Impact of Cache Coherence Protocols on the Power Consumption of STT-RAM-Based LLC

Mu-Tien Chang†‡, Shih-Lien Lu*, and Bruce Jacob†
†University of Maryland *Intel Corporation
†{mtchang, blj}@umd.edu *shih-lien.l.lu@intel.com

Abstract—To gain higher density and lower leakage, STT-RAM has been considered an alternative to SRAM for implementing last-level caches (LLCs). However, STT-RAM requires high write energy to program. Consequently, frequent write-backs from the upper-level caches or cache fills from the main memory will result in high LLC power.
Both the broadcast and write-back traffic are affected by the cache coherence protocol. In this paper, we study the impact of coherence protocols on the power consumption of the STT-RAM LLC, and the entire cache hierarchy (including the interconnection power). Based on full-system simulation executing multi-threaded benchmarks, we show that although for some of the workloads, different protocols produce very different broadcast or write-back traffic, for these workloads, the interconnection and the write-back power are only a small fraction of the overall power consumption. Cache coherence protocol thus has very little impact on the power of the STT-RAM LLC and the cache hierarchy.

Index Terms—Cache design, cache power analysis.

1 Introduction

Traditional processors use SRAMs (static random access memories) for cache implementations. SRAM-based caches are fast but dissipate high leakage, making them the major contributor to the processor power consumption, especially when idle. For instance, for a typical 32nm multi-core processor, an SRAM-based last-level cache (LLC) contributes around 15% of the processor peak power and 30% of the standby power [1].

To reduce the high SRAM-based LLC power consumption, one solution is to implement LLCs with STT-RAM (spin-transfer torque magnetic random access memory), an emerging non-volatile memory technology. STT-RAM offers low leakage and high density, with satisfying read performance and endurance. However, it requires long write time and high write energy to program [2]. Frequent writes to an STT-RAM cache would therefore result in high cache power.

There are two sources to LLC writes: (1) write-backs from upper-level caches; and (2) cache fills from the main memory. The number of write-backs is affected by both the replacement policy and the cache coherence protocol. For example, compared to the MESI coherence protocol, using the MOESI protocol reduces the number of write-backs by as high as 24%. The goal of this work is to study the impact of coherence protocols on the power consumption of STT-RAM (read-write asymmetric) LLC.

The main contributions of this paper are as follows:

- We evaluate the power consumption of an STT-RAM-based shared LLC with respect to various cache coherence implementations, including MSI, MESI, MOSI, and MOESI. To the best of our knowledge, this paper is the first work to explore the impact of coherence protocols on STT-RAM LLC power. We also evaluate the power consumption of the entire cache hierarchy, including the SRAM-based L1 caches, the STT-RAM-based L2 cache (LLC), and the interconnections between each of the cache elements.
- We show that cache coherence protocol has insignificant impact on both the STT-RAM LLC power and the cache hierarchy power. Based on full-system simulations executing PARSEC multi-threaded benchmarks, although different protocols result in different number of broadcasts or write-backs for some workloads, the interconnection and write-back power are not the deciding factors for the overall power consumption.

The remainder of this paper is structured as follows: Section 2 presents a qualitative comparison of different cache coherence protocols. Section 3 describes our experimental setup. Section 4 shows the results and analysis. Section 5 concludes this work.

2 Cache Coherence Protocols

This section summarizes the differences between the MSI, MESI, MOSI, and MOESI cache coherence protocols. MSI is the basis of the three other protocols. When using MSI, a cache line is in one of the three states: *Modified*, *Shared*, or *Invalid*. The MESI protocol [3] adds an additional *Exclusive* state. The benefit of adding the *Exclusive* state is to reduce the number of broadcasts when writing to a line that is present in only one of the caches. Because an *Exclusive* line exists in only one cache, when writing to it, broadcasting an invalidation signal becomes unnecessary. On the contrary, when writing to a *Shared* line, broadcasting an invalidation signal is required because the remote caches may contain the shared copy.

The MOSI protocol adds an additional *Owned* state to MSI, which potentially reduces the number of write-backs. *Owned* is similar to *Shared*, in which other caches can hold a shared copy of the data. However, unlike the *Shared* state in MSI, an *Owned* line can be dirty. When a *Modified* line is read by a remote cache, the local copy will transition to *Owned*, and the line filled in the remote cache will become *Shared*. Different from the MSI protocol, write-back is not required on a remote read hit.

‡ *Mu-Tien Chang is a Senior Engineer at Samsung.*
Email: mutien.chang@ssi.samsung.com

The MOESI protocol [4] adds both the *Exclusive* state and the *Owned* state, with the benefit of both reducing the number of broadcasts and the number of write-backs.

