NOVELLA: Nonvolatile Last-Level Cache Bypass for Optimizing Off-Chip Memory Energy

Aritra Bagchi, Ohm Rishabh, and Preeti Ranjan Panda

***Abstract*—Contemporary multiprocessor systems-on-chips (MPSoCs) continue to confront energy-related challenges, primarily originating from off-chip data movements. Nonvolatile memories (NVMs) emerge as a promising solution with their high-storage density and low leakage, yet they suffer from slow and expensive write operations. Writebacks from higher-level caches and responses from off-chip memory create significant contention at the shared nonvolatile last-level cache (LLC), affecting system performance with increased queuing for critical reads. Previous research primarily addresses the performance issues by trying to mitigate contention through the bypassing of NVM writes. Nevertheless, off-chip memory energy, one of the most critical components of system energy, remains unaddressed by state-of-the-art bypass policies. While certain energy components, such as leakage and refresh, depend on system performance, performance-optimizing bypass policies may not ensure energy efficiency. Aggressive bypass decisions aimed only at performance enhancement could degrade cache reuse, potentially outweighing reductions in leakage and refresh energies with the increase in off-chip dynamic energy. While both performance and off-chip memory energy are influenced by both cache contention and reuse, the tradeoffs for achieving optimal performance versus optimal energy are different. We introduce nonvolatile last-level cache bypass for optimizing off-chip memory energy (NOVELLA), a novel bypass policy for the nonvolatile LLC, to optimize off-chip memory energy by exploiting tradeoffs between cache contention and reuse, achieving a balance across different components of the energy. Compared to a naïve no-bypass baseline, while state-of-the-art reuse-aware bypass solutions reduce off-chip memory energy consumption by up to 8%, and a contention- and reuse-aware bypass baseline by 12%, NOVELLA achieves significant energy savings of 21% across diverse SPEC workloads.**

***Index Terms*—Cache bypass, energy-efficient memory systems, last-level cache (LLC), memory, nonvolatile memory (NVM).**

Manuscript received 11 August 2024; accepted 12 August 2024. This article was presented at the International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES) 2024 and appeared as part of the ESWEEK-TCAD Special Issue. This article was recommended by Associate Editor S. Dailey. *(Corresponding author: Aritra Bagchi.)*

Aritra Bagchi is with the Department of Computer Science and Engineering, Indian Institute of Technology Delhi, New Delhi 110016, India (e-mail: aritra.bagchi@cse.iitd.ac.in).

Ohm Rishabh is with the Department of Electrical Engineering, Indian Institute of Technology Delhi, New Delhi 110016, India (e-mail: ohm.v@alumni.iitd.ac.in).

Preeti Ranjan Panda is with the Department of Computer Science and Engineering and the Khosla School of IT, Indian Institute of Technology Delhi, New Delhi 110016, India (e-mail: panda@cse.iitd.ac.in).

Digital Object Identifier 10.1109/TCAD.2024.3446720

I. Introduction

SYSTEM-LEVEL energy efficiency is a pivotal aspect of modern SoC design. Although contemporary multiprocessor systems-on-chips (MPSoCs) integrate several processor cores and accelerators, off-chip data movements remain the energy Achilles heel, responsible for over 60% of the total system energy [1]. Last-level cache (LLC) serves as the final line of defense against expensive off-chip accesses, making the requirement for larger LLCs crucial in today's systems. As conventional technologies, such as SRAM and DRAM, consume excessive leakage power when scaled, nonvolatile memory (NVM) technologies, such as STT-MRAM, PCM, DWM, ReRAM, and FeRAM, have received serious research attention because of their higher-storage densities and low-leakage power. However, NVMs suffer from major limitations. STT-MRAM, which offers the highest endurance among all other NVMs and emerges as one of the most promising alternatives to SRAM in designing future LLCs, still struggles with inefficient write operations, which are 2-5× slower than reads [2], [3].

Fig. 1 illustrates the architectural overview of our MPSoC. The first two levels of caches (L1, L2) are private to each processor core, whereas the LLC is shared among all cores. As fast access speed is a crucial requirement for private caches, they are typically implemented with SRAM. To meet emerging applications' demand for reducing costly off-chip memory accesses through larger LLCs, we target an NVM implementation for the LLC, which offers more density at low leakage. When a core requests data, it first checks its private caches. If the data is not found (cache miss), a *read* request (Arrow ①) is sent to the LLC, where it is enqueued into the *request queue*. If the data is present in the LLC (cache hit), it responds to the corresponding L2 cache with the requested data. Otherwise, the request is sent to the off-chip main memory. Upon fetching the data from the main memory, a response arrives back at the LLC. The LLC controller then creates a copy of the response data and forwards it to the requesting core for maintaining its progress (Arrow ③). The original response is inserted into the *response queue* for later writing (Arrow ④). Apart from responses, evictions from L2 caches generate another source of NVM write operations, known as *writebacks*, which are enqueued into the request queue (Arrow ②). Because NVM writes are $2-5\times$ slower than reads (shown in different colors, Fig. 1), writebacks and responses contend with read requests for the available LLC bandwidth, introducing longer queueing delays

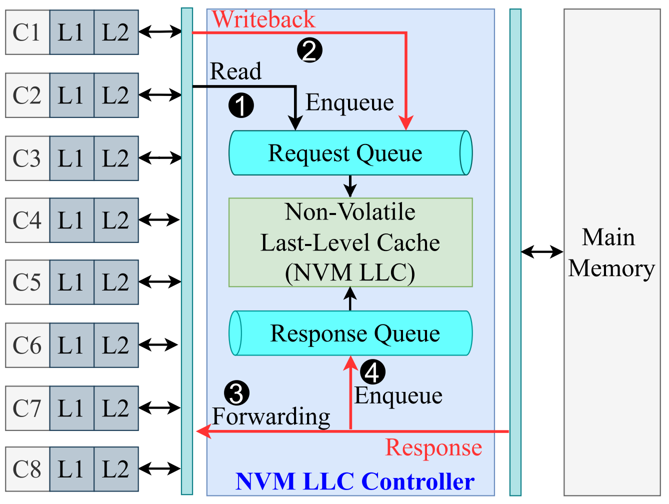

Fig. 1. Overview of system architecture. *Writebacks* from higher-level caches and *responses* from off-chip main memory exacerbate LLC contention, delaying critical *reads*, and affecting overall system performance.

for critical reads, thereby creating performance bottlenecks. Such contention is a major concern even in contemporary MPSoCs with SRAM LLCs [4], [5], [6], and gets aggravated in next-generation MPSoCs with NVM LLCs.

Prior works unanimously focused on improving system performance or reducing NVM cache energy [3], [7], [8], [9], neglecting off-chip memory energy considerations. However, across our real-world workloads, off-chip memory accounts for 64% of the total memory-system energy, with LLC contributing only around 9%. The off-chip memory energy comprises three components: 1) *static*; 2) *dynamic*; and 3) *refresh*. Static and refresh energies, accounting for 60% of the total off-chip memory energy, correlate closely with overall system performance, while the dynamic component is influenced by LLC reuse, which controls the volume of off-chip memory traffic. Prior bypass policies either overemphasize cache reuse [3], [7], [10], [11], or mitigate contention for enhancing system performance at the expense of cache reuse [6]. An excessively aggressive bypass, despite having the potential for enhancing performance, could affect cache reuse and overpower improvements in static and refresh energies with the increase in dynamic memory energy (DynE). Both performance and off-chip memory energy are influenced by cache contention and reuse, but the tradeoffs for optimizing performance versus energy differ. We exploit this tradeoff in NOVELLA, a dynamic cache bypass policy for NVM writes, to mitigate off-chip memory energy bottlenecks and help scale next-generation systems against the memory power wall. In an NVM cache, any write operation creates bottlenecks, so NOVELLA is designed to make bypass decisions for all sources of NVM writes. Our major contributions are summarized as follows.

1) While existing bypass policies for NVM cache focus only on performance and/or cache energy, we, to the best of our knowledge, are the first to primarily target optimizing off-chip memory energy through NVM cache bypass.
2) A balance across different memory energy components is manifested by tradeoffs between the implications of bypass decisions on NVM cache reuse and contention. We investigate these tradeoffs and exploit them in NOVELLA.
3) Energy implications of bypassing different sources of NVM writes vary across different workloads. We identify and leverage such application-specific tradeoffs in NOVELLA.
4) To model the asymmetric implications of NVM reads and writes on shared LLC contention, we enhanced the cache timing model of the gem5 simulator and plan to make an open-source release of the enhancement.

