AN ENERGY EFFICIENT CACHE DESIGN USING SPIN TORQUE TRANSFER (STT) RAM

A Thesis
Presented to
The Academic Faculty

by

Mitchelle Rasquinha

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2011

AN ENERGY EFFICIENT CACHE DESIGN USING SPIN TORQUE TRANSFER (STT) RAM

Approved by:

Professor Sudhakar Yalamanchili, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Hsien-Hsin Sean Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 20 July 2011

For Vincent Rasquinha and
Philomena Rasquinha:

Like a flower pressed between the pages of life
your fragrence permeates everythin I do
Mom and Dad...
this is for you.

ACKNOWLEDGEMENTS

Special thanks to my parents, Vincent and Philomena, my sister Rochelle and brother Mark, for giving me the encouragement, understanding, support and care.

I would like to express my utmost gratitude to my adviser Dr. Sudhakar Yalamanchili, for his support and advice in helping to bring this thesis to its completion. My committee members, Dr. Saibal Mukhopadhyay and Dr. Hsien-Hsin S. Lee for their helpful comments. Furthermore, my thanks go to my co-authors, Dhruv Choudhary and Subho Chatterjee for sharing their research interests with me.

Last but not least, thank you to my friends and all the people that have help me in this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The advent of many core architectures has coincided with the energy and power limited design of modern processors. Projections for main memory clearly show widening of the processor-memory gap. Cache capacity increased to help reduce this gap will lead to increased energy and area usage and due to small growth in die size, impede performance scaling that has accompanied Moore's Law to date. Among the dominant sources of energy consumption is the on-chip memory hierarchy, specifically the L2 cache and the Last Level Cache (LLC). This work explores the use of a novel non-volatile memory technology - "Spin Torque Transfer RAM (STT RAM)" for the design of the L2/LLC caches. While STTRAM is a promising memory technology, it has some limitations, particularly in terms of write energy and write latencies. The main objectives of this thesis is to use a novel cell design for a non-volatile 1T1MTJ cell and demonstrate its use at the L2 and LLC cache levels with architectural optimizations to maximize energy reduction. The proposed cache hierarchy dissipates significantly lesser energy (both leakage and dynamic) and uses less area in comparison to a conventional SRAM based cache designs.

CHAPTER I

INTRODUCTION

1.1 Introduction

The projected energy growth of multicore processors if left unimpeded will stall the performance scaling that has accompanied Moores Law to date. Among the dominant sources of energy consumption is the on-chip memory hierarchy, specifically the L2 cache and the Last Level Cache (LLC). This work explores the use of a new emerging non-volatile memory technology as a replacement for SRAM based L2 and LLC caches - Spin Torque Transfer(STT) RAM. Several recent publications [13], [12] have explored the use of STTRAM as a technology replacement for SRAM. This work illustrates the limitations of using STTRAM as a simple drop-in replacement for SRAM caches and develops and demonstates a circuit and microarchitecture co-design approach for an effective STTRAM cache hierarchy. Consequently and unlike

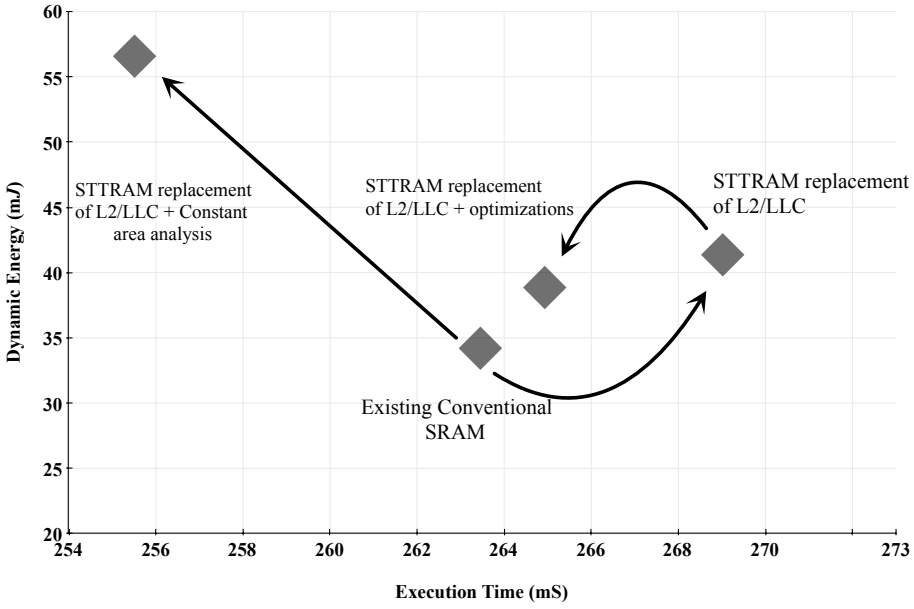

Figure 1: Impact of STTRAM caches on dynamic energy and execution time.

previous efforts, for a nonvolatile 1T1MTJ cell the latency and energy of a read operation is comparable to its SRAM equivalent while cell write energies and latencies remain only 2x higher - significantly better than previous approaches. The increase in dynamic energy and program execution times due to store operations is further mitigated by novel microarchitecture optimizations. The results can be summarized in Figure 1 which shows the average latency and dynamic energy consumed by applications when run on different system configurations. We see that drop-in replacement of an STTRAM of equivalent size produces increases in dynamic energy and latency due to the store operations while our microarchitecture optimizations tailored to STTRAM recovers most of the increase over SRAM maximizing the leakage energy gains. Constant area replacement produces significant gains due to lower miss rates but retains some increases in energy due to the increased energy of store operations although mitigated by the proposed optimizations. The remainder of this chapter discusses the motivation for STTRAM based caches.

