An Aging-Aware Early Cache Eviction Strategy to Enhance Static Random-Access Memory Cells’ Lifetime

Mohammad Maghsoudloo[a*]

[a] *Department of Computer Engineering, Faculty of Engineering, Golestan University*

KEYWORDS

cache memory;
bias temperature instability;
static noise margin;
soft errors;
static random-access memory;
aging.

Abstract

This paper presents a comprehensive analysis of the impact of early cache eviction on the aging of cache cells. It highlights that, in addition to the previously identified factors contributing to static noise margin (SNM) degradation in cache cells, the state of cached data plays a critical role in this process. The analysis reveals that the uneven distribution of clean and dirty data blocks across the lines of a cache set can also be a significant factor in SNM degradation. To address this issue, this study proposes an early cache eviction strategy aimed at balancing the distribution of dirty and clean data blocks over cache lines, thereby mitigating SNM degradation. To achieve this, the decision tree of the Pseudo-LRU replacement policy is redesigned to incorporate cache line states and address conflict miss types. Experimental results demonstrate that the enhanced cache improves the hold and read SNM by approximately 10% and 12%, respectively, while incurring negligible cache hit reduction and minimal area and energy overheads.

1. Introduction

As CMOS technology continues to scale, reliability concerns have become increasingly critical in the design of integrated circuits [1]. Among these, transistor aging has emerged as a significant reliability challenge in nanoscale CMOS technology, often leading to permanent hardware faults [2]. Bias Temperature Instability (BTI) is the primary aging mechanism, causing an increase in the absolute value of transistor threshold voltage and a reduction in charge carrier mobility [3]. These

[*] Corresponding author. Tel: +98 911 371 7182, E-mail address: mo.maghsoudloo@gu.ac.ir, Postal address: Golestan University, Shahid Beheshti St., Gorgan, Iran, postal code: 49138-15759

BTI-induced effects can significantly increase propagation delays in combinational circuits [4] and degrade the Static Noise Margin (SNM) of sequential elements such as Static Random-Access Memory (SRAM) [5]. Furthermore, the operating conditions of circuits—including temperature, voltage bias, and current density—can exacerbate BTI-induced SNM degradation, leading to premature vulnerabilities [4, 6].

SNM is a critical metric for evaluating the stability and reliability of SRAM cells, representing the maximum voltage noise that a memory cell can tolerate without a state flip [7, 8, 9]. Reliability concerns in SRAM must be prioritized, as these cells occupy a significant portion of the chip area and are central to on-chip cache memories, which make up over 60% of a modern microprocessor's transistors [6]. This makes SRAM highly susceptible to hard and soft errors, especially as SNM degradation compromises their stability [10]. Notably, BTI-induced SNM degradation is most severe when a constant value is stored in an SRAM cell for extended periods [11, 12, 13, 14]. A typical SRAM cell comprises six MOSFETs configured as two cross-coupled inverters and two access transistors. When storing a continuous "0" or "1," one PMOS and one NMOS transistor are particularly vulnerable to BTI-induced threshold voltage variation [2].

Efforts to address BTI-induced SNM degradation can be categorized into two main areas: aging sensing and aging mitigation. Aging sensing methods often involve embedding sensors in SRAM memory arrays to monitor the system's runtime behavior and detect signs of degradation [15-18]. However, these methods face challenges due to reduced sensor accuracy caused by process variation and environmental changes [19-21], as well as the complexity of implementing circuit-level modifications in SRAM cells [22]. Aging mitigation techniques focus on preventing SRAM cells from storing fixed values for extended periods, aiming to balance the Signal Probability (SP) factor of cache cells to 50% [8, 9, 19, 20, 22].

This paper provides a detailed analysis of the SP factor's probability distribution for cache line bits. To this end, the Multi-facet GEMS (gem5) [23] simulation environment, was utilized. Additionally, the study evaluated the Stanford Parallel Applications for Shared Memory (SPLASH-3) benchmark suite [24]. BTI-induced aging was analyzed using the HSPICE reliability tool [25], while cache energy consumption was estimated using the CACTI 6.5 simulator [26]. An RTL model of the cache control logic was implemented for area analysis, with hardware synthesis performed using the Synopsys Design Compiler [27]. Experimental results revealed that the observed SP factor distribution deviates significantly from the normal distribution, with an average deviation of 88.3%. This paper identifies that, alongside locality principles and Narrow-Width

Values (NWVs), the unbalanced distribution of clean and dirty data blocks across the lines of a cache set also contributes to this deviation.

Based on these findings, this paper proposes a BTI-aware cache eviction strategy designed to balance the placement of data blocks associated with different states across cache lines. The proposed method modifies the cache control logic to balance the cumulative intervals between consecutive writes on various cache lines in a set. The strategy leverages two key factors—the state of cache lines (clean or dirty) and the type of address conflict miss. The enhanced cache achieves improvements of approximately 10% and 12% in Hold-SNM and Read-SNM, respectively, for the 4-way, 8-way and 16-way caches, with minimal performance penalties: about 1% reduction in cache hit ratio, about 3% increase in write-backs, about 4% dynamic energy consumption overhead, and area overheads of about 1% and 2% for cell and net counts, respectively.

The ideas, implementations, and results presented in this paper build upon and extend the foundational analyses from our initial work published in [28]. The enhancements and improvements outlined below are aimed at extending the scope and depth of the original study. While the initial work in [28] was designed specifically for 4-way set-associative caches, the extended version introduces a generalizable early cache eviction strategy that can be effectively applied to caches with varying degrees of associativity (e.g., 8-way and 16-way caches). It also includes deeper SP factor analysis, scalability tests for 8-way and 16-way caches, and detailed pseudocode for the new eviction strategy, ensuring adaptability to workload types.

The remainder of this paper is structured as follows: Sections 2 and 3 discuss the background and experimental setup. Sections 4 and 5 describe the motivation and structure of the proposed BTI-aware eviction strategy. Finally, Sections 6 and 7 summarize the results and present conclusions.

2. Background

Numerous studies have addressed the challenges posed by BTI-induced errors in SRAM cells, which can be broadly divided into two categories: aging sensing and aging mitigation strategies [29]. Aging sensing approaches aim to detect critical signs of BTI-induced degradation in SRAM cells to prevent catastrophic system failures [30, 31]. These techniques often employ sensors that monitor the runtime behavior of memory systems at various levels of abstraction. At the circuit level, they leverage electrical parameters such as subthreshold leakage current, threshold voltage, drain current, transconductance, signal rise and fall rates, margin delays, reference voltages, and bit-line currents as indicators of aging [14, 15]. Additionally, architectural-level techniques, such as Error Detecting/Correcting Codes (EDC/ECC), are used to identify soft and hard errors in SRAM cells by validating stored data integrity [31]. However, despite their effectiveness, aging sensing techniques face notable limitations, including substantial area and power overheads and the complexity of integrating circuit-level modifications into SRAM designs [16]. In contrast to sensing, aging mitigation strategies focus on proactively addressing conditions that accelerate BTI-induced degradation in SRAM cells [16]. The primary goal of these methods is to prevent SRAM cells from storing fixed values ("0" or "1") for extended durations, as this exacerbates BTI stress [16, 19]. Structural enhancements, such as enabling drowsy mode for idle cache regions, power-gating specific cells, or even processor overclocking, and SP factor balancing techniques, such as bit flipping, dynamic cache indexing, and NWV-aware cache management, have been proposed to extend cache relaxation times and reduce stress on SRAM cells [16, 19, 20, 22, 33, 34, 35]. Another architectural-level technique exists that specifically targets a 4-way set-associative cache architecture [28]. The proposed enhancement to the replacement policy, including modifications to the cache management structure and the integration of coherency-state-aware victim selection, is structurally bound to the binary-tree format of 4-way caches, and the techniques were not generalized for 8-way or 16-way cache organizations [28]. Generally, unlike aging sensing methods, mitigation techniques often present a more practical solution by significantly reducing the area and power overheads associated with sensing methods [16].

3. Experimental setup

For evaluating both the existing and newly introduced design decisions, an extended version of Multi-facet GEMS (gem5) [23] was utilized. The specifications of the baseline system are summarized in Table 1. In this study, the entire set of applications and kernels from the Stanford Parallel Applications for Shared Memory (SPLASH-3) benchmark suite was analyzed within the simulation environment. The SPLASH-3 suite is widely regarded as a comprehensive collection of parallel programs designed for cache-coherent shared address space architectures and has been extensively used in modern research [24]. Each program was simulated for 900 million instructions, with the initial 100 million instructions fast-forwarded to ensure the evaluation focused on steady-state execution.

4. Motivation

In an ideal scenario for preventing SNM degradation, the value of a bit stored in an SRAM cell should be an independent random variable with equal probabilities of being “0” or “1.” In this case, the probability distribution of the SP factor for all bits stored in the cache over time would follow a normal distribution with a mean of 50%. A detailed analysis was conducted to investigate the probability distribution of the SP factor for the simulated cache bits. Fig. 1 illustrates the distribution of the SP factor for 524,288 bits (64 KB) of the cache, categorized into clean and dirty data. As shown in Fig. 1, the Baseline does not exhibit a perfect normal distribution with a mean of 0.5. More than 68% of the SP factors are below 0.5, reflecting a higher likelihood for the cache bits to hold constant “0” values during their lifetime. This deviation is attributed to two key reasons: locality principles and NWVs [16]. This behavior is exacerbated in dirty cache lines, where sequences of write operations lead to a greater frequency of bit changes compared to clean data. Consequently, dirty cache lines exhibit an SP factor distribution closer to the normal distribution, with an average error of 28.9%, compared to 91.2% for clean cache lines. This observation underlines the potential for fairer distribution of dirty data blocks across cache lines.