The remainder of this article is organized as follows. Section II summarizes relevant prior works, Section III motivates the key energy tradeoff, Section IV discusses the proposed policy, and Section V presents its hardware implementation. Section VI covers the experimental setup, evaluation, results, and overheads, followed by the conclusion in Section VII.

II. Related Work

Prior works addressed NVM cache performance and energy challenges primarily through hardware-based approaches [3], [8], [12], [13], emphasizing cache bypass techniques to address inefficient NVM writes. Korgaonkar et al. [3] proposed write congestion aware bypass (WCAB), which, based on the average occupancy of the request queue and the liveness of data, bypasses writebacks to mitigate NVM cache contention. WCAB primarily focused on enhancing system performance but also estimated its impact on memory energy, serving as one of our baselines in Section VI. Using sampling predictors, Ahn et al. [7] bypassed various sources of redundant NVM writes to reduce NVM cache energy. Zhang et al. [9] designed an NVM cache bypass technique based on a theoretical model of data reuse statistics. Analytically estimating the benefits of caching data in terms of their impact on access latency, Wang et al. [8] proposed a runtime bypass policy for NVM writes. A selective NVM cache inclusion policy is designed by Cheng et al. [14] to store only a subset of data from higher-level caches for improving NVM cache energy.

Bypass policies have also been explored for conventional caches. Wu et al. [10] proposed a signature-based hit predictor (SHiP), which associates each cache reference with a signature, incrementing a counter on hits and decrementing on evictions without reuse. SHiP bypasses a cache fill if its signature corresponds to a counter of value zero, anticipating distant reuse. Li et al. [11] proposed a bypass policy that tracks reuse distances of cache fills and associated victims, caching data only when the fill has incurred reuse distance less than that of associated victims in the past. Park et al. [15] proposed a bypass first policy (BFP), bypassing fills by default, and caching them later if they are predicted to offer spatial or temporal reuse. Bagchi et al. [6] proposed a response bypass mechanism that dynamically exploits the tradeoff between cache contention and reuse to enhance system throughput, addressing contention concerns overlooked by other policies. These policies are considered baselines in Section VI.

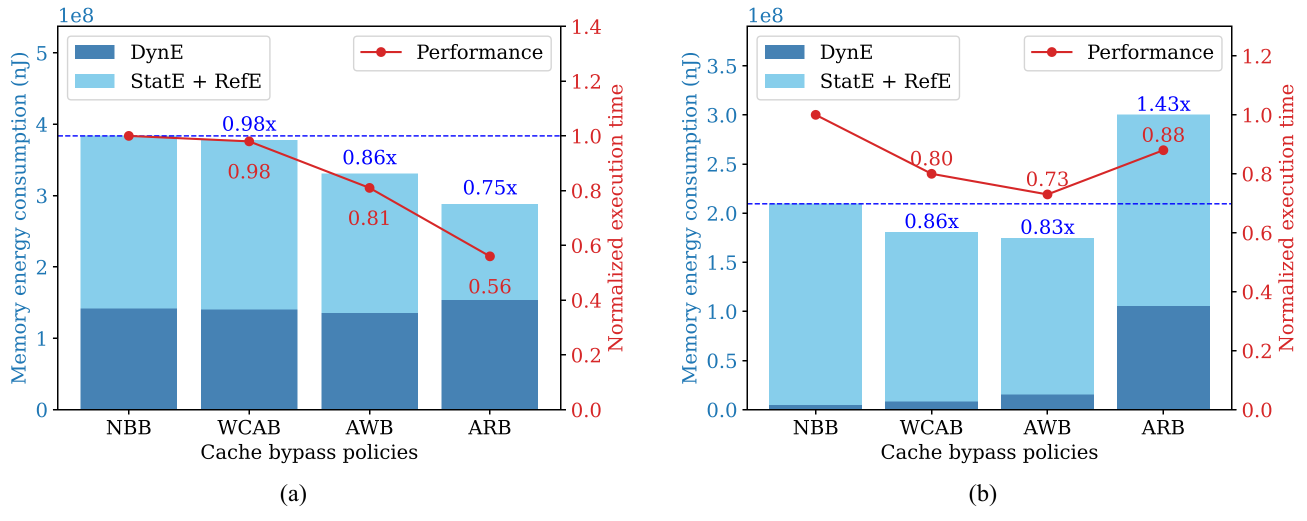

Fig. 2. Motivational example demonstrating off-chip memory energy and performance profiles of two SPEC workloads (from Table III). In (a), performance enhancements lead to overall energy efficiency when LLC reuse (which controls dynamic energy) is minimally impacted. In (b), bypassing NVM writes of certain types degrades LLC reuse significantly, increasing DynE so much that it outweighs reductions in static and refresh energies achieved through performance enhancement. This motivates the need for balancing different off-chip memory energy components to achieve overall energy efficiency.

In hybrid SRAM-NVM caches, write-intensive data are dynamically identified and migrated to SRAM for mitigating NVM write overheads [13], [16]. NVM caches are also implemented with regions of different retention times, while placing data according to the cache access pattern [12], [17], [18]. Zhou et al. [19] proposed a circuit-level optimization to proactively terminate redundant NVM writes. Zero-valued NVM cache data are encoded by Jung et al. [20] to save NVM write energy. To circumvent DRAM scaling issues, high-density NVMs (e.g., PCM) have been explored for main memories [21]. Researchers also explored energy-aware memory controller policies. Wu et al. [22] dynamically migrated memory pages to consolidate shorter idle periods into longer ones, therefore exploiting the low-power states to control the background power consumption. Lai et al. [23] demoted ranks with less critical reads to low-power modes, while prioritizing the service of critical reads for maintaining performance. Energy-aware memory controller policies [22], [23] are orthogonal to our cache bypass-based approach and could be applied in parallel for further energy efficiency.

III. Motivation

For next-generation NVM caches, bypass techniques are shown to be beneficial for improving overall system performance as well as reducing cache energy consumption [3], [7], [8], [9], both of which are synergistic objectives for cache bypass policies. These bypass policies circumvent the performance penalties of slow NVM writes, reducing the cache leakage energy, while also reducing the cache dynamic energy as a direct consequence of bypassing some NVM writes. Nevertheless, prior works did not explore the potential of NVM cache bypass in addressing off-chip memory energy, which is the most challenging system energy bottleneck.

For an LLC, which is shared by an increasing number of cores and accelerators, both contention and reuse are crucial system-level factors controlling the consumption of off-chip memory energy. Bypassing NVM writes aggressively could alleviate LLC contention and reduce the consumption of static and refresh components of off-chip memory energy through improvements in overall system performance. However, aggressive bypass could disturb LLC reuse (or locality) and, if not controlled carefully, could outweigh the energy efficiency achieved through performance improvements. Also, while the latencies for multiple off-chip memory accesses can be hidden (*memory-level parallelism*), the energy overheads cannot be hidden in the same manner as delays. Therefore, an energy-efficient bypass policy for NVM LLC should carefully consider this key tradeoff. Based on profiling of real-world SPEC applications (from the pool of evaluation workloads in Table III, Section VI-A) with a state-of-the-art NVM cache bypass solution [3] and a few baseline strategies, we demonstrate the tradeoff in detail as follows.

Fig. 2 illustrates the trends in different components of off-chip memory energy alongside corresponding trends in system performance measured as the workload execution time. Fig. 2a and b show energy and performance profiles of two SPEC workloads, i.e., mixes 10 and 2, respectively, from Table III, Section VI-A. Energy components are plotted along the left Y-axis (in nanojoules), while the normalized execution time is shown along the right Y-axis. In Fig. 2, the DynE, influenced by overall LLC reuse, is depicted separately in each bar plot, while the static and refresh energies, both dependent on overall execution time, are combined (StatE + RefE). For each mix, results across four different cache bypass policies are presented along the X-axis. While WCAB [3] is the state-of-the-art NVM cache bypass policy, the other three are constructed by us. The aggressive writeback bypass (AWB) bypasses writebacks aggressively, writing all responses. Conversely, aggressive response bypass (ARB) bypasses responses, caching all writebacks. The no-bypass baseline (NBB) is a naïve policy serving as a global baseline. NBB does not apply any bypass, writing all data to

NVM cache. The execution time of each policy is normalized relative to that of NBB.

In Fig. 2(a), we observe that the dynamic off-chip memory energy remains almost the same in NBB and WCAB, whereas AWB and ARB lead to a 4% reduction and a 9% increment in the dynamic off-chip memory energy compared to NBB. As the implications of these bypass strategies on LLC locality are similar, the dynamic energy does not undergo drastic variations. WCAB, AWB, and ARB mitigate NVM LLC contention to varying extents, resulting in 2%, 19%, and 44% improvements in overall execution time, respectively, over NBB. As a result, the static and refresh energies are also reduced almost proportionally, leading to overall improvements in off-chip memory energy consumption by 2%, 14%, and 25% over NBB for WCAB, AWB, and ARB, respectively. In summary, Fig. 2(a) demonstrates a real-world execution scenario where improvements in overall system performance lead to reductions in off-chip memory energy, given the similar impacts of bypass decisions on LLC locality.