1.2 The Memory Wall: Need for non-volatile memory

The memory hierarchy, an important component of computing platforms, has in the past been composed of SRAM based caches and DRAM for main memory. The hierarchy has proved a reliable cost effective solution for bridging the processor memory gap. With the onset of many-core architectures, the memory hierarchy has a renewed focus on pushing back the effects of the "the memory wall" a real barrier to the continued performance gains of computing platforms. This focus is driving new cache architectures that are both performance and energy effective. In the recent past DRAM scaling is becoming a serious bottleneck and various novel NVM technologies are being evaluated as suitable candidates for various levels of the memory hierarchy.

The processor-memory gap is primarily a result of the continued improvement in processor cycle speeds occurring at a much faster rate than DRAM access latency.

In [12] Wulf et.al. show how further increases in processor speeds will have little to no performance gains as memory access time becomes the primary bottleneck. The multilevel memory hierarchy is a direct outcome of minimizing the need to access main memory along the common execution path. While large caches with high hit ratios certainly minimizes the need to access memory, they are both power and area hungry. In modern many-core architectures the increased memory bandwidth demand forces much larger caches to be realized and this can lead to reduced performance under fixed area constraints.Performance scaling is now achieved via core scaling rather than frequency scaling. For a fixed die area, the goal is to maximize the number of cores which competes for die area with the cache hierarchy. Increasing the number of cores and reducing the aggregate cache size increases main memory demand and reduces performance. Further, larger caches are power inefficient primarily to the lower utilizations and growth in leakage power. The use of more efficient(in terms of area/bit) memory technologies and non-volatile properties promises to alleviate some of these issues and thereby improve core scaling.

In this thesis STTRAM, a new emerging NVM - Spin Torque Transfer(STT) RAM, is evaluated as a replacement for SRAM based L2 and LLC caches. In particular, the focus is on the interelationships and co-design of the circuit and microarchitecture to reap significant gains in energy efficiency.

1.3 Spin Torque Transfer(STT) RAM

This section provides a brief overview of STT RAM technology and the design of a STT RAM cell targeted for L2 caches and the LLC.

The structure of a STTRAM cell consists of a magnetic tunnelling junction (MTJ) connected in series with a transistor as shown in Figure 2 [9]. This cell is connected between the bit lines and the source lines whereas the word line is responsible for switching of the transistor. The MTJ consists of two ferromagnetic layers separated

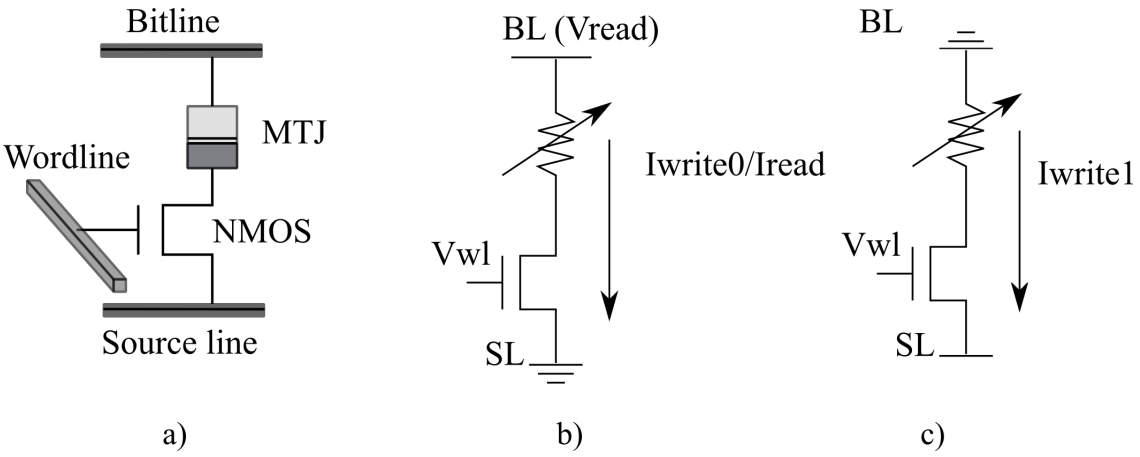

Figure 2: a) A Single STTRAM Cell b) STTRAM Cell Read Operation c) STTRAM Cell Write Operation

by a dielectric layer referred to as the spacer (usually MgO) The magnetization of one ferromagnetic layer is fixed. The magnetization of the other layer can be controlled by the injection of spin polarized electrons. Switching occurs if current greater than the critical value flows through the structure in the proper direction. The MTJ offers different resistances in the two modes of magnetization (parallel and antiparallel). For storage options parallel is taken to correspond to '0' and antiparallel corresponds to '1'. The resistance of the MTJ in the antiparallel state is higher than that in the parallel state. The difference between the resistances in parallel and antiparallel states with respect to resistance of the antiparallel state is called the Tunneling Magneto-Resistance (TMR). During read operations the bitline is pre-charged with a small voltage (Vread) and current flows in the direction of bitline (BL) to source line (SL). If the cell stores logic '1' (i.e. antiparallel state), the MTJ resistance is higher and hence, the read current is lower. On the other hand, if the cell stores '0' (i.e. parallel state) MTJ resistance is lower and hence the read current is higher. A high TMR indicates a larger difference between resistances in the parallel and antiparallel modes. The switching current required to change state is higher than the read current. In fact, it has been shown in [1], that the required MTJ switching current is a strong function of the write time; a smaller write time requires a higher MTJ switching

current (and therefore higher write energy). Hence, given a write time target, based on [1], one can decide the required MTJ switching current.

