Improving Cache Lifetime Reliability at Ultra-low Voltages

Zeshan Chishti, Alaa R. Alameldeen, Chris Wilkerson, Wei Wu and Shih-Lien Lu
Oregon Microarchitecture Research, Intel Labs

ABSTRACT

Voltage scaling is one of the most effective mechanisms to reduce microprocessor power consumption. However, the increased severity of manufacturing-induced parameter variations at lower voltages limits voltage scaling to a minimum voltage, Vccmin, below which a processor cannot operate reliably. Memory cell failures in large memory structures (e.g., caches) typically determine the Vccmin for the whole processor. Memory failures can be persistent (i.e., failures at time zero which cause yield loss) or non-persistent (e.g., soft errors or erratic bit failures). Both types of failures increase as supply voltage decreases and both need to be addressed to achieve reliable operation at low voltages.

In this paper, we propose a novel adaptive technique to improve cache lifetime reliability and enable low voltage operation. This technique, multi-bit segmented ECC (MS-ECC) addresses both persistent and non-persistent failures. Like previous work on mitigating persistent failures, MS-ECC trades off cache capacity for lower voltages. However, unlike previous schemes, MS-ECC does not rely on testing to identify and isolate defective bits, and therefore enables error tolerance for non-persistent failures like erratic bits and soft errors at low voltages. Furthermore, MS-ECC's design can allow the operating system to adaptively change the cache size and ECC capability to adjust to system operating conditions. Compared to current designs with single-bit correction, the most aggressive implementation for MS-ECC enables a 30% reduction in supply voltage, reducing power by 71% and energy per instruction by 42%.

Categories and Subject Descriptors

B.3.4 [**Memory Structures**]: Reliability, Testing, and Fault-Tolerance.

General Terms

Design, Reliability

1. INTRODUCTION

As semiconductor technology continues to scale, energy efficiency is becoming the key design concern for computer systems. Microprocessors often use multiple power modes to exploit the power-performance tradeoff in order to improve energy efficiency. Many processors (e.g., the Intel® Celeron® processor [11]) have high-performance and low-power modes of operation. In the high-performance mode, the processor uses a high voltage and runs at a high frequency to achieve the best performance. In the low-power mode(s), the processor runs at a lower frequency and uses a lower voltage to conserve energy. Such power saving features are becoming prevalent in current processor designs.

Reducing supply voltage is one of the most effective methods to reduce power consumption. However, as supply voltage decreases, manufacturing-induced parameter variations increase in severity, causing many circuits to fail. These variations restrict voltage scaling to a minimum value, often called Vccmin (or Vmin), which is the minimum supply voltage for a die to operate reliably. Failures in memory cells typically determine the Vccmin for a processor as a whole [31]. Reducing Vccmin in the context of memory failures is important for enabling ultra-low power modes that are more energy-efficient.

Prior work [21, 31] has previously proposed techniques to enable ultra-low voltage cache operation in the context of high memory cell failure rates. The proposed techniques trade off cache capacity for reliable low voltage operation. In the high-voltage mode of operation, cell failure rate is low and the entire cache is available for use. During the low-power, low-voltage mode, cache size is sacrificed to increase reliability. These techniques enable a significant reduction in supply voltage. However, they require conducting thorough memory tests at low voltages to isolate defective bits. Memory tests must be performed whenever the processor boots, and the location of defective bits needs to be stored in the memory hierarchy. While these tests can detect voltage-dependent, persistent bit failures, other non-persistent sources of bit failures are dynamic in nature and cannot be detected by testing. Examples of such non-persistent bit failures include erratic bit failures [1] and soft errors. Non-persistent failures also increase when supply voltage decreases as we show in Section 3. Techniques like those proposed in [21, 31] cannot mitigate non-persistent errors and therefore must rely on a voltage guardband to enable reliable cache operation.

Other prior work addresses non-persistent, transient bit failures at normal operating voltages. Examples of such work include error-correcting schemes such as the two-dimensional ECC proposal by Kim et al. [14]. These techniques effectively tolerate multiple bit errors due to non-persistent faults. However, prior work focused only on failure rates at normal operating voltages, and did not address persistent failures at low voltages which result in yield loss.

To enable a cache design that tolerates both non-persistent and persistent failures at low voltages, we need a unified solution that does not rely on testing. We propose using redundancy to enable ultra-low voltage cache operation. Our solution, multi-bit segmented ECC (MS-ECC) employs error correction codes to tolerate both persistent and non-persistent bit failures at low voltages. During low-voltage operation, some ways in each cache set are used to store ECC check bits for the remaining ways, thereby increasing reliability against high failure rates. The number of ways used for storing ECC can be adaptively chosen

Micro'09, December 12–16, 2009, New York, NY, USA.

by the operating system on the basis of the desired reliability level, which in turn depends on the target Vccmin. Increasing the number of ECC ways increases the redundancy and thus the reliability but decreases the cache capacity available during low-voltage operation.

To simplify ECC implementation, MS-ECC divides a cache line into multiple small segments and corrects errors on a per-segment basis. Performing error correction at finer granularities enables more errors to be fixed with lower latency and complexity. To further reduce the logic complexity of error correction, we leverage the previously proposed Orthogonal Latin Square Codes (OLSC) [9]. OLSC enables faster encoding and decoding than traditional ECC, at the cost of more check bits. Furthermore, OLSC uses modular error correction hardware which can be used, adaptively, to correct a varying numbers of errors. This adaptive design can be used to trade off reliability for performance in the low-voltage operating mode. If performance is insensitive to cache size in the low-voltage operating mode as shown in [31], then the design should target maximum reliability. If some application-specific performance is sensitive to cache size, then the error correction capability can be sacrificed to increase cache size.

This paper makes the following main contributions:

1. To our knowledge, our paper is the first to quantify the impact of *both* persistent and non-persistent (erratic, soft errors) failures on cache yield loss *and* lifetime reliability.
2. We propose a novel error tolerance mechanism, multi-bit segmented ECC (MS-ECC), which uses Orthogonal Latin Square Codes to reduce Vccmin by supporting multi-bit error correction for small cache line segments and cache tags.
3. We propose an adaptive mechanism that enables a variable part of the cache to be used for error correction. This mechanism can correct 1-4 errors for each 64-bit segment, where a higher correction capability increases reliability at the expense of sacrificing a bigger percentage of the cache size.
4. We show that the most aggressive implementation of MS-ECC can achieve reliable cache operation at 520 mV, incurring minimal additional latency, while sacrificing half of the cache capacity at low voltages. Compared to previous schemes, our proposal addresses both persistent and non-persistent failures, and reduces the overhead of thorough testing at boot time.

In the remainder of this paper, we discuss the impact of bit failures on Vccmin in Section 2. We describe two types of non-persistent failures and their impact on lifetime reliability in Section 3. We discuss our proposed technique in detail in Section 4. We introduce the experimental methodology in Section 5 and evaluate our technique in Section 6. We conclude in Section 7.

2. BACKGROUND

2.1 Bit Failures and Vccmin

Large SRAM caches make up a significant percentage of transistors in a microprocessor die. Parameter variations, induced by imperfections in the semiconductor process, limit the minimum operational supply voltage to Vccmin, below which an SRAM cell fails to operate reliably. For each of the SRAM caches, the bit with the highest Vccmin determines the Vccmin of the cache as a whole [31].

Bit failures can be classified into two broad categories: *persistent* and *non-persistent.* The first category contains the majority of bit failures, where bits exhibit persistent failing behavior. Several papers [2, 15, 31] have analyzed persistent failures in detail and have shown that intra-die random dopant fluctuations (or RDF) play a primary role in these failures. Prior work has also demonstrated that persistent failures exhibit a strong dependence on supply voltage. For example, Kulkarni et al. [15] show that the bit failure rate increases exponentially with a decrease in voltage. Since these types of failures can be reliably identified using standard memory testing methodologies, we refer to them as *testable* failures. Memory tests are performed on each die before it is shipped to a customer, and dies with irreparable failures identified by the memory tests are disposed of. As a result, persistent failures typically contribute to yield loss but do not play a direct role in determining the lifetime reliability of a microprocessor. The lifetime reliability of a processor is usually represented by FIT (Failures In Time), the number of failures that occur in a billion hours for a particular unit.

