.*, “A process-variation-tolerant dual-power-supply sram with 0.179μm 2 cell in 40nm cmos using level-programmable wordline driver,” in *Solid-State Circuits Conference-Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pp. 458–459, IEEE, 2009.

[3] M. Yamaoka, K. Osada, and T. Kawahara, “A cell-activation-time controlled SRAM for low-voltage operation in DVFS SoCs using dynamic stability analysis,” in *Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European*, pp. 286–289, Sept 2008.

[4] S. O. Toh, Z. Guo, T.-J. Liu, and B. Nikolic, “Characterization of Dynamic SRAM Stability in 45 nm CMOS,” *IEEE J. Solid-State Circuits*, vol. 46, pp. 2702–2712, Nov. 2011.

[5] K. Kuhn, M. Giles, D. Becher, P. Kolar, A. Kornfeld, R. Kotlyar, S. Ma, A. Maheshwari, and S. Mudanai, “Process Technology Variation,” *Electron Devices, IEEE Transactions on*, vol. 58, pp. 2197–2208, Aug 2011.

[6] D. Reid, C. Millar, G. Roy, S. Roy, and A. Asenov, “Analysis of Threshold Voltage Distribution Due to Random Dopants: A 100000-Sample 3-D Simulation Study,” *Electron Devices, IEEE Transactions on*, vol. 56, pp. 2255–2263, Jan. 2009.

[7] E. Karl, Y. Wang, Y.-G. Ng, Z. Guo, F. Hamzaoglu, U. Bhattacharya, K. Zhang, K. Mistry, and M. Bohr, “A 4.6 GHz 162Mb SRAM design in 22nm tri-gate CMOS technology with integrated active V MIN-enhancing assist circuitry,” in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pp. 230–232, IEEE, 2012.

[8] M. Sinangil, H. Mair, and A. Chandrakasan, "A 28nm high-density 6T SRAM with optimized peripheral-assist circuits for operation down to 0.6V," in *Int. Solid-State Circuits Conf. Dig. Tech. Papers*, pp. 260–262, 2011.

[9] V. Ramadurai, H. Pilo, J. Andersen, G. Braceras, J. Gabric, D. Geise, S. Lamphier, and Y. Tan, "An 8 Mb SRAM in 45 nm SOI Featuring a Two-Stage Sensing Scheme and Dynamic Power Management," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 155–162, Jan 2009.

[10] S. Rusu, H. Muljono, D. Ayers, S. Tam, W. Chen, A. Martin, S. 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*, pp. 102–103, Feb 2014.

[11] D. A. Patterson and J. L. Hennessy, "Computer Organization and Design," *Morgan Kaufmann*, pp. 474–476, 2007.

[12] S.-L. Lu, A. Alameldeen, K. Bowman, Z. Chishti, C. Wilkerson, and W. Wu, "Architectural-level error-tolerant techniques for low supply voltage cache operation," in *IC Design Technology (ICICDT), 2011 IEEE International Conference on*, pp. 1–5, May 2011.

[13] R. W. Mann, J. Wang, S. Nalam, S. Khanna, G. Braceras, H. Pilo, and B. H. Calhoun, "Impact of circuit assist methods on margin and performance in 6T SRAM," *Solid State Electronics*, vol. 54, pp. 1398–1407, Nov. 2010.

[14] L. Dolecek, M. Qazi, D. Shah, and A. Chandrakasan, "Breaking the simulation barrier: SRAM evaluation through norm minimization," in *IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 322–329, 2008.

[15] M. Qazi, M. Tikekar, L. Dolecek, D. Shah, and A. Chandrakasan, "Loop flattening & spherical sampling: highly efficient model reduction techniques for SRAM yield analysis," in *Design, Automation & Test in Europe*, pp. 801–806, Mar. 2010.

[16] G. Chen, D. Sylvester, D. Blaauw, and T. Mudge, "Yield-Driven Near-Threshold SRAM Design," *IEEE Trans. VLSI Syst.*, vol. 18, pp. 1590–1598, Nov. 2010.

