Resilient Design Methodology for Energy-Efficient SRAM

by Brian Zimmer

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor B. Nikolić
Research Advisor

Date

* * * * * *

Professor K. Asanović
Research Advisor

Date

Resilient Design Methodology for Energy-Efficient SRAM

Brian Zimmer
bmzimmer@eecs.berkeley.edu

May 16, 2012

Abstract

Increasing process variability limits energy reduction in SRAM design by increasing the need for margining and preventing optimal supply voltage scaling. However, tolerating variability with resilient designs can prevent these limitations and enable future energy-efficiency improvements. Understanding resiliency requires understanding how design decisions affect error rates—therefore we propose a unified analytical framework for SRAM design that uses importance sampling of dynamic failure metrics to quantify the effect of different assist techniques, array organization, and timing on failure rates of a 28nm arrays at design-time. Dynamic voltage and frequency scaling (DVFS) systems show great potential to improve energy-efficiency, but require designs that are both operational and efficient over a wide range of supply voltages. We propose replica circuits that adapt array settings based on corners and supply voltage to optimize energy-efficiency over the entire supply voltage range. Last, we extend our results to improve micro-architectural design of an on-chip cache system.

Contents

1 Introduction 5

2 Background 6

2.1 SRAM Operation 6

2.1.1 6T Cell 6

2.1.2 8T Cell 9

2.2 SRAM Array Organization 9

2.2.1 6T Array 9

2.2.2 8T Array 10

2.3 SRAM Failure 11

2.4 Static vs. Dynamic Metrics 12

2.5 Readability Failure 13

2.6 Read Stability Failure 13

2.7 Writeability Failure 13

2.8 Hold Failure 13

2.9 Leakage 13

2.10 Other 14

3 Analysis Methodology 15

3.1 Importance Sampling 15

3.1.1 Definition 15

3.1.2 Algorithm 17

3.1.3 Implementation 19

3.1.4 Usage 19

3.1.5 Example: Analyzing results 20

3.1.6 Verification 21

3.2 SRAM Failure Analysis 21

3.3 Dynamic Failure Metrics Test Chip . . . 22
3.3.1 6T . . . 22
3.3.2 8T . . . 24

4 SRAM Vmin Analysis 26

4.1 Effect of bitline capacitance on 6T SRAM failures . . . 26
4.2 Effect of bitline capacitance on 8T SRAM failures . . . 27
4.3 Effect of clock period on 6T SRAM failures . . . 27
4.4 Effect of process corners . . . 28
4.5 Effect of sense amplifier offset on readability failure . . . 29
4.6 Effect of assist techniques on 6T SRAM failures . . . 29
4.6.1 Effect of wordline boost as a readability and writeability assist . . . 32
4.6.2 Effect of wordline droop as a read stability assist . . . 34
4.6.3 Effect of cell Vdd boost as a readability and read stability assist . . . 34
4.6.4 Effect of cell Vdd collapse as a writeability assist . . . 34
4.6.5 Effect of cell GND boost as a writeability assist . . . 34
4.6.6 Effect of negative bitline as a writeability assist . . . 35
4.6.7 Effect of partial bitline precharge as a stability assist . . . 35
4.6.8 Effect of negative cell GND as a readability assist . . . 35
4.6.9 Effect of other assist techniques . . . 36
4.7 Effect of assist techniques on 8T SRAMs . . . 37
4.7.1 Leakage . . . 37
4.8 Effect of hierarchical bitlines . . . 37
4.9 Effect of ECC . . . 37
4.9.1 (9,8) Parity Detection . . . 38
4.9.2 ECC: (72,64) SEC-DED . . . 39
4.10 Comparison of Scheme Effectiveness . . . 40
4.11 Effect of Cell Sizing . . . 41

4.12 6T Overall Failure Minimization . . . 41
4.13 8T Overall Failure Minimization . . . 42

5 SRAM Emin Analysis 43

5.1 Overview . . . 43
5.2 Emin . . . 43
5.3 Energy measurement . . . 44
5.3.1 L1 Cache Data Array . . . 44
5.3.2 L2 Cache Data Array . . . 44
5.4 Energy per read operation . . . 45
5.4.1 Nominal . . . 45
5.5 Optimal read timing . . . 45
5.6 Generating optimal read timing . . . 47
5.7 Hierarchical vs. timing . . . 48
5.8 Energy per write operation . . . 48
5.8.1 Optimal write timing . . . 49
5.8.2 Generating optimal write timing . . . 49
5.9 Energy per inactive cycle . . . 50
5.10 Assist Overhead . . . 50
5.11 Error Monitoring and Response Cost . . . 51
5.11.1 BIST . . . 51
5.11.2 ECC . . . 51
5.12 Adaptivity . . . 51

6 Optimizing SRAM for caches 53

6.1 Necessity of Error Correction . . . 53
6.2 L1 Correction Methods . . . 53
6.3 System Assumptions . . . 54
6.3.1 ECC Schemes . . . 55

6.4 Cache Simulator 55

6.4.1 Write-through 56

6.4.2 Write-back 56

6.5 Design Intuition 56

6.6 Write-through 57

6.7 Write-back 57

6.8 Sensitivity of model to assumptions 58

6.9 Effect of voltage scaling 58

6.10 Effect of performance on energy 60

6.10.1 6T vs. 8T 62

6.10.2 Summary of aggressive scaling in caches 62

7 SWAAT: SRAM with Adaptive Assist Techniques 63

7.1 Methods to reduce Vmin 63

7.2 Methods to reduce energy 64

7.3 Methods to detect errors 64

7.3.1 Methods to respond to errors 65

8 Conclusion 66

1 Introduction

Embedded memories are critical in CMOS integrated-circuit designs. Different memory options can be best differentiated by their speed, energy, and memory density characteristics. SRAM fills a sweet-spot between DRAM and flip-flops, and have moderate speed, density and energy. Improving all of these metrics in SRAM enables larger amounts of faster on-chip memory, which is particularly important for providing new opportunities in mobile devices. We investigate the effect of various design decisions on all three primary design metrics—speed, energy, and area density—with a specific focus on energy efficiency. Because many integrated circuits contain large SRAMs with many cells, failures caused by manufacturing variability become common, and a fourth metric—resiliency—must be considered. Through better understanding of failure cases and self-monitoring SRAM design, we can both remove the margins currently required to maintain error-free operation, and understand how design decisions can improve energy efficiency. We propose a methodology to quantify resiliency and use this to perform an extensive design space exploration of SRAM.

Previous works have investigated the various pieces necessary to understand how errors affect SRAM energy-efficiency, but none have combined the results to form a cohesive conclusion. Importance sampling has been used to estimate SRAM stability [1]. Recently, various types of assist circuitry have been proposed to enable low voltage operation [2], [3], [4], [5]. A comprehensive analysis comparing many different assist techniques has been performed, but used only static metrics [6]. The effect of error correction on cache energy has been studied, but does not account for correction cost and neglects details that have significant impact on final results [7].

Section 2 introduces SRAM design, discusses operation details, and defines metrics that measure SRAM failure.

Section 3 explains our resiliency measurement scheme, which allows for accurate analysis of rare failure events.

Using this resiliency measurement scheme, Section 4 examines how assist techniques, array organization, and error correction affect bit-error rate (BER) and minimum operating voltage (Vmin).

Section 5 extends our analysis to incorporate energy, and discusses possible methods to improve average energy per operation, and absolute minimum energy per operation (Emin).

Using the methodology and modeling intuition developed, Section 6 serves as a case study to analyze how supply scaling and finite error rates affect cache design for a dynamic voltage and frquency scaling (DVFS) processor.

Finally, we use the results of our design exploration to propose a SRAM design that optimizes energy efficiency in Section 7. We optimize for use in a DVFS system, which requires dynamic configuration to obtain maximum energy efficiency over a wide range of possible supply voltages.

2 Background

2.1 SRAM Operation

There are many different ways to design an SRAM cell, but almost every type of cell will always have a pair of cross-coupled inverters (I1/I2) that act as a storage element—holding both the true and complementary value of the data on two different nodes (Q/Q'), as shown in Figure 1.

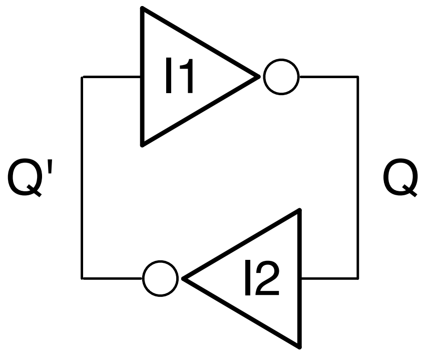


Figure 1: Two inverters (I1/I2) form the storage element in an SRAM cell.

As long as the supply voltage remains high enough, any voltage change on either node will be counteracted by the feedback and tend to return to the original value. For example, if Q is at VDD and Q' is at 0, then any positive charge deposited on Q' will be discharged through the NMOS in I2. This feedback makes SRAM static, as opposed to other types of memory such as DRAM which rely on charge held on a floating capacitor.

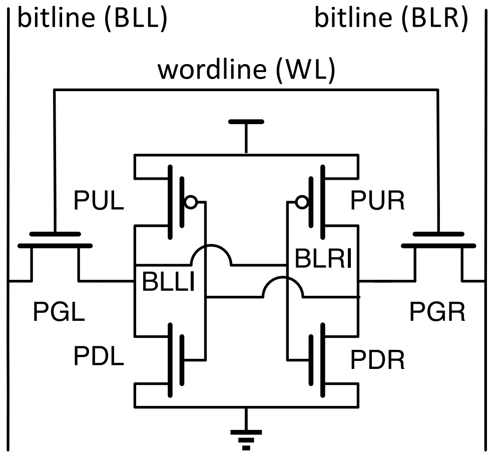
Cell designs differ by how many devices provide access to these internal nodes, and are typically named by the total number of transistors in the cell. This paper will focus on the two most popular flavors–the 6T cell and the 8T cell.

2.1.1 6T Cell

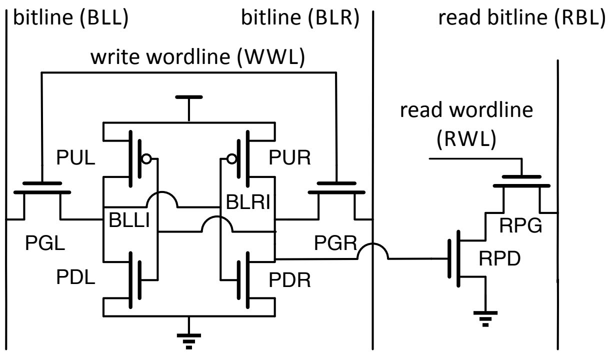
The 6T SRAM cell is shown in Figure 2a. Pull-up-left (PUL), pull-down-left (PDL), pull-up-right (PUR), and pull-down-right (PDR) together form the cross-coupled inverter pair. Two extra NMOS devices, pass-gate-left (PGL) and pass-gate-right (PGR), provide access to the internal nodes from two shared bitlines, bit-line-left (BLL) and bit-line-right (BLR).

Figure 3a shows a read operation. During the read operation, both bitlines are precharged high, and then left to float. Their voltage is held by the wire capacitance of the bitlines combined with the diffusion capacitance of PGR or PGL. The gate of PGL and PGR will be turned on by driving the wordline high. Assuming BLLI is high and BLRI is low, then the gates of PDR and PGR will both be high, and together they will form a conductive path to decrease the voltage of BLR. External circuitry can then respond to the voltage difference between BLL and BLR to output the value of the cell if the cell established a large enough voltage difference by the end of the cycle. The voltage at BLRI will be set by the relative strengths of PDR and PGR, and if this node jumps above the switching threshold of the PUL and PDL inverter, the value of the cell could accidentally flip, as shown in Figure 3b.

Figure 3c shows a write operation. During the write operation, one bitline is precharged high, while the other bitline is held at 0 through NMOS devices connected to ground. The pass-gate transistors PGL and PGR are turned on by raising the wordline voltage to VDD. Depending on the relative strengths of the NMOS pass-gate and PMOS pull-up, the internal node on the side where the bitline is at 0 should get pulled below the switching threshold of the inverter driving the other side of the memory cell, and positive feedback will flip the value stored in the cell. If the pass gate isn't sufficiently stronger than the pull-up device, then this node will not decrease enough to flip the value of the cell.

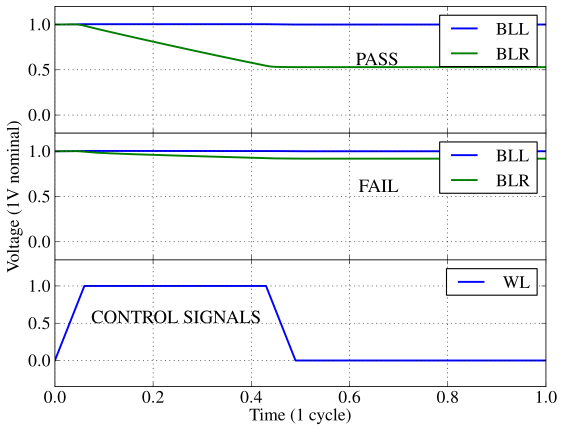


(a) Schematic of a six-transistor (6T) SRAM cell

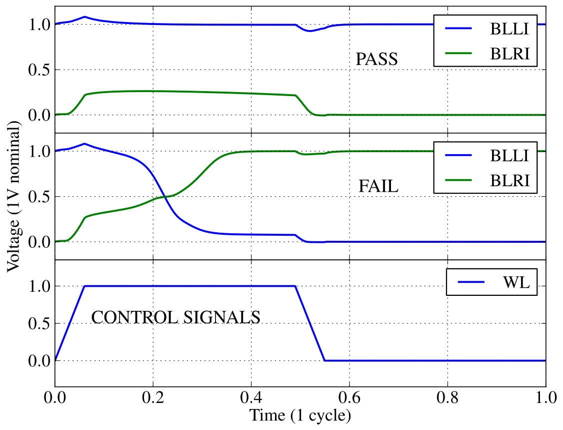


(b) Schematic of an eight-transistor (8T) SRAM cell

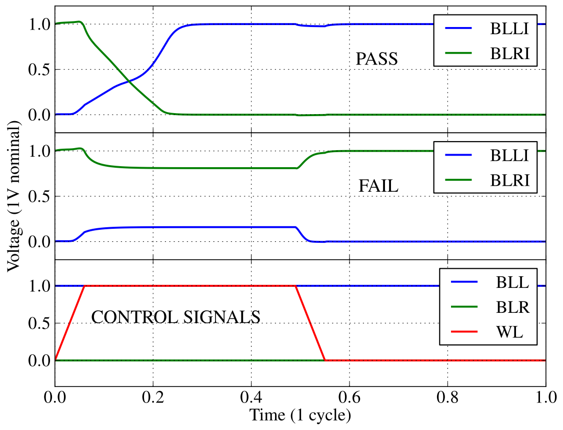
Figure 2: SRAM Cell Schematics



(a) Example of SRAM readability.



(b) Example of SRAM read stability.



(c) Example of SRAM writeability.

Figure 3: SRAM operation waveforms.

2.1.2 8T Cell

The 8T SRAM cell is shown in Figure 2b, and is identical to the 6T cell except for the addition of two extra devices, read-pull-down (RPD) and read-pass-gate (RPG). Write operations are performed in exactly the same way as in the 6T cell by turning on the write wordline. However, reads are done by turning on the read wordline (RWL) and leaving the write wordline (WWL) off. If BLRI is high, then RPD and RPG will both have a high voltage at their gates, and will discharge the read bitline. A skewed inverter connected to the read bitline can recognize this voltage drop and determine the correct read value. There are a few advantages to this approach. First, reads cannot upset the cell as RPD is decoupled from BLRI through a gate. Because of this, the pull down devices can be made minimum size because they don't need to be stronger than the pass gates to prevent a read upset. Second, the additional transistor allows a simultaneous read and write operation to the cell because the read and write bitlines are independent and are controlled by separate wordlines. Third, reads are no longer differential, whenever BLTI stores a 0, read bitline will not change voltage and therefore will not consume energy. When used as a cache, the realization that the majority of bits stored will be 0 allows the 8T SRAM to save energy during read operations [8].

2.2 SRAM Array Organization

SRAM cells are organized into arrays in order to share control circuitry. However, their organization will impact overall SRAM effectiveness. For example, when more cells share a bitline, there is a larger bitline capacitance that a pull-down transistor must discharge.

2.2.1 6T Array

Figure 4c shows a small 8 by 8 array with 64 total cells. All cells on the same row share the same wordline (WL) wire. All cells on the same column share the same bitline (BLL and BLR) wires. Often the number of entries will be much larger than the word size, which would result in many rows (and very long bitlines) and few columns (and very short wordlines). To keep prevent excessive bitline capacitances, multiple columns are interleaved together. In this example, BLL[0] is connected to BLL for both cells in the column of C1 and C2 through a multiplexer. To read cell C1, the wordline will still be on for cell C2, in a state known as "half-write". For reads, every column's bitlines will experience a read due to an accessed cell on the wordline, but only one column will be connected to a sense amplifier through the multiplexer. In this case, the select signal for the multiplexer will select between the value in C1 and C2. For writes, the bitlines will be forced through the multiplexer (or with column-specific circuitry), but all other cells on the activated wordline will be unintentionally reading.

Figure 4a shows the transistor layout for a given cell, where poly is red, NMOS active is green, and PMOS active is brown. Note that the pull-down devices are larger than the pass gates (to ensure read stability) and the pull-up devices are smaller than the pass gates (to ensure write-ability).

Figure 4b shows the metal layers that connect to each cell. The direction of metal wires will determine side effects for assist techniques discussed later. VDD runs vertically along a column, and GND is supplied from different wires for the left and right sides and is shared with the rows above and below.

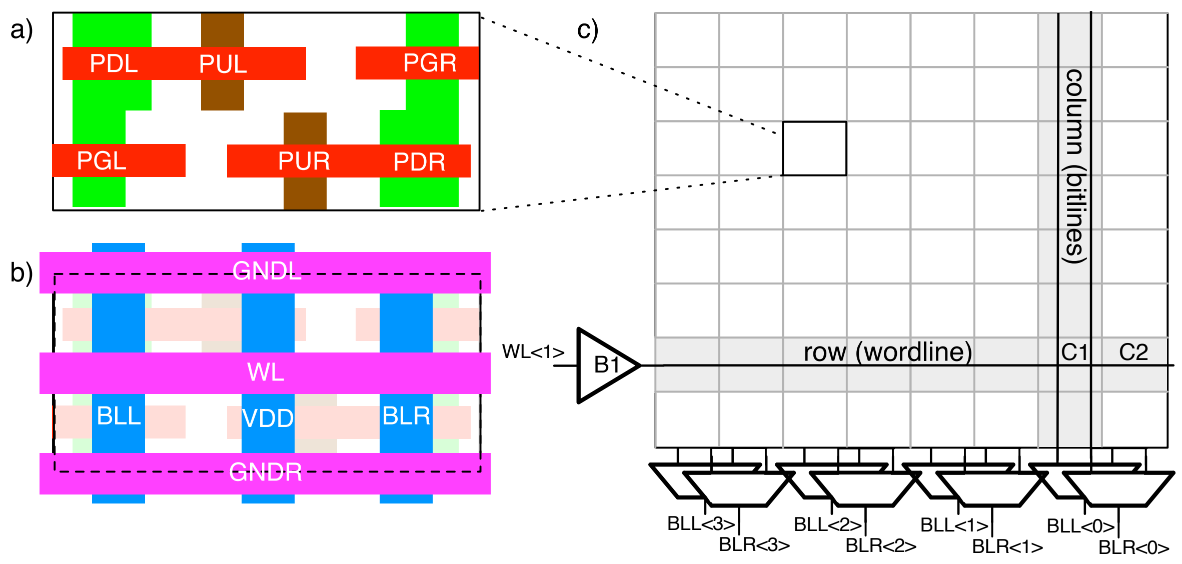


Figure 4: (a) Layout of transistors in a 6T SRAM cell. (b) Layout of metal layers for a 6T SRAM cell. (c) Generic overview of SRAM array design.

2.2.2 8T Array

8T arrays are organized in the same manner as 6T arrays. Figure 5a shows transistor layout for an 8T cell. The six transistors in common with the 6T array have not changed, and the two additional devices are added to the right side of the device.

Figure 5b shows a very different wiring scheme than the 6T cell. Now that each row needs two separate wordlines, the horizontal GND wires have been moved to become vertical in the extra space provided by the wider cell. Also, the extra read bitline needed is routed vertically.

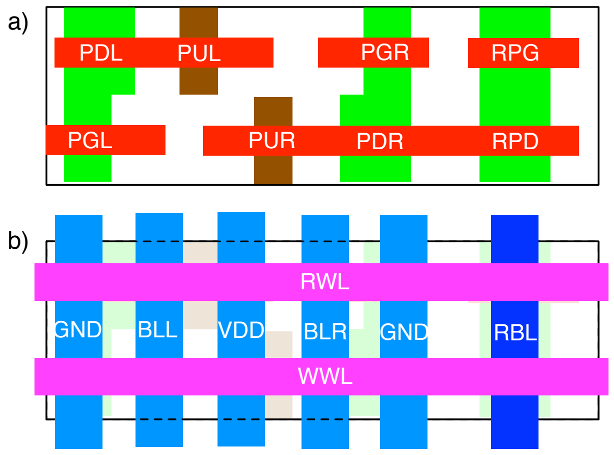


Figure 5: (a) Layout of transistors in a 8T SRAM cell. (b) Layout of metal layers for a 8T SRAM cell.

2.3 SRAM Failure

All transistors must be sized at design time to minimize failure. Unfortunately, there are many trade-offs that prevent an optimal cell size. One of the biggest constraints is area. For example, readability can be improved by making the PG and PD larger, but this increases the area of the cell. Also, the trade-off between read stability and writeability creates another constraint. The PG needs to be stronger than the PU to allow writeability, but needs to be weaker than the PD to prevent read stability problems. Incorrect sizing can produce cells that are much more likely to fail.

