Microarchitectural Exploration of STT-MRAM Last-level Cache Parameters for Energy-efficient Devices

TOMMASO MARINELLI, JOSÉ IGNACIO GÓMEZ PÉREZ, and CHRISTIAN TENLLADO,
Universidad Complutense de Madrid, Spain
MANU KOMALAN, MOHIT GUPTA, and FRANCKY CATTHOOR, IMEC v.z.w., Belgium

As the technology scaling advances, limitations of traditional memories in terms of density and energy become more evident. Modern caches occupy a large part of a CPU physical size and high static leakage poses a limit to the overall efficiency of the systems, including IoT/edge devices. Several alternatives to CMOS SRAM memories have been studied during the past few decades, some of which already represent a viable replacement for different levels of the cache hierarchy. One of the most promising technologies is the spin-transfer torque magnetic RAM (STT-MRAM), due to its small basic cell design, almost absent static current and non-volatility as an added value. However, nothing comes for free, and designers will have to deal with other limitations, such as the higher latencies and dynamic energy consumption for write operations compared to reads. The goal of this work is to explore several microarchitectural parameters that may overcome some of those drawbacks when using STT-MRAM as last-level cache (LLC) in embedded devices. Such parameters include: number of cache banks, number of miss status handling registers (MSHRs) and write buffer entries, presence of hardware prefetchers. We show that an effective tuning of those parameters may virtually remove any performance loss while saving more than 60% of the LLC energy on average. The analysis is then extended comparing the energy results from calibrated technology models with data obtained with freely available tools, highlighting the importance of using accurate models for architectural exploration.

CCS Concepts: • **Hardware → Spintronics and magnetic technologies**; • **Computer systems organization → Embedded systems;**

Additional Key Words and Phrases: Energy efficiency, gem5, last-level cache, nvsim, spec cpu2017, stt-mram

ACM Reference format:
Tommaso Marinelli, José Ignacio Gómez Pérez, Christian Tenllado, Manu Komalan, Mohit Gupta, and Francky Catthoor. 2022. Microarchitectural Exploration of STT-MRAM Last-level Cache Parameters for Energy-efficient Devices. *ACM Trans. Embedd. Comput. Syst.* 21, 1, Article 3 (January 2022), 20 pages.
https://doi.org/10.1145/3490391

1 INTRODUCTION

Spin-transfer torque magnetic RAM (STT-MRAM) has drawn the attention of computer architects in the past decade. The promise of good-enough performance, low energy, and high

This work has been supported by the European Commission (Regional Development Fund), the Spanish Ministry of Science and Innovation, and the Community of Madrid, under grants RTI2018-093684-B-I00 and S2018/TCS-4423.
Authors' addresses: T. Marinelli, J. I. G. Pérez, and C. Tenllado, Universidad Complutense de Madrid, Avenida de Séneca, 2, Madrid, Spain, 28040; emails: {tommarin, jigomez, tenllado}@ucm.es; M. Komalan, M. Gupta, and F. Catthoor, IMEC v.z.w., Remisebosweg 1, Leuven, Belgium, 3001; emails: {manu.perumkunnil, mohit.gupta, francky.catthoor}@imec.be.

1539-9087/2022/01-ART3 $15.00
https://doi.org/10.1145/3490391

integration capacity was too good to let it leave. Moreover, SRAM is always in the spotlight due to its strongly growing leakage while technology scaling progresses.

At the time of writing, STT-MRAM is likely the more mature of the existing (and promising) non-volatile memory technologies. Nonetheless, it has not yet walked the full path to replace or at least co-exist with SRAM. Despite strong improvements in STT-MRAM endurance, even the optimistic lifetime of $\sim 10^{16}$ cycles [2] may not be enough to fulfill typical L1 cache requirements. Moreover, access latency remains a fundamental stopper for STT-MRAM to enter the **high-performance computing (HPC)** segment even in L2 cache layers. According to literature, there is a $\times 2$–$\times 5$ write performance penalty for STT-MRAM compared to SRAM in a 5nm technology [20].

Energy-wise, however, STT-MRAM has shown to be potentially superior to SRAM. Leakage current is virtually zero in STT-MRAM cells, though there is still some contribution from the peripherals. As reported in a work from Komalan et al. [14], STT-MRAM memories scale better than SRAM regarding dynamic energy, although energy per write access remains higher in STT-MRAM. Overall, the combined dynamic/static energy reduction makes STT-MRAM a perfect candidate for large caches in energy-sensitive systems, like IoT sensor nodes and edge devices, where the duty cycle of the last levels of the memory hierarchy is expected to be lower than in server contexts. Furthermore, the increased area efficiency allows to almost double the amount of memory for the same area, potentially reducing the number of cache misses and expensive writebacks.

In this work, we provide a feasibility study and performance comparison of systems with an STT-MRAM **last-level cache** (**LLC**, typically L2 in the IoT/edge domain) with respect to traditional ones, modeling and testing different CPU architectures for a set of representative benchmarks. The final goal is to quantify the performance impact of this technology change and identify bottleneck patterns in the execution sequence.

The analysis is then extended, observing how different microarchitectural choices impact the technology performance. In particular the focus is on the role of cache banks, write buffer entries, **miss status handling registers (MSHRs)**, and the hardware prefetcher. Other than performance, energy breakdown information is also evaluated to measure the benefits of STT-MRAM compared to SRAM in different configurations. We emphasize the importance of tuning the energy model, since the output of architectural exploration may be significantly different when the energy/performance models are not accurate enough. We explore different configurations of NVSim [7], the most popular STT-MRAM performance, energy and area estimation tool, and propose some minor changes in its cell configuration to come up with a model close to commercial non-disclosable, highly accurate models.

This study shows that adopting STT-MRAM as LLC brings an average energy saving of 68% compared to the SRAM equivalent, with a worst-case performance penalization below 8% on average. If we also consider the area reduction, estimated to be between 30% and 50%, then the STT-MRAM makes itself an excellent choice for most applications, especially in constrained devices where energy is key.

The primary contributions of this article can be summarized as:

(1) An in-depth analysis of the cost-benefit tradeoff obtained by using STT-MRAM technology in LLC instead of a high-performance SRAM, including microarchitectural tuning.
(2) An analysis of the contribution of each cache event to the increase in memory access latency for STT-MRAM caches. We explore the impact of several microarchitecture features to the cache contention.
(3) Tuning of the NVSim energy models to reproduce the main trends of accurate non-public models. We show certain NVSim configurations that may be used for meaningful architectural explorations in contexts similar to the one here assumed.

2 RELATED WORK

STT-MRAM has attracted considerable attention, since it is the most promising logic compatible non-volatile memory that is suitable for advanced logic nodes (28nm and beyond), in terms of endurance, speed, and power. Low-power and high-density properties, however, come at the cost of large write latency. Previous work has focused on either directly reducing this latency or minimizing its impact.

An important body of work has traded off write latency with retention lifetime. Jog et al. [12] proposed an application-driven methodology to determine the optimal retention-time, thus increasing write speed without refreshing overhead. Chang et al. [5] also applied the same tradeoff in their comparison between SRAM, eDRAM, and STT-MRAM technologies for L3 (LLC) caches. A similar approach was employed by Smullen et al. [23], who explored the possibility of lowering the retention time of LLC cache banks to obtain faster writes, at the cost of extra leakage. Khoshavi et al. [13] split the STT-MRAM LLC in two arrays: one mostly read-only with higher retention time and the second one with reduced data retention time. Kuan et al. [17] proposed a highly adaptable last level STT-RAM cache (HALLS) that allows the LLC configurations and retention time to be adapted to applications runtime execution requirements in multi-core environments.

Hybrid SRAM-MRAM caches are also being explored to minimize the high latency impact on performance. Komalan et al. [15] explored the feasibility of STT-MRAM as first-level data cache (L1D), extending the MSHRs to effectively become a pseudo victim-buffer, hiding the increased read and write latencies. However, endurance may prevent the choice of STT-MRAM in layers too close to the core. Our decision to focus on the LLC for the current work was made taking this consideration into account. Different mechanisms were proposed to drive write-intensive traffic to SRAM banks while exploiting STT-MRAM banks in LLC for lowering overall energy consumption [6, 27]. However, converting portions of the LLC into SRAM results in an overall lower capacity than a pure STT-MRAM-based LLC.

Other than reducing the latency itself, previous work seeks to reduce the number of writes to LLC, thus minimizing contention. Wang et al. [26] presented OAP, a methodology to detect contention and prevent writes (either write-backs or write-fills) to happen. The authors mention the bank contention problem but do not further analyze the sources. Other works from Ahn et al. [1] and Korgaonkar et al. [16] involved selective LLC bypassing to reduce write congestion. These previous works target the problem we explore in this article, but we believe that a thorough performance and energy co-exploration for a broad representative set of benchmarks is still missing.