This operation of non-volatile STTRAM shows a critical difference between design targets for SRAM and STTRAM. The cell access timing in SRAM is primarily determined by the read delay (the time required to discharge the bitlines must be performed within a given clock cycle). This makes a write operation a non timing-critical cell access operation. However, for STTRAM the write operation takes much longer. Therefore, the STTRAM cells need to be designed to ensure a target write time and read time become non timing-critical cell access operations. The critical consideration for reading in STTRAM is the read margin. If the cell read current increases beyond the MTJ switching current, read disturb (flipping of the cell content while reading) failures can occur [5]. Hence, to reduce read disturb failures, the cell read current needs to be sufficiently smaller than the switching current, which in turn must be selected to provide sufficiently fast write times but with increased energy.

1.3.1 Technology Issues

The preceding is a conservative design in the following sense. First, the STTRAM can actually perform the read much faster than the SRAM because the lower cell area for STTRAM implies a much lower(than we have used) bitline/wordline capacitance for same sub-array size. Second, the switching current scaling method overestimates the switching current for an MTJ at 45nm node. This will directly reduce the write energy dissipation in the cell. Moreover, it will also reduce the required transistor size to support the switching current which will further lead to lower bitline/wordline parasitics and hence faster read operation, lower switching energy, and lower leakage energy for unselected cells in a selected column. Consequently, the results should be viewed as a conservative estimate as to the energy gains.

STTRAM is a back-end memory techology where the memory device is fabricated

in the Back-End-Of-Line and within the Inter-Layer-Dielectrics (i.e. in between metal layers). Hence, it does not require any change in the silicon bulk process. The memory is compatible with regular bulk silicon or SOI process and can use the existing silicon wafer. Further, MTJ can be fabricated in between low metal layers and hence, does not necessarily increase the requirement of metal resources in the memory. This also makes STTRAM compatible with the SRAM process. Consequently hybrid designs are possible. For example, fabrication of SRAM based tag is highly possible for an STTRAM array as all transistors can be fabricated on the same bulk silicon and the number of metal layers used in the entire array (STTRAM + SRAM Tag) is comparable to fully SRAM based array.

This thesis makes the following contributions

- Advocacy and creation of a microarchitecture-circuit co-design approach for a STTRAM based cache.
- Energy and performance models for STTRAM based caches on an optimized 1T1MTJ cell
- Microarchitecture optiizations are proposed for addressing the energy and latency properties of store operations of STTRAM caches — i) write biasing and ii) write cache.
- An exploration of the latency and energy behavior of STTRAM caches for various benchmarks using optimizaed cache cell designs. The optimizations reduce dynamic energy consumption in the STTRAM by an average of 8% ranging from 2% to 33%.

CHAPTER II

STTRAM BASED CACHE DESIGN

There has been several recent publications [13], [12] that explore the use of STTRAM as a technology replacement for SRAM. While this work also proposes an STTRAM cache, the specific array design is based on a circuit and architecture co-design approach that utilizes an optimized cell design that has improved access latencies and energy properties [8]. This chapter describes the circuit and micro-architecture co-design approach for the proposed STTRAM cache array design.

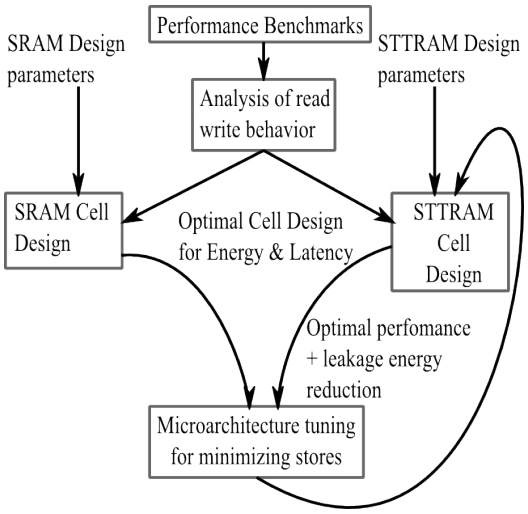

Figure 3: Methodology Flow-graph

Figure 3 describes the co-design approach adopted for the proposed optimizations. For a set of benchmarks, the performance at different cell access latencies was evaluated and compared to that of SRAM. For an STTRAM cell, write access speeds close to SRAM can be achieved, however this would require very high write currents. For a set of benchmarks the STTRAM cell access time was varied and the performance and energy per access tuned to match an equivalent SRAM cache. One of the important

Table 1: Summary of Energy Values for SRAM and STTRAM Cells

Cache	Memory Technology	Wordline Energy (pJ)	Read Energy/Cell (pJ)	Write Energy/Cell (pJ)	Leakage Energy/Cell (pJ)
L2 (64x128)	SRAM	69.68	6.032	16.955	6.635 x 10^-3
L2 (64x128)	STTRAM	20	1.25	26.88	0
LLC (128x128)	SRAM	69.68	6.225	44.225	6.635 x 10^-3
LLC (128x128)	STTRAM	20	2.45	26.88	0

factors for cell design is the ratio of read-write energy and the percentage of reads and writes in the application memory stream. This makes the optimal cell design dependent on application specific parameters such as the number of store operations.

