2025 IEEE International Symposium on High-Performance Computer Architecture (HPCA)

Rethinking Dead Block Prediction for Intermittent Computing

Gan Fang and Changhee Jung
Department of Computer Science, Purdue University
{fang301,chjung}@purdue.edu

***Abstract*—Existing dead block predictors have proven to be effective in reducing cache leakage power of conventional systems. However, prior work is significantly less effective in energy harvesting systems in that it does not take into account their unique characteristic, i.e., frequent power failure during program execution. Even if some cache blocks are predicted to be live, they may not be used due to their loss upon power failure. In response, this paper introduces EDBP, an extension to existing dead block predictors, to enhance their performance in various energy harvesting environments. EDBP can identify and deactivate those cache blocks that are not reused before upcoming power failure—though they are considered live by the existing predictor—thereby lowering cache leakage and preserving more energy for forward execution progress. Experimental results show that for 20 applications from Mediabench and MiBench, EDBP alone reduces total energy consumption by 6.5% and improves performance by 6.9% compared to the baseline with no dead block predictor. When combined with a conventional dead block predictor (Cache Decay), EDBP achieves 9.8% reduction in total energy consumption—approaching the theoretical minimum—and 11.9% performance improvement.**

I. Introduction

Energy harvesting systems have become a popular alternative to battery-powered embedded devices, offering benefits such as environmental friendliness and self-sustainability owing to the battery-free nature [53]. These systems can be widely adopted across various domains, including batteryless IoT [10], [22], [30], [67], stream and river surveillance [31], [60], health and wellness monitoring [12], [13], [20], [51], and wearable computing [14], [17], [39], [48], [49]. Instead of relying on a battery, they harvest energy from ambient sources such as radio frequency (RF) and WiFi. However, these sources are often unstable and weak, resulting in frequent power failure on which all volatile data is lost [1], [6], [11], [27], [28], [33], [40], [41], [43], [57], [62]. To avoid losing critical data and guarantee correct recovery from power failure, researchers have leveraged nonvolatile memory (NVM) as the main memory and developed crash consistency mechanisms—that typically perform checkpoint and restore operations across the failure [8], [16], [69], [75].

A critical component of energy harvesting systems is the capacitor that serves to buffer the harvested energy. Along with frequent power interruptions, this characterizes energy harvesting systems as *intermittent computing systems*. That is, they compute only when the capacitor is sufficiently charged. Upon depletion, they lose all volatile states due to the resulting power outage and should hibernate to recharge. They can only resume after securing enough energy in the capacitor. In the literature, the period from the resumption to the next power outage is called a *power cycle*.

The takeaway is that energy availability directly impacts the forward execution progress, i.e., the more energy the longer power cycle, and in turn influences the performance of energy harvesting systems. In a sense, improving the energy efficiency is the key to achieving performance gains. Previous research has primarily focused on reducing checkpointing costs to conserve harvested energy for forward progress. One prevalent and effective approach is so-called just-in-time (JIT) checkpointing that can minimize the costs by making a single checkpoint at the end of each power cycle and restoring the data at the start of the next. This approach improves energy efficiency by dedicating more of the available energy to program execution rather than frequent checkpoint operations.

To further improve energy efficiency, researchers have recently proposed techniques for incorporating SRAM caches without compromising the crash consistency. Data reuse in the cache can save energy a lot by avoiding the NVM access that is the most energy-consuming operation in the processor. For example, a prior cache-enabled energy harvesting system achieves about 90% energy consumption reduction and the resulting speedup of 8.5x compared to the cacheless baseline [69]. However, the introduction of SRAM caches causes significant leakage current [2], [32], possibly wasting harvested energy that could otherwise be spent on forward execution progress. Traditionally, dead block predictors can effectively mitigate the leakage problem [2], [32], [74], e.g., Cache Decay [32] marks blocks as dead if they have been unused for a period of time since their last access. By deactivating such dead blocks [52], Cache Decay reduces cache leakage energy by roughly 80% with little impact on performance [32].

Nevertheless, existing dead block predictors are originally designed without taking into account intermittent computing and just treat a block as dead if it has no further access before eviction. As a result, they fail to achieve optimal energy savings in energy harvesting systems where power outages act as an additional eviction mechanism. That is, even if some cache blocks are predicted to be live, they may disappear on a power outage before their access; caching such unusable blocks wastes hard-won energy in vain, and we refer to them as **zombie** blocks in that they look alive but are dead. Unfortunately, this problem is exacerbated by the frequent power outages of energy harvesting systems [15],

[69], resulting in many zombie cache blocks and significant performance degradation.

In response, this paper introduces EDBP, an **E**xtension to existing **D**ead **B**lock **P**redictors so that when needed, they let EDBP handle zombie cache blocks—on their manifestation near a power outage. If power is steady, i.e., no sign of zombie blocks yet, the existing predictor just works fine; otherwise EDBP takes over to deactivate them. That way EDBP creates a synergy with the existing predictor in reducing cache leakage energy, thereby significantly improving the performance of energy harvesting systems. For the successful realization of EDBP, it is important to enable EDBP in a timely manner.

To achieve this, EDBP makes an important observation that as a power outage is approaching, more cache blocks turn into zombie blocks. This implies that EDBP can anticipate their manifestation based on the proximity to the next power outage. Therefore, EDBP tracks the capacitor's voltage, the degradation of which triggers EDBP to take over. When the voltage drops below a preset threshold, i.e., a power outage occurs soon, EDBP starts to deal with the potential zombie blocks. EDBP then regards near-LRU blocks as zombies—with the upcoming power outage in mind—and deactivates them for power saving. More precisely, EDBP leverages multiple thresholds that control how aggressive the zombie detection is, e.g., at a higher voltage threshold, EDBP takes a conservative approach which only treats the LRU blocks as zombies. The upshot is that EDBP can easily get the recency information of cache blocks, which is the basis for its zombie detection, as long as LRU caches or their variants are used.

The experiment with 20 applications from Mibench [25] and Mediabench [39] shows that for a real RF energy-harvesting power trace [23], [55], EDBP alone achieves 6.5% energy consumption reduction and 6.9% overall performance improvement compared to the baseline without dead block prediction. Moreover, it turns out that EDBP works well in combination with a conventional dead block predictor (Cache Decay [32]), resulting in 9.8% energy consumption reduction and 11.9% overall performance improvement. Note that **EDBP delivers robust performance across diverse experimental conditions varying capacitor size, energy condition, cache size, cache associativity, and NVM technology.** The contributions of this paper can be summarized as follows:

- EDBP adapts dead block prediction for energy harvesting systems with their unique characteristics in mind.
- EDBP significantly improves the energy efficiency and performance of energy harvesting systems.
- EDBP is lightweight, piggybacking on the existing LRU cache structure.

II. Energy Harvesting System Basics

Energy harvesting systems often take a just-in-time (JIT) checkpointing approach which saves volatile states right before impending power failure and restores them in the wake of the failure [5], [19], [26], [27], [35], [43], [45]–[47], [58], [70], [72]. To trigger the checkpoint just in time, this approach takes advantage of a voltage monitor that continuously inspects the variation of supply voltage. Whenever the capacitor voltage dips below a predefined threshold, i.e., power failure is about to occur, the voltage monitor immediately and signals the checkpointing logic to save volatile registers including a program counter (PC); the threshold should be high enough to dedicate a sufficient amount of energy for failure-atomic checkpointing. When the detected voltage rises above another restoration threshold, the register file is restored from the checkpointed registers.

There are three-fold benefits of the JIT checkpointing approach. First, since the PC has been checkpointed as a part of the register file, program can pick up from when it got interrupted by the power failure. Second, this allows for roll-forward recovery, which is free from crash inconsistency issues, e.g., write-after-read also known as anti-dependence, and thus the JIT checkpointing approach can safely ignore them achieving simple yet effective recovery. Third, the JIT checkpointing approach can minimize the energy consumption for the volatile state saving by making a single checkpoint at the end of each power cycle, i.e., only at the moment of power failure, rather than periodically creating checkpoints while power is stable.

Recently, energy harvesting systems have adopted SRAM cache on top of nonvolatile main memory to improve their performance. For correct power failure recovery, the cache-enabled energy harvesting systems integrate the JIT checkpointing approach with the volatile cache so that it can maintain data across the failure. Among the existing systems, NVSRAMCache is the most popular and promising [5], [27], [43], [46], [47], [58], [59], [64]. Instead of relying on slow nonvolatile main memory as checkpoint storage, NVSRAMCache instantly checkpoints all necessary volatile states, i.e., registers and dirty cache blocks, to their neighboring nonvolatile counterparts. Thus, NVSRAMCache can significantly reduce both the checkpoint and the restoration costs and improve the performance a lot by turning expensive NVM access into SRAM access in the cache.