5. BTI-aware early cache eviction strategy

Today, most primary caches employ a Pseudo-Least Recently Used (PLRU) algorithms to avoid the disadvantages of a complex hardware design of base Least Recently Used (LRU) [36, 37]. In order to determine how close the decisions made by the PLRU to LRU, we extended the experiments to assess the chances of each cache block in a set to be selected as the replacement candidate by the PLRU and LRU considering their access history. Fig. 2 shows the distribution of block selection as the replacement victim in an 8-way cache. The blocks are ranked based on their access history, and the block holding the first/last rank is the least/most recently accessed one. The experimental results reveal that the PLRU algorithm selects blocks of 8-way cache from the first, second, third, and fourth ranks with an average frequency of 21.5%, 19.4%, 15.1%, and 13.0%, respectively. The selection of blocks with ranks two, three, and four deviates from the least recently accessed block (rank one) by approximately 2.1%, 6.4%, and 8.5%, respectively, highlighting the non-uniformity of the PLRU replacement policy. Based on the experimental results, the block holding the second rank has almost the same chance as the least-recently-used block to be victimized by the PLRU. Generally, the results show that the data block replaced by the PLRU is not the least-recently-used one.

In order to investigate how much different replacement decisions can affect the cache hit ratio, an experiment was conducted so that different blocks were forcibly selected as the replacement candidate concerning their access histories. Fig. 3 shows the hit ratio of the simulated cache after executing a specified number of instructions in the case of victimizing different blocks holding different ranks in a cache set for an 8-way cache. The results are the average cache hit ratio for all the workloads. As demonstrated by Fig. 3, for the 8-way cache set, replacing the blocks ranked second, third and fourth results in approximately less than 10% reduction in the cache hit ratio. In a set-associative cache, the data blocks ranked first and second are located in the same branch of the replacement tree structure, either left or right. As a result, selecting any leaf from the left or right branch yields similar outcomes, depending on the configuration of the tree.

Taking advantage of relative freedom to victimize the least recently used cache items, the structure of the decision tree has been modified to balance the distribution of dirty cache blocks over different ways of a cache set. In other words, the main concentration of the enhanced eviction strategy is on the fact that giving the cache lines approximately the same chances of storing dirty/clean data could reduce the time cache lines store fixed values. The pseudo-code for the enhanced tree-based PLRU replacement policy for a n-way associative cache is shown in Fig. 4. Table 2 describe symbols and functions used by Algorithm 1. The cache set is structured as a binary tree, with $log_2(n)$ internal decision bits (B) guiding traversal to find the least recently used portion of the set. Each decision bit indicates whether the traversal should continue into the left or right half of the current candidate pool. This descent continues recursively using *LeftHalf()* and *RightHalf()* helper functions. This process continues until only two candidates remain, referred to as *C1_candidates*, which are then evaluated for replacement. The final decision considers the dirty/clean status of the two candidates to balance cache performance and reliability. Clean lines are prioritized for eviction during write misses to minimize unnecessary write-backs, while dirty lines are preferred during read misses to preserve clean data for future modifications. This scalable algorithm maintains the core principles of PLRU replacement while introducing enhanced logic to prevent SNM degradation caused by an imbalance in clean and dirty cache blocks.

An essential feature of the enhanced PLRU algorithm is its adaptability to dynamic workloads. By leveraging the *ConflictType* parameter, the algorithm differentiates between read and write misses, enabling it to apply enhanced replacement strategies. This context-awareness ensures that the replacement decisions align with the current access patterns of the workload. In write-intensive scenarios, the prioritization of clean blocks for eviction helps mitigate the overhead of excessive write-back operations. On the other hand, during read-heavy workloads, retaining clean blocks improves future cache efficiency by reducing the need for reloads from main memory.

Moreover, the hierarchical decision-making process of the binary tree inherently supports parallelism. Each level of the tree operates independently, allowing decision bits for different branches to be evaluated concurrently. This feature can be particularly advantageous in hardware implementations, where parallel processing of decision bits can significantly reduce the latency of replacement decisions. Furthermore, the compact representation of decision bits in the *B* array minimizes memory overhead, making the algorithm lightweight. The logic depth of tree-based eviction strategies grows with the associativity. In hardware terms, for a general *n*-way cache (where *n* is a power of 2), the proposed mechanism generalizes this procedure. It uses $log_2(n)$ decision bits to traverse a binary tree of *n* lines, halving the candidate pool at each level until only two remain (the final two candidates). The algorithm then applies the same final-step check on those two lines' dirty bits and the *ConflictType*. A tree-based PLRU policy requires storing ~$n–1$ bits per set to represent the binary decision tree. Our strategy uses the same bits, which is $O(n)$ growth in storage. It leverages the existing PLRU tree bits and dirty flags, plus the runtime *ConflictType* signal (read vs. write miss) which is an input, not stored in the cache. The PLRU tree logic itself scales with *n* roughly in proportion to the number of internal nodes (also $O(n)$ gates in total), but this is a very small structure in absolute terms (e.g. a 16-way set uses 15 1-bit nodes). The dirty-bit comparator is a simple XNOR/AND gate checking two 1-bit values, and the logic to choose victim based on *ConflictType* is a tiny multiplexer. Therefore, the incremental area overhead of the proposed strategy over PLRU is negligible (on the order of a few gates, independent of *n*). In summary, area scales ~$O(n)$ (dominated by the need to store more bits as ways increase), and enhancements do not change that asymptotic growth. Both PLRU and enhanced strategy have a time complexity on the order of $O(log\ n)$ for choosing a victim. The critical path in hardware will pass through roughly $log_2(n)$ bit-checks. Our strategy adds at most one additional comparison and a final 2-way select based on either a PLRU bit or the *ConflictType*. These additions are constant-time operations and thus do not change the overall $O(log\ n)$ scaling.

Fig. 5 illustrates the transition probabilities between different states of a cache block (clean and dirty) during workload execution. These probabilities are derived by monitoring state changes in the cache coherence directory and analyzing the transitions among various coherency states associated with cache lines. The results reveal that, on average, a cache block loaded into a cache line due to an address conflict read miss remains clean until eviction in over 72% of cases. Similarly, for an address conflict write miss, a cache line is updated with a data block fetched from lower-level memory and remains dirty until a write-back occurs, with this behavior observed in more than 80% of cases on average.

To address the above findings, the proposed mechanism aims to balance the distribution of clean and dirty data blocks across the lines of a cache set. This is achieved by selecting the victim cache line based on the dirty bits of the blocks ranked first and second in the set, as well as the type of address conflict miss prompting the replacement. The process involves two scenarios:

- **Scenario 1:** For an address conflict write miss, a cache line containing a clean data block is chosen for eviction to make room for a new data block with a high likelihood of being modified.
- **Scenario 2:** For an address conflict read miss, a cache line containing a dirty data block is selected for eviction to accommodate a new data block with a lower probability of modification.

The action proposed for the second case aims to increase the likelihood of storing incoming data blocks with high modification potential in cache lines that currently contain clean data. Experimental results further reveal that the majority of changes to dirty cache blocks occur within the first 50% of their lifetime in the cache. Fig. 6 illustrates the average percentages of writes performed on modified cache blocks during different intervals of their lifetime. On average, approximately 67% and 75% of updates to dirty cache blocks are made within the first half of their lifetime for Kernels and Applications workloads, respectively. Additionally, dirty blocks tend to move from the third or fourth ranks to the first or second ranks in the replacement hierarchy after 82.3% of their lifetime. Beyond this point, the number of write accesses to these modified data blocks decreases, on average, to less than 3%. This indicates that most dirty data blocks ranked first and second remain unchanged until replacement. Therefore, the early eviction of such dirty blocks could help mitigate SNM degradation.

To investigate the impact of increasing associativity on design parameters, we also repeated the experiments for caches with 8-way and 16-way configurations. Fig. 7 compares the cache hit ratio reduction in the case of victimizing blocks holding second to fourth ranks in a cache with different levels of associativity. In this experiment, the second/third/fourth-ranked blocks were forcibly selected as the replacement candidate concerning their access histories. The results, presented in Fig. 7, are the average cache hit ratio reduction compared to the case in which the first-ranked block is evicted. Regarding Fig. 7, the cache hit ratio is less affected by the eviction of the second/third/fourth-ranked blocks as the level of the cache associativity increases. Using a 4-way cache, victimizing the blocks holding the second, third, and fourth ranks (instead of the least-recently-used one) leads to about 1%, 23%, and 31% reduction in the cache hit ratio, respectively. These values are, on average, about 0.2%, 15%, and 14% for an 8-way cache, and 0.2%, 9%, and 6% for a 16-way cache. By increasing the level of cache associativity, more choices will be revealed as the alternative for the least-recently-used data block to be replaced with the aim of SNM degradation prevention. The conditions intended in the structure of the proposed decision tree can be similarly extended to manage the chance of storing the data blocks with the higher potential of modification among cache lines of a set with higher levels of associativity. In this case, the lower-ranked cache lines can also be considered along with the first and second-ranked ones, while adjusting the distribution of clean/dirty data blocks among cache lines is taken into account.

6. Result analysis and discussion

6.1. Aging mitigation

Metal Oxide Semiconductor Reliability Analysis (MOSRA) [25] has been used for BTI-induced aging analysis of the 32nm SRAM cells before and after applying modification to the controller circuits of the simulated cache. The amount of threshold voltage variation (ΔV_{th}) of the SRAM cell transistors can be estimated at different temperatures using the built-in aging model of the MOSRA. As a first step, the probability distribution of the SP factor for cache bits is investigated after using the enhanced replacement policy, and the average percentage error between the observed and normal distributions is compared with the results of the baseline cache. Fig. 8 (a) shows the distributions of the SP factor of cache bits in the case of using Baseline and Enhanced architectures. The Fair distribution curve has been included in the graph to give a clear comparison in terms of the average percentage error between the obtained results and the completely fair distribution. Regarding Fig. 8 (a), the average percentage error between the results of Enhanced cache and normal distributions is about 71%, implying about 20% improvement compared to the average percentage error between the Baseline and Normal distributions. Moreover, the average percentage error between the results of the Enhanced cache and Fair distribution is about 16% which indicates the effectiveness of the proposed replacement policy in reaching a nearly fair distribution of the data blocks with a high potential for modification among cache lines.

