Operating SECDED-Based Caches at Ultra-Low Voltage with FLAIR

Moinuddin K. Qureshi
Georgia Institute of Technology
Atlanta, GA, USA
moin@ece.gatech.edu

Zeshan Chishti
Intel Labs
Hillsboro, Oregon, USA
zeshan.a.chishti@intel.com

***Abstract*—Voltage scaling is often limited by bit failures in large on-chip caches. Prior approaches for enabling cache operation at low voltages rely on correcting cache lines with multi-bit failures. Unfortunately, multi-bit Error Correcting Codes (ECC) incur significant storage overhead and complex logic. Our goal is to develop solutions that enable ultra-low voltage operation while incurring minimal changes to existing SECDED-based cache designs. We exploit the observation that only a small percentage of cache lines have multi-bit failures. We propose *FLexible And Introspective Replication (FLAIR)* that performs two-way replication for part of the cache during testing to maintain robustness, and disables lines with multi-bit failures after testing. FLAIR leverages the correction features of existing SECDED code to greatly improve on simple two-way replication. FLAIR provides a Vmin of 485mv (similar to ECC-8) and maintains robustness to soft-error, while incurring a storage overhead of only one bit per cache line .**

I. Introduction

Energy efficiency is becoming the single most important design constraint for computing systems. Successive generations of microprocessors have relied on supply voltage reduction as one of the most effective techniques to reduce the power consumption of a microprocessor. However, as supply voltage continues to decrease, the effects of semiconductor process variations become more pronounced, resulting in increased circuit failures. These failures limit the safe operating voltage of a microprocessor to a value *Vmin*, beyond which the processor ceases to operate reliably. Failures in large memory structures, such as caches, which dominate the die area, typically determine the Vmin for a processor [14]. For example, for the baseline 8MB L3 cache, the Vmin would be limited to approximately 850 mv.

Several recent papers have proposed architecture-based techniques to improve the reliability of large caches at low operating voltages [1][2][3][6][11][14][13]. These techniques allow cache bits to fail, but use additional redundancy, such as multi-bit error correcting codes (ECC) to tolerate high bit failure rates. Figure 1 shows the Vmin for our baseline 8MB L3 cache as the error correction level is changed on a per-line basis. We denote ECC-N as an error correction scheme that can correct up-to N errors in the line. Vmin reduces as the ECC strength per line is increased. We define ultra-low voltage to be a region below 500mv, which needs ECC-7 or higher level of ECC.

We want to operate the cache at well within the ultra low voltage region (say 485mv) so we will need the capability of ECC-8 to support such an aggressive Vmin target. Implementing such high levels of error correction requires

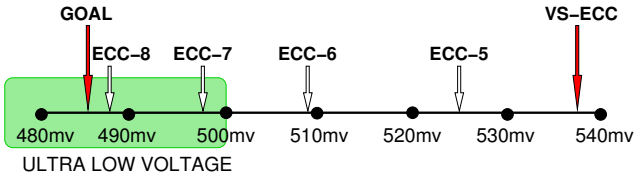

Fig. 1. Impact of ECC on Vmin. To operate in an ultra low voltage regime (sub 500mv) we need ECC-7 or stronger code. Our goal is to obtain Vmin of 485mv, without incurring significant hardware or latency overheads for a SECDED-based cache.

significant storage overhead for ECC check bits and complex logic for ECC encoding and decoding. Higher levels of ECC also reduces performance because of the long latency of ECC decoding. Thus, the storage, complexity, and latency overhead of multi-bit ECC make it less appealing for practical implementation. From a commercial viewpoint, it is desirable to have a single mainstream chip (say designed for high performance operation) which can still be made to work in low voltage domains with minor changes to existing design.

Given that existing caches already employ Single-Error-Correcting Double-Error-Detecting (SECDED) codes to tolerate single bit failures due to soft errors (such as [5], [12]), we would like to leverage this existing hardware to tolerate both hard errors due to low voltage operation in addition to soft errors. Ideally, we would like to have a practical hardware solution that satisfies the following six requirements:

1) It should enable the cache to operate at ultra-low voltage, say 485 mv in our case.
2) It should incur negligible storage overhead and almost no changes to the existing hardware. This rules out prior schemes that rely on multi-bit ECC.
3) It should be robust to soft errors, without relying on additional storage overhead (we want to maintain error tolerance at-least at the level of SECDED).
4) It should provide almost all of the cache capacity ($> 90\%$) during normal program execution.
5) It should have a cache access latency similar to baseline during the normal program execution (this again eliminates multi-bit ECC due to high latency).
6) It should not rely on having non-volatile memory on-chip or require software changes for deployment.

This paper proposes a scheme that satisfies all the six requirements. To devise our solution, we exploit the insight that even at ultra low voltages only a very small percentage of cache lines have multi-bit failures [2], [13]. Table I shows

978-1-4799-0181-4/13/$31.00 ©2013 IEEE

the percentage of cache lines that have zero error, one error, or two-or-more errors at the target Vmin of 485mv[1] For our target Vmin, less than 10% of the lines would have more than one-bit error. If we can identify such faulty lines, then we can enable low-voltage operation by disabling these lines.

TABLE I. LINE FAILURE STATISTICS AT 485MV

Percentage lines with		
0-Error	1-Error	2+ Errors
60.0%	30.7%	9.3%

A recent study, *Variable Strength ECC (VS-ECC)* [2], proposed a mechanism to perform runtime testing to characterize the error level of each line and allocate appropriate amount of ECC for each line. Unfortunately, the runtime test runs for a long time (several tens of seconds) and VS-ECC still relies on operating a quarter of the cache with high-strength ECC-4 during the testing phase. Thus, the Vmin obtained with VS-ECC is limited by ECC-4 and it still incurs the storage overhead (albeit for only quarter of the cache) and logic complexity of expensive multi-bit ECC decoding. As we seek a Vmin of 485mv, we would need to extend VS-ECC to have ECC-8 in the quarter of the cache during testing phase, further exacerbating the storage, latency, and complexity overheads.

In this paper, we obviate the need for storage and logic for multi-bit ECC. Instead, we use the existing cache structure for reliable operation during testing phase. We call this proposal *FLexible Replication (FLexR, pronounced as "flexer")* as replication is enabled only during the testing phase (to tolerate multi-bit errors) and disabled during the post-testing phase (to provide more cache capacity). As *Dual Modulo Replication (DMR)* is effective only at detecting faults, but not at correcting them, we modify the cache architecture during testing phase to make error detection as the primary objective instead of error correction. We do this by making the cache write through during the testing phase (for the post testing phase the cache is still used as a write-back cache). If the DMR check of FLexR detects an uncorrectable error during the testing phase, FLexR simply invalidates both lines in the pair and reads the value from memory.

We found that simple two-way replication (even after per-line correction of SECDED) is unable to tolerate more than three errors in the pair, and is vulnerable to soft-error. We enhance the robustness of FLexR by leveraging the observation that each line in the pair undergoes SECDED operation and we can use the SECDED status of each line in DMR (whether the line is good, or has a correctable error, or has a detectable but uncorrectable error) for improved robustness. We propose, *FLexible And Introspective Replication (FLAIR)* which performs a self-check (introspection) on the SECDED status of the line in addition to the DMR check. FLAIR can ignore the output of the DMR comparison, depending on SECDED correction status of each line in the DMR group. We show that FLAIR is robust at Vmin of 485mv and can tolerate soft errors both during the testing phase as well as during the post-testing phase. Thus, FLAIR provides a Vmin similar to ECC-8, without adding any extra storage (except one bit per line) and incurring negligible complexity.