In addition, there are many non-idealities in silicon that can cause SRAM failure even for well-designed cells:

- Individual cell variability (eg. threshold voltage, width, length, etc)
- Systematic variability (eg. different corners)
- Random Telegraph Noise
- Sleep supply voltage
- Supply noise
- Aging
- Temperature

Beyond this, there are effects that can cause array failure even for well-behaving cells:

- Sense-amplifier offset
- Control-timing variation
- Cell leakage

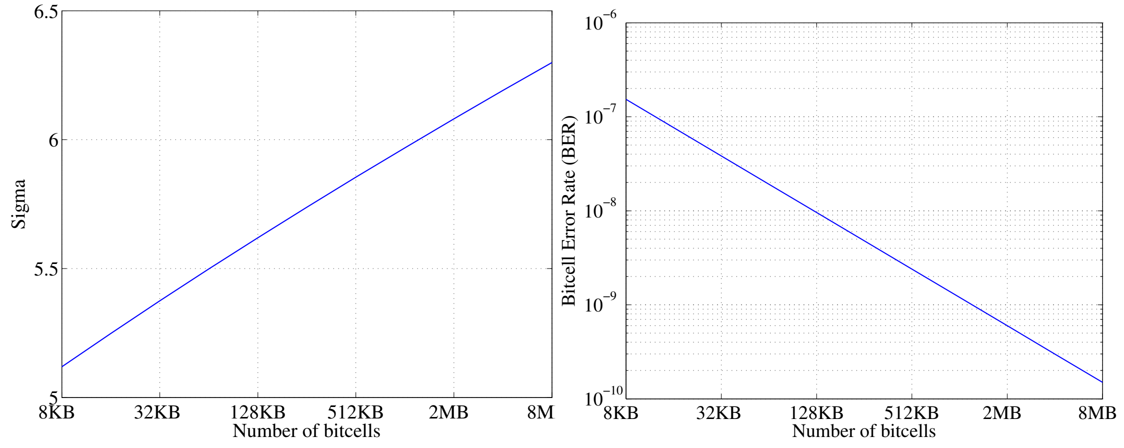
In order for SRAMs to function, these effects must be margined, but margining will cause overhead. For example, an SRAM can be margined such that the bitline discharges more than needed for typical cells so that in the worst case the bitline discharge is still enough, but this wastes energy every cycle.

Ideally, the SRAM should adapt dynamically to remove the need for margining. But because cell sizing cannot be changed, other techniques must be used to change the strength of each device. In the past, body biasing has been used to change threshold voltages, but our simulations showed that this has very little effect in bulk 28nm. Therefore the only possible way to change strengths is to change the wordline voltage, bitline voltage, cell VDD, and cell GND. Also, these voltages can be set differently for different operations, which can decouple constraints in traditional SRAM design. For example, by changing the bitline voltage, the strength of the pass gate can be made different for read and write operations, and therefore remove the constraint between writeability and read stability.

For an SRAM to operate correctly, failures need to be very rare events. The maximum probability of failure is set by both the number of cells on a chip and the required yield. The yield can be calculated with the following equation, where *n* is the number of bits on the chip and *e* is the probability that a bit causes an error (bit-error rate, also referred to as BER).

$$yield = (1-e)^n \tag{1}$$

Traditionally, metrics that predict failure have been assumed to be Gaussian, and report the probability of failure in terms of σ. In order for a design to be considered functional, a metric's value must be considered passing under x σ (x being typically between 5 and 7) deviations from the mean. While this metric is intuitive, it complicates analysis for SRAMs with ECC, and assumes that the error distribution is Gaussian, so it will not be used. Figure 6 plots the BER (e) or σ required for 99% yield for different SRAM cell counts per chip. We will assume a BER of 10^{-9} is required, but this rate can be as high as 10^{-6} for small designs.



(a) Sigma (b) Bit Error Rate

Figure 6: Effect of chip SRAM cell count on required failure probability.

2.4 Static vs. Dynamic Metrics

There are four ways a cell can fail: it can fail to write (writeability error), fail to read (readability error), change values during a read (stability error), or change values during a retention state (retention error). Traditionally, static metrics (calculated as simulated DC operating points) have been used to quantify failure rates. For example, the static noise margin is defined as the size of the square inside the plot of the left and right-side transfer functions during a read [9], known as the Static Noise Margin (SNM). To find how likely failure is for a large array, a short Monte Carlo simulation is run to determine a distribution of SNM values, and the results are fit to a Gaussian distribution. The standard deviation σ is computed for the fit distribution, and the failure metric is extrapolated to Nσ (where N is typically between 5 and 7 depending on the target array size). If the SNM at Nσ from the mean is larger than 0, then a large array will not be functional. However, this approach suffers from two problems. First, SNM is a static metric that does not accurately track stability because it assumes infinite access time and does not account for any transient effects. Second, there is no guarantee that SNM is actually Gaussian at Nσ from the mean. Silicon measurements have shown that static read stability metrics overestimate failure rates while writeability metrics underestimate failure rates [10]. Writeability traditionally uses a static metric that measures how low the bitline voltage must be for the cell to change state. Note that readability does not have a corresponding static metric as readability strongly depends on the peripheral circuitry implementation. We will focus on the effect of design decisions on dynamic metrics and will use statistical sampling that does not assume Gaussian distributions and accounts for finite access times and transient effect.

2.5 Readability Failure

If a cell cannot be successfully read during a cycle, a readability failure has occurred. For a 6T SRAM, a read is successful if the voltage of the bitline on the side holding a 0 is more than ΔV below the voltage of the other bitline when the wordline turns off. The offset of the sense amplifier sets the required ΔV. Figure 3a shows both a successful and unsuccessful read. For an 8T SRAM, reads are single ended, so a read is successful if the read bitline drops below a predetermined switching threshold.

2.6 Read Stability Failure

If the value of a cell changes during a read cycle, a read-stability failure has occurred. Figure 3b shows a read with and without an upset. This type of failure can occur to any cell on the activated wordline. The worst read-stability case occurs when there are multiple read operations in a row, as the weakened cell can flip on the second cycle, so stability is checked after an additional read. We use this pessimistic measure of stability, which accounts for successive reads.

2.7 Writeability Failure

If the value of a cell cannot be changed during a write cycle, a writeability failure has occurred. Figure 3c shows both a successful and unsuccessful write operation. For marginally writeable cells, it is possible that by the end of a clock cycle, the value inside has not been completely written and is near the meta-stability point. An optimistic measure would wait a long period of time before checking the value of the internal nodes while a pessimistic measure would check the internal nodes immediately after the wordline turns off. If there is a read-after-write or write-after-write operation, an optimistic measure will be wrong. Therefore, we use a pessimistic measure by checking to make sure that the high internal node is within 30% of Vdd at the end of the write cycle. We found cells that met this target completed ultimately completed the correct write operation.

2.8 Hold Failure

One of the best ways to reduce energy consumption of an idle SRAM is to decrease leakage by decreasing the supply voltage. Because no read or write operations need to be performed while the array is idle, the minimum acceptable supply will be lower than Vmin (the minimum voltage required for error-free reads and writes) during lower operation. However, if the supply becomes too low, the feedback provided by the cross-coupled inverters becomes too weak and the cell can flip. The voltage at which cells begin to flip is known as the data retention voltage (DRV).

A dynamic test can also be implemented to measure the data retention voltage. Based on an assumption about the rise and fall times of the power rails, the supply is dropped from nominal to V_{TEST} for a long period of time (eg. 1ms) then raised again. If the cell does not retain the same value throughout the test an error is recorded. Figure 7 shows the result of this test. The DRV is about 375mV for a BER of 10^{-9}.

2.9 Leakage

For our cell/technology, leakage is very low and has no effect on failures. However, for bitcells that leak, many more issues appear. For example, the readability of a cell becomes dependent on the value of the other cells on the same bitline, as leakage can pull down the bitline that should be left high and increase the amount of discharge needed on the other bitline in order to establish enough differential for the sense

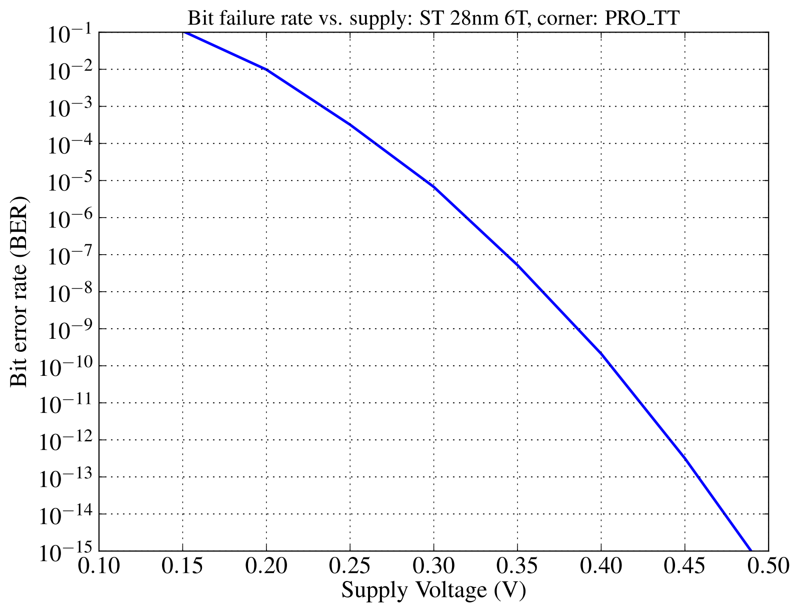


Figure 7: Hold failure rate (data retention voltage) for a 6T SRAM.

amplifier. Also, leakage of the low node through the pull down can cause the entire array to draw a large amount of energy even during inactivity, so leakage reduction design techniques need to be investigated. Last, various assist techniques that affect the pass gate will have a strong effect on leakage, which will increase the probability that the assist causes a stability failure and increase the energy overhead.

2.10 Other

Some assist techniques will create other possibilities for failure. If the value of an non-accessed cell changes during a write operation, a write-stability failure has occurred. This type of failure can occur on any column that is being written. Typically, it will only occur if a negative bitline assist is used as the overdrive of the PG on not accessed cells is not zero.

3 Analysis Methodology

SRAM arrays can fail if one out of millions of cells fail. Monte Carlo simulations can be used to measure the effect of variability, but in order to find failure events, hundreds of millions of simulations would be required, which would be unfeasible. If the quantity under measurement is known to be Gaussian, these results can be predicted with much less simulation time. A few simulations can be made to approximate the mean and deviation of the Gaussian distribution, and the probability of failure many σ from the mean (for example, six σ) can be used to estimate what the actually failure rate will be if there were millions of cells. This has been done for metrics defining failure, such as Static Noise Margin, that look Gaussian. However the tails do not behave as Gaussian. Ideally we should run Monte Carlo only at the tail of the distribution, and then determine how the probability of failure at the tail relates to the nominal case—a process known as importance sampling [1].

This approach will not only determine failure probability, it will also determine the relative strengths of devices that cause failure, and therefore explains *why* a cell fails. From this information, interesting observations can be made about how cells fail that are often non-intuitive or contradictory to traditional explanations.

Most importantly, this algorithm can be adapted to different circuits with a different number of variables in a few lines of code. All that is needed is a netlist describing which variables should be part of the design space which returns a pass/fail signal for a given threshold shift.

3.1 Importance Sampling

3.1.1 Definition

Figure 8 shows a graphical representation of the main idea behind importance sampling for a simplified case of two devices. The two axes represent the threshold voltages of the two devices, with their nominal values labeled as $V_{th0,n}$. 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 $\frac{A_{Vt}}{\sqrt{W*L}}$. For this simplified example, assume the failure contour exists as shown in the upper right quadrant, where if both devices are weak, the metric represented by the failure contour fails. Ordinary Monte Carlo will only sample at a failure event for the very small probability shown by region $\textcircled{1}$.

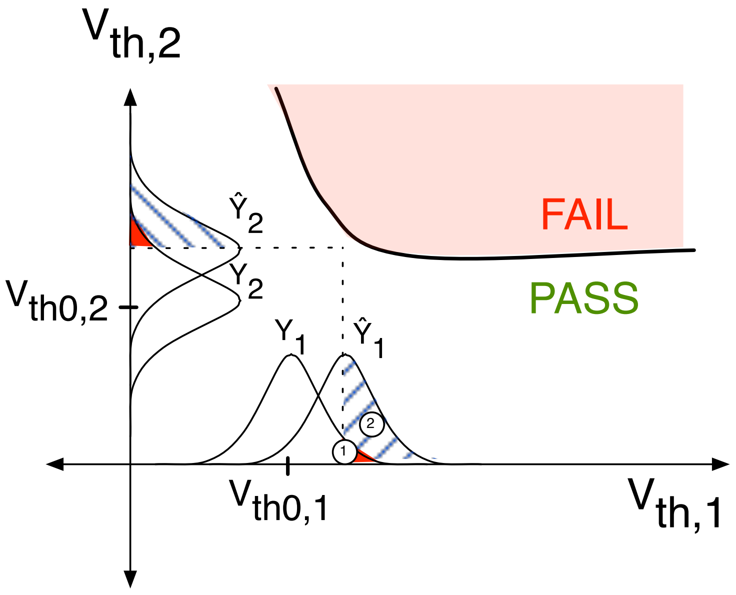


Figure 8: Importance sampling provides more information about the failure region by shifting the mean of each device's threshold voltage.

For importance sampling, we will change the mean of Y_1 and Y_2 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 $\textcircled{2}$. To determine how often these failures would occur without the artificial shift, we need to unbias these samples. If we define $f(x)$ as the joint PDF of the original distribution, and $\hat{f}(x)$ as the joint PDF of the shifted distribution, we can define the weight factor in Equation 4. Now, we can estimate the probability of failure as p_{IS}, and terminate sampling when $\rho(\hat{p}_{IS})$ is < 0.1 as shown in Equation 6 [1].

$$f(x) = PDF\ of\ Y_1, \cdots, Y_6 \tag{2}$$

$$\hat{f}(x) = PDF\ of\ \hat{Y}_1, \cdots, \hat{Y}_6 \tag{3}$$

$$w(x) = \frac{f(x)}{\hat{f}(x)}, \text{ for all } x. \tag{4}$$

Now, we can estimate the probability of failure as:

$$p_{\text{IS}} = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} w(\hat{X}_i) \text{ if } \hat{X}_i \in A \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

To determine when this estimate has reached the confidence levels required, we use the metric defined in [1] to track the variance of our estimate.

$$\rho(\hat{p}_{IS}) = \frac{\sqrt{VAR(\hat{p}_{IS})}}{\hat{p}_{IS}} \tag{6}$$

However, running importance sampling as described solves only half of the problem, as finding the sampling distribution $\hat{f}(x)$ becomes complicated for a many-dimensional design space. This sampling distribution can be described as a set of *n* threshold voltage mean shifts for *n* devices, and can be described graphically as a point in an n-dimensional space of ΔV_{th}.

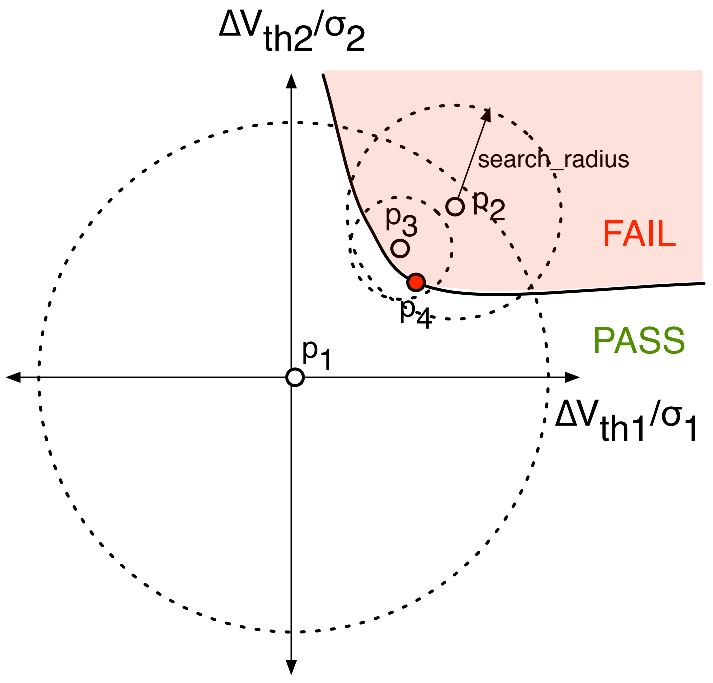


Figure 9: Graphical example of our variable-radius algorithm for hypothetical two-device circuit.

3.1.2 Algorithm

However, finding the optimal sampling distribution $\hat{f}(x)$ becomes complicated 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. We propose an improvement that uses uniform sampling of a variable radius n-dimensional sphere around changing points with a similar idea to the method proposed in [11].

Pseudo-code for the algorithm is provided in Algorithm 1, and Figure 9 shows a simple case of a two-device circuit graphically. 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 (p1) and the closest failure point p2 is found. Now, 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.

Algorithm 1 Find the BER for a given schematic

```
$\vec{p} = [0\ 0\ ...\ 0\ ]$    ▷ Each entry in the vector represents the amount of threshold voltage shift (from the mean), in terms of $\sigma_{Vt}$, for each device.
function SAMPLE($search\_radius, current\_shifts, num\_sims$)
    repeat
        For each entry in the vector $\vec{p}$, choose a shift from a random uniform distribution with endpoints equal to [current_shift - search_radius,current_shift + search_radius]
        Run circuit simulator to determine whether metric is pass/fail for device thresholds given by chosen shift
    until $num\_sims$ simulations completed
    return Failed point with smallest $||p||$
end function
$r = 7$
while TRUE do
    $\vec{next\_p}$ = SAMPLE($r, \vec{p}, 1000$)
    if $next\_p$ empty then                                    ▷ didn't find a failure point
        $r = 2 * r$
    else
        $r = ||p|| - ||next\_p||$
        $\vec{p} = \vec{next\_p}$
        if $r < 0.002$ then
            break                                              ▷ no longer making progress
        end if
    end if
end while
Run Importance Sampling at $\vec{p}$
```

An alternative implementation uses sensitivity analysis to converge to the most probable failure point [12]. In the proposed algorithm and [11], only a binary result is determined—pass or fail. However, if a metric can be developed that has a numerical value, the sensitivities of this variable from each transistor can be calculated in order to move through the multidimensional sample space. For example, to test for read stability, at each point the critical wordline pulse needed to upset a cell can be measured. Then, each device threshold voltage is changed slightly (while the other devices remain at their nominal values), and the effect on the critical wordline pulse is measured. Using convex optimization, the total threshold shift of each device is minimized while the metric is kept close to the target. However, we found that importance sampling fails to complete in a reasonable time in about half of trials for this approach. An in-depth exploration of methods to improve this convergence found that due to the computational cost of calculating these sensitivities, the correlation between sensitivities, and widely varying start locations, even a heavily optimized (and no longer general) algorithm failed to outperform the variable scatter algorithm described here.

3.1.3 Implementation

Threshold voltage variation due to random dopant fluctuation has been shown to be the primary source of failure in SRAM cells. Therefore this algorithm models variability by changing the thresholds of all transistors in the SRAM cell. Other sources of variability, such as width, length, or mobility differences, could also be modeled using this same system by just increasing the number of variables, but would create a larger solution space and require more simulation time.

For each dynamic test (writeability, readability, and read stability) a small testbench is developed to measure for pass or failure through a transient simulation. This netlist is parameterized for different assist techniques and voltages, bitline capacitance, wordline pulse width, access time, sense amplifier offset, corners, and each device threshold mean and deviation can be set outside of the netlist.

A Matlab program runs importance sampling by changing the input to these netlists, running short bursts of Monte Carlo, then collects and parses the results to decide the next step in the algorithm.

3.1.4 Usage

The importance sampling script infrastructure was designed to be as flexible as possible. In order to simulate failure rates, only three things are needed:

- 1) Model files that allow changing the mean and deviation of the threshold voltage
- 2) A simulation netlist that a) returns a pass/fail signal that is 1 for pass and 0 for fail b) parameters to each device whose variability needs to be simulated to set the mean and deviation c) returns the threshold voltages of each device under study
- 3) A function call sends a list of all devices to change the threshold for

This flexibility enables the in-depth study of SRAM failures in a short amount of time. One simulation point takes about 4 minutes on an 8-core machine. This code has been successfully extended to measure other circuits such as a sense-amplifier.

3.1.5 Example: Analyzing results

This methodology allows us to examine very specific failure mechanisms. For example, we will later show that read stability is worse in the FF corner than the FS corner. Traditional thinking suggests that the FF corner should be better due to the following scenario shown in Figure 10 and explained below.

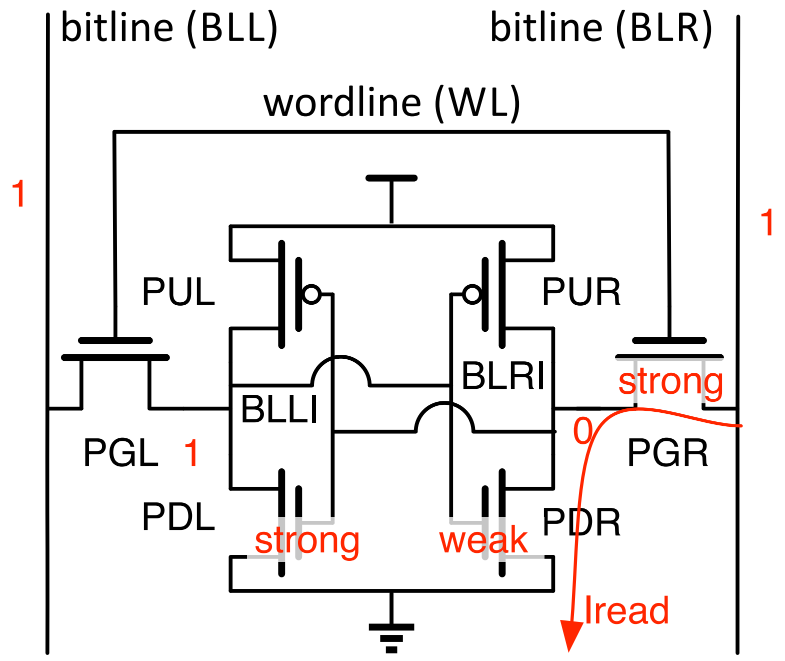


Figure 10: A 6T cell that is very susceptible to a read stability upset.