The second category of failures consists of non-persistent bit failures, where bits exhibit sporadic failing behavior. Failures resulting from particle strikes (soft errors) as well as erratic bit failures, both discussed in greater detail in Section 3, are examples of this category. Since these failures are non-persistent and occur randomly, they can’t be identified with memory tests. As a result, these failures don’t directly contribute to yield loss, instead contributing directly to a unit’s FIT rate. We classify failures of this type as *non-testable* failures.

2.2 Related Work

One solution to improve cache reliability is to implement true column and/or row redundancy by adding multiple spare rows and/or columns to the cache array [24]. This solution is able to tolerate errors that cause a few rows or columns to become defective. However, it cannot deal with thousands of randomly distributed cache bits in large caches which would become defective in the low voltage mode due to high cell failure rates.

Kim et al. [14] propose a scheme to use two-dimensional ECC to correct multi-bit errors. This scheme is tailored to deal with clustered multi-bit failures that cause contiguous bits across multiple rows and columns to fail concurrently. The ability of this scheme to correct errors is strongly dependent upon the location of defective bits. While this scheme is able to tolerate correlated failures that affect several contiguous bits, it cannot tolerate failures in multiple randomly distributed bits in each cache line that become defective at low voltages due to random dopant fluctuations.

Another solution to improve cache reliability at low voltages is to change the SRAM cell design. The designer can upsize the transistors or use cell design variants such as the 8T, 10T, and ST SRAM cells [15]. However, the resulting Vccmin reduction comes at the cost of significant increases in area (e.g., 100% area increase for the ST cell). Furthermore, larger SRAM cells result in increased leakage power in both high-performance and low-power modes.

A recent paper [31] proposed two architectural schemes to enable cache operation at ultra-low voltages. The first scheme, word-disable, disables 32-bit words that contain one or more defective bits. Physical lines in two consecutive ways combine to form one logical line, where only non-failing words are used. A similar scheme was proposed by [21]. The second scheme, bit-fix, uses a quarter of the cache ways to store the location of defective bits in other cache ways, as well as the correct value of these bits. That work focused on testable, voltage dependent, persistent failures, and evaluated Vccmin in the context of yield loss.

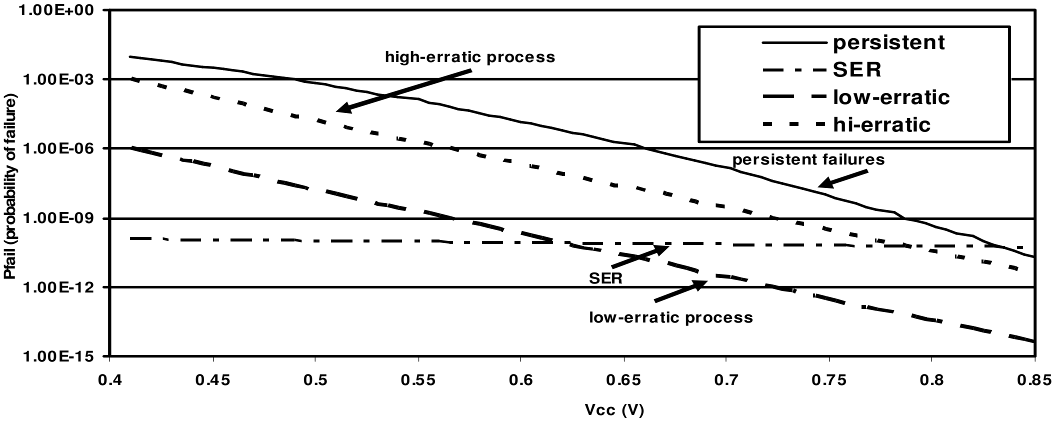


Figure 1. Probability of Persistent Failures, Soft Errors and Erratic Failures per Hour vs. V_{cc}.

Vccmin was defined as the voltage at which the cache is functional in 999 of every 1000 dies. However, the paper did not evaluate the impact of FIT rate on Vccmin, although the authors acknowledged that additional ECC and supply voltage guardbands might be required. In comparison, our work addresses both persistent failures and non-persistent failures, and extends the previously proposed failure probability model to account for the impact of FIT rates on Vccmin.

Because several previous papers [2, 15, 31] have discussed the causes of persistent cell failures in detail and attributed them to random intra-die dopant fluctuations (RDF), we omit the discussion of persistent failures and instead focus on non-persistent bit failures in the next section.

3. NON-PERSISTENT BIT FAILURES

3.1 Soft Errors

Soft errors have increased in significance in recent years due to the increasing number of devices per die while the soft error rate per device remained constant or slightly decreased across technology generations [8, 16, 25]. This increase, as well as future expected increases in soft error rates (SER), triggered research to mitigate the impact of soft errors on the correctness of a program execution. Computer architecture research has recently focused on providing architecture-level solutions to mitigate soft errors [2, 19, 26, 30, 32].

Our target of running a processor in a low-power mode at low voltages further exacerbates the soft error problem. Previous measurement studies have shown that reducing supply voltage increases the soft error rate exponentially [12, 13, 22, 29]. These measurements for SRAM cells and flip-flop designs from multiple vendors all confirm that soft error rates will increase at lower voltages. This increase is caused primarily by the exponential relationship between the soft error rate and the charge stored at a particular node, which in turn changes linearly with supply voltage [29].

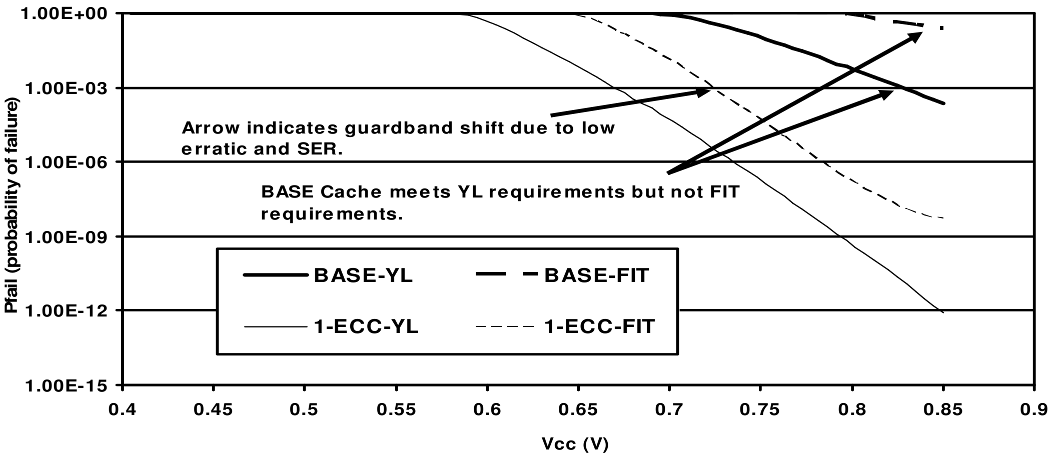
In our studies, we used the data measured by Ünlü, et al. [29] to estimate the soft error rate per SRAM bit. We extrapolated this data to lower voltages. However, since this data was measured by inducing neutrons from a nuclear reactor, we scaled the soft error rates by a factor of one billion to estimate the soft error rate under normal conditions at sea-level [33].

While previous measurements show that soft error rates increase exponentially with reduction in supply voltage, the rate of increase is limited to 2.5x-3x for every 500mV decrease in supply voltage. This soft error rate increase is much lower than the increase in persistent failures, where a similar decrease in supply voltage leads to an increase in failure probability of more than a billion times [15, 31]. Figure 1 compares the probability of different types of cell failures as a function of supply voltage. The persistent failure probabilities in Figure 1 are based on results reported in [15] which were obtained with circuit simulations validated against measured data. Compared to voltage-dependent, persistent failures, Figure 1 shows that soft errors are more significant at higher voltages. At low voltages, however, persistent and erratic failures significantly overshadow soft errors.