[1]Similar to previous studies, we assume that the Vmin is calculated at the voltage at which one out of 1000 caches is expected to fail.

After the testing phase finishes, the two-way replication gets disabled and the lines can be used for normal operation. As shown in Table I, operating at a Vmin of 485mv would result in 31% lines with one bit hard faults. Given the vulnerability of cache lines to soft errors, SECDED protection alone will not be enough for such lines, and previous approaches would argue for disabling such lines after the testing phase finishes. Thus, during the normal operating phase the program would get only 60% of the available cache capacity. We observe that for the lines with only one hard error (31% of the lines), if a single bit soft error strikes such a line, then SECDED can still identify the error. We leverage this key observation, and propose a *Weak Line Reclamation (WLR)* scheme that restrict such lines to only store clean data. Thus, if a soft error happens in a line with 1 hard error, the SECDED mechanism identifies the fault. If the line is clean, we can simply invalidate the line and read the copy from memory. With WLR, the cache can retain more than 91% capacity after the testing phase. Thus, our proposal enables low voltage operation, at near full capacity, while incurring negligible storage and logic overhead and maintaining tolerance to soft errors, which makes practical to implement ultra low-power mode in future processors.

II. BACKGROUND AND MOTIVATION

A. Impact of Cache Reliability on Vmin

Most of the modern microprocessors dedicate a majority of their transistor budget to large SRAM caches. To guarantee correct software execution, these large on-chip caches must operate in an error-free manner. However, parametric variations induced by the imperfections in semiconductor manufacturing process make SRAM cells susceptible to failures. Figure 2 shows the bit failure data from [9], [6], which specifies the failure probability of a single bit of cache at different supply voltages. This data shows that bit failure rate increases with reduction in supply voltage. Consequently, the supply voltage of a microprocessor is often limited to a value Vmin, below which the failure rate increases beyond the error mitigation capability of the cache. The Vmin for a microprocessor product is often specified as a function of acceptable yield loss. For example, recent papers on reliable low voltage operation have defined Vmin as the voltage at which at least 999 out of 1000 caches operate reliably.

Modern microprocessors often use multiple power modes to execute in an energy-efficient manner. When higher performance is desirable, the processor operates at a higher supply voltage, whereas when performance is not as critical, the processor transitions to a low voltage mode to save energy. Therefore, reducing the Vmin of a processor is critical towards enabling higher energy efficiency. However, mechanisms that reduce Vmin should not have a significant impact on performance during the high voltage execution mode.

B. Prior Work: Circuit Solutions and Dual VDD

Both circuit- and architecture-level solutions have been proposed to mitigate bit failures at low voltages. Circuit solutions typically make changes to the SRAM cell design to neutralize the impact of process variations [9]. These schemes either upsize the transistors or use cell design variants such

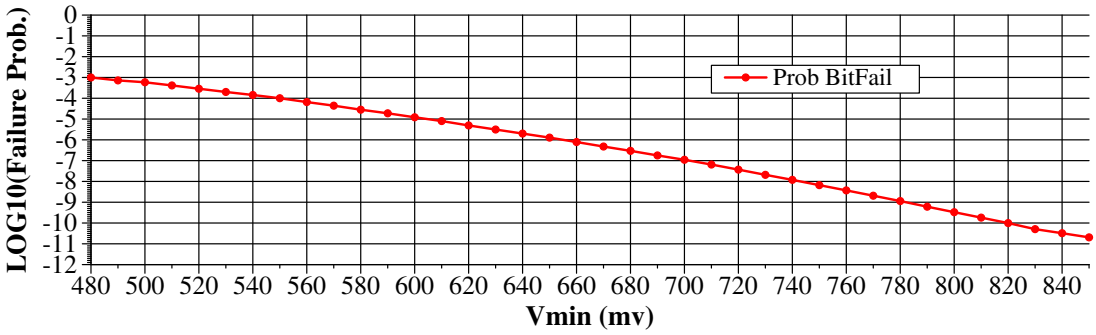

Fig. 2. Probability of bit failure as a function of operating voltage. Bit failure data is derived from prior studies [9], [14], [6], [2]. Note that at the target voltage of 485mv the probability of bit failure is 1 out of 996.

as the 8T, 10T, and ST SRAM cells. However, the resulting Vmin reduction comes at the cost of significant increases in area (e.g., 100% area increase for ST cell [9]). While such solutions may work well for small caches (such as L1 and L2), they incur impractical overhead for large last-level caches (LLC). In our work, we will assume that L1, L2, and the LLC tag-store are protected by such circuit techniques and focus on practical solutions for LLC data-store (which tends to consume majority of the on-chip transistors).

Another option is to use Dual-VDD, whereby cache and core operates on different power supply circuits. Unfortunately, Dual-VDD incurs significant area and complexity. Furthermore, if the core supply voltage is reduced significantly but the LLC continues to run at a higher voltage, then the LLC power starts to dominate the platform power.

C. Prior Work: Avoiding Faulty Cells with Fault Maps

If the data about which cells in the cache are faulty is available then we can employ efficient error correction to tolerate faulty cells. Architectural mechanisms for Vmin reduction typically rely on such off-line information about cell failures and trade-off cache capacity for increased reliability. Wilkerson et al. [14], Roberts et al. [11], and Ansari et al. [4] proposed techniques to disable defective portions of the cache during the low voltage mode. These techniques sacrifice a portion of the cache to either save the locations of defective bits and the values of correct data, or pair faulty lines. These techniques degrade performance at low voltages due to the smaller cache capacity, incur higher latency due to parallel accesses to demand lines and repair locations.

D. The Rationale for Avoiding Non-Volatile Fault-Maps

Most of the architectural work on making caches robust at low voltages rely on having the faulty cell locations available. Unfortunately, passing this information from design time testing to runtime system is non-trivial. This would require that the processor chip be equipped with substantial amount of non-volatile memory (few bits per cache line). Integrating such large-scale non-volatile memory on-chip would require embedding a separate technology, thus making such solutions a high-cost and complex proposition. While current processor and memory designs do employ fuses to decommission faulty storage (at large granularity), extending these fuses to store faulty bit locations is quite expensive. For example, the fuses that are use to disable DRAM rows incur an area of approximately 5000 DRAM cells for each bit of fuse [7]. If we translate this into SRAM overheads, each bit of fuse would cost approximately few hundred SRAM cells, making it impractical to employ such fuses on a per cache-line basis.

E. Runtime Testing and VS-ECC

While the need for having locations of faulty cells can be avoided with multi-bit ECC (for example, as done by Chishti et al. [6]), simply having high strength multi-bit ECC for all lines is costly in terms of area, latency and complexity. A recent paper proposed *Variable-Strength ECC (VS-ECC)* [2], which addresses the limitations of previous ECC solutions by using the ECC budget in a non-uniform manner.

VS-ECC dedicates SECDED ECC to each line for soft error mitigation, while using additional ECC check bits to protect 4 out of 16 cache lines in each set from up to 4 bit failures. To identify lines with multi-bit failures, VS-ECC runs a testing phase before each transition to the low voltage mode. During the testing phase, VS-ECC stresses each line with different pre-determined testing patterns to uncover the different modes of bit failures. It was shown that several tens of seconds of testing is required to achieve acceptable coverage.