2.1 STTRAM Cache Cell Design

The STTRAM cell design from [8] is discussed in this section. An SRAM cell at $45nm$ is first designed to achieve a read access time (considering wordline driver delay and bitline discharge delay) to be less than $250ps$ for a 64×128 or 128×128 sub-array to be used for the L2 and LLC respectively (described in Section 3.1. A target system frequency of $2Ghz$ is used and hence a single clock of $500ps$ is used. The total cell access time is divided between row-decoder delay and the sense-amplifier delay. We have ensured the cell timing is met even under worst-case $3\sigma V_t$ variations. The SRAM cell is designed first to meet the target delay for the 64×128 sub-array. The bitline and wordline capacitances are estimated based on the cell area of $80F^2$ (where F is the feature size) and the *wide cell layout* (the cell width along the wordline direction is two times the cell height along the bitline direction) with $2:1$ aspect ratio and $0.2fF/mm$ metal capacitance. The designed cell is used in the sub-array to estimate the read, write, and leakage energy as shown in Table 1.

The STTRAM cell is designed to ensure a target write time of $5ns$ based on acceptable current capacity and to stay within $10X$ of the SRAM read time. At the $180nm$ node this required $450\mu A$ or a current density of $2.6 \times 10^6 A/cm^2$. We considered the $2X$ area scaling model per generation for MTJ device and assumed constant current density. Our results were a close match with properties of fabricated

Table 2: Read/Write Energies for Single Line to SRAM and STTRAM caches

Cache	SRAM(pJ)		STTRAM(pJ)	
	Read	Write	Read	Write
DL2	12.38	128.17	124.06	261.78
LLC	283.7	284.76	282.26	549.06

Table 3: Access Latency in Clock Cycles for the SRAM and STTRAM caches

Cache	SRAM cycles		STTRAM cycles	
	Read	Write	Read	Write
DL2	4	4	4	13
LLC	7	7	7	26

MTJ devices at $65nm$ [4]. Based on this assumption, we predict $75\mu A$ of required switching current for $45nm$ MTJ device at $5ns$ switching time. Considering a 7% write margin (from switching current), we have selected a write current of $80\mu A$. Unlike previously published work for STTRAM caches, our design produces much lower energy for STTRAM read operations and a read latency that is equivalent to SRAM sub-array designs. Further, the proposed design balances the write energy and latency at approximately the knee of the energy-latency curve for STTRAM as reported in [1]. Given the cell design the per access energy/latency for a SRAM and STTRAM cache-line of varying sizes can be calculated.

2.2 L2 and LLC Caches

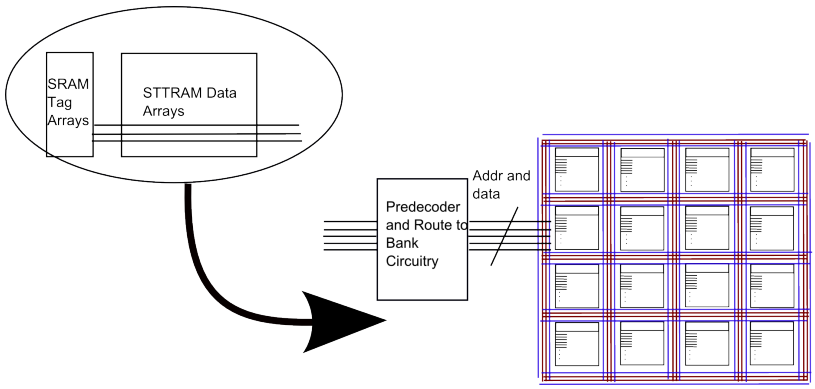

Figure 4: Cache Array structure

For the L2/LLC our design uses a $64/128 \times 128$ cell array building block interconnected to form a $1MB/4MB$ cache. The access energy includes the wiring to/from the banks and the energy of the peripheral components (pre-decoder, H-tree interconnecting the banks etc. as shown in figure 4). Cacti [5] was used to compute the energy of the peripheral components and the corresponding SRAM cache. The latency for a cache access is also computed similarly. The L2 and LLC cache design assumes the tags are realized using SRAM and the data array realized using STTRAM cell design.

CHAPTER III

ARCHITECTURAL OPTIMIZATIONS FOR IMPROVED CACHE ENERGY CONSUMPTION

This chapter presents an evaluation of the total cache energy consumption from a direct replacement of the SRAM caches and proposes a set of architectural optimizations that further improve the effective cache energy performance. The following is a description of the system evaluated and the effective improvement from a direct replacement of STTRAM based caches.

3.1 System Model

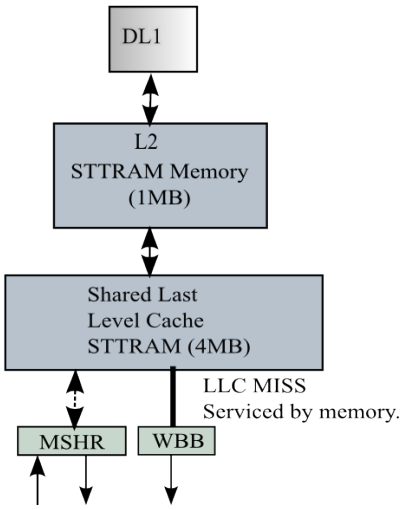

Figure 5: Baseline Cache Hierarchy

The system model comprises a processor configuration which represents a out-of-order (OOO) core. The cache configuration is described in Table 4, with the simplified model shown in figure 5. The OOO processor model has four cores each with separate data and instruction DL1 caches and a private L2. The cores share a last level cache

Table 4: Cache Configuration

Data DL1 Cache(256Kb)	64/32 sets, 4/8 way
L2 Cache (1MB)	512 sets, 8 way
Last Level Cache (LLC) (4MB)	4096 sets, 16 way
64 byte cache lines for all the caches	

(LLC). Store requests to the cache follow a write allocate with write-back cache policy. Due to limitations in the simulation environment no multi-threaded benchmarks were evaluated.