As a second step, the amount of BTI-induced threshold voltage shift (ΔV_T) is studied according to the built-in aging model of MOSRA. Fig. 8 (b) illustrates the average amount of ΔV_T for the whole SRAM cell transistors of a cache line with 32-nm technology for 200 months. The results have been obtained based on the duty cycles of the SRAM cell transistors extracted from gem5 after running all workloads on the simulated architectures, including the cache design decisions. Regarding Fig. 8 (b), V_T shifts of SRAM cell transistors of the Baseline and Enhanced cache can be increased over time up to 101.33 mV and 87.12 mV, leading to about 14.0% threshold voltage degradation reduction by the proposed architecture. Moreover, the average percentage error between the results of the Enhanced and Fair caches is about 1% compared to the about 17% percentage error between the results of the Baseline and Fair caches.

The stability of SRAM cells depends on the SNM, and asymmetric ΔV_T of SRAM cell transistors can degrade the SNM of the cell by unbalancing the VTC. SNM of an SRAM cell can be measured in terms of two sub-stability metrics: Hold- and Read-SNM (H-SNM and R-SNM). The H-SNM and R-SNM degradation of the SRAM cells used in the Baseline and Enhanced cache is compared in Fig. 9. The H-SNM and R-SNM of the fresh SRAM cells in 32-nm technology are about 108.7 *mV* and 102.2 *mV*. The H-SNM and R-SNM are reduced to about 77.7 *mV* and 74.7 *mV* after 180 months of cache operation, leading to 29% and 27% SNM degradation, respectively. However, the SNM of the SRAM cells degrades more moderately under the management of the Enhanced cache. In this case, the H-SNM and R-SNM are degraded by about 20% and 19% after 180 months, translating into about 10% and 12% improvement in H-SNM and R-SNM degradation.

For the experiments conducted on the 8-way cache, the results indicate a more significant improvement in the reduction of H-SNM and R-SNM in the Enhanced cache compared to the Baseline cache. As Fig. 10 shows, In the Baseline cache, the degradation of R-SNM and H-SNM after 180 months of operation is approximately 26.9% and 28.5%, respectively. However, in the Enhanced cache, these values reduce to approximately 19.2% and 19.6%, respectively. As observed, the baseline exhibits a pronounced ΔV_T shift in specific SRAM cells due to imbalanced switching activity, particularly where dirty lines persist for extended periods. In contrast, the proposed early cache eviction strategy (ECE), by managing dirty and clean block distribution adaptively based on access type, leads to a more uniform toggle probability across all cache bits. This translates into reduced ΔV_T drift and improved aging balance.

To comprehensively evaluate the aging mitigation effectiveness of the ECE, we analyze its impact on ΔV_T and SNM degradation under various Process–Voltage–Temperature (PVT) conditions over an extended operational period. The results summarized in Table 3 are derived from BTI-aware simulations using MOSRA [25] over 180 months, reflecting realistic lifetime behavior of SRAM-based caches. Under nominal conditions (25°C, 1.0V), which reflect typical operating environments, the baseline cache shows significant degradation: ΔV_T reaches 101.3 *mV*, while SNM degrades by 29% (hold) and 27% (read). In contrast, the ECE limits ΔV_T to 87.1 *mV* and reduces SNM degradation to 20% and 19%, respectively. These improvements stem from ECE's signal-balancing policy, which adaptively evicts clean or dirty lines based on miss type.

Under stress-prone conditions (85°C, 0.9V), where aging effects are amplified due to increased BTI sensitivity and slower charge recovery, degradation is naturally more severe. The baseline configuration shows a ΔV_T of 115.5 *mV* and SNM degradation levels of 33% (hold) and 31% (read). Despite this challenging environment, ECE continues to provide substantial mitigation, lowering ΔV_T to 98.7 *mV* and improving SNM margins by 10% points in both hold and read modes. This confirms that ECE's architectural-level balancing remains effective even under accelerated degradation scenarios. Conversely, in conservative corners (–10°C, 1.2V), aging slows due to reduced BTI stress and better carrier mobility. As expected, the baseline system performs better here, with ΔV_T at 84.9 *mV*, and SNM degradations of –24% (hold) and –22% (read). Still, ECE further enhances stability, achieving ΔV_T of 69.3 *mV*, and lowering SNM degradation to –16% and –15%, respectively.

6.2. Performance overhead

Two metrics are considered to compare the effects of design decisions on the cache performance: cache hit/miss ratio and the number of write-backs. Fig. 11 demonstrates the effects of the Enhanced cache on the aforementioned metrics. Regarding Fig. 11, the ECE increases the cache miss ratio in comparison with the Baseline cache on average by about 1.25%, 0.81%, and 0.67% in the case of using 4-way, 8-way, and 16-way set-associative caches. The experimental results demonstrate that the early eviction of the selected data blocks does not drastically degrade the hit ratio since the proposed technique invalidates most replacement candidates after being idle. Increased associativity decreases the number of conflict misses and subsequently can generally improve the cache miss rate. Thus, the hit rate degradation, observed due to the proposed early eviction, is moderated by increasing the degree of associativity. Moreover, the proposed technique increases the number of write-backs compared to the Baseline cache on average by about 3.66%, 3.17%, and 2.34% in 4-way, 8-way, and 16-way caches. By increasing the level of cache associativity, the distance between the practical results of selecting two consecutive data blocks in replacement ranking is reduced. Therefore, evicting the second-ranked replacement candidate, as an alternative for the least-recently-used data block, imposes less performance overhead while cache associativity increases from 4-way to 16-way.

6.3. Energy and area overhead

The activity factor of the SRAM cells, which is mainly concentrated to prevent SNM degradation by the aging-aware cache management techniques, can directly affect the dynamic energy consumption of the cache. The CACTI 6.5 simulator [26] is used to estimate the dynamic energy consumption of Baseline and Enhanced caches using the 32nm technology node. The dynamic energy consumption increases as a result of the increased bit-switching activities of the Enhanced cache's cells. Balancing the probability distribution of the SP factor of cache lines' bits by the proposed selective early eviction of unused clean/dirty data blocks leads to the increased cache miss rate and the activity factor. The Enhanced 4-way, 8-way and 16way caches increase the dynamic cache energy consumption compared to the Baseline cache on average by about 4.07%, 4.39% and 4.74%, respectively.

Moreover, the additional circuity in the cache controller, inserted to implement the enhanced replacement policy, raises the need for area consumption analysis. Due to the limitations of the CACTI tool for area modelling of the cache-controlling circuits, designated to enforce the replacement policy and coherence protocol, we take advantage of RTL-level modelling and synthesis for the area evaluation. To this intent, an RTL model of the cache control logic is implemented. Further, the Synopsys Design Compiler [27] is used to perform hardware synthesis. We used the Nangate 45nm standard-cell library which is based on the open FreePDK45 PDK [38]. Fig. 12 shows the results of the aforementioned hardware synthesis of the HDL models of the cache design decisions. The effects of the Enhanced cache on the area of the different parts of the data cache and its control logic, including Replacement Policy Enforcement (RPE) circuits, Coherence Protocol Enforcement (CPE) circuits, and other cache control logic are reported. The 4-way, 8-way and 16-way caches are completely modelled with respect to the configuration mentioned in Table 1. Summarizing the results of area analysis, the numbers of cells and nets of the total control logic of the Enhanced cache grow by about 4.78% and 6.51% in the 4-way cache implementation, by approximately 4.81% and 6.52%, for the 8-way cache implementation, and by 5.42% and 7.56%, for the 16-way cache implementation. Concerning total cache circuits (memory banks and total control logic), using the Enhanced cache leads to about 1.08% and 1.56% area overhead in terms of the numbers of cells and nets, respectively, for the 4-way cache implementation. The 8-way cache implementation results in about 1.1% and 1.58% area overhead in these terms, and the 16-way cache leads to about 1.2% and 1.7% area overhead in the same terms.

6.4. Comparison with the state-of-the-art

In this section, we present a comprehensive comparison between the proposed method (ECE) and four representative aging mitigation techniques that, like ECE, aim to reduce SRAM aging by regulating signal probability, include: Intra-word Bit Swapping (IBS) [34], Cell Flipping with

Distributed Refresh (CFDR) [35], Double Capacity Cache (DCC) [19], and Cache Size Management (CSM) [22].

Fig. 13 presents the signal probability values for each individual bit position (0 to 63) across a 64-bit cache line. In the base configuration, the signal probabilities show significant imbalance, with extreme values ranging from as low as 1% to as high as 65%. IBS improves signal balance for some bits through bit permutation, it fails to normalize the extremes of the word-line and introduces switching overhead. CFDR performs better in global balancing but is constrained by refresh timing and workload sensitivity. DCC does not explicitly target signal balancing and, as a result, leaves many bits with biased activity patterns, particularly when narrow-width data is not dominant. CSM, while indirectly reducing stress via power gating, lacks per-bit resolution and exhibits uneven effectiveness. In contrast, ECE maintains bit-level signal probabilities tightly within the 48–52% range across all bit positions.