F. Need for Operational System During Runtime Test

Storing the testing information on disk requires software support. In order to avoid such software changes, and to provide periodic testing, VS-ECC recommended that testing be performed whenever the machine restarts. Given that testing requires few tens of seconds, this would increase boot time on every power up and degrade user experience if the machine is unavailable for few tens of seconds (e.g. think boot-up latency of SSD vs HDD). Hence it is desirable to have a working system even during testing.

To mitigate the performance overhead of the testing phase, VS-ECC performs testing in a pipelined fashion. It divides the cache into multiple portions and keeps one of the portions active for normal program execution while the other portion(s) is being tested. To tolerate bit failures in the active portion, VS-ECC dedicates all the check bits for multi-bit correction to the active portion. Since, only one quarter of the cache ways (4 out of 16) can be protected by multi-bit ECC, the size of the

active portion is limited to one quarter of the cache. Once the testing phase finishes, the information collected during testing is used to re-assign the check bits in proportion to the number of errors in the line.

G. Need for a Practical Solution

Ideally we would like to have a solution that enables low voltage operation of the processor chip without relying on significant storage or logic overhead and causing negligible changes to existing structures. Changing cache structure, and implementing complex circuitry not only requires area overhead but also entails effort from design, verification and testing teams. We would like to minimize these overheads. Unfortunately, all previous approaches, including VS-ECC, require significant changes to existing cache structure (to add extra ECC bits), and complex ECC decoding circuitry. Furthermore, because VS-ECC uses ECC-4 in the quarter of the cache during testing time, its effectiveness is limited to Vmin obtained with ECC-4 (540mv regime). We can obtain even lower Vmin and avoid the hardware changes required by VS-ECC if we can use existing cache circuitry for reliable operation during the testing phase and simply disable faulty lines during the normal phase. Based on this insight, we propose *FLexible Replication (FLexR)* that performs dynamic two-way replication of lines for robustness during testing phase. We will first describe the basic architecture of FLexR, before describing the enhanced version our proposal in Section IV.

III. Flexible Replication: Design and Analysis

Similar to VS-ECC, our proposal relies on runtime testing to identify faulty lines. However, unlike VS-ECC, it does not need precise number of faults in the line and instead bins the lines into three categories: no faults, exactly one fault, and two-or-more faults. Testing is still performed in pipelined fashion on a way-by-way basis. As testing takes a long time (50 seconds or more [2]), it is desirable to have at-least some portion of the cache usable during the testing phase. Unlike VS-ECC, which relies on having ECC-4 for a quarter of the cache, our design relies on alternative low-cost means to provide robustness during the testing phase.

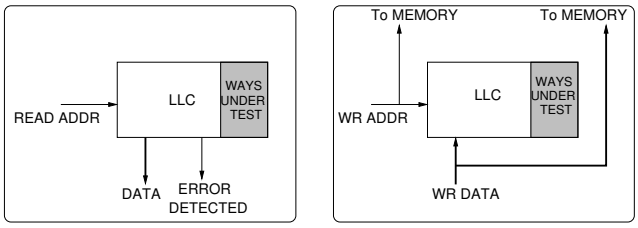

Fig. 3. Basic architecture to support Flexible Replication. The data-flow is shown only for the testing phase, for read (left) and for write(right).

A. Flexible Replication Architecture

One of the major design choice that we make to keep FLexR simple and practical is to eliminate the presence of dirty lines in the last level cache during the testing phase, as shown in Figure 3. While this increases the traffic to memory, we rely on two factors to keep this overhead manageable. First, even if the LLC is used as write-through, the L2 cache is still architected to be a writeback cache. So, only the dirty lines evicted out of the L2 cache will be written to memory and not the unfiltered stream emanating from the processor (we found that with our write-through design, the memory traffic during testing mode increased by 2% compared to VS-ECC on average, See Appendix A). Second, the extra memory traffic due to write through is incurred *only* during the testing phase (which although is few tens of second, would be much smaller than the up-time of the machine). After the testing phase, the cache still operates as a write-back cache.

Making the cache write through in the testing phase has the major advantage that it transforms the cache reliability problem from an error-correction problem into a much more tractable error-detection problem.

What we ideally want is a storage-efficient on-demand error detection, that does not incur any storage overhead compared to SECDED. Unfortunately, SECDED codes can detect only up-to two errors.[2] For our target operating Vmin a line can have up-to 8 errors, therefore if we want to provision each line with an error detection code, that code will require to ensure that the minimum hamming distance between valid code words is 9, and will incur a hardware overhead similar to ECC-4, which we are trying to avoid. Another alternative is to consider checksum codes or cyclic redundancy codes (CRC). However, these codes can detect multi-bit faults efficiently only if the errors happen in spatially close positions. For, multi-bit errors that can happen randomly, simple CRC codes and checksum codes are not effective at provide guaranteed detection. Furthermore, we want to avoid the storage and logic overheads associated in designing another coding scheme, in addition to the existing SECDED.

Thus, we need a way to enable part of the cache with multi-bit error detection capability without the storage and logic overheads associated with typical multi-bit error detection schemes. We leverage the insight that during the testing phase, only part of the cache is operational anyways, so we can used the non-operational part for storage-efficient error detection. We propose to replicate two lines in a *Dual Modulo Redundancy (DMR)* fashion for error detection. While this may seem to reduce the effective size to only half the cache capacity, we note that prior VS-ECC proposal have enabled only quarter of the cache capacity in order to provide cache space during the testing phase. So, the effective cache capacity with our design is higher than provided by VS-ECC.

B. Cache Structure with FLexR

Figure 4 shows one of the set of an 8-way set associative cache with FLexR, where we have used DMR for error detection. Ways 0 and 1 each store a copy of Line A; ways 2 and 3 each store a copy of Line B; and ways 4 and 5 each store a copy of line C. Way 6 and 7 are not available as they are undergoing testing. On a read access, the two lines in the DMR pair are read, SECDED correction is performed on each of the two lines. If SECDED detects a correctable failure, it

[2]In reality, SECDED codes can detect up-to three errors if we give up on correction of lines with single bit failure. However, given that for our target operating Vmin of 485mv we have about 30% of the lines with single bit error. Therefore, the correction capability of SECDED is important even in the testing phase, and hence we use per-line error correction storage as SECDED instead of ZECTED (Zero Error Correction Triple Error Detection).

performs correction and then supplies a corrected copy of the line for the DMR comparison. Whereas, if SECDED detects an error that is uncorrectable it simply supplies the original value (as correction is likely to increase the number of bit errors). DMR is performed on the two lines on a bit-by-bit basis. If at-least one of the bits do not match, then an error is detected and the two lines in the DMR pair are both invalidated, and the corresponding data line is read from memory. When an error is identified, FLexR marks both lines in the pair as disabled (using a cache line disable bit) during testing mode so that we can avoid future invalidation on these lines.[3] The logic overhead of DMR checking is 512 two-input ex-or gates and 256 2-input OR gates, this logic complexity is almost an order of magnitude lower than required for SECDED, and the latency for this DMR check is approximately 9 FO4 (at-most one processor cycle).

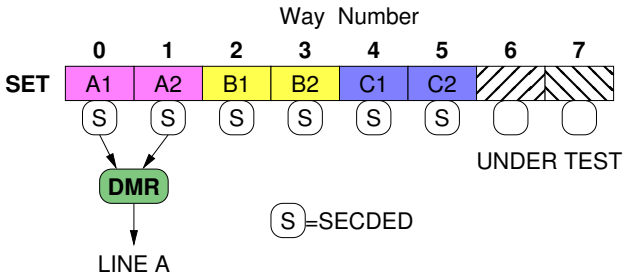

Fig. 4. One set of an 8-way set-associative cache that facilitates FLexR. The first six ways provide 3 ways of DMR, the other two ways are reserved for testing.