[17] X. Deng, W. K. Loh, B. Pious, T. W. Houston, L. Liu, B. Khan, and D. Corum, "Characterization of bit transistors in a functional SRAM," in *VLSIC*, pp. 44–45, 2008.

[18] P. Roche, J.-L. Autran, G. Gasiot, and D. Munteanu, "Technology downscaling worsening radiation effects in bulk: SOI to the rescue," in *IEEE International Electron Devices Meeting*, pp. 31.1.1–31.1.4, Dec 2013.

[19] H. Miki, N. Tega, M. Yamaoka, D. Frank, A. Bansal, M. Kobayashi, K. Cheng, C. D'Emic, Z. Ren, S. Wu, J. Yau, Y. Zhu, M. Guillorn, D. Park, W. Haensch, E. Leobandung, and K. Torii, "Statistical measurement of random telegraph noise and its impact in scaled-down high-κ/metal-gate MOSFETs," in *IEEE International Electron Devices Meeting*, pp. 19.1.1–19.1.4, 2012.

[20] L. Chang, Y. Nakamura, R. Montoye, J. Sawada, A. Martin, K. Kinoshita, F. Gebara, K. Agarwal, D. Acharyya, W. Haensch, *et al.*, "A 5.3 GHz 8T-SRAM with operation down to 0.41 V in 65nm CMOS," in *VLSI Circuits, 2007 IEEE Symposium on*, pp. 252–253, IEEE, 2007.

[21] K.-H. Koo, L. Wei, J. Keane, U. Bhattacharya, E. A. Karl, and K. Zhang, "A 0.094um^2 High Density and Aging Resilient 8T SRAM with 14nm FinFET Technology Featuring 560mV VMIN with Read and Write Assist," in *VLSI Circuits (VLSIC), 2015 Symposium on*, 2015.

[22] H. Fujiwara, L.-W. Wang, Y.-H. Chen, K.-C. Lin, D. Sun, S.-R. Wu, J.-J. Liaw, C.-Y. Lin, M.-C. Chiang, H.-J. Liao, S.-Y. Wu, and J. Chang, "A 64kb 16nm asynchronous disturb current free 2-port SRAM with PMOS pass-gates for FinFET technologies," in *Solid- State Circuits Conference - (ISSCC), 2015 IEEE International*, pp. 1–3, Feb 2015.

[23] M. Yabuuchi, K. Nii, Y. Tsukamoto, S. Ohbayashi, Y. Nakase, and H. Shinohara, "A 45nm 0.6V cross-point 8T SRAM with negative biased read/write assist," *IEEE Symp. VLSI Circuits Dig.*, 2009.

[24] H. Pilo, I. Arsovski, K. Batson, G. Braceras, J. Gabric, R. Houle, S. Lamphier, C. Radens, and A. Seferagic, "A 64 Mb SRAM in 32 nm High-k Metal-Gate SOI Technology With 0.7 V Operation Enabled by Stability, Write-Ability and Read-Ability Enhancements," *IEEE J. Solid-State Circuits*, vol. 47, pp. 97 –106, Jan. 2012.

[25] T. Mahmood, S. Kim, and S. Hong, "Macho: A failure model-oriented adaptive cache architecture to enable near-threshold voltage scaling," in *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 532–541, 2013.

[26] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu, "Trading off cache capacity for reliability to enable low voltage operation," in *Computer Architecture, 2008. ISCA'08. 35th International Symposium on*, pp. 203–214, IEEE, 2008.

[27] J. Abella, J. Carretero, P. Chaparro, X. Vera, and A. González, "Low Vccmin fault-tolerant cache with highly predictable performance," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pp. 111–121, 2009.

[28] Z. Chishti, A. R. Alameldeen, C. Wilkerson, W. Wu, and S.-L. Lu, "Improving cache lifetime reliability at ultra-low voltages," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pp. 89–99, 2009.