III. Motivation

However, due to resource constraints—and the energy secured for the failure-atomicity of the JIT checkpointing, energy harvesting systems typically use very small SRAM caches, leading to frequent cache misses and high energy consumption due to their resulting NVM accesses. A naive solution is to increase the cache size, accommodating more data therein. Unfortunately, larger SRAM caches come with significantly higher leakage power, to the point of compromising the overall efficiency and performance of energy harvesting systems. As shown in the second row of Table I, the leakage power of a 4-way SRAM cache increases significantly as its size grows from 256B to 16kB. Specifically, the leakage rises from 0.09 mW to 3.54 mW, under the same configuration described in our evaluation Section VI-A. Additionally, the third row of the table presents the ratio of static energy to the total energy consumption of the SRAM data cache. As cache size

TABLE I: SRAM cache leakage power (mW) and the ratio (%) of static energy to the total energy consumption of the SRAM data cache.

Size	256B	512B	1kB	2kB	4kB	8kB	16kB
Power (mW)	0.09	0.17	0.35	0.69	1.22	2.43	3.54
StaticEnergy%	17.04	29.11	43.58	59.29	72.79	83.88	90.89

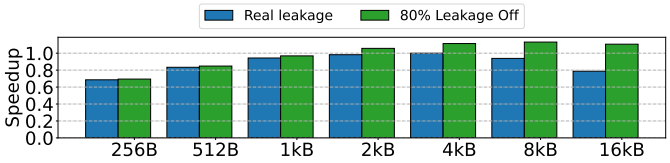

Fig. 1: Performance across different cache sizes. All speedups are normalized to 4-way 4kB caches with real leakage.

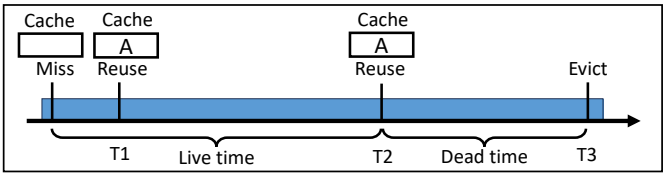

Fig. 2: Dead block definition. Blue block means continuous program execution.

increases, static energy increasingly dominates overall cache energy consumption.

The takeaway is that energy harvesting systems must pick a suitable cache size, which would otherwise waste a substantial amount of hard-won energy and lead to performance degradation. To find the most suitable cache size, we evaluated different cache sizes on top of NVSRAMCache for 20 applications from Mediabench and MiBench [25], [39], using the default settings in Table II. Figure 1 shows that while increasing the cache size from 256B to 4kB enhances performance, enlarging the cache beyond 4kB rather degrades performance due to the substantial energy wasted by higher cache leakage power. Therefore, this paper selects 4kB as the default cache size of EDBP.

To stress-test the impact of cache leakage on performance, we also evaluated the performance of NVSRAMCache by magically reducing cache leakage by 80% without impacting the cache hit rate. The results, shown in Figure 1 (labeled as *80% Leakage Off*), clearly demonstrate that reducing cache leakage allows NVSRAMCache to achieve higher performance gains as the cache size increases from 256B all the way up to 16kB. This implication is that the performance of energy harvesting systems is severely constrained by excessive cache leakage energy.

A straightforward approach to addressing the cache leakage problem is to use existing dead block predictors [2], [32], [74] for identifying and deactivating cache blocks that are not usable but just consume energy in vain. Figure 2 shows an example of the dead block in conventional processors. *Block A* is live since it has reuse at $T1$ and $T2$. However, it becomes

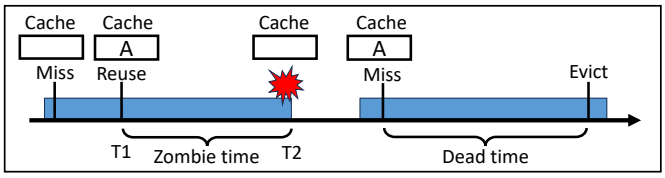

Fig. 3: Zombie block definition in energy harvesting systems. Blue block means continuous program execution.

a dead block from $T2$ to $T3$ since it does not have any accesses before its eviction at $T3$. In light of this, dead block predictors leverage gate-Vdd [52] to deactivate these dead blocks to reduce cache leakage power. One notable example is Cache Decay [32] that considers blocks dead if they remain unused for a specific duration since their last access. However, when existing dead block predictors are integrated into energy harvesting systems, they can be easily fooled by power outages which serve as an additional eviction mechanism. As a result, those cache blocks predicted to be live may not receive any accesses before their loss upon a power outage. We refer to such blocks as zombie blocks. By deactivating these zombie blocks, static cache energy consumption can be further reduced without impacting performance.

Figure 3 shows this scenario. At $T1$, the existing dead block predictor (Cache Decay) can accurately predict that *Block A* will be reused in the future, so *Block A* remains in the cache with no power gating. However, a power outage occurring before its reuse causes the cached data to be lost ($T2$). Consequently, *Block A* is classified as a zombie during the time period from $T1$ to $T2$. Here, the static energy consumed by *Block A* from its last access until the power outage is entirely wasted, as it does not contribute to any cache hits at all.

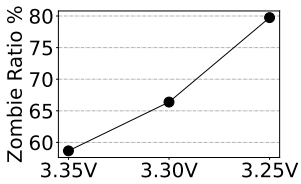

Fig. 4: The ratio of zombie blocks to total blocks with varying capacitor voltage.

Unfortunately, this kind of energy waste can be significant since the portion of these zombie blocks increases as the capacitor voltage decreases—i.e., approaching a power outage—as shown in Figure 4. Notably, when the voltage falls below 3.25 V, approximately 80% of the cache blocks become zombies.

Even worse, due to the high frequency of power outages in energy harvesting systems, such zombie blocks can occur repeatedly throughout program execution. As a result, current dead block predictors miss many opportunities to reduce cache leakage energy more aggressively and efficiently. Therefore, the goal of this paper is to tailor existing dead block predictors for energy harvesting systems by identifying zombie blocks and turning them off to reduce energy consumption.

IV. Zombie, Dead and Live Blocks

To accurately assess energy savings of dead block prediction in energy harvesting systems, it is necessary to redefine the

metrics used in the literature by considering the zombie blocks. These metrics include correct/wrong dead block prediction, coverage, and accuracy. For better understanding, we refer to those cache blocks that are reused before the power outage and their eviction as *live* blocks in the following.

Correct predictions consist of true positives, where dead and zombie blocks are accurately deactivated, and true negatives, where live blocks are correctly retained. On the other hand, incorrect predictions include false positives, where live blocks are mistakenly deactivated, and false negatives, where dead and zombie blocks are unnecessarily kept active in the cache. Based on this, *coverage* in dead block prediction can be redefined by the ratio of correctly identified dead and zombie blocks (true positives) to the total number of dead and zombie blocks which includes not only correctly identified dead and zombie blocks (true positives) but also those that were incorrectly marked as live (false negatives), as shown in Equation 1. Similarly, *accuracy* can be redefined by the ratio of correct predictions, which is the sum of correctly identified dead and zombie blocks (true positives) and correctly identified live blocks (true negatives), to the total number of predictions, as shown in Equation 2.

$$Coverage = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (1)$$

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalPredictions} \quad (2)$$

V. Implementation

The goal of EDBP is to find and deactivate zombie blocks with the anticipation of power outages. For this purpose, EDBP monitors the capacitor's voltage level to gauge the proximity of the next power outage and determine the optimal timing to start recognizing and killing zombie blocks. Specifically, when the voltage drops below a predefined threshold, indicating an imminent power outage, EDBP takes over to deal with the occurrence of zombie blocks. This raises the following questions: *Which cache blocks should be identified as zombies? When to detect the zombie blocks? How aggressive the zombie block detection needs to be depending on the proximity to upcoming power failure?* The following sections address these questions in detail.

A. Which Block to Deactivate

EDBP follows two key principles for the identification of potential zombie cache blocks. The first principle is to track down those blocks that are least likely to be accessed again—before the next power outage. In this way, EDBP can avoid the occurrence of cache misses to the greatest extent, while deactivating those with little recency as zombies. Note that achieving this design is simple and does not require heavy modification on cache microarchitecture. EDBP can refer to any cache replacement policy in capable of holding the information about which cache blocks are least likely to be accessed. For instance, with a (pseudo) LRU replacement policy, EDBP prioritizes those blocks near the LRU position as zombies by checking their LRU bits.

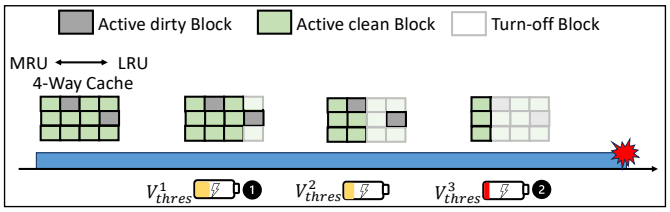

Fig. 5: Zombie blocks detection and deactivation.