Fig. 2(b) presents another scenario where different bypass policies have considerably different implications for LLC locality. For instance, WCAB enhances performance by 20% over NBB, resulting in a 14% reduction in overall off-chip memory energy consumption compared to NBB. Similarly, for AWB, a 27% reduction in the overall execution time leads to a 17% reduction in the consumption of off-chip memory energy. For WCAB and AWB, despite the dynamic component of off-chip memory energy increasing over NBB by 1.71× and 3.25×, respectively, the improvements in system performance successfully translate to overall energy reductions. However, the situation differs significantly for ARB, which, in its attempt to mitigate LLC contention through ARB, disrupts cache locality to such an extent that the dynamic off-chip memory energy escalates by a factor of 21.68× over NBB. This drastic deterioration in dynamic energy outweighs the comparatively smaller reductions in the other two performance-dependent energy components (static and refresh), leading to a 43% increase in overall off-chip memory energy. Fig. 2(a) and (b) emphasize the tradeoffs between different components of off-chip memory energy, with one being dependent on LLC reuse and the other two being correlated to overall system performance. Also, bypassing different sources of NVM writes (writebacks versus responses) explores this tradeoff disproportionately for different workloads. This important observation motivates us to develop an adaptive bypass policy for NVM LLC that strategically exploits this tradeoff and dynamically modulates its emphasis on mitigating LLC contention and maintaining LLC reuse so as to reduce the consumption of off-chip memory energy across diverse workloads.

IV. Proposed Methodology

We introduce nonvolatile LLC bypass for optimizing off-chip memory energy (NOVELLA), which dynamically bypasses NVM writes to reduce off-chip memory energy consumption. NVM writes are significantly slower than reads, exacerbating LLC contention. Bypassing NVM writes could enhance performance by alleviating LLC contention. However, our focus is on reducing off-chip memory energy. By exploiting tradeoffs between LLC contention and reuse for balancing different memory energy components, NOVELLA achieves overall energy efficiency.

A. Key Ideas and Execution Scenarios

We present an overview of NOVELLA by illustrating different execution scenarios and how NOVELLA adapts its bypass decision for different sources of NVM writes across these scenarios. For this purpose, we consider three categories of real-world applications: 1) LLC agnostic (LA); 2) LLC heavy (LH); and 3) memory heavy (MH). While LA applications produce less LLC contention, the other two result in significant contention. However, while LH applications lead to most of their memory accesses being served from LLC (high-LLC access rate with the majority producing hits), MH applications lead to most of their accesses being served from off-chip main memory (high-LLC access rate with the majority producing misses). The key ideas are summarized as follows.

1) When MH applications are co-executed, there is already significant pressure on off-chip memory due to the majority of memory accesses being directed there, causing us to be conservative in bypassing both sources of NVM writes. Although an aggressive bypass could mitigate LLC contention, it could worsen DynE significantly, leading to overall energy inefficiency.
2) The two sources of NVM writes, writebacks and responses, have asymmetric implications for off-chip memory energy. While bypassing responses may increase memory energy by compromising LLC locality and potentially losing future LLC hits, bypassed writebacks not only risk converting future LLC hits into misses but also immediately increase off-chip memory traffic, affecting off-chip memory energy in a more direct way.
3) Responses exhibit a range of reuse patterns, with some being more valuable than others. Certain responses may be deemed *dead* if they are never referenced between their fill and eviction. We prefer caching of more important response data while potentially bypassing the rest. Such selective response writes could help mitigate LLC contention, but without increasing off-chip memory traffic significantly. If the underlying reuse profile is more diverse, selective writes could also enhance cache locality by preventing premature evictions of useful data.
4) For the co-execution of LA applications, where the majority of memory accesses are handled by private caches, a significant portion of responses filled to LLC are not reused from LLC. However, evicted cache lines from private caches (writebacks) stored in LLC present more opportunities for reuse. Therefore, despite the main memory experiencing less pressure, we prefer caching writebacks while selectively caching useful responses.

In Fig. 3, we present four execution scenarios: cases A (co-execution of LH applications) and *B* (co-execution of MH applications), *C* (co-execution of LH and MH applications), and *D* (co-execution of LA applications) in Fig. 3(a)–(d),

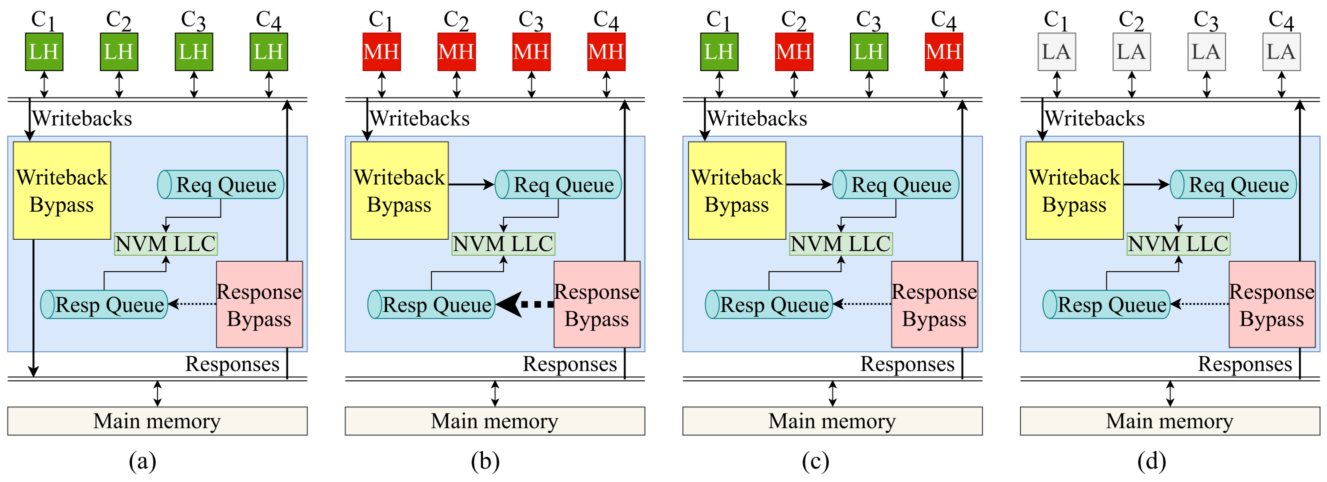

Fig. 3. Illustrative example of NOVELLA, demonstrating NVM write bypass decisions across different execution scenarios. (a) *Case A:* As main memory experiences less pressure, we opt for writeback bypass because of their high contention and low-reuse potential, while selectively caching responses based on their predicted usefulness. (b) *Case B:* We opt for caching of writebacks and conservative bypassing of responses due to the main memory already experiencing significantly high pressure, generated by high volume of off-chip memory traffic. (c) *Case C:* We opt for caching of writebacks because of the off-chip main memory experiencing considerable pressure, while selectively caching responses based on their predicted usefulness. (d) *Case D:* Despite the main memory experiencing low pressure, we prefer caching writebacks because of their higher-reuse potential, while selectively caching responses based on their predicted usefulness.

respectively. For a simpler representation, we do not show private caches and cores separately but highlight LLC controllers, which comprise two different modules for making bypass decisions for two different sources of NVM writes: 1) *Writeback Bypass* and 2) *Response Bypass*. These modules control the flow of incoming NVM writes into their respective controller queues (*Req Queue* and *Resp Queue*), enqueuing a writeback or a response only if its data is decided to be cached. For illustrative purposes, we show a four-core system. In Fig. 3, conditional actions (e.g., enqueuing of responses) are represented by *dotted* arrows, whereas actions that are always performed (e.g., response forwarding, scheduling of pending accesses from the queues into the NVM LLC, enqueuing of writebacks) are shown in *solid* arrows. While the thick dotted arrow indicates frequent actions, thin dotted arrows represent relatively infrequent ones. Building upon the ideas discussed above, we outline NOVELLA's bypass decisions as follows.