[29] A. R. Alameldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S.-L. Lu, "Energy-efficient cache design using variable-strength error-correcting codes," in *Proceedings of the 38th annual international symposium on Computer architecture*, ISCA '11, pp. 461–472, 2011.

[30] R. Naseer and J. Draper, "DEC ECC design to improve memory reliability in sub-100nm technologies," in *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, pp. 586–589, IEEE, 2008.

[31] N. S. Kim, S. C. Draper, S.-T. Zhou, S. Katariya, H. R. Ghasemi, and T. Park, "Analyzing the Impact of Joint Optimization of Cell Size, Redundancy, and ECC on Low-Voltage SRAM Array Total Area," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 12, pp. 2333–2337, 2012.

[32] S. S. Mukherjee, J. Emer, and S. K. Reinhardt, "The soft error problem: An architectural perspective," in *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pp. 243–247, IEEE, 2005.

[33] B. Zimmer, S. O. Toh, H. Vo, Y. Lee, O. Thomas, K. Asanovic, and B. Nikolic, "SRAM Assist Techniques for Operation in a Wide Voltage Range in 28-nm CMOS," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 59, no. 12, pp. 853–857, 2012.

[34] K. Agarwal and S. Nassif, "Characterizing process variation in nanometer CMOS," in *DAC '07: Proceedings of the 44th annual Design Automation Conference*, ACM Request Permissions, June 2007.

[35] A. Chavan, E. MacDonald, J. Neff, and E. Bozeman, "Radiation Hardened Flip-Flop Design for Super and Sub Threshold Voltage Operation," *Aerospace Conference*, pp. 1–6, 2011.

[36] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," *TNS*, vol. 47, no. 6, pp. 2586–2594, 2000.

[37] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *ACM SIGARCH Computer Architecture News*, vol. 32, p. 276, IEEE Computer Society, 2004.

[38] E. Ibe, H. Taniguchi, Y. Yahagi, K.-i. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule," *Electron Devices, IEEE Transactions on*, vol. 57, no. 7, pp. 1527–1538, 2010.

[39] S. M. Khan, A. R. Alameldeen, C. Wilkerson, J. Kulkarni, and D. A. Jimenez, "Improving multi-core performance using mixed-cell cache architecture," in *High Performance Computer Architecture (HPCA), 2013 IEEE 19th International Symposium on*, pp. 119–130, 2013.

[40] J. P. Kulkarni and K. Roy, "Ultralow-voltage process-variation-tolerant schmitt-trigger-based SRAM design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 319–332, 2012.

[41] S. Jain, S. Khare, S. Yada, V. Ambili, P. Salihundam, S. Ramani, S. Muthukumar, M. Srinivasan, A. Kumar, S. Gb, *et al.*, "A 280mV-to-1.2 V wide-operating-range IA-32 processor in 32nm CMOS," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pp. 66–68, IEEE, 2012.

[42] P. Packan, S. Akbar, M. Armstrong, D. Bergstrom, M. Brazier, H. Deshpande, K. Dev, G. Ding, T. Ghani, O. Golonzka, *et al.*, "High Performance 32nm Logic Technology Featuring 2 nd Generation High-k+ Metal Gate Transistors," in *IEEE International Electron Devices Meeting*, pp. 1–4, IEEE, 2009.

[43] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *Electron Devices, IEEE Transactions on*, vol. 53, no. 11, pp. 2816–2823, 2006.

[44] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pp. 469–480, IEEE, 2009.

[45] S. Nalam, V. Chandra, R. Aitken, and B. Calhoun, "Dynamic write limited minimum operating voltage for nanoscale SRAMs," in *Design, Automation & Test in Europe*, pp. 1–6, 2011.

[46] F. Arnaud, A. Thean, M. Eller, M. Lipinski, Y. Teh, M. Ostermayr, K. Kang, N. Kim, K. Ohuchi, J.-P. Han, D. Nair, J. Lian, S. Uchimura, S. Kohler, S. Miyaki, P. Ferreira, J.-H. Park, M. Hamaguchi, K. Miyashita, R. Augur, Q. Zhang, K. Strahrenberg, S. ElGhouli, J. Bonnouvrier, F. Matsuoka, R. Lindsay, J. Sudijono, F. Johnson, J. Ku, M. Sekine, A. Steegen, and R. Sampson, "Competitive and cost effective high-k based 28nm CMOS technology for low power applications," in *IEEE International Electron Devices Meeting*, pp. 1–4, Dec. 2009.