BLRI holds a 0 and BLLI holds a 1. BLRI experiences a bump due to the voltage divider between PGR and PDR, and if PGR is strong and PDR is weak, this node unintentionally bumps even higher. Therefore PUL and PDL will both be on, especially if PDL is strong. This will cause a fight between PGL/PUL and PDL where PDL is unintentionally decreasing the node from 1. One might guess that the FF corner would be more stable than the FS corner (different PMOS strengths) because in the FF corner, PUL is stronger than PDL, so BLLI will be held high and the cell should be stable, while in the FS corner, PUL is weaker so it is easier for PDL to pull BLLI down and the cell to flip.

However, in reality, FS is a more stable case and importance sampling can explain why. For the FS corner, PUR is intrinsically weak because of the corner, and cannot complete the read stability upset regardless of how high BLLI jumps because it cannot pull BLRI high. So even though PUL cannot hold BLLI high, PUR also cannot pull BLRI high. This failure situation needs PUR to be strong in addition to the requirements shown in Figure 10, so is less likely to occur, and FS is more stable. Importance sampling provides a way to view waveforms at these failure points and determine actual causes for failure, yielding much more intuition about how to prevent SRAM failure.

3.1.6 Verification

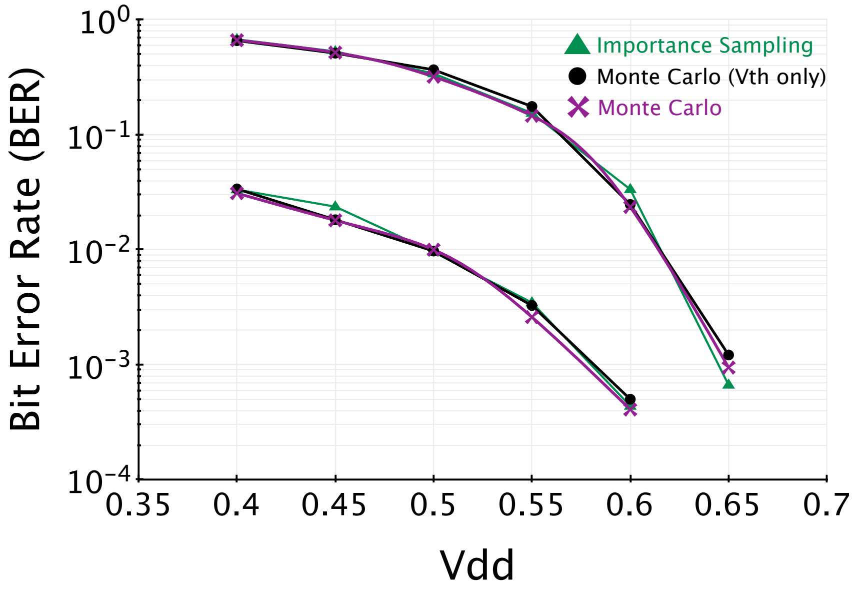


Figure 11: Comparison of predicted bit-error rates using both Monte Carlo and importance sampling for $\rho < 0.1$ for both readability and writeability tests.

Our importance sampling results match closely with Monte Carlo simulation, as shown in Figure 11. Note that our IS implementation assumes that the only source of variation is Vth variation, yet can be seen to track full MC well. BER smaller than 10^{-4} cause an excessive MC runtime. Other studies have shown that IS matches MC for longer simulations [1].

3.2 SRAM Failure Analysis

Figure 12 plots the failure rates for each mode of failure—readability, writeability, and read stability—versus supply voltage. Points at the top of the graph represent almost constant failures, while points at the bottom of the graph represent very improbable failures.

For an array to be considered working, both its read and write operations must fail less than some target BER (set by number of elements on a die and desired yield). In this paper, 10^{-9} is commonly used. For the case of readability and writeability failure curves, the one with the highest BER (the closest to the top of the graph) is chosen to be dominant, because these operations are independent. However readability and read stability can both occur on the same cycle. Generally, the failure rate of one is orders of magnitude higher than the other, so will still assume that the highest BER dominates.

There are two ways to interpret this graph. First, for a given supply voltage, each mode of failure will occur with a different probability. Therefore to optimize the cell, the curves can be shifted up and down until the BER are similar for a given supply voltage. Second, for a given target BER, different designs will have different minimum voltages, so to enable a more energy-efficient operating point, the curves need to be moved to the left. Changing cell sizing, array design, and assist techniques can all move these curves and allow for cell optimization.

Vmin is defined as the voltage such that all failure metrics have a BER of less than 10^{-9}. In this figure, Vmin is 830mV.

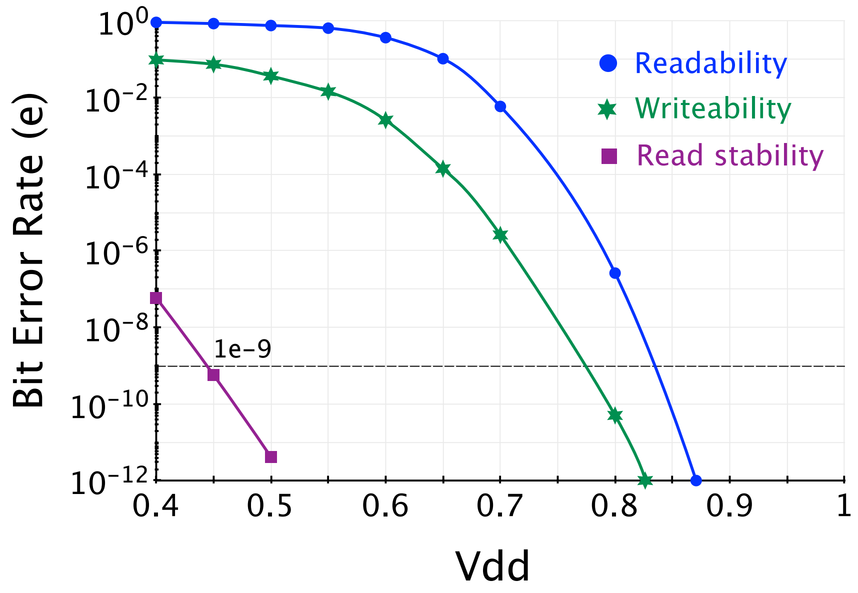


Figure 12: Readability, writeability, and read-stability failure rates for a 28nm 6T SRAM bitcell.

3.3 Dynamic Failure Metrics Test Chip

While our estimates of the BER based on importance sampling are interesting, for them to be truly useful, we would like to verify the results in an actual silicon design. In addition, we also need to find the connection of assumptions used to calculate the BER, such as bitline capacitance and timing, with an actual SRAM design. Last, we would like to calculate accurate energy measurements, which requires an actual baseline SRAM design. For these reasons, we describe two baseline arrays, designed in bulk 28nm, and taped out in May 2011.

3.3.1 6T

The floorplan of our full array implementation is shown in Figure 13. It is formed by combining 4 sub-arrays of Figure 4 to form an 8kB memory. Based on the address, the predecoder and decoder will turn on the wordline of an entire row (including both the left and right sub-arrays connected to the decoder) to access 512 columns. The column IO performs 8 to 1 interleaving to read or write a 64 bit word. Note that the effective size of the array is 128 rows by 512 columns, yet both the wordline and bitline are split in half in this simple hierarchical implementation to reduce series resistance on the wordline, and to cut bitline capacitance in half.

Timing is controlled by the positive and negative edge of the clock. The positive edge of the clock turns on the wordline, and turns off precharge. The negative edge of the clock turns off the wordline, turns on the sense amplifier, and turns on precharge. All data is registered for the entire cycle, with the exception of the address input, which is latched with a small transparent window before the clock edge so that addresses have time to propagate through the decoder.

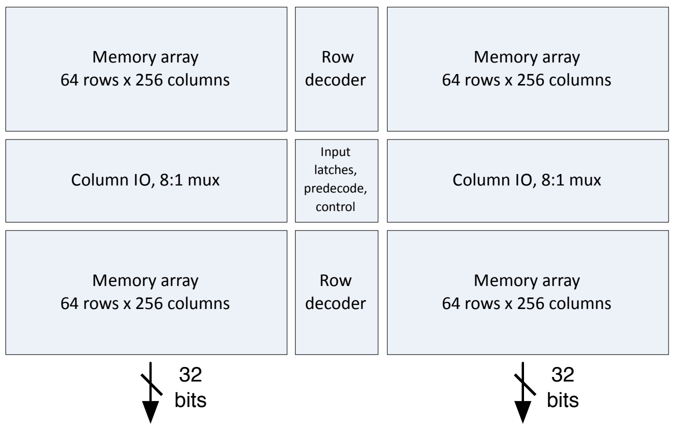
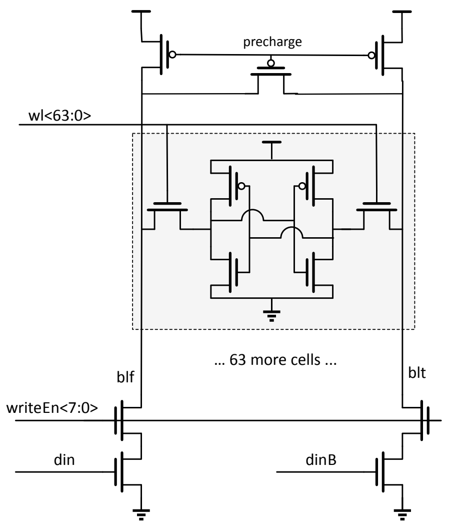
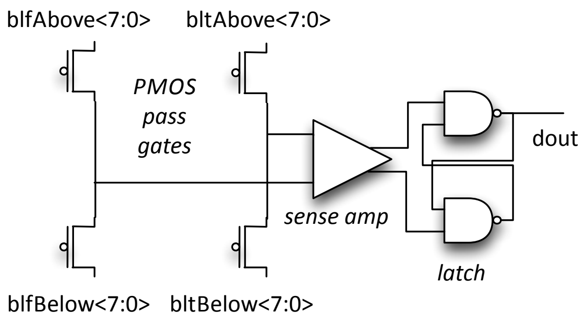


Figure 13: Physical organization of our 6T SRAM.

The write path is shown in Figure 14a. writeEn selects the column to write and can be turned off by a write mask. Depending on the data, a single column is pulled down while the other remains high. An active wordline writes the cell.



(a) 6T SRAM write path.



(b) 6T SRAM read path.

Figure 14: 6T periphery circuitry for read and write operations.

The read path is shown in Figure 14b. Eight columns from above and eight columns from below are muxed together to a single shared node for both the true bitline and the complement bitline. Only PMOS pass gates are needed, as the sense amplifier compares values near Vdd. A StrongARM sense-amplifier reads the data, and stores it in a SR' latch. The sense amplifier differential pair uses non-minimum length transistors to reduce offset. All column IO must be bit pitch-matched in groups of eight columns.

3.3.2 8T

The floorplan of our 8T full array implementation is shown in Figure 15. Because the size of the cell is double that of the high-density cell, our wordlines can only have half the total number of cells before the series resistance becomes too large. Additionally, because this array is a two-port design, we need separate read and write decoders. Because each row now needs two wordline drivers, lack of space for wiring requires that the read and write row decoders drive the wordline from opposite ends of the array, so the wordline cannot be split in half and the size of the sub-array is 2kB.

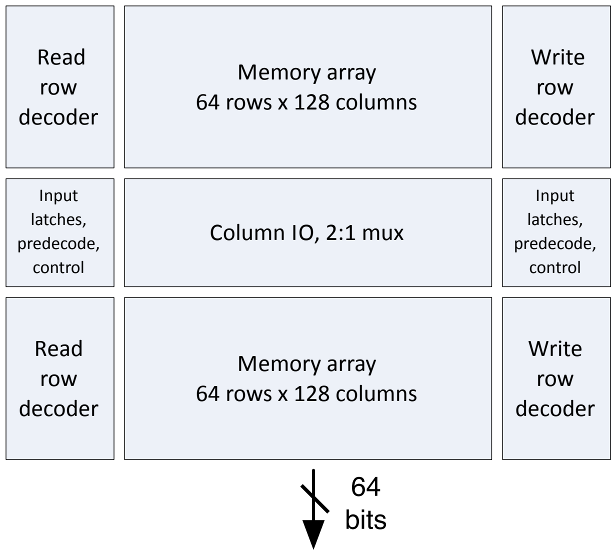
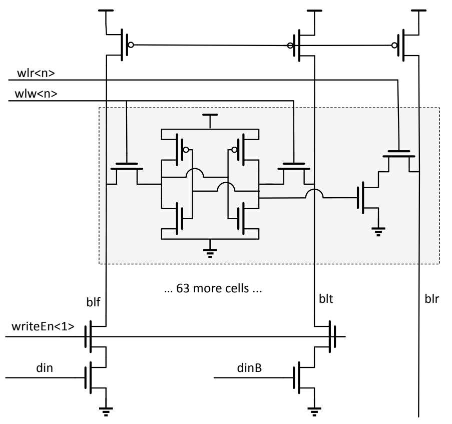

Read row decoder
Memory array 64 rows x 128 columns
Write row decoder
Input latches, predecode, control
Column IO, 2:1 mux
Input latches, predecode, control
Read row decoder
Memory array 64 rows x 128 columns
Write row decoder
64 bits

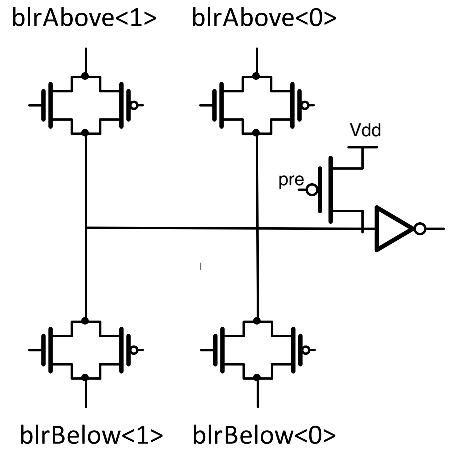
Figure 15: Physical organization of our 8T SRAM.

The write path is shown in Figure 16a. Writing is performed through blt and blf in the same manner as the 6T. An additional read bitline (blr) creates the separate read port.

The read path is shown in Figure 16b. Two columns from above and two columns from below (blrAbove and blrBelow are muxed together to a single shared read node. Before the read operation, the precharge signal (pre) precharges both the read bitline and the shared read node to Vdd. Transmission gates need to be used to ensure the domino-style read inverter can fully transition.



(a) 8T SRAM write path.



(b) 8T SRAM read path.

Figure 16: 8T periphery circuitry for read and write operations.

4 SRAM Vmin Analysis

Using importance sampling and simulation of an extracted baseline array, we can determine how a large variety of design decisions and techniques affect BER and Vmin. Each technique will be analyzed independently. In reality, more than one technique will be implemented and there will be an interaction between each that cannot be accurately predicted by independent studies. Instead, we hope to generate intuition about the sensitivity of BER to various techniques.

4.1 Effect of bitline capacitance on 6T SRAM failures

Figure 17 shows how different bitline capacitances (due to different number of rows) affects each mode of failure.

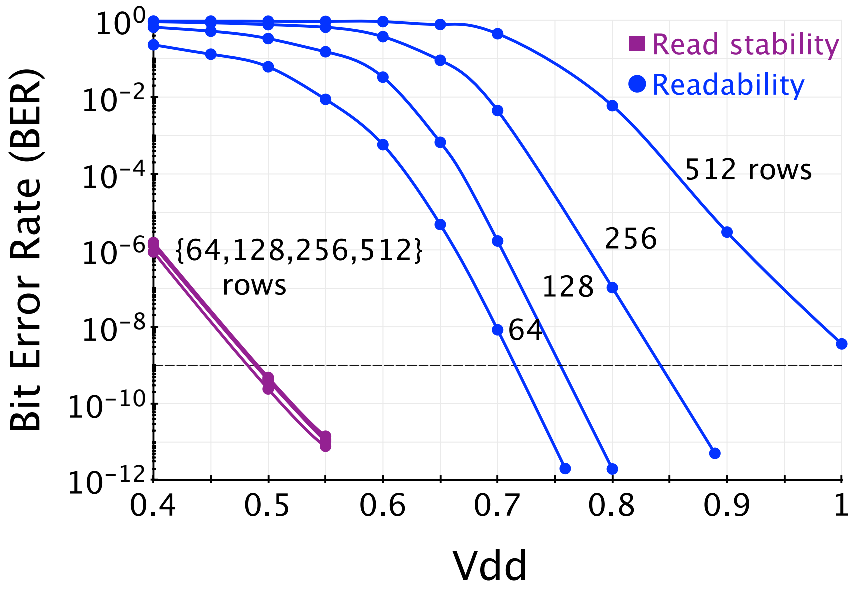


Figure 17: Effect of bitline capacitance on failure rates.

Bitline capacitance has no effect on writeability because writing is done through a bitline forced to ground, and the bitline floating high loses very little charge. Simulations with a floating bitline confirm this observation. Read stability is improved a little for bitlines with smaller capacitance because the dangerous internal node bump on the side holding a 0 decreases over time. However the high bitline loses charge slowly so this only has a minor effect on stability.

Smaller bitline capacitances drastically improve readability. Reads require a certain amount of charge to be removed from the bitline on the side reading a 0. The cell will always pull a constant current, but bitlines with more capacitance will need more charge removed to produce the same voltage differential as a bitline with less capacitance. Readability appears to be the dominant mode of failure unless the bitline capacitance is very small. As shown in Figure 17, the BER of readability and writeability for a target of 10^{-9} will be equal for a bitline capacitance of 15fF.

This finding suggests that creating a small number of rows (short columns) is critical to improving low-voltage operation of SRAMs. However, this increases area overhead, because column and control circuity can be amortized over less cells.

4.2 Effect of bitline capacitance on 8T SRAM failures

For an 8T SRAM, there are two different bitline capacitances–the write bitline (equivalent to the bitlines in the 6T array) and the read bitline. For the write bitline, the effect of capacitance for read stability and writeability will be exactly the same as for the 6T case, even though absolute results will differ a little due to different cell sizing. Read stability failures will only occur for arrays with interleaving.

Figure 18 shows how different read bitline capacitances effect single-ended readability. In this case, a domino read is required, so a pass metric requires an 80% transition past the buffer. The read bitline has slightly higher capacitance than the 6T case, because the pass gate for the read port is larger, however there is less wiring capacitance because the wires can be spaced further apart. In comparison to the 6T, the dedicated read port of the 8T enables a Vmin decrease of around 100mV for a domino read. If single-ended sensing is used, Vmin can decrease even further as bitline discharge requirement is lowered.

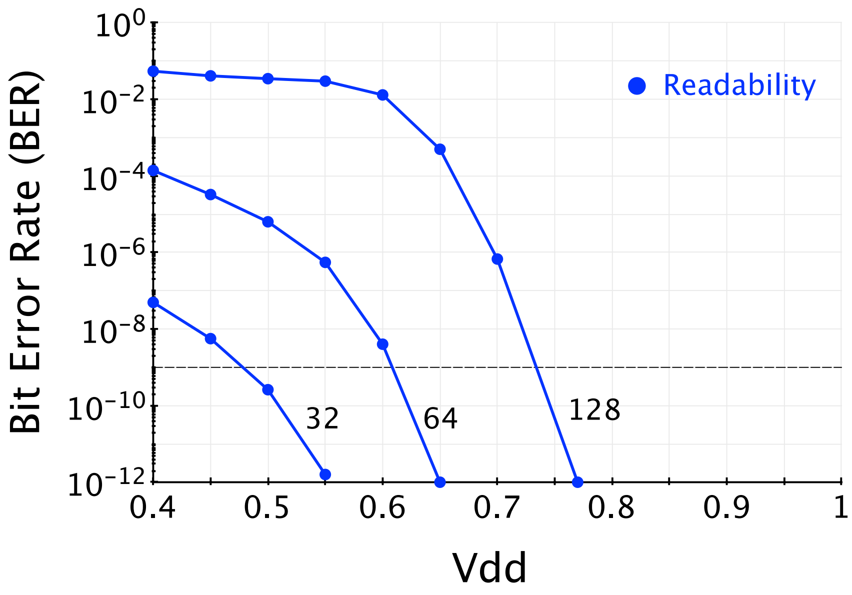


Figure 18: Effect of read bitline capacitance on 8T readability failure rates.

4.3 Effect of clock period on 6T SRAM failures

The SRAM arrays under analysis are assumed to be synchronous memories that must return one read or one write during a single clock period. For failure analysis, the wordline is assumed to be on for half of the nominal clock period (and the other half assumed to be used for precharge and margins).

Figure 19 shows failure rates for SRAMs with clock frequencies ranging from 250Mhz (4e-9) to 2Ghz (5e-10).

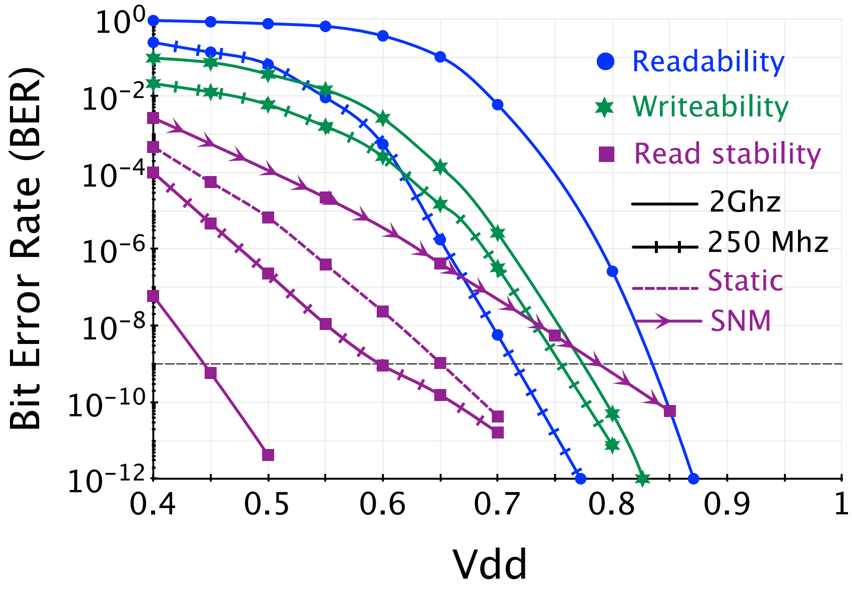


Figure 19: Effect of clock period on failure rates.