A work from Hameed and Tahoori [10] considered STT-MRAM as an alternative to DRAM for large LLC in high-performance systems, introducing modifications to the row buffer to reduce the number of conflicts and counterbalance the increased write latency and energy. Their row buffer proposals were updated in, a more recent article [9].

Non-volatile magnetic memories have shown to be appealing also at the opposite side of the application spectrum. Patrigeon et al. [19] designed and analyzed an STT-MRAM memory for ultra-low power microcontrollers, as a replacement for code memory—traditionally flash-based—or both code and data memories.

Most of previous works rely on NVSim [7] to obtain their energy/performance model, but the actual numbers used in each work vary significantly: In some cases, write and read energy are similar, while in others write energy is 20× larger than the read counterpart. STT-MRAM area efficiency is sometimes too optimistic, leading to assume that capacity can be increased by a factor 4 (compared to SRAM) in a similar area, as reported in a work from Zhan et al. [28]. STT-MRAM cell is smaller than SRAM, but the impact of periphery will decrease that factor to a maximum of 2 in real designs. Also, leakage power observes large variations across literature, although trends

are more consistent, with the STT-MRAM leakage being around 10× smaller than the SRAM counterpart.

A study of the effect of bank contention in STT-MRAM and SRAM Last-Level Caches has already been performed in a work from Evenblij et al. [8]. The work is to some extent similar to the one we are presenting, with some key differences. The microarchitectural exploration we propose is deeper and not limited to the cache banks. We include additional information concerning the utilization and tuning of publicly available tools to generate a realistic STT-MRAM energy model. The benchmark suite used for our experiments better reflects the needs of modern computer systems.

Finally, STT-MRAM is not the only emerging memory that is attracting the interest of researchers and industries. For example, **spin-orbit torque MRAM (SOT-MRAM)**, which is based on the same physical principles as STT-MRAM using magnetic tunnel junctions for memorization, has better characteristics in terms of write latency/energy and reliability, with an almost negligible overhead in area. Oboril et al. [18] simulated the utilization of this memory technology in different levels of cache, making comparisons with SRAM and STT-MRAM and highlighting the advantages of hybrid SRAM-SOT or fully SOT-MRAM solutions. They also used NVSim to extract memory parameters, tweaking the MRAM basic cell models with values from real silicon data. In our experiments with NVSim, on the contrary, we decided to stay as close as possible to the provided cells even for STT-MRAM, since we mainly care about validating the trends with respect to a modern fabrication process. Furthermore, we decided to leave SOT-MRAM analysis for future work and focus on the STT-MRAM technology, since it has a higher degree of maturity.

3 EXPLORATION AND SIMULATION FRAMEWORK

This work is mainly an exploratory analysis of the impact of STT-MRAM technology replacing SRAM for the L2 cache in energy constrained devices typically found in the IoT/edge computing context (where, if present, it has the role of LLC). Therefore, we first explain the experimental framework before we dive into the analysis of the obtained results.

3.1 Memory Models

One of the most valuable assets of this work is the use of accurate memory models for the experiments. This is especially true for STT-MRAM given the diversity of performance and energy values found in literature. An accurate model is key to enable valid architectural exploration.

The STT-MRAM cache model employs the design outlined in the work published on ISCAS 2017 by Komalan et al. [14], based on the high-performance **perpendicular magnetic tunnel junction (p-MTJ)** stacks detailed by Van Beek et al. [25]. The MTJ pillar diameter is 40nm, with a **tunnel magnetoresistance (TMR)** of 150% and an intrinsic **resistance-area (RA)** product of 10.9 $\Omega\mu m^2$. The STT-MRAM array is designed in the 28nm technology node. The access transistor is planar NMOS. A butterfly architecture is chosen for the layout of the memory macro. While the 1T-1MTJ MRAM bit-cell (260 nm × 220 nm; optimal for LLCs) array occupies only 33.3% of the area of a conventional SRAM bit-cell, the complete STT-MRAM design occupies around 50% of an SRAM design of similar capacity due to having a larger periphery for the same amount of banks. The standby power of an STT-MRAM cell is close to zero, and the standby leakage power of an STT-MRAM cache is solely from the peripheral circuits in the data array, the tag array, and cache logic circuits. An aspect ratio of 1:1 or 1:2 (rows:columns) is followed for the design. V_{dd} supply voltage is 0.9 V.

We adopt a behavioral time-dependent and current-controlled STT-MRAM model based on a macro-spin model that becomes increasingly relevant at the smaller MTJ dimensions considered in our design. The switching dynamics is dominated by the incubation time here. Both

Table 1. Simulated Hardware Models

Model	Frequency	L1I		L1D		L2	
		Size	Assoc.	Size	Assoc.	Size	Assoc.
In-order	1.4 GHz	16 kB	2	16 kB	4	256 kB	16
Out-of-order	2.1 GHz	32 kB	2	32 kB	2	1 MB	16

precessional and thermal switching regimes are modeled. The TMR module in the model is geometry- and bias-dependent. This, along with the Voltage and Cycling-dependent breakdown, is modeled analytically. The model is calibrated based on measurements from the fabricated pMTJ devices in a custom characterization array. Our design is fully custom and designed in the Cadence Virtuoso [11] environment. The memory characterization is carried out via specific scripts to extract timing and power information. It is extended to a form a "pseudo-compiler," wherein the schematics for varying memory capacities can be generated.

The SRAM cache is based on a commercial SRAM compiler for a *High Performance Single Port Ultra High Density Leakage Control* SRAM, built in a 28nm *HPM High-K Metal Gate P-Optional Vt/Cell Std Vt* CMOS Process. The tag array design, cache controller logic circuits, and buffers are kept identical for both the SRAM and STT-MRAM caches to ensure a fair comparison. The access to tags and data is serialized, and the number of banks in the tag array is the same as in the data array.

SRAM and STT-MRAM power gating information is added manually based on the simulations output. STT-MRAM specific power gating implementations are present at the macro level and data bank level. These can be activated using SRAM-like power gating controls in the system via pins for *Memory Enable*, *Light Sleep*, *Deep Sleep*, *Shut-Down*, and *Read-Write Margin*.

Power states can operate differently in SRAM and STT-MRAM, considering also the non-volatility property of this one. In the *Light Sleep* mode, for example, power gating is applied to the decoders in both designs, but while it is possible to completely shut down the array in STT-MRAM, this is unfeasible in SRAM without having loss of data, so this mode only involves array source-biasing for the latter.

For this work, we assume that the LLC operates on *Normal* (active) and *Light Sleep* (standby) modes only. We are also neglecting the energy and latency overhead caused by power states transitions, however, a fair treatment to both memory technologies is guaranteed.

3.2 Simulation Environment

The gem5 architectural simulator [3] has been used to perform all the tests presented in this article. Our setup consists of two ARM64-based CPU models: one in-order and one out-of-order, configured similarly to ARM Cortex A53 and Cortex A15, respectively. The latter is a 32-bit CPU, but we imagined a 64-bit variant of it to make use of a gem5 CPU model from previous work [8]. The A53 model is instead based on the ARM **high-performance in-order (HPI)** model provided with the software itself and slightly tuned with information from real hardware. We have modified gem5 to support asymmetric read/write operations to memory and accurate cache banking.[1]

The memory subsystem in each module has been configured to include an STT-MRAM-based L2 cache (which, as mentioned above, is also the LLC). Table 1 shows some details of the simulated architectures, i.e., operating frequency and caches configuration.

[1] https://gitlabce.dacya.ucm.es/tommarin/gem5-cb.

Table 2. LLC Latencies, in Cycles

Model	SRAM		STT-MRAM typical		STT-MRAM worst	
	Read	Write	Read	Write	Read	Write
In-order	8	8	8	14	14	28
Out-of-order	12	12	12	21	21	40

Table 3. Selected Benchmarks (*train* Set)

Benchmark	Type	Domain	Intervals	Clusters
602.gcc (*200.c*)	INT	GNU C compiler	1,012	19
605.mcf	INT	Route planning	1,349	24
607.cactusBSSN	FP	Physics: relativity	1,547	23
623.xalancbmk	INT	XML to HTML	2,555	14
625.x264	INT	Video compression	3,736	19
628.pop2	FP	Ocean modeling	6,459	24
638.imagick	FP	Image manipulation	3,027	11
641.leela	INT	AI: Monte Carlo	3,958	18
649.fotonik3d	FP	Electromagnetics	1,719	15
654.roms	FP	Ocean modeling	12,017	22

We simulate three different LLC technologies for each architecture: *SRAM* (our baseline), *STT-MRAM typical* (expected STT-MRAM performance and energy values), and *STT-MRAM worst* (a pessimistic model assuming extreme influence from process variation and temperature). Table 2 enumerates the different read/write latencies for each of the target caches. Note that, in all cases, we consider an SRAM-based tag array, so overheads only come from accesses to the data array.