C. Pair Failure Rate at Target Vmin

Our baseline 8MB cache is 16-way, and contains 2^{17} lines. During testing mode we dedicate two ways for testing and use the remaining 14 ways to implement DMR. Thus the total number of pairs that must be supported during testing is $\frac{7}{16} \cdot 2^{17}$. To guarantee that no more than 1 in a 1000 caches fail, we would need to ensure that no more than one in every $1000 \cdot \frac{7}{16} \cdot 2^{17} = 2^{25.7}$ pair can fail. Thus, we seek an effective target failure probability of the pair to be $2^{-25.7}$. We will call this critical value as *Target Pair-Failure (TPF)*.

Table II shows the probability of number of errors in the pair of two lines (64 bytes each) at an operating voltage of 485mv. For the purpose of analyzing the vulnerability of DMR, we need to consider only up-to four errors in the pair, hence we club five or more errors into a single bin. Analysis with a larger number of errors will be performed in Section IV.

TABLE II. EXPECTED NUMBER OF ERRORS IN THE PAIR AT 485MV (5+ DENOTES FIVE OR MORE ERRORS).

Num Errors in Pair					
0	1	2	3	4	5+
35.9%	36.8%	18.8%	6.4%	1.6%	0.4%

Table II shows why DMR without SECDED correction would not be very useful. DMR without SECDED would cause failure if each line in the pair has one failed bit and the positions of the faulty bits overlap. The vulnerability of DMR without SECDED can be computed as a product of three components: First, probability of two errors in the pair (prob=18%). Second, the probability that each line in pair will get one error each (prob=0.5). Third, the two bits will land in the same position of two lines (prob=1/512). Thus, the vulnerability of DMR is approximately 2^{-12}, which is much higher than the TPF we seek ($2^{-25.7}$). Furthermore, if we want soft error tolerance, then DMR without SECDED may cause fault even if the pair has one hard fault (if the soft error happens in the other line in the pair at the same position). Thus, DMR alone (without SECDED) falls quite short in terms of robustness for our requirements.

D. Shortcoming of FLexR

FLexR uses SECDED correction before employing DMR. While SECDED is effective in correcting one-bit errors, it does not attempt to correct lines with two errors (as miscorrection can increase the number of errors in the line, and the position of the miscorrected bit would depend on data value). We analyze the detection capability of FLexR under two settings: with and without soft-error tolerance.

For DMR to cause failure the component lines must have at least one failed bit after SECDED correction. Thus, for DMR to fail, we need at-least two erroneous bits per line. The most dominant failure case would then be when the pair has four errors, each line gets two errors, and those the positions of the errors overlap, as shown in Figure 5.

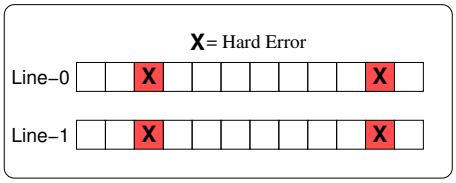

Fig. 5. Dominant mode of failure for a pair with FLexR, if we consider only hard errors.

Thus, failure probability of FLexR can be computed as the product of three components. First, the probability that the pair has four errors (prob: 1.6%). Second, each line gets two errors each (prob: 6/16). Third, the position of errors overlap in the two lines (prob: $\frac{2}{512 \cdot 511}$). Thus, the vulnerability of a given pair under these conditions is equal to $2^{-24.5}$, which is higher than our TPF of $2^{-25.7}$. While this difference may seem small (a difference of only about 2x), this analysis assumes that hard errors are the only source of vulnerability and does not take into account soft errors.

We want our solution to not compromise on soft error tolerance at all compared to SECDED. Therefore, we have to provision for the case that any bit in the line can get a fault due to soft error. Given the low rate of soft-error we will assume that there can be at-most one error in the pair. When we include the vulnerability to soft-error in our analysis, the dominant failure case for FLexR is when the pair has 3 errors, there is a (1,2) split of errors between the two lines in the pair, and the position of error in 1 error line overlaps with the position of error of one of the two errors in the line with 2

[3]Note that such a disable with DMR is valid only before the pairs are tested. After testing, we may chose to disable only one line in the pair.

errors. Then, if a soft-error strikes the line with 1 error, in a position overlapping with the second error in the 2-error line, FLexR will be unable to detect this error, as shown in Figure 6.

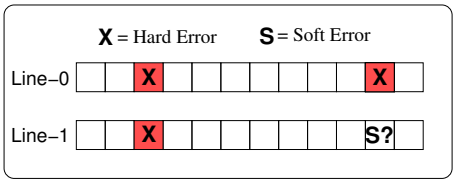

Fig. 6. Dominant mode of failure for a pair with FLexR, considering both hard and soft error.

The vulnerability in this case can be calculated as the product of three components. First, the pair has three failure (prob: 5.9%). Second, one line gets two errors and the other gets one error (prob: 6/8). Third, there are two errors in the same bit position (prob: 2/512). Thus, the overall vulnerability is approximately 2^{-14} which is almost an order of magnitude higher than the TPF we seek $2^{-25.7}$.

In summary, employing simple DMR after SECDED, as done with FLexR, is not sufficient to provide 485mv operation while tolerating soft errors. The next section describes a simple and effective extension that can provide reliable operation at 485mv while maintaining soft-error resilience.

IV. FLexible And Introspective Replication

One of the common case of failure of FLexR happens when both lines in the DMR pair have a detectable error. To enhance the detection capability of FLexR, we observe that FLexR does not utilize the correction status of SECDED. We could obtain improved robustness in such a scenario by noting that two lines in the pair had an uncorrectable error, and simply indicating that the pair has an undetectable error, rather than providing the incorrect data. Based on this insight, we propose *FLexible And Introspective Replication (FLAIR)*. FLAIR performs Flexible Replication (DMR is employed only in the testing phase and disabled during the normal mode) and Introspective Replication (SECDED status of each line is checked for decision making). Before we discuss the detection algorithm of FLAIR, we will first provide the basic working of SECDED code as the number of errors in the line is varied (Figure 7). This understanding is the key in developing the effective detection algorithm of FLAIR.

A. A Primer on SECDED Code

We will limit our discussion to SECDED implementations based on Hamming code. Hamming codes can perform error correction by adding a few extra bits (called check bits) to the data word, and this combination of data-word and check-bits is called a *codeword*. The check bits are determined as a parity over a subset of data-bits and the parity bits. The key to hamming code is to have the parity bits overlap, such that they manage to check each other as well as the data.

In order to correct a single erroneous bit, we need that the valid code words are at least a hamming distance of three away from each other. In that case, if one error occurs, then the regeneration of parity bits (called syndrome) will indicate the position of the bit that has failed. A zero syndrome indicates that there is no error detected (bit positions are numbered from 1 onward). For implementing such a Single Error Correcting (SEC) code for data-word consisting of N bits, we need $1 + log_2(N)$ check-bits. Thus, implementing SEC for a line with 512 bits requires 10 check bits.

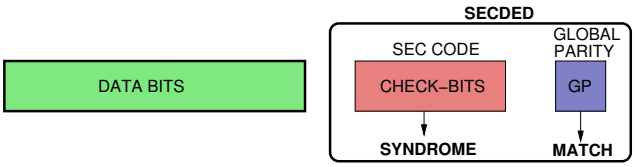

Fig. 8. Typical structure of SECDED, consisting of SEC and global parity. SYNDROME indicates error-position and MATCH indicates global-parity agreement.