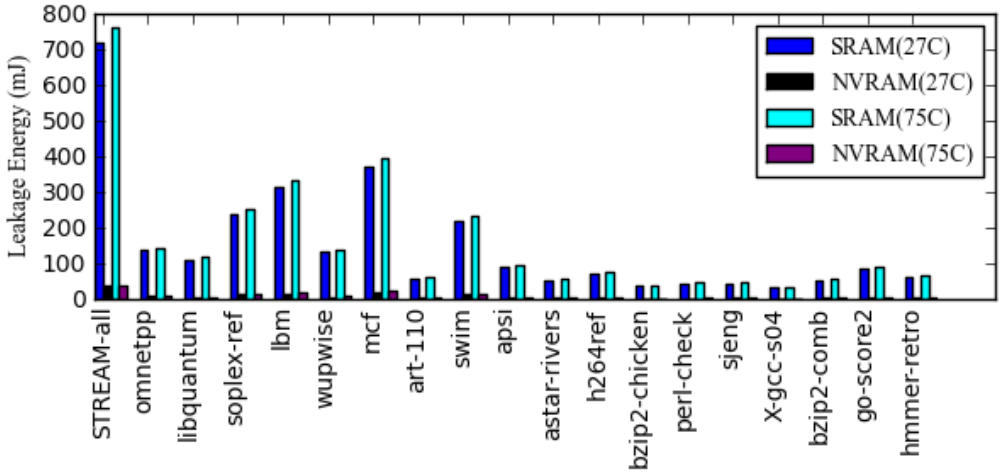

Figure 6: Comparison of Leakage Energy Profiles for SRAM and STTRAM Designs

By simply replacing the SRAM L2 and LLC with STTRAM designs based on the cell design from Chapter 2 the savings in leakage energy are, as expected substantial and shown in Figure 6. On average, there is a savings of 90% in leakage energy across the applications evaluated. A very different picture emerges for dynamic energy behavior as illustrated in figure 7. Dynamic energies increase on an average by 60% for memory intensive benchmarks and 30% for compute intensive benchmarks.

3.2 Architectural Optimizations

As a result of the preceding analysis the architectural run-time optimization goal is focused on reducing the total dynamic energy spent in the cache due to the increase in energy consumption of write operations (stores in the memory reference stream).

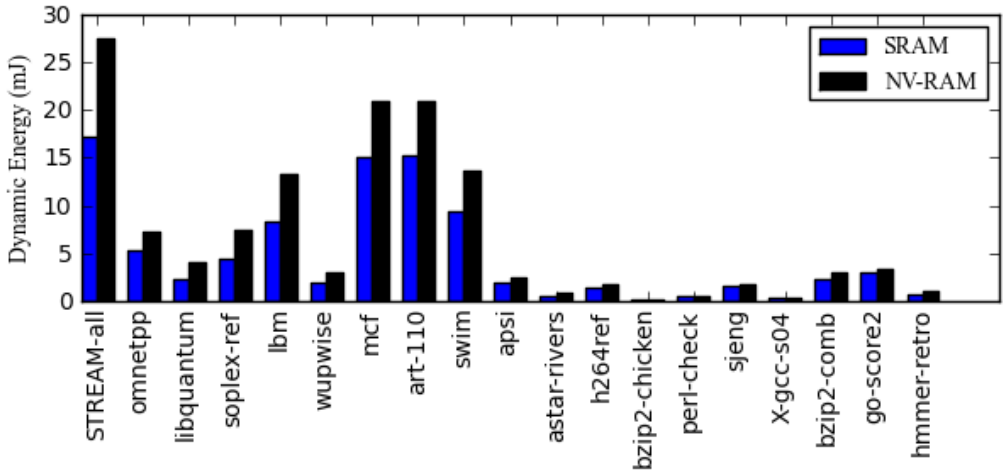

Figure 7: Comparison of Dynamic Energy Profiles for SRAM and STTRAM Designs

Among the various type of cache accesses only certain types of accesses contribute to the increase in dynamic energy. For example, a L1 cache access may be a load/store hit to the L1, a load/store miss for which there was a lookup in the L1, or a L1 eviction of a dirty line (L1 write back). Only the last one is a write to the L2. The other source of writes to the L2 is the load/store miss allocation for a line, which is critical both from an energy and latency viewpoint. Run-time microarchitecture optimizations to address these issues are described next.

3.2.1 Analysis of store behavior

The architectural optimizations are focused on reductions of the store operations to the STTRAM caches. We first analyze the behavior of store operations to the L2 and develop a characterization to drive subsequent optimizations.

Consider the memory access stream for store operations to the L2 cache. Figure 8(a) illustrates a 3-D histogram of the inter-reference time (IRT) of the store operations to the L2, for a compute intensive benchmark - SPEC06.h264 with the standard least recently used replacement policy applied to the L1 cache. The two major sources of stores to the L2 are write-backs from the DL1 and L2 misses which