3.2 Erratic Bit Failures

Erratic bit failures have played a key role in setting Vccmin in the past, and they are likely to re-emerge as a reliability concern in the future [1, 6]. Erratic behavior in the Vccmin of an SRAM cell can occur when an NMOS pull down device in an SRAM cell experiences soft breakdown. In soft breakdown, random telegraph signal noise in the gate oxide leakage of the NMOS device can cause the SRAM cell Vccmin to erratically fluctuate by as much as 200mV [1]. Due to their random nature, erratic cells may escape standard testing and appear as normal cells, but may cause bit failures later. Erratic behavior in SRAM cells depends strongly on process parameters. Agostinelli et al. [1] report discovering erraticism on the 90nm process technology node, but were able to mitigate it successfully. However, the authors point out that continued device scaling is likely to re-introduce erraticism in future process technologies.

Detailed information on erratic bit failures is scarce and good physical models are non-existent. Despite the lack of good physical models, it is important to consider erratic bit failures especially when considering operation at low voltages. In our studies, we use a very simple model for erratic bit failures and address the sensitivity of erratic bit failures to process parameters by modeling two hypothetical processes with both high and low rates of erraticism. Since cell stability and oxide strength play a role in both erratic bit failures and persistent failures, we expect that the probability of an erratic bit failure will be proportional to the probability of a voltage–dependent, persistent failure. Our studies reflect this by setting the probability of an erratic bit failure as a fixed percentage of persistent failures. Furthermore,



Arrow indicates guardband shift due to low erratic and SER.

BASE Cache meets YL requirements but not FIT requirements.

Figure 2. FIT Rate and Yield Loss (YL) for a Baseline 2 MB Cache, and a Cache with SECDED ECC.

we expect that the frequency and severity of erratic bit failures will be highly process-dependent. To reflect process sensitivity, we model both a high-erratic process with a high rate of erratic bit failures, and a low-erratic process with a low rate of erratic bit failures. Our high-erratic process produces one erratic bit failure for every ten persistent failures. The probability of an erratic failure on our low-erratic process is 1000 times lower, or one erratic failure for every 10,000 persistent failures. We model the frequency and duration of erratic bit failures in the same way that we model soft error failures; the probability of an erratic bit failure reflects the probability of an erratic failure in an hour; and we assume that the erratic failure (or soft error) lasts an hour.

Figure 1 shows the probability of different types of failures as a function of voltage. We note that since we model erratic failures as a fixed proportion of persistent failures, the probability of erratic failures is sensitive to supply voltage. It is also worth noting that since Figure 1 shows different failure rates on the same Y-axis, the increase in SER (2.5x – 3x for 500 mV decrease in supply voltage) appears flat relative to the much larger increase in persistent errors (higher than one billion times for 500 mV decrease in supply voltage).

3.3 Comprehensive Yield Loss/FIT Model for Cache Failures

To account for the impact of FIT rate on Vccmin, we use a comprehensive model for cache failures that includes the impact of voltage–dependent, persistent failures on yield loss as well as the impact of soft error rates (SER) and erratic bits on FIT rate. Our model for yield loss is similar to the model evaluated in [31], but we extend the model with a time component to model failures in time. To calculate the FIT rate of our cache, we divide the cache lifetime into discrete 1-hour periods and compute the probability of a bit failing during each 1-hour period.

We conservatively assume that SER failures in the cache will last an hour before the bit is either rewritten or scrubbed. We assume the same duration for erratic bit failures. Figure 1 shows the probability of SER and erratic failures per hour, and the probability that a bit suffers from a voltage dependent persistent failure. Using these three probabilities, we can determine the overall probability that a bit will fail for one of these three reasons in a 1-hour period. For longer periods (e.g., 250 hours), we combine the probabilities of failure for each hour to get the overall probability of failure. The probability of failure for the entire cache is (1 – probability of success), where the probability of success is the probability that each bit in the cache stays failure-free for every hour in the period. Figure 2 shows the pfail of a base 2MB cache as derived from our model. There are two pfail curves for each cache type: without ECC (BASE-YL, BASE-FIT) and with ECC (1-ECC_YL, 1-ECC-FIT). The line marked BASE-YL refers to pfail of the cache due solely to persistent failures. A pfail of 10^{-3} corresponds to a yield loss of 0.1%. A separate line, marked BASE-FIT, indicates the probability that the cache will fail during a 250 hour period. A pfail of 10^{-3} on that line corresponds to one failure every 250,000 hours, or a FIT rate of 4000 failures in a 10^9 hour period. In the remainder of our paper, we target a yield loss of 0.1% as suggested in [31] and a FIT rate of 4000 as suggested in [26]. To meet these requirements both FIT and YL pfails must be less than or equal to 10^{-3}.

Data in Figure 2 shows that although the simple 2MB cache (BASE) meets yield loss requirements at 830mV, FIT requirements are not met. Adding 1-bit ECC to the cache enables it to meet yield loss requirements at 670mV and FIT requirements at 725mV. Since both FIT requirements and yield loss requirements must be met, Vccmin for this cache would be set at 725mV. In the next section, we propose a technique to reduce Vccmin further.

4. MULTI-BIT ERROR CORRECTION FOR VCCMIN REDUCTION

Redundancy is the most widely used approach to increase reliability. There are several options to implement redundancy. Employing error detection and correction codes is one of the methods that improve the reliability of a system through information redundancy. Single Error Correction, Double Error Detection (SECDED) codes such as the Hamming code [17] are well known and have been used in memory chips and on-chip caches due to their simplicity. With continued transistor scaling, multi-bit error correcting codes are becoming more important for large on-chip memory arrays. For example, a Double Error Correction Triple Error Detection (DECTED) code can be designed based on a binary BCH code [4]. While SECDED and DECTED codes provide sufficient redundancy to deal with bit failures at normal operating voltages, neither technique provides enough redundancy to allow cache operation at an ultra-low

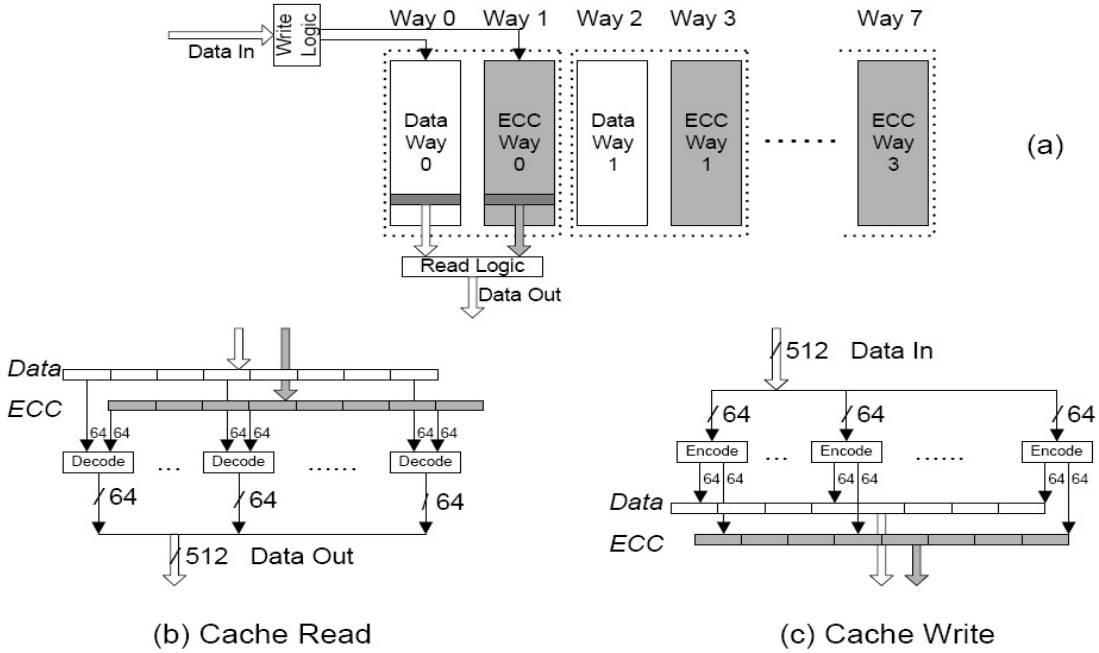


Figure 3. Example of Multi-Bit Segmented ECC with Eight 64-bit Segments.

voltage level (e.g., 500 mV). Wilkerson et al. [31] showed that enabling 500mV operation in a 2MB cache requires 10-bit error correction for each 512-bit cache line. Extending conventional ECC to correct 10-bit errors over an entire 512-bit cache line requires significant area, latency, and power overheads [5, 14], which we briefly describe later in this section. We next propose our multi-bit segmented ECC that achieves the same purpose with lower overhead.