As SEC has a minimum distance of 3, it can either only correct 1 bit error or detect 2 bit error but not both. To extend SEC to have guaranteed double error detection, we add a global parity bit GP which tracks the parity over both the data-word and check-bits, as shown in Figure 8. If there are two errors in the (data-word+check-bits) the SEC will provide a non-zero syndrome (possibly causing error in an error free bit). However, if there are 2 errors then the GP bit will result in a parity match, indicating either zero or even number of errors. Given non-zero syndrome for SEC we know that the number of errors is the codeword is non-zero, and we can flag a detectable-but-uncorrectable error with SECDED.

For SECDED, the codewords are at least a distance of 4 from each other. However, this does not mean that the valid code word are placed exactly at a distance of 4, they may be placed at any distance greater than or equal to 4. For our work, we analyze what happens to SECDED at a distance larger than 4, as shown in Figure 7.

The operation of SECDED can be generalized to any number of errors. If there is a GP match, and SEC indicates zero syndrome, then SECDED estimates such a line to have no error. If GP indicates mismatch, and SEC indicates non-zero syndrome then SECDED estimates that this is a correctable error. If the parity and syndrome do not agree (one indicates no error, and the other indicates error) then SECDED can denote such lines as detectable-but-uncorrectable error lines.[4]

B. Classifying Lines Based on SECDED Status

SECDED is based on two outcomes: one from SEC and second from GP. Based on the result of these two outcomes, the lines can be classified into one of three types after SECDED correction:

1) **Good Line (G Line):** Indicates a line for which SECDED estimates no error. It will not modify the contents of the line. This estimation if correct

[4]When GP indicates mismatch but SEC provides non-zero syndrome, this can happen if there is a single bit error only in the GP bit. During the testing phase we will pessimistically assume that a strike only on the GP bit still results in an uncorrectable error. During the post-testing phase the SECDED logic can ignore the error if GP indicates a mismatch and SEC syndrome is zero (as this will happen either for a triple bit error or error only in GP, and we do not expect 3-bit errors in the same line during the post-testing mode).

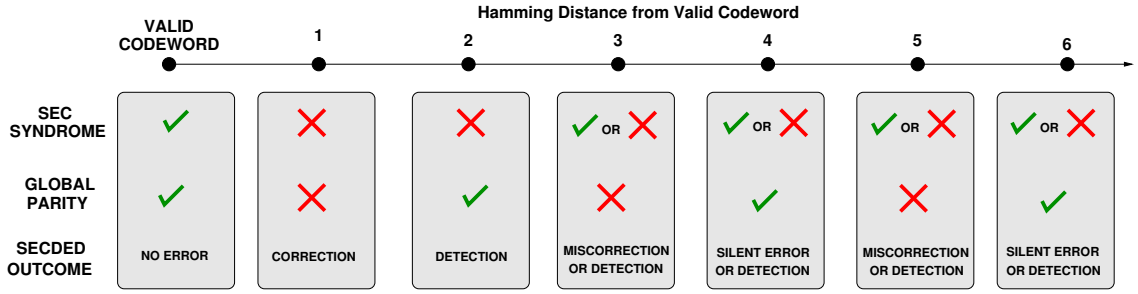

Fig. 7. Dissecting Hamming code based SECDED for introspective replication. For SEC a checkmark indicates zero syndrome and cross indicates non-zero syndrome. For global parity, a check indicates parity matches and a cross indicates mismatch. The status of SEC and GP considered separately can be used to detect a class of multi-bit errors.

signifies a good line, and we denote such a line as *G+ line*. However, if it is incorrect (4 or 6 or 8 etc. errors) then this line will be incorrectly regarded as a fault-free line resulting in silent error, and we denote such a line as a *G- line*.

2) **Correctable Line (C Line):** Indicates a line that has an odd number of errors. If there is one bit error, SECDED will correct the error, and we denote such a line as a *C+ line*. However, if the line has more than one errors then SECDED will cause miscorrection, and the number of errors in the line would increase by one, and we denote such a miscorrected line as a *C- line*.
3) **Detectable Line (D Line):** Indicates a line that has an either two or more error, and the syndrome and parity mismatch. SECDED does not modify the contents of a *D line*.

On a read access, the two lines in the DMR pair are sent through the SECDED circuit. We augment the SECDED circuit to provide the line type as deemed by SECDED (G,C,or D). This line type information is used to drive the detection decision of FLAIR.

C. Detection Algorithm of FLAIR

FLAIR relies on two levels of error detection, as shown in Figure 9. First, the status of the SECDED and second DMR. The overall detection mechanism of FLAIR can be tuned towards high robustness or more cache capacity. For example, we can implement FLAIR in a conservative fashion where DMR is performed only if both lines are estimated to be G lines from SECDED. However, from Table II we observe that 39% of the pairs are expected to have one error, therefore this simple approach will discard almost one-third of the pairs in testing mode.[5] Therefore, we chose to have capability to correct at-least one error in the pair. The detection algorithm for FLAIR we employ is to rely on the DMR check only if the pair has at-least one G line. Otherwise, the pair is deemed as faulty, without the need to check DMR.

[5]The capability of correcting one error in each of the two lines is not as essential as it has much smaller impact on capacity. This can be computed as the product of: probability that the pair has two errors (prob=18%) and the probability that both lines will get one error each (prob=50%), so a capacity loss of 9% of the total pairs

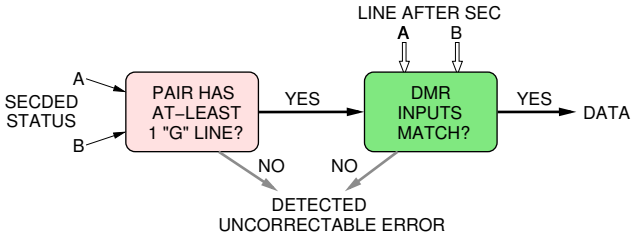

Fig. 9. The two-phase detection algorithm of FLAIR. DMR is employed only if SECDED estimates that at-least one line is a G line (in reality it may be G- or G+).

D. Robustness of FLAIR to Multi-bit Errors

To assess the vulnerability of FLAIR to multi-bit errors, we will consider case by case the number of of errors in a pair. To separate the effect of mechanism robustness from bit failure probabilities, we will divide the analysis into two parts. First, we will compute the *FLAIR Vulnerability Factor (FVF)*, which indicates the probability that if a given multi-bit error event happens, what is the likelihood that FLAIR will not be able to detect it. We can multiply FVF to *Raw Event Probability (REP)* (which will be a function of operating voltage) to get the *Event Vulnerability Factor (EVF)* for that event. For analysis with soft-error strike and miscorrection of SECDED, we will be overly pessimistic in our calculations, and assume that these events will *always* flip the most vulnerable bit.

1) **Pair has up-to 5 errors:** FLAIR failure probability is zero. FLAIR needs at least one line to be G- (which needs at least four errors). If the pair has 5 errors, and one line has four errors (for G-) then the other line will have 1 error and SECDED will repair the line. DMR will detect error between the corrected line and the fault line. Thus, FVF for this case is 0.
2) **Pair has 6 errors:** Failure can occur if one line has 4 errors (G- line) and the other line has the two error positions overlap. Then, if a soft error strike happens in the position of third error, and it gets miscorrected such that the new error is in the position of fourth error. The probability that there is a (2,4) split of 6 errors is $\frac{30}{64}$, and the probability that the positions of the two errors match can be computed[6] as $\frac{12}{523 \cdot 522}$.