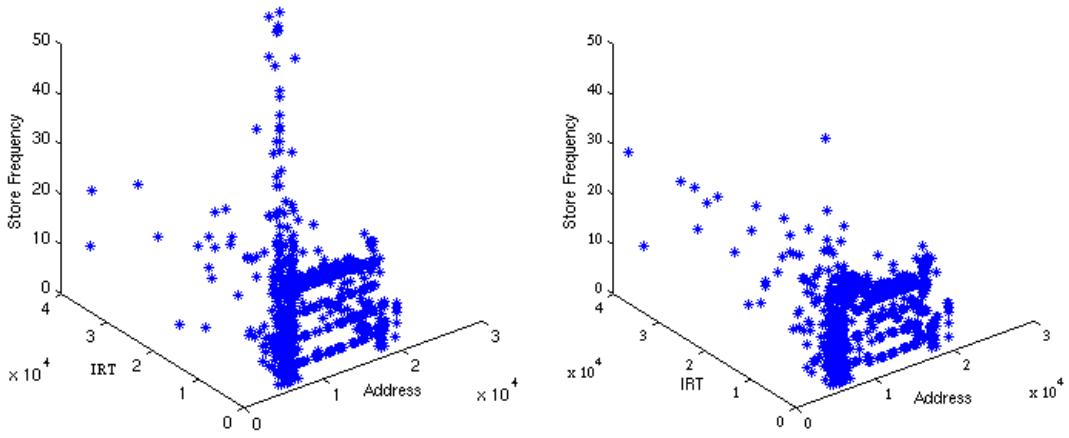

Figure 8: Cache access pattern for store operations (a)LRU (b)Write Biasing

fetch a line from the lower level. *The key idea is to increase the residency of lines that are dirty in the cache*, so that successive store operations can be coalesced - they hit in the cache and avoid a cycle of eviction and read back on a future miss to the same line. The benchmark programs were analyzed to understand the store-to-store inter-reference time to the same address and the number of intervening store operations to unique addresses. These two measures were then used to drive the development of the tunable cache replacement algorithms to retain dirty lines just long enough to maximize coalescing thereby minimizing STTRAM store energy. Importantly, to facilitate formulation of the optimizations of store behavior, we introduce a finer classification of store operations.

- Stores to dirty lines referred to as store-to-store or s2s.
- Stores to clean lines referred to as store-to-load or s2l.
- Load from a dirty line referred to as load-to-store or l2s.

The importance of the classification stems from their use in exploiting the preceding behaviors in reducing store operations to the L2 and LLC. Finally, if a store is the

last reference before a line becomes dead (never referenced again before eviction), we will refer to that store as a dead store. Otherwise the store is live. We now propose two optimizations for minimizing store operations to the L2 and LLC.

3.2.2 Write Biasing

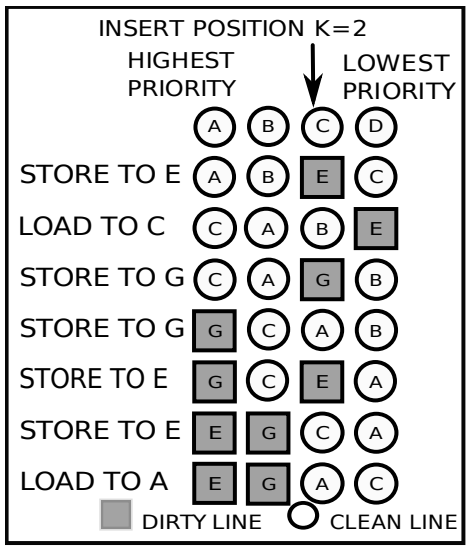

Figure 9: Write Biasing Cache Line Replacement Algorithm

We propose a novel line replacement algorithm for L1/L2 sets that biases replacement in favor of dirty lines to increase the residency of dirty lines even at the expense of increasing read misses to the next level (since store operations are 2X more energy expensive than read operations). In conventional Least Recently Used(LRU) replacement policy, the most recently referenced line on a hit or a miss is placed at the top of the reference stack (TOS). We proposed new differential promotion and insertion policies that *operate differently for loads and stores*. The combination of these new insertion/promotion policies is referred to as *write biasing*. We define a parameter K which is the distance from the top of the LRU stack at which a line is inserted or promoted to. The following implementation is referred to as Write biasing-base (WB-base).

- *Insertion policy*: Target lines of all load misses are inserted at distance K from

the TOS. Target lines of all store misses are inserted at TOS.

- *Promotion policy*: On a hit, target lines of stores are promoted to the TOS. Target lines of loads are promoted to a distance K from TOS.

- *Eviction policy*: The line at the bottom of the stack is evicted.

WB-base has two major drawbacks. First, the combination of insertion and promotion policies leads to increased miss rates for loads. If there is no dirty line in a set, all entries from TOS to position K will be vacant because clean lines cannot be promoted past the position K. Thus we suggest a modification to the insertion and promotion policies for loads.

- *Insertion policy for Loads*: Target lines of loads are inserted at position K if all stack positions higher than K are dirty. Otherwise it is inserted at TOS similar to LRU.

- *Promotion policy for Loads*: Target lines of loads are promoted to position K if all stack positions higher than K are dirty. Otherwise it is promoted to TOS similar to LRU.

Second, one-time store operations (dead stores) can cause dirty lines to occupy stack positions higher than K for very long periods unless replaced by another store operation. This artificially inflates the load miss rate, which increases energy due to higher load miss allocation. Thus we suggest a modification to the insertion and promotion policy of stores that helps act as a dead store filter.

- *Insertion policy for Stores*: Target lines of stores are always inserted at position K similar to loads.

- *Promotion policy for Stores*: Target lines of store hits to dirty lines are promoted to TOS. Store hits to clean lines are promoted to position K.

Figure 9 shows an example of the steps in write biasing with all the policy modifications suggested. We utilize write biasing in the L1 and L2. The use in L1 is particularly effective as avoiding a single store due to eviction from L1 to the L2 saves $261pJ$ (write energy per access to L2). We should point out that write biasing itself is not well suited for SRAM caches where there is no disparity in the read and write energies and therefore nothing to be gained by simply inflating the read miss rate. It can be seen from figure 8(b) how the peaks in the memory access stream is reduced due to write-biasing.