[47] A. Bhavnagarwala, S. Kosonocky, Y. Chan, K. Stawiasz, U. Srinivasan, S. Kowalczyk, and M. Ziegler, "A Sub-600mV, Fluctuation Tolerant 65nm CMOS SRAM Array with Dynamic Cell Biasing," *IEEE Symp. VLSI Circuits Dig.*, pp. 78–79, 2007.

[48] B. Zimmer, O. Thomas, S. O. Toh, T. Vincent, K. Asanovic, and B. Nikolic, "Joint impact of random variations and RTN on dynamic writeability in 28nm bulk and FDSOI SRAM," in *2014 44th European Solid State Device Research Conference (ESSDERC)*, pp. 98–101, 2014.

[49] S. O. Toh, *Nanoscale SRAM Variability and Optimization.* PhD thesis, University of California, Berkeley, 2011.

[50] N. Planes, O. Weber, V. Barral, S. Haendler, D. Noblet, D. Croain, M. Bocat, P. Sassoulas, X. Federspiel, A. Cros, A. Bajolet, E. Richard, B. Dumont, P. Perreau, D. Petit, D. Golanski, C. Fenouillet-Beranger, N. Guillot, M. Rafik, V. Huard, S. Puget, X. Montagner, M. A. Jaud, O. Rozeau, O. Saxod, F. Wacquant, F. Monsieur, D. Barge, L. Pinzelli, M. Mellier, F. Boeuf, F. Arnaud, and M. Haond, "28nm FDSOI technology platform for high-speed low-voltage digital applications," in *VLSIT*, pp. 133–134, 2012.

[51] N. Tega, H. Miki, Z. Ren, C. P. D. Emic, Y. L. Zhu, D. J. Frank, J. Cai, M. A. Guillorn, D.-G. G. Park, W. E. Haensch, and K. Torii, "Reduction of random telegraph noise in High- / metal-gate stacks for 22 nm generation FETs," in *Electron Devices Meeting (IEDM), 2009 IEEE International*, pp. 1–4, Jan. 2009.

[52] S. O. Toh, Y. Tsukamoto, Z. G. Guo, L. Jones, T.-J. King Liu, and B. Nikolic, "Impact of random telegraph signals on Vmin in 45nm SRAM," in *IEEE International Electron Devices Meeting*, pp. 1–4, 2009.

[53] J. Kulkarni, M. Khellah, J. Tschanz, B. Geuskens, R. Jain, S. Kim, and V. De, "Dual-VCC 8T-bitcell SRAM Array in 22nm tri-gate CMOS for energy-efficient operation across wide dynamic voltage range," in *VLSI Circuits (VLSIC), 2013 Symposium on*, pp. C126–C127, June 2013.

[54] M.-F. Chang, M.-P. Chen, L.-F. Chen, S.-M. Yang, Y.-J. Kuo, J.-J. Wu, H.-Y. Su, Y.-H. Chu, W.-C. Wu, T.-Y. Yang, and H. Yamauchi, "A Sub-0.3 V Area-Efficient L-Shaped 7T SRAM With Read Bitline Swing Expansion Schemes Based on Boosted Read-Bitline, Asymmetric-Vth Read-Port, and Offset Cell VDD Biasing Techniques," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 2558–2569, Oct 2013.

[55] E. Karl, Z. Guo, J. Conary, J. Miller, Y.-G. Ng, S. Nalam, D. Kim, J. Keane, U. Bhattacharya, and K. Zhang, "A 0.6V 1.5GHz 84Mb SRAM design in 14nm FinFET CMOS technology," in *Solid- State Circuits Conference - (ISSCC), 2015 IEEE International*, pp. 1–3, Feb 2015.