[6]For a line with 512 bits, SECDED would require 11 bits, so the total number of bits in the protected line is 523.

TABLE III. COMPUTING THE VULNERABILITY OF FLAIR FOR DIFFERENT NUMBER OF ERRORS IN THE PAIR.

Num Errors in Pair	Raw Event Prob. (REP) at 485mv	FLAIR Vulnerability Factor (FVF)	Event Failure Probability (EVF=REP*FVF)
0-5	99.99%	0	0
6	$2^{-10.8}$	$2^{-15.5}$	$2^{-26.3}$
7	$2^{-13.5}$	$2^{-23.4}$	2^{-37}
8	$2^{-16.5}$	$2^{-33.4}$	2^{-50}
9	$2^{-19.6}$	$2^{-33.2}$	2^{-52}
10	$2^{-23.0}$	$2^{-29.0}$	2^{-52}
11+	$2^{-26.4}$	$\ll 2^{-20}$	$\ll 2^{-46}$
		Sum of EVF	$\mathbf{2^{-26.3}}$

TABLE IV. COMPARISON OF VARIOUS PROTECTION SCHEMES FOR BASELINE 8MB CACHE.

	NoECC	ECC-1	ECC-4	ECC-8	VS-ECC*	FLAIR
Vmin	849mv	684 mv	546mv	487mv	538mv	485mv
Soft error tolerance	Extra	Extra	Extra	Extra	Extra	**Included**

Thus, FVF for this case is $2^{-15.6}$.

3) **Pair has 7 errors:** To get a G- line, one line must have four errors, therefore the other line must have three errors. Failure would occur when the line with 3 errors gets miscorrected (or through a soft error strike) into a line with 4 errors. The probability that 7 errors cause a (4,3) split is $\frac{70}{128}$, and the probability that there will be overlap of the 3 three-error bits in two lines is $\frac{24}{523 \cdot 522 \cdot 521}$. Note that for other splits such as (5,2), DMR will be effective and thus avoid silent error. Thus, the FVF for this case is $2^{-23.4}$.
4) **Pair has 8 errors:** Given that we need at-least one G- line, one line must have either 4 or 6 errors. A (6,2) split will be detected by DMR. Thus, the vulnerable case is only when there is a split of (4,4). The probability that we will get a (4,4) split in 8 errors is $\frac{70}{256}$. The probability that the errors in both lines will overlap is $\frac{24}{523 \cdot 522 \cdot 521 \cdot 520}$. Thus, the FVF for this case is 2^{-33}.
5) **Pair has 9 errors:** For failure there must be at-least one G- line, so one line must have either 4 or 6 errors. A (6,3) split will be detected by DMR. Thus, the vulnerable case is only when there is a split of (4,5) and the line with 5-errors gets changed to 4 bit-error line, either by soft error strike or miscorrection. The probability that we will get a (5,4) split in 9 errors is 0.5. The probability that there will be overlap of 4 errors in both lines is $\frac{120}{523 \cdot 522 \cdot 521 \cdot 520}$. Thus, the FVF for this case is 2^{-33}.
6) **Pair has 10 errors:** For failure there must be at-least one G- line, so one line must have 4 errors and the other 6 errors. The line with 4 error can get alpha particle strike, making it seem like 5 errors which could get miscorrected to the line with 6 errors. The probability that we will get a (5,4) split in 9 errors is 0.5. The probability that there will be overlap of 4 errors in both lines is $\frac{120}{523 \cdot 522 \cdot 521 \cdot 520}$. Thus the FVF for this case is 2^{-29}.
7) **Pair has 11+ errors:** As these many errors in the line is negligible to begin with (2^{-27} or less), we simplify the calculations of these case by simply using a lose upper bound on FVF. For these cases, we need at least five error bits to have overlapping positions. Given that the line (with SECDED) has 523 bits, this probability is $\ll 2^{-20}$, if bit failure probability is < 0.5. So, we will use FVF $\ll 2^{-20}$.

The total failure rate of FLAIR mechanism can be estimated by multiplying FVF with the raw probability that the event will happen. For example, if the probability that the pair has 8 errors is X and the EVF is Y then the contribution to overall vulnerability from 8 error case would be $X \cdot Y$. Table III shows the effective vulnerability of FLAIR. The effective pair failure rate with FLAIR is $2^{-26.3}$ which meets our target for the pair failure probability $2^{-25.7}$. Thus, FLAIR will be able to provide a Vmin of 485mv, similar to ECC-8.

Note that we have made two severely pessimistic assumptions in our analysis. First, miscorrection always results in match with faulty bit of the other line. Second, soft error strike happens exactly in the position of the most vulnerable bit in the line. Given the inherent likelihood of these events is low ($<< 0.2\%$), the Vmin of FLAIR in practice can be expected to be lower than 485mv.

E. Vmin Comparisons

Table IV compares different ECC schemes with VS-ECC.[7] and FLAIR. The Vmin of VS-ECC is limited by ECC-4 for quarter of the cache (as the protected cache has only one-fourth the lines, the Vmin is slightly lower than ECC-4 for baseline cache). FLAIR provides a Vmin similar to ECC-8. An important aspect to consider in these Vmin calculations is that we assumed that for all schemes, except FLAIR, there is alternative mechanism to handle soft-errors, and all the available ECC is used only for tolerating hard errors. Reserving one of the available ECC units to only tolerate soft errors would increase the Vmin of these schemes (significantly). The analysis of FLAIR, on the other hand, already accounts for soft error tolerance. Thus, FLAIR provides low Vmin as well as soft error tolerance, while using only existing SECDED code.

While Table IV compares FLAIR to schemes that avoid non-volatile fault map, we also evaluated FLAIR versus

[7]Table 3 of [2] shows VS-ECC obtains 500mv Vmin with cache line disable, which incorrectly assumes that there are no errors in training phase.

schemes that rely on having non-volatile memory with faulty locations stored. The most advanced (albeit complex) proposal in this class is Archipelago [4]. We found that for failure rate of Figure 2, and yield target of 99.9%, Archipelago tolerates a bit failure rate similar to ECC-8.[8] Thus, FLAIR obtains Vmin similar to Archipelago without the need for non-volatile fault map, while avoiding dual-line read on each cache access, and obviating complex pairing of cache lines.

F. Operation During Post Testing Phase

FLAIR uses replication to provide robustness only during the testing phase. As cache ways get tested, the testing information gets stored in the line, and these ways become available for normal use. Then, a pair of two ways that are storing replicated information for one of the way are freed up for testing. This procedure continues till the entire cache has gone through the testing phase and finally there would be no replicated ways in the cache. After the testing phase is over, the cache operates similar to traditional cache with SECDED and the lines with faults can simply be discarded. Read and write operations in the post testing phase get satisfied by single access at at the normal latency (without any latency overheads).

V. Reclaiming Discarded Lines

Once the testing phase is over, the replication mode of FLAIR is discontinued, the faulty lines identified during the testing phase are disabled, and cache operates only with reliable lines. Figure 10 shows the percentage of lines that have exactly 1 error, or 2 or more errors as the operating voltage is changed form 485mv to 540mv. At target Vmin of 485mv, 30.7% of the lines would have one bit of hard fault, and 9.3% of lines will have two or more errors.

As our cache employs only SECDED we will need to disable lines with 2+ errors. For lines with 1 error we can use SECDED code to correct one bit error. However, the line with single bit fault would then become vulnerable to soft errors. For example, if such a line is struck by a soft error then SECDED would be able to detect the error but not correct it. Therefore, prior approaches [2] would recommend simply disabling lines with even one bit fault. Unfortunately, that would discard 40% of cache lines in normal mode at 485mv operation.