3.3 Benchmark Set

The selection of the program set to put under test is a critical part in any analysis work, as architectural and technology modifications can have a different impact on different application domains and platforms. We have selected a subset of the SPEC CPU2017 benchmark suite [4], trying to remain representative of a wide set of workloads, from extremely memory intensive to more balanced loads.

We started from literature to select an appropriate subset of benchmarks to analyze. A work from Song et al. [24] showed that it is possible to group the benchmarks according to their similarity and retain the general suite characteristics and areas coverage even with a minor number of applications. For this study, we have made sure to include in the set some programs that stress the memory hierarchy (especially the L2) as well as some less cache-dependent ones, to analyze the impact of our architectural changes over different kind of workloads. That is indeed crucial, because sensor/edge devices span a wide range of memory related behaviors. Specifically, the selection of a balanced mix has been based on the analysis performed by Singh and Awasthi [22]. According to literature, some of the benchmarks have a large working set and exhibit a high miss rate in the L1 and L2 caches (*649.fotonik3d_s*, *654.roms_s*), some others have a small working set (*638.imagick_s*), while others have a mix of small and big ones according to the execution phase (*623.xalancbmk_s*). There are also differences in terms of instructions executed, with some of them having a

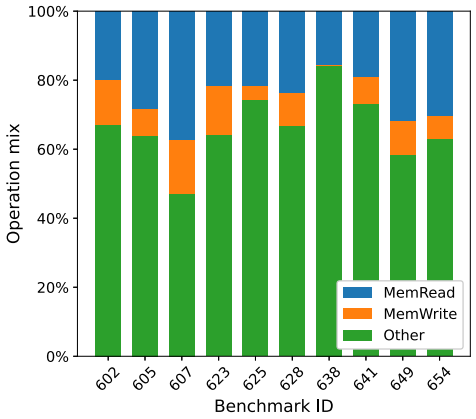


Fig. 1. Instructions classification.

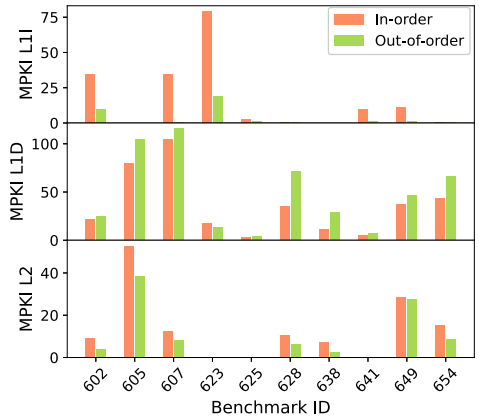


Fig. 2. Cache MPKI.

preponderance of arithmetic and logic operations (*628.pop2_s*) and others with more memory operations (*605.mcf_s*, *607.cactuBSSN_s*).

SPEC CPU2017 benchmarks are provided with three different input datasets: *test*, *train,* and *ref*, in order of complexity. We decided to target the *test* and *train* datasets for practicality, but results are shown only for the latter, since the conclusions are not different. For the same reason, in case of multiple subsets, only one of them has been selected.

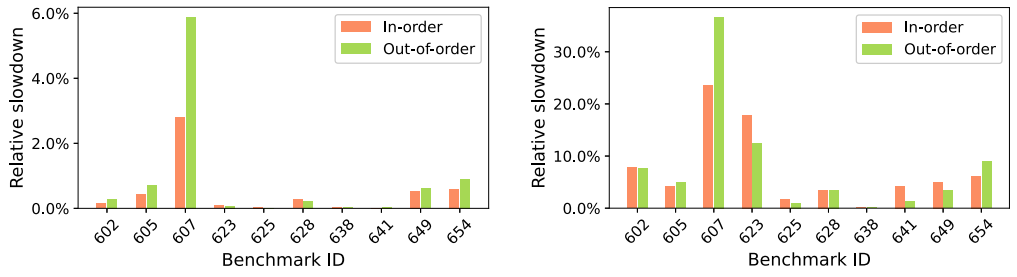
Finally, we have extracted representative execution phases from each selected benchmark using the SimPoint utility [21] to help reduce the simulation time. Information about the benchmarks choice and their simulation points can be found in Table 3. Each interval corresponds to one hundred million instructions, thus the number of intervals gives an estimation of the total execution time. Out of this number, a maximum of 30 clusters has been extracted for each benchmark. The selected clusters have then been simulated with gem5 independently. Following the SimPoint methodology, the final metrics have been obtained as a weighted average of data extracted from the selected clusters. It is important to mention, though, that some benchmarks exhibit huge variations across execution phases.

The benchmarks in their aggregated form have been characterized in the following way: Figure 1 shows the instruction mix of the different benchmarks. *607.cactusBSSN_s* is the most memory intensive: load/store operations represent more than 50% of total instructions. However, *638.imagick_s* is the most CPU-intensive with only 10% of its operations targeting memory, many of which are reads. Figure 2 represents the memory behavior of the selected benchmarks, depicting the cache **misses per kilo instructions (MPKI)** for each of them. Again, we can see that the selected set displays a rich variety of scenarios, from the higher miss rate of *607.cactusBSSN_s* in both L1 and L2, to the nearly optimal behavior of *641.leela_s*.

4 PERFORMANCE ANALYSIS

4.1 Relative Slowdown

Figure 3 shows the relative slowdown of the two proposed STT-MRAM LLC configurations compared to a fully SRAM memory hierarchy. From the data, we can draw a series of conclusions. While in the worst-case scenario the average execution time can be up to 40% higher than the corresponding pure SRAM configuration, the typical case lies on quite low values and never exceeds 10% of slowdown. We remind that in the typical case the read latency is the same as SRAM, only the write latency is higher, which means that the slowdown is entirely attributable to the



(a) STT-MRAM typical case

(b) STT-MRAM worst case

Fig. 3. Relative slowdown per benchmark (eight banks).

increase of the latter. The benchmarks also behave differently according to the CPU type. The out-of-order core presents, on average, a higher slowdown compared to the in-order one, despite out-of-order cores are more able to hide long memory latencies. The higher **instruction-level parallelism (ILP)** achieved stresses more the memory system, which aggravates cache contention in the slower STT-MRAM LLC. Most benchmarks shown in Figure 2 exhibit a higher L1D cache miss ratio (measured in MPKI) in the larger out-of-order cache (32 kB) compared to the smaller in-order counterpart (16 kB). Even the different associativity (4-way in the in-order core vs. 2-way in out-of-order core) does not explain those higher miss rates. The explanation lies in the behavior of the MSHRs: The out-of-order L1D receives many more accesses, most of them hits. The higher CPU frequency, combined with the out-of-order capabilities, increases the density of accesses in L1D, which in turn increases the number of secondary misses (the target word is not in L1D cache, but a previous miss targeting the same cache block is already in progress). The MSHRs work as an extremely efficient filter to L2 accesses.

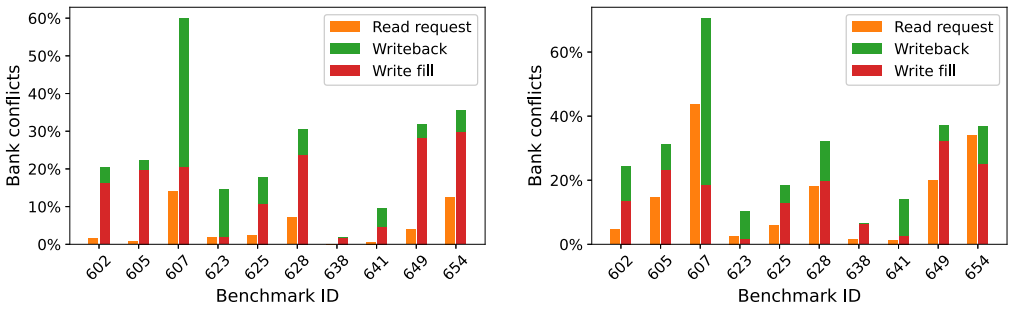
The benchmark that shows the highest relative slowdown is *607.cactuBSSN_s*, a relativity physics equation solver. Among the selected workloads, it is the one that has the highest percentage of memory operations (see Figure 1) and the highest memory bandwidth [22]. Furthermore, it presents frequent misses on the L1D but a low miss rate on the L2, indicating a working set compatible with the L2 size (see Figure 2).