1) Case A: In this case [Fig. 3(a)], responses are found to be more crucial than writebacks from the perspective of LLC data reuse. Writebacks, on the other hand, do not impact LLC locality as strongly as responses do, but contribute more toward LLC contention. As previously discussed (Point 1), the main memory experiences relatively less pressure in case A because of LLC filtering the majority of memory accesses from reaching main memory. Therefore, we can afford to bypass writebacks (solid arrow in Fig. 3(a) directing writebacks toward the main memory) to alleviate the NVM LLC contention. We take response reuse into consideration while making bypass decisions for them (Point 3). As highlighted in Point 1, we have the liberty to be selective in caching responses in case A [indicated by a thin dotted arrow leading responses to the response queue in Fig. 3(a)].

2) Case B: In this case [Fig. 3(b)], we experimentally observe that even an aggressive form of writeback bypass (such as AWB, discussed in Section III) could enhance system performance minimally, but ends up increasing off-chip memory energy considerably. Therefore, because of the critical impact of bypassing writebacks on memory energy (Point 2), particularly in this scenario (Point 1), we prefer to cache writebacks, as represented by the solid arrow following writebacks into the LLC request queue in Fig. 3(b). On the other hand, bypassing responses very aggressively helps mitigate NVM LLC contention (and improve performance) but at the expense of degraded LLC reuse, which might outweigh the improvements in static and refresh energies with increased DynE. Therefore, we are conservative also in bypassing responses (Point 1), as indicated by the thick dotted arrow in Fig. 3(b) leading responses to the response queue. We prioritize caching writebacks, following the most conservative approach, due to their asymmetrically stronger impact on memory dynamic energy compared to the other NVM write source (as indicated in Point 2).

3) Case C: In this diverse execution scenario [Fig. 3(c)], we prefer caching writebacks [as shown by the solid arrow leading writebacks to the LLC request queue in Fig. 3(c)] because of their crucial impact on the energy of off-chip main memory (Point 2), which already experiences considerable memory traffic from MH applications (Point 1). As indicated in Point 3, responses exhibit diverse reuse behavior in this case. This creates opportunity for us to be selective in caching only useful responses, as shown in Fig. 3(c) by the thin dotted arrow directing responses to the LLC response queue.

4) Case D: In this scenario [Fig. 3(d)], although the co-execution of LA applications exerts less pressure on off-chip main memory due to limited memory accesses beyond private L2 caches, we prioritize caching writebacks [as shown by the solid arrow leading writebacks to the request queue in Fig. 3(d)] due to their high-LLC reuse potential (Point 4). Meanwhile, responses demonstrate comparatively lower reuse at LLC compared to writebacks, allowing us to be selective in caching only useful response data.

B. NOVELLA Methodology

The NOVELLA controller has two major components:

1) *Writeback Bypass:* Each time an evicted dirty cache line from one of the higher-level caches arrives at NVM LLC, we make a decision regarding whether to write the data into LLC or bypass it. If the data is selected to be written, we insert the writeback into the request queue. Otherwise, we send the writeback to the main memory to store the updated data there.
2) *Response Bypass:* Whenever a response from the main memory reaches NVM LLC, we immediately forward a copy of the data to the core waiting for it. Simultaneously, we make a decision regarding whether to write the data into LLC. If so, we insert the response into the response queue. Otherwise, the response is simply dropped, as the data has already been forwarded.

Based on the ideas discussed in Section IV-A, we make runtime bypass decisions for writebacks and responses. This involves identifying cases A, B, C, and D at runtime. While a lower-LLC access rate indicates case D (co-execution of LA applications), higher-access rates correspond to the other three scenarios, i.e., cases A, B, and C. Among these three cases, a significantly low-LLC miss rate indicates case A (co-execution of LH applications), a much higher-miss rate represents case B (co-execution of MH applications), while intermediate miss rate values correspond to case C (co-execution of both LH and MH applications).

Algorithm 1 describes the steps involved in NOVELLA. While Lines ①- ⑦ capture the steps involved in *Writeback Bypass*, Lines ⑧- ㉑ discuss steps involved in *Response Bypass*. When a writeback W_{Acc} reaches NVM LLC (line ①), we check whether the LLC access rate (AR) is above a threshold (AR_{Th}), and the miss rate (MR) is within a certain lower threshold, i.e., MR_{ThL} (line ②). If so, we identify the execution scenario to be case A, and decide to bypass W_{Acc} (line ⑤). Because writebacks create significant LLC contention in case A (discussed in Section IV-A1), such bypass decisions help alleviate NVM LLC contention. However, in case the data for W_{Acc}, i.e., W_{Blk}, is already present in LLC, we invalidate the data to prevent any future request from fetching stale data from the LLC (line ④). In all other execution scenarios (cases B, *C*, or *D*), we decide to cache the writeback data, and therefore, insert W_{Acc} into the LLC request queue (line ⑦).

When a response W_{Acc} returns from the main memory containing the requested data, we create a copy of it (W'_{Acc}, line ⑨) and forward it to the higher-level cache (line ⑩). This facilitates the requesting processor core to obtain data faster and maintain progress. Meanwhile, we decide whether to write the response or bypass it (Lines ⑪- ㉑). For this decision, we estimate whether the response data is useful. In cases C and D, we assess the usefulness of responses differently than in the other two cases, as a single criterion is not effective across all four cases (more insight in Section VI-B). Again, to identify different execution scenarios, we use both LLC access and miss rates. If the access rate (AR) is low, we identify the scenario as case D (line ⑪) and consult the reuse prediction table (RPT), a small buffer maintaining the reuse history of certain prior responses [11]. Each RPT entry comprises a 3-bit saturating counter. We index into the RPT using hashed instruction program counter (PC) of the incoming response. Initially, all the reuse counters in RPT are reset to 0. If the hashed instruction PC of the incoming response W_{Acc} matches against an existing entry in the RPT, the corresponding counter is incremented. When RPT becomes full and there is a need to allocate a new entry, we evict the oldest entry. So, when the

Algorithm 1: NOVELLA: Nonvolatile LLC Bypass for Optimizing Off-Chip Memory Energy

```
   Input: W_Acc: Write access to NVM LLC
   Input: W_Blk: Target LLC block for NVM write
   Input: MR: NVM LLC miss rate
   Input: AR: NVM LLC access rate
   Input: DeadC_Blk: Number of past dead fills for W_Blk
   Input: FillC_Blk: Number of past fills for W_Blk
   Input: RC: Predicted RC for W_Acc
   Input: MR_ThL and MR_ThH: Lower and higher thresholds
          on NVM LLC miss rate
   Input: AR_Th: Threshold on NVM LLC access rate
   Input: DP_Th: Threshold on W_Blk's dead probability
   // Writeback Bypass
1  if W_Acc is a writeback then
2      if MR < MR_ThL AND AR > AR_Th then
3          if W_Blk ≠ null then
4              Invalidate(W_Blk) // upon a cache
                  hit, invalidate the data
5          Send W_Acc to main memory
6      else
7          Enqueue W_Acc to the request queue
8  else
       // Response Bypass
9      W'_Acc = Copy(W_Acc) // creating a
          forwardable copy of data
10     Forward W'_Acc to higher-level cache
11     if AR ≤ AR_Th then
12         if RC > 0 then
               // data anticipated useful
13             Enqueue W_Acc to the response queue
14     else
15         if MR ≥ MR_ThL AND MR < MR_ThH then
16             if RC > 0 then
                   // data anticipated useful
17                 Enqueue W_Acc to the response queue
18         else
19             DP_Blk = (DeadC_Blk / FillC_Blk)
20             if DP_Blk < DP_Th then
                   // data anticipated useful
21                 Enqueue W_Acc to the response queue
```

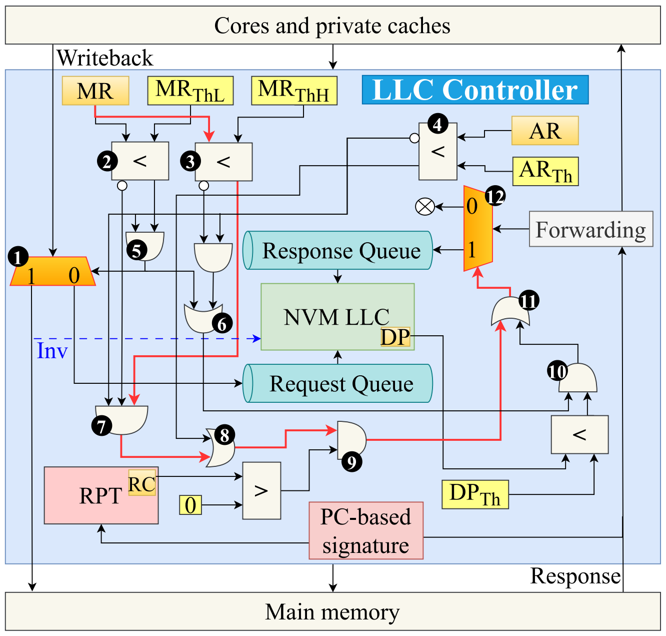

Fig. 4. Hardware implementation of NOVELLA controller. Demultiplexers ① and ⑫ execute the decisions taken for LLC writebacks and responses, respectively, regarding whether to bypass them or insert them into their respective controller queues for LLC writes. The critical path is shown in red.