The second principle is to select cache blocks that can accomplish more energy savings when they are deactivated. In reality, the energy cost for deactiviating a cache block varies depending on its state, i.e., dirty or clean. That is because dirty blocks require their write back before deactivation, which would otherwise lead to data loss, whereas clean blocks do not. This implies that deactivating clean blocks may allow energy harvesting systems to save more energy. Therefore, EDBP prioritizes clean cache blocks over dirty ones during its zombie block detection. Based on these two principles, EDBP can identify suitable blocks as zombies.

B. When to Deactivate Zombie Blocks

To decide when to take care of zombie blocks, EDBP sets the voltage thresholds (V_{thres}) at different levels, the reach of which triggers their detection and deactivation. As the capacitor voltage progressively drops below each predefined threshold, it is a sign that the energy harvesting system moves closer to a power outage. With these thresholds, EDBP can control the aggressiveness of its zombie block detection and deactivation. A higher voltage threshold allows for earlier cache block deactivation, which raises the risk of deactivating blocks that may still be needed and potentially cause cache misses. In response, EDBP can take a conservative approach which only treats the LRU blocks across all sets as zombies. On the other hand, a lower voltage threshold, clear sign of upcoming power failure, allows EDBP to deactivate blocks more confidently since it is less likely to deactivate live blocks in the anticipation of the failure. Thus, in this case, EDBP can be more aggressive by deactivating not just the LRU block but also near-LRU blocks.

Figure 5 presents a high-level view of how EDBP detects and deactivates zombies cache blocks. For an n-way associative cache, EDBP prepares $n-1$ voltage thresholds ranging from V^1_{thres} (highest) to V^{n-1}_{thres} (lowest). Whenever capacitor voltage (V) dips below a threshold V^i_{thres}, the corresponding i-th LRU clean blocks are turned off—with an exception for reaching V^{n-1}_{thres} in which case all non-MRU blocks are turned off whether they are clear or dirty. For example, when V drops below V^1_{thres} (❶), only the LRU blocks are deactivated. When the voltage drops below the lowest threshold V^{n-1}_{thres}, it means a power outage is very likely to happen soon; so EDBP more aggressively deactivates both dirty and clean blocks that are not in the MRU position (❷). Notably, EDBP consistently keeps blocks at the MRU position active, regardless of the system's proximity to a power outage. This heuristic is based

on the observation that data in the MRU position is highly likely to be reused shortly [42]. With this design, EDBP can accurately detect the zombie blocks and achieve high cache leakage energy reduction. Then, the remaining question is to determine the values for each threshold.

1) Voltage Values Determination: To identify the optimal threshold values that can maximize the performance of EDBP, we empirically test various benchmark applications [25], [39] under different energy harvesting conditions. However, a fixed voltage threshold may be suboptimal in several scenarios. First, when energy availability fluctuates unpredictably, a transient voltage dip tends to mistakenly deactivate usable cache blocks. Second, slow voltage decreases, indicating it takes a while to encounter a power outage, can cause blocks to be treated as zombies too early, which may lead to high false positive rates. Third, fixed thresholds may falsely deactivate usable blocks especially for those workloads that have a high demand of cache accesses. In summary, all three scenarios arise from variations of harvested energy, rendering fixed thresholds suboptimal. They may result in additional cache misses and higher false positive rates. Therefore, in the pursuit of high performance, EDBP must be capable of adapting to varying energy condition.

For this purpose, EDBP proposes an adaptation technique that adjusts voltage thresholds based on false positive rates on the fly. By monitoring the false positive rate, EDBP can minimize the risk of imprecisely classifying too many live blocks as zombies. To realize this, EDBP empirically determines a reference rate for voltage threshold adjustments. If the current false positive rate exceeds this reference, indicating excessive deactivation of live blocks, EDBP lowers all thresholds by 50 mV[1] to be more conservative in the zombie block detection. Conversely, if the rate is below the reference, it is hard to tell whether the deactivation is overly conservative or appropriately balanced. In that case, EDBP resets the thresholds to their initial values if they are currently lower; otherwise, it keeps them unchanged.

EDBP calculates the false positive rate and accordingly adjusts the voltage thresholds at reboot time, i.e., in the wake of power failure. For each power cycle, EDBP collects statistics necessary for the calculation and saves them at the end of the cycle via the JIT checkpointing—as with other volatile states. In the beginning of the next power cycle, i.e., when the system reboots, EDBP restores these statistics to calculate the false positive rate. To implement this, EDBP requires three extra volatile registers: $R_{WrongKill}$, R_{Total}, and R_{FPR}, along with an SRAM-based FIFO deactivation buffer. Here, $R_{WrongKill}$ tracks the number of live blocks incorrectly deactivated, R_{Total} records the total number of predicted blocks, and R_{FPR} stores the calculated false positive rate ($R_{WrongKill}/R_{Total}$). In addition, EDBP implements a *deactivation buffer* for maintaining the addresses of all the cache blocks that have been deactivated for a given power cycle. Each time a block in the sample set is deactivated, its address is stored in the deactivation buffer—if it is not full; otherwise, EDBP evicts the oldest address to make room.

However, since there are many blocks in the cache, recording their statistics may lead to frequent updates on the deactivation buffer. For example, when LRU blocks need to be deactivated in a 64-set cache, the buffer should be able to accommodate the total 64 block addresses, resulting in significant energy consumption. To solve this problem, EDBP selects a single cache set as a representative sample and collects statistics exclusively from this set. That is, $R_{WrongKill}$ tracks incorrectly deactivated live blocks only for the sample set; likewise, R_{Total} only counts predicted blocks of the same set. That way, the deactivation buffer only needs to follow the deactivated blocks of the sample set. With the help of the sampling mechanism, EDBP can calculate a false positive rate for each power cycle on the cheap without degrading the accuracy (only 3.7% loss).

VI. Evaluation

A. Methodology

TABLE II: Simulation Configuration

	NVSRAMCache	SDBP	Cache Decay	EDBP
V_{max}/V_{min}	3.5/2.8	3.5/2.8	3.5/2.8	3.5/2.8
V_{ckpt}/V_{rst}	3.2/3.4	3.2/3.4	3.2/3.4	3.2/3.4
MCU Power	160 $\mu W/MHz$			
Capacitor	0.47 μF			
Energy Trace	RFHome			
Deact. Buffer	N/A	N/A	N/A	8
Data Cache	4kB, SRAM, 4-way, 16B block, LRU, Access: 5.30 ns/1.05 nJ, Leak: 1.22 mW			
Inst. Cache	4kB, ReRAM, 4-way, 16B block, LRU, Hit: 19.44 ns/3.65 nJ, Miss: 9.99 ns/0.9 nJ, Write: 202.35 ns/3.55 nJ, Leak: 0.22 mW			
Memory	16MB, ReRAM			

1) Simulator, Baseline, and Competitors: We model EDBP and other schemes on the gem5 [9], simulating a 25 MHz single-core in-order nonvolatile processor based on ARM ISA with 16 registers as in NVPsim [23]. Table II details the simulation configurations for EDBP, the baseline NVSRAMCache, and two competitors, i.e., SDBP [44] and Cache Decay [32]. For the cache setting, all these schemes leverage SRAM as the write-back data cache and ReRAM serves as the instruction cache, as with prior studies [5], [16], [27], [43], [46], [47], [58], [59], [64], [69], [75]. To accurately model cache and memory access latencies and power consumptions, we utilized NVSim [18] calibrated with parameters from 180 nm technology [7], [24], [50], [54].

For the baseline architecture, we choose NVSRAMCache [23] which makes a JIT checkpoint of the entire register file and all dirty cache blocks to their nonvolatile counterparts right before each power outage. SDBP enhances the performance of NVSRAMCache by utilizing dead block prediction. Specifically, it mitigates the cold cache effect caused by power failure by checkpointing only the cache blocks likely to be reused in the future. In the wake of the power failure, SDBP

[1] 50 mV is the resolution of the state-of-the-art voltage monitor [63] used in this work.

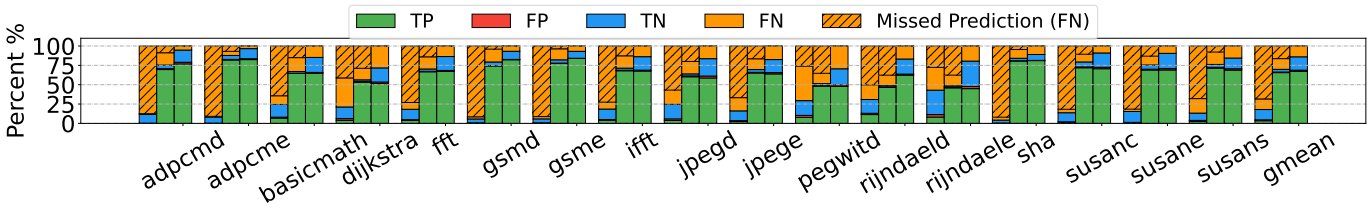

Fig. 6: True/false rates. TP=True Positives; FP=False Positives; TN=True Negatives; FN=False Negatives; Misses Prediction (FN) means dead blocks are not predicted, which can also be categorized as FN. There are three bars for each application. **From left to right: Cache Decay, EDBP, Cache Decay+EDBP.**

restores these blocks to the cache, effectively reducing the cache miss rate. Unlike other approaches that save the entire cache, SDBP selectively avoids checkpointing and restoring dead blocks, conserving more harvested energy for forward progress.