However, the image manipulation benchmark *638.imagick_s* seems to be completely unaffected by the change, which can be explained by the limited number of memory operations, especially writes (see Figure 1). *623.xalancbmk* is almost not affected when assuming the typical values for STT-MRAM, but it suffers a severe slowdown when the pessimistic values are assumed: The extra read latency enormously hits its performance.

While certain behaviors may be easily explained from the benchmarks profile shown in previous figures, some others need further investigations, which leads us to dig into the details of internal cache mechanisms.

4.2 Origin of Conflicts in LLC

We have modified *gem5* to account for the number of cache bank conflicts and their origin. We define bank conflict as the event that happens each time a bank is targeted by a new request while it is busy performing a previous operation. The new request has to wait for the completion of the previous one to be executed. L1 misses following a previous L2 update or high write buffers occupancy may lead to increased delays in STT-MRAM-based caches. Regarding the source of those conflicts, we distinguish three possible scenarios: A bank in the LLC could be busy because of a read operation, a writeback from a previous cache level, or a write fill from main memory.



(a) In-order (b) Out-of-order

Fig. 4. Percentage and categorization of bank conflicts (STT-MRAM worst case, eight banks).

Collected data about the nature of LLC bank conflicts are represented in Figure 4, for an eight-banks configuration. The red/green bars depict the percentage of write operations that generated a conflict, broken down to its source (writeback or write fill). The orange bar depicts the percentage of read operations that generated a conflict. Only the STT-MRAM worst-case configuration is shown, but differences between the three configurations are not significant with this amount of banks. Rather than the pure number of conflicts, it is the impact of increased latencies that affects performance.

607.cactusBSSN is the most severely affected benchmark, and its slowdown is clearly linked to the high percentage of write conflicts. It is also relevant to note that it is the only benchmark where writebacks are generating more conflicts than write fills: The larger traffic between L1 and L2 caches is stalling the pipeline more often. *623.xalanbcmk* displays another relevant behavior: The impact of typical STT-MRAM is negligible, but if shifted from its nominal values it badly hits performance: L1 cache misses are more often in the critical path and cannot be effectively hidden, so the extra L2 read latency rockets the slowdown. The moderately high percentage of bank conflicts of *628.pop2* is not translated into a significant performance penalty. Its DL1 cache behavior is similar to *649.fotonik.3d* and *654.roms*, as shown in Figures 1 and 2, but it has less L2 cache misses and the required memory bandwidth is lower and more stable (see Reference [22]), which could explain the reduction in negative impact. *638.imagick_s*, which has virtually no slowdown, exhibits a low percentage of conflicts.

As expected, there is a general growth of the number of conflicts switching from an in-order to an out-of-order model, but this trend is less evident when considering a pessimistic worst-case STT-MRAM. Out-of-order cores are more capable to hide long memory latencies by issuing independent instructions while waiting from data to reach the pipeline but, however, they also increase the pressure on the memory system. Low-power IoT devices often rely on simpler in-order cores, but edge devices claim for more and more computational power, so both architectures are relevant for this context.

4.3 Effect of Cache Banks

Increasing the number of cache banks allows for higher memory parallelism without including extra ports. Several requests can be served simultaneously as long as they target different banks. While increasing the number of banks is known to improve memory performance, in this context it is relevant to analyze whether STT-MRAM can benefit even more than SRAM from the expected drop in number of conflicts.

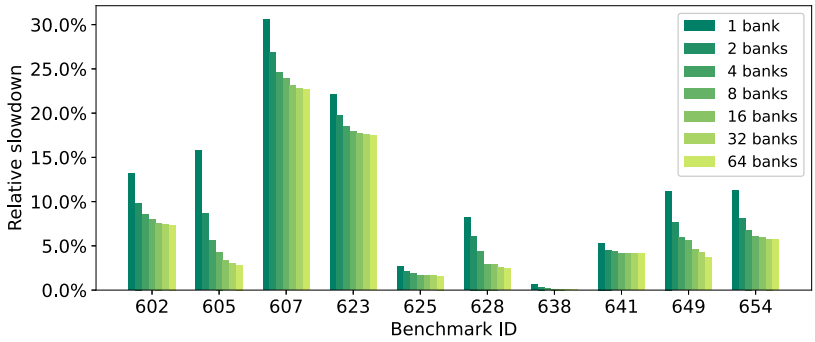


Fig. 5. Relative slowdown varying the number of banks (In-order, STT-MRAM worst case).

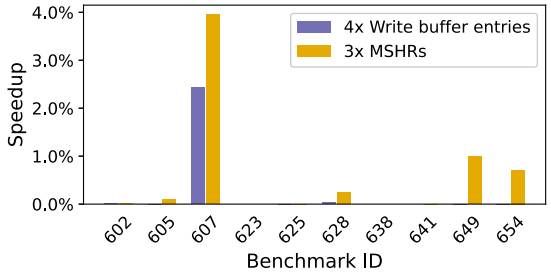


Fig. 6. Effect of L1D write buffer and MSHRs increase (Out-of-order, STT-MRAM worst case, eight banks).

In Figure 5, we illustrate the relative slowdown of a sample configuration for a number of banks between 1 and 64. As expected, the STT-MRAM penalization tends to decrease when more banks are used. Variations between benchmarks are notable, both in terms of maximum-minimum difference and rapidity to saturate. The benchmark that exhibits the highest slowdown difference is *605.mcf_s*. Even if not shown in any plot, the number of LLC conflicts in this workload has a fast decrease when adding more banks and most of them are caused by write fills, which can well explain the behavior.

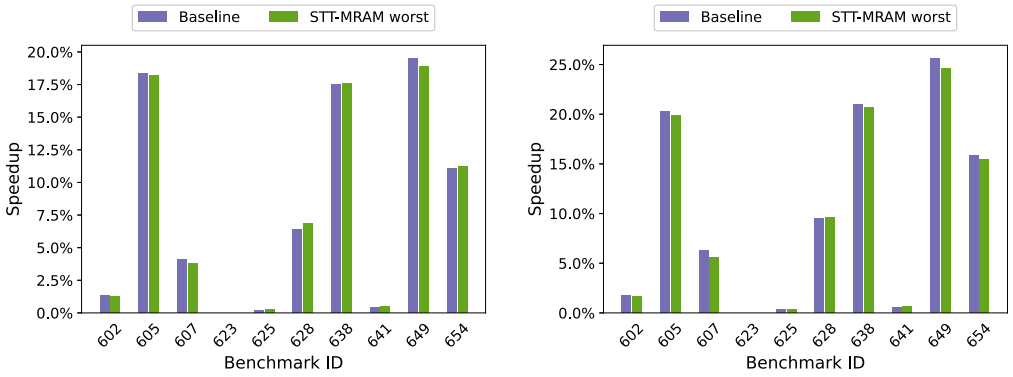
In general, very marginal relative performance gains are achieved with more than eight banks. In Section 6 of this work the analysis is further extended adding energy information.

4.4 Effect of Write Buffer and MSHRs

The write buffer and MSHRs are support structures typically present at any cache level. The first stores pending writebacks are directed to the next level, to serve them in a later moment. The seconds have a similar purpose, but store outstanding misses instead (in non-blocking caches). Both structures when completely filled prevent new requests to be sent forward.

We have analyzed the effect of having larger support structures in the L1 data cache, which could potentially give space to other requests and help counteracting the higher latencies in STT-MRAM.

Increasing the size of the write buffer—from 4 to 16 entries in the in-order core, from 16 to 64 entries in the out-of-order core—has little to no effect on the benchmarks execution. No visible improvement is detected, with the exception of *607.cactuBSSN_s* where the execution time decreases by 1.3% (in-order) and 2.5% (out-of-order, Figure 6).



(a) Data cache prefetcher (b) Data cache + LLC prefetcher

Fig. 7. Stride prefetcher effect on performance (In-order, eight banks).

Increasing the number of MSHRs by three times has a low impact as well. The highest gain, still for *607.cactuBSSN_s*, is of 4.1% in the out-of-order platform.