access rate is low, we decide to bypass W$_{Acc}$ if its predicted reuse (RC) is zero; otherwise, we enqueue W$_{Acc}$ into the response queue for writing data later (line ⑬). If the access rate is high (line ⑭), and the miss rate lies in between MR$_{ThL}$ and MR$_{ThH}$ (line ⑮), we identify the scenario as case C and decide to write responses of nonzero reuse count (RC), as estimated from RPT (line ⑰). When the access rate is high (line ⑭), but the miss rate is below MR$_{ThL}$ (case A) or above MR$_{ThH}$ (case B), we assess the usefulness of the response fill by calculating the current probability of it being dead, i.e., DP$_{Blk}$, based on the fill history. To compute the dead-probability, we divide the number of dead fills so far for the cache block (DeadC$_{Blk}$) by the number of times the block is filled so far (FillC$_{Blk}$), as indicated in line ⑲. A block-fill is marked dead if we observe no reuse for it before its eviction. If the probability exceeds a certain threshold, DP$_{Th}$, we anticipate the fill to be dead and decide to bypass the response. Otherwise (line ⑳), we insert the response into the response queue for a future write (line ㉑). While AR$_{Th}$ is decided from LLC access profiles of our workloads, the size of the RPT, miss rate thresholds (MR$_{ThL}$ and MR$_{ThH}$), and dead probability threshold (DP$_{Th}$) are determined through sensitivity analysis.

V. Controller Implementation

In Fig. 4, we illustrate the implementation of NOVELLA controller. The miss and access rates of NVM LLC (MR, AR) and the other thresholds are available as registers inside the LLC controller. Using gate ⑤, we identify whether the execution scenario resembles case A, and, based on the output of ⑤, demultiplexer ① decides whether to send the writeback to the main memory (bypass) or enqueue it into the request queue (write). To maintain functional correctness, we send a control signal from the bypass path of the demultiplexer ① to the LLC so as to invalidate the existing cache block in case of an LLC hit. In Fig. 4, this control signal, named *Inv*, is represented by a dotted blue line.

The output of AND gate ⑦ indicates case C, while the output of gate ⑥ represents cases A or *B*. Combining the output of gate ⑦ and the output of comparator ④ which indicates the access rate to be within AR$_{Th}$ (i.e., case D), OR gate ⑧ represents scenarios where NOVELLA consults the RPT to estimate the reuse of an incoming response. For that, we use hashed instruction PC of the instruction corresponding to the response cache block as our signature. The bit-width of the signature determines the size of the RPT, with RPT containing 2^K entries if the signature is of K bits. In Section VI-F, we empirically determine K with a sensitivity analysis. If the predicted reuse (RC) is greater than zero, we signal the demultiplexer ⑫, through gates ⑨ and ⑪, to consider caching the response and enqueue it into the response queue. If RC is zero, we signal the demultiplexer ⑫ to drop the response (bypass). For cases A or *B*, indicated by the outcome of gate ⑥, we estimate the dead probability (DP) of the response fill using the cache block's fill history, recorded as additional cache metadata. We signal demultiplexer ⑫, through gates ⑩ and ⑪, to bypass the response if the dead probability is beyond the threshold DP$_{Th}$. If a response is decided to be bypassed, we simply drop the response (represented in Fig. 4 by the symbol ⊗), as the data for every response is forwarded to the cores.

For any incoming NVM LLC write, NOVELLA's decisions can be processed in parallel to the actual service of accesses from the controller queues. Such decisions involve combinational circuits (the critical path is highlighted in Fig. 4), and the only scenario it lies on the critical path is when there are no pending accesses in any of the controller queues and the LLC is idle. However, in our experiments across diverse SPEC workloads, we did not observe this scenario.

VI. Experiments and Results

A. Setup

We use the gem5 simulator for experimental evaluation. Table I captures the details of our underlying system architecture. Essential cache parameters are collected from CACTI and NVSim for 22 nm technology node. To measure DRAM energy consumption, we use DRAMPower [24] tool, which extracts from gem5 performance counters necessary for the energy calculation. We use gem5's system emulation (SE) mode, which offers limited support for SPEC CPU 2017 benchmarks. Therefore, we use SPEC CPU 2006 benchmarks, known to offer memory access characteristics comparable to that of SPEC CPU 2017 benchmarks [25]. We fast-forward through 1 billion instructions, warm up caches for an additional 150 million instructions, and subsequently execute 250 million instructions for any workload. For characterizing

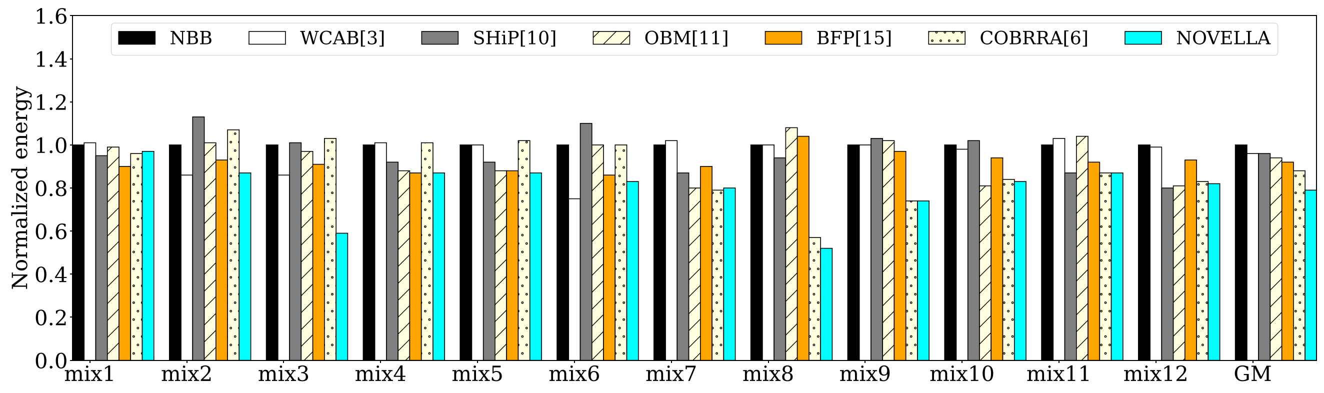

Fig. 5. Comparison of DRAM energy consumption for NOVELLA and other baselines.

TABLE I
System Configurations

System Unit	**Configuration**
CPU	8 X86@2 GHz out-of-order CPU cores
L1 (SRAM)	Private, 32KB L1-D/I, 64B cache line, 8-way set-associative, parallel-access, tag and data latency 1 cycle, MSHR queue size 4
L2 (SRAM)	Private, 256KB, 64B cache line, 8-way set-associative, parallel-access, tag latency 1 cycle, data latency 2 cycles, MSHR queue size 8
L3 (STT-MRAM)	shared, 8MB, SRAM tag and STT-MRAM data arrays, 64B cache line, 16-way set-associative, sequential-access, non-inclusive, tag latency 2 cycles, data read latency 9 cycles, data write latency 25 cycles, MSHR and response queue size 64
Main Memory (DRAM)	DDR3, 1600MHz, 8GB, single channel, 2 ranks/channel, 8 banks/rank, page size 1KB.

TABLE II
SPEC Workload Categorization

Category	**Benchmarks**
L3 Agnostic (LA)	povray, sphinx3, namd, sjeng, astar, gobmk, wrf, bzip2, dealII, gromacs, hmmer, GemsFDTD
L3 Heavy (LH)	gamess, soplex, omnetpp
Memory Heavy (MH)	milc, zeusmp, bwaves, libquantum, leslie3d, mcf, lbm

benchmarks (Table II), we execute them in a standalone environment. While LA applications send fewer accesses to the LLC, LH and MH applications have most of their accesses producing LLC hits and misses, respectively. NOVELLA estimates cache miss rate from performance counters collected for cumulative LLC miss and access counts. While this approach captures the long-term behavior of LLC miss rates, NOVELLA's reuse-aware careful bypass decisions disturb LLC reuse minimally, preserving the sanctity of the classification (e.g., preventing LH workloads from being misclassified as MH). Baselines WCAB [3], SHiP [10], OBM [11], BFP [15], COBRRA [6] and NBB are already discussed in Sections II and III. We use two different sets of workloads: 1) *sensitivity* and 2) *evaluation*. The sensitivity set consists of 32 workloads [26] for determining the values of policy parameters (Section VI-F). Once parameters are finalized, a set of 12 evaluation workloads (Table III) is used for the final evaluation. Both sets are diverse in their compositions, created randomly, with more representatives from LH and MH applications, which are more interesting to us than LA applications that have limited interactions with LLC. We select a different set for determining parameters to demonstrate that parameters tuned on any similar set can effectively work on the evaluation set.