In particular, we evaluate the performance of EDBP in two different settings. First, we analyzed EDBP independently to determine its impact on the baseline architecture. Second, we combined EDBP with Cache Decay to evaluate the enhancements it brings to a conventional dead block predictor. These evaluations help quantify the individual and synergistic benefits of EDBP in optimizing energy efficiency and the performance of energy harvesting systems.

2) Benchmarks, Traces, and Sensitivity Analyses: We utilize 20 applications from Mediabench and MiBench benchmark suites [25], [39] for evaluation. In the following, we assess the performance of all schemes using multiple real-world energy harvesting traces, including RFHome, RFOffice, solar, and thermal [23], [55]. The solar and thermal sources have higher portions of high energy while RFHome and RFOffice have less. As a result, energy harvesting systems experience fewer power outages when running with thermal and solar traces, but encounter more frequent outages with RFHome and RFOffice traces. Besides, we conduct sensitivity analysis on the capacitor size, cache replacement policy, cache size, cache associativity, memory size, and NVM technology. This helps to evaluate the robustness of EDBP and determine how each parameter impacts the overall energy efficiency and performance of the energy harvesting system.

B. Hardware Cost Analysis

EDBP incurs low hardware overhead since it leverages many existing hardware structures. For example, it uses the sleep transistors in Cache Decay to deactivate zombie blocks. Besides, EDBP gets recency information from the LRU cache replacement policy to select zombie blocks. Finally, it directly uses the existing voltage monitor in energy harvesting systems to detect the capacitor voltage and tell when to detect and deactivate zombie blocks. Except these, EDBP introduces three registers and an SRAM buffer for voltage threshold adjustment, and a simple comparator for each block to determine whether the block should be deactivated. In the default setting (Table II), EDBP requires 256 comparators incurring approximately 0.0098% of the core chip area (3.37 mm^2, including 0.80 mm^2 data cache and 0.48 mm^2 instruction cache), with the core area size calculated using CACTI [61] This featherweight design not only saves valuable silicon area but also minimizes energy consumption that could arise from extra circuitry.

C. True/False Rate

Figure 6 illustrates all four prediction scenarios and the proportion of blocks that remain unpredicted. From this figure, we find that Cache Decay cannot detect the occurrence of the zombie blocks which thus leads to the high false positive rate. To show this clearly, we call these false negatives caused by zombie blocks as **Missed Prediction (FN)**, which contributes to 68.4% of predicted blocks. In contrast, EDBP can identify these zombie blocks and deactivate them to lower the false negative rate. The experimental results show that EDBP successfully predicts 83.7% of cache blocks while maintaining an accuracy rate of 82.5%.

However, the benefit brought by EDBP is not significant since it can only handle zombie blocks. To allow the energy harvesting systems to be able to address dead blocks as well, EDBP must be integrated with existing dead block predictors. This combined approach not only increases coverage but also maintains a high accuracy of 78.3%, demonstrating the benefits of integrating EDBP with the traditional predictor in energy harvesting systems.

Although coverage and accuracy are important metrics to evaluate EDBP, they do not directly indicate the system's overall energy efficiency and performance. The efficiency and performance are influenced by the duration for which blocks are deactivated; longer deactivation periods typically result in larger energy savings. Even if EDBP can achieve high coverage and accuracy, this does not necessarily mean substantial energy reductions, as many blocks may only be deactivated briefly. To better evaluate EDBP, the following sections will delve into a comprehensive analysis of the overall energy efficiency and performance.

D. Energy Efficiency

Before evaluating the energy efficiency of different schemes, we first present the energy consumption and average power of the baseline system to provide a sense of the actual values (Figure 9). The power and energy consumption vary

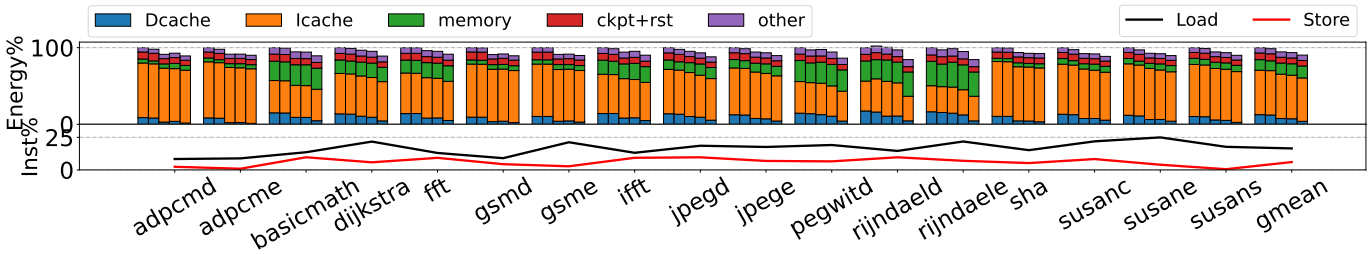

Fig. 7: Energy Breakdown and store/load percentage on the RFHome trace. There are five bars for each application. **From left to right: NVSRAMCache, SDBP, Cache Decay, EDBP, Cache Decay+EDBP.**

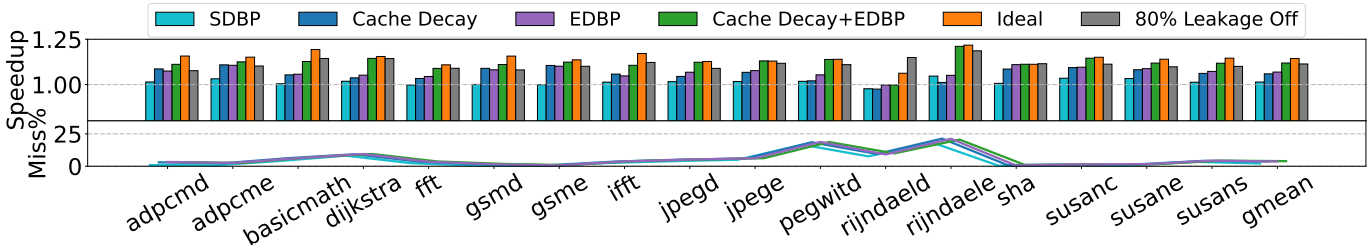

Fig. 8: Performance and cache miss rate on the RFHome trace. Speedup is normalized to NVSRAMCache.

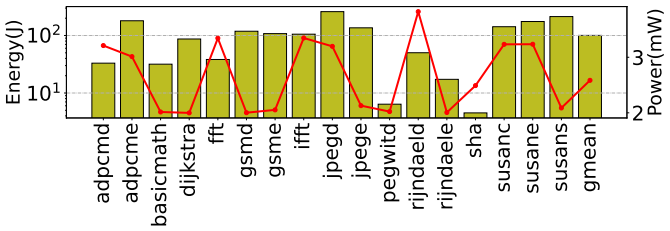

Fig. 9: Absolute average power (red line) and total energy (yellow bar) of NVSRAMCache.

across different applications, averaging 2.58 mW and 100.5 J, respectively.

In Figure 7, we evaluate the energy consumption of all schemes, normalized to NVSRAMCache on the RFHome trace. The total consumption is divided into four parts: cache, memory, checkpoint/restoration, and others (e.g., capacitor leakage and computing). In the baseline architecture, the data and instruction cache are the most energy-consuming components, accounting for 12.5% and 58.0% of the total energy, respectively. The high energy consumption of the instruction cache stems from the costly access operations of the ReRAM cache.

Compared to the baseline, SDBP achieves only around 1.3% of total energy reduction. There are three reasons explaining the poor energy efficiency of SDBP. First, unlike the baseline NVSRAMCache, which only saves dirty cache blocks, SDBP increases the energy requirements for both checkpointing and restoration. Second, if the dead block prediction result in SDBP is false negative, i.e., treating a live block as dead, the energy spent on the checkpoint and restoration of associate data block becomes wasted. Third, even if the prediction is correct, the reuse happens far in the future spanning multiple power cycles, which leads to unnecessary static energy costs in maintaining these blocks active in the cache.

In contrast, EDBP can save 6.5% of total energy compared to the baseline, and together with Cache Decay, achieves an 9.8% energy savings. This promising energy reduction attributes to EDBP detects and deactivates zombie blocks without incurring higher cache misses and associated memory accesses. The experimental results show that the combined approach only raises memory consumption slightly, from 13.8% to 15.6%.

E. Performance

We evaluate the overall performance of EDBP and other schemes across a variety of benchmarks as shown in Figure 8. The results demonstrate that EDBP and Cache Decay consistently outperform the baseline due to their enhanced ability to save energy from cache leakage reduction. Across all benchmarks, Cache Decay and EDBP achieve an average of 5.9% and 6.9% improvement over the baseline architecture respectively. Moreover, the combined approach of EDBP and Cache Decay further improves the performance to 11.9% compared to the baseline. These experimental results prove that EDBP can effectively enhance both the baseline architecture and existing dead block predictors.