Fig. 14 illustrates the normalized dynamic energy consumption of cache memory cells across various benchmark applications, considering different cache associativities. In the 4-way configuration, the average normalized energy for ECE is 0.041, outperforming IBS (0.232), CFDR (0.085), DCC (0.172), and CSM (0.102). As associativity increases, energy consumption slightly rises for all methods, however, ECE maintains its advantage, with averages of 0.044 and 0.047 for 8-way and 16-way caches, respectively. This efficiency stems from ECE’s fine-grained control over signal probability and switching activity, which directly reduces unnecessary transitions in SRAM cells. In contrast, IBS and DCC, while effective in aging mitigation, lead to increased switching overhead and energy waste. CFDR perform moderately well, but its global refresh and power gating strategies lack the precision targeting of ECE. Also, CSM performs well in some scenarios but lacks workload consistency. These findings confirm that ECE offers a favorable reliability–energy trade-off, as it achieves substantial aging mitigation with only a marginal increase in dynamic energy, making it a compelling solution even for moderately energy-constrained systems.

The impact of each method on cache miss ratio is detailed in Fig. 15, which shows normalized miss rates across standard workloads. ECE maintains miss ratios nearly identical to the baseline cache, indicating that its aging mitigation logic does not interfere with standard access or replacement behavior. IBS and CFDR introduce minor increases in miss rates, likely due to their additional access redirection logic. DCC exhibits larger deviations, especially in higher-associativity caches, likely due to the complexity of packing and tag matching. Fig. 16 compares the normalized write-back rate to L2 cache. ECE shows a moderate increase in write-back activity compared to other methods, as it proactively evicts blocks that contribute disproportionately to aging.

Fig. 17 illustrates the area overhead associated with both the control logic and the overall cache system, relative to a baseline design. Techniques such as IBS and DCC require additional circuitry for bit manipulation, multi-mode handling, or access redirection, leading to noticeably larger control and overall area footprints. CFDR is particularly hardware-intensive due to its distributed refresh logic and the need to track and coordinate refresh phases across cache blocks. While CSM also maintains a low area overhead similar to ECE, it does so at the expense of precise control over aging and is less effective in maintaining performance consistency. In contrast, ECE achieves its goals through a streamlined control structure that integrates seamlessly with existing cache architecture, avoiding invasive modifications to data paths or memory structures.

7. Conclusion

This paper introduces an enhanced cache eviction strategy designed to extend the cache's lifespan under aging stress. The proposed enhancement specifically targets mitigating hard errors caused by Static Noise Margin (SNM) degradation. Experimental results indicate that the uneven distribution of data blocks with varying states across the lines of a cache set intensifies the SNM degradation in SRAM cells. To address this, the paper presents a BTI-aware cache replacement policy that ensures a balanced placement of data blocks with different states across cache lines. This is achieved by revisiting the decision tree of the pseudo-LRU algorithm, incorporating considerations for the dirty bits of cache lines and the type of address conflict miss. The enhanced cache demonstrates improvements in both hold and read static noise margin degradation by approximately 10% and 12%, respectively, while incurring less than a 1.0% reduction in cache hit ratio.

References

[1] Agarwal, A., Paul, B., Mahmoodi, H., Datta, A., and Roy, K. "A process-tolerant cache architecture for improved yield in nanoscale technologies", IEEE Trans. Very Large Scale Integr. Syst., 13(1), pp. 27–38 (2005). doi: 10.1109/TVLSI.2004.840407

[2] Bansal, A., Rao, R., Kim, J., Zafar, S., Stathis, J., and Chuang, C. "Impacts of nbti and pbti on sram static/dynamic noise margins and cell failure probability", Microelectron. Reliab., 49(6), pp. 642–649 (2009). doi: 10.1016/j.microrel.2009.03.016

[3] Bagatin, M., Gerardin, S., Paccagnella, A., and Faccio, F. "Impact of nbti aging on the single-

event upset of sram cells", IEEE Trans. Nucl. Sci., 57(6), pp. 3245–3250 (2010). doi: 10.1109/TNS.2010.2084100

[4] Jafari, A., Raji, M., and Ghavami, B. "BTI-Aware Timing Reliability Improvement of Pulsed Flip-Flops in Nano-Scale CMOS Technology", IEEE Trans. Device Mater. Reliab., 21(3), pp. 379–388 (2021). doi: 10.1109/TDMR.2021.3102521

[5] Firouzi, F., Kiamehr, S., and Tahoori, M. "Statistical analysis of BTI in the presence of process-induced voltage and temperature variations", 18th Asia South Pacific Des. Autom. Conf., pp. 594–600 (2013). doi: 10.1109/ASPDAC.2013.6509663

[6] Lanzieri, L., Kietzmann, P., Fey, G., Schlarb, H., and Schmidt, T. "Ageing Analysis of Embedded SRAM on a Large-Scale Testbed Using Machine Learning", arXiv preprint arXiv:2307.06693 (2023). [Online]. doi: 10.48550/arXiv.2307.06693

[7] Listl, A., Mueller-Gritschneder, D., and Schlichtmann, U. "Application-aware aging analysis and mitigation for SRAM design-for-reliability", Microelectron. Reliab., 134, p. 114548 (2022). doi: 10.1016/j.microrel.2022.114548

[8] Dounavi, H.-M., Sfikas, Y., and Tsiatouhas, Y. "Aging prediction and tolerance for the SRAM memory cell and sense amplifier", J. Electron. Test., 37(4), pp. 1–14 (2021). doi: 10.1007/s10836-021-05932-6

[9] Dounavi, H.-M. and Tsiatouhas, Y. "An aging monitoring scheme for SRAM decoders", Microprocess. Microsyst., 94, p. 104623 (2022). doi: 10.1016/j.vlsi.2022.09.009

[10] Ding, J. and Asenov, A. "The analysis of static random access memory stability under the influence of statistical variability and bias temperature instability-induced ageing", Semicond. Sci. Technol., 36(2), p. 025008 (2020). doi: 10.1088/1361-6641/abcbd6

[11] Gul, W., Shams, M., and Al-Khalili, D. "SRAM cell design challenges in modern deep sub-micron technologies: An overview", Micromachines, 13(8), p. 1332 (2022). doi: 10.3390/mi13081332

[12] Lin, Y., Li, M., and Gupta, S. "Predictive testing for aging in SRAMs and mitigation", Proc. IEEE Int. Test Conf., pp. 293–302 (2024). doi: 10.1109/ITC51657.2024.00050

[13] Hovanes, J. "Aging-induced long-term data remanence in SRAM cells", M.S. thesis, Auburn Univ., Auburn, AL, USA (2023). [Online]. Available: https://etd.auburn.edu/handle/10415/8673

[14] Wang, M., Hou, Z., Wang, C., Yan, Z., Li, S., Du, A., Cai, W., Li, J., Zhang, H., Cao, K., Shi, K., Wang, B., Zhao, Y., Xiang, Q., Wang Z., and Zhao, W., "NAND-like SOT-MRAM-

based approximate storage for error-tolerant applications", arXiv preprint arXiv:2404.05528 (2024). [Online]. Available: 10.48550/arXiv.2404.05528

[15] Inci, A., Isgenc, M., and Marculescu, D. "DeepNVM++: Cross-layer modeling and optimization framework of non-volatile memories for deep learning", arXiv preprint arXiv:2012.04559 (2020). [Online]. doi: 10.48550/arXiv.2012.04559

[16] Hu, J., Wang, S., and Ziavras, S. "In-Register Duplication: Exploiting Narrow-Width Value for Improving Register File Reliability", Int. Conf. Dependable Syst. Netw., pp. 281–290 (2006). doi: 10.1109/DSN.2006.43

[17] Hovanes, J., Zhong, Y., and Guin, U. "Beware of Discarding Used SRAMs: Information is Stored Permanently", arXiv preprint arXiv:2208.02883 (2022). [Online]. doi: 10.48550/arXiv.2208.02883

[18] Shaik, J., Guo, X., Singhal, S., "Impact of Aging and Process Variability on SRAM-Based In-Memory Computing Architectures", IEEE Tran. on Cir. and Syst., 71, p. 2696-2708 (2024). doi: 10.1109/TCSI.2024.3381935

[19] Imani, M., Patil, S., and Rosing, T. "DCC: Double capacity Cache architecture for narrow-width values", Int. Great Lakes Symp. VLSI, pp. 113–116 (2016). doi: 10.1145/2902961.2902990

[20] Flautner, K., Kim, N., Martin, S., Blaauw, D., and Mudge, T. "Drowsy caches: simple techniques for reducing leakage power", Proc. 29th Annu. Int. Symp. Comput. Archit., pp. 148–157 (2002). doi: 10.1109/ISCA.2002.1003572

[21] Lanzieri, L., Martino, G., Fey, G., Schlarb, H., Schmidt, T., and Wählisch, M.,. "A Review of Techniques for Ageing Detection and Monitoring on Embedded Systems," ACM Comput. Surv. 57(1), pp. 1-34 (2025). doi: 10.1145/3695247

[22] Rohbani, N., Ebrahimi, M., Miremadi, S., and Tahoori, M. "Bias Temperature Instability Mitigation via Adaptive Cache Size Management", IEEE Trans. Very Large Scale Integr. Syst., 25(3), pp. 1012–1022 (2017). doi: 10.1109/TVLSI.2016.2606579

[23] Binkert, N., Beckmann, B., Black, G., Reinhardt, S., Saidi, A., Basu, A., Hestness, J., Hower, D., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M., and Wood, D. "The gem5 simulator", ACM SIGARCH Comput. Archit. News, 39(2), pp. 1–7 (2011). doi: 10.1145/2024716.2024718

[24] Sakalis, C., Leonardsson, C., Kaxiras, S., and Ros, A. "Splash-3: A properly synchronized benchmark suite for contemporary research", IEEE Int. Symp. Performance Anal. Syst.