Both readability and writeability improve with longer clock periods because more time is allowed to discharge a bitline (read) or flip a cell (write). Read stability worsens with longer clock periods, as more time is given to accidentally flip the cell, but for this cell/technology, its failure rate is dominated by the other modes of failure so should be ignored for design decisions.

Most importantly, note the large discrepancy of Vmin in this figure between traditional metrics and our methodology's read stability. At 2Ghz, Vmin for read stability is about 450mV, while a static test would predict a Vmin of 650mV. Furthermore, if we fit a Gaussian distribution to the SNM instead of using importance sampling, we would predict a Vmin of 800mV. Designing using pessimistic metrics such as SNM and distribution approximations results in a drastic overdesign of the bitcell for stability.

Notice that readability is much more sensitive to the nominal period than writeability. After a write, a bitline needs to be charged from 0 to Vdd while after a read, a bitline only needs to be charged from Vdd-Vread to Vdd (for example, 150mV). Therefore it would make sense to turn on the wordline for longer during a read cycle, as less time is needed for precharge, and readability will be dramatically improved.

4.4 Effect of process corners

So far we have been studying inter-device variation, as these variations introduce strong mismatches between devices within an SRAM cell and can increase the probability of failure. But due to variability in processing, there can also be variation that affects each type of device on the die uniformly. This variability would correspond to shifting the mean of the thresholds of NMOS devices by an amount and PMOS devices by another amount. For example, SF represents NMOS devices which are on average slower by 3 σ and PMOS devices that are faster by 3 σ. By changing the average behavior of a device, we also change the BER of reads and writes.

The Vmin of the high-density cell for different process corners is summarized in Figure 20.

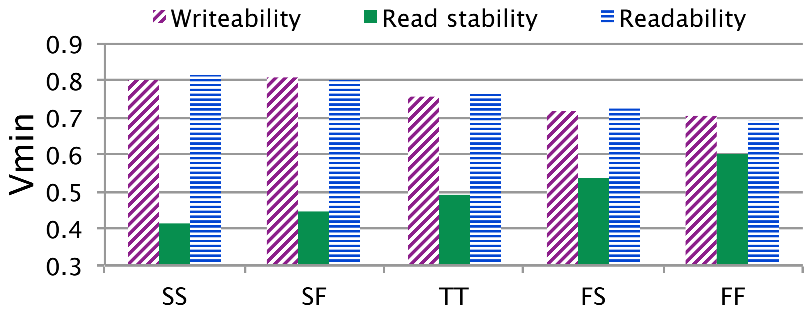


Figure 20: Effect of process corners on BER.

There are a few interesting observations that can be made from these plots. As a primary effect, faster NMOS reduce BER for a given supply for both readability and writeability. For readability, this makes sense as faster PD and PG improve the speed at which charge can be pulled off of the bitline. For writeability, the PG becomes stronger than the PU which makes it easier to change the internal node of a cell.

As a secondary effect, faster PMOS also reduce BER for both readability and writeability. For writeability writeability, faster PMOS helps because it becomes easier for the write to complete. For readability, a better PMOS holds the half of the cell holding a 1 high better, so that the bitline connected to the half holding a 0 needs to be less discharged for the same bitline differential to be established.

Faster NMOS hurts stability, because a voltage bump on the side holding a 0 turns on the pull-down for the side holding a 1. A stronger device is more likely to pull this node down and flip the cell. Additionally, stronger PMOS hurts stability as well because a strong PMOS is needed to pull the 0 node high to complete the stability upset.

4.5 Effect of sense amplifier offset on readability failure

Two main components determine readability—the read current that discharges the bitline, and the sense amplifier offset that measures this discharge. Figure 21 shows how different sense amplifier offsets affect readability. Larger sense amplifiers improve matching in the differential pair and can lower the offset, enabling a significant Vmin reduction.

4.6 Effect of assist techniques on 6T SRAM failures

Up until this point, we have analyzed how array organization (eg. clock period, bitline capacitance, etc) affects failure rate. Circuit techniques known as assist techniques can be used on cycle-by-cycle basis to improve Vmin. A comprehensive analysis of the effectiveness of assist techniques was performed in [6], but static metrics were used to quantify effectiveness, which have been shown to be a poor match to silicon failures [10].

For the investigation of each assist technique in this section, the following assumptions were made:

- Corner: TT
- Design: 28nm High-Density 6T Cell
- Voltage: 1V

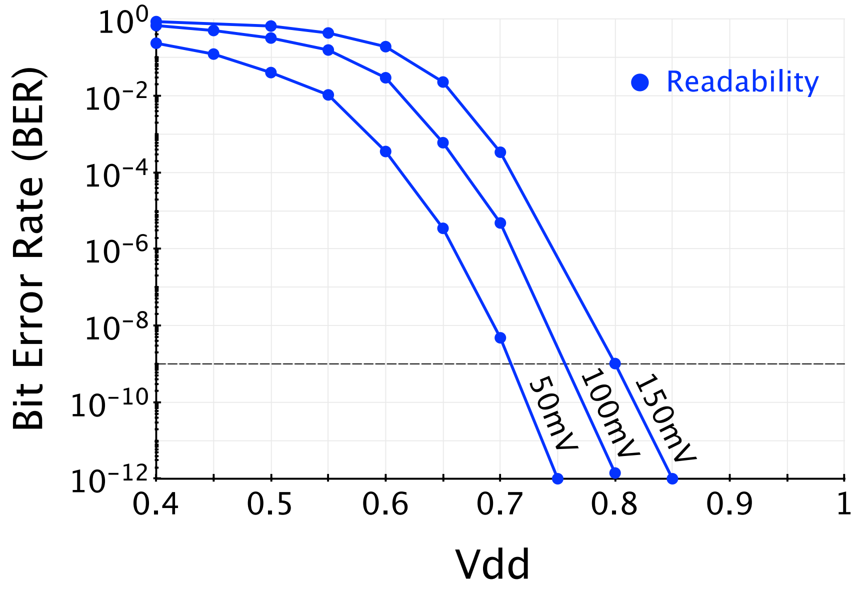


Figure 21: Effect of sense amplifier offset on readability.

- Period: 50 FO4 ($\approx$ 1ns at 1V)
- Wordline Pulse Period: 25FO4
- Sense-amplifier offset at 3σ: 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 increases with the FO4 as the supply is reduced to track the assumption that the SRAM operating frequency will be set by the critical path of a processor.

Different degrees of assist are defined as a proportion of the supply voltage (and not an absolute quantity) because assist voltages are generally set with voltage dividers or charge redistribution. However, stronger assists are needed at lower voltages, so routing a constant supply could improve assists at low voltage for less capacitor area.

Also to determine the side effects of each assist technique, the following assumptions were made about SRAM layout. The reason for these assumptions can be seen in the cell layout in Figure 4.

- Vdd runs vertically (parallel with bitlines)
- Gnd runs horizontally (parallel with wordlines), and is shared between two adjacent cells

The voltage waveforms for a variety of assist techniques that target each mode of failure—readability, write-ability, and read stability—are summarized in Figure 22.

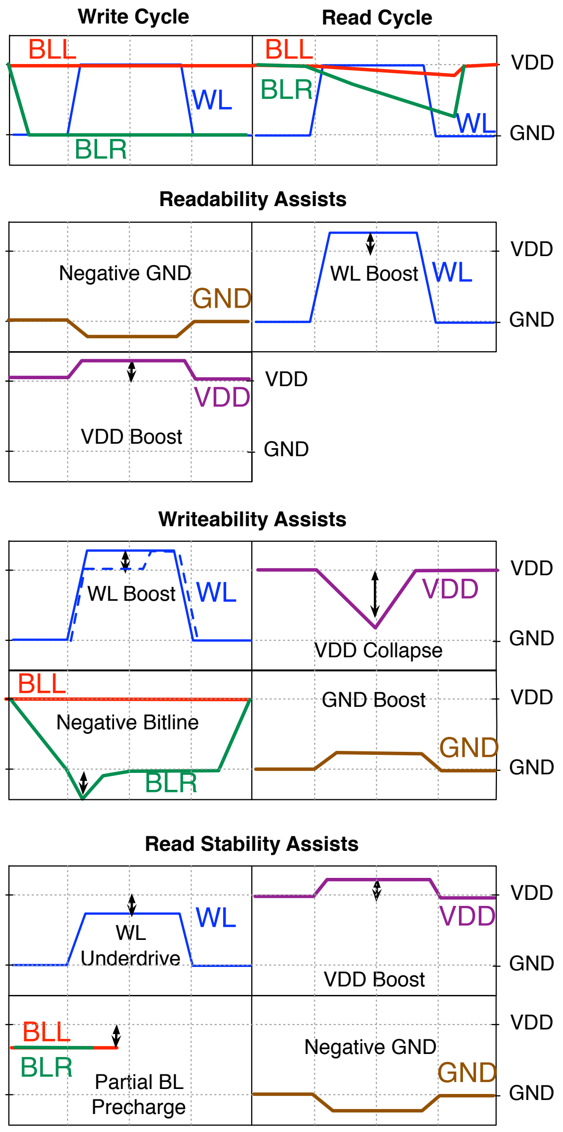
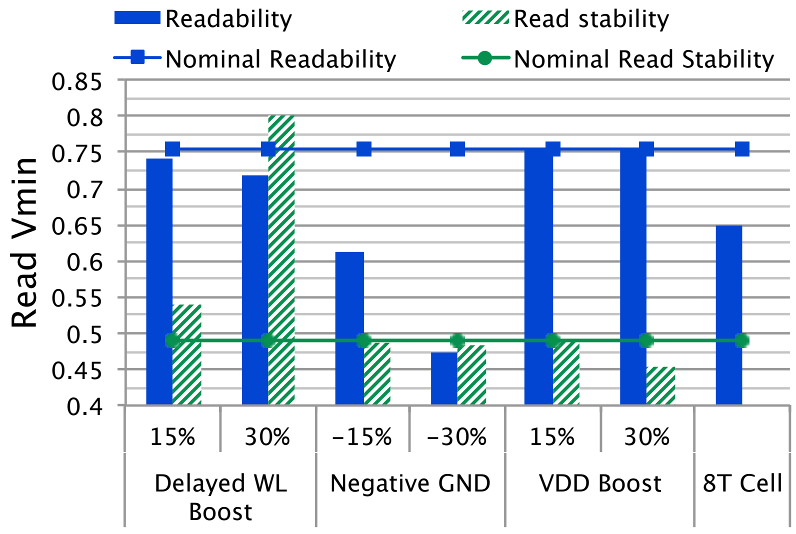
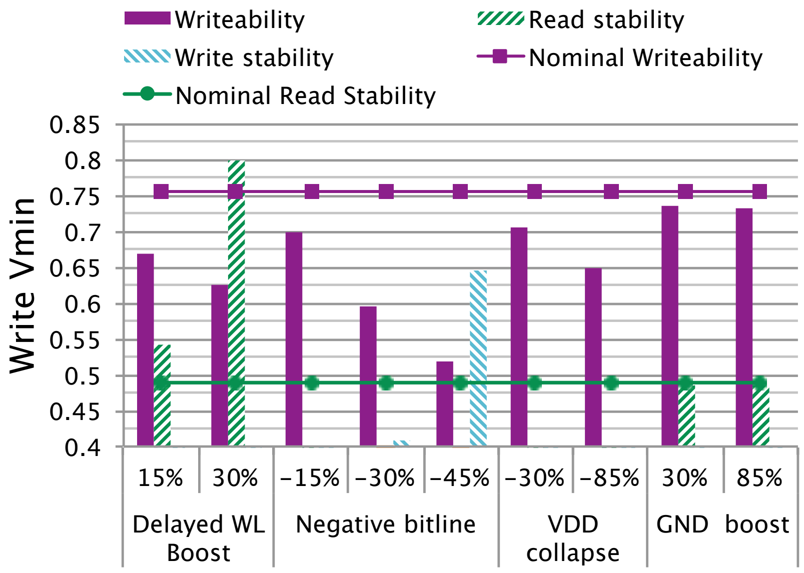


Figure 22: Summary of assist techniques: Negative GND, GND boost, WL boost, WL underdrive, Vdd boost, Vdd collapse, negative bitline, partial bitline precharge.

We have completed a thorough design-space exploration of popular or effective assist techniques. Our results are summarized in Figure 23 for readability and writeability assists, and explanations of each method are described in the following subsections. For our technology/cell, stability assists were rare enough to not warrant investigation. Each Vmin measure also includes any stability consequences caused by the assist. Overall Vmin will be determined by the maximum of Vmin for writeability and Vmin for readability. Because all assists can be applied on a per-operation basis, our results for Vmin are independent.



(a) Readability.



(b) Writeability.

Figure 23: Impact of assist techniques on Vmin.

4.6.1 Effect of wordline boost as a readability and writeability assist

Using a wordline boost can improve both the readability and writeability of cells, but if an array is interleaved, it will diminish the read stability of all half-selected cells, as shown in Figure 24. To avoid the half-select issue, [4] starts the boost part of the way through the wordline pulse, so that half-selected cells have already begun to read and the bitline voltage matches the internal voltage more closely. Figure 24 shows that while this 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 circuitry, as done in [5].

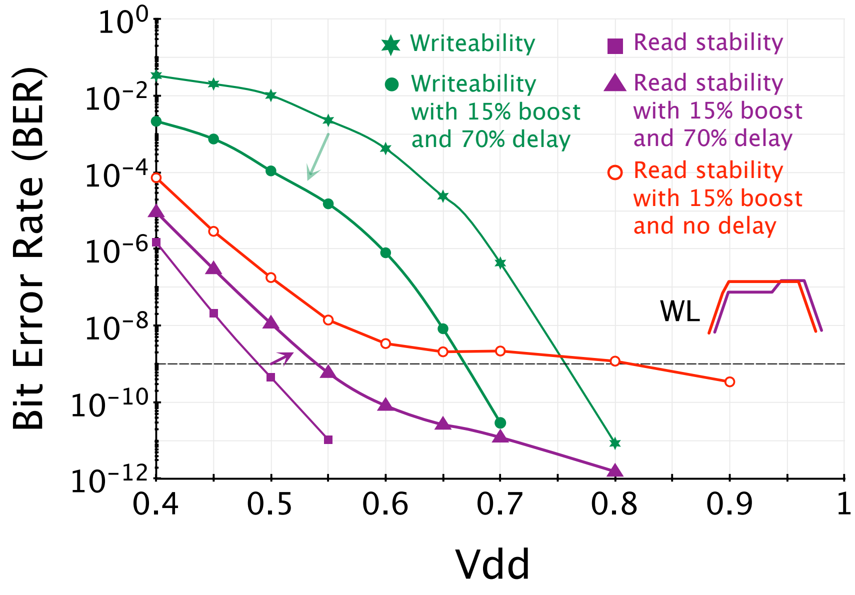


Figure 24: Wordline boost improves writeability while reducing read stability.

- **Operation**: During reads or writes, use of a larger wordline voltage to drive the access transistors along a single row.
- **Implementation**: One possible implementation is to use a separate supply for the wordline buffers, but this requires an extra supply, and up-sizing the buffers to maintain the same wordline buffer strength. Also, all off-row drivers will have a non-zero Vsg, causing short circuit current. Full level-shifting circuits are required to avoid this, but will increase the cycle time of the SRAM.

  To avoid a second supply, capacitive boost can be used on the Vdd node of the row drivers. This capacitor can be shared with multiple rows, yet the more rows it is shared between, the more parasitic source capacitance needs to be moved for the same boost—increasing the size requirement of the boost capacitor.
- **Pros**: The overdrive is increased on the pass-gates, making both stronger than the pull-ups and therefore improving the writeability of the cell. Both the pass-gate and pull-down need to be strong to improve readability. Since the pass-gate is smaller than the pull-down, a weak pass-gate is much more likely to limit readability. Therefore assists that improve the pass-gate (such as wordline boost), are more effective than assists that improve the pull-down (such as Vdd boost).
- **Cons**: As this assist is applied to a row, all other non-accessed cells on the same row will be reading, and a stronger PG will hurt the ratio between the PG and PD and diminish read stability.

  This assist can be very dangerous once we account for the effect of corners, as read stability is very sensitive to changes in wordline boost. As seen in Figure 24, any wordline boost larger than 15% will push the read stability BER above the 10^{-9} limit. The FF corner increases failure rate by more than 3 orders of magnitude over the TT corner. Applying even a 15% boost to help the TT case will break all FF cases. Therefore some adaptive scheme that can determine read stability failure rates is needed to tune the boost. Other assists can avoid this problem by targeting a single mode of failure without affecting other modes, and would not need a complicated control loop.
- **Overhead**: An energy overhead will be caused by a larger voltage on the wordline capacitance, the overhead needed to implement the boost (either flying capacitors charging parasitic source nodes or an additional supply), and the extra discharge caused by improved reads on half-selected columns.

4.6.2 Effect of wordline droop as a read stability assist

Activating the wordline with a voltage less than nominal ($V_{WL} < V_{DD}$) can improve the read stability of a cell by decreasing the strength of the PG, but will dramatically worsen readability and writeability. For this cell/technology, this technique is not needed and will not be analyzed further, but it could potentially be used for cells that were skewed to be unstable or in combination with other assist techniques. For example, using negative bitline will counteract the wordline droop's effect on the pass-gate to keep writeability for accessed cells the same, but will increase stability on half-select columns.

4.6.3 Effect of cell Vdd boost as a readability and read stability assist

Increasing a cell's Vdd voltage should improve both readability and read stability by increasing the strength of the pull-down transistor. Vdd boost helps read stability a lot, but does not appear to help readability. Readability failures occur most often if the pull-down device is weak or if the pass-gate is weak. However, if we use Vdd boost, the pull-down devices are always strong, and failures will occur only if the pass-gate is weak. So as a very approximate explanation, only half of the failure conditions are now avoided, so the BER changes by less than a factor of 2 compared to other assist techniques that generally change by orders of magnitude.

This suggests that cell sizing will affect the results of this study. If the pass-gate was made stronger, then Vdd boost would actually improve readability because the pass-gate is no longer constraining the results.

4.6.4 Effect of cell Vdd collapse as a writeability assist

Vdd collapse decreases write Vmin by decreasing the strength of the cross-coupled inverters, but not as much as expected from [5]. Since we analyze a bulk (not tri-gate) device with different sizing characteristics, differing results can be expected.

The dangerous consequence of Vdd collapse is potential violation of the data retention voltage of accessed SRAMs in the same column. However, our IS tests of this condition show a BER $< 10^{-9}$ for all cases of interests. [5] also found negligible errors for reasonably short access times.

4.6.5 Effect of cell GND boost as a writeability assist

GND boost weakens the cross-coupled inverters, improving writeability [13]. 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 Vdd-Vth. This limitation does not exist for Vdd collapse, as NMOS can pass low voltages without limitation. So while Vdd collapse and GND boost both weaken the cross-coupled pair, the limitations imposed by the PG make Vdd collapse more effective.

4.6.6 Effect of negative bitline as a writeability assist

Negative bitline improves writeability by increasing the Vgs on the PG [3]. Our IS simulation 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 Vgs will appear across the PG of non-accessed rows. If the internal node of an non-accessed bitcell on this side is high, then the value of the cell could flip, causing a write stability error. We generated an IS test to determine the probability of this occurrence and found that the BER was $< 10^{-9}$ for boost amounts $<= 30\%$. As this assist is applied only during writes to only the columns being written, it does not affect readability or read stability.

- **Operation**: During writes only, a negative voltage is applied to the bitline which is writing a false value while the wordline is high.
- **Implementation**: In general, the negative voltage is usually generated by charging a capacitor, then connecting it to the bitline with a flipped polarity. The degree of assist (how negative the bitline goes) can be controlled at design-time by changing the capacitor size, and at run-time by pre-discharging the capacitor before connecting it to the bitline.
- **Pros**: The overdrive is increased on PG, making it stronger than PU and therefore improving the writeability of the cell for more negative bitline voltages.
- **Cons**: Write stability of cells on the same column is diminished because a negative bitline causes a small overdrive on all other PG on the same column which are supposed to remain off. If the PG has a low-enough threshold, the cell could be accidentally written. But for this cell/technology, importance sampling analysis determined it was not a problem.
- **Overhead**: Energy overhead is minimal due to only a slightly large change in bitline voltage.

4.6.7 Effect of partial bitline precharge as a stability assist

Precharging bitlines to around 70% of Vdd with a regulator has been shown to improve yield from 5 to 5.7 σ, or equivalently, from a BER of 2.87e-7 to 6e-9 [2]. Importance-sampling analysis of read stability confirmed their results, with a BER improvement of around 1.5 orders of magnitude at 0.7V. But the assist becomes less helpful at low supplies and only achieves a Vmin reduction of 25mV. While this technique causes less of a bump on the side holding a 0, the opposite side pass-gate is less helpful in holding that node at 0 which offsets part of the advantage. Note that readability is slightly diminished due to the decreased Vds on the PG.

4.6.8 Effect of negative cell GND as a readability assist

Reducing the voltage of GND has been shown to improve readability. [3]. Negative GND is the most effective of all readability-assist techniques as it increases the Vgs 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 wires running horizontally, and the GND wire is coupled into the internal node of non-accessed cells. Also, all read current will flow through the GND wire, so a charge pump used to generate a negative voltage will need to sink all current flowing off of the bitlines. For 8T arrays, the GND wire is routed vertically and many of these problems are somewhat reduced. Cells on rows above and below will have different GND voltages for either side and could cause a stability issue, yet simulation showed BER lower than 10^{-9} for all relevant configurations.

