Cache Revive: Tuning Retention times of STT-RAM Caches for Enhanced Performance in CMPs.

Anonymous MICRO submission

Paper ID 761

Abstract

Spin-Transfer Torque RAM (STT-RAM) is a CMOS compatible emerging non-volatile memory (NVM) technology that has the potential to replace the conventional on-chip SRAM caches for designing a more efficient memory hierarchy for future multicore architectures. However, it's high write latency and dynamic write energy are major obstacles for being competitive with the SRAM-based cache hierarchy. On the other hand, STT-RAM technology has another adaptable feature that it is possible to reduce its write latency by reducing its retention time, thereby making it volatile. In this paper, we exploit this volatile property of the STT-RAM for designing an efficient L2 cache architecture. The paper addresses several critical design issues such as how do we decide a suitable retention time for last level cache, what is the relationship between retention time and write latency, and how do we architect the cache hierarchy with a volatile STT-RAM. Through an extensive execution driven analysis of the inter-write time of several PARSEC and SPEC 2006 banchmarks, we observe that retention time in the order of 10-40 ms is a good design point to handle most of the writes. Then for the rest of the cache blocks that have a higher inter-write time than the STT-RAM retention time, we propose an architectural solution to identify these blocks with a per block 2 bit counter, temporarily save a limited number of MRU blocks in a buffer, and writeback the rest of the dirty blocks to avoid any data loss. Our experiments with 4

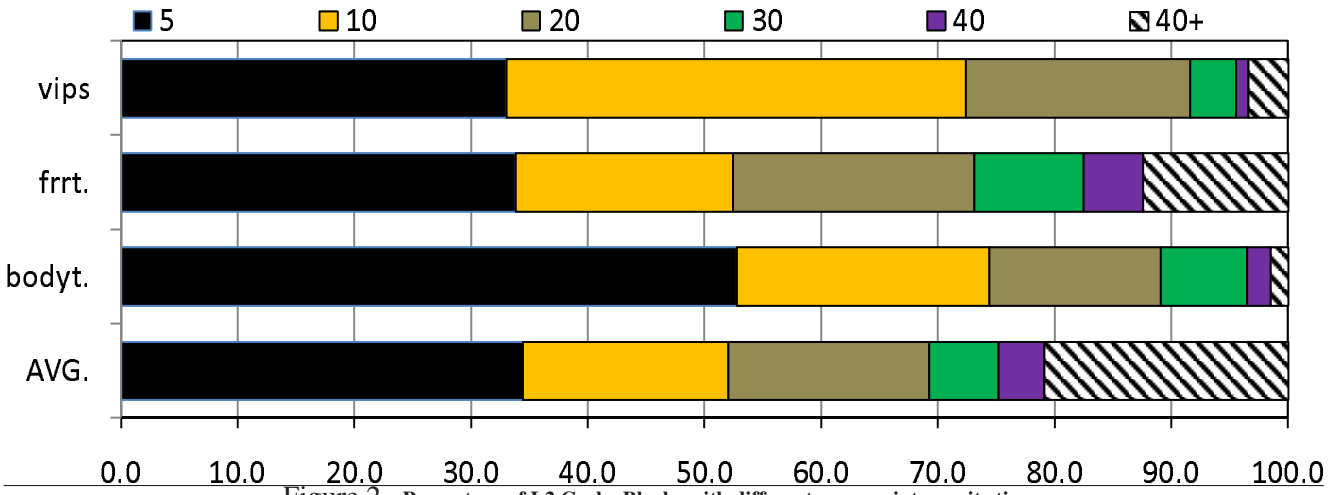

Figure 2. **Percentage of L2 Cache Blocks with different average inter-write times**