Softw., pp. 101–111 (2016). doi: 10.1109/ISPASS.2016.7482078

[25] Tudor, B., Wang, J., Sun, C., Chen, Z., Liao, Z., Tan, R., Liu, W., and Lee, F. "MOSRA: An efficient and versatile MOS aging modeling and reliability analysis solution for 45nm and below", IEEE Int. Conf. Solid-State Integr. Circuit Technol., pp. 1645–1647 (2010). doi: 10.1109/ICSICT.2010.5667399

[26] Muralimanohar, N., Balasubramonian, R., and Jouppi, N., "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," IEEE/ACM Int. Sym. on Microarchitecture, Chicago, IL, USA, pp. 3-14, (2007), doi: 10.1109/MICRO.2007.33

[27] SYNOPSYS. "Design Compiler", [Online]. Available: https://www.synopsys.com

[28] Maghsoudloo, M. "On the Prevention of Coherence-Induced Static Noise Margin Degradation of SRAM Cells", 5th CPSSI Int. Symp. Cyber-Phys. Syst., Tehran, Iran, pp. 1–8 (2024). doi: 10.1109/CPSAT64082.2024.10745400

[29] Saun, S. and Kumar, H. "Design and performance analysis of 6T SRAM cell on different CMOS technologies with stability characterization", IOP Conf. Ser. Mater. Sci. Eng., 561 (2019). doi: 10.1088/1757-899X/561/1/012093

[30] Karimi, M., Rohbani, N., and Miremadi, S. "A Low Area Overhead NBTI/PBTI Sensor for SRAM Memories", IEEE Trans. Very Large Scale Integr. Syst., 25(11), pp. 3138–3151 (2017). doi: 10.1109/TVLSI.2017.2734839

[31] Liu, B. and Chen, C.-H. "Testing, diagnosis and repair methods for NBTI-induced SRAM faults", IEEE Int. Conf. IC Design & Technol., pp. 1–4 (2014). doi: 10.1109/ICICDT.2014.6838608

[32] Maghsoudloo, M. and Zarandi, H. R. "Design space exploration of non-uniform cache access for soft-error vulnerability mitigation", Microelectron. Reliab., 55(11), pp. 2439–2452 (2015). doi: 10.1016/j.microrel.2015.07.049

[33] Gebregiorgis, A., Ebrahimi, M., Kiamehr, S., Oboril, F., Hamdioui, S., and Tahoori, M. "Aging mitigation in memory arrays using self-controlled bit-flipping technique", 20th Asia South Pacific Des. Autom. Conf., pp. 231–236 (2015). doi: 10.1109/ASPDAC.2015.7059010

[34] Sadeghi, M. and Nikmehr, H. "Aging-mitigation of cache memories by intra-word bit swapping", Microprocess. Microsyst., 72 (2020). doi: 10.1016/j.micpro.2019.102941

[35] Duan, S., Halak, B., and Zwolinski, M. "Cell Flipping with Distributed Refresh for Cache

Ageing Minimization", IEEE Asian Test Symp., pp. 98–103 (2018). doi: 10.1109/ATS.2018.00029

[36] Hennessy, J. and Patterson, D. "Computer Architecture: a quantitative approach, sixth edition", Morgan Kaufmann Publishers Inc., San Francisco, USA (2017). [Online]. Available: https://dl.acm.org/doi/book/10.5555/3207796

[37] Abel, A. and Reineke, J. "Reverse engineering of cache replacement policies in Intel microprocessors and their evaluation", IEEE Int. Symp. Performance Anal. Syst. Softw. (ISPASS), pp. 141–142 (2014). doi: 10.1109/ISPASS.2014.6844475

[38] Stine, J., Ozev, S., Brown, A., and Butts, K. "FreePDK3D45: An Open-Source Predictive Technology Model for 3D IC Research", Department of Electrical and Computer Engineering, North Carolina State University, [Online]. Available: https://eda.ncsu.edu/freepdk/freepdk3d45/ (accessed June 3, 2024).

Mohammad Maghsoudloo received his M.Sc., and Ph.D. degrees, all in the department of computer engineering at Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2012, and 2016, respectively. He is currently an assistant professor in the computer engineering and information technology department at Golestan University since 2017. His research interests include dependability evaluation, fault-tolerant computing, dependable computer architecture, high-performance computing, cloud computing/storage architectures, and the Internet of things. He established the “Cloud of Things (CoT)” research center at Golestan University in 2018. He is a member of the IEEE Computer Society and the Computer Society of Iran (CSI).

List of captions:

Fig. 11. Effects of the Enhanced cache on the cache performance metrics (results are normalized to the results of the Baseline cache).

Fig. 12. Effects of the enhanced cache on the area consumption metrics.

Table 3. Long-Term SNM and Threshold Voltage Degradation Across PVT Corners.

Fig. 13. The signal probability values for each individual bit position (0 to 63) across a 64-bit cache line, evaluated under six different configurations: Base, IBS, CFDR, DCC, CSM, and ECE.

Fig. 14. The normalized dynamic energy consumption of cache memory cells across various benchmark applications, considering different cache associativities: 4-way, 8-way, and 16-way.

Fig. 15. The normalized miss rates across standard workloads for 4-way, 8-way, and 16-way associative caches.

Fig. 16. The normalized write-back rate to L2 cache across standard workloads for 4-way, 8-way, and 16-way associative caches.

Fig. 17. The area overhead associated with both the control logic and the overall cache system, relative to a baseline design.

List of figures and tables:

Table 1.

Feature	Details	Feature	Details
Frequency	5 GHz	Branch Penalty	17 Cycles
Branch Predictor	16K entry, Bimodal and Gshare	RAS	32 entries
Fetch/Issue/Commit	4/4/4	BTB	2K entries, 4-way
IL1/DL1 Cache	64KB, 4-way, Line size: 64B	L2 Cache	1MB, 4-way, Line size: 32B
Coherency Protocol	MESI	Replacement Policy	Tree-based P-LRU
Write Policy	Write-back, Inclusive	Pipeline Depth	14 stages
Execution Units	2 Integer ALUs, 1 FPU, TDP: 95W	Prefetching	Prefetcher for L2, Memory Bandwidth: 25 GB/s

Fig. 1.

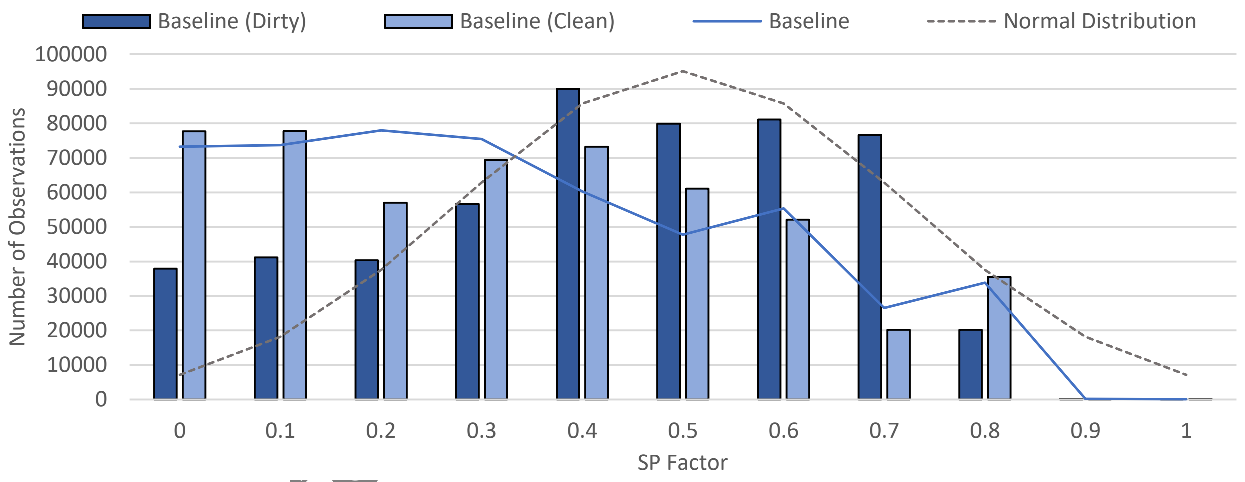


Fig. 2.

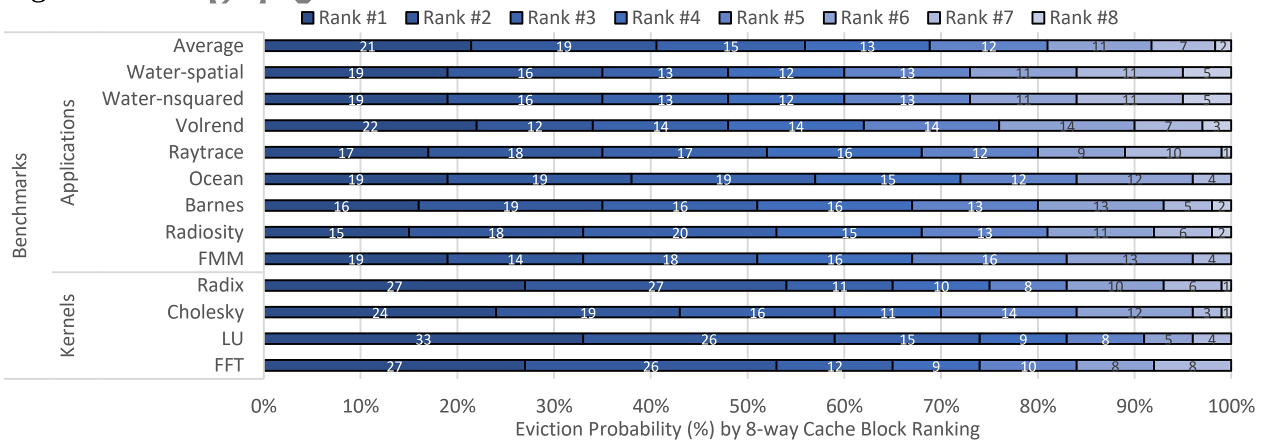


Fig. 3.