4.1 Multi-bit Segmented ECC

To enable correcting a large number of bits with manageable complexity and latency overhead, we propose multi-bit segmented ECC (MS-ECC). In this technique, we trade off cache capacity for reliability at low voltage. The main idea behind MS-ECC is to correct multi-bit errors by implementing error correction at finer granularity segments within a cache line. In the high-voltage, high- performance operating mode, the entire cache capacity is available for use by the processor, and conventional ECC is used. In the low voltage, low power operating mode, a portion of the cache is used to store additional ECC information on granularities finer than a cache line, thereby enabling more errors to be fixed on a per-line basis. Because the size of each segment is smaller than the entire cache line, the latency and complexity for MS-ECC is significantly less than conventional (un-segmented) ECC.

Example: Consider a 2MB 8-way L2 cache with 64-byte lines. In the low voltage mode, we divide the eight physical ways in each set amongst (i) data ways and (ii) ECC ways. The ratio of data ways to ECC ways depends on the desired reliability level, which in turn depends on the target Vccmin. If the operating system decides to operate at a higher reliability level, this ratio would be adaptively increased to increase the redundancy and decrease the cache capacity available during low voltage operation. We analyze the impact of this ratio on Vccmin and performance in Section 6.2. For this example, assume that there is one ECC way for every data way (50% cache capacity available in low voltage mode). We use a fixed mapping to associate data ways with their corresponding ECC ways (Figure 3(a)): physical way 1 stores the ECC for physical way 0, physical way 3 stores the ECC for physical way 2, and so on. Thus, in the low-voltage mode, the cache effectively becomes a 1MB 4-way set-associative cache.

We divide each data way into multiple segments and store the ECC for each segment in the corresponding ECC way. Figure 3 shows an example of multi-bit segmented ECC with eight 64-bit segments in each 512-bit line. On a read hit (Figure 3(b)), we fetch both the data line and the corresponding ECC line. There are separate ECC decoders for each of the eight segments that decode segments in parallel by using information from both the data and ECC ways. The decoded segments are then concatenated to obtain the entire 512-bit line. On a write hit to the L2 cache (Figure 3(c)), we first use the ECC encoders to obtain the ECC for the data line. Like the ECC decoders, there are separate encoders for each segment that perform ECC encoding in parallel. We then write the new data to the data line and the new ECC to the corresponding ECC line. A similar encoding is performed when a new line is brought into the L2 cache upon a cache miss. We note that each cache access in the low-voltage mode requires access to both the data way and the ECC way. To avoid increasing cache latency, we assume double-width buses for concurrent transfer of the data and ECC ways to the ECC encoder/decoder. Alternatively, one could also read the two cache lines sequentially, thereby incurring additional latency overhead in the low-voltage mode.

4.2 Orthogonal Latin Square Codes (OLSC)

Performing error correction at finer granularities helps in decreasing the complexity of multi-bit error correction. However, our evaluation shows that enabling ultra-low voltage operation requires three or more errors to be fixed per segment, even for small segment sizes. Conventional error correction codes based on BCH codes usually fix one (SECDED) or two errors (DECTED). These codes have been optimized for storage overhead (i.e.,

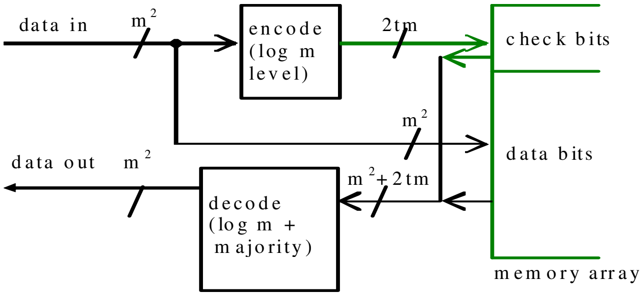


Figure 4. OLSC-based Error Correcting Technique.

number of check bits) at the cost of logic complexity. The complexity and latency of these codes grow rapidly with the increase in the number of error corrections [14]. To enable ultra-low voltage operation, MS-ECC needs to use an error correction code whose complexity scales well with the number of error corrections. Hsiao et al. [9] proposed a coding methodology called Orthogonal Latin Square Codes (OLSC) to correct multi-bit errors. While OLSC requires more check bits than traditional ECC, it has modular correction hardware, lower logic complexity and can be encoded and decoded faster than traditional ECC [9]. Compared to traditional ECC techniques, OLSC-based MS-ECC has a lower latency and offers the operating system flexibility in sacrificing a varying percentage of cache capacity based on the desired Vccmin and reliability levels in the low-power mode.

OLSC uses the general principle of a redundant system based on majority voting. A simple example of a majority voting redundancy-based design is triple modular redundancy (TMR). TMR "votes out" any incorrect behavior by having three copies of the design and adopting the results from two out of the three modules as the correct one. Instead of duplicating each data bit three times, OLSC encodes "orthogonal" groups of bits to form check bits. At decoding time, each data bit generates the final value through a voting process from a group of orthogonally coded data and check bits. Thus, OLSC does not need to generate a syndrome but can "correct" errors directly from majority voting.

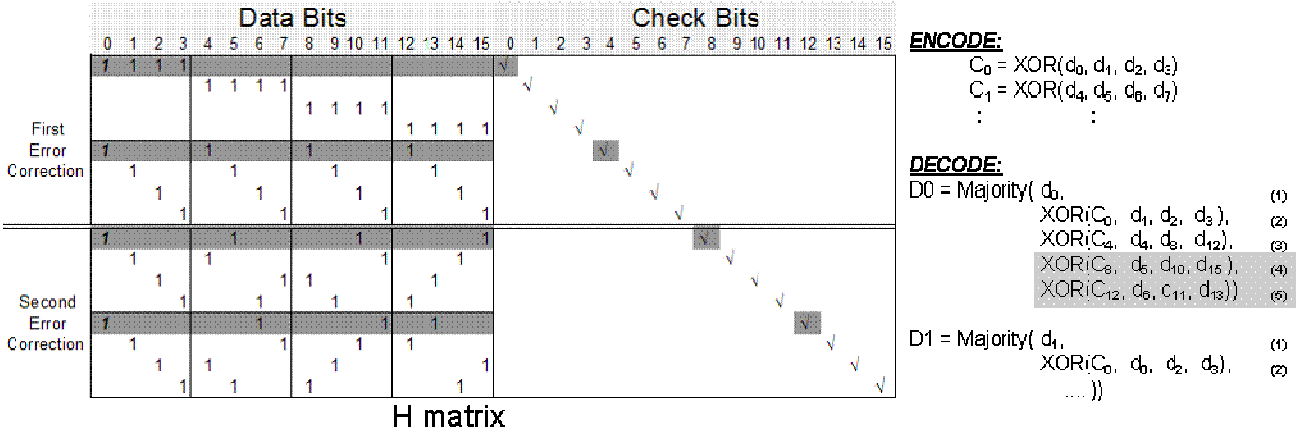
Figure 4 shows a high-level block diagram of the error correcting technique based on OLSC. As indicated in the figure, there are N=m^2 data bits (d_0, d_1,…, d_{N-1}). To perform t error corrections, OLSC requires 2tm check bits. The OLSC circuitry contains two parts, the check bits generation (*Encoder*) and error correction (*Decoder*). The orthogonal property of OLSC allows the encoder and decoder to be constructed modularly. Each added module provides extra correction ability without disturbing existing modules.

Encoder: The Encoder computes check bits from the input data. The encoder circuits are derived from the binary form of the H-matrix. Each check bit C_i is computed as the XOR over data bits corresponding to columns of H-matrix that have a '1'. As an example, Figure 5 shows the H-matrix and the logic equations for generating check bits for 16-bit data with 2 error corrections. Each row in the H-matrix has exactly m bits of '1's. Thus the calculation for each check bit requires an m-input XOR operation. If we only correct one error, then we only need the first 2m=8 check bits and we can disable the rest of the encoder circuit. The logic complexity is proportional to the square root (m) of the number of check bits (m^2), with the critical path being ceil($\log_2(m)$) levels of 2-input XOR gates.