- **Operation**: During reads, the GND of an entire row of cells is pulled below ground. A negative GND increases the V_{GS} on the PD devices, and also pulls the low node below ground as well, which increases the V_{GS} on the PG as well. These two transistors are responsible for discharging a bitline capacitance as fast as possible, so their increased strength improves readability.
- **Implementation**: A charge pump can be used to generate a negative voltage in the same manner as the negative bitline assist.
- **Pros**: Vdd boost was shown to not help readability because it only strengths the PD but not the PG, and either device can cause failure if it is weak. Negative GND helps both devices.
- **Cons**: One internal node in each cell is coupled to the GND wire through the pull-down device on the side holding a 0. Therefore, moving GND requires moving all internal nodes along an entire row. Also, each cell uses two separate horizontal ground lines (see Figure 4), both lines need to be made negative, and this disturbs all cells above and below the target row. For the side of the cell holding a 1, the PD device has 0 at its gate. But the source of PD will get pulled below ground, accidentally turning on the pull-down device and possibly flipping the contents of the cell (because the PD device is much stronger than the PU device). But this has been simulated and even with the maximum degree of assist, the BER is less than 10^{-15} at all supplies, so for this cell/technology, it is not a major concern.
- **Overhead**: In addition to the large capacitance of the GND wire, any read assist will have a large energy overhead because the assist will have a strong effect on how much a cell discharges a bitline during a read cycle.

  The overhead will be increased by both the voltage change on GND, and the extra discharge caused by improved reads on half-selected columns. Note that the capacitance of the GND wire will be similar to the wordline, as the wire length is the same, and GND is connected to two gates.

Note that negative GND slightly improves read stability Vmin. To understand why negative GND does not change the active row read stability, consider a situation where BLLI is logical 1 and BLRI is logical 0. Three components are needed for a cell to flip: PGR must be strong and PDR must be weak to pull BLRI high, and PDL must be strong to pull BLLI low to compete the stability upset. A negative GND boost increases the Vgs on PDR, strengthening this device and helping to prevent flips, however it will also increase the Vgs on PDL, which will help cause flips, and these effects cancel.

4.6.9 Effect of other assist techniques

We have only discussed assist techniques that are effective, or assist techniques published in literature which appear to not be as effective as predicted. Assists that change the body voltage were ineffective due to the low body-effect coefficient in this technology. New technologies such as FDSOI would allow a resurgence of body biasing, however substrates generally have a large capacitance so body techniques would be better used to offset process variations than to optimize cycle-by-cycle for read or write operations as other assists do.

There are also many potential combinations of assists. For example, during a write, the wordline can be drooped to improve stability of half-selected cells while negative bitline can be used on only the columns being written to undo the effect of the wordline droop for cells that actually need to be written. Also, these effects can be used in combination with cell sizing. For example, the PD device can be made smaller to save area, and negative GND can be used to overcome this weakness during read operations. The analysis methodology explained above can be used to quantify the result of any design decision on failure rates.

4.7 Effect of assist techniques on 8T SRAMs

8T bitcells have the same writeability assists as 6T bitcells. If 8T cells are interleaved, half-select during 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 23a 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.

Because cell sizing and array layout is different for 8T arrays, results differ in a few cases. Note that for an 8T array, GND runs along a column and is shared with all three pull-down devices, so unlike the 6T case, only one GND wire needs to be pulled negative instead of two. In our cell, the read stack shares GND with the pull-down devices, but there is wiring room for these to be split into two wires to reduce the capacitive load of a negative ground assist. However, by keeping both GND connected to the same negative voltage, the RPD for cells holding low on the read side will maintain a V_{GS} = 0 as intended. If the GND wire was split, the RPD will have a nonzero V_{GS} in this case and the wrong value will be read.

4.7.1 Leakage

Our technology 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 by this methodology by using Monte Carlo to characterize the leakage current of N worst case cells as a Gaussian, and including it into IS as an additional variable described by the fit distribution.

4.8 Effect of hierarchical bitlines

Hierarchical bitlines improve cell operation and energy-efficiency while increasing area overhead. Because the devices that couple local to global bitlines can be made larger, variability will have little effect, and analysis of the effect of hierarchical bitlines on failure and energy can be made within the existing infrastructure just by decreasing the number of cells on each row and column to their respective local sizes. Then by multiplying these results by appropriate factor, we can compare a hierarchical design to a non-hierarchical one. However, the main consequence of hierarchical bitlines is area, and this is heavily dependent on layout design rules for a specific process, so a general observation on their effectiveness is difficult.

4.9 Effect of ECC

Error-correcting codes (ECC) are commonly used in memories to prevent bit failures from causing system failures. Different ECC schemes are commonly characterized by both the number of bits errors per word that they can correct and the number of bit errors that they can detect. So far, all analysis has focused on bit-error rate (BER). However, the system using an SRAM array (and possibly ECC) only cares about codeword error rate (CWER)–the probability that an entire group of bits has a certain error rate.

The probability that a n-bit codeword without ECC fails when each bit has an error probability e is given by

Equation 7:

$$p_{fail} = 1 - (1 - e)^k \tag{7}$$

So the probability of failure is one minus the probability that all bits are correct.

A common ECC scheme known as a Hamming code can correct a single error. The probability that a n-bit Hamming covered codeword fails is given by Equation 8:

$$p_{fail} = 1 - [(1 - e)^k + \binom{n}{1} * e * (1 - e)^{n-1}] \tag{8}$$

Now there are two different conditions for success. The first condition is 0 errors, and this is represented by the first term. The second condition is 1 error, and a combination is used because it doesn't matter which bit it is (in this case, there are n different ways one bit can be wrong).

This equation can be extended for more any number of errors by adding more terms. The probability of failure with ECC that can correct K errors is given by Equation 9.

$$p_{fail} = 1 - [\sum_{k=0}^{K} \binom{n}{k} * e^k * (1 - e)^{n-k}] \tag{9}$$

In this section, we explore the relationship between bit error probability, operation error probability, and Vmin for different detection/correction schemes inside of a hypothetical cache system. In particular, the two schemes we investigate are (74,64) SEC-DED and (9,8) parity. We chose these two schemes because they are area neutral—they both require the same number of checkbits—for protection of a 64 bit word.

4.9.1 (9,8) Parity Detection

This common detection scheme XORs eight bits to generate a parity bit. Eight sets of these groups covers a 64 bit word. During a read, the bits are XORed again, then compared to this parity bit. If they are different, then the scheme has detected an error. We use an RTL implementation that produces one parity bit for every 8 bits. Synthesis in 28nm LVT process produces the following result:

Encoding delay:	8.0 FO4
Decoding detection delay:	12.1 FO4
Decoding correciton delay:	L2 latency
Encoding energy/op:	0.07 pJ/op @ 1V
Decoding (no error) energy/op:	0.186 pJ/op @ 1V
Decoding (error) energy/op:	allocation cost
Checkbit overhead:	(8/64) = 12.5%
Encoding Gates	8x 9-input XOR gate
Decoding Gates	8x 9-input XOR gate

Table 1: 8x(9,8) parity characterization in 28nm.

The probability that a bit needs correction and a bit error is uncorrectable is shown below for different bitcell error rates, and is plotted in Figure 25.

$$
\begin{aligned}
&\textit{P(exactly-zero-errors-in-eight-bits)} = (1 - ber)^8 \\
&\textit{P(exactly-one-error-in-eight-bits)} = 8 * ber * (1 - ber)^7 \\
&\textit{P(an-eight-bit-chunk-can-detect)} = \\
&\qquad \textit{P(exactly-zero-errors-in-eight-bits)} \\
&\qquad + \textit{P(exactly-one-error-in-eight-bits)} \\
&\textit{P(all-eight-chunks-can-detect)} = \\
&\qquad (\textit{P(an-eight-bit-chunk-can-detect)})^8 \\
&\textit{P(all-eight-chunks-are-error-free)} = \\
&\qquad (\textit{P(exactly-zero-errors-in-eight-bits)})^8 \\
&\textit{P(uncorrectable)} = 1 - \textit{P(all-eight-chunks-can-detect)} \\
&\textit{P(a-bit-is-corrected)} \approx \\
&1 - \textit{P(all-eight-chunks-are-error-free)}
\end{aligned}
$$

4.9.2 ECC: (72,64) SEC-DED

We use an RTL implementation of a odd-weight-column SEC-DED code [14]. Synthesis in 28nm LVT process produces the following result:

Encoding delay:	12.9 FO4
Decoding detection delay:	17.6 FO4
Decoding correction delay:	26.5 FO4
Encoding energy/op:	0.2 pJ/op @ 1V
Decoding (no error) energy/op:	0.875 pJ/op @ 1V
Decoding (error) energy/op:	0.358 pJ/op @ 1V
Checkbit overhead:	(8/64) = 12.5%
Encoding Gates	8x 27-input XOR gate
Decoding Gates	8x 27-input XOR gate 72x 8-input XOR gate 72x 2-input XOR gate

Table 2: (72,64) SEC-DED characterization in 28nm.

Note that the no-error case does not need to localize the error, so the tree of AND gates and the correction XOR gates do not change during normal operation. Any sub-64 bit writes will require a 2-cycle read-modify-write (RMW) operation to regenerate all checkbits. The encoder and decoder are separate.

The probability that a bit needs correction and a bit error is uncorrectable is shown below for different bitcell error rates, and is plotted in Figure 25.

$$
\begin{aligned}
&\textit{P(exactly-zero-errors)} = (1 - ber)^{64} \\
&\textit{P(exactly-one-error)} = 64 * ber * (1 - ber)^{63} \\
&\textit{P(uncorrectable)} = 1 - (\textit{P(exactly-zero-errors)} \\
&\qquad + \textit{P(exactly-one-error)}) \\
&\textit{P(a-bit-is-corrected)} \approx \textit{P(exactly-one-error)}
\end{aligned}
$$

4.10 Comparison of Scheme Effectiveness

The ability of each scheme to correct and detect errors is shown in Figure 25, with an additional inclusion of the SEC-DED+write-through scheme which could correct two errors. As both schemes cover 64 bits at a time, the probability that there is a single error in 64 bits is the same, and therefore the probability that a correction occurs will be the same. However, each scheme responds differently to more than one error. SECDED can detect only 1 error out of 64, while parity can detect up to 8 errors, but only with a probability that all 8 are located in different words and therefore covered by different ECC bits. When parity is combined with write-through, this means that parity can correct up to 8 errors as long as there is no more than one error in each 8-bit block. Note that this analysis assumes nothing about locality of errors. Error locality would diminish the gap between parity uncorrectable rate and SEC-DED uncorrectable rate.

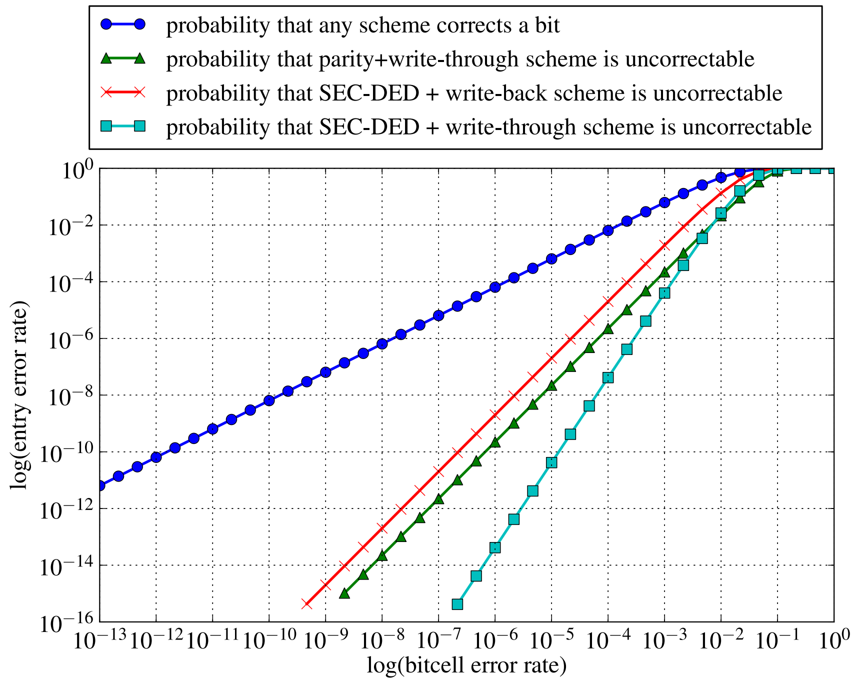


Figure 25: Probability of correction and uncorrectable errors for different error detection and correction schemes.

By knowing these limits, we can determine the amount of energy savings enabled by operating in a finite error region. To operate in a traditional mode of operation, we need the probability that there are any errors at all to be $2.44e^{-5}$ (to get 90% yield on a 32kB array). This corresponds to a bitcell error rate of $3.8e^{-7}$.

When operating in an aggressive mode of operation, limited by uncorrectable errors, we have already found that the bitcell error probability needs to be $3.3e^{-4}$. Figure 26 shows the probability of permanent errors versus supply voltage. As the supply voltage decreases, weak cells begin to start failing, and the probability of an error increases. The increases in acceptable BER for the aggressive mode of operation corresponds to a Vmin reduction of 61mV for a readability-limited design, or a Vmin reduction of 84mV for a writeability-limited design. When considered in addition to guardband removal, ECC shows potential for reducing Vmin. The cost of error recovery cost will be addressed in Section 6.

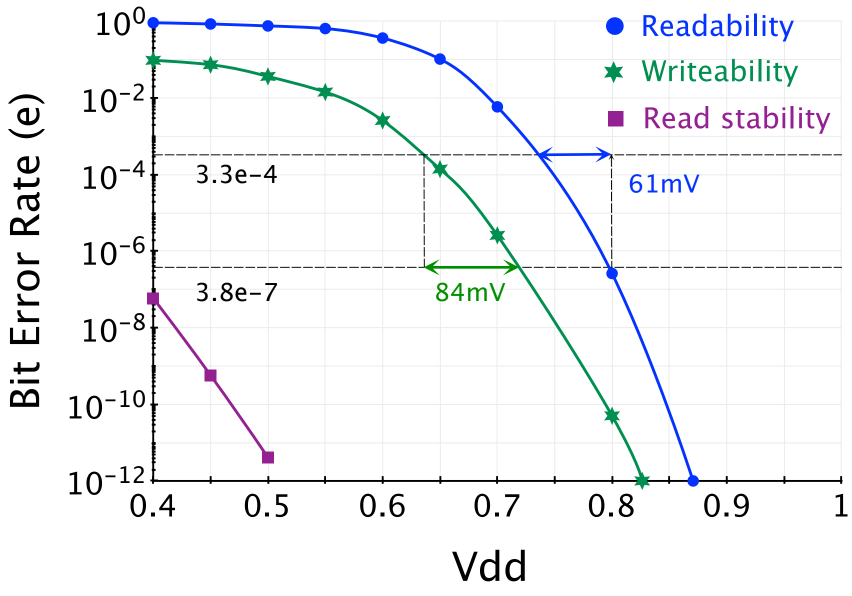


Figure 26: Relationship between BER and Vmin.

4.11 Effect of Cell Sizing

As cell sizing is highly coupled to a particular process and not always an available parameter for the designer, analysis is performed on a commercial SRAM cell with fixed sizing, and area trade-offs are only analyzed between different SRAM cell flavors.

However, subtle changes to cell sizing can have a major impact on failure analysis. Therefore for a complete energy-performance optimization, cell sizing would also be a variable and new options would become available. For example, we could increase the strength of the PG to help readability and writeability, then compensate with a read stability assist. However, cell sizing changes requires close collaboration with manufacturing companies, and is not always a viable option.

In general, we believe that designing a cell to be as stable as possible would be the best approach. It is more efficient to assist the exact row and columns used in an access to improve readability or writeability than to assist all other non-accessed rows and columns to improve stability. As the cell we use is very stable, this assumption is used in our analysis.

4.12 6T Overall Failure Minimization

In general, assist techniques that change the strength of the pass-gate are the most effective—negative bitline, negative GND (by pulling the internal node lower), and wordline changes. For our cell/technology, a combination of short bitlines and negative bitline appear to the best way to modulate the BER for readability and writeability. The cells are very read-stable, so no stability assist is required.

4.13 8T Overall Failure Minimization

For the 8T SRAM, we have found that only writeability needs improvement, so negative bitline would be most effective. For our cell/technology, the 8T pull-down devices are still large, so the cells are very stable and interleaving is possible. However, many 8T cells increase the size of the access and make the pull-down minimum size to improve writeability and decrease area, but don't interleave cells because stability would be greatly reduced. For this case, the addition of a stability assist to enable interleaving would be interesting.

5 SRAM Emin Analysis

5.1 Overview

The most straightforward way to design an energy-efficient SRAM is to design for a low minimum supply voltage (referred to as Vmin). In the last section, we explored how design decisions affect failure rates, and examined how reducing failure rates could lower Vmin. However, Emin (minimal energy per op) and average energy/op over all supplies should be the main design targets, not Vmin. Therefore care must be taken to ensure that the overhead of techniques that enable low-voltage operation do not counteract the energy efficiency gains provided by lowering Vmin. For example, turning on a readability-assist technique will decrease Vmin and decrease energy per operation at low supply voltages, but at high supply voltages, the assist will cause complete bitline discharge which will actually increase energy per operation.

In this section, we examine how design decisions affect energy per operation through the entire DVFS voltage range. Accurate measurements of both failure rates and energy usage are critical for meaningful results.

5.2 Emin

Both dynamic and static energy decrease with decreasing supply voltage. At high supplies, dynamic energy dominates, while at low supplies, leakage energy dominates because cycle time increases exponentially and leakage power is integrated over a longer period of time. There is an optimal supply voltage at which energy per operation is minimized, and enabling operation at this optimal voltage (Vopt) is vital for DVFS systems. In this paper, Emin is the minimum possible energy per operation, and Vopt is the voltage at which Emin occurs. Figure 27 shows this trade-off for a 256kB L2 cache with 5% switching activity, and proposes a Vopt of 650mV. Using this figure, we can understand how activity and design changes affect Emin and Vopt.

If the leakage increases, leakage contributes to a much larger fraction of energy per operation, and supply scaling is much less effective. The leakage curve will shift up, increasing Vopt.

If active energy or the activity factor increase, switching energy is much more important, so supply scaling is more effective (switching energy is proportional to CV^2). The switching curve will shift up, decreasing Vopt.

Unfortunately, SRAMs are likely to fail at voltages well above Vopt. Vmin analysis attempts to decrease Vmin to enable operation at Vopt, but regularly increases energy per operation at high voltages.

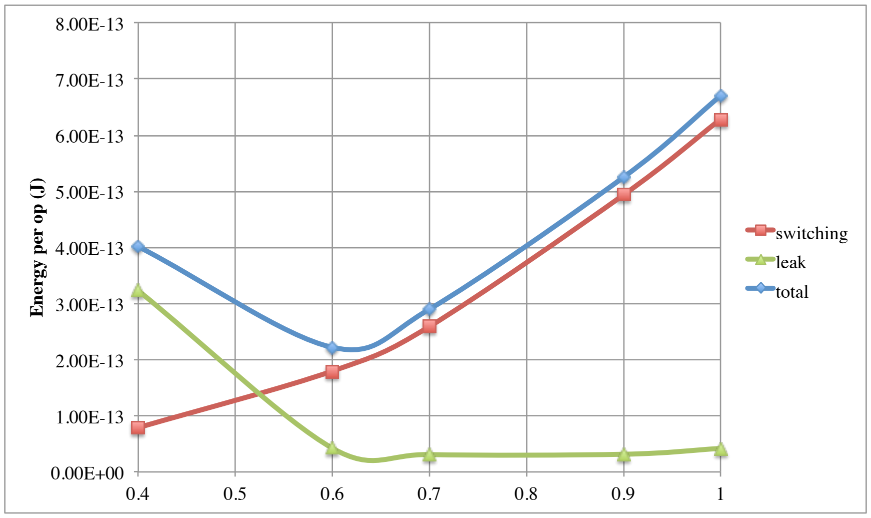


Figure 27: Energy per operation over a range of supply voltages.

An SRAM in a large L1 cache that is accessed every cycle will have different design constraints than a small and rarely accessed SRAM. By knowing the energy for a read, a write, an inactive cycle, and possibly a sleep cycle, for different organizations, assist schemes, and ECC schemes, we can multiply these energies by expected operating conditions (both the proportion of read/write/inactive cycles and expected supply voltages) to come up with an average energy number and performance measurement.

5.3 Energy measurement

We performed energy analysis on both the 6T and 8T arrays discussed later in Section 7. A full schematic level simulation with approximated parasitics for bitcells and long wires measures energy consumed by critical components of the SRAM designs—the decoder, control circuitry, and bitline precharge.

5.3.1 L1 Cache Data Array

An L1 cache has a high activity factor. For a variety of measured SPEC benchmarks, the L1 instruction cache has an activity rate of around 90%, and the L1 data cache has an activity rate of around 30%, where 30% of memory operations are stores and 70% of memory operations are loads. Therefore, switching energy should be minimized, while leakage energy has less importance.

5.3.2 L2 Cache Data Array

An L2 cache has a very low activity factor, and generally is very large, so even active operations only access a small portion of the total structure. Therefore minimizing leakage is critical. A variety of benchmarks suggest an activity rate of 2% for a write-back cache, and 10% for a write-through cache, so the optimal design point will be dependent on the L1 write policy.