Ideally we would like to have almost all of the cache capacity during normal program operation. We observe that lines with a single bit hard fault can still be used for reliably storing clean lines. If a soft error happens in a weak line (line with 1-bit hard error), then it becomes uncorrectable double bit error and we can detect it with existing SECDED circuitry. If such a lines is restricted to store clean data, we can simply invalidate the line and read the data from memory. We call this concept *Weak Line Reclamation (WLR)*.

To enable WLR, we need information about whether the faulty line has exactly one bit error or more than one bit error. During the testing phase, the lines are identified as such. We convey this information from testing phase to execution phase

[8]The original Archipelago study [4] claimed Vmin of 375mv. This is because they use more optimistic pfail-to-Vmin curve than shown in Figure 2, a yield target of 99%, and 2MB cache. For their assumptions, both ECC-8 and Archipelago are operational till a failure of about 1 in 700 cells (375mv).

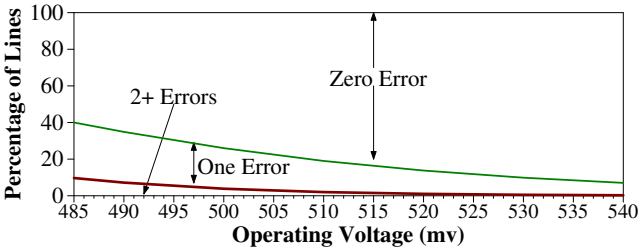

Fig. 10. Percentage of fault-free lines, lines with exactly one error, and lines with 2 or more errors, as operating voltage is varied. At target Vmin of 485mv, there are 9.3% lines with 2+ errors, 30.7% lines with exactly one error, and 60% of the lines have no errors.

in a storage efficient manner. Each line is augmented with a *Faulty Cache Line (FCL)* bit. For all lines that have at least one bit error FCL is set to 1. We note that if FCL=1, that line cannot be used to store dirty data so we can reuse the DirtyBit to convey information about the number of faults in the line. If, FCL=1 and DirtyBit=1 then the line is deemed to have more than 1 error, and is disabled. If FCL=1 and DirtyBit=0, then the line has exactly one error and can be used to store only clean lines. The disable status as conveyed by combination of FCL and DirtyBit is captured in Table V.

TABLE V. Implementing Weak Line Reclamation.

FCL	Dirty	Status
0	0	Fault-free line storing clean data
0	1	Fault-free line storing dirty data
1	0	Faulty Line with 1 hard-error, can store clean data
1	1	Faulty Line with 2+ errors, disabled

If a clean line with FCL=1 becomes dirty then a clean line (with FCL=0) from that set is selected (using the replacement policy of the cache) and swapped with the line that has become dirty. If there are no clean lines in the set, then a victim line (with FCL=0) is identified and the dirty data is written to that location. With Weak Line Reclamation, we can have 90.7% of the cache capacity available during normal program operation (instead of 60% usable cache capacity).

VI. Evaluations and Analysis

A. Experimental Methodology

For our performance studies, we use CMP$im [8], a trace-driven x86 simulator. As a baseline, we model a quad-core out-of-order processor, which is similar to the Intel Core i7 processor. We simulate 32 KB 8-way associative L1 instruction and data caches with 3-cycle latency, 256 KB 8-way associative L2 cache with 8-cycle latency and 8 MB, 16-way associative L3 cache with 20-cycle latency. All caches use a linesize of 64-bytes. We assume L3 is protected with SECDED.

To analyze FLAIR, we extend CMP$im to simulate the following effects: (i) disable cache lines with multi-bit failures, (ii) restrict the use of cache lines with 1-bit failure to clean data. Furthermore, to quantify the performance overhead of reduced cache capacity of FLAIR; we also simulate an ideal, defect-free, low-voltage baseline that has reliable caches without any loss in capacity. We use a slice of 500 million

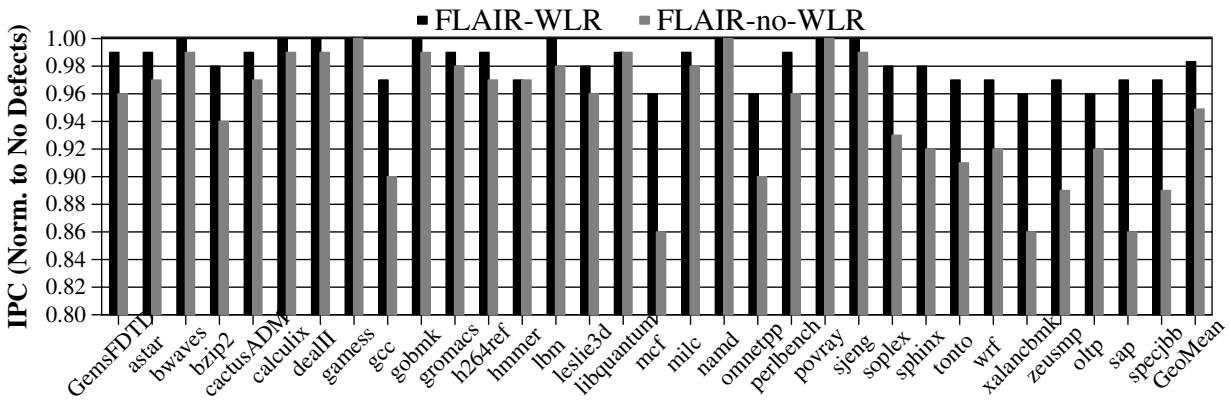

Fig. 11. IPC of FLAIR (with and without Weak Line Reclamation) normalized to an ideal defect-free baseline.

TABLE VI. POWER AND ENERGY COMPARISON.

Design	Vmin (mV)	Clock Frequency (MHz)	Normalized Power	Normalized EPI
No-ECC	848	2000	1.00	1.00
ECC-1	684	1380	0.47	0.69
VS-ECC	538	820	0.20	0.50
ECC-8*	487	580	0.14	0.47
FLAIR	485	575	0.14	0.47

instructions for each SPEC2006 benchmark, obtained after fast-forwarding first 500 million instructions. We run these benchmarks in a rate-mode on the quad-core processor. We also use three commercial benchmarks (oltp, sap, specjbb). We show performance results only for the normal execution phase and not for the testing phase. We report performance in terms of number of committed instructions per cycle (IPC).

B. Performance in Normal Mode

Figure 11 shows the performance of FLAIR (with and without WLR) for all the benchmarks, normalized to defect-free baseline. The bar labeled *GeoMean* show the geometric mean of normalized IPC across all workloads. We only show the performance impact of L3 design changes while keeping the L1 and L2 caches uniform across all the configurations.

In comparison with the defect-free baseline, FLAIR-WLR shows a negligible drop in performance for majority of the benchmarks, with the maximum performance loss limited to 4%. Averaging across all the benchmarks, FLAIR-WLR degrades performance by only 1.5% . The performance loss is negligible because FLAIR-WLR disables only 9% of the cache lines, while restricting 30% of the cache lines to storing clean data. Comparing FLAIR-WLR with FLAIR-no-WLR, we note that FLAIR-WLR provides substantial performance improvements over FLAIR-no-WLR. Weak Line Reclamation improves the performance by 3% on average and brings down the maximum performance degradation caused by our technique during post-testing mode from 14% to 4%, for mcf. For commercial benchmarks, the effectiveness is even more pronounced compared to SPEC benchmarks. WLR improves performance of oltp by 4%, sap by 11%, and specjbb by 8%.