4.5 Effect of Prefetcher

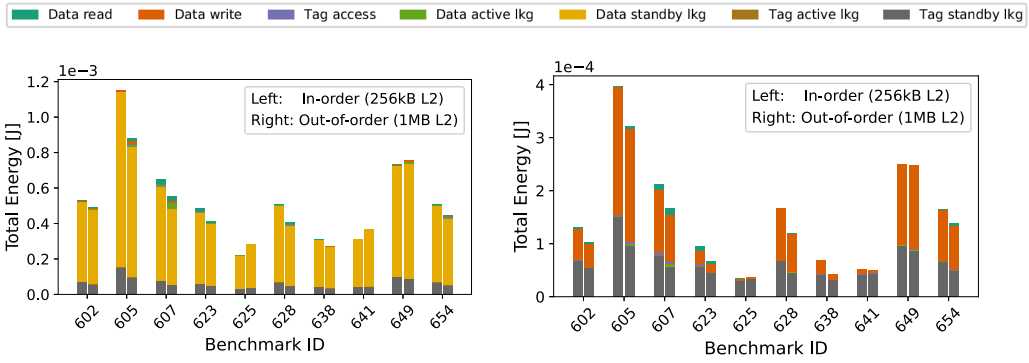
We have explored the effect of a traditional hardware prefetcher with an STT-MRAM-based LLC. Prefetching increases the cache traffic, since it speculatively fetches more blocks than the actually requested. The extra pressure in the LLC could potentially lead to an increase in bank contention that, in the case of STT-MRAM, likely translates in cutting down performance. Therefore, we need to investigate this risk for different types of benchmarks.

Two common scenarios have been tested: one including a degree-4 stride prefetcher on the L1D cache, the other with the same L1 data cache configuration and an additional degree-8 prefetcher on the LLC. We were interested to see the effect before and after the technology change to STT-MRAM. Figure 7 shows the relative gain (compared to the corresponding configuration without prefetching) in performance after the prefetcher(s) activation. The prefetcher appears to be very effective in some situations, most notably with *605.mcf_s*, *638.imagick_s*, and *649.fotonik3d_s*, and completely ineffective in others, like with *623.xalancbmk_s*, *625.x264_s*, and *641.leela_s*. Most importantly in the context of this work, there does not seem to be any difference on how the prefetcher affects the different technologies: Approximately the same amount of speedup is provided to the SRAM as to the STT-MRAM cache configuration. Therefore, we can safely leave this effect out.

5 ENERGY ANALYSIS

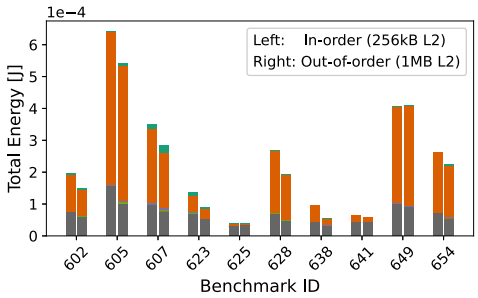
In the first place, we have performed an extensive energy analysis from SPEC CPU2017 simulation data using our accurate 28 nm model. It provides detailed static and dynamic energy information about the different elements composing the cache, including banks, data arrays, arbitration logic, buffers, and the H-tree interconnect.

The results of this first analysis are shown in Figure 8. The purely SRAM configuration is dominated by the static leakage. Even in the case of memory-intensive benchmarks the LLC bank activity has been found to be very low, explaining why the static energy has a big impact. The benchmark that makes more use of banks parallelism, which is *607.cactuBSSN_s*, does not use more than one bank for the vast majority of the execution time (Figure 9). On average, the LLC stays idle for the 83% of the time in the in-order core and for the 74% in the out-of-order one. Switching from



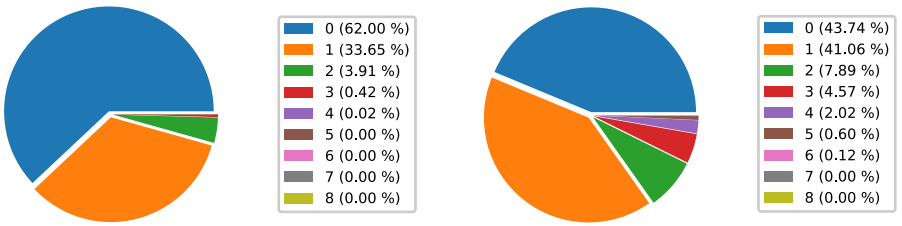
(a) SRAM

(b) STT-MRAM typical case



(c) STT-MRAM worst case

Fig. 8. In-house accurate energy model—Average LLC energy consumption per $100 \cdot 10^6$ instructions (eight banks).



(a) In-order

(b) Out-of-order

Fig. 9. Concurrent LLC banks utilization during *607.cactuBSSN_s* execution (relative time, eight banks).

the first to the second, it is possible to notice a slight increase in the dynamic energy contribution and decrease in leakage due to the higher activity. In general, we can safely conclude that the duty cycle of the different banks is quite low, and hence leakage is very crucial in the overall power budget.

Differences are evident when using STT-MRAM instead of SRAM, as shown in the plots (b) and (c). The standby leakage of the data array is not visible anymore, since it is negligible with this

Table 4. STT-MRAM Energy Saving (Eight Banks—LLC Only)

Benchmark	STT typical		STT worst	
	In-order	Out-of-order	In-order	Out-of-order
602.gcc	75.37%	79.25%	63.26%	69.59%
605.mcf	65.63%	63.41%	44.33%	38.39%
607.cactusBSSN	67.29%	69.96%	45.88%	49.19%
623.xalancbmk	80.31%	83.99%	71.90%	78.38%
625.x264	84.44%	87.40%	82.16%	86.46%
628.pop2	67.07%	70.14%	47.24%	52.04%
638.imagick	77.58%	84.00%	68.58%	79.77%
641.leela	83.16%	86.26%	79.27%	84.18%
649.fotonik3d	65.84%	67.20%	44.69%	46.16%
654.roms	67.71%	69.27%	48.21%	49.77%

kind of technology, while the tag leakage becomes more prominent. The write energy not only is higher than SRAM but also not overshadowed anymore by the static energy, becoming the main source of energy consumption for all the presented benchmarks. It appears that, with a reasonable number of banks, it is always advantageous to use STT-MRAM, especially for less write-intensive benchmarks like *638.imagick_s*.

The relative energy saving numbers for the experiments shown in Figure 8 are synthetically reported in Table 4. It is important to remind that the energy efficiency is dependent on the cache internal configuration and number of banks, as will be shown in the next section.

Reported values are relative to the last-level cache only. To have a more clear idea of what could be the overall energy impact of an STT-MRAM LLC, we need to provide some energy data from the first-level caches as well. The average energy consumption of both L1D and L1I—normalized to 100 million instructions—is 850 *pJ* for the in-order core and 774 *pJ* for the out-of-order core. This value, which includes dynamic and static contributions, is lower in the out-of-order core because of the reduced leakage due to faster execution times. It is also interesting to mention that, while the SRAM LLC is dominated by the static leakage (as visible in Figure 8(a)), the active energy seems to be the main contribution for the overall L1 cache and constitutes 69% of its consumption—on average—in the in-order core, growing to 89% in the out-of-order configuration. This happens because the activity factor is higher than in the LLC.

The L1 energy values reported above correspond to the 62% of the entire cache consumption if we consider a 8-banks SRAM LLC. Given the STT-MRAM savings previously calculated, an LLC technology replacement can bring an average improvement of 23% (worst case) to 28% (typical case) to the cache subsystem.

6 PARETO ENERGY-PERFORMANCE ANALYSIS

One of the important focus points in our analysis is to find out which of the LLC technology choices gives the best results in terms either of performance or energy for each workload. In this regard the number of banks represents an important design choice: A higher number helps reduce the memory contention, as shown in Section 4.3, but it also increases the overall energy consumption (static due to the data array and logic partitioning, dynamic due to the longer interconnect).