B. Overall DRAM Energy

Fig. 5 illustrates the comparison of total DRAM energy consumption of NOVELLA against six baseline policies across 12 evaluation workloads. The Y-axis shows normalized DRAM energy, calculated by dividing each policy's DRAM energy by that of naïve NBB, with the X-axis showing different workloads. As shown in Fig. 5, while WCAB [3], SHiP [10], OBM [11], BFP [15], and COBRRA [6] reduce the overall DRAM energy consumption by 4%, 4%, 6%, 8%, and 12%, respectively, over NBB, NOVELLA, with a significant energy gain of 21%, outperforms all baselines. While WCAB, SHiP, OBM, and BFP overlook LLC contention, emphasizing LLC reuse, COBRRA, designed for an SRAM LLC, does not address all sources of NVM writes and focuses specifically on system throughput, leading to suboptimal energy gains.

For workloads entirely consisting of LH applications (mixes 2 and 3), bypassing writebacks is more energy-efficient. The AWB policy, discussed in Section III, offers a 28% energy gain for these mixes, while ARB increases DRAM energy by 25%. As shown in Fig. 5, WCAB achieves an average energy gain of 14% over NBB. For these LH workloads, BFP and OBM yield energy gains of 8% and 1%, respectively, while SHiP increases the DRAM energy by 7%. Although COBRRA exploits the tradeoff between LLC contention and reuse, it increases DRAM energy by 5% because it aggressively bypasses responses and caches writebacks. Nevertheless, NOVELLA, with adaptive bypass decisions for the two sources of NVM writes, i.e., writebacks and responses, achieves an energy saving of 28% across these workloads, outperforming state-of-the-art techniques by significant margins.

Across workloads consisting entirely of MH applications (mixes 4 and 5), aggressive bypassing (especially of writebacks) is energy-inefficient. While aggressively bypassing

TABLE III
EVALUATION WORKLOADS

Mix No.	Core 1	Core 2	Core 3	Core 4	Core 5	Core 6	Core 7	Core 8
mix1	wrf	astar	dealII	namd	gromacs	dealII	dealII	sphinx3
mix2	omnetpp	gamess	omnetpp	gamess	omnetpp	soplex	gamess	gamess
mix3	omnetpp	omnetpp	omnetpp	soplex	omnetpp	omnetpp	omnetpp	gamess
mix4	libquantum	mcf	lbm	zeusmp	bwaves	libquantum	leslie3d	bwaves
mix5	leslie3d	mcf	libquantum	bwaves	milc	zeusmp	lbm	bwaves
mix6	dealII	gobmk	astar	astar	dealII	bzip2	gamess	gamess
mix7	hmmer	bwaves	mcf	leslie3d	lbm	zeusmp	leslie3d	leslie3d
mix8	gamess	zeusmp	milc	leslie3d	libquantum	mcf	libquantum	zeusmp
mix9	gamess	gamess	omnetpp	lbm	bwaves	libquantum	mcf	zeusmp
mix10	soplex	gamess	omnetpp	gamess	lbm	zeusmp	zeusmp	mcf
mix11	gamess	omnetpp	omnetpp	soplex	soplex	soplex	bwaves	leslie3d
mix12	mcf	zeusmp	wrf	sjeng	hmmer	omnetpp	omnetpp	soplex

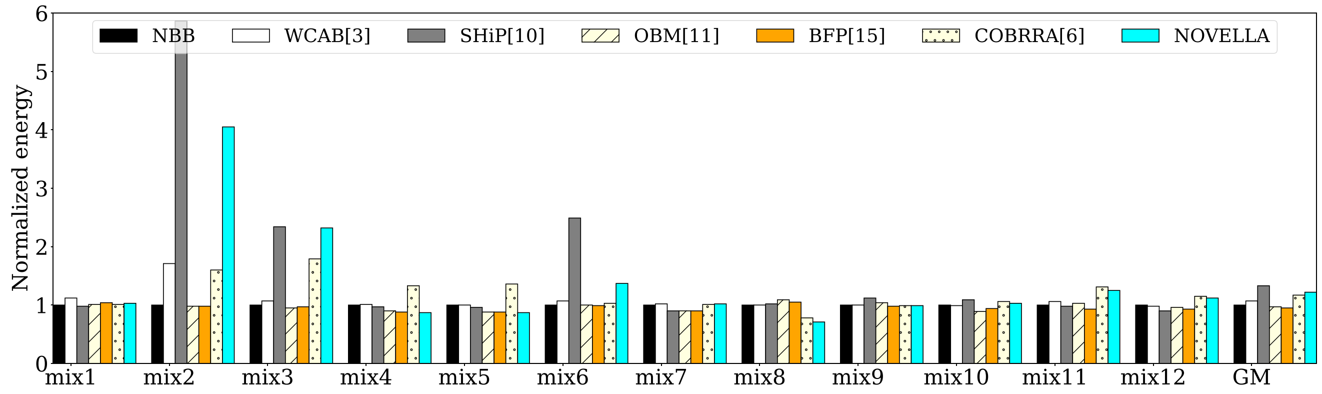

Fig. 6. Comparison of DRAM dynamic energy consumption for NOVELLA and other baselines.

writebacks is found to increase DRAM energy by 8% (AWB of Section III), WCAB, with conservative writeback bypass, shows marginal energy gain. While reuse-emphasizing bypass policies, such as SHiP, OBM, and BFP offer energy gains of 8%, 12%, and 13%, respectively, over NBB, COBRRA increases DRAM energy by 1%, reinforcing the inefficiency of aggressive bypass decisions for these workloads where DRAM is already burdened with LLC misses. However, NOVELLA achieves a 13% reduction in DRAM energy consumption.

For mixes 8-11 co-executing LH and MH applications, NOVELLA offers 27%, 24%, 26%, 25%, and 3% higher-energy gain over baselines WCAB, SHiP, OBM, BFP, and COBRRA, respectively. Selective caching of responses proves energy-efficient for these workloads. For mix1 (LA workload), while WCAB increases DRAM energy by 1%, SHiP, OBM, BFP, COBRRA, and NOVELLA produce energy reductions worth 5%, 1%, 10%, 4%, and 3%, respectively, over NBB.

NOVELLA achieves 9% and 14% higher-energy reductions over bypass strategies using only RPT and only DP, respectively. DP captures a long-term history of response fills, making it suitable for careful bypassing of responses in cases A and B, while the short-term history of fills captured by RPT facilitates the preferred ARB for cases C and D. In cases A and B, the RPT-based reuse criterion tends to bypass responses in an overly aggressive way (82.7% on average), leading to overall energy inefficiency, particularly due to significantly high-DRAM dynamic energy (up to an 8.42× increase over NBB), whereas the DP-based criterion follows a more balanced bypass approach, which is preferred in such scenarios. Conversely, for cases C and D, the DP-based overly conservative bypass (1.44% of responses) is significantly less efficient than NOVELLA's RPT-based bypass approach, which benefits such workloads by bypassing 67% of responses. This is why we use RPT in cases C and D and DP in cases A and B to determine dead response fills.

We evaluated NOVELLA's energy gains across 5-, 6-, 7-, 9-, and 10-core workloads (apart from 8-core workloads), and observed DRAM energy reductions of 8%, 13%, 16%, 21%, 21%, and 19% compared to NBB across 5-, 6-, 7-, 8-, 9-, and 10-core workloads, respectively.

C. DRAM Energy Breakdown

Figs. 6 and 7 show normalized consumption of different DRAM energy components for the policies across evaluation workloads. Fig. 6 shows that WCAB, SHiP, COBRRA, and NOVELLA increase DRAM dynamic energy by 7%, 33%, 17%, and 22%, respectively, while OBM and BFP reduce it by 3% and 5%. Fig. 7 exhibits that WCAB, SHiP, OBM, BFP, COBRRA, and NOVELLA reduce DRAM static and refresh energies by 5%, 9%, 9%, 9%, 27%, and 32%, respectively, compared to NBB.