C. Power and Energy Efficiency

Table VI summarizes the achievable Vmin, frequency, power consumption and energy per instruction (EPI) of different configurations during the low voltage mode. In addition to the baseline configuration and FLAIR, we also show results for ECC-1 and VS-ECC. We normalize the power and energy results for different designs to that of the baseline, while showing absolute results for Vmin and frequency. For frequency calculations, we perform circuit simulations to predict the frequency at different voltages. For our power evaluations, we use an industry grade power tool. The power numbers are based on data from a commercial processor on 32nm node.

FLAIR achieves the lowest power and EPI amongst all the configurations. FLAIR reduces power by 86%, 71%, and 30%, compared with baseline, ECC-1, and VS-ECC, respectively, while reducing EPI by 53%, 33%, and 6% compared with baseline, ECC-1, and VS-ECC, respectively. FLAIR achieves better energy efficiency compared to VS-ECC, while obviating the need for complex multi-bit ECC encoding and decoding logic and the storage overhead of ECC-4 for one fourth of the cache.

For power and energy comparisons with ECC-8, we pessimistically assume that circuitry for encoding and decoding ECC-8 incurs zero power and latency. Nonetheless, even with such a pessimistic assumption ,FLAIR has comparable power and EPI as ECC-8, while avoiding the overheads of ECC-8.

D. Hardware Overhead of FLAIR

Implementing FLAIR requires only minor changes to the cache controller. The ternary output of (G,C,D) status of a line is already available from SECDED. We simply need to implement one line-comparator circuit for DMR, which incurs negligible logic. The storage overhead of FLAIR is one bit

per cache line (to indicate faulty cache line). Thus, FLAIR not only avoids the storage overhead of multi-bit ECC but also the complex circuitry required for ECC decoding.

VII. Summary

While several recent proposals have tried to tolerate multi-bit errors in cache lines to enable low-voltage operation, they typically require significant storage overhead and hardware complexity. Our aim is to enable low power operation without requiring significant design changes to cache structure and avoiding the hardware overhead. We rely on runtime testing to identify faulty lines. We propose *FLAIR*, a highly effective dynamic replication scheme to provide robustness during testing phase, and *Weak Line Reclamation* to enable reliable use of lines with 1-bit hard error during the post-testing phase.

In this paper, we set out six requirements for reliable low voltage cache operation and developed a practical, low overhead solution, which meets all these requirements:

1) Our solution enables the cache to operate at a voltage of 485 mV, almost 50mv below the minimum voltage achieved by the previous state-of-the art solution.
2) Our solution incurs negligible logic overhead and the storage overhead of only a single (faulty cache line bit per cache line; substantially lower than previous solutions.
3) Our solution tolerates soft errors in both testing and post-testing phases, without relying on additional storage overhead except for the existing SECDED-code.
4) Our solution provides 91% of the cache capacity during normal program execution, resulting in only 1.3% average performance loss compared to a defect-free baseline.
5) Our solution retains a cache read latency similar to that of a baseline cache with SECDED ECC during normal program execution (post-testing phase).
6) Our solution obviates the need for a non-volatile memory based fault map, and does not rely on software changes for deployment.

Unlike previous low-voltage cache proposals that largely ignored soft-error resilience, FLAIR protects the cache from both hard errors as well as soft errors. FLAIR is a simple and effective solution which enables the use of same chip design in different domains with negligible hardware changes. We believe such practical and effective solutions will be driver for flexible low-power operation modes in future processors.

In this paper, we analyzed FLAIR on a cache that implements SECDED. However, the general idea of FLAIR can also be applied to other cache designs that do not have ECC (No-ECC) or have built-in multi-bit ECC codes [10] .

Acknowledgments

Thanks to Wei Wu for discussions on Multi-bit Error Correction Code. Moinuddin Qureshi is supported by NetApp Faculty Fellowship and Intel Early Career Award.

References

[1] J. Abella et al. Low vccmin fault-tolerant cache with highly predictable performance. In *MICRO-2010*.

[2] A. Alameldeen et al. Energy-efficient cache design using variable-strength error correcting codes. In *ISCA-2011*.

[3] A. Ansari et al. Zerehcache: Armoring cache architectures in high defect density technologies. In *MICRO-2009*.

[4] A. Ansari, S. Feng, S. Gupta, and S. Mahlke. Archipelago: A polymorphic cache design for enabling robust near-threshold operation. In *HPCA-2011*, feb. 2011.

[5] D. Bossen, J. Tendler, and K. Reick. Power4 system design for high reliability. In *IEEE Micro, vol. 22, No. 2, pp. 16-24, Mar. 2002*.

[6] Z. Chisti et al. Improving cache lifetime reliability at ultra-low voltages. In *MICRO-2009*.

[7] A. Garg and P. Dubey. Fuse area reduction based on quantitative yield analysis and effective chip cost. In *Defect and Fault Tolerance in VLSI Systems, 2006. DFT '06. 21st IEEE International Symposium on*, oct. 2006.

[8] A. Jaleel et al. Cmpsim: A pin-based on-the-fly multi-core cache simulator. In *Fourth Annual Workshop on Modeling, Benchmarking and Simulation (MoBS), 2008*.

[9] J. Kulkarni, K. Kim, and K. Roy. A 160 mv robust schmitt trigger based subthreshold sram. In *IEEE Journal of Solid-State Circuits, vol. 42, no. 10, pp. 2303-2313, Oct. 2007*.

[10] M. Manoochehri, M. Annavaram, and M. Dubois. Cppc: correctable parity protected cache. In *ISCA-38*, 2011.

[11] D. Roberts, N. Kim, and T. Mudge. On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. In *Digital System Design Architectures, Methods and Tools, pp. 570-578, Aug. 2007*.

[12] H. M. S. Rusu and B. Cherkauer. Itanium 2 processor 6m: Higher frequency and larger l3 cache. In *IEEE Micro, vol. 24, No. 2, pp. 10-18, Mar. 2004*.

[13] C. Wilkerson et al. Reducing cache power with low cost, multi-bit error-correcting codes. In *ISCA-2010*.

[14] C. Wilkerson et al. Trading off cache capacity for reliability to enable low voltage operation. In *ISCA-2008*.

Appendix A: Memory Bandwidth Consumption in Testing Mode for VS-ECC and FLAIR

We compare our proposal (FLAIR) with the state-of-the-art VS-ECC design in terms of their memory bandwidth consumption during the testing mode. VS-ECC disables 12 cache ways out of 16 ways, resulting in an effective capacity of 25%. In case of FLAIR, the effective cache capacity is higher than 25%, because only 2 out of the 16 ways undergo testing, while the 14 ways are used (with DMR) to provide an operational cache. Table VII compares the average memory bandwidth consumption for VS-ECC and FLAIR. FLAIR reduces read traffic as compared to VS-ECC, by an average of 6%. This reduction is due to larger effective cache capacity. However, the write-through design of FLAIR during the testing mode increases the write traffic to memory. The overall memory traffic consumed by the two approaches differs by only 2% on average. During the testing phase, the overall system performance of VS-ECC and FLAIR are comparable, as FLAIR has fewer read misses compared to VS-ECC.

TABLE VII. Bandwidth Consumption Breakdown in Testing (BW numbers normalized to VS-ECC).

	VS-ECC	FLAIR
Read BW	75.7%	70.7%
Write BW	24.3%	31.0%
Total BW	100.0%	101.7%