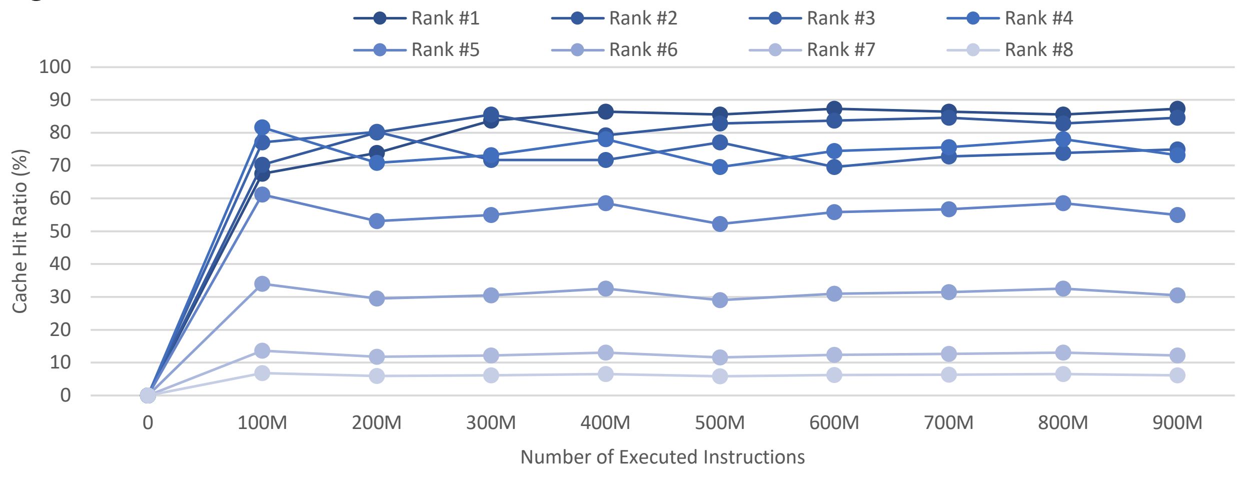


Fig. 4.

Algorithm 1 Enhanced Early Cache Eviction Strategy for a n-way Cache

```
1: procedure ENHANCED-PLRU-NWAY(C, B, D, ConflictType)
2:     C ← {Line1, Line2, ..., Linen}  //Cache lines (n-way cache)
3:     B ← {B0, B1, ..., Blog2(n)-1}  //Decision bits for the n-way tree
4:     D ← {Dirty[Line1], Dirty[Line2], ..., Dirty[Linen]} //Dirty bits
5:     repeat
6:         current_level ← 0
7:         candidates ← C
8:         while |candidates| > 2 do
9:             branch_bit ← B[current_level]
10:            candidates ← LeftHalf(candidates) if branch_bit == 0 else RightHalf(candidates)
11:            current_level ← current_level + 1
12:        end while
13:
14:        C1_candidates ← candidates //Final 2 candidates
15:
16:        if Dirty[C1_candidates[0]] == Dirty[C1_candidates[1]] then
17:            victim ← C1_candidates[0] if B[current_level] == 0 else C1_candidates[1]
18:        else
19:            if ConflictType == "WriteMiss" then
20:                victim ← line ∈ C1_candidates where Dirty[line] == 0
21:            else if ConflictType == "ReadMiss" then
22:                victim ← line ∈ C1_candidates where Dirty[line] == 1
23:            end if
24:        end if
25:
26:        Replace(victim, NewData)  //Replace victim line with new data
27:        UpdateTreeBits(victim, B) //Update tree bits to reflect replacement
28:
29:    until CacheAccess == Complete
30:    return CacheStatus
31: end procedure

# Helper functions
32: function LeftHalf(candidates)
33:     return candidates[0 : |candidates| / 2]
34: end function
35: function RightHalf(candidates)
36:     return candidates[|candidates| / 2 : |candidates|]
37: end function
```

Table 2.

Symbol / Function	Description
`C`	A list of cache lines in the current set
`B`	An array of PLRU decision bits representing internal nodes of the PLRU binary tree.
`D`	A list of dirty status bits corresponding to each cache line.
`ConflictType`	A runtime parameter identifying the type of cache miss.
`C1_candidates`	The final two cache lines selected from the tree traversal and subject to eviction evaluation.
`Victim`	The chosen cache line to be evicted and replaced.
`Replace(victim, NewData)`	A procedure that evicts the victim line and inserts the new data block.
`UpdateTreeBits(victim, B)`	Updates the PLRU decision bits to reflect the most recent use of the victim line.
`current_level`	An index or depth indicator used to iterate over PLRU decision bits during tree traversal.
`LeftHalf(candidates)`	A helper function that returns the left half of a given candidate list.
`RightHalf(candidates)`	A helper function that returns the right half of a candidate list.

Fig. 5.

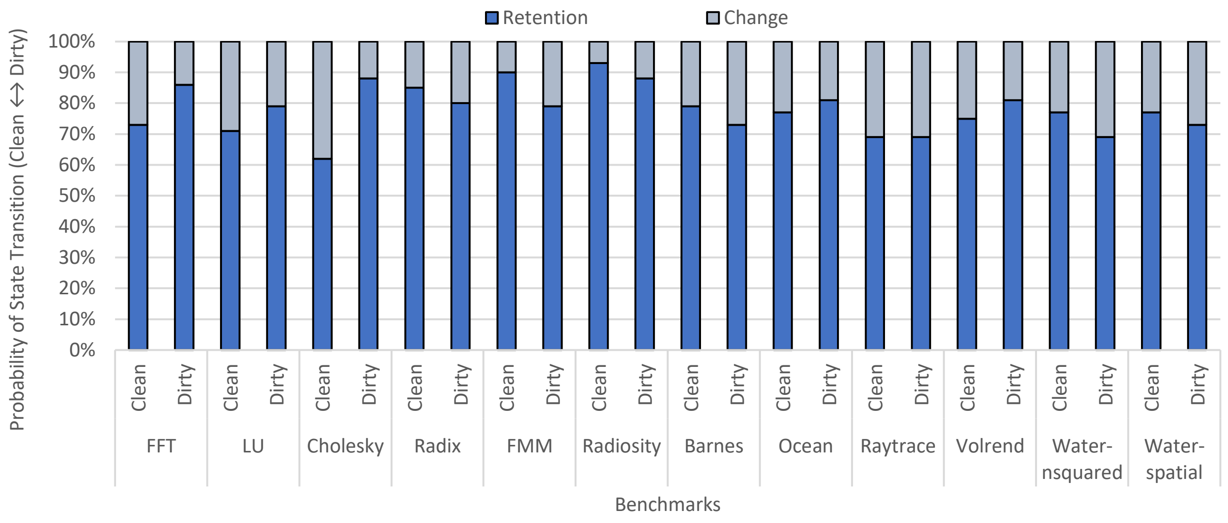


Fig. 6.

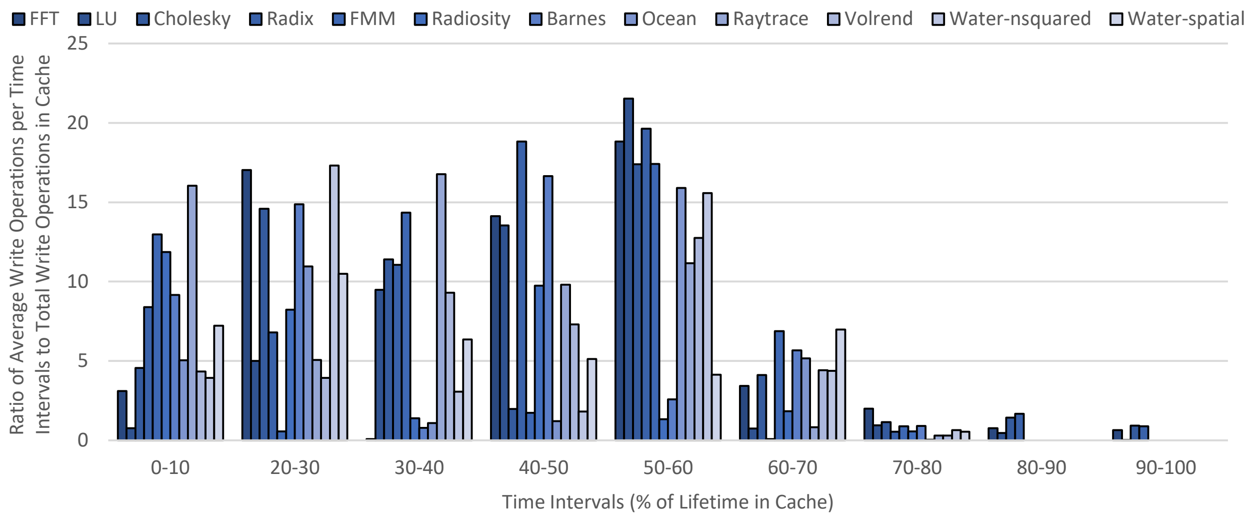

FFT
LU
Cholesky
Radix
FMM
Radiosity
Barnes
Ocean
Raytrace
Volrend
Water-nsquared
Water-spatial
Ratio of Average Write Operations per Time Intervals to Total Write Operations in Cache
25
20
15
10
5
0
0-10
20-30
30-40
40-50
50-60
60-70
70-80
80-90
90-100
Time Intervals (% of Lifetime in Cache)

Fig. 7.