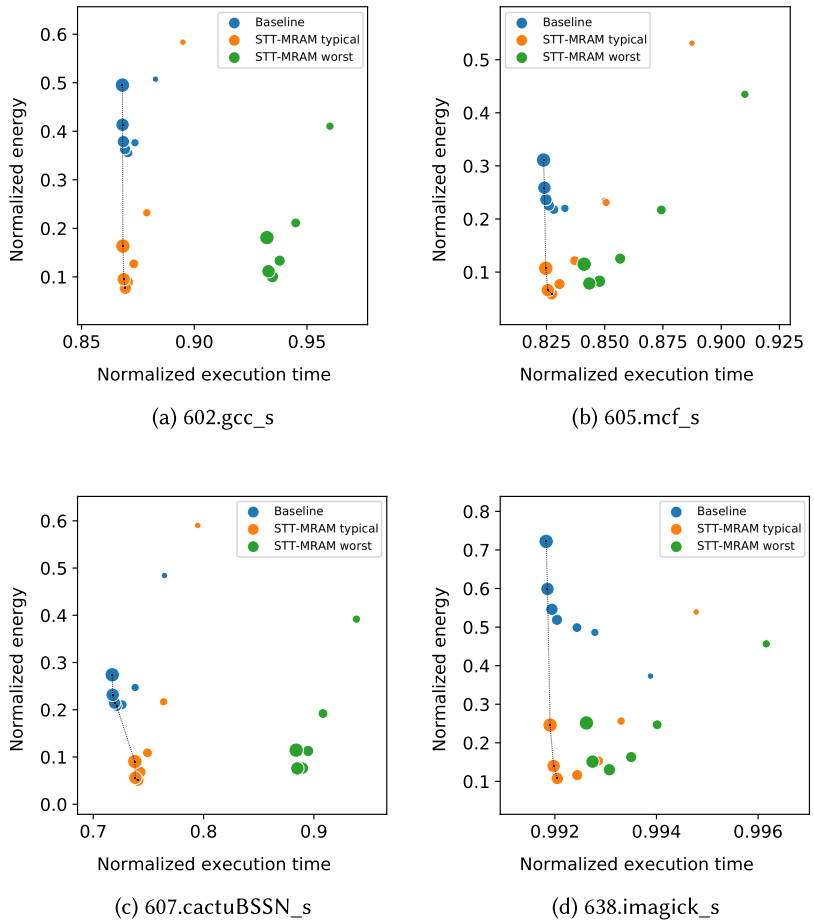
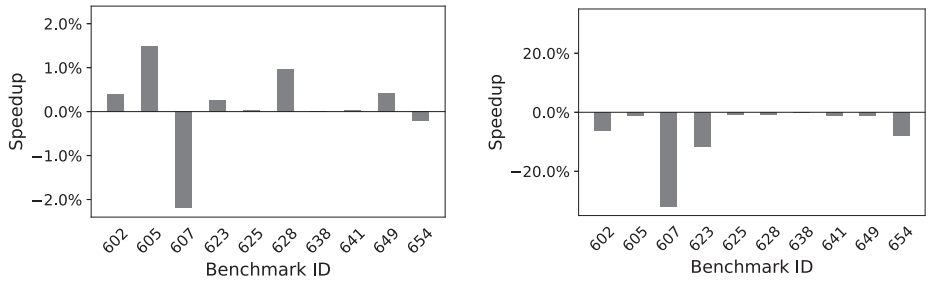


Fig. 10. Energy vs. performance with Pareto front for different number of LLC banks (In-order).

Figure 10 represents the result of the energy-performance analysis for some of the benchmarks. The size of the dots is the number of banks, with the smallest being a single bank and the biggest a 64-banks configuration. The dotted line represents the Pareto front, highlighting the configurations where it is not possible to improve one parameter without degrading the other. As expected, the worst-case STT-MRAM configuration has no point on the Pareto front.

Some of the conclusions are already clear from the previous pages. Although performance-wise the SRAM is always the best choice when using a high number of banks, the difference with the typical case of STT-MRAM is usually negligible. From the system-level energy point of view, STT-MRAM can provide consistent savings, even in the worst-case scenario as long as enough banks are employed.

An important remark is that the "inversion point," where the energy stops decreasing, is different for the two technologies. In general, STT-MRAM benefits more from additional banks compared to SRAM, as already observed in previous work [8]. Using the in-order core, the optimal number of banks (energy-wise) for an SRAM LLC is 4 in most cases, with the notable exception of *625.x264_s* and *638.imagick_s*, where more savings are possible with a single bank at a very small performance price. The STT-MRAM configuration has its sweet spot at 16 banks, which in the worst case grow



(a) Typical case

(b) Worst case

Fig. 11. Performance using STT-MRAM with 16 banks instead of SRAM with 4 banks (Out-of-order).

up to 32. With the out-of-order core the trends are similar, but differences between values shrink near the optimum point. Doubling the banks of the SRAM cache would seem to be beneficial in this case, but given the risible energy/performance gain and the area overhead it is probably not convenient.

At this point, we want to perform a direct comparison between optimal SRAM and STT-MRAM configurations. Considering its increased density, we assume that a 16-banks STT-MRAM can be directly compared to a 4-banks SRAM solution. Figure 11 represents the performance difference between the aforementioned configurations. Due to the increased number of banks, a typical STT-MRAM that is not extremely penalized by process variations seems to perform better than SRAM in most situations, except for memory-intensive benchmarks like *607.cactuBSSN_s*. This is obviously not the case for the worst-case scenario: The average penalization is not significant (6.8%), but can go up to 32% with the usual "problematic" benchmark. The energy saving, though, can well justify the choice of this technology: It is 73% for the in-order and 77% for the out-of-order configuration on average, and it is always above 60% for all the benchmarks regardless of the scenario.

7 NVSIM MODEL

The following part of the work consisted in repeating the experiments with a different energy model. We made use of the memory design tool called NVSim [7], which is publicly available and aims specifically at emerging non-volatile memories. As shown already in the related work (Section 2), it is widely used in the academic community. The idea is to configure such tool to provide realistic STT-MRAM parameters to be used in architectural explorations.

One of the limitations of the NVSim tool is that—as of now—there is no support for multi-bank memory designs. Additionally, it provides a single value for the leakage power without differentiating between active mode or standby mode. To provide a fair comparison with the previous results, the multi-bank trends have been obtained from the reference model and applied to NVSim numbers. Specifically, the per-bank energy ratios between single and multiple banks configurations have been extracted from the in-house model and used in calculation formulas. A similar process has been applied to derive the standby power from the NVSim leakage value, which was assumed to be the leakage in active mode.

The tool is provided with some sample basic cell designs, including a traditional SRAM cell model and two different STT-MRAM cell models, a classic one and a more optimized one (*aggressive* design). We detected a few critical issues in the STT-MRAM cells designs. First, the cell area F^2 is too optimistic, especially when compared to the SRAM model. More importantly, the read power (30 uW in the normal cell, 23.4 uW in the aggressive) is too high and can greatly shorten the

Table 5. Energy Models Comparison (1 MB Cache, Single Bank, 0.5 < Mantissa ≤ 5)

	SRAM			
	LSTP - ReadEDP	LOP - ReadEDP	LSTP - Leakage	LOP - Leakage
Read Energy	E-10	E-11	E-10	E-10
Write Energy	E-10	E-11	E-11	E-11
Leakage Power	E-04	E-01	E-04	E-02
R/W Energy Ratio	1.20	1.20	15.31	15.01

	STT-MRAM aggressive			
	LSTP - ReadEDP	LOP - ReadEDP	LSTP - Leakage	LOP - Leakage
Read Energy	E-10	E-10	E-10	E-10
Write Energy	E-10	E-10	E-10	E-10
Leakage Power	E-04	E-03	E-04	E-03
R/W Energy Ratio	0.44	0.53	0.67	0.66

	Reference values	
	SRAM	STT-MRAM
Read Energy	E-10	E-10
Write Energy	E-10	E-10
Leakage Power	E-03	E-04
R/W Energy Ratio	1.40	0.12

lifespan of the device. This last issue was addressed reducing the read power to 5 uW in the model.

Known cache parameters such as size, associativity, line width, technology node have been specified in the configuration files. The cache access mode has been set to *sequential*, since it was the only configuration where the relation between read and write energy was realistic for STT-MRAM (read energy < write energy). Different combinations of the optimization target and device roadmap parameters have been explored. The device types that can be specified in NVSim are the ones defined by the International Technology Roadmap for Semiconductors: **high-performance (HP), low operating power (LOP), low standby power (LSTP)**. We had to exclude the HP roadmap from the analysis, since it gives surprisingly high leakage values. Latencies have been verified to be compatible with the ones used in the reference model for the typical case.