Decoder: The decoder for each data bit involves one (2t+1)-input majority voter. One input of the voter is the received d_i' itself, the other 2t are derived from check bits that contain d_i as its encoding variable, each is an m-input XOR tree. Figure 5 shows the decoding logic needed for data bit d_0 in 16-bit data with t=2 correction. If we are correcting for only one error (t=1), then the last two inputs to each majority function are set to 0 and 1, respectively, so that they offset each other and not affect the outcome of the majority of other bits. This can be achieved with an additional level of multiplexing, selected by whether we correct two bits, where each input can be either the XOR in the figure or a fixed 0 or 1 value. The critical path for the decoder is ceil($\log_2(m)$) levels of 2-input XOR, one level of 2:1 MUX, plus (2t+1)-input majority function. This modular design allows us the flexibility in choosing different levels of error correction.

4.3 Logic/Delay Analysis and Optimizations

In this subsection, we discuss the logic and delay analysis for MS-ECC using OLSC. To keep the analysis simple, we assume that the lowest ratio of data bits to ECC bits is 1:1, similar to the example in Section 4.1. We use the following terminology in our analysis. S represents the segment size in bits. The maximum segment size is 512 bits for a whole 64-byte cache line. t denotes the maximum number of correctible bits per segment. To build OLSC for each single segment, we need mutually orthogonal Latin Squares with side length m, where m is equal to ceil($\sqrt{S}$). For a t-error correction code, we need 2*t*m check bits. Since our



	Data Bits																Check Bits															
	0	1	2	3	4	5	6	7	8	9	10	11	12	'3	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
First Error Correction	1	1	1	1													√															
					1	1	1	1										√														
									1	1	1	1							√													
													1	1	1	1				√												
	1				1				1				1								√											
		1				1				1				1								√										
			1				1				1				1								√									
				1				1				1				1								√								
Second Error Correction	1					1					1					1									√							
		1			1							1			1											√						
			1					1	1					1													√					
				1			1			1			1															√				
	1						1					1		1															√			
		1						1			1		1																	√		
			1		1					1						1															√	
				1		1			1						1																	√

H matrix

ENCODE:

C_0 = XOR(d_0, d_1, d_2, d_3)
C_1 = XOR(d_4, d_5, d_6, d_7)
: :

DECODE:

D0 = Majority(d_0, (1)
XOR(C_0, d_1, d_2, d_3), (2)
XOR(C_4, d_4, d_8, d_{12}), (3)
XOR(C_8, d_5, d_{10}, d_{15}), (4)
XOR(C_{12}, d_6, c_{11}, d_{13})) (5)

D1 = Majority(d_1, (1)
XOR(C_0, d_0, d_2, d_3), (2)
….))

Figure 5. An Example for a 16-bit OLSC.

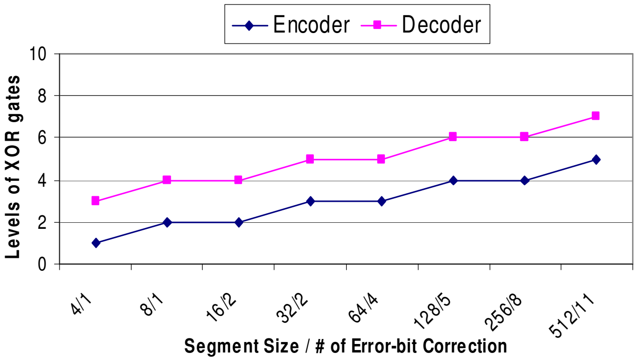


Figure 6. Critical Path Delays.

most aggressive proposal has the same number of data and ECC bits ($2*t*m = S$), the storage overhead for OLSC is 50% in the worst case.

We showed in Section 4.2 that the encoding latency is the delay due to ceil($\log_2(m)$) levels of 2-input XOR gates, while decoding has the same latency plus ($2t$+1)-input majority function and a 2:1 MUX. Conventional majority voter usually involves a counter or a full adder to calculate the sum of the inputs. The number of logic gates and delay for this voter increases with an increase in number of inputs. Alternatively, one could use majority voter implementation based on a voltage sense amplifier (SA) [10, 20]. A voltage SA evaluates the voltage difference between a pair of bitlines which are discharged by two groups of complemented logic. The dominant logic value discharges one of the bitlines more than the other and leads to a voltage difference. Schinkel et al. [23] propose a 90nm technology voltage SA for memory bitlines with a setup plus hold time of 18 ps, well within one XOR gate delay [28]. The majority voter used by MS-ECC has no more than 23 inputs (for 11-bit error correction assuming the largest segment size of 512 bits), far fewer than the number of gate inputs associated with regular memory bitlines. Therefore, to account for the 2:1 MUX and the majority voter, we assume the MS-ECC decoder delay to be approximately two levels of XOR gates higher than the encoder delay.

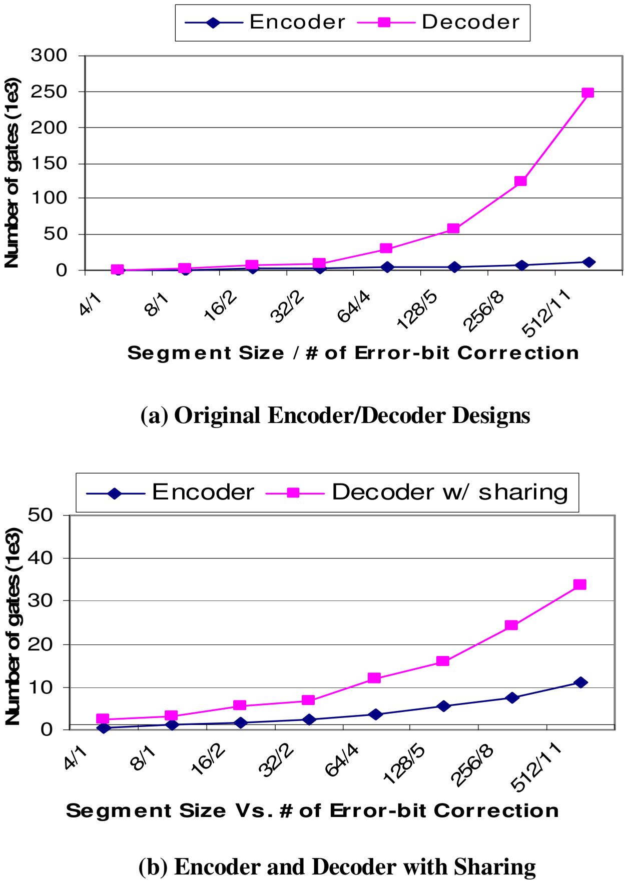
In Figure 6, we show the critical path delay for different segment sizes and number of correctible bits per segment. It has to be noted that the number of bit corrections is different for different segment sizes and is determined by keeping the check bits overhead fixed at 50% similar to our worst case. The delay is measured in terms of XOR gate levels. For the encoder, the delay approximates to $\log_2(\sqrt{S})$. For a 4-bit segment, the delay is one level. For a 512-bit segment, which is 128 times larger, the delay only increases to five levels. We note that the numbers here only show the gate delay; and do not include any interconnect or wiring delay, which can be significant for large segments.

The OLSC scheme has a much smaller delay compared to other error correcting codes (e.g. Hamming or BCH [18]), especially when the segment size is large. Conventional ECC schemes usually depend on a global parity bit for error detection, which requires $\log_2(512)$=9 levels of XOR gates and extra wiring overhead, in addition to the complexity of the error correction phase.