Despite increasing DRAM dynamic energy by 35% for LH workloads (mixes 2 and 3), WCAB achieves a 14% overall energy gain by reducing the other energy components by 16%. SHiP increases DRAM dynamic energy by 3.7× compared to NBB, while only marginally reducing other energy components, increasing DRAM energy by 7%. For

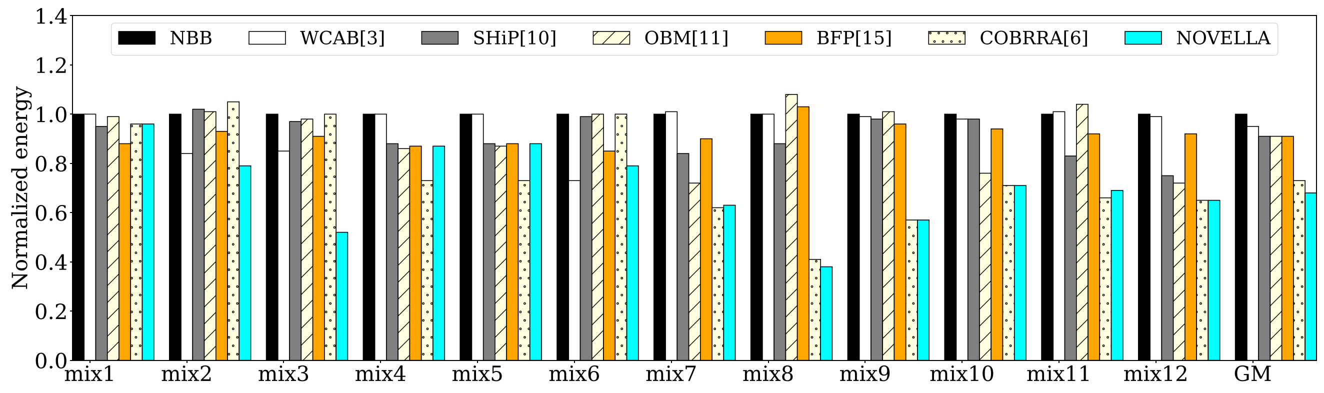

Fig. 7. Comparison of DRAM static and refresh energy consumption for NOVELLA and other baselines.

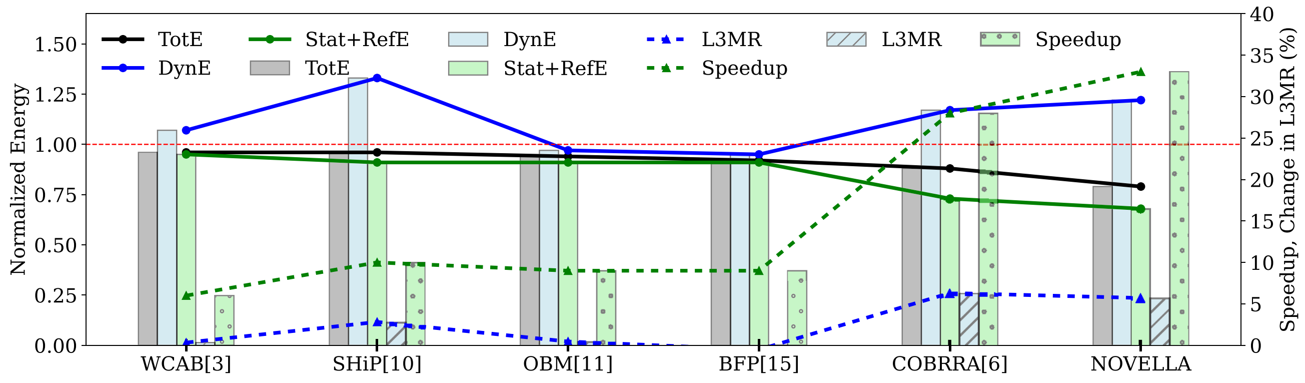

Fig. 8. Summary of key tradeoffs. The LLC miss rate trends align with the DRAM dynamic energy (blue lines), while the performance speedup strongly influences static and refresh energies together (green lines).

these mixes, OBM and BFP reduce DRAM's dynamic energy by 4% and 3%, and combined static and refresh energies by 1% and 8%, respectively, over NBB. For these mixes, responses contribute significantly to LLC reuse. COBRRA's ARB increases dynamic energy by 69%, but fails to reduce static and refresh energies sufficiently, resulting in a 5% increase in total DRAM energy over NBB. NOVELLA's adaptive strategy of bypassing writebacks aggressively and responses carefully increases dynamic energy by 3.07× over NBB, but reduce the other energy components by 36%, resulting in a 28% overall energy gain across these workloads.

For workloads consisting solely of MH applications (mixes 4 and 5), While WCAB minimally impacts different energy components, SHiP improves both dynamic and combined static and refresh energies by 4% and 12%, respectively, over NBB, showing an overall energy savings of 8%. OBM and BFP reduce dynamic energy by 11% and 12% and static and refresh energies by 14% and 13%, achieving overall energy gains of 12% and 13%, respectively, over NBB. COBRRA's ARB increases DRAM dynamic energy by 34%, and, despite a 27% reduction in static and refresh energy components, the overall DRAM energy still increases by 1% compared to NBB. Because of high-DRAM traffic volume, aggressive bypass decisions prove energy inefficient. Surpassing all these energy gains, as shown in Figs. 6 and 7, NOVELLA reduces both DRAM energy components by 13%, emphasizing the efficiency of dead-probability based conservative response bypass for these workloads.

For mixes 8-11, co-executing LH and MH applications, while WCAB shows marginal effects (within ±1%) on different energy components, SHiP increases DRAM dynamic energy by 5% but achieves a 4% overall energy saving with an 8% reduction in static and refresh energies. OBM and BFP, with minimal impact on dynamic energy and 4% reductions in combined static and refresh energies, offer overall energy gains of 2% and 3%, respectively. Caching responses very selectively proves energy efficient for these mixes. While COBRRA increases dynamic energy by 2% and reduces other energies by 42%, with an overall energy gain of 25%, NOVELLA, with its RPT-based bypassing of dead response fills, improves dynamic and static energy components by 2% and 43%, respectively, achieving an overall energy gain of 27% over NBB.

D. Summary of Key Trade-Offs

Fig. 8 summarizes the key tradeoff between reuse-dependent (dynamic energy) and performance-dependent components (static and refresh energies) of DRAM energy. While the left Y-axis shows normalized consumption of various DRAM energy components, the right Y-axis displays percentage improvements in execution time and percentage increases in NVM LLC miss rate. Different baseline policies are represented along the X-axis, with naïve NBB serving as the global baseline. WCAB, SHiP, OBM, BFP, COBRRA, and NOVELLA result in normalized static and refresh energies (combined) of 0.95, 0.91, 0.91, 0.91, 0.73, and 0.68, respectively (solid green trend line, Fig. 8). This strongly correlates with the performance speedups of 6%, 10%, 9%, 9%, 28%, and 33%, respectively (dotted green trend line, Fig. 8). The correlation is high, because of these energy components being almost proportional to the overall execution time. NOVELLA reduces average access delays at DRAM, LLC, L2, and L1 caches by 13%, 55%, 43%, and 43%, respectively, compared to NBB. Since accesses are served in parallel at different memory levels, these delays are inherently parallel. WCAB, SHiP, OBM, BFP, COBRRA, and NOVELLA result in normalized DRAM dynamic energy of 1.07, 1.33, 0.97, 0.95, 1.17, 1.22 (solid blue trend line in Fig. 8), correlating with the trend observed in increases in LLC miss rate of 0.33%, 2.82%, 0.46%, −0.52%, 6.31%, and 5.71% by WCAB, SHiP, OBM, BFP, COBRRA, and NOVELLA, respectively, over NBB (dotted blue line in Fig. 8).

The trend in overall DRAM energy is determined by the trends within its energy components. Therefore, in Fig. 8, the solid black line lies in between the solid blue and green lines. Because the combined static and refresh energy is more dominant than the dynamic energy, the overall energy trend aligns more closely with the trend in these two components. However, performance efficiency alone is not sufficient for achieving energy efficiency. Our comprehensive experimental analysis demonstrates that neither reuse-aware prior bypass policies, such as WCAB, SHiP, OBM, and BFP, nor contention- and reuse-aware policies, such as COBRRA, are the most energy-efficient, underscoring the effectiveness of NOVELLA.

E. LLC and Memory-System Energy

Across our workloads, DRAM, LLC, L2, and L1 caches consume 63.59%, 8.48%, 4.92%, and 23.01% of the whole memory-system energy, respectively. NOVELLA consistently reduces the static energy components of DRAM and all levels of caches by mitigating NVM LLC contention and improving execution time. Performance improvement reduces LLC leakage energy by 33%, and bypassing of expensive NVM writes reduces LLC dynamic energy by 41% compared to NBB, resulting in an overall LLC energy reduction of 34% over NBB. While NOVELLA reduces energy consumption of whole memory hierarchy by 20% over NBB, our optimizations are independent of processor energy consumption, and therefore, can be applied together with core energy optimization techniques for a system-wide energy efficiency.