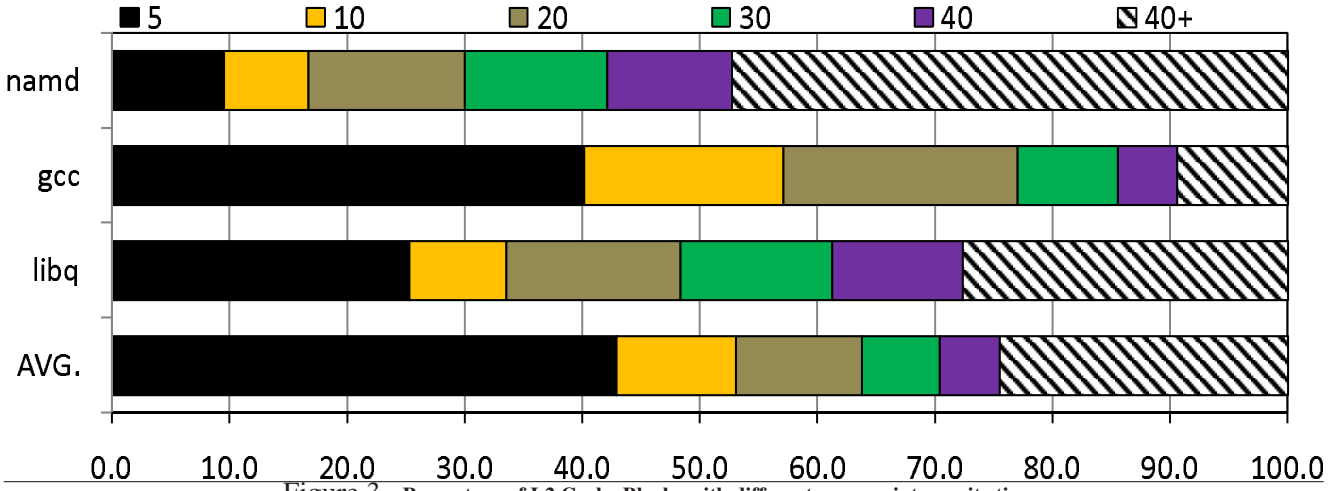

Figure 3. **Percentage of L2 Cache Blocks with different average inter-write times**

and 8-core architectures with an SRAM-based L1 cache and STT-RAM-based L2 cache indicate that not only we can eliminate the high write overhead of an NVM STT-RAM, but can provide on an average 10-12% improvement in IPC compared to the traditional SRAM-based design, while reducing the energy consumption significantly