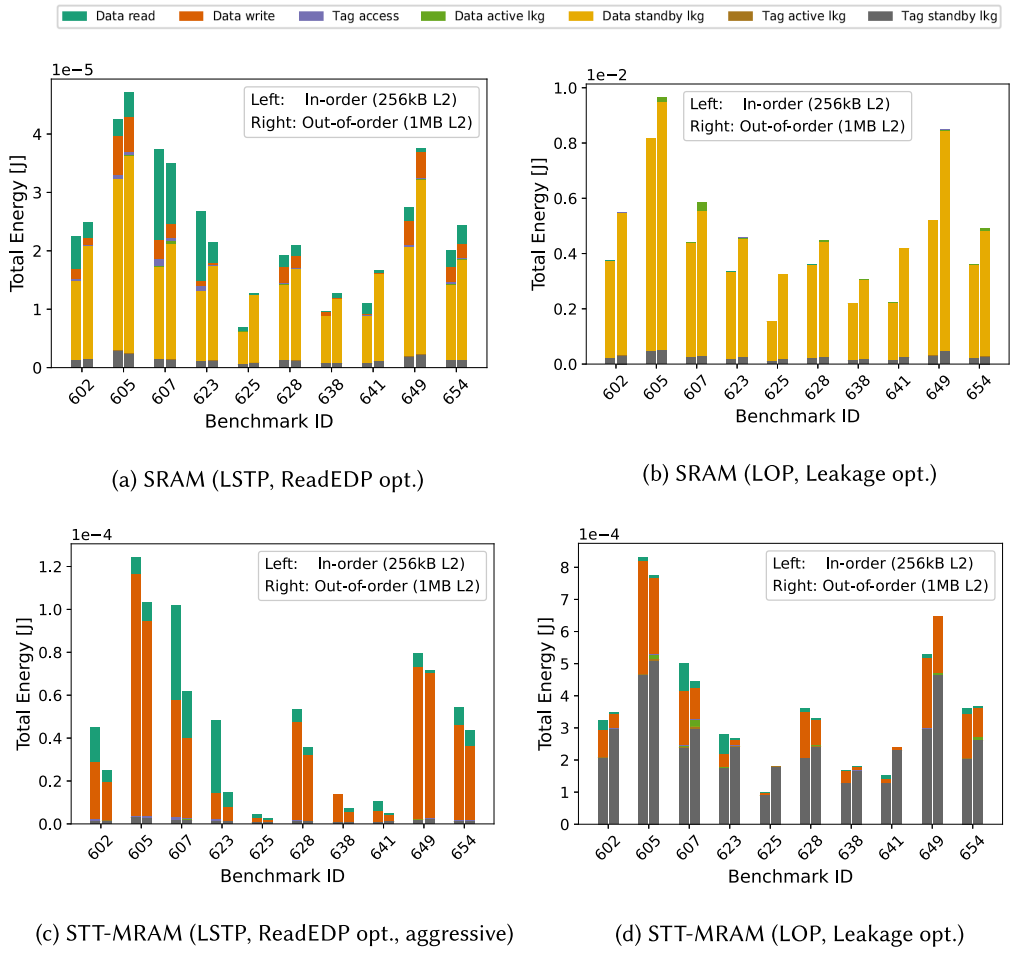
After a thorough exploration, it was not possible to find an energy model matching closely all the characteristics of the internal reference. Table 5 reports the energy parameters of the tested memory technologies. Data are generically expressed as orders of magnitude, except for the read to write energy ratios. While the LSTP STT-MRAM values are comparable—but not identical—to the reference model, the SRAM values are either too optimistic or pessimistic in at least one metric. However, for certain architectural explorations, absolute values are not fully required, and it may be enough to keep the trends between metrics compatible with the actual ones. Table 6 shows SRAM vs. STT-MRAM ratios for read energy, write energy, and leakage power for different NVSim configurations.

Table 6. STT-MRAM vs. SRAM Consumption Parameters Ratios
(1 MB Cache, Single Bank)

		Read Energy	Write Energy	Leakage Power
		STT-MRAM aggressive / LSTP - ReadEDP		
SRAM	LSTP - ReadEDP	1.11E+00	3.01E+00	4.82E-01
SRAM	LOP - ReadEDP	2.45E+00	6.62E+00	1.66E-03
SRAM	LSTP - Leakage	3.23E-01	1.12E+01	5.25E-01
SRAM	LOP - Leakage	7.04E-01	2.39E+01	1.81E-03
		STT-MRAM aggressive / LSTP - Leakage		
SRAM	LSTP - ReadEDP	1.31E+00	2.34E+00	4.38E-01
SRAM	LOP - ReadEDP	2.88E+00	5.14E+00	1.51E-03
SRAM	LSTP - Leakage	3.80E-01	8.66E+00	4.77E-01
SRAM	LOP - Leakage	8.28E-01	1.85E+01	1.65E-03
		STT-MRAM aggressive / LOP - ReadEDP		
SRAM	LSTP - ReadEDP	1.07E+00	2.41E+00	1.95E+01
SRAM	LOP - ReadEDP	2.36E+00	5.29E+00	6.71E-02
SRAM	LSTP - Leakage	3.11E-01	8.92E+00	2.12E+01
SRAM	LOP - Leakage	6.78E-01	1.91E+01	7.32E-02
		STT-MRAM aggressive / LOP - Leakage		
SRAM	LSTP - ReadEDP	1.15E+00	2.09E+00	6.35E+00
SRAM	LOP - ReadEDP	2.53E+00	4.58E+00	2.19E-02
SRAM	LSTP - Leakage	3.33E-01	7.72E+00	6.92E+00
SRAM	LOP - Leakage	7.27E-01	1.65E+01	2.39E-02

Reference values		
Read Energy	Write Energy	Leakage Power
2.30E-01	2.62E+00	1.20E-01

We have selected the two configurations that better approximate the trends observed in the reference model and repeated the experiments of the previous section with these numbers. The results are visible in Figure 12. The first choice has been performed with the aim of finding an STT-MRAM to SRAM consumption ratio similar to the reference model. This was obtained minimizing the Euclidean distance between the combined consumption trends reported in Table 6 and the reference ones. The procedure was applied to data from the out-of-order core LLC (1MB), but the choice has been verified to be valid for combined 256 kB and 1 MB sizes. For both technologies the winning model turned out to be the LSTP variant with **read energy-delay product (ReadEDP)** as the optimization target, represented in plots (a) and (c). Comparing it to the previous results, this configuration is clearly penalized by having a significantly smaller leakage in SRAM—which also affects the STT-MRAM counterpart nullifying the tag leakage contribution—and a relatively lower impact of the write energy in STT-MRAM. Also, the total energy of the SRAM configuration appears to be lower than the one with STT-MRAM, which obviously is against the previous findings. This property will make it nonusable for architectural explorations.



(a) SRAM (LSTP, ReadEDP opt.)

(b) SRAM (LOP, Leakage opt.)

(c) STT-MRAM (LSTP, ReadEDP opt., aggressive)

(d) STT-MRAM (LOP, Leakage opt.)

Fig. 12. NVSim models - Average LLC energy consumption per $100 \cdot 10^6$ instructions (eight banks).

The second configuration, shown in plots (b) and (d), has been selected as the closest one that keeps the physical advantages of STT-MRAM over SRAM. The plots better resemble the energies in Figure 8, but with an overhead in leakage of about one order of magnitude. Clearly, the most suitable configuration will hence depend on the exploration context and objectives.

In both SRAM configurations the benchmarks exhibit a higher leakage energy consumption with the out-of-order core, which was not happening using the in-house model. This is attributable to a different—and more pessimistic—growth trend with the memory size.

Overall, we could not find an NVSim configuration that perfectly matches the values obtained with the in-house energy model. However, with the cell modifications noted above, we would recommend to use the *Low-Power* (LOP) variant with *Leakage* as the optimization target. The SRAM leakage is overestimated, but the overall trends are maintained.

To summarize, the purpose of this analysis was to configure NVSim to obtain STT-MRAM to SRAM trends similar to our detailed in-house models. This was only partially achieved, since the leakage was either underestimated or overestimated in all the configurations. A word of caution is needed here: This tuning approach is obviously not portable and makes sense only if related

to specific technology models. Any significant change on the target memory device, let alone on the technology, would likely require a repetition of the analysis process to find the closest configuration.

8 CONCLUSIONS

STT-MRAM has been under development for a long time and it is a mature technology. With its high density, low static leakage, and reasonable endurance it may be a good replacement for an SRAM LLC despite its longer write latency. Memory banking has shown to be the only explored feature that benefits more STT-MRAM compared to SRAM, due to the increased parallelism that allows to hide longer latencies. Extending the MSHRs/write buffer or adding hardware prefetching improves both technologies to the same extent. The experiments show that in the majority of cases the performance penalization is more than acceptable in the embedded context. The benefit in terms of energy is usually significant, which makes STT-MRAM especially good for IoT/edge devices. Those experiments were conducted with accurate in-house memory models that we also used to tune the NVSim tool. Data resulting from the tool can be used for architectural exploration in similar scenarios, but can bring erroneous conclusions if not properly integrated with extra information.