F. Parameter Selection

Parameters of NOVELLA are determined through experiments across 32 sensitivity workloads, discussed as follows:

1) Miss Rate Thresholds: MR$_{ThL}$ is varied in {0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5}, resulting in energy gains of 8.6%, 11.2%, 11.6%, 8.8%, 8.9%, 7.9%, and 2.4%, respectively. Higher values, such as 0.8 and 1.0, increase the memory energy by 2.4% and 6.1%, respectively, due to AWB. Therefore, we set MR$_{ThL}$ = 0.1 which leads to the highest-energy reduction. MR$_{ThH}$ is varied in the range 0.3-1 with a gap of 0.05 in between, resulting in the energy gain to vary within 5%–12%. We select MR$_{ThH}$ = 0.8 as it produces the highest-energy gain.

2) Dead Probability Threshold: We observe the energy gain to be 10.1%, 9.5%, 11.6%, 11.3%, 11.2%, and 9.2%, respectively, for DP$_{Th}$'s value of 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6. We opt for DP$_{Th}$ = 0.3 as it leads to the maximum energy gain.

3) Reuse Predictor Table Size: We experiment with the signature width (K) being 7, 8, 9, 10, 11, 12, 13, and 14 LSBs of the instruction PC. RPTs of 128, 256, 512, 1024, 2048, 4096, 8192, and 16384 entries result in energy gain of 5%, 5%, 6%, 6%, 4%, 12%, 13%, and 12%, respectively. We choose RPT's size to be 4096 in order to reduce hardware overhead, without significantly sacrificing energy gain.

G. Overhead

NOVELLA's decisions do not lie on the critical path because they are evaluated in parallel with the service of accesses from LLC controller queues. NOVELLA's energy and area overheads are estimated using Synopsys technology library, CACTI, and NVSim for 22nm technology. 9-bit fill and 6-bit dead counters are added to each LLC line as additional metadata, increasing tag array size from 720 KB to 960 KB. NOVELLA's storage (flip-flops, RPT) and logic units incur an area overhead of 0.0792 mm^2, which is 3% of total LLC area (2.64 mm^2). Across our workloads, DRAM consumes 0.297 J of energy, with NOVELLA adding an overhead of 0.0035 J, which is 1.18% of the total DRAM energy.

VII. CONCLUSION

Addressing energy bottlenecks related to off-chip memory accesses is crucial for sustainable computing. NOVELLA reduces off-chip memory energy by exploiting tradeoffs between LLC contention and reuse through NVM write bypass. Across SPEC workloads, NOVELLA achieves 21% and 10.23% higher-energy-savings compared to a no-bypass solution and a state-of-the-art bypass solution, respectively. We show that performance-optimal aggressive bypass policies are not energy-optimal. In the future, we plan to investigate a coordinated approach for energy-efficiency, involving both LLC and DRAM controllers. We also plan to study the problem in the presence of both demand and prefetch traffic.

REFERENCES

[1] A. Boroumand et al., "Google workloads for consumer devices: Mitigating data movement bottlenecks," in *Proc. 23rd Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2018, pp. 316–331.

[2] A. Salahvarzi, M. Khosroanjam, A. M. H. Monazzah, H. Beitollahi, U. Y. Ogras, and M. Fazeli, "WiSE: When learning assists resolving STT-MRAM efficiency challenges," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 1, pp. 43–55, Jan.–Mar. 2023.

[3] K. Korgaonkar et al., "Density tradeoffs of non-volatile memory as a replacement for SRAM based last level cache," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2018, pp. 315–327.

[4] S. Tiwari, S. Tuli, I. Ahmad, A. Agarwal, P. R. Panda, and S. Subramoney, "REAL: REquest arbitration in last level caches," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 6, pp. 1–24, 2019.

[5] G. Modi, A. Bagchi, N. Jindal, A. Mandal, and P. R. Panda, "CABARRE: Request response arbitration for shared cache management," *ACM Trans. Embedded Comput. Syst.*, vol. 22, no. 5S, pp. 1–24, 2023.

[6] A. Bagchi, D. Joshi, and P. R. Panda, "COBRRA: Contention-aware cache bypass with request-response arbitration," *ACM Trans. Embedded Comput. Syst.*, vol. 23, no. 1, pp. 1–30, 2024.

[7] J. Ahn, S. Yoo, and K. Choi, "DASCA: Dead write prediction assisted STT-RAM cache architecture," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2014, pp. 25–36.

[8] J. Wang, X. Dong, and Y. Xie, "OAP: An obstruction-aware cache management policy for STT-RAM last-level caches," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2013, pp. 847–852.

[9] C. Zhang, G. Sun, P. Li, T. Wang, D. Niu, and Y. Chen, "SBAC: A statistics based cache bypassing method for asymmetric-access caches," in *Proc. Int. Symp. Low Power Electron. Design*, 2014, pp. 345–350.

[10] C.-J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely Jr., and J. Emer, "SHiP: Signature-based hit predictor for high performance caching," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2011, pp. 430–441.

[11] L. Li, D. Tong, Z. Xie, J. Lu, and X. Cheng, "Optimal bypass monitor for high performance last-level caches," in *Proc. 21st Int. Conf. Parallel Archit. Compilation Techn.*, 2012, pp. 315–324.

[12] K. Kuan and T. Adegbija, "HALLS: An energy-efficient highly adaptable last level STT-RAM cache for multicore systems," *IEEE Trans. Comput.*, vol. 68, no. 11, pp. 1623–1634, Nov. 2019.

[13] J. Ahn, S. Yoo, and K. Choi, "Write intensity prediction for energy-efficient non-volatile caches," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 2013, pp. 223–228.

[14] H.-Y. Cheng et al., "LAP: Loop-block aware inclusion properties for energy-efficient asymmetric last level caches," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 103–114, 2016.

[15] J. J. K. Park, Y. Park, and S. Mahlke, "A bypass first policy for energy-efficient last level caches," in *Proc. Int. Conf. Embedded Comput. Syst. Archit. Model. Simulat. (SAMOS)*, 2016, pp. 63–70.

[16] Z. Wang, D. A. Jiménez, C. Xu, G. Sun, and Y. Xie, "Adaptive placement and migration policy for an STT-RAM-based hybrid cache," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2014, pp. 13–24.

[17] K. Kuan and T. Adegbija, "LARS: Logically adaptable retention time STT-RAM cache for embedded systems," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2018, pp. 461–466.

[18] M. Baranwal, U. Chugh, S. Dalal, S. Agarwal, and H. K. Kapoor, "DAMUS: Dynamic allocation based on write frequency in multi-retention STT-RAM based last level caches," in *Proc. 22nd Int. Symp. Qual. Electron. Design (ISQED)*, 2021, pp. 469–475.

[19] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "Energy reduction for STT-RAM using early write termination," in *Proc. Int. Conf. Comput.-Aided Design*, 2009, pp. 264–268.

[20] J. Jung, Y. Nakata, M. Yoshimoto, and H. Kawaguchi, "Energy-efficient spin-transfer torque RAM cache exploiting additional all-zero-data flags," in *Proc. Int. Symp. Qual. Electron. design (ISQED)*, 2013, pp. 216–222.

[21] D. Shin, H. Jang, K. Oh, and J. W. Lee, "An energy-efficient dram cache architecture for mobile platforms with PCM-based main memory," *ACM Trans. Embedded Comput. Syst.*, vol. 21, no. 1, pp. 1–22, 2022.

[22] D. Wu, B. He, X. Tang, J. Xu, and M. Guo, "RAMZzz: Rank-aware DRAM power management with dynamic migrations and demotions," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2012, pp. 1–11.

[23] C.-Y. Lai, G.-Y. Pan, H.-K. Kuo, and J.-Y. Jou, "A read-write aware DRAM scheduling for power reduction in multi-core systems," in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2014, pp. 604–609.

[24] K. Chandrasekar et al., "DRAMPower: Open-source DRAM power & energy estimation tool," Nov. 2022. [Online]. Available: https://github.com/tukl-msd/DRAMPower

[25] A. Limaye and T. Adegbija, "A workload characterization of the SPEC CPU2017 benchmark suite," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2018, pp. 149–158.

[26] A. Bagchi, O. Rishabh, and P. R. Panda, "NOVELLA: Non-volatile last-level cache bypass for optimizing off-chip memory energy." Jul. 2024. [Online]. Available: https://github.com/marg-tools/NOVELLA