Table 1 compares the MSI, MESI, MOSI, and MOESI cache coherence protocols.

TABLE 1: Comparison of cache coherence protocols.

Protocol	Feature
MSI	Low complexity
MESI	Reduce #broadcasts; medium complexity
MOSI	Reduce #write-backs; medium complexity
MOESI	Reduce #broadcasts and #write-backs; high complexity

3 Experimental Setup

3.1 Methodology

Our experiments are based on a full-system, multi-core simulator, MARSS [5]. Currently, MARSS supports a snoopy-based coherence protocol, MESI; and a directory-based coherence implementation based on the MOESI protocol. We extend the simulator with three other snoopy-based coherence implementations: MSI, MOSI, and MOESI. Furthermore, we modify MARSS such that it supports asymmetric cache read and write latencies. This allows users to evaluate caches based on asymmetric memory technologies, such as STT-RAM.

The baseline processor model is an 8-core, Atom processor, with two-levels of caches. The L1 caches are private to each core, implemented using SRAM. The last-level L2 cache is shared between all cores, implemented using STT-RAM. Moreover, cache coherence is maintained via snoopy-based coherence protocols (MSI, MESI, MOSI, and MOESI), and a pseudo-LRU replacement policy [6] is utilized for all the caches. Table 2 summarizes the system configuration.

We use a modified version of CACTI [7] to obtain the power and performance parameters for the caches [8]. To estimate the bus energy and speed, we utilize the on-chip bus model and parameters described in [9]. Both the cache and the bus models are based on the 32nm technology node. The leakage component of the STT-RAM cache comes from the peripheral circuitry (e.g., decoders, multiplexers, sense-amplifiers, precharge circuits, write drivers). They are based on the PTM low power CMOS model [10]. The performance and power/energy parameters are summarized in Table 3.

TABLE 2: Baseline system configuration. Note that the private L1 caches are managed using the MSI, MESI, MOSI, or the MOESI snoopy-based coherence protocols.

Processor	8-core Atom, 2GHz, 4-wide issue width
L1D (SRAM)	Private 32KB per core, 8-way, 64B lines
L1I (SRAM)	Private 32KB per core, 8-way, 64B lines
L2 (STT-RAM)	Shared 16MB write-back cache, 16-way, 64B lines
Main memory	8GB

3.2 Workloads

We simulate multi-threaded benchmarks to evaluate various cache coherence protocols. These benchmarks include *blackscholes*, *bodytrack*, *canneal*, *facesim*, *fluidanimate*, *freqmine*, *raytrace*, and *swaptions*, all from the PARSEC benchmark suite [11]. For each of the workloads, we simulate with three different input sizes: *simsmall*, *simmedium*, and *simlarge*. In general, a larger input size has a larger working set size, and is more memory intensive. All of them executes on top of Ubuntu 9.04 (Linux 2.6.31), executing 2.4 billion instructions starting from the region of interest.

TABLE 3: Performance and power/energy parameters used for the experiments.

L1	Latency: 2 cycles Read energy: 0.2 nJ/access Write energy: 0.2 nJ/access Leakage: 8 mW
L2	Read latency: 5 cycles Write latency: 62 cycles Read energy: 0.8 nJ/access Write energy: 20 nJ/access Leakage: 480 mW
Bus	0.16 nJ/broadcast

4 Results and Analysis

In this section, we first show the differences in the **number of broadcasts** and the **number of write-backs** when applying different coherence protocols. Then, we demonstrate the **STT-RAM LLC power breakdown** and the **cache hierarchy (L1, L2, and bus) power breakdown** with respect to various protocols. Finally, we show the **sensitivity to number of cores**.

4.1 Number of Broadcasts and Write-backs

Table 4 shows the normalized number of broadcasts and the normalized number of write-backs, when using the MSI, MESI, MOSI, and MOESI protocols. Across all the benchmarks and input sizes, MESI and MOESI reduce the number of broadcasts by 7% on average (up to 24%). On the other hand, when using MOSI and MOESI, the number of write-backs is reduced by 5% on average (up to 23%). These numbers indicate that the MOESI protocol has the best potential to reduce the bus energy and the LLC write-back power.