1. Introduction

2. Motivation

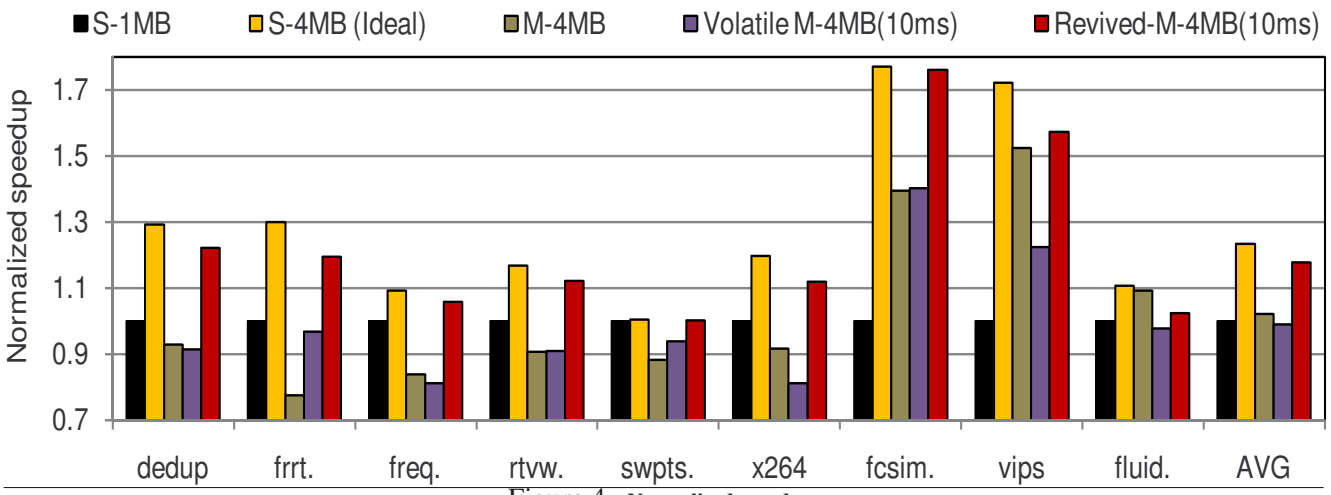

Figure 4. **Normalized speedup**

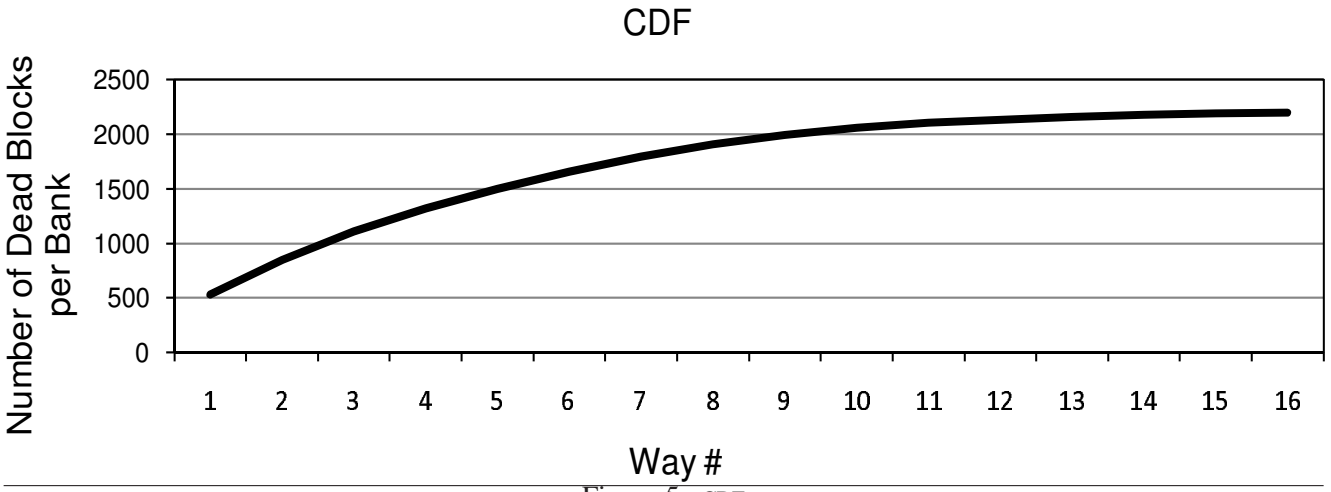

Figure 5. **CDF**

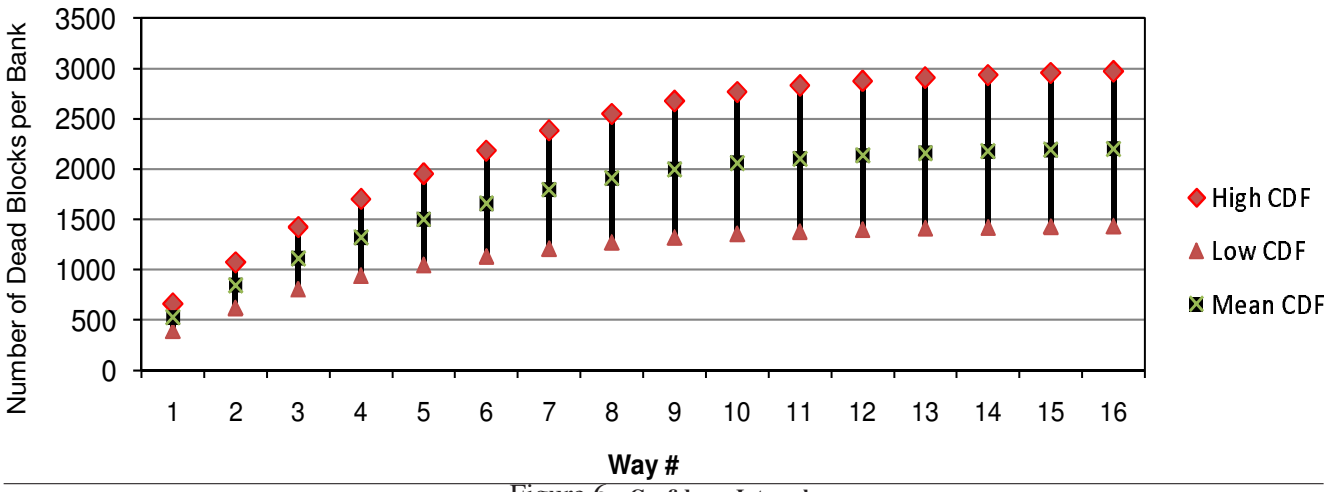

Figure 6. **Confidence Intervals**

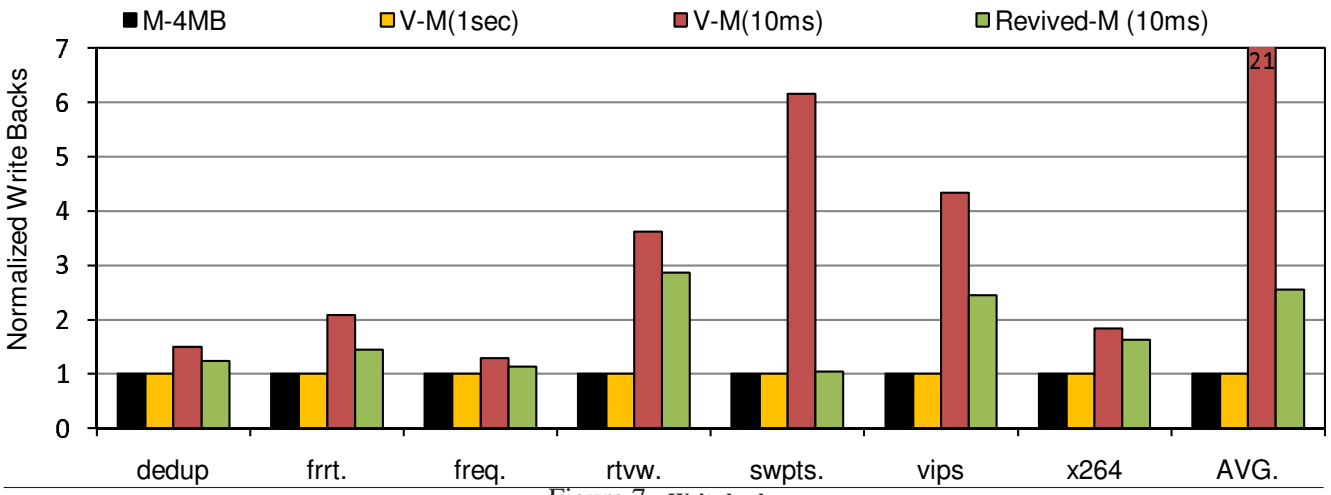

Figure 7. **Write backs**

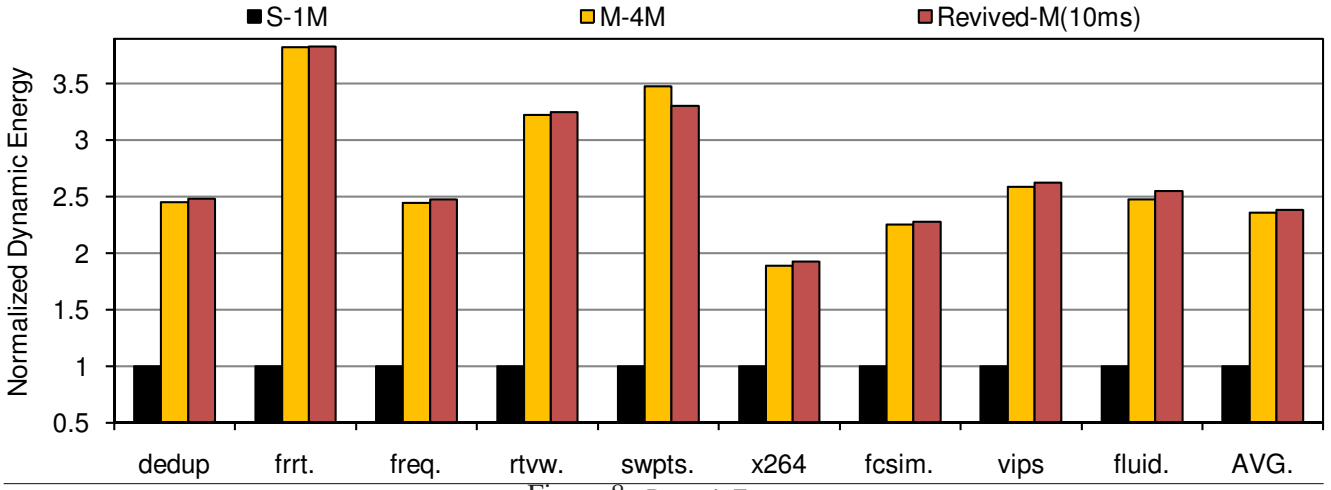

Figure 8. **Dynamic Energy**

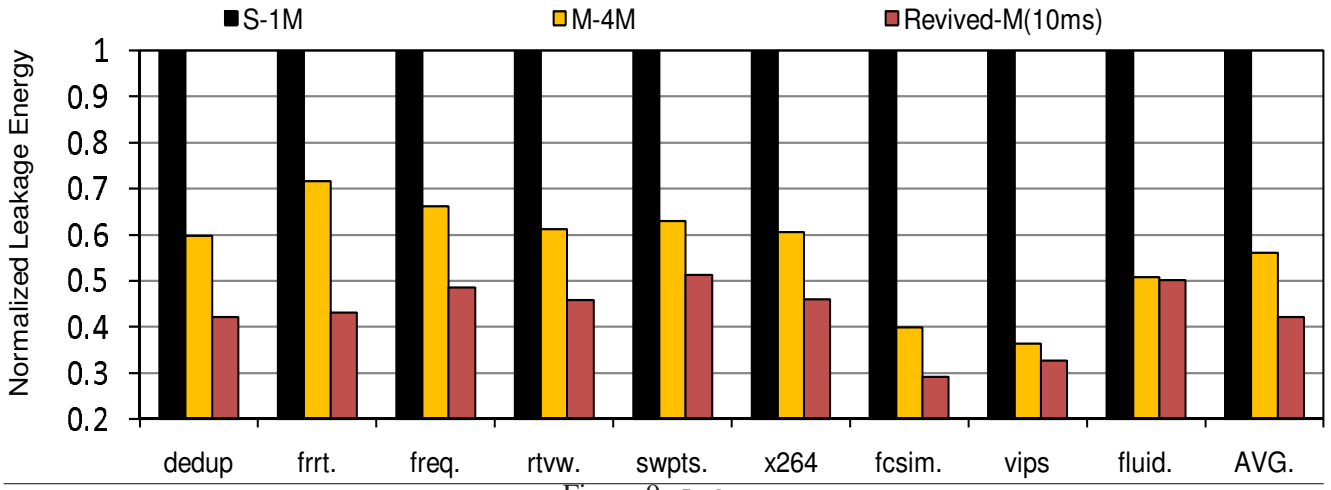

Figure 9. **Leakage**

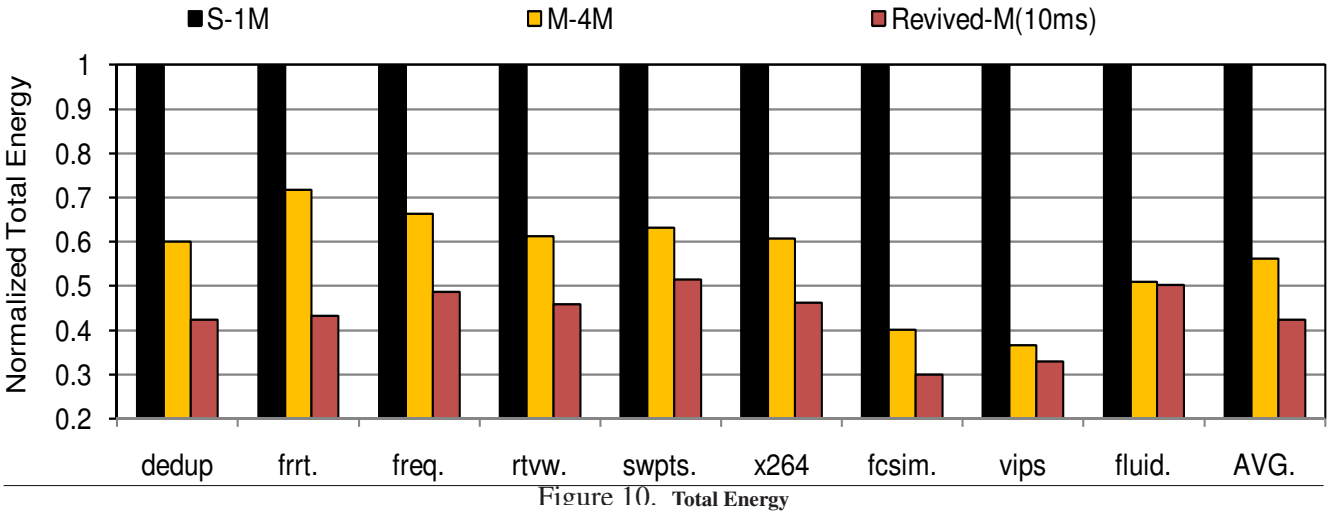

Figure 10. **Total Energy**

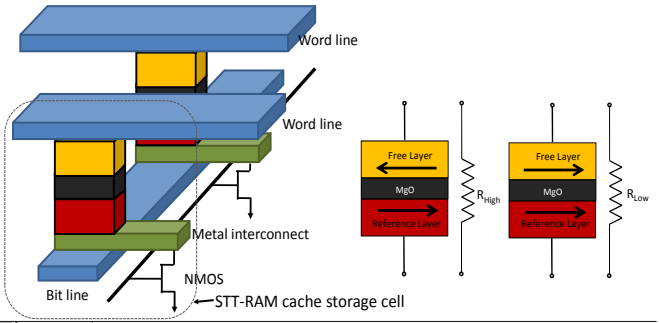

Figure 1. **(a) Structural view of of STT-RAM Cache Cell (b) Anti Space Parallel (High Resistance, Indicating "1" state (c) Parallel (Low Resistance, Indicating "0" state**

Architects always envision to have a cache hierarchy which not only has fast access times but also consume very less leakage and dynamic power.To bridge the gap between the current and the utopian cache hierarchy, exploration of new properties of emerging memory technologies is imperative. [**?**] et.al proposed that reducing the planar area of STT-RAM, reduces the retention time which in turn leads to lower write time. In current era, where the STT-RAM cell size is in the range of 3-5 F^2, there is not enough scope for reducing write time by this technique. Figure shows how write time reduces when retention