Figure 7 shows the number of gates for MS-ECC under different configurations. For each configuration, we show the total number of gates in the MS-ECC logic for the entire cache, and not the number of gates per segment. "Encoder" has a gate count proportional to $\sqrt{S}$, which is m. However, the "Decoder" size grows much faster as segment size increases. The size is directly proportional to S, since there is a $2*t$ m-input XOR tree for the majority voter of each single data bit. "Decoder" for a 4-bit segment requires 1K gates. For a 512-bit segment, the number increases to 247K. To decrease the decoder logic overhead, we investigated a simplified implementation called "decoder with sharing", which trades off logic complexity for a small increase in latency. This implementation shares the decoder XOR tree logic between majority voter inputs for different data bits. For example in Figure 5, the XOR operation for second majority input in the decoder equations for D0 and D1 differ in only one bit. By sharing the XOR trees across multiple majority inputs, we decrease the logic overhead considerably. Figure 7(b) shows the gate overhead for the decoder with sharing design. The number of XOR gates for this design is proportional to t, which is close to $\sqrt{S}$, as opposed to being proportional to S for the original decoder. This decrease in logic overhead comes at the expense of adding one XOR gate delay to the decoding latency.

4.4 Tag Error Correction

Failures in cache tags can have serious consequences on the correctness of program execution. A failure in a tag bit can cause a false positive, where a cache miss erroneously becomes a cache hit to a different line. This can cause reading or writing to the wrong address, thereby corrupting program data. A tag bit failure can also cause a false negative, where a cache hit to a line in the



(a) Original Encoder/Decoder Designs

(b) Encoder and Decoder with Sharing

Figure 7. Gate Counts.

cache erroneously becomes a cache miss. This can be harmless in a write-through cache, but can cause loss of updated data in a writeback cache (if the line was modified in the cache). To avoid these correctness problems, error-correcting code (ECC) can be used to protect tag bits.

A simple implementation for tag error correction adds standard n-bit ECC to the tag. On a cache access, both the tag and the ECC bits are read from the tag array. The ECC bits are used to correct any erroneous address tag bits. The correct tag bits are then used to determine whether the incoming address hits or misses in the cache. However, this simple implementation increases the length of the critical path of a cache hit/miss determination. This can cause a significant slowdown, especially in the high-performance mode.

Our proposal to address this issue is to address these errors differently in the high-performance and the low-power modes. In the high-performance mode, a single parity bit is used in addition to a standard SECDED ECC mechanism. The parity bit is used to detect errors, which adds a relatively small additional latency to the critical path. However, when the parity test fails on an address tag, SECDED is used to correct the failure, which can be done using an interrupt service routine. In the case of a double error when the error is detectable but cannot be corrected, the cache line is invalidated if it is clean, and a failure signal is issued if the line was modified.

In the low-voltage mode, failures are more frequent, and cache access time is not as critical. Furthermore, some of the data lines are used as ECC lines, so their tags are not used. We propose using the unused tags, corresponding to ECC lines, to store the Orthogonal Latin Square Codes (OLSC) of the used tags. Each tag that corresponds to a data line contains actual tag bits, and each tag that corresponds to ECC lines contains OLSC for one or more associated actual tags. While OLSC requires more check bits, it can be encoded and decoded faster than traditional ECC. This technique is only used in the low-power mode, so it does not add to the critical path in the high-performance mode. In addition, this technique does not require significant additional die area.

4.5 Implementation Issues and Overheads

Cache lines used to store ECC in low-power mode behave similar to data lines in the high-performance mode. In a write-back cache, these lines may contain dirty data at the time of switching from high-performance to low-power mode. The mode switching mechanism needs to write back all the dirty lines from the ECC ways to memory in a similar fashion to how current caches disable cache ways to save power. The number of lines written back depends on the size of cache, the number of ECC ways, and the percentage of dirty lines. For the example in Figure 3, we need to write back half of the cache in the worst case. While we can decrease the number of written-back lines by either controlling the placement of dirty lines or choosing ECC ways dynamically, we leave such optimizations to future work. In addition to writing back dirty lines, switching to low-power mode requires ECC encoding of all the data ways and storing the codes in the ECC ways. This encoding can be overlapped with writing back dirty lines.

During the switch from high-voltage to low-voltage operating modes, the operating system can adaptively choose the number of correctible bits per segment which affects the voltage and reliability levels. This choice can only be done during the power mode switch, and afterwards the ECC ways and the data ways will be fixed (until the next power mode switch). The operating system can decide on the voltage and reliability levels performance requirements in the low-voltage mode. This adaptability feature needs OS support, but we do not focus on evaluating OS trade-offs in this paper.

Table 1. Baseline Processor Parameters.

Voltage Dependent Parameters		
	High Vol.	Low Vol.
Processor frequency	3 GHz	500 MHz
Memory latency	300 cycles	50 cycles
Voltage	1.3v	0.5v
Voltage Independent Parameters		
ROB size	256	
Register file	256 fp, 256 int	
Fetch/schedule/retire width	6/5/5	
Scheduling window size	32 FP, 32 Int, 32 Mem	
Load buffer size	32	
Store buffer size	32	
Cache line size	64 bytes	
L1 Instruction cache	32k bytes, 8-way, 3 cycles	
L1 Data cache	32k bytes, 8-way, 3 cycles	
L2 Unified cache	2M bytes, 8-way, 20 cycles	

The encoding and decoding logic in MS-ECC increases the cache area. The area overhead depends on the segment size and the number of error corrections. In our evaluation, we chose 64-bit segments with a maximum of 4 corrections per segment, so the total number of gates for MS-ECC logic is less than 15K (Figure 7(b)). This overhead is insignificant in comparison with the total cache size. For example, the data arrays alone for 32 KB and 2 MB caches have 256K and 16M 6-transistor cells respectively, not counting the additional transistors for tag arrays and decoder circuitry.

The additional encoding/decoding and the fact that MS-ECC drives two ways of the cache for each data access increase the cache dynamic power. However, we use MS-ECC only in the low-voltage mode (~500mV) where conventional caches cannot operate reliably. Furthermore, cache dynamic power (in particular for L2 cache) is a small component of the total system power. Because of MS-ECC's large supply voltage reduction (from 750 mV to 500 mV), the resulting decrease in both dynamic and static power would far outweigh the small increase in dynamic power due to reading two cache ways for MS-ECC.

5. SIMULATION METHODOLOGY

We use a cycle-accurate, execution-driven simulator running IA32 binaries. The simulator is micro-operation (uOp) based, executes both user and kernel instructions, and models a detailed memory subsystem. Table 1 shows the parameters of our baseline out-of-order processor, which is similar to one core of the Intel® Core™ 2 Duo processor on 65nm technology [7]. For the baseline processor, both the L1 and L2 caches use SECDED ECC on a per-line granularity to tolerate persistent as well as non-persistent bit failures.

To quantify the performance overhead of our technique due to cache capacity and latency changes, we also simulate an ideal, defect-free, low-voltage baseline that has reliable caches without any latency overhead or capacity loss. Because transistor switching delay decreases with an increase in supply voltage, we perform circuit simulations to predict the frequencies at different voltages.

In our experiments, we simulate nine categories of benchmarks. For each individual benchmark, we carefully select multiple sample traces that well represent the benchmark behavior. Table 2 lists the number of traces and example benchmarks included in each category. We use instructions per cycle (IPC) as the performance metric. The IPC of each category is the geometric mean of IPC for all traces within that category. We then normalize the IPC of each category to the baseline to show performance.

6. RESULTS

In this section, we present the experimental results for our proposed technique. Section 6.1 compares the FIT rate and Vccmin of MS-ECC with previously proposed techniques for reducing Vccmin. Section 6.2 evaluates the performance overhead of MS-ECC in the low-voltage mode and also compares the energy efficiency of MS-ECC with previous techniques.

6.1 Reliability

In this section, we compare the Vccmin of MS-ECC with the previously proposed bit-fix scheme [31]. We do not compare against the word-disable scheme from [31] because that paper showed that bit-fix enabled lower Vccmin than word-disable. The MS-ECC results in this section assume our most aggressive implementation with a segment size of 64 bits and 4-bit error corrections per segment. The bit-fix mechanism discussed in [31] lacks mechanisms to tolerate non-persistent failures. As a result, when soft errors and erratic failures are considered, the cache FIT rate is unacceptable unless SECDED ECC is added to address the high FIT rates. In light of this, we compare against a version of bit-fix augmented with SECDED ECC, referred to as BFXECC. Figure 8 shows the Vccmin of MS-ECC and BFXECC in the context of both the yield loss and FIT. At a yield loss of 10^{-3}, without considering FIT, BFXECC and MS-ECC enable Vccmin of 475mV, and 490mV, respectively. When considering FIT on a low-erratic process, both mechanisms deliver a Vccmin of 520mV and the difference between BFXECC and MS-ECC is negligible. On a high erratic process, MS-ECC does significantly better than BFXECC as the high frequency of erratic bit failures at low voltages overwhelms the SECDED ECC that augments BFXECC, thereby increasing its Vccmin to 630mV.