4.2 STT-RAM LLC Power Breakdown

Figure 1 shows the STT-RAM LLC power breakdown when using the MSI coherence protocol. **Applying MSI, MESI, MOSI, or MOESI results in very similar LLC power consumption.** This is because of the following:

- Although MOSI and MOESI substantially reduce the number of write-backs for workloads such as *blackscholes-simsmall*, *bodytrack*, and *fluidanimate*, the power consumed by write-back operations for these workloads is relatively low (see Figure 1). The biggest contributor of LLC power is leakage instead. Recall that although STT-RAM cells are non-volatile, the peripheral circuitry of an STT-RAM cache dissipates leakage.
- Write-back power is shown to be the main source of LLC power for *canneal*, *facesim*, *swaptions* (see Figure 1). However, for *canneal* and *facesim*, cache coherence protocols has negligible impact on the number of write-backs, therefore applying different cache coherence protocols has very little effect on the total write-back power consumption. For *swaptions-simmedium*, MOSI and MOESI reduce the LLC power by around 4%. Among the workloads considered, the 4% reduction is the biggest power saving we have observed.

4.3 Cache Hierarchy Power Breakdown

Figure 2 illustrates the power breakdown of the entire cache hierarchy, including the energy consumed by the interconnection. Our simulation results show that for *canneal*, *fluidanimate*, *freqmine*, the MESI and MOESI protocols substantially reduce the number of broadcasts (see Table 4). However, although

TABLE 4: Normalized number of broadcasts and write-backs. The MESI and MOESI protocols result in fewer broadcasts on average; the MOSI and MOESI protocols result in fewer write-backs on average.

		#Broadcasts				#Write-backs			
Benchmark	Input	MSI	MESI	MOSI	MOESI	MSI	MESI	MOSI	MOESI
blackscholes	simsmall	1	0.96	0.99	0.95	1	0.98	0.77	0.77
	simmedium	1	1	1	0.99	1	1	0.96	0.96
	simlarge	1	1	1	0.99	1	1	0.98	0.98
bodytrack	simsmall	1	0.99	0.99	0.99	1	1	0.92	0.92
	simmedium	1	0.99	1	0.99	1	1	0.90	0.90
	simlarge	1	0.98	1	0.98	1	1	0.90	0.90
canneal	simsmall	1	0.81	1	0.81	1	1	0.99	0.99
	simmedium	1	0.81	1	0.81	1	1	0.99	0.99
	simlarge	1	0.82	1	0.82	1	1	1	1
facesim	simsmall	1	0.96	1	0.96	1	1	1	1
	simmedium	1	0.97	1	0.97	1	1	1	1
	simlarge	1	0.96	1	0.96	1	1	1	1
fluidanimate	simsmall	1	0.83	0.98	0.83	1	1	0.84	0.84
	simmedium	1	0.87	0.98	0.87	1	1	0.89	0.89
	simlarge	1	0.88	1	0.86	1	1	0.91	0.91
freqmine	simsmall	1	0.91	1	0.91	1	1	1	1
	simmedium	1	0.88	1	0.88	1	1	1	1
	simlarge	1	0.76	1	0.76	1	1	1	1
raytrace	simsmall	1	0.99	1	1	1	1	0.96	0.96
	simmedium	1	0.98	1	0.97	1	1	0.95	0.94
	simlarge	1	0.98	1	0.98	1	1	0.95	0.95
swaptions	simsmall	1	0.99	1	0.99	1	1	0.94	0.93
	simmedium	1	0.94	1	0.94	1	0.99	0.90	0.91
	simlarge	1	0.98	1	0.98	1	1	0.96	0.95
Average		1	0.93	1	0.93	1	1	0.95	0.95