Figure 8 also shows the baseline with 80% data cache leakage reduction (i.e., data cache leakage magically reduced by 80% without impacting the cache hit rate) and the maximum performance achievable by dead block predictors (marked as *Ideal* in the figure). With 80% data cache leakage reduction, the baseline performance improves by 11.4%, which is comparable to the performance achieved by the combination of EDBP and Cache Decay. In the ideal scenario, we assume perfect knowledge about which cache blocks will become dead or zombie. With this in mind, we magically turn off these

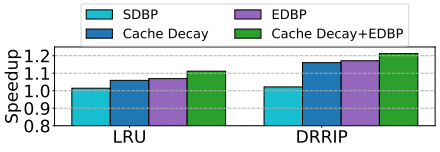

Fig. 10: Cache replacement.

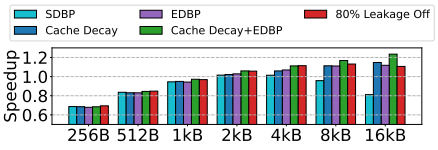

Fig. 11: Cache size.

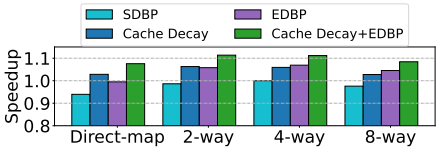

Fig. 12: Cache associativity.

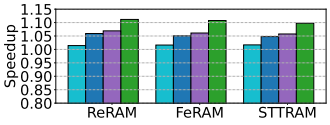

Fig. 13: NVM technology.

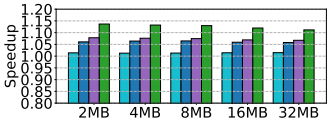

Fig. 14: Memory size.

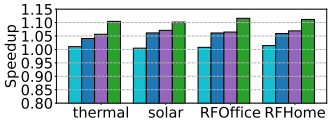

Fig. 15: Energy conditions.

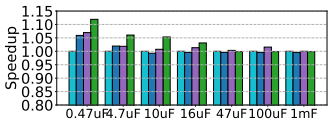

Fig. 16: Capacitor size.

Fig. 17: Sensitivity: simulated on RFHome trace. Normalized to NVSRAMCache with default settings in Table II.

blocks to generate the theoretical optimal performance. The experimental results demonstrate that the ideal performance achieves a 14.4% improvement over the baseline and outperforms Cache Decay by 8.5%. Fortunately, by integrating EDBP to Cache Decay, the performance gap becomes the ideal predictor and the combined approach is much smaller (2.5%). These findings underscore EDBP's effectiveness in boosting the performance of existing dead block predictors, pushing them closer to their theoretical maximum boundary.

F. Cache Miss Rate

Figure 8 also shows the data cache miss rate after applying Cache Decay and EDBP. EDBP alone increases the miss rate from 2.1% to 3.7%, and the combination of EDBP and Cache Decay raises it to 3.9%. Since the benefit brought by EDBP is higher than the cache miss penalty, it ultimately improves overall system energy efficiency and performance.

G. Load and Store Instruction Ratio

Figure 7 illustrates the load/store ratio as a percentage of total committed instructions. This ratio is a key factor influencing the effectiveness of dead block predictors and EDBP. A high load/store ratio results in increased cache accesses, reducing the number of dead and zombie blocks. Conversely, a low ratio indicates fewer cache accesses, providing more opportunities for dead block predictors and EDBP to identify and deactivate dead or zombie blocks. As shown in the figure, the load/store ratios for Mibench and Mediabench are relatively low, enabling dead block predictors and EDBP to achieve strong performance.

H. Sensitivity Analysis

1) Cache Replacement Policies: The performance of EDBP depends on the cache replacement policy's ability to identify suitable blocks for deactivation. Different policies impact EDBP's performance differently. For example, a naive policy like LRU is more likely to select incorrect zombie blocks, resulting in higher cache miss penalties for EDBP. In contrast, a more sophisticated policy like DRRIP reduces cache miss penalties, improving EDBP 's performance. To demonstrate the robustness of EDBP across different cache replacement policies, we evaluated it with both a naive (LRU) and a sophisticated (DRRIP) policy. As shown in Figure 10, the sophisticated policy significantly boosts EDBP's performance (17.1% improvement over the baseline with DRRIP, compared to 6.91% with LRU) by reducing the likelihood of treating live blocks as zombies. Thus, EDBP remains compatible and robust across various cache replacement policies.

2) Cache Size: We evaluate the sensitivity of EDBP to different cache sizes, ranging from 256B to 16KB as presented in Figure 11. The speedup is normalized to NVSRAMCache with 4kB 4-way caches. The results illustrate a clear trend where increasing the cache size from 256B to 4KB leads to substantial performance improvements. The reason is the reduction in cache misses, as a larger cache can buffer more data and thus reduce the frequent accesses to the slower and more energy-consuming NVM. However, when the cache size is further increased to 8kB and 16KB, the performance gains of the baseline degrades by 6.2% and 21.3% respectively. This is mainly caused by the increased leakage power associated with the larger caches, which offsets the benefits of the reduction in cache misses. Fortunately, EDBP reduces this high cache leakage and achieves 11.1% and 11.7% improvement for 8kB and 16kB caches respectively. Moreover, the combined approach further improves performance to 16.8% and 23.4% compared to the baseline using 4kB cache (the tallest bar in Figure 11). Therefore, with the help of EDBP, energy harvesting systems can effectively utilize larger caches to enhance performance, thereby addressing the dilemma outlined in Section II.

3) Cache Associativity: We evaluate the impact of cache associativity on the performance of EDBP by varying the associativity from direct-mapped to 8-way caches as depicted in Figure 12. The speedup is normalized to NVSRAMCache with 4kB 4-way caches. For the direct-mapped caches, EDBP simply sets one voltage threshold that deactivates all blocks. The results show that increasing the associativity from direct-mapped to 2-way yields a noticeable performance boost. For example, the performance of EDBP increases from -0.01% to 5.7% and the performance of the combined approach increases from 7.5% to 11.3%. Further increasing the associativity to a 4-way only provides marginal gains (only 0.6% for EDBP and 0.8% for the combined approach). When further increasing to

8-way, the performance of all schemes starts to decline due to the increased cache access consumption associated with high associativity. The experimental results require us to pick the most suitable associativity (i.e., 4-way) to achieve the performance for energy harvesting systems. Overall, EDBP and the combined approach always demonstrate promising performance across different associativity levels, confirming its adaptability and efficiency in various cache configurations.

4) NVM Technology: NVM technology significantly impacts the efficiency and performance of EDBP and other dead block predictors. Different NVM types' access latencies and energy consumption influence the cache miss penalties. To see the robustness of EDBP, we evaluate the sensitivity of EDBP and Cache Decay to different NVM technologies, including ReRAM, FeRAM, and STTRAM, to understand their impact on performance as shown in Figure 13. ReRAM, with its relatively low latency and energy consumption, incurs low cache miss penalties which thus provides the best performance improvements for the combined approach (11.9% over baseline). On the contrary, STTRAM requires much higher access latency and energy, making the cache miss penalties more expensive. As a result, the combined approach can only help the system to achieve a 9.7% performance gain. Despite these variations, both EDBP and Cache Decay consistently enhance system performance across different NVM technologies.

5) Memory Size: Memory size is another important factor impacting the effectiveness of EDBP. Larger memory sizes come with higher access latencies and increased energy overhead during reads and writes, which can amplify the cache miss penalty of dead block predictors. To assess EDBP's robustness, we evaluate its sensitivity to different memory sizes, as shown in Figure 14. Our experiments show that smaller memory sizes provide better performance for EDBP due to lower cache miss penalties, while larger memory sizes are less favorable. When the memory size increases from 2MB to 32MB, EDBP's speedup decreases from 7.8% to 6.7%. Similarly, the combination of EDBP and Cache Decay sees a performance drop, from 13.6% to 11.1%.

6) Energy Conditions: Energy conditions play a crucial role in determining the performance of energy harvesting systems (Section II). To evaluate the sensitivity of EDBP to varying energy conditions, we test it under four different conditions. The results, depicted in Figure 15, illustrate that EDBP consistently improves the baseline architecture across all energy conditions. In stable and high-energy environments, EDBP has fewer opportunities to reduce leakage as the frequency of power interruptions is low. However, in unstable and low-energy conditions, the frequency of power outages increases so that EDBP has more opportunities to reduce cache leakage and thus shows significant performance improvements over the baseline. For instance, on the thermal trace, EDBP increases the performance gain by 5.6%, while on the RFHome trace, it enhances performance by 6.9%. Additionally, the combined approach shows substantial improvements across various energy conditions, achieving a 10.4% performance increase on the thermal trace and 11.9% on the RFHome trace.