[56] M. Yabuuchi, Y. Tsukamoto, M. Morimoto, M. Tanaka, and K. Nii, "20nm High-density single-port and dual-port SRAMs with wordline-voltage-adjustment system for read-/write assists," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pp. 234–235, Feb 2014.

[57] Y. H. Chen, G. Chan, S. Y. Chou, H.-Y. Pan, J.-J. Wu, R. Lee, H. Liao, and H. Yamauchi, "A 0.6 V dual-rail compiler SRAM design on 45 nm CMOS technology with adaptive SRAM power for lower VDD_min VLSIs," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1209–1215, 2009.

[58] O. Thomas, B. Zimmer, S. O. Toh, L. Ciampolini, N. Planes, R. Ranica, P. Flatresse, and B. Nikolic, "Dynamic single-p-well SRAM bitcell characterization with back-bias adjustment for optimized wide-voltage-range SRAM operation in 28nm UTBB FD-SOI," in *IEEE International Electron Devices Meeting*, pp. 3.4.1–3.4.4, 2014.

[59] R. Ranica, N. Planes, O. Weber, O. Thomas, S. Haendler, D. Noblet, D. Croain, C. Gardin, and F. Arnaud, "FDSOI process/design full solutions for ultra low leakage, high speed and low voltage SRAMs," in *VLSI Technology (VLSIT), 2013 Symposium on*, pp. T210–T211, June 2013.

[60] M.-F. Chang, C.-F. Chen, T.-H. Chang, C.-C. Shuai, Y.-Y. Wang, and H. Yamauchi, "A 28nm 256kb 6T-SRAM with 280mV improvement in VMIN using a dual-split-control assist scheme," in *Solid- State Circuits Conference - (ISSCC), 2015 IEEE International*, pp. 1–3, Feb 2015.

[61] T. Song, W. Rim, J. Jung, G. Yang, J. Park, S. Park, K.-H. Baek, S. Baek, S.-K. Oh, J. Jung, S. Kim, G. Kim, J. Kim, Y. Lee, K. S. Kim, S.-P. Sim, J. S. Yoon, and K.-M. Choi, "A 14nm FinFET 128Mb 6T SRAM with VMIN-enhancement techniques for low-power applications," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pp. 232–233, Feb 2014.

[62] Y.-H. Chen, W.-M. Chan, W.-C. Wu, H.-J. Liao, K.-H. Pan, J.-J. Liaw, T.-H. Chung, Q. Li, G. Chang, C.-Y. Lin, M.-C. Chiang, S.-Y. Wu, S. Natarajan, and J. Chang, "A 16nm 128Mb SRAM in high-K metal-gate FinFET technology with write-assist circuitry for low-VMIN applications," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pp. 238–239, Feb 2014.

[63] S. Tanaka, Y. Ishii, M. Yabuuchi, T. Sano, K. Tanaka, Y. Tsukamoto, K. Nii, and H. Sato, "A 512-kb 1-GHz 28-nm partially write-assisted dual-port SRAM with self-adjustable negative bias bitline," in *VLSI Circuits Digest of Technical Papers, 2014 Symposium on*, pp. 1–2, June 2014.

[64] B. S. Amrutur, M. Horowitz, *et al.*, "A replica technique for wordline and sense control in low-power SRAM's," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 8, pp. 1208–1219, 1998.

[65] A. Kawasumi, Y. Takeyama, O. Hirabayashi, K. Kushida, F. Tachibana, Y. Niki, S. Sasaki, and T. Yabe, "A 47timing-generation scheme utilizing a statistical method for ultra low voltage SRAMs," in *VLSI Circuits (VLSIC), 2012 Symposium on*, pp. 100–101, June 2012.

[66] 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 Journal of Solid-State Circuits*, vol. 46, pp. 2545–2551, Nov 2011.