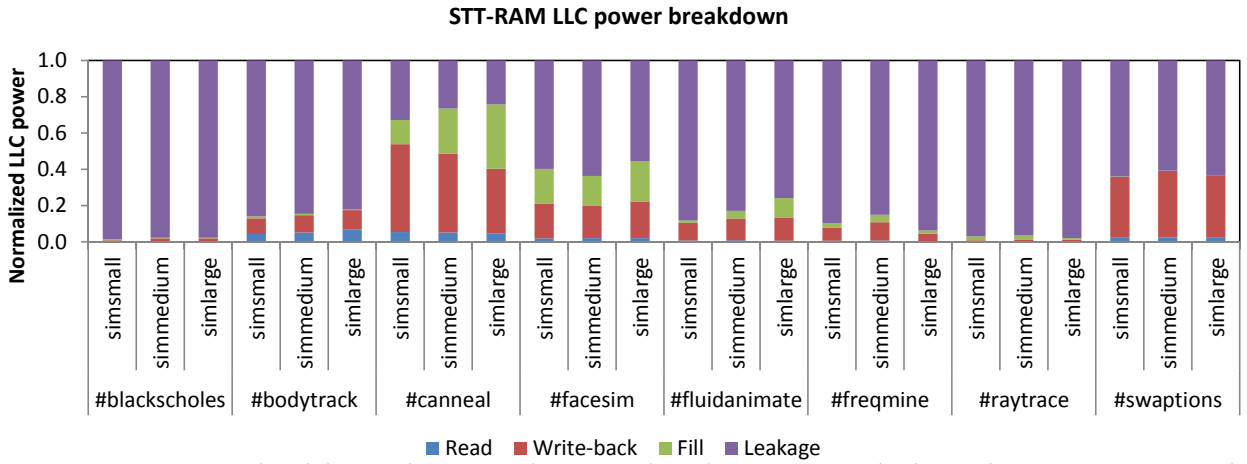

Fig. 1: STT-RAM LLC power breakdown when using the MSI cache coherence protocol. The total LLC power is normalized to 100%. Note that for *blackscholes-simsmall*, *bodytrack*, and *fluidanimate*, MOSI and MOESI result in substantially fewer write-backs, but write-back is not the key factor that determines the total LLC power. Also note that for *canneal*, *facesim*, although write-back contributes to a significant portion of the LLC power, different coherence protocols result in similar number of write-backs (see Table 4). Applying different cache coherence protocols thus bring very similar LLC power consumption.

MESI and MOESI reduce the number of broadcasts and thus the L1-L2 interconnection power, from Figure 2, the power consumption of the interconnections is relatively low compared to the power consumed by the L1 and L2 caches. Cache coherence protocol therefore has very little impact on the power consumption of the cache hierarchy.

4.4 Sensitivity to Number of Cores

We also conduct a sensitivity analysis showing the number of broadcasts and write-backs with respect to the number of cores, as shown in Figure 3. On average, as the number of cores increases, MESI and MOESI bring more reduction in broadcast traffic. This is because in general, a snoopy-based multi-core processor will experience more bus traffic when there are more cores; therefore it is more likely to see the effectiveness of MESI and MOESI.

Contrary to the observation regarding broadcast traffic versus core count, the reduction in write-back traffic when using MOSI (or MOESI) is almost the same for 2-, 4-, and 8-core processors. Although the number of write-backs grows with higher core counts in general, MOSI and MOESI are only showing very little increasing benefits with regard to write-back traffic reduction.

To sum up, from the results of using an 8-core processor, we already conclude that on average, cache coherence policy has insignificant effect on the power of the LLC as well as the entire cache hierarchy. Given that with fewer cores, cache coherence policy has less impact on the broadcast and write traffic, the implication is that it also has negligible effect on cache power

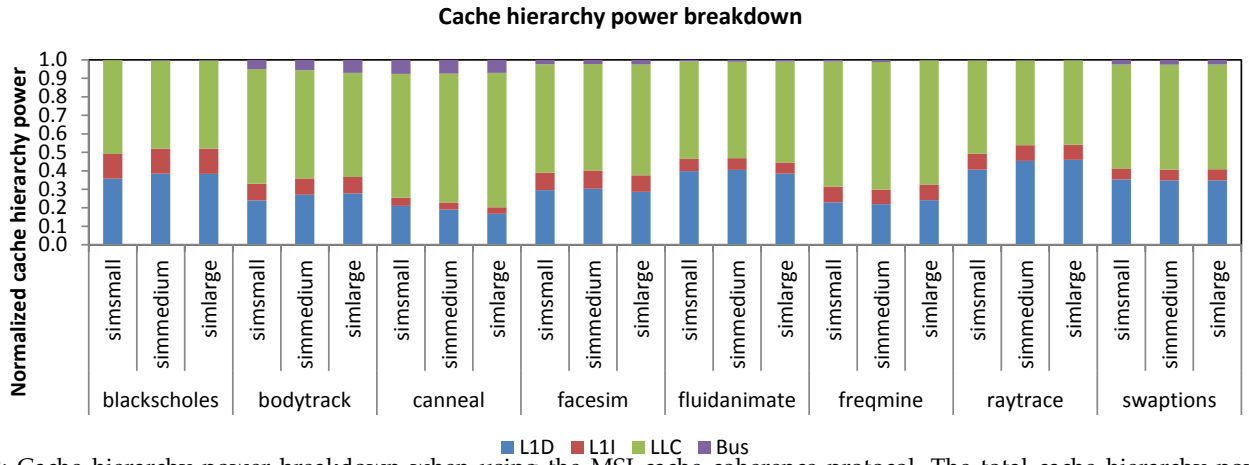

Fig. 2: Cache hierarchy power breakdown when using the MSI cache coherence protocol. The total cache hierarchy power is normalized to 100%. Note that although for *canneal*, *fluidanimate*, and *freqmine*, MESI and MOESI result in a lot fewer broadcasts, the bus energy is not the determining factor of the total power. Coherency protocol therefore has insignificant impact on the cache hierarchy power consumption.