Table 2. Benchmarks

Category	# of traces	Example Benchmarks
Digital Home (DH)	60	H264 decode/encode, flash
FSPEC2K (FP2K)	25	www.spec.org
ISPEC2K (INT2K)	26	www.spec.org
Games (GM)	49	Doom, quake
Multimedia (MM)	77	Photoshop, raytracer
Office (OF)	52	Excel, outlook
Productivity	43	File compression,
Server (SERV)	48	SQL, TPCC
Workstation (WS)	82	CAD, bioinformatics
ALL	462	

It is interesting to note that the Vccmin for the low and high erratic processes are within 5 mV of each other for MS ECC. Because of this, we only include a single curve, MS-ECC-FIT (Hi/Low), to show the FIT pfail of MS-ECC for both process types. The type of erratic process has little effect on the FIT for MS-ECC because MS-ECC is a single repair mechanism being used to address all sources of failures.

On a high erratic process, MS-ECC clearly outperforms BFXECC. This is intuitive since BFXECC can only correct one erratic bit or soft error in every 512 bits, while MS-ECC can correct up to four such errors in each 64-bit segment if that segment contains no persistent failures. Even with some persistent failures in a 64-bit segment, it is likely that the 4-bit ECC can cover more than one erratic bit or soft error.

6.2 Performance and Energy Efficiency

In this section, we evaluate the performance overhead of MS-ECC. Like in Section 6.1, the default MS-ECC implementation in this section uses a segment size of 64 bits and 4-bit error corrections per segment. As failures are infrequent in the high voltage mode, the logic required for MS-ECC can be bypassed, resulting in no performance overhead in the performance-critical high voltage mode. During the low voltage mode, MS-ECC has 1-cycle latency overhead and sacrifices half of the cache, resulting in performance degradation.

To quantify the performance overhead of using MS-ECC in the low-voltage mode, Figure 9 shows normalized IPC relative to the defect-free low-voltage baseline, when applying MS-ECC to both the L1 and L2 caches. Note that the defect-free low voltage baseline has no persistent or non-persistent bit failures in the low-

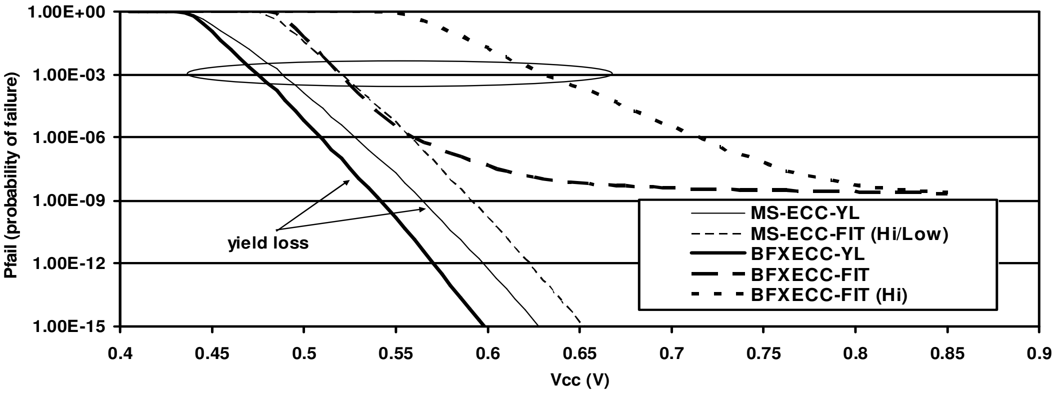


Figure 8. Yield Loss and FIT Rates for MS-ECC and BFXECC

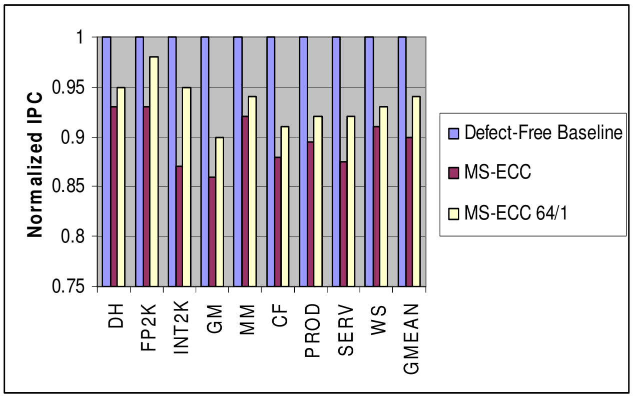


Figure 9. Performance Overhead of MS-ECC.

voltage mode. Averaging across all the workloads, MS-ECC results in 10% IPC degradation relative to the defect-free baseline. However, it is worth noting that this performance overhead is relative to an ideal configuration that has reliable caches with no area or latency overhead at supply voltages as low as 500 mV. Furthermore, the low-voltage mode is usually used when the processor load is low and energy efficiency, rather than performance, is the primary concern.

In Section 4, we proposed an adaptive mechanism where the OS can change the ratio of data ways to ECC ways for MS-ECC in order to trade off cache capacity for the desired Vccmin. Our default implementation labeled as "MS-ECC" in Figure 9 sacrifices 50% of the cache capacity by correcting 4 errors in every 64-bit segment. To demonstrate the cache capacity vs. Vccmin trade-off which the OS can make, we also analyzed an MS-ECC implementation, labeled as *MS-ECC 64/1* in Figure 9, which corrects one error for each 64-bit segment, and so only needs one ECC way for each 4 data ways. Since our default configuration is an 8-way cache, the closest approximation for *MS-ECC 64/1* uses two total ECC ways, thereby sacrificing 25% of the cache capacity in the low voltage mode. Figure 9 shows that *MS-ECC 64/1* degrades IPC by only 6% on average over the defect-free baseline (compared to a 10% average IPC degradation for the default MS-ECC). Other implementations permitted by our adaptive design, i.e., *MS-ECC 64/2* and *MS-ECC 64/3*, have performance and Vccmin characteristics between the two extremes (*MS-ECC 64/1* and *MS-ECC 64/4*).

Table 3 summarizes the achievable Vccmin and the frequency, power consumption, and energy-per-instruction (EPI) for different techniques during low-power mode operation. Note that, in contrast to the defect-free baseline cache of Figure 9, we use realistic caches as our baseline in Table 3. Also recall from Section 5 that the baseline caches use only SECDED ECC to tolerate both persistent and non-persistent bit failures. We normalize the power and EPI results for each technique to the corresponding results for the baseline, while showing absolute results for Vccmin and frequency. As mentioned in Section 5, we predict frequencies at different voltages by running circuit simulations. Note that since EPI represents power per unit performance, a lower value of EPI implies better energy efficiency, For power calculations, we assume that dynamic power scales quadratically with supply voltage and linearly with frequency. We also assume that static power scales with the cube of supply voltage [27]. The results for BFXECC in Table 3 assume a high erratic process. We omit the results for BFXECC in a low erratic process because they exhibit a similar Vccmin to that of MS-ECC.

Table 3. Comparison between Bit-Fix with ECC and MS-ECC (Cache with SECDED ECC is Used as Baseline)

Scheme	Vccmin (mV)	Frequency (MHz)	Norm. Power	Norm. EPI
Baseline	725	1400	1	1
BFXECC	630	1000	0.57	0.8
MS-ECC	520	700	0.29	0.58
MS-ECC 64/1	670	1200	0.75	0.88

Compared to the baseline processor with SECDED ECC, MS-ECC reduces Vccmin by 200mV while BFXECC reduces Vccmin by only 90mV. MS-ECC's lower Vccmin results in more power savings than other schemes during the low-power mode. Compared to the baseline configuration, the most aggressive MS-ECC implementation (i.e., MS-ECC 64/4) achieves a 71% reduction in power, significantly higher than BFXECC's 43% power reduction. As frequency decreases with supply voltage, MS-ECC's lower supply voltage results in longer execution time as compared to other schemes. However, as the low-voltage mode is normally used when the processor load is low, energy efficiency, rather than performance is the primary concern. Thus, even though MS-ECC 64/1 provides more cache capacity and higher performance than MS-ECC during low-voltage operation, MS-ECC 64/1's higher Vccmin makes it less energy-efficient than MS-ECC. While MS-ECC 64/1 and BFXECC decrease EPI by 12% and 20% respectively, MS-ECC's lower Vccmin results in a significantly better 42% decrease in EPI relative to the baseline configuration.