5.4 Energy per read operation

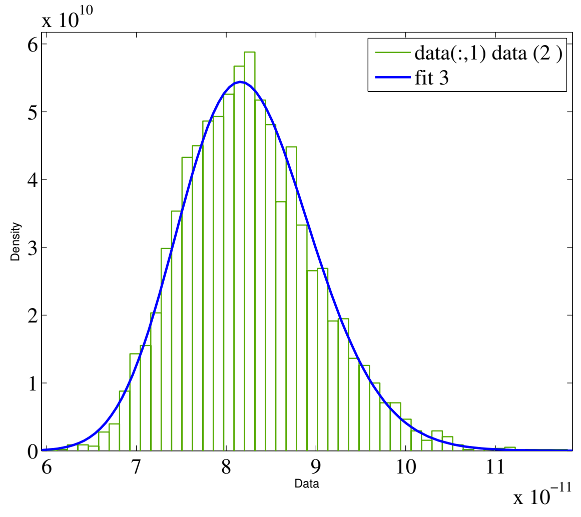
5.4.1 Nominal

For read operations, variability has a major effect. The strength of the pull down and access device strongly affects how much of the bitline gets discharged and therefore how much energy is used. However, these energy numbers get averaged over many bitlines, and detailed Monte Carlo results that account for variability are identical to results from a single simulation at nominal value.

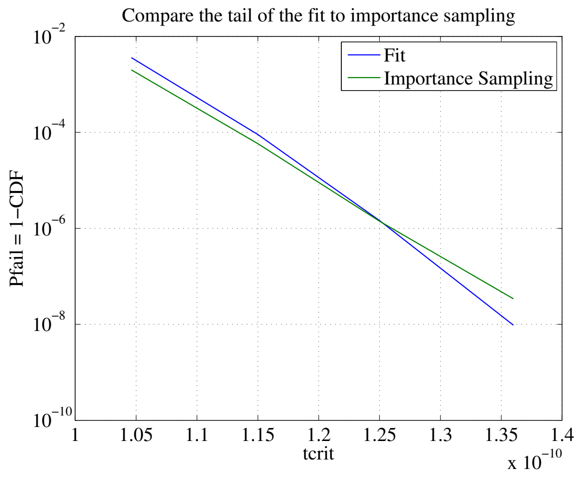
Array organization has a minimal effect on read energy. Because the wordline turns on every cell in a row, every single bitline pair will experience a read. Interleaving will decrease bitline capacitance, but also increase the number of bitlines proportionally, so there is very little change in read energy. Making smaller sub-arrays (shorter bitlines, but keep the degree of interleaving the same) could potentially decrease energy. However, to meet the same size specifications, more total sub-arrays will be needed, and the area and energy overhead of extra periphery circuitry will diminish these savings. The best way to improve read energy is to either use hierarchical wordlines and bitlines, or ensure that bitlines discharge only the minimum amount of voltage required for a successful read by using a low offset sense amplifier and careful timing.

5.5 Optimal read timing

Readability requires the weakest cell to discharge the bitline by a voltage equal to the sense amplifier offset—typically 50-100mV. However, this means that the average cell discharges much more than needed. To save energy, we should turn off the wordline and turn on the sense amplifier immediately after the weakest cell has produced the required voltage drop—a time we call t_{crit}. Using Monte Carlo, we can identify the PDF of t_{crit}, as shown in Figure 28a. and can be approximated with a lognormal distribution. This approximation has been verified using importance sampling, as shown in Figure 28b.



(a) PDF of $t_{crit}$with fit.

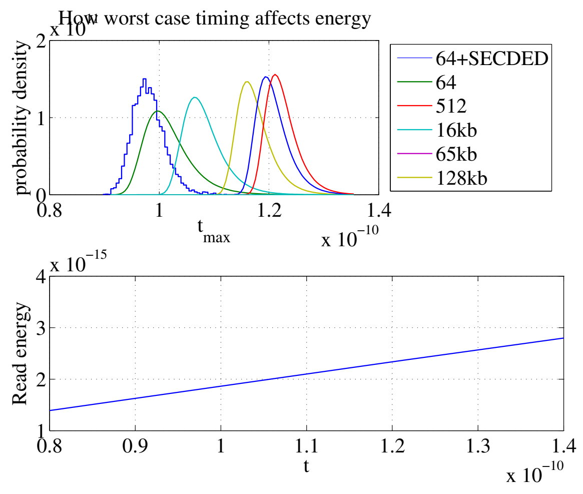


(b) Lognormal Fit.

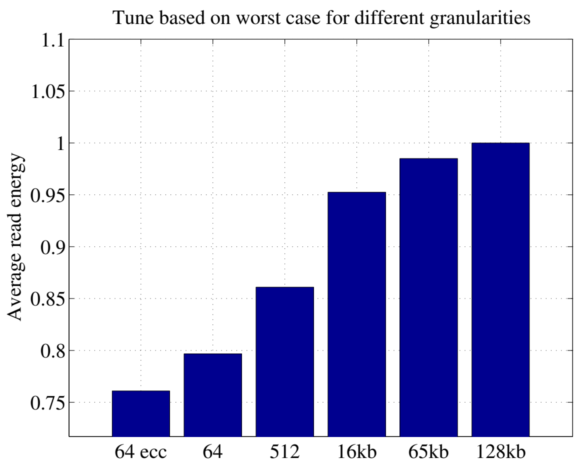
Figure 28: Modeling t_{crit}at 1V.

Larger t_{crit} means that all of the bitlines discharge for longer, which means energy per operation is higher. Ideally, we would like to choose different t_{crit} for different rows. Tcrit will be smaller if we are only reading

64 cells than if t_{crit} needs to cover the worst cell in an entire cache. Using the extreme value distribution and the PDF of t_{crit} generated earlier, we can plot the PDF of the worst case cell out of a group of n cells. The extreme value distribution can be validated by numerical methods. One further optimization assumes that ECC can be used to avoid waiting for the worst cell, and instead turn off the wordline after the second to worst cell has finished. The results of t_{crit} for the worst cell in different n-bit groups is plotted in Figure 29a. The top plot shows t_{crit}, and the bottom plot shows total energy dissipated for each t_{crit}. Figure 29b shows the energy results of this analysis. Optimizing t_{crit} for a given row (512 bits) saves about 15% in read energy. Using ECC saves an additional 5% energy. However, achieving these gains requires some sort of memory per region. Fuses or registers could be used to add additional delay to the weak rows. However, the absolute difference in t_{crit} between a 64 bit region and an entire memory is only 20ps (1FO4), so trying to optimize with different settings for different groups is impractical. Instead, these results suggest that precise tuning of the pulse width is much more important.



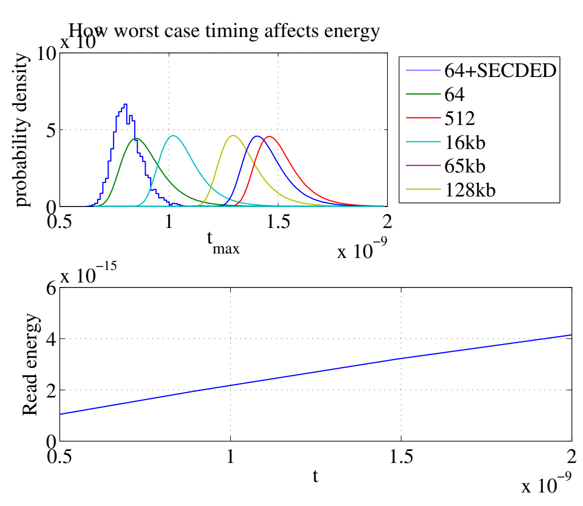
(a)



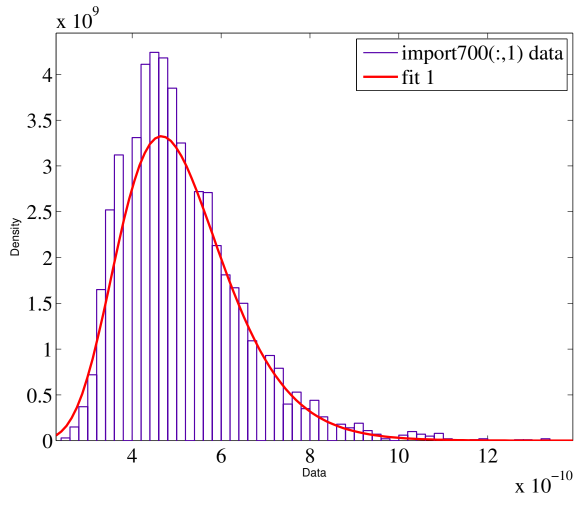
(b)

Figure 29: Relationship between t_{crit} and energy at 1V.

Another interesting result of this analysis can be seen from rerunning these experiments at 700mV. A plot at 700mV is shown in Figure 30a. For the 128kB case, the read energy is actually higher than for the 1V case, suggesting that a supply voltage decrease actually increases energy per operation. There are a few reasons for this. First, sense-amp offset is relatively constant over all voltages (even though it does decrease slightly with voltage scaling), meaning that energy will decrease linearly, not quadratically. Second, the tail of the t_{crit} distribution increases dramatically, as shown in Figure 30b, to be over 3x the mean t_{crit} (was only 1.5x the mean at 1V). So the linear savings in energy is counteracted by the increase in variance of t_{crit}.



(a)



(b)

Figure 30: Relationship between t_{crit} and energy at 700mV.

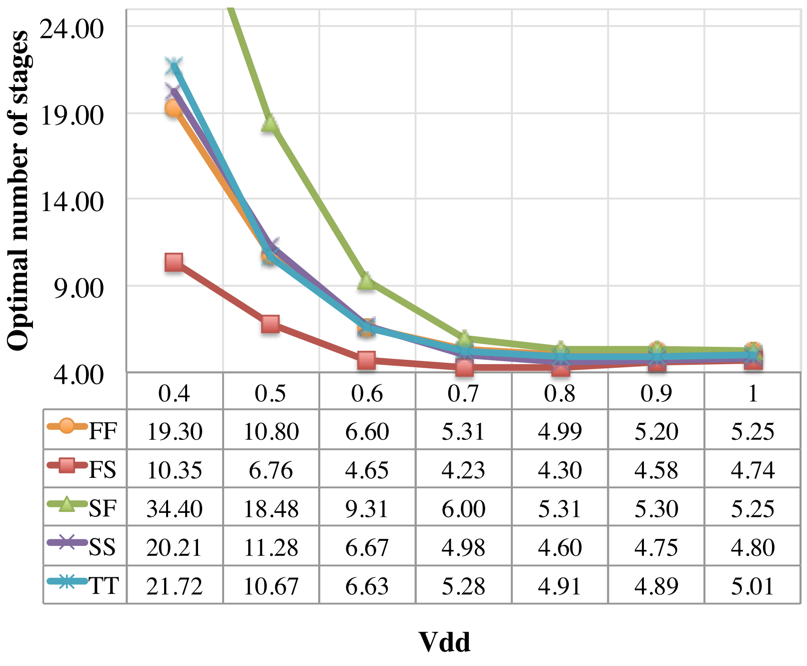
Therefore, for a well-designed system, supply scaling has limitations, and is more useful to save system energy (if the data array must be on the same supply as a processor for example). It should be noted that this analysis assumes ideal timing—in reality, RC wire delay at high voltages causes the last cell on the row to take much longer than the first cell, extending t_{crit} dramatically. At lower voltages, R of the wires becomes less of a factor, and t_{crit} can be more ideal. In addition, all periphery circuitry benefits from supply scaling, and periphery makes up for about half of switching energy if we assume optimal bitline timing.

5.6 Generating optimal read timing

To save read energy, turning off the wordline exactly at t_{crit} is vitally important. Using importance sampling, we have found t_{crit} for all corners and all voltages. Figure 31 shows how many RVT FO4 delay stages would be needed to generate t_{crit}. First, no fixed number of stages works across a large voltage range, and each corner requires a different number of fixed stage. By potentially switching these settings through a BIST to set to the correct corner, then changing configurations for different voltage regions, t_{crit} can be closely approximated.

However, a better scheme would automatically track corners and voltages. Importance sampling showed us that t_{crit} is caused by about 2.5σ variation on both the pull-down and pass-gate, or 5σ on either the pull-down or the pass-gate. Therefore, to emulate this, we propose a replica cell which either has a Vdd of Vdd-100mV or a GND of 50mV. The new Vdd or GND changes the Vgs of the NMOS transistors and emulates the σ variation point that causes failure. The GND option will change the Vgs of both the pull-down and the pass-gate, so it matches the first case, while the VDD option will change the Vgs on only the pull-down, so it will match the second case. The 50mV (or 100mV for the Vdd case) needs to stay constant over all supply voltages (because the threshold stays fairly constant over all supply voltages).

In order to use this replica to generate a timing signal, we use a replica column with 12 cells on to generate a quick, full-swing, transition. The delay between the clock and this quickly falling bitline is multiplied to generate the wordline turn-off signal. Figure 32 shows the multiplier number needed to generate $t_{crit}$for GND = 50mV. This scheme tracks t_{crit} over all voltages and corners. Decreasing Vdd by 100mV yields



	0.4	0.5	0.6	0.7	0.8	0.9	1
● FF	19.30	10.80	6.60	5.31	4.99	5.20	5.25
■ FS	10.35	6.76	4.65	4.23	4.30	4.58	4.74
▲ SF	34.40	18.48	9.31	6.00	5.31	5.30	5.25
✕ SS	20.21	11.28	6.67	4.98	4.60	4.75	4.80
✱ TT	21.72	10.67	6.63	5.28	4.91	4.89	5.01

Figure 31: Number of delay stages need to generate t_{crit}.

similar results.

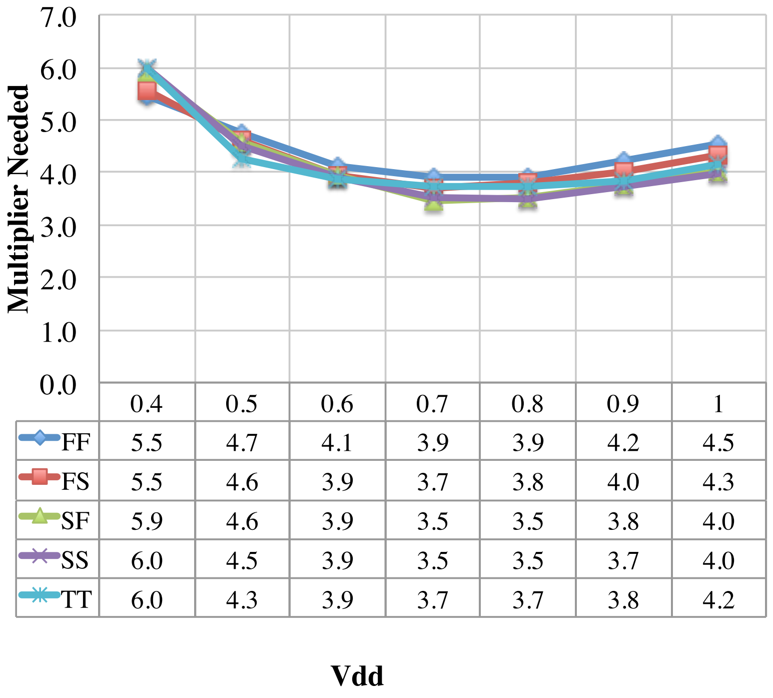
5.7 Hierarchical vs. timing

Hierarchical bitlines provide a partial solution to the problem of waiting for the most extreme cell to finish. Local bitlines (with smaller total capacitance) swing is limited to be Vdd, placing a bound on energy wasted by waiting for the worst cell. However, hierarchical bitlines actually add more total capacitance, because of the need for an extra global bitline and local drivers. To minimize energy, the global bitlines should be low swing. However, this just extends the timing problem to the global bitlines, as weak cells will take longer to transfer their results to the global bitlines. Gating global bitline drivers such that they only turn on after local results are finished can remove this problem, but introduces a large distributed timing signal with a large fanout that will diminish energy savings.

5.8 Energy per write operation

Write energy will depend on array organization, as only n bitlines need to be fully discharged (where n is the word length) and all other bitlines will experience a read with only a partial discharge. Therefore energy consumed during a write operation will come from both bitlines that are reading and bitlines that are writing.

For our baseline organization, we assume the organization shown in Figure 13. Therefore write operation energy equals 64 times column write energy plus 448 times column read energy. Column write energy will always be higher as the bitline must completely discharge.



	0.4	0.5	0.6	0.7	0.8	0.9	1
◆ FF	5.5	4.7	4.1	3.9	3.9	4.2	4.5
■ FS	5.5	4.6	3.9	3.7	3.8	4.0	4.3
▲ SF	5.9	4.6	3.9	3.5	3.5	3.8	4.0
✕ SS	6.0	4.5	3.9	3.5	3.5	3.7	4.0
✳ TT	6.0	4.3	3.9	3.7	3.7	3.8	4.2

Figure 32: Multiplier needed to generate t_{crit} for 12 replica cells with cell GND = 50mV.

5.8.1 Optimal write timing

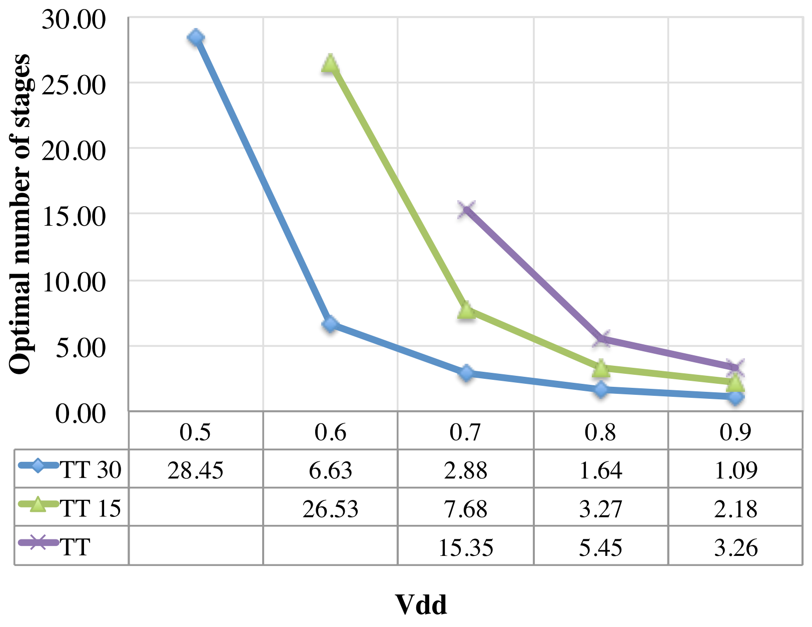
Unless the degree of interleaving is small, write operations can benefit just as much from optimal timing as read operations because the majority of columns are inadvertently reading. Completing the write operation as quickly as possible can enable energy savings. Therefore, assists such as negative bitline could be used even when not completely needed to improve read speed and therefore decrease energy per operation.

5.8.2 Generating optimal write timing

Using importance sampling, we have identified write operation t_{crit} over a wide range of supply voltages and for different degrees of negative bitline assist. Just like the read case, a typical delay line does not match corners or voltages well, as shown in Figure 33.

However, for the writeability case, a replica is much less useful. Any timing of less than 5 stages is impractical, as such a narrow pulse would not have time to propagate to the last cell. Then below a certain voltage, the required time jumps very quickly to be beyond the cycle time of a processor. This behavior is caused by the fact that most writeability failures are static failures, and wordline pulse width has much less of an effect on error rate.

A replica that tracked writeability would need to track a weak pull-up transistor. However, this becomes difficult to do when a cell is embedded inside an array, because the only access is through a pass-age, which cannot pull high. Therefore trying to time how long it takes a pull-up to pull a bitline up will be limited by the threshold voltage of the pass-gate.



	0.5	0.6	0.7	0.8	0.9
TT 30	28.45	6.63	2.88	1.64	1.09
TT 15		26.53	7.68	3.27	2.18
TT			15.35	5.45	3.26

Figure 33: Number of delay stages need to generate t_{crit} for write operations.

5.9 Energy per inactive cycle

During an inactive cycle, leakage should be the only contribution to energy. Clock gating for the array can ensure that no switching activity occurs.

5.10 Assist Overhead

Any readability assist will increase read current. Without hierarchical bitlines or optimal timing, this would cause a substantial energy overhead as every cell (not just the worst case cell) would experience extra read current.

Ideally, read assists would be applied directly to the read column to avoid extra energy in the half-select columns. However, techniques such as wordline boost will increase read current in all half-selected current, making this technique very expensive. Negative GND is difficult to make worthwhile as something must carefully regulate a negative voltage while sinking the read current of every cell on the bitline, and an entire row of cells internal voltages must be moved. In this case, negative bitline incurs a very small energy overhead, because it only effects 64 columns. VDD droop is more expensive, because more capacitance must be change by more voltage.

If we set the assist degree at design time, we would have to examine the frequency of operations at different supply voltages, and determine an optimal assist degree that would minimize total energy for a total number of operations. However, greater energy efficiency can be achieved if we vary the assist degree for different supply voltages at run-time. This would disable the assist at high supplies and prevent the additional energy overhead from the unneeded bitline discharge, but enable the assist at low supplies to decrease Vmin.

5.11 Error Monitoring and Response Cost

To operate in a finite error region, some scheme must be used to either identify if errors will occur before they happen (BIST), or catch and repair errors once they do (ECC). Both schemes have their own advantages and disadvantages as well as very different overheads.

5.11.1 BIST

BIST costs are dependent on the complexity of the algorithm, and how often the test is ran. BIST algorithms themselves involve little area, especially if they are shared among many arrays. Deciding how much information to store also dramatically affects area. For example, remembering area locations takes large amounts of area because information a single bit requires a many-bit address to be stored. Instead, a BIST system should interpret results often and merely remember error counts for each type of error.

5.11.2 ECC

The cost of encoding and decoding for the parity and SEC-DED schemes have been shown in Table 1 and Table 2. In addition to these costs, the need to read and write checkbits causes a large energy and area overhead. In particular, for the schemes suggest previously, a 12.5% energy overhead is incurred every cycle.

ECC suffers from a similar trade-off as assist techniques—more powerful ECC coverage will improve energy efficiency at low voltages by decreasing Vmin, but incur overhead at high supplies as checkbits are unnecessarily encoded, written, read, and decoded. An ideal system would only enable higher levels of ECC at low supply voltages, but the checkbits would need to be generated before entering low voltage operation, and this require either rewriting the entire cache or always encoding and writing checkbits at high supplies (but not reading or decoding them).