REFERENCES

[1] Junwhan Ahn, Sungjoo Yoo, and Kiyoung Choi. 2014. DASCA: Dead write prediction assisted STT-RAM cache architecture. In *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 25–36. DOI: https://doi.org/10.1109/HPCA.2014.6835944

[2] Dmytro Apalkov, Alexey Khvalkovskiy, Steven Watts, Vladimir Nikitin, Xueti Tang, Daniel Lottis, Kiseok Moon, Xiao Luo, Eugene Chen, Adrian Ong et al. 2013. Spin-transfer torque magnetic random access memory (STT-MRAM). *J. Emerg. Technol. Comput. Syst.* 9, 2 (May 2013). DOI: https://doi.org/10.1145/2463585.2463589

[3] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 simulator. *SIGARCH Comput. Archit. News* 39, 2 (Aug. 2011), 1–7. DOI: https://doi.org/10.1145/2024716.2024718

[4] James Bucek, Klaus-Dieter Lange, and Jóakim v. Kistowski. 2018. SPEC CPU2017: Next-generation compute benchmark. In *ACM/SPEC International Conference on Performance Engineering (ICPE'18)*. Association for Computing Machinery, New York, NY, 41–42. DOI: https://doi.org/10.1145/3185768.3185771

[5] Mu-tien Chang, Paul Rosenfeld, Shih-lien Lu, and Bruce Jacob. 2013. Technology comparison for large last-level caches (L3Cs): Low-leakage SRAM, low write-energy STT-RAM, and refresh-optimized eDRAM. In *IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 143–154. DOI: https://doi.org/10.1109/HPCA.2013.6522314

[6] Hsiang-Yun Cheng, Jishen Zhao, Jack Sampson, Mary Jane Irwin, Aamer Jaleel, Yu Lu, and Yuan Xie. 2016. LAP: Loop-block aware inclusion properties for energy-efficient asymmetric last level caches. In *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 103–114. DOI: https://doi.org/10.1109/ISCA.2016.19

[7] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. 2012. NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Trans. Comput.-aid. Des. Integ. Circ. Syst.* 31, 7 (2012), 994–1007. DOI: https://doi.org/10.1109/TCAD.2012.2185930

[8] T. Evenblij, M. Perumkunnil, F. Catthoor, S. Sakhare, P. Debacker, G. Kar, A. Furnemont, N. Bueno, J. I. Gómez Pérez, and C. Tenllado. 2019. A comparative analysis on the impact of bank contention in STT-MRAM- and SRAM-based LLCs. In *IEEE 37th International Conference on Computer Design (ICCD)*. IEEE, 255–263. DOI: https://doi.org/10.1109/ICCD46524.2019.00039

[9] Fazal Hameed, Asif Ali Khan, and Jeronimo Castrillon. 2018. Performance and energy-efficient design of STT-RAM last-level cache. *IEEE Trans. Very Large Scale Integ. Syst.* 26, 6 (2018), 1059–1072. DOI: https://doi.org/10.1109/TVLSI.2018.2804938

[10] Fazal Hameed and Mehdi B. Tahoori. 2016. Architecting STT last-level-cache for performance and energy improvement. In *17th International Symposium on Quality Electronic Design (ISQED)*, Vol. 2016-May. IEEE, 319–324. DOI: https://doi.org/10.1109/ISQED.2016.7479221

[11] Cadence Design Systems Inc. 2013. Virtuoso 6.1.6-64b. Retrieved from https://www.cadence.com.

[12] Adwait Jog, Asit K. Mishra, Cong Xu, Yuan Xie, Vijaykrishnan Narayanan, Ravishankar Iyer, and Chita R. Das. 2012. Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs. In *DAC Design Automation Conference*. IEEE, 243–252. DOI: https://doi.org/10.1145/2228360.2228406

[13] N. Khoshavi and R. F. Demara. 2018. Read-tuned STT-RAM and eDRAM cache hierarchies for throughput and energy optimization. *IEEE Access* 6 (2018), 14576–14590. DOI: https://doi.org/10.1109/ACCESS.2018.2813668

[14] Manu Komalan, Sushil Sakhare, Trong Huynh Bao, Siddharth Rao, Woojin Kim, Christian Tenllado, José Ignacio Gómez Pérez, Gouri Sankar Kar, Arnaud Furnemont, and Francky Catthoor. 2017. Cross-layer design and analysis of a low power, high density STT-MRAM for embedded systems. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–4. DOI: https://doi.org/10.1109/ISCAS.2017.8050923

[15] Manu Perumkunnil Komalan, Christian Tenllado, Jose Ignacio Gómez Pérez, Francisco Tirado Fernández, and Francky Catthoor. 2015. System level exploration of an STT-MRAM-based level 1 data-cache. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Vol. 2015-Ap. IEEE, 1311–1316.

[16] Kunal Korgaonkar, Ishwar Bhati, Huichu Liu, Jayesh Gaur, Sasikanth Manipatruni, Sreenivas Subramoney, Tanay Karnik, Steven Swanson, Ian Young, and Hong Wang. 2018. Density tradeoffs of non-volatile memory as a replacement for SRAM-based last level cache. In *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 315–327. DOI: https://doi.org/10.1109/ISCA.2018.00035

[17] Kyle Kuan and Tosiron Adegbija. 2019. HALLS: An energy-efficient highly adaptable last level STT-RAM cache for multicore systems. *IEEE Trans. Comput.* 68, 11 (2019), 1623–1634. DOI: https://doi.org/10.1109/TC.2019.2918153

[18] Fabian Oboril, Rajendra Bishnoi, Mojtaba Ebrahimi, and Mehdi B. Tahoori. 2015. Evaluation of hybrid memory technologies using SOT-MRAM for on-chip cache hierarchy. *IEEE Trans. Comput.-aid. Des. Integ. Circ. Syst.* 34, 3 (2015), 367–380. DOI: https://doi.org/10.1109/TCAD.2015.2391254

[19] Guillaume Patrigeon, Pascal Benoit, Lionel Torres, Sophiane Senni, Guillaume Prenat, and Gregory Di Pendina. 2019. Design and evaluation of a 28-nm FD-SOI STT-MRAM for ultra-low power microcontrollers. *IEEE Access* 7 (2019), 58085–58093. DOI: https://doi.org/10.1109/ACCESS.2019.2906942

[20] S. Sakhare, M. Perumkunnil, T. Huynh Bao, S. Rao, W. Kim, D. Crotti, F. Yasin, S. Couet, J. Swerts, S. Kundu, D. Yakimets, R. Baert, Hr Oh, A. Spessot, A. Mocuta, G. Sankar Kar, and A. Furnemont. 2018. Enablement of STT-MRAM as last level cache for the high performance computing domain at the 5nm node. In *IEEE International Electron Devices Meeting (IEDM)*, Vol. 2018-Dec. IEEE, 18.3.1–18.3.4. DOI: https://doi.org/10.1109/IEDM.2018.8614637

[21] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. 2002. Automatically characterizing large scale program behavior. *SIGPLAN Not.* 37, 10 (Oct. 2002), 45–57. DOI: https://doi.org/10.1145/605432.605403

[22] Sarabjeet Singh and Manu Awasthi. 2019. Memory centric characterization and analysis of SPEC CPU2017 suite. In *ACM/SPEC International Conference on Performance Engineering (ICPE'19)*. Association for Computing Machinery, New York, NY, 285292. DOI: https://doi.org/10.1145/3297663.3310311

[23] Clinton W. Smullen, Vidyabhushan Mohan, Anurag Nigam, Sudhanva Gurumurthi, and Mircea R. Stan. 2011. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *IEEE 17th International Symposium on High Performance Computer Architecture*. IEEE, 50–61. DOI: https://doi.org/10.1109/HPCA.2011.5749716

[24] Shuang Song, Qinzhe Wu, Steven Flolid, Joseph Dean, Reena Panda, Junyong Deng, and Lizy K. John. 2018. *Experiments with spec CPU 2017: Similarity, Balance, Phase Behavior and Simpoints*. Technical Report. Laboratory for Computer Architecture, Department of Electrical and Computer Engineering, The University of Texas at Austin.

[25] Simon Van Beek, Koen Martens, Philippe Roussel, Yueh Wu, Woojin Kim, Siddharth Rao, J. Swerts, Davide Crotti, Dimitri Linten, Gouri Kar, and Guido Groeseneken. 2018. Thermal stability analysis and modelling of advanced perpendicular magnetic tunnel junctions. *AIP Adv.* 8, 5 (2018), 055909. DOI: https://doi.org/10.1063/1.5007690

[26] Jue Wang, Xiangyu Dong, and Yuan Xie. 2013. OAP: An obstruction-aware cache management policy for STT-RAM last-level caches. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*. IEEE, 847–852. DOI: https://doi.org/10.7873/DATE.2013.179

[27] Zhe Wang, Daniel A. Jiménez, Cong Xu, Guangyu Sun, and Yuan Xie. 2014. Adaptive placement and migration policy for an STT-RAM-based hybrid cache. In *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 13–24. DOI: https://doi.org/10.1109/HPCA.2014.6835933

[28] Jia Zhan, Onur Kayiran, Gabriel H. Loh, Chita R. Das, and Yuan Xie. 2016. OSCAR: Orchestrating STT-RAM cache traffic for heterogeneous CPU-GPU architectures. In *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Vol. 2016-Dec. IEEE, 1–13. DOI: https://doi.org/10.1109/MICRO.2016.7783731

Received February 2021; revised July 2021; accepted October 2021