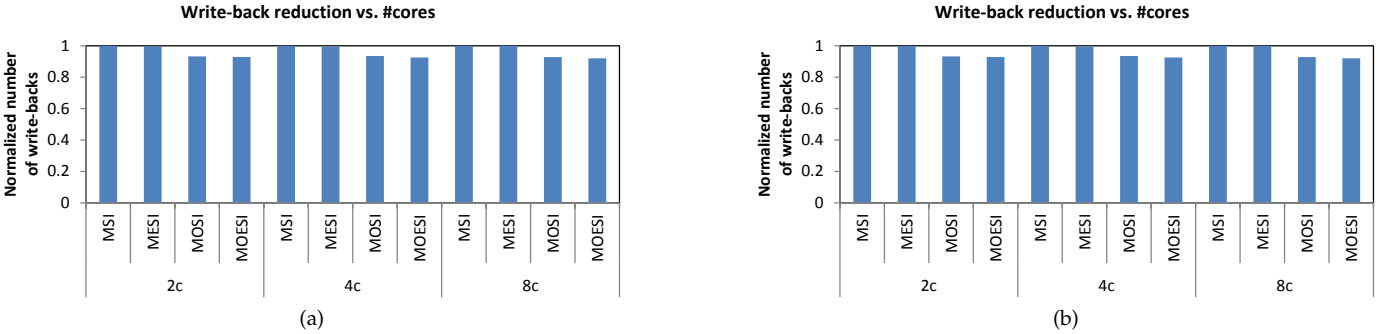

Fig. 3: Sensitivity to number of cores. (a) Broadcast reduction with respect to number of cores. (b) Write-back reduction with respect to number of cores. The results are normalized using MSI as the baseline.

consumption for 2- and 4-core processors.

5 Conclusion

STT-RAM has been considered a potential replacement for SRAM in the context of LLC, due to features including high density and low leakage. However, because STT-RAMs require high write energy to program, frequent write-backs from the upper-level caches will result in high LLC power. The number of write-backs is affected by the coherence protocol used by the upper-level caches. In this paper, we model four typical cache coherence protocols (MSI, MESI, MOSI, MOESI) and demonstrate their impact on the power consumption of STT-RAM-based LLC. Based on full-system simulations, we show that coherence protocol has insignificant impact on STT-RAM LLC power and the power of the entire cache hierarchy.

Acknowledgments

This research was funded in part by Intel Labs University Research Office, the United States Department of Energy, Sandia National Laboratories, and the United States Department of Defense.

References

[1] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *MICRO*, 2009.

[2] H. Li, X. Wang, Z.-L. Ong, W.-F. Wong, Y. Zhang, P. Wang, and Y. Chen, "Performance, Power, and Reliability Tradeoffs of STT-RAM Cell Subject to Architecture-Level Requirement," *IEEE Trans. Magnetics*, vol. 47, pp. 2356–2359, Oct. 2011.

[3] M. S. Papamarcos and J. H. Patel, "A Low-Overhead Coherence Solution for Multiprocessors with Private Cache Memories," in *ISCA*, 1984.

[4] P. Sweazey and A. J. Smith, "A Class of Compatible Cache Consistency Protocols and Their Support by the IEEE Futurebus," in *ISCA*, 1986.

[5] A. Patel, F. Afram, S. Chen, and K. K. Ghose, "MARSSx86: A Full System Simulator for x86 CPUs," in *DAC*, 2011.

[6] H. Al-Zoubi, A. Milenkovic, and M. Milenkovic, "Performance Evaluation of Cache Replacement Policies for the SPEC CPU2000 Benchmark Suite," in *ACMSE*, 2004.

[7] "CACTI 6.5." http://www.hpl.hp.com/research/cacti/.

[8] M.-T. Chang, P. Rosenfeld, S.-L. Lu, and B. Jacob, "Technology Comparison for Large Last-Level Caches (L3Cs): Low-Leakage SRAM, Low Write-Energy STT-RAM, and Refresh-Optimized eDRAM," in *HPCA*, 2013.

[9] M. Ghoneima, Y. Ismail, M. M. Khellah, J. Tschanz, and V. De, "Serial-Link Bus: A Low-Power On-Chip Bus Architecture," *IEEE Trans. Circuits and Systems I*, vol. 56, no. 9, pp. 2020–2032, 2009.

[10] "Predictive Technology Model." http://ptm.asu.edu/.

[11] C. Bienia, *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, Jan. 2011.