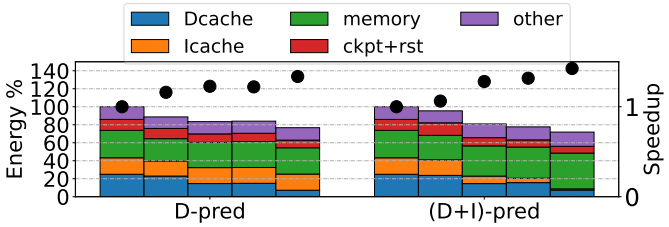

Fig. 18: Energy Breakdown (*bars*) and speedup (*dots*) on the RFHome trace. There are five bars for each design choice on the x-axis. **From left to right: NVSRAMCache, SDBP, Cache Decay, EDBP, Cache Decay+EDBP.** Energy and speedup are normalized to NVSRAMCache.

7) Capacitor Size: The capacitor size directly influences the frequency and duration of power interruptions in energy harvesting systems, which in turn affects overall performance, particularly for weak energy sources like RF. Larger capacitors can store more energy, reducing the frequency of power outages and allowing the system to have longer power cycles. However, larger capacitors also require longer charging times and cause higher leakage currents. Conversely, smaller capacitors have shorter charging times and lower leakage power but make the system more susceptible to power interruptions, as they drain quickly and lead to frequent power outages.

To assess the influence of different capacitor sizes on EDBP's performance, we conducted a comprehensive sensitivity analysis. The results, depicted in Figure 16, show that EDBP enhances the baseline architecture's performance across all tested capacitor sizes. We also find that the effectiveness of EDBP decreases as the capacitor size increases. For instance, when increasing the capacitor size from 0.47 μF to 100 μF, the performance of EDBP also degraded from 7.5% to 1.6%. This outcome aligns with our expectations, as larger capacitors store more energy, supporting longer power cycles and consequently reducing the frequency of power outages. This reduction in power outages diminishes the opportunities for EDBP to effectively reduce cache leakage and enhance performance. Similarly, Cache Decay also experiences diminished performance gains with larger capacitors. This is because the energy stored in larger capacitors allows some smaller applications, such as *fft* and *ifft*, to complete without experiencing power outages. As a result, these applications do not benefit from reduced halting times for recharging, limiting Cache Decay's opportunities to enhance performance through reduced cache leakage.

I. EDBP for Instruction Cache

So far, we have demonstrated EDBP's ability to reduce leakage power in the SRAM data cache. This section explores its effectiveness on the instruction cache. Since the original ReRAM-based instruction cache is not a target for EDBP, we first introduce a new baseline architecture using SRAM for both caches, with the rest of the settings unchanged (Section VI). We then compare the impact of applying EDBP to the data

cache only versus both the data and instruction caches. Figure 18 presents the energy breakdown and speedup for the new baseline and other schemes, with two design choices (applying the dead block predictor to the data cache only or to both caches) shown on the x-axis.

In the new baseline, the data and instruction caches consume 24.9% and 18.3% of the total energy, respectively. Applying EDBP to the data cache reduces its consumption by 10%, and when combined with Cache Decay, this reduction increases to 17.7%. Overall, EDBP alone saves 16.1% of total energy, while the integration with Cache Decay boosts savings to 23.3%. Similarly, when EDBP is applied to both data and instruction caches, the energy consumption of caches is reduced by 22.8%. When integrated with Cache Decay, this reduction increases to 28.4%. In total, EDBP saves 22.4% of energy, and when combined with Cache Decay, achieves 28.2% savings.

Figure 18 also illustrates the performance of the new baseline and other schemes. Enabling EDBP for the data cache alone results in a 22.9% improvement over the new baseline while applying EDBP to both caches achieves a 31.7% performance boost. Additionally, the combination of EDBP and Cache Decay for the data cache increases performance to 33.6%, and their application to both caches further enhances performance to 42.5%. Therefore, EDBP is also effective for the SRAM instruction cache.

VII. Discussion

A. Integration with Other Dead Block Predictors

Although our evaluation only shows the integration of EDBP with Cache Decay, it does not lose generality that EDBP can also seamlessly integrate with other existing predictors [2], [74] to detect zombie blocks caused by power outages and further enhance energy efficiency in energy harvesting systems. The reason is that all these dead block predictors never consider zombie blocks that are caused by power outages in energy harvesting systems.

B. Integration with Other Energy Harvesting Systems

Although the default baseline architecture used in this paper is JIT checkpointing (i.e., NVSRAMCache), EDBP maintains its effectiveness across a variety of energy harvesting systems [3], [4], [21], [29], [36], [37], [56], [65], [66], [68], [71], [73], [76]. No matter in which energy harvesting systems, as long as they have volatile caches, they will anyway suffer performance degradation due to the zombie blocks caused by power outages. EDBP is designed to detect and deactivate these zombie blocks to reduce energy waste and achieve higher performance.

VIII. Limitations

As discussed in Section VI-H6 and VI-H7, EDBP is highly affected by two design configurations. The first one is energy conditions. When the energy condition is stable and sufficient, each power cycle lasts longer, allowing more program execution before a power outage. Consequently, power outages are less frequent for the program, providing fewer opportunities for EDBP to deactivate zombie blocks and enhance performance. In an extreme scenario where the energy supply is infinite, no power outages and zombie blocks will occur, so EDBP never starts to work making it useless to improve performance. The second design configuration is the capacitor size. A large capacitor buffers enough energy to support longer power cycles and diminish the frequency of power outages. As a result, EDBP has fewer opportunities for zombie block deactivation, which thus makes it unable to provide significant benefits to improve performance.

However, it is important to note that the aforementioned scenarios—consistently good energy conditions and large capacitors—are not typical in most energy harvesting systems, especially in RF-based systems [5], [16], [27], [43], [46], [47], [58], [59], [64], [69], [75]. Due to the unstable and unpredictable nature of RF energy sources, energy harvesting systems frequently experience power outages. Moreover, these energy harvesting systems are often designed to be compact with low hardware overhead. Thus, a small capacitor requiring less hardware area is more suitable for these systems [4]. Although EDBP might show reduced performance in the aforementioned scenarios, its design is well-suited to the more common, challenging environments faced by typical energy harvesting systems.

IX. Related Work

Researchers have developed various dead block predictors tailored for different purposes, including cache prefetching, cache replacement, and cache leakage power reduction, aimed at enhancing overall performance and energy efficiency.

Lai et al. [38] pioneered the concept of dead block prediction with their trace-based dead block predictor, known as RefTrace, for the L1 cache. RefTrace identifies dead blocks by comparing the memory reference sequences of each block with recorded traces that lead to dead blocks. If the comparison matches, it indicates that the cache block is dead. Otherwise, RefTrace still treats it as a live block. Achieving this design requires two hardware tables, i.e., one for recording the reference sequence of each block and the other for storing the access sequences associated with dead blocks.

Kharbutli and Solihin subsequently introduced a counting-based dead block predictor [34]. The fundamental idea is to consider a cache block as dead once its access count reaches a certain threshold. To achieve this, each cache block is augmented with a counter that tracks both the number of times the block has been referenced and the program counter (PC) that first missed on the block. When a counter reaches the threshold value, the associated block is predicted to be dead. The challenge of this work lies in determining an appropriate counter threshold for each cache block. A simple approach is to use a fixed threshold for all blocks; however, this may not be optimal, as the access count varies across blocks. To address this issue, the authors propose an adaptive technique that dynamically adjusts the counter threshold to be suitable for each cache block.

Kaxiras et al. introduced a time-based dead block predictor called Cache Decay [32], which considers a cache block as dead if it has not been accessed within a certain time period since its last access. Cache Decay uses a clock counter as a timer for each cache block. When the counter reaches its maximum value, the associated block is treated as dead and deactivated via gate-Vdd [52]. A key challenge here is implementing the clock counter efficiently to avoid high energy consumption that could offset the savings from reduced cache leakage. To this end, the authors utilize a global counter in combination with a 2-bit counter for each cache block. The global counter increments with every clock cycle and signals all 2-bit counters to increment when they reach their maximum. If a 2-bit counter reaches its maximum, the corresponding block is marked as dead. However, if the block is accessed before this, its 2-bit counter is reset. Like the counting-based predictor, Cache Decay also faces the challenge of determining an appropriate time threshold. A straightforward approach is to select a fixed threshold empirically and apply it throughout execution, but this may yield suboptimal energy savings. To overcome this, the authors employ an adaptive technique for dynamically adjusting the threshold for each block.

Adaptive Mode Control (AMC) [74] is a time-based dead block predictor similar to Cache Decay, with dynamically adjustable time intervals for each cache block. To achieve this, AMC monitors extra cache misses that occur due to block deactivation and uses it to guide the time interval adjustment. If the extra cache miss rate is too high, i.e., AMC is too aggressive in deactivating cache blocks, the system must enlarge the time interval allowing more time to check whether a block is a true dead block. On the contrary, if the miss rate is low, AMC will shrink the time interval. For this purpose, AMC keeps the tag array active to track these misses.