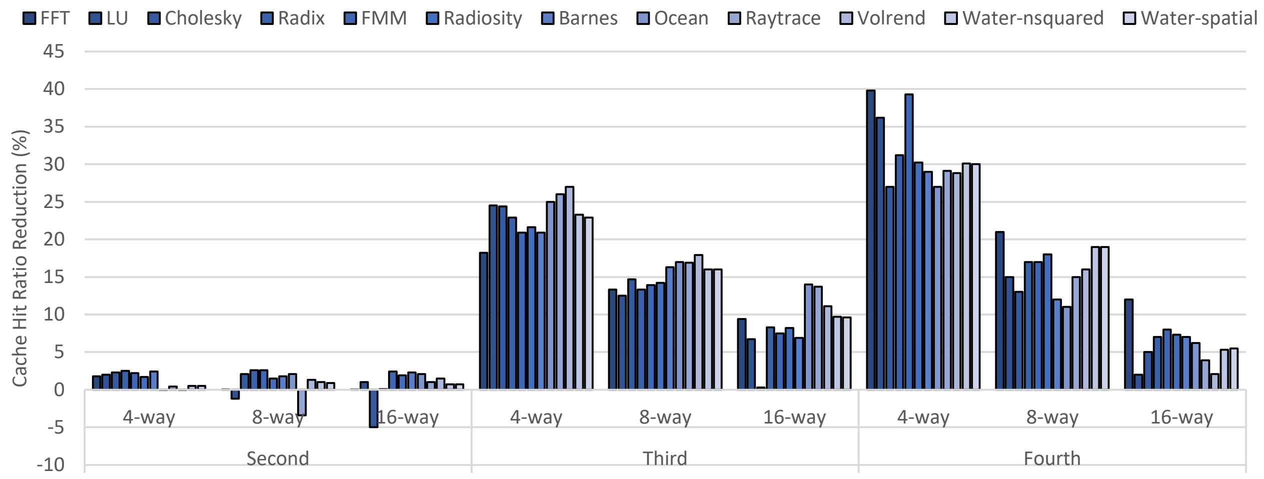

FFT
LU
Cholesky
Radix
FMM
Radiosity
Barnes
Ocean
Raytrace
Volrend
Water-nsquared
Water-spatial
Cache Hit Ratio Reduction (%)
45
40
35
30
25
20
15
10
5
0
-5
-10
4-way
8-way
16-way
4-way
8-way
16-way
4-way
8-way
16-way
Second
Third
Fourth

Fig. 8 (a).

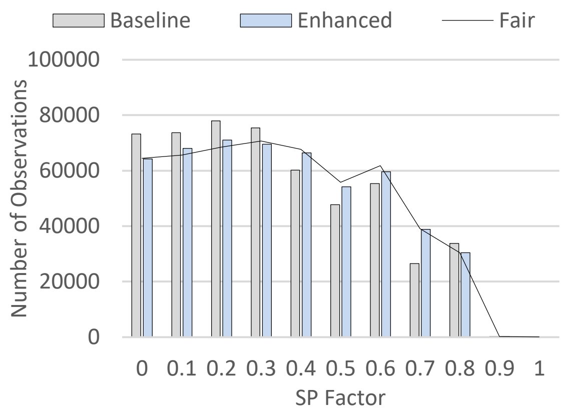

Baseline
Enhanced
Fair
Number of Observations
100000
80000
60000
40000
20000
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
SP Factor

Fig. 8 (b).

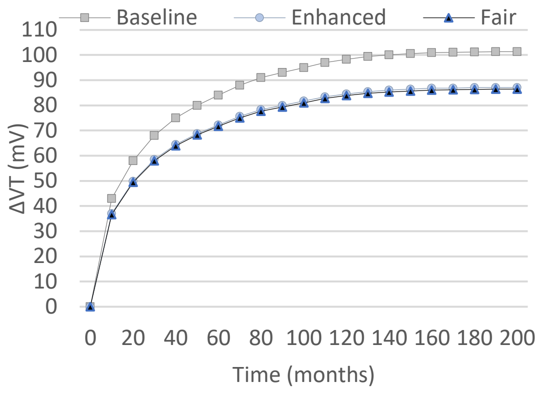


Fig. 9 (a).

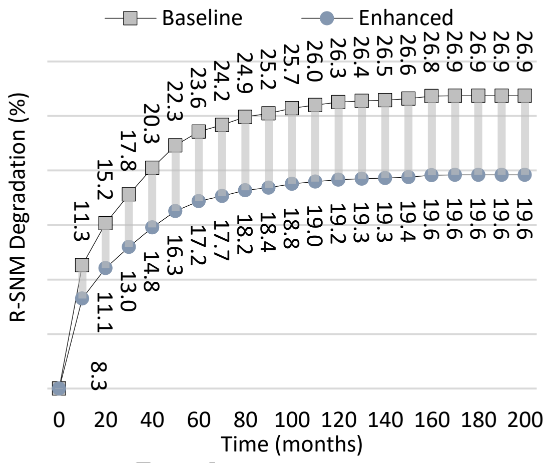


Fig. 9 (b).

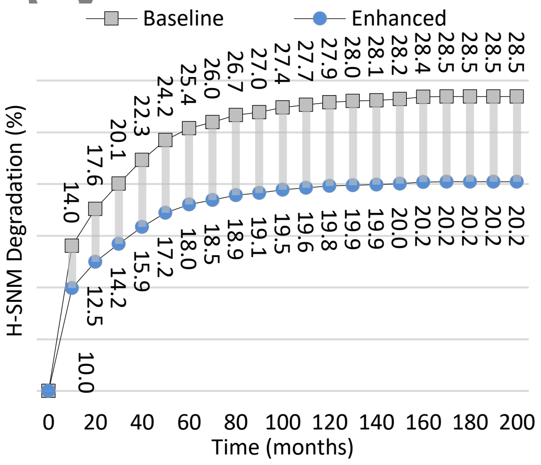


Fig. 10 (a).

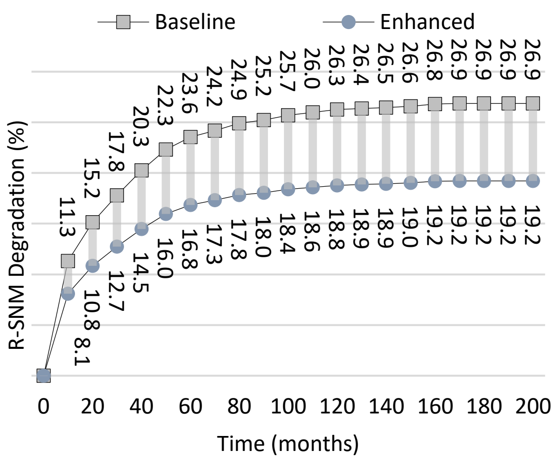


Fig. 10 (b).

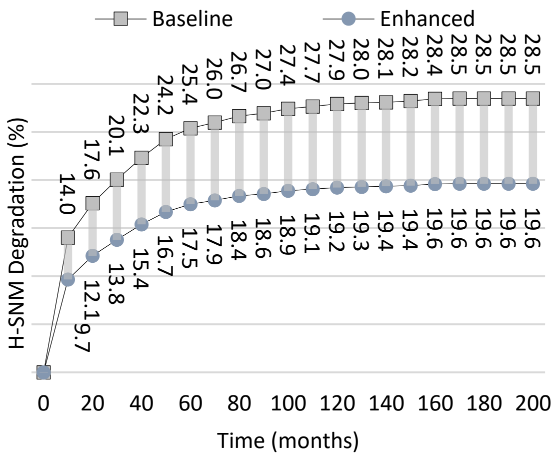


Fig. 11.

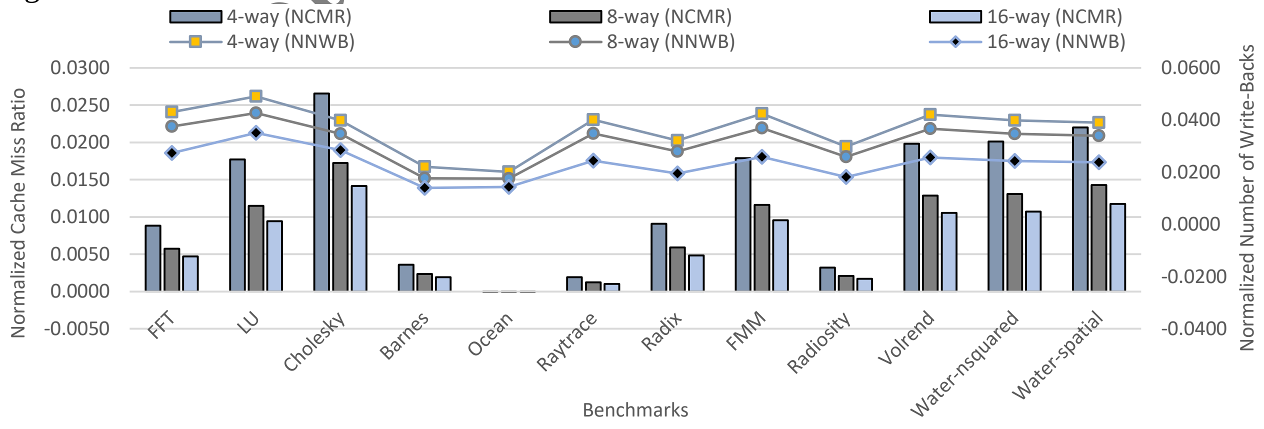


Fig. 12.