7. CONCLUSIONS

In this paper, we proposed a novel error tolerance technique called multi-bit segmented ECC (MS-ECC) to enable reliable ultra-low voltage cache operation. MS-ECC leverages error correction codes based on Orthogonal Latin Square Code (OLSC) to tolerate both persistent and non-persistent bit failures. Unlike previous proposals for reliable ultra-low voltage cache operation, MS-ECC does not incur additional testing overhead to isolate defective bits. MS-ECC uses a modular error correction mechanism which allows the operating system flexibility in deciding on the reliability and performance levels of the low-voltage operating mode. We showed that MS-ECC enables reliable cache operation at 520mV at the cost of 50% decrease in cache capacity during low-voltage operation. Compared to current designs with single-bit ECC correction, the most aggressive implementation of MS-ECC enables a 30% supply voltage reduction, which reduces power by 71% and energy per instruction by 42%.

8. ACKNOWLEDGMENTS

We would like to thank Dinesh Somasekhar and Muhammad Khellah for their invaluable feedback and Hongliang Gao for his contributions. We are also grateful to Prof. Kaushik Roy and Jaydeep Kulkarni for helping us with the bit failure data. We thank the anonymous reviewers for their comments and suggestions.

9. REFERENCES

[1] M. Agostinelli, et al., “Erratic Fluctuations of SRAM Cache Vmin at the 90nm Process Technology Node,” IEDM Technical Digest, pp. 655-658, Dec 2005.

[2] T. Austin, “DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design,” Proc. 32nd Annual Symposium on Microarchitecture (MICRO-32), pp. 196-207, November 1999.

[3] A. Bhavnagarwala, et al., “The impact of intrinsic device fluctuations on CMOS SRAM cell stability,” IEEE Journal of Solid State Circuits, Vol. 36, No. 4, pp. 658-665, April 2001.

[4] R. C. Bose and R. K. Ray-Chaudhuri, “On a Class of Error-Correcting Binary Group Codes,” Information and control, Vol. 3, pp. 68-79, 1960.

[5] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art-review," IBM J. Research Development, vol. 28, no. 2, pp. 124-134, Mar. 1984.

[6] C. Constantinescu, “Impact of Intermittent Faults on Nanocomputing Devices” DSN 2007 Workshop on Dependable and Secure Nanocomputing, June, 2007.

[7] J. Doweck, “Inside the Core™ Microarchitecture,” Proc. 18th IEEE Symposium on High-Performance Chips, August, 2006.

[8] S. Hareland, et al., “Impact of CMOS Scaling and SOI on Soft Error Rates of Logic Processes,” VLSI Technology Digest of Technical Papers, pp. 73-74, 2001.

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

[10] J. Ihm, et al., "An 80nm 4Gb/s/pin 32b 512Mb GDDR4 Graphics DRAM with Low-Power and Low-Noise Data-Bus Inversion." Proceedings of the 2007 IEEE International Solid State Circuits Conference. pp.492-493.

[11] Intel Corporation, “Intel® Celeron® Processor – Low Power/Ultra Low Power,” October 2001, http://download.intel.com/design/intarch/datashts/27350901.pdf.

[12] T. Karnik, et al., “Impact of Body Bias on Alpha- and Neutron-Induced Soft Error Rates of Flip_flops,” Symposium On VLSI Circuits Digest of Technical Papers, pp. 324-325, 2004.

[13] Y. Kawakami, et al., “Investigation of Soft Error Rate Including Multi-Bit Upsets in Advanced SRAM using Neutron Irradiation Test and 3D Mixed-Mode Device Simulation,” IEEE International Electron Devices Meeting, pp. 945-948, Dec. 2004.

[14] J. Kim, et al., “Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding,” In 40th International Symposium on Micro-architecture (Micro-40), December 2007.

[15] J. P. Kulkarni, K. Kim and K. Roy, “A 160 mV Robust Schmitt Trigger Based Subthreshold SRAM,” IEEE Journal of Solid-state Circuits, Vol. 42, no. 10, pp. 2303-2313, October, 2007.

[16] X. Li, et al., “Scaling of Architecture Level Soft Error Rates for Superscalar Processors,” Proc. 1st Workshop on the System Effects of Logic Soft Errors (SELSE), April 2005.

[17] S. Lin and D. J. Costello, “Error Control Coding,” Second Edition. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[18] W. Liu, J. Rho, and W. Sung, "Low-Power High-Throughput BCH Error Correction VLSI Design for Multi-Level Cell NAND Flash Memories," in Proc. IEEE Workshop on Signal Processing Systems (SIPS), Banff, 2006, pp. 248-253.

[19] S. Mukherjee, J. Emer, and S. Reinhardt, “The Soft Error Problem: An Architectural Perspective,” Proc. 11th International Symposium on High-Performance Computer Architecture (HPCA-2005), pp. 243-247, February 2005.

[20] K. Nakamura, and M. Horowitz, “A 50% Noise Reduction Interface Using Low-Weight Coding,” Symposium on VLSI Circuits Digest of Technical Papers, pp. 144 –145, June 1996.

[21] D. Roberts, N. S. Kim, and T. Mudge, “On-chip Cache Device Scaling Limits and Effective Fault Repair Techniques in Future Nanoscale Technology,” Proc. 1oth Euromicro Conference on Digital System Design (DSD 2007), pp. 570-578, 2007.

[22] F. Ruckerbauer and G. Georgakos, “Soft Error Rates in 65nm SRAMs – Analysis of new Phenomena,” Proc. 13th IEEE International On-Line Testing Symposium (IOLTS 2007), pp. 203-204, 2007.

[23] D. Schinkel, et al., "A Double-Tail Latch-Type Voltage Sense Amplifier with 18ps Setup + Hold Time." Proceedings of the 2007 IEEE International Solid State Circuits Conference, pp. 314-315.

[24] S. E. Schuster, “Multiple Word/Bit Line Redundancy for Semiconductor Memories,” IEEE Journal of Solid-State Circuits, Vol. SC-13, No. 5, pp. 698-703, October 1978.

[25] P. Shivakumar, et al., “Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic,” Proc. International Conference on Dependable Systems and Networks, pp. 389-398, June 2002.

[26] J. Srinivasan, et al., “The Case for Lifetime Reliability-Aware Microprocessors,” Proc. 31st International Symposium on Computer Architecture (ISCA '04), pp. 276-287, June 2004.

[27] Y. Taur and T. H. Ning, “Fundamentals of Modern VLSI Devices,” Cambridge University Press, 1998, pp . 144.

[28] TSMC standard cell libraries. http://www.cadence.com/partners/tsmc/SC_Brochure_9.pdf

[29] K. Ünlü, et al., “Neutron-induced Soft Error Rate Measurements in Semiconductor Memories,” Nuclear Instruments and Methods in Physics Research Section A, Volume 579, Issue 1, pp. 252-255, 2007.

[30] C. Weaver, et al., “Techniques to Reduce the Soft Error Rate of a High-Performance Microprocessor,” Proc. 31st International Symposium on Computer Architecture (ISCA-31), pp. 264-275, June 2004.

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

[32] K. Wu and D. Marculescu, "Soft Error Rate Reduction Using Redundancy Addition and Removal," in Proc. IEEE/ACM Asian-South Pacific Design Automation Conference (ASPDAC), Seoul, Korea, Jan. 2008

[33] J.F. Ziegler, et al., “Accelerated Testing for Cosmic Soft-Error Rate,” IBM Journal of Research and Development, Vol. 40, No. 1, pp. 51-72, January 1996.