[67] I. Arsovski, T. Hebig, J. Goss, P. Grzymkowski, and J. Patch, "Tail-Bit Tracking circuit with degraded VGS bit-cell mimic array for a 50and 200mV Vmin improvement in a Ternary Content Addressable Memory," in *Custom Integrated Circuits Conference (CICC), 2013 IEEE*, pp. 1–4, Sept 2013.

[68] J. Chang, M. Huang, J. Shoemaker, J. Benoit, S.-L. Chen, W. Chen, S. 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 Journal of Solid-State Circuits*, vol. 42, pp. 846–852, April 2007.

[69] A. Ansari, S. Feng, S. Gupta, and S. Mahlke, "Archipelago: A polymorphic cache design for enabling robust near-threshold operation," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pp. 539–550, IEEE, 2011.

[70] T. N. Miller, R. Thomas, J. Dinan, B. Adcock, and R. Teodorescu, "Parichute: Generalized turbocode-based error correction for near-threshold caches," in *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 351–362, IEEE Computer Society, 2010.

[71] M. Zhang, V. M. Stojanovic, and P. Ampadu, "Reliable Ultra-Low-Voltage Cache Design for Many-Core Systems," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 59, no. 12, pp. 858–862, 2012.

[72] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 141–152, ACM, 2010.

[73] M. Huang, M. Mehalel, R. Arvapalli, and S. He, "An Energy Efficient 32-nm 20-MB Shared On-Die L3 Cache for Intel® Xeon® Processor E5 Family," 2013.

[74] M.-Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM Journal of Research and Development*, vol. 14, no. 4, pp. 395–401, 1970.

[75] B. Arnold, N. Balakrishnan, and H. Nagaraja, *Order Statistics from Some Specific Distributions*, ch. 4, p. 68. Society for Industrial and Applied Mathematics, 2008.

[76] H. Nho, P. Kolar, F. Hamzaoglu, Y. Wang, E. Karl, Y.-G. Ng, U. Bhattacharya, and K. Zhang, "A 32nm High-k metal gate SRAM with adaptive dynamic stability enhancement for low-voltage operation," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pp. 346–347, Jan. 2010.

[77] S. Sawant, U. Desai, G. Shamanna, L. Sharma, M. Ranade, A. Agarwal, S. Dakshinamurthy, and R. Narayanan, “A 32nm Westmere-EX Xeon enterprise processor,” in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pp. 74–75, Feb 2011.

[78] S. Damaraju, V. George, S. Jahagirdar, T. Khondker, R. Milstrey, S. Sarkar, S. Siers, I. Stolero, and A. Subbiah, “A 22nm IA multi-CPU and GPU System-on-Chip,” in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pp. 56–57, Jan. 2012.

[79] S. Jahagirdar, V. George, I. Sodhi, and R. Wells, “Power Management of the Third Generation Intel Core Micro Architecture formerly codenamed Ivy Bridge,” Presented at the 24th Hot Chips Symposium, 2012.

[80] Y. Lee, A. Waterman, R. Avizienis, H. Cook, C. Sun, V. Stojanovic, and K. Asanovic, “A 45nm 1.3GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators,” in *European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014 - 40th*, pp. 199–202, Sept 2014.

[81] UC Berkeley, “Rocket chip generator,” 2015. https://github.com/ucb-bar/rocket-chip.git.

[82] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avizienis, J. Wawrzynek, and K. A. , “Chisel: Constructing Hardware in a Scala Embedded Language,” in *Design Automation Conference (DAC)*, 2012.

[83] Synopsys.com, “DesignWare STAR Memory System,” 2015.

[84] A. J. van de Goor, “Using march tests to test SRAMs,” *Design & Test of Computers, IEEE*, vol. 10, pp. 8–14, Jan. 1993.

[85] M. Linder, A. Eder, U. Schlichtmann, and K. Oberlander, “An Analysis of Industrial SRAM Test Results - A Comprehensive Study on Effectiveness and Classification of March Test Algorithms,” *Design & Test, IEEE*, p. 1, Jan. 2013.

[86] R. W. Hamming, “Error detecting and error correcting codes,” *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.