Although the aforementioned previous works achieve effective dead block prediction, none of them is directly applicable to energy harvesting systems due to the lack of consideration on the frequent power failure and the resulting zombie blocks—that can easily fool the existing dead block predictors. The beauty of EDBP is that it can serve as a complementary solution to help them take care of zombie blocks with no hassle, thereby improving the performance.

X. Conclusion

This paper found out that existing dead block predictors do not effectively work for energy harvesting systems. That is because their frequent power failure turns many cache blocks into zombie blocks that have no reuse before the failure and thus waste hard-won energy in vain. With that in mind, we proposed EDBP that can detect and deactivate the zombie blocks in a precise and lightweight manner. EDBP works well in combination with the existing dead block predictors, creating a synergistic effect on the reduction of cache leakage energy consumption that accounts for a considerable portion of harvested energy. Consequently, EDBP can significantly improve the energy efficiency and the performance of energy harvesting systems.

Acknowledgements

We appreciate anonymous reviewers and our shepherd for their invaluable comments and constructive feedbacks. We also thank Jianping Zeng for helping with some of the rebuttal response as well as Sikang Sun and Dongho Choi for their contributions to analyzing some existing dead block predictors on our simulator. This work was supported by NSF grants 2001124, 2153749, and 2314681.

References

[1] H. Aantjes, A. Y. Majid, and P. Pawełczak, "A testbed for transiently powered computers," in *arXiv preprint arXiv:1606.07623 (2016)*, 2016.

[2] J. Abella, A. González, X. Vera, and M. F. P. O'Boyle, "Iatac: a smart predictor to turn-off l2 cache lines," *ACM Trans. Archit. Code Optim.*, vol. 2, no. 1, p. 55–77, mar 2005. [Online]. Available: https://doi.org/10.1145/1061267.1061271

[3] A. Abulila, I. E. Hajj, M. Jung, and N. S. Kim, "Asap: architecture support for asynchronous persistence," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 306–319.

[4] M. Alshboul, P. Ramrakhyani, W. Wang, J. Tuck, and Y. Solihin, "Bbb: Simplifying persistent programming using battery-backed buffers," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 111–124.

[5] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," in *IEEE Embedded Systems Letters 7, 1 (2014), 15–18*, 2014.

[6] S. Beeby and N. White, "Energy harvesting for autonomous systems," in *Artech House, Incorporated. https://books.google.fr/books?id=7H9xdFd4sikC*, 2014.

[7] R. G. Belloco, A. G. Macapayad, R.-A. C. O. Calimpusan, and J. T. Dellosa, "Development of an integrated hybrid energy harvesting system for wireless sensor network applications using 180nm cmos process technology," in *2024 IEEE Symposium on Industrial Electronics & Applications (ISIEA)*. IEEE, 2024, pp. 1–6.

[8] A. Bhattacharyya, A. Somashekhar, and J. S. Miguel, "Nvmr: Non-volatile memory renaming for intermittent computing," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1–13. [Online]. Available: https://doi.org/10.1145/3470496.3527413

[9] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[10] J. Bito, R. Bahr, J. G. Hester, S. A. Nauroze, A. Georgiadis, , and M. M. Tentzeris, "A novel solar and electromagnetic energy harvesting system with a 3-d printed package for energy efficient internet-of- things wireless sensors," in *IEEE Transactions on Microwave Theory and Techniques 65, 5 (2017)*, 2017, pp. 1831–1842.

[11] P. Cahill, R. O'Keeffe, N. Jackson, A. Mathewson, , and V. Pakrashi, "Energy-harvesting thermoelectric sensing for unobtrusive water and appliance metering," in *In Proceedings of the 2nd International Workshop on Energy Neutral Sensing Systems, ENSsys '14, Memphis, Tennessee, USA, November 6, 2014. 7–12. https://doi.org/10.1145/2675683.2675692*, 2014.

[12] P. Cahill, R. O'Keeffe, N. Jackson, A. Mathewson, , and V. Pakrashi, "Structural health monitoring of reinforced concrete beam using piezoelectric energy harvesting system," in *In EWSHM-7th European workshop on structural health monitoring*, 2014.

[13] S. Cao and J. Li, "A survey on ambient energy sources and harvesting methods for structural health monitoring applications," in *Advances in Mechanical Engineering 9, 4 (2017)*, 2017.

[14] Q. Cheng, Z. Peng, J. Lin, S. Li, , and F. Wang, "Energy harvesting from human motion for wearable devices," in *10th IEEE International Conference on Nano/Micro Engineered and Molecular Systems*. IEEE, 2015, p. 409–412.

[15] J. Choi, Q. Liu, and C. Jung, "Cospec: Compiler directed speculative intermittent computation," in *In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. 399–412*, 2019.

[16] J. Choi, J. Zeng, D. Lee, C. Min, and C. Jung, "Write-light cache for energy harvesting systems," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ser. ISCA '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3579371.3589098

[17] Y.-W. Chong, W. Ismail, K. Ko, , and C.-Y. Lee, "Energy harvesting for wearable devices: A review," in *IEEE Sensors Journal 19, 20 (2019)*, 2019, p. 9047–9062.

[18] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[19] G. Fang, J. Choi, and C. Jung, "Hybrid power failure recovery for intermittent computing," in *ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, 2024.

[20] T. Galchev, J. McCullagh, R. Peterson, and K. Najafi, "A vibration harvesting system for bridge health monitoring applications," in *In Proc. PowerMEMS*, 2010, p. 179–182.

[21] V. Gogte, S. Diestelhorst, W. Wang, S. Narayanasamy, P. M. Chen, and T. F. Wenisch, "Persistency for synchronization-free regions," *ACM SIGPLAN Notices*, vol. 53, no. 4, pp. 46–61, 2018.

[22] M. Gorlatova, J. Sarik, G. Grebla, M. Cong, I. Kymissis, , and G. Zussman, "Movers and shakers: Kinetic energy harvesting for the internet of things," in *ACM international conference on Measurement and modeling of computer systems*, 2014, p. 407–419.

[23] Y. Gu, Y. Liu, Y. Wang, H. Li, and H. Yang, "Nvpsim: A simulator for architecture explorations of nonvolatile processors," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016, pp. 147–152.

[24] A. Gunti, D. K. Biswas, I. Mahbub, P. R. Adhikari, and R. C. Reid, "Highly efficient rectifier and dc-dc converter designed in 180 nm cmos process for ultra-low frequency energy harvesting applications," in *2020 IEEE 14th Dallas Circuits and Systems Conference (DCAS)*. IEEE, 2020, pp. 1–5.

[25] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *In Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538). IEEE, 3–14*, 2001.

[26] S.-Y. Huang, J. Zeng, X. Deng, S. Wang, A. Sifat, B. Bharmal, J.-B. Huang, R. Williams, H. Zeng, and C. Jung, "Rtailor: Parameterizing soft error resilience for mixed-criticality real-time systems," in *2023 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2023, pp. 344–357.

[27] H. Jayakumar, A. Raha, and V. Raghunathan, "Quickrecall: A low overhead hw/sw approach for enabling computations across power cycles in transiently powered computers," in *In 2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems. IEEE, 330–335*, 2014.

[28] H. Jayakumar, A. Raha, and V. Raghunathan, "Quickrecall: A low overhead hw/sw approach for enabling computations across power cycles in transiently powered computers," in *In VLSI Design. IEEE Computer Society, 330–335*, 2014.

[29] J. Jeong, J. Zeng, and C. Jung, "Capri: Compiler and architecture support for whole-system persistence," in *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, 2022, pp. 71–83.

[30] P. Kamalinejad, C. Mahapatra, Z. Sheng, S. Mirabbasi, V. C. Leung, , and Y. L. Guan, "Wireless energy harvesting for the internet of things," in *IEEE Communications Magazine 53, 6 (2015)*, 2015, p. 102–108.

[31] E. Kamenar, S. Zelenika, D. Blažević, S. Maćešić, G. Gregov, K. Marković, , and V. Glažar, "Harvesting of river flow energy for wireless sensor network technology," in *Microsystem Technologies 22, 7 (2016)*, vol. 22, 2016.

[32] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: exploiting generational behavior to reduce cache leakage power," in *Proceedings 28th Annual International Symposium on Computer Architecture*, 2001, pp. 240–251.

[33] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive wi-fi: Bringing low power to wi-fi transmissions," in *In NSDI, Vol. 16. 151–164.*, 2016.

[34] M. Kharbutli and Y. Solihin, "Counter-based cache replacement and bypassing algorithms," *IEEE Transactions on Computers*, vol. 57, no. 4, pp. 433–447, 2008.

[35] H. Kim, J. Zeng, Q. Liu, M. Abdel-Majeed, J. Lee, and C. Jung, "Compiler-directed soft error resilience for lightweight gpu register file protection," in *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2020, pp. 989–1004.

[36] A. Kolli, V. Gogte, A. Saidi, S. Diestelhorst, P. M. Chen, S. Narayanasamy, and T. F. Wenisch, "Language-level persistency," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 481–493.