time is reduced by means of changing the write pulse duration. The 10 year retention time STT-RAM cache design is traditional non-volatile STT-RAM. As we observe from the graph that write time significantly reduces, as we tend towards volatile STT-RAM.

The reduction of retention time of cache blocks has opened plethora of challenges for the architects. The main challenges are (1) To avoid any data loss because of volatility of cache blocks. (2) To ensure that the data is correct and no random bit flips have happened. 3) To architect cache hierarchy, which deals with issues (1) and (2) and still reap performance and energy benefits of reduced write time. This paper systematically addresses these challenges by finding a ideal retention time which can lead to

minimum possible write latency with minimal architectural overheads.

By means of many experiments we came up with a term called revival time which is defined as the time between consecutive writes to the same block. We can say that after every revival time interval, physical cache block is refreshed and ready to be used again.

Figure **??** shows the percentage of L2 cache blocks binned into different average revival time slots for PARSEC and SPEC 2006 Benchmarks. We collected these results by modeling a 4MB STT-RAM L2 Unified Banked Cache using M5 Simulator with 2GHz processor consisting of 4 cores.Table contains the details of the configuration of the simulated system. While collecting results, we ensured that block is valid when the consecutive writes are performed on this block. If the block gets invalidated in between, we only consider the time between, when the block is last written and when the block gets invalidated.

We observe from the graph that there are significant number of blocks which gets refreshed frequently and also a good percentage of blocks which remain unrefreshed for longer time. This graph gives us the basis on which we can choose the optimal retention time. Reducing the retention time too much will make the cache too volatile and increasing it will aggravate the write time. We choose 10ms as the optimal retention time, as there are majority of blocks which gets refreshed within 10ms and hence there is no worry of them getting lost. There is a good probability that the hashed blocks in the figure can get kicked out from the cache by LRU replacement policy as they may not have been accessed in near past, and it is ok to loose them unless they are dirty. There could be some blocks in the same region which are frequently read and not written. In section 5, we describe how these types of blocks are dealt with. We also propose a scheme in Section 5 to handle the blocks which are in the region 10-40ms.

3. STT-RAM Design

4. Architecting Volatile STT-RAM

5. Experimental Evaluation

We evaluate our design and schemes using ALPHA M5 Simulator [] operating in Full System (FS) mode for PARSEC benchmark suite [] and in System Emulation (SE) mode for SPEC 2006 benchmark

suite. For performing experiments in FS mode, we model a 2 GHz processor with four out of order cores. We modified M5 simulator to model low retention time STT-RAM for last level cache. Last level cache is banked with different read and write latencies, with all the banks connected via a shared memory bus. We assume a fixed 400 cycles main memory latency for all our simulations. Table [] details our experimental system configuration.

We collected results for PARSEC benchmarks in the region of interest (ROI), skipping the initialization and termination phases. We warmed up caches for X million instructions in the ROI before collecting results. We assumed 4 threads are mapped to our modeled processor with four cores. For SPEC benchmarks, we collected results by fast forwarding 1B Instructions, warmed up 500M instructions and collected statistics for further 1B instructions.

6. Results

7. Prior Work

8. Conclusions

Sample bibliography [1]".

References

[1] Authors. Frobnication tutorial, 2011. Supplied as additional material `tr.pdf`. 7