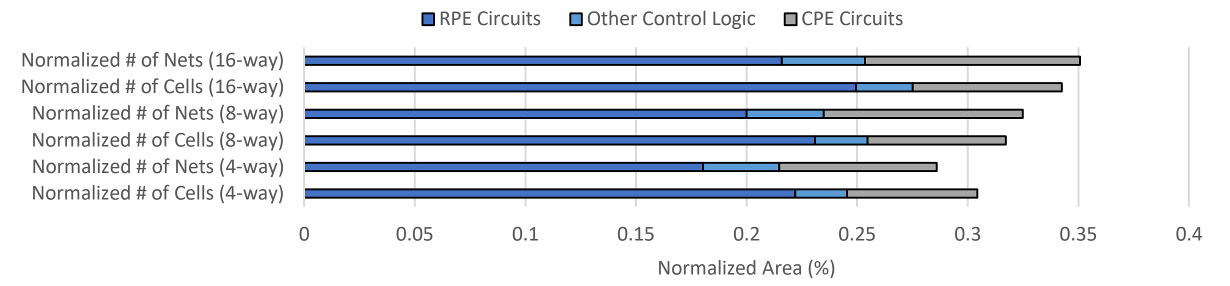


Table 3.

Operating Condition	**ΔV$_t$ (Base)**	**ΔV$_t$ (ECE)**	**SNM Hold (Base)**	**SNM Hold (ECE)**	**SNM Read (Base)**	**SNM Read (ECE)**
25°C, 1.0V *(Nominal)*	101.3 mV	87.1 mV	-29%	-20%	-27%	-19%
85°C, 0.9V *(Stress corner)*	115.5 mV	98.7 mV	-33%	-23%	-31%	-22%
-10°C, 1.2V *(Conservative)*	84.9 mV	69.3 mV	-24%	-16%	-22%	-15%

Fig. 13.

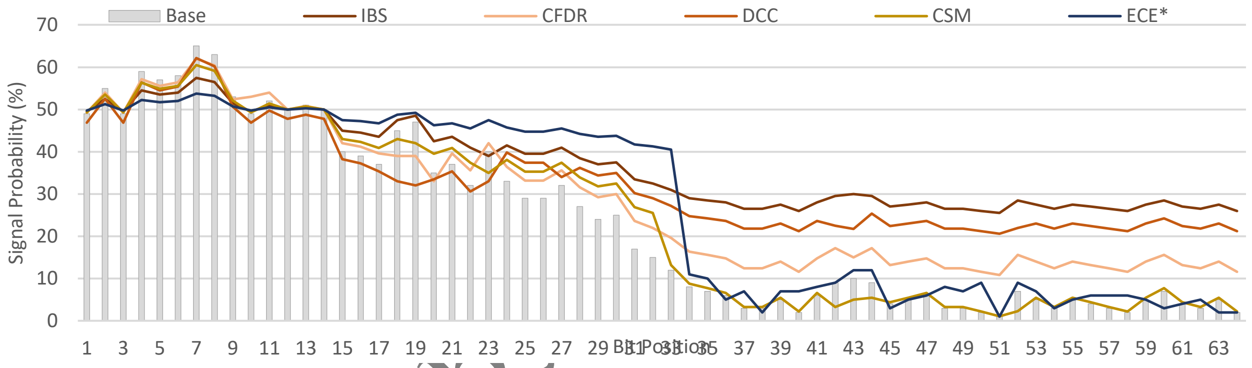


Fig. 14.

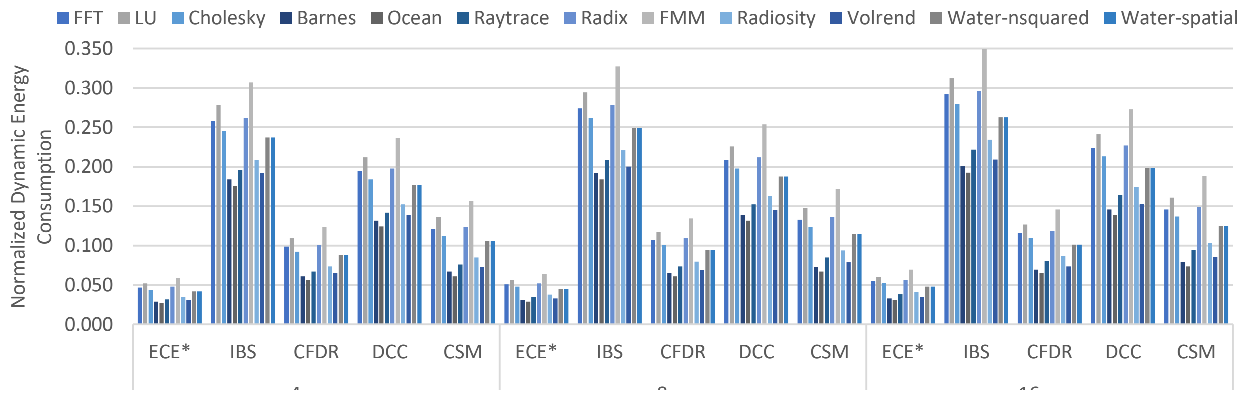


Fig. 15.

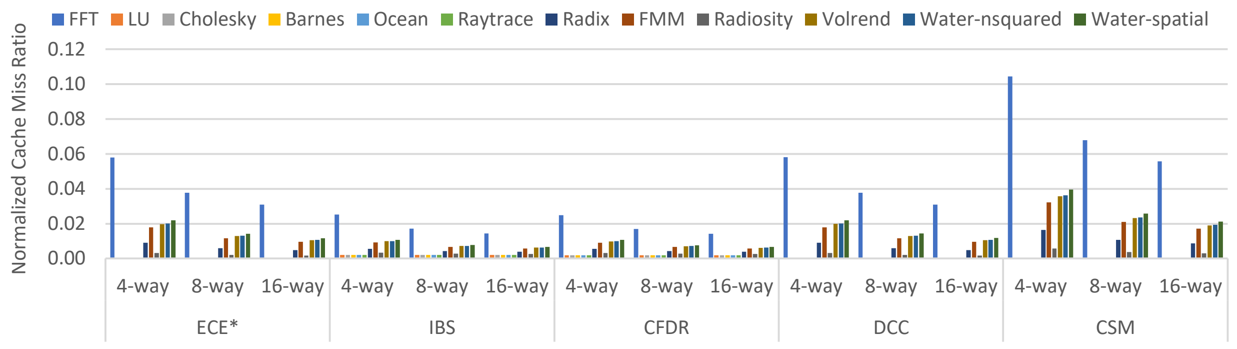


Fig. 16.

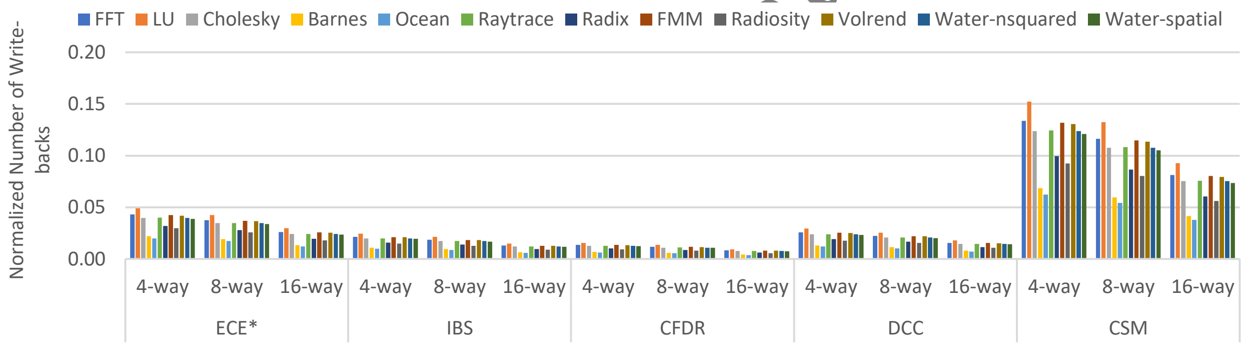
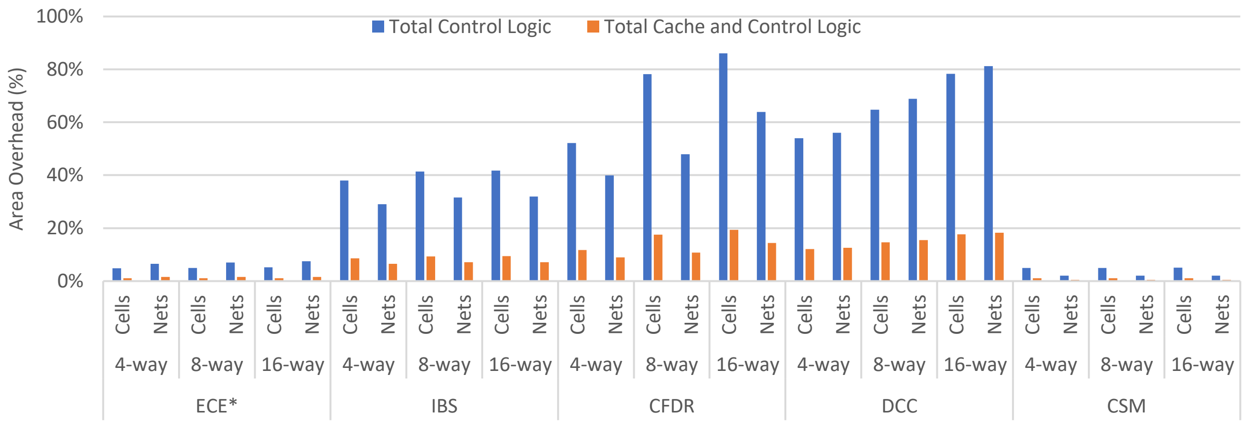


Fig. 17.


Total Control Logic
Total Cache and Control Logic
Area Overhead (%)
Cells
Nets
4-way
8-way
16-way
ECE*
IBS
CFDR
DCC
CSM