[37] A. Kolli, J. Rosen, S. Diestelhorst, A. Saidi, S. Pelley, S. Liu, P. M. Chen, and T. F. Wenisch, "Delegated persist ordering," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–13.

[38] A.-C. Lai, C. Fide, and B. Falsafi, "Dead-block prediction dead-block correlating prefetchers," in *Proceedings 28th Annual International Symposium on Computer Architecture*, 2001, pp. 144–154.

[39] C. Lee, M. Potkonjak, , and W. H. Mangione-Smith, "Mediabench: A tool for evaluating and synthesizing multimedia and communications systems," in *In Proceedings of 30th Annual International Symposium on Microarchitecture*. IEEE, 1997, p. 330–335.

[40] H. G. Lee and N. Chang, "Powering the iot: Storage-less and converter-less energy harvesting," in *In Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific. IEEE, 124–129*, 2015.

[41] W. S. Lee, H. Jayakumar, and V. Raghunathan, "When they are not listening: Harvesting power from idle sensors in embedded systems," in *In Proceeding of the 5th International Green Computing Conference (IGCC)*, 2014.

[42] H. Liu, M. Ferdman, J. Huh, and D. Burger, "Cache bursts: A new approach for eliminating dead blocks and increasing cache efficiency," in *2008 41st IEEE/ACM International Symposium on Microarchitecture*, 2008, pp. 222–233.

[43] Y. Liu, Z. Li, H. Li, Y. Wang, X. Li, K. Ma, S. Li, M.-F. Chang, S. John, Y. Xie, J. Shu, and H. Yang, "Ambient energy harvesting nonvolatile processors: From circuit to system," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[44] Y. Liu, J. Yue, H. Li, Q. Zhao, M. Zhao, C. J. Xue, G. Sun, M.-F. Chang, and H. Yang, "Data backup optimization for nonvolatile sram in energy harvesting sensor nodes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1660–1673, 2017.

[45] G. Lukosevicius, A. R. Arreola, and A. S. Weddell, "Using sleep states to maximize the active time of transient computing systems," in *Proceedings of the Fifth ACM International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, ser. ENSsys'17. New York, NY, USA: Association for Computing Machinery, 2017, p. 31–36. [Online]. Available: https://doi.org/10.1145/3142992.3142998

[46] K. Ma, X. Li, S. Li, Y. Liu, J. J. Sampson, Y. Xie, and V. Narayanan, "Nonvolatile processor architecture exploration for energy-harvesting applications," *IEEE Micro*, vol. 35, no. 5, pp. 32–40, 2015.

[47] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan, "Architecture exploration for ambient energy harvesting nonvolatile processors," in *In 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). IEEE, 526–537*, 2015.

[48] M. Magno and D. Boyle, "Wearable energy harvesting: From body to battery," in *In 2017 12th International Conference on Design and Technology of Integrated Systems In Nanoscale Era (DTIS)*. IEEE, 2017, pp. 1–6.

[49] M. Magno, D. Kneubuhler, P. Mayer, and L. Benini, "Micro kinetic energy harvesting for autonomous wearable devices," in *In 2018 International symposium on power electronics, electrical drives, automation and motion (SPEEDAM)*. IEEE, 2018, p. 105–110.

[50] M. Megahed and T. Anand, "A sub-μw energy harvester architecture with reduced top/bottom plate switching loss achieving 80.66% peak efficiency in 180-nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 5, pp. 1386–1399, 2023.

[51] G. Park, T. Rosing, M. D. Todd, C. R. Farrar, and W. Hodgkiss, "Energy harvesting for structural health monitoring sensor networks," in *Journal of Infrastructure Systems 14, 1 (2008)*, vol. 14, 2008, p. 64–79.

[52] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-vdd: a circuit technique to reduce leakage in deep-submicron cache memories," in *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, ser. ISLPED '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 90–95. [Online]. Available: https://doi.org/10.1145/344166.344526

[53] S. Priya and D. J. Inman, "Energy harvesting technologies," vol. 21. Springer, 2009.

[54] H. Rahmani and A. Babakhani, "A fully integrated electromagnetic energy harvesting circuit with an on-chip antenna for biomedical implants in 180 nm soi cmos," in *2016 IEEE SENSORS*. IEEE, 2016, pp. 1–3.

[55] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on rfid-scale devices," *SIGARCH Comput. Archit. News*, vol. 39, no. 1, p. 159–170, mar 2011. [Online]. Available: https://doi.org/10.1145/1961295.1950386

[56] J. Ren, J. Zhao, S. Khan, J. Choi, Y. Wu, and O. Mutlu, "Thynvm: Enabling software-transparent crash consistency in persistent memory systems," in *Proceedings of the 48th International Symposium on Microarchitecture*, 2015, pp. 672–685.

[57] L. Rizzon, M. Rossi, R. Passerone, and D. Brunelli, "Wireless sensor networks for environmental monitoring powered by microprocessors heat dissipation," in *In Proceedings of the 1st International Workshop on Energy Neutral Sensing Systems (ENSSys '13). ACM, New York, NY, USA, Article 8, 6 pages. https://doi.org/10.1145/2534208.2534216*, 2013.

[58] F. Su, Y. Liu, Y. Wang, and H. Yang, "A ferroelectric nonvolatile processor with 46mus system-level wake-up time and 14mus sleep time for energy harvesting applications," in *IEEE Transactions on Circuits and Systems I: Regular Papers 64, 3 (2016), 596–607*, 2016.

[59] F. Su, K. Ma, X. Li, T. Wu, Y. Liu, and V. Narayanan, "Nonvolatile processors: Why is it trending?" in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 966–971.

[60] W. Sun, T. Tan, Z. Yan, D. Zhao, X. Luo, , and W. Huang, "Energy harvesting from water flow in open channel with macro fiber composite," in *AIP Advances 8, 9 (2018)*, vol. 8, 2018.

[61] D. Tarjan, S. Thoziyoor, and N. Jouppi, "Cacti 4.0," 07 2006.

[62] C. Wang, N. Chang, Y. Kim, S. Park, Y. Liu, H. G. Lee, R. Luo, and H. Yang, "Storage-less and converterless maximum power point tracking of photovoltaic cells for a nonvolatile microprocessor," in *In Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific. 379–384. https://doi.org/10.1109/ASPDAC.2014.6742919*, 2014.

[63] H. Williams, M. Moukarzel, and M. Hicks, "Failure sentinels: Ubiquitous just-in-time intermittent computation via low-cost hardware support for voltage monitoring," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 665–678.

[64] T. Wu, K. Ma, J. Hu, J. Xue, J. Li, X. Shi, H. Yang, and Y. Liu, "Reliable and efficient parallel checkpointing framework for nonvolatile processor with concurrent peripherals," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 1, pp. 228–240, 2023.

[65] Z. Wu, K. Lu, A. Nisbet, W. Zhang, and M. Luján, "Pmthreads: Persistent memory threads harnessing versioned shadow copies," in *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2020, pp. 623–637.

[66] S. Yadalam, N. Shah, X. Yu, and M. Swift, "Asap: A speculative approach to persistence," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 892–907.

[67] C.-W. Yau, T. T.-O. Kwok, C.-U. Lei, , and Y.-K. Kwok, "Energy harvesting in internet of things. in internet of everything," in *IEEE Communications Magazine 53, 6 (2015)*. Springer, 2018, p. 35–79.

[68] J. Zeng, "Compiler and architecture co-design for reliable computing," Ph.D. dissertation, Purdue University, 2024.

[69] J. Zeng, J. Choi, X. Fu, A. P. Shreepathi, D. Lee, C. Min, and C. Jung, "Replaycache: Enabling volatile cachesfor energy harvesting systems," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 170–182. [Online]. Available: https://doi.org/10.1145/3466752.3480102

[70] J. Zeng, S.-Y. Huang, J. Liu, and C. Jung, "Soft error resilience at near-zero cost," in *Proceedings of the 38th ACM International Conference on Supercomputing*, 2024, pp. 176–187.

[71] J. Zeng, J. Jeong, and C. Jung, "Persistent processor architecture," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 1075–1091.

[72] J. Zeng, H. Kim, J. Lee, and C. Jung, "Turnpike: Lightweight soft error resilience for in-order cores," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 654–666.

[73] J. Zeng, T. Zhang, and C. Jung, "Compiler-directed whole-system persistence," in *Proceedings of the 51th Annual International Symposium on Computer Architecture*, 2024.

[74] H. Zhou, M. Toburen, E. Rotenberg, and T. Conte, "Adaptive mode control: a static-power-efficient cache design," in *Proceedings 2001 International Conference on Parallel Architectures and Compilation Techniques*, 2001, pp. 61–70.

[75] Y. Zhou, J. Zeng, J. Jeong, J. Choi, and C. Jung, "Sweepcache: Intermittence-aware cache on the cheap," in *MICRO-56: 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023.

[76] Y. Zhou, J. Zeng, and C. Jung, "Lightwsp: Whole-system persistence on the cheap," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 215–230.