5.12 Adaptivity

Many techniques that allow for operation at low voltages generate additional overhead at higher voltages (when they are not needed). In this situation, average energy per operation is dependent on how often an array operates at each supply voltage. Different activity predictions may predict different optimal energy results.

Ideally, array assist techniques would adapt themselves dynamically based on the current supply voltage, which would prove useful when running a diverse range of programs. There are two general schemes to accomplish this. The first, and ideal, case involves a continuous-replica that changes settings directly based on the supply voltage. Then, no matter how quickly the supply voltage changes or if there is a ripple on the supply, settings are automatically changed to operate at the current supply. A second scheme involves separating the voltage range into discrete regions. Knowledge of the regions dictates assist settings. For example, different DC-DC configurations dictate different regions of operation. In both cases, the scheme generally requires tuning.

It is possible to only turn on ECC in regions where there is a danger of error occurrence, even though many times ECC should be left on for soft-error protection anyway. However, transitioning turning on ECC requires a read-modify-write operation for every piece of valid data in the memory, because the checkbits were not originally written.

6 Optimizing SRAM for caches

A thorough understanding of failure rates and energy in SRAM enables an accurate study of energy efficiency and resiliency for complex systems. One very important system is the on-chip memory hierarchy for microprocessors.

In particular, L1 cache design contributes significantly to the performance and energy consumption of microprocessors due to L1's large proportion of die size and high activity factor. Voltage reduction reduces energy per operation, however, increased process variability in modern technology nodes causes an exponential growth in SRAM failure probability for linear voltage reduction, necessitating some form of error correction. In this case study, we compare the implications on energy, area, performance, and error rate for two methods of error correction—a write-back cache with SEC-DED and a write-through cache with parity detection—and identify system factors (such as cache size and latency) that define the optimal solution. Additionally, we explore potential benefits of running a processor at finite error rates (aggressive voltage scaling) by analyzing recovery costs and system energy. Energy measurements come from full transistor-level simulation of functional 28nm SRAM designs over varying operating conditions, and hit/miss rate measurements come from a cache simulator embedded into the RISC-V ISA simulator [15].

6.1 Necessity of Error Correction

For this study, we assume that both the L1 and core share the same supply voltage—therefore, reducing the supply voltage of the SRAM enables system energy savings, not just memory energy savings. There are a few reasons we believe that this assumption is valid. First, level shifting requires a delay of around 100ps in our 28nm process. For a target core speed of 25 FO4 (around 500ps period), this penalty represents a large addition to the critical path (integrating level shifters into the row drivers and sense amplifier can decrease this overhead). Second, multiple power grids increases the complexity and decreases the effectiveness of the power grid. On-chip DC-DC converters have been proposed to decrease this cost, yet high efficiency converters produce an output supply voltage ripple that will not match between the L1 and core, which can cause dangerous situations where the L1 is much slower than the core and critical paths are violated. Error correction or detection for the SRAM in the L1 cache allows for the removal of this guardband because a finite number of errors can be allowed to occur—providing substantial energy savings. Additionally, error correction is already required in many commercial designs to deal with potential soft errors. However, it is unclear which particular error correction technique most effectively provides the required resiliency while minimizing overhead costs. Therefore, we perform an in-depth analysis of cache write policy and its effect on error tolerance, performance, area, and energy.

6.2 L1 Correction Methods

In general, there are two different ways to tolerate errors in the L1 cache—a write-back cache with single error correction, double error detection (SEC-DED) or a write-through cache with parity detection that corrects by retrieving data from L2. A write-back cache cannot use merely detection because there would be no way to correct dirty data, while a write-through cache could potentially use SEC-DED—where 1 bit errors are corrected locally and double bit errors are corrected from the L2.

This study has two main goals. Assuming that error resiliency is needed on the L1 cache, our first goal is to identify the important factors needed to determine what write policy minimizes energy per operation on a DVFS core and memory system. Because energy is our primary metric, we use accurate energy measurements of real designs instead of relying on models, and because results are very sensitive on assumptions, we focus on intuition and sensitivities rather than absolute numbers. Our second goal is to understand the effect of supply scaling and finite error rates to determine whether aggressive voltage scaling (operating in a region with a high error rate) is worthwhile.

In order to accomplish these goals, we need accurate measurements of two pieces—hit/miss rates and energy costs—and to provide these, we have built a cache simulator for the RISC-V ISA simulator and made transistor-level energy measurements of all system components.

6.3 System Assumptions

We use the Rocket RISC-V core developed at UC Berkeley as platform because it embodies the properties of a modern mobile in-order processing core. Energy analysis from a recent chip suggests that the instruction cache takes 12% of total core power while the data cache takes 15%, and most cache energy is dissipated in the SRAMs.

The L1 SRAM is optimized for speed so that it can achieve a single cycle latency. The data cache is built from 8kB sub-arrays of 8T cells, allowing for 1 read and 1 write operation of 64 bits per cycle. At 1V, a write operation takes 2.9pJ and a read operation takes 2.0pJ. Reads can only occur at a 64-bit granularity, and write masking allows writes to happen at 8-bit granularity. To avoid increasing the latency, all ways are read in parallel for load operations.

The L2 SRAM typically uses much denser cells that are even worse than cells in the L1 at operating at low supply voltages, so we assume that the L2 resides on its own supply voltage. The data array is built from 8kB sub-arrays that are interleaved 8:1 and access 64 bit words at a time. Read masking allows for only 32 bits to be read. Write masking is not implemented to save area as no sub-double word stores are needed. At 1V, a typical write operation takes 7pJ and a typical read operation takes 3.9pJ, and can maintain 1Ghz operation in 28nm. The activity of the L2 is largely dependent on the voltage of the L1, so we assume the L2 voltage scales at the same rate as the L1, but stops at a different Vmin. A latency of 7 cycles for L2 accesses is assumed based on a consensus of various recent publications, where a data SRAM access will take 2 cycles. The L2 is protected by ECC because it is a write-back cache, and ECC granularity is 128 bits to save on checkbit overhead. Larger granularities of ECC save checkbit area as shown in Table 3, yet increase encoding and decoding delay slightly, increase energy linearly, and increases RMW operations for write-through caches with coalescing write buffers. Because the interface between the L1 and L2 is 128-bits wide, we have chosen a 128-bit granularity because it simplifies design and represents a fair compromise between area overhead and encoding/decoding complexity and energy. We assume a total L2 size of 256kB. We could easily extend our results to include the L2 to main memory interface.

# of data bits	# of 1s in H matrix	Checkbit (overhead)	n-input XOR	Delay
64 (8B)	216	8 (12.5%)	27	240ps
128 (16B)	481	9 (7.03%)	53	280ps
256 (32B)	1050	10 (3.9%)	105	320ps
512 (64B)	2241	11 (2.1%)	204	330ps

Table 3: SEC-DED Granularity Comparison.

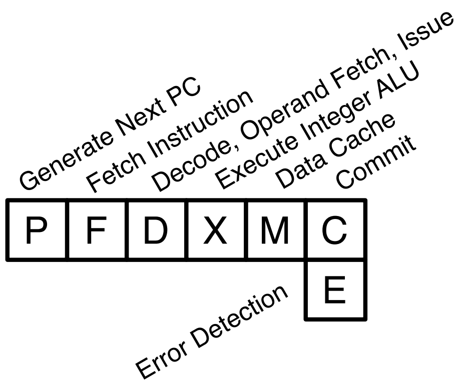


Figure 34: Pipeline diagram with an added error correction stage.

6.3.1 ECC Schemes

From Section 4.10 and Figure 25, we understand the correction capabilities of each scheme, and the BER range enabled with both techniques.

In context of a cache, the parity scheme will not require a read-modify-write (RMW) for sub-word writes because parity will be assigned as 1 bit for every 8 bits, so only the parity bits for the written word need to be changed. For ECC, sub-word writes will require a RMW operation because 8 bits of checkbits cover 64 bit granularity. However, for the parity case, RMW operations are instead performed inside the L2, but the number of RMW operations will be diminished because of the coalescing write buffer.

Both schemes incur a delay to encode checkbits during stores and to decode checkbits and detect/correct errors on loads. A typical in-order low-power core such as Rocket typically has a critical path of 25FO4 (or around 500ps) through the L1 cache for loads. Therefore waiting for detection to complete for either scheme would increase the operating frequency by an unacceptable amount, and we propose placing all decode logic in a subsequent pipeline stage for loads. Errors will only be detected during load operations, regardless of whether the actual failure happened during writing, retention, or reading. For loads going to the pipeline, as shown in Figure 34, on errors, incorrect data will have already been written to the register file and forwarded into the execute stage, so the pipeline must be flushed, incurring a cycle penalty of X cycles. The incorrect data in the register file will be overwritten when the offending load instruction replays. For loads going to the L2 in our system, there is a queue between L1/L2 because both domains are asynchronous, so data can be corrected in the queue the following cycle.

6.4 Cache Simulator

A cache simulator was written in C++ to interface with the RISCV ISA simulator. The cache can be parameterized for write-through or write-back, along with size, associativity, and line size. Counters are maintained for relevant operations to allow energy calculation at the end of simulation.

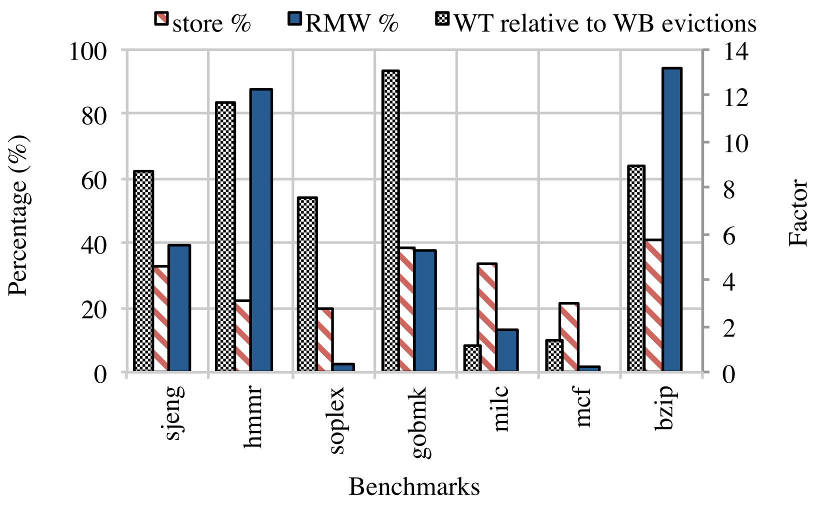


Figure 35: Important characteristics for a variety of benchmarks.

6.4.1 Write-through

The write-through cache has a line-sized coalescing write buffer implemented as a first-in first-out (FIFO) queue. All stores result in the data being added to the write buffer. If the store is a hit, the L1 is also updated with the data and computed parity bit When a store occurs to a full write-buffer and the associated line is not in the write-buffer, the oldest line is sent to L2. A 64 bit dirty mask, one bit for every byte, is sent along with the line so the L2 can correctly store the data. A read-modify-write will occur at the L2 if the data at ECC granularity is not completely dirty. Otherwise, the data can just be written to L2. On a load miss, the cache line is retrieved incremented) from L2 and the data and computed parity bits are stored in L1. A load hit returns the data and checks the parity bits.

6.4.2 Write-back

Any load will eventually read from L1 and check ECC, and any store will eventually write to L1 and compute ECC. On a load or store miss, the write-back must first retrieve the line from L2 and compute the ECC bits before writing to L1. If dirty data is being evicted by a random eviction policy incremented), it must be read from L1, have ECC checked, and be sent to L2. Because ECC on L1 is at 64-bit granularity, any store less than double-word must be a read-modify-write operation where ECC is both checked and computed. For any size load, the double-word must be read to check ECC before returning the data.

6.5 Design Intuition

Figure 35 shows the results from our ISA simulator for a subset of the SPEC benchmark suite. Lower store % (as a proportion of total memory operations) helps the write-through because stores are more expensive in the write-through case and loads require more energy in the write-back case because ECC decoding takes more energy and a full word must always be read. A higher RMW percentage hurts write-back caches. The number of write-throughs past the end of the write buffer can be many times the number of evictions in a write-back cache. Most programs have about twice as many loads as stores.

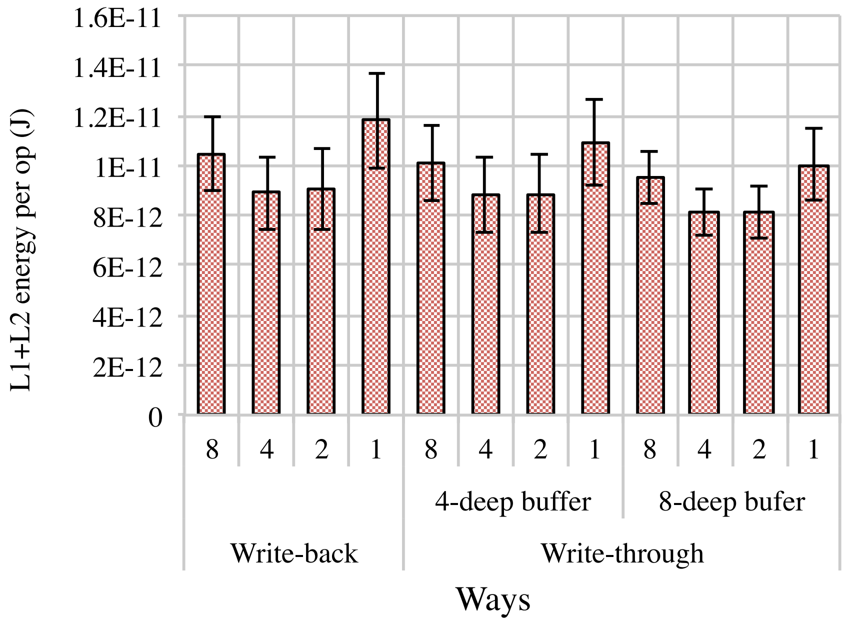


Figure 36: Energy per operation for different cache configurations.

6.6 Write-through

A cost is assigned to each counter value to obtain total energy usage for the memory. Each size of load hit must read the respective number of bits and check parity bits, and each way must be loaded. Each size of store hit must write the respective number of bits and compute parity bits. The write-through of stores from the write buffer has communication costs and varying costs depending if L2 is a write or read-modify-write. Lastly, a load miss requires reading a line from L2, sending to L1, and writing data and parity bits to L1.

Figure 36 shows how design decisions effect memory hierarchy performance for a write-through cache. Coalescing write buffer depth has little effect on energy (preliminary results showed that no buffer at all drastically harmed performance). Also, cache size has little effect beyond 8kB. More ways reduce conflict misses, but every load operation needs to read more ways so an optimum is found at 2 or 4 ways, depending on the benchmark.

6.7 Write-back

The same cost analysis is performed for the write-back cache. All loads require reading a double word to be able to check ECC for every way. To correctly compute the ECC, all double-word stores can just write, but sub-double word stores must first read the double word—a read-modify-write operation. However, RMW operations can wait for tag check and do not need to speculatively read all ways. If an eviction occurs, the line must be read from L1 with ECC checked and sent and written to L2. Lastly, a load miss requires reading a line from L2, sending to L1, and writing data and parity bits to L1.

Figure 36 shows how design decisions effect memory hierarchy performance for a write-back cache. Again, increasing size has little effect on energy, and 2 or 4 ways are optimal.

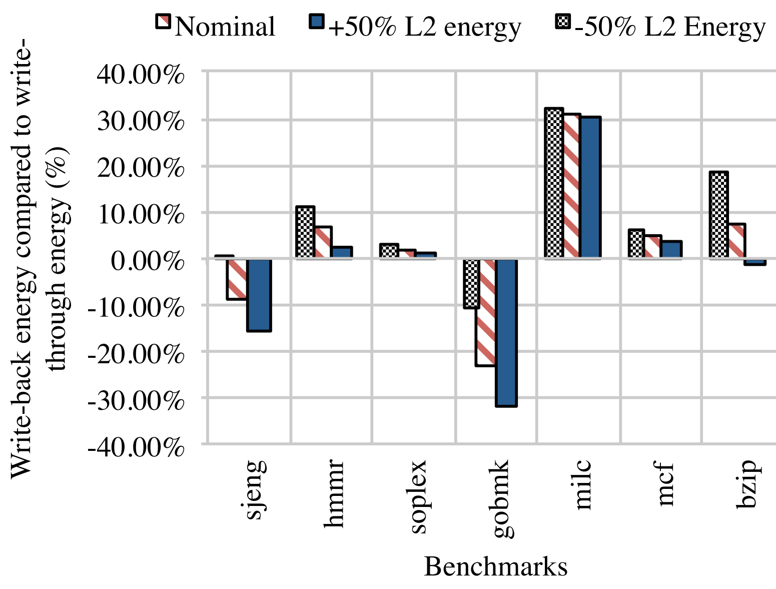


Figure 37: Comparison of write-back and write-through caches for differing L2 energy.

6.8 Sensitivity of model to assumptions

Our results are more dependent on benchmark characteristics than any other factor. Figure 37 compares cache system energy per operation for the write-through and write-back cases with different L2 energy costs. Either case consumes very similar energy per operation, and benchmarks are much more of a differentiating factor than L2 energy costs.

6.9 Effect of voltage scaling

At high voltages, there are never errors, so the energy of different schemes depends only on a traditional write-back vs. write-through trade-off. However, as the supply voltage decreases, the SRAM error rate increases as shown in Figure 26. For our SRAM, readability is the limiting factor at low supply voltages.

If we assume there are a finite number of errors, then error recovery costs become a concern and a possible differentiator for both schemes. When we allow finite errors, we can lower the supply voltage and make the common case energy-efficient. The main concern with this strategy is that we must still avoid uncorrectable errors. Figure 38 can be used to understand this limit. The two lines shown correspond to the parity scheme. If the L1 cache is 32kB and entries are 64 bits, there are 4096 entries. For 90% yield, on average we can allow the probability that there are uncorrectable to be $1/4096 * 1/10$ or approximately $2.44e^{-5}$. Using Figure 38, we can relate this entry error rate to a bitcell error rate of $3.3e^{-4}$. At this bit error rate, the probability that we need to correct an error is then equal to $2.11e^{-2}$. Therefore operating in the most aggressive possible error mode without experiencing any undetectable errors results in about 2% of load operations requiring a correction operation.

Because error rates are now finite, we need to include their effect into the energy per operation calculation of the processor. First, we calculate the correction energy. For the write-back case, the error is corrected locally and has virtually no effect on system energy, so we assign each error no extra correction cost. For the write-through case, each error requires an L2 line load and L2 communication to retrieve the data. This data

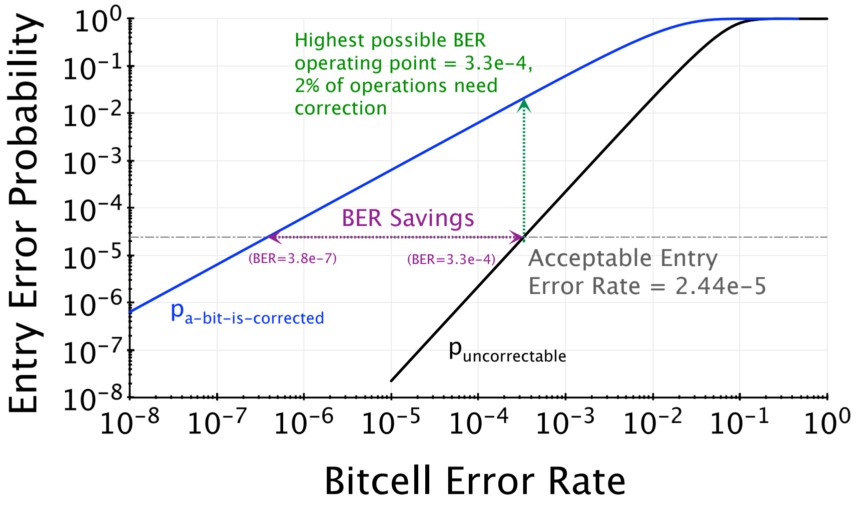


Figure 38: Translating entry error requirements to acceptable bitcell error rate for a single error correcting scheme.

No error correction	725mV
SEC-DED + Write-back	652mV
Parity + Write-through	635mV
SEC-DED + Write-through	619mV

Table 4: Potential Vmin reduction for different degrees of error correction.

could be optionally written to the L1 as well, however this decision relies on knowledge of fault longevity (if the cell is always not going to work, there is no reason to write back). One possible optimization is to send less than a single line from the L2. Second, each scheme increases CPI because of correction latency. For the write-back case, we assume a latency of 4 cycles to allow for a pipeline flush. For the write-through case, we assume a latency of 7 cycles (the L2 load latency). These idle cycles add energy of idle cycles to the total energy. These costs are incurred by calculating the probability of a correction for each supply voltage, based on the relationship determined by Figures 38 and 39.

Using this method of analysis, we can calculate system energy at different supply voltages, accounting for a finite error rate and energy penalty. Remember from Figure 25 that the probability of uncorrectable errors varies between each scheme, so therefore Vmin will be different for each scheme, as summarized in Table 4. However this is a minor effect—SEC-DED with write-back can achieve a 652mV Vmin while parity with write-through can achieve a 635mV Vmin. The third potential scheme, SEC-DED with write-through to correct two errors, can achieve a Vmin of 619mV, suggesting that the overhead required will not be worthwhile because it will increase overhead at all supply voltages to enable a slightly lower Vmin. In general, more complication correction schemes yield diminishing results because decreases in bitcell error rate decrease for higher numbers of errors. In addition, these schemes require a much larger constant overhead due to more complicated correction schemes and extra checkbits. For this reason, we have limited our implementation to single bit error correction. The probability that a bit is corrected remains the same except for the SEC-DED + write-back case which needs to correct both single and double errors.