3.2.3 Write Cache

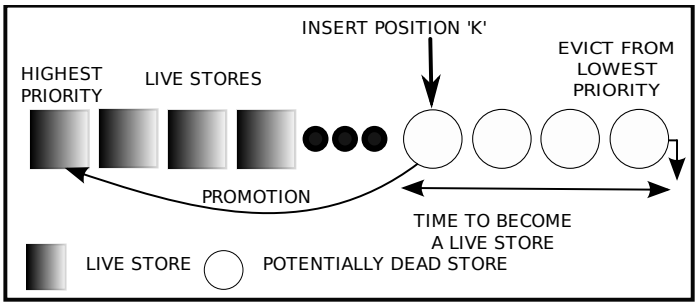

Figure 10: Write Cache Replacement Policy

Several benchmarks exhibited higher store inter-reference times. To further extend the ability to coalesce stores across these situations we propose a second optimization - the addition of a small write cache between the L1 and L2. The write cache (WC) is a fully associative 32/64 way cache that contains only dirty lines evicted from the L1; whose contents are mutually exclusive with the L2; and is accessed in parallel with the L2 with a hit in one inhibiting access from the other (we do account for the energy of parallel look-ups). On a store, if a line is present in L2 it is transferred to the WC. Lines evicted from the WC are sent to the L2 and inserted as dirty lines. The WC and the L2 share the MSHR and thus it allows us to allocate store misses exclusively in the WC whereas load misses are allocated in the L2. The insertion and eviction policy for the WC is as described in Figure 10. We insert at a fixed position

K from TOS in the replacement stack of the 32-way cache. On a conflict we evict the line lowest in the stack. When a store hits in the WC (live store), it is promoted to the top of the replacement stack. Load hits do not change priority order. The value of K is a tradeoff between the number of live stores we want to place in the WC, to the amount of time we want to give to a dirty line to get a store hit(recognize it as a live store) before it is eventually evicted from the bottom of the stack assuming its a dead store.

CHAPTER IV

PERFORMANCE EVALUATION

All experiments report the energy spent in the L2 and the LLC to execute 250 million instructions. The following evaluation and analysis is structured around understanding two phenomena - write biasing policies in the L1 and L2 data cache, and the behavior of the WC. In particular the interactions between these two optimizations produces subtle migration patterns of lines in response to which we further refine our optimizations as described. Whenever a store from the L1 hits in the L2, the line is migrated to the WC causing high traffic in the WC. Thus we need to refine the migration pattern. Write biasing in the L2 promotes live stores higher in the stack, thus filtering out dead stores. Thus we modify the policy to migrate a line to the WC (from the L2) only if it is in the *Most Recently Used* (MRU) position in the L2. We call this *Selective Write Cache Migration* (SWCM). We assume SWCM in all results presented. Write biasing in the L1 determines the number of stores placed in the L1 and the duration. This in turn determines the traffic going into the WC. High store miss rate in L2 leads to very high traffic in the WC because all store misses are allocated in the WC.

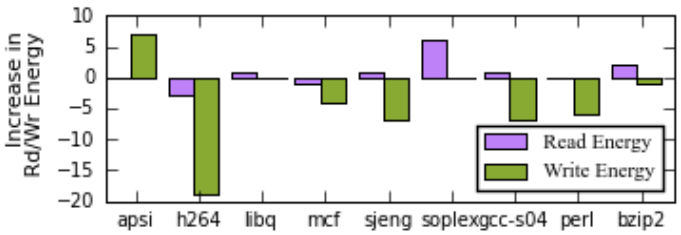

Figure 11: Impact of Write Biasing on Read/ Write Energy of DL2

Figure 11 shows the results of write biasing in L1 with K=1 (without the WC

optimization). As can be seen, write biasing tends to increase L1 miss rate and hence increases the read energy of the L2. However the 2X factor of write energy and reduction in store operations helps offset most of this energy increase. This behaviour can be seen in benchmarks such as sjeng and gcc. Benchmarks like h264 have a high store miss rate. Thus keeping stores in L1 helps the overall miss rate, consequently decreasing both the L2 read and write energies.

Figure 12 shows the simulation results for the best performing configurations of write biasing in L1 and L2 with the WC. The results assume the given policy nomenclature - K value in L1 : K value in L2 : K value in WC. eg. $k1\_k2\_k16$ implies K=1 in L1, K=2 in L2 and K=16 in WC. The results reflect the exploration of the degree of biasing represented by the selection of the value of K at each level. The WC employs an insertion policy parameterized by K - the insertion point in the LRU stack. Considering a 32-way WC, we analyzed values of K from 12 to 24 and determined that the value of $K = 16$ was the most effective. Increasing the size of the WC beyond 32 entries provided little additional benefit. Figure 12 also shows how each of the optimizations compare to a base configuration of SRAM caches.

All the policies described are simple local decisions that have minimal hardware overhead. We use write biasing in upto 16 way caches. In general a K value of 2 performs well. Thus additional hardware cost involves a K-input AND operation per set for performing the logical AND of the dirty bits of the K ways in the set (512 2-input AND gates in our example). We now try to present some insights into these policies.