Figure 40 summarizes the difference both schemes for a collection of benchmarks. Note that write-through becomes very expensive at high error rates because correction requires reading from the L2 instead of re-

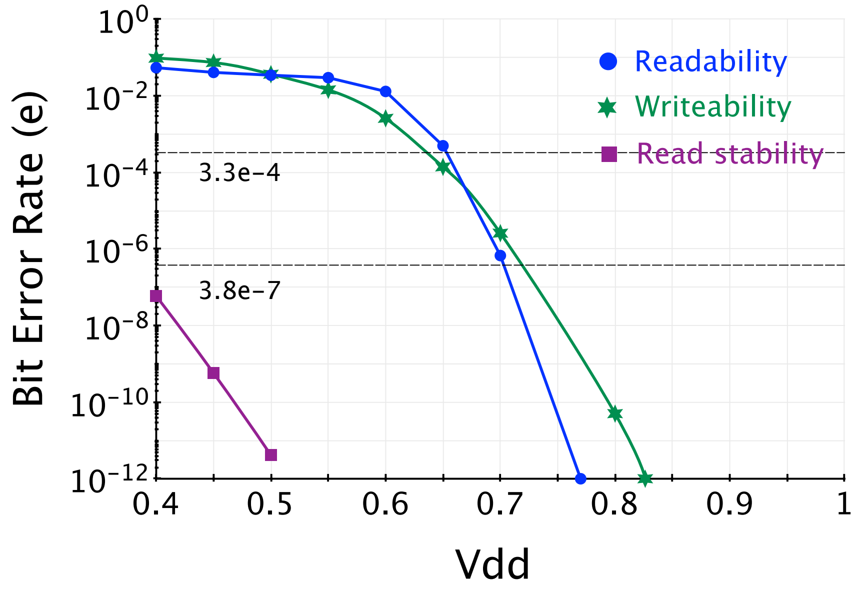


Figure 39: Translating acceptable bitcell error rate increase into Vmin reduction.

pairing locally, but Vmin is only 650mV, so this effect is unimportant. By the time the error rate becomes significant enough that a large amount of energy is spent on correction, the probability that there are uncorrectable errors becomes too high and this operating region becomes unusable. So in reality, correction cost does not differentiate between the schemes, and only comparison at error free operation is needed.

This result tells us that ECC will allow us to operate at a minimum energy point of 21pJ/op at 650mV (for this system). Without ECC and operating without any margin, the minimum energy point would be 26.1pJ/op at 725mV. Assuming 100mV of margin, the safe system would use 34pJ/op at 825mV. ECC allows for the removal of this margin because it can sense Vmin based on error rates. Together ECC saves around 20% of energy per operation.

Because ECC will incur a 12.5% overhead at all supply voltages, unless the core always runs at the minimum energy supply, this technique might not decrease overall energy per operation for the core. However, there are a couple reasons that ECC can be beneficial. First, because the core and L1 are assumed to be tied to the same supply, the 12.5% overhead applies only to memory energy, while the 17% savings applies to both the core and memory energy. Second, if correction is needed anyway to deal with transient and soft errors, and the rate of these errors is much smaller than transient errors, we can save 20% energy by operating at an aggressive supply.

6.10 Effect of performance on energy

Figure 40 shows that finite error recovery cost is almost irrelevant for an aggressive DVFS system. Instead, we have determined that a more important factor is simply performance.

Figure 41 shows the effect of CPI increase on system energy at 0.8V. A simple analysis says that we increase energy proportionally to the increase in cycles. We assume that a stall cycle only requires half of the energy of a normal cycle, and a more accurate estimate would require processor simulation.

However, in a DVFS system, voltage is set such that a total number of operations can complete in a constant

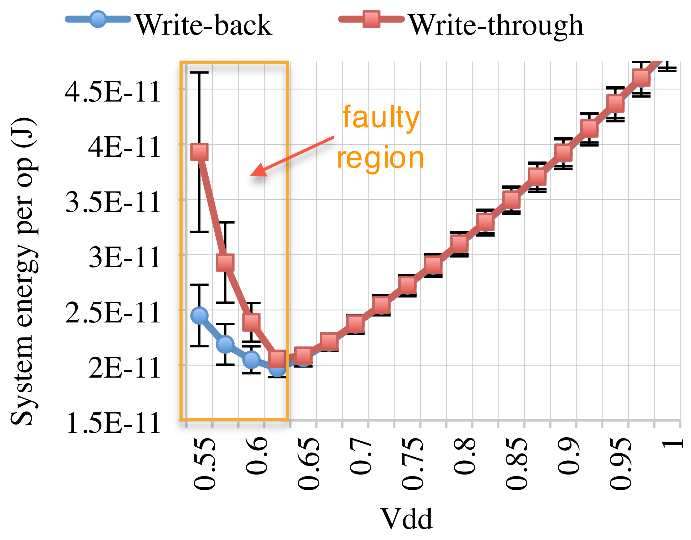


Figure 40: Comparison of average cache energy for write-back and write-through caches at different supply voltages.

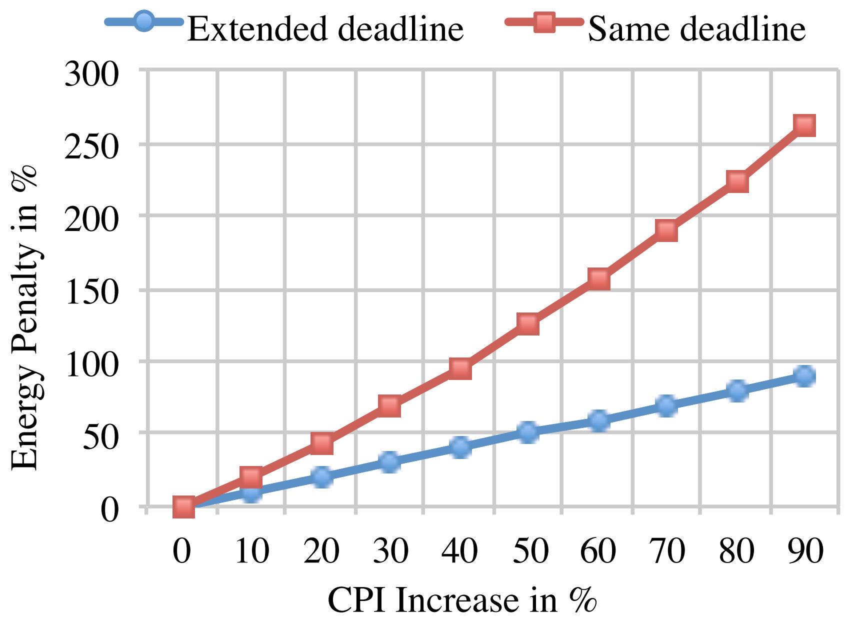


Figure 41: Energy penalty caused by CPI increases.

deadline in terms of seconds. Therefore, an increase in CPI also dictates an increase in supply voltage to ensure that the same deadline is met. By using the FO4 of the technology, we calculate the voltage increase required to continue to meet the deadline, and add this to the calculation. Note that this effect diminishes considerably at lower voltages, because at low voltages, a very small change in voltage greatly changes the delay, so increase in CPI barely effects voltage.

Determining CPI increase for each scheme well enough to make informed design decisions requires cycle-accurate RTL simulation and energy measurement, and is beyond the scope of this paper.

6.10.1 6T vs. 8T

The L1 data cache can be designed with either a single port 6T cell or a dual-port 8T cell. For a write-back cache, it is very advantageous to use an 8T cell to reduce the cost of three-cycle read-modify-write operations. As seen in Figure 41, performance is critical to achieve energy-efficiency, and blocking the data cache to perform read-modify-write operations on a single port could prove expensive.

6.10.2 Summary of aggressive scaling in caches

Due to increasing variability in modern technology generations, SRAM is becoming increasingly susceptible to errors. Error rates are exacerbated by DVFS systems that attempt to aggressively lower the supply voltage to operating at the system's minimal energy per operation. Together with the threat of soft errors, these trends dictate an error-correction requirement for the L1 cache in processors. We analyze two methods enable this error resiliency in the L1 cache: parity + write-through and SEC-DED + write-back. A cache simulator is used to determine the frequency of important parameters such as write-throughs vs. evictions and number of sub-word stores for a variety of SPEC benchmarks. These frequencies are multiplied by energy costs extracted from real processor and SRAM designs in 28nm to accurately determine the energy per operation difference between different schemes over a wide range of supply voltages. For our system, the decision between write-through and write-back is benchmark dependent, and more dependent on performance implications. Energy cost of correction measurements was also measured, but found to be irrelevant as the threat of an uncorrectable error limits supply scaling well before repair costs become significant. Also, Vmin difference between each scheme was only 20mV, so Vmin cannot be used to differentiate between the schemes.

7 SWAAT: SRAM with Adaptive Assist Techniques

With intuition provided by the findings of the last two sections, we propose an adaptive SRAM design that obtains optimal energy efficiency through resilient circuit design. Corners have a large effect on the cause of failures in SRAMs. Therefore, resilient circuit solutions that adapt the SRAM to deal with variability removes the requirement of a voltage guardband. In addition, the SRAM supply voltage will be set by an on-chip DC-DC converter that generates up to a 120mV voltage ripple. Therefore continuous time replicas are needed to adjust assists quickly in response to supply voltage changes.

7.1 Methods to reduce Vmin

Importance sampling shows that negative bitline is the most effective way to improve writeability. Figure 42 shows the negative bitline assist embedded into the write path for a 2:1 interleaved design. The NOR gate is used to ensure the NMOS pass gates do not turn on when the negBoost node is less than 0. The pass gates and pull down NMOS must be large to quickly discharge the bitline. First, the nWrite signals will begin pull charge off of the desired bitline, then both the boosting signal and the wordline are timed to turn on when the bitline has decreased to 0V.

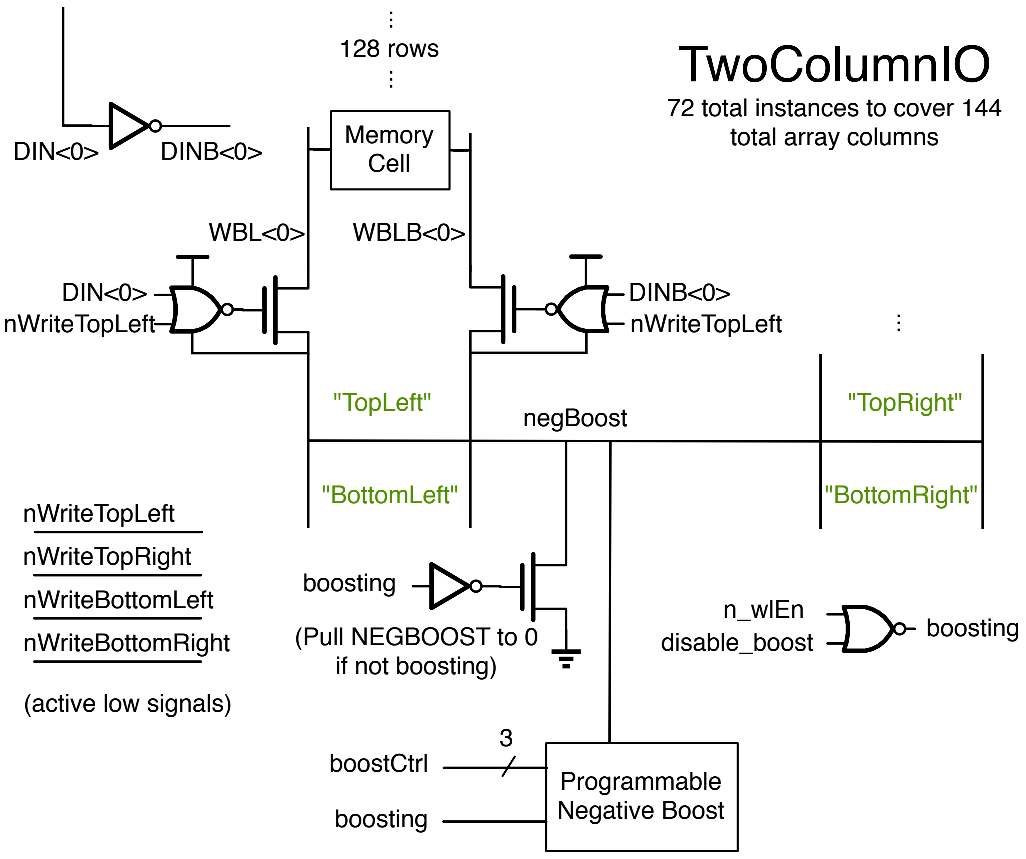


Figure 42: Write path for 8T design including negative bitline boost

The boosting signal controls the negative boost circuit shown in Figure 43. A digital control signal allows capacitance values between 3.5fF and 24.5fF with steps of 3.5fF.

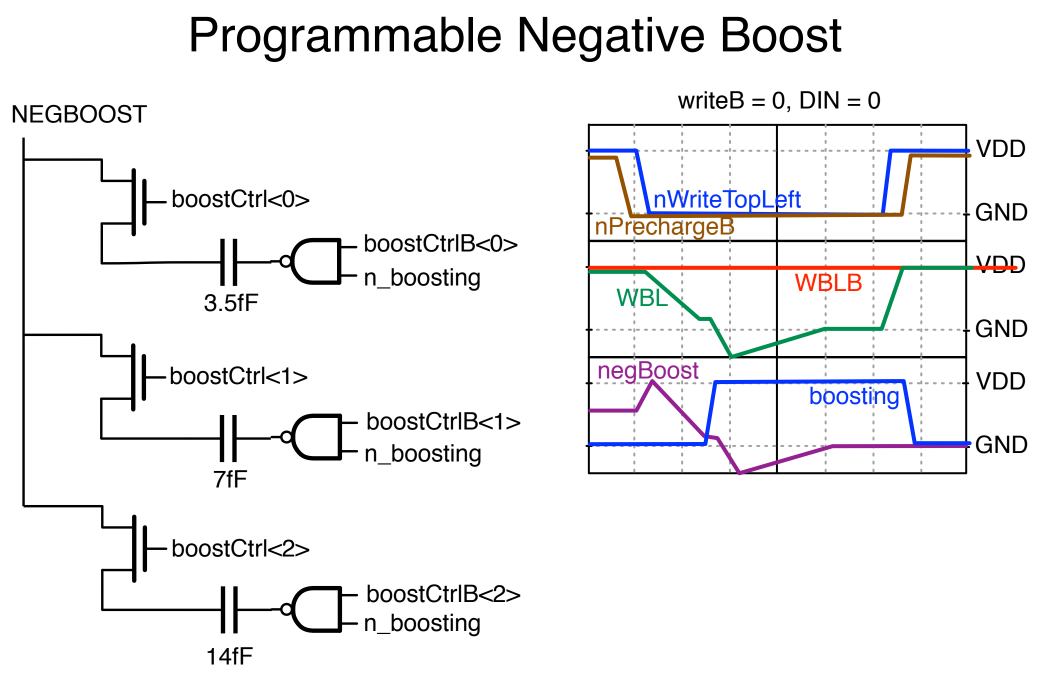


Figure 43: Programmable negative voltage generator

7.2 Methods to reduce energy

As shown in Section 5.6, a group of 12 replica cells with cell Vdd set to Vdd-100mV, combined with a timing multiplier, generates optimal wordline pulse width and sense amplifier timing across corners and voltages. The timing multiplier uses is shown in Figure 44 along with operational waveforms.

7.3 Methods to detect errors

A BIST can intrinsically determine the cause of each error through correct test design. However, ECC only detects errors during read operations, and cannot determine whether the error occurred during the write or read. Therefore, ECC provides no useful information for feedback algorithms. In our implementation, we perform a quick test to attempt to identify the cause of the error. The SRAM performs a ”safe read” operation (multiple cycles of the same read without intervening precharge ensure plenty of bitline discharge). If the error disappears, we know it was a readability error. If the error persists, the cause must have been a stability fault or a writeability fault. Because our system is biased to be stable, we believe that we can correctly identify these faults as writeability faults, and make the appropriate adjustments. Because the pipeline must flush anyway due to incorrectly forwarded data, there are a few cycles of spare time, so this technique does not impact latency.

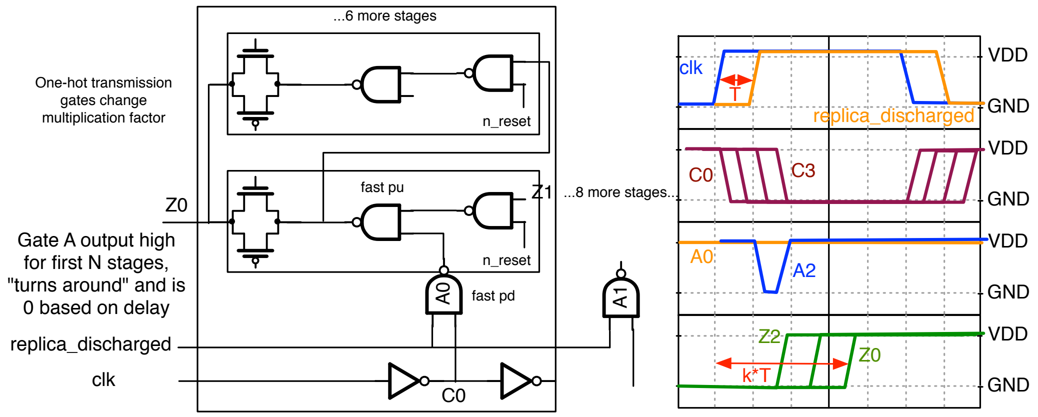

...6 more stages
One-hot transmission gates change multiplication factor
n_reset
fast pu
Z0
Z1
...8 more stages...
Gate A output high for first N stages, "turns around" and is 0 based on delay
A0
fast pd
A1
replica_discharged
clk
C0
clk
T
replica_discharged
VDD
GND
C0
C3
A0
A2
Z2
Z0
k*T

Figure 44: Programmable timing multiplier

7.3.1 Methods to respond to errors

The processor counts errors for a programmable number of cycles. If the error count is above a threshold, the processor traps and responds based on control registers containing a count of the different types of errors. Writes to control registers can adjust programmable SRAM settings.

8 Conclusion

Our main objective was to develop a methodology and framework for quantifying the ”resiliency” of a design. Improving energy-efficiency in SRAM design requires complete understanding of design decision implications on error rate, and importance sampling was used to quantify these results. We used this methodology to approach energy efficiency improvement in four different ways. First, we investigated how organization, assist techniques, and error correction allowed for the reduction of Vmin. Second, we showed how careful wordline timing could minimize switching energy over the entire supply voltage range. Third, we used our understanding of energy consumption and the relationship between voltage and error rate to perform an energy optimization for a L1 and L2 cache system. Fourth, we applied these principles to the design of an SRAM that adapts itself to minimize energy per operation for the entire range of viable supply voltages.

As technology features shrink, and device variation increases, dealing with errors will become a critical component of digital design. Due to the ubiquity of SRAM in modern designs, failures in SRAM will remain a major concern, and the margining required to prevent SRAM failures from causing system failures will prevent energy-efficiency improvements. The technology-independent methodology described here can be used to optimize SRAM designs for energy, area, speed, and resiliency over a wide range of desired operating conditions. The steps described in this methodology can easily be converted into a tool that can be used to improve the results of memory compilers because inputs to our analysis can measured automatically for any technology. For example, technology-independent netlists perform readability, writeability, and read stability analysis over a range of voltages, corners, and cell sizes to determine error rates. Additional netlists measure bitline capacitance, cell leakage, and switching energy for a generic SRAM design. Then, activity factor estimates can be added to account for leakage and switching energy trade-offs. Last, constraints can be imposed to choose optimum designs from the design space.

Further research in this field needs to continue to propagate understanding of failures at a bitcell level to improve system-level efficiency. The techniques discussed here can hopefully provide a basis for investigating efficient ways to improve resiliency against increasing error probabilities in deeply-scaled technology nodes.

References

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

[2] H. Pilo, I. Arsovsić, and Batson, "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," *Solid-State Circuits, IEEE Journal of*, no. 99, p. 1, 2011.

[3] 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," *VLSI Circuits, 2009 Symposium on*, pp. 158–159, 2009.

[4] 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 *ISSCC*, 2011, pp. 260–262.

[5] E. Karl *et al.*, "A 4.6GHz 162Mb SRAM Design in 22nm Tri-Gate CMOS Technology with Integrated Active VMIN-Enhancing Assist Circuitry," in *ISSCC*, Feb. 2012.

[6] 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, no. 11, pp. 1398–1407, Nov. 2010.

[7] M. Manoochehri, A. Ejlali, and S. Miremadi, "Joint write policy and fault-tolerance mechanism selection for caches in DSM technologies: Energy-reliability trade-off," *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, pp. 839–844, 2009.

[8] B. Ahsan, L. Ndreu, I. Sideris, S. Idgunji, and E. Ozer, "Eliminating energy of same-content-cell-columns of on-chip SRAM arrays," in *ISLPED '11: Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*. IEEE Press, Aug. 2011.

[9] E. Seevinck, F. List, and J. Lohstroh, "Static-noise margin analysis of mos sram cells," *Solid-State Circuits, IEEE Journal of*, vol. 22, no. 5, pp. 748 – 754, oct 1987.

[10] S. O. Toh, Z. Guo, T.-J. Liu, and B. Nikolic, "Characterization of Dynamic SRAM Stability in 45 nm CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 11, pp. 2702–2712, 2011.

[11] 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 *DATE '10: Proceedings of the Conference on Design, Automation and Test in Europe*, Mar. 2010.

[12] S. O. Toh, "Nanoscale SRAM Variability and Optimization," Ph.D. dissertation, UCB Thesis Draft, Jan. 2011.

[13] A. Bhavnagarwala *et al.*, "A Sub-600mV, Fluctuation tolerant 65nm CMOS SRAM Array with Dynamic Cell Biasing," *VLSI Circuits, 2007 IEEE Symposium on*, pp. 78–79, 2007.

[14] 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.

[15] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović, "The risc-v instruction set manual, volume i: Base user-level isa," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2011-62, May 2011. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-62.html