4.0.4 Benchmark Analysis

The most effective configuration for a majority of the benchmarks is $k1\_k2\_k16$. Some benchmarks do well under $k2\_k4\_k16$ because these exhibit high store miss rates. Thus higher value of K increases residency of dirty lines, thus improving the store

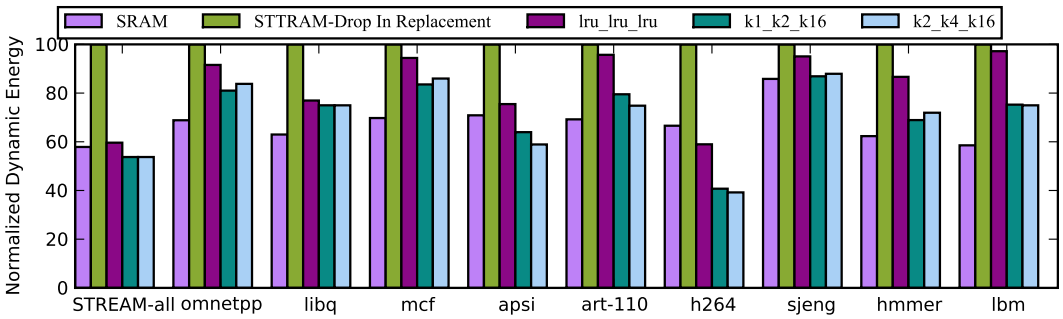

Figure 12: Energy reduction with Write cache and Write biasing.

miss rate and reducing traffic to the WC. Memory intensive benchmarks like *omnetpp* and *mcf* do not have high store miss rates, but still have very high traffic to the WC. Thus *lru_lru_lru* does not do well in such applications. This is where *Selective Write Cache Migration* is effective in filtering dead stores. Benchmarks with good temporal locality of stores such as *h264* show dynamic energy reduction as much as 60% due to high store coalescing. The same phenomenon is graphically visible in Figure 8.

At one extreme libquantum recorded only 12 store hits in the write cache indicating large store inter-reference time. Due to the selective write cache migration in the L2 line migration to the write cache is avoided. It should be noted though that we still record energy savings of under 2% in libquantum due to the fact that many stores that would have ordinarily been evicted from the L2 into the LLC(due to the large working set size of benchmark [2]), are now kept in the L2 due to write biasing. Saving a write into the LLC STTRAM which is very expensive in energy and latency.The other end of the spectrum we have applications like spec00.art which are memory intensive but have small inter-reference times between stores.

4.0.5 Interaction of Write biasing and Write cache

All the policies described are simple local decisions that have minimal hardware overhead. We use write biasing in upto 16 way caches. In general a K value of 2 performs well. Thus additional hardware cost involves a K-input AND operation per set for performing the logical AND of the dirty bits of the K ways in the set (512 2-input

AND gates in our example). We now try to present some insights into these policies. The replacement policies for the three caches - L1,L2 and WC interact with each other in the following way.

- Write biasing in L1: Moderate intensity of write biasing in the L1 cache is effective in decreasing WC traffic. It potentially increases the store residency in the WC, thus increasing chances of coalescing stores that have relatively larger inter-reference times.

- Write biasing in L2: Helps as a dead store filter. By migrating live stores to higher positions in the replacement stack, it allows the implementation of SWCM. Thus one time stores get filtered, giving opportunity for live stores to occupy the WC for a longer time.

- Latency impact of Write biasing: Our optimizations have negligible impact on performance as demonstrated in Figure 1. Even though we increase the misses, we recover some of the performance loss because we reduce high latency stores.

CHAPTER V

RELATED WORK

The most relevant work is a sequence of papers by Wu et.al. [11, 10] that explores a philosophy of partitioned caches - fast SRAM partitions coupled with slow STTRAM partitions. While we do not address the packaging impact as they do, in that our approach is fundamentally different in its design approach in that we start with the circuit design of STTRAM cells guided by cache access requirements to push cell design towards a specific target. Consequently, the physical properties of our STTRAM design is very different from that assumed in those studies. Notably our designs have read latency and read energy in comparison to SRAM. Moreover, our write latency and write energies are also significantly better than the values assumed in those studies. This substantive narrowing of disparity between SRAM and STTRAM properties by our work is a major contribution and leads to changes in the tradeoffs that will admit different architectural solutions. More recently Zhou et.al [6] proposed a novel circuit optimization that terminated write operations to an STTRAM array if the contents matched the write value thereby saving the substantive write energy. This technique complements our proposal by relying on value locality. Finally, there has been a burgeoning of efforts in the study of phase change memory (PCRAM) [7, 3]. These studies have largely focused in the use of PCRAM as a potential DRAM replacement in main memory systems rather than caches.

CHAPTER VI

CONCLUDING REMARKS

This work explores optimizations for energy efficient STTRAM L2 and LLC caches beginning with a clock-rate driven cell design and supported by two microarchitectural optimizations to recover most of the dynamic energy increase due STTRAM write operations thereby maximizing leakage enery gains of the technology. The design approach can be used to to tailor cache designs for specific datapath environments and their workload characteristics. Furture work will focus on furthering the energy gains via adaptive write biasing schemes to match workload variations.

Some key insights from this work are as follows:

- Relationship of L1, L2, and Write cache management policies pivotal for energy reduction.
- Proper tuning of replacement policies to reduce high energy high latency stores
- Policy decisions made to reduce system energy of large structures. Example: DL2 Write biasing for LLC energy reduction, Write cache and DL1 biasing for DL2 Energy Reduction
- Latency reduction equally important
- All management policies are local decisions and hence do not access any predictors or state information like counters that may be energy inefficient

LIST OF PUBLICATIONS

[1] Chatterjee, S., Rasquinha, M., Yalamanchili, S., and Mukhopadhyay, S., "A scalable design methodology for energy minimization of sttram: A circuit and architecture perspective," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –9, 2010.

[2] Chatterjee, S., Rasquinha, M., Yalamanchili, S., and Mukhopadhyay, S., "A methodology for robust, energy efficient design of spin-torque-transfer ram arrays at scaled technologies," in *Proceedings of the 2009 International Conference on Computer-Aided Design*, ICCAD '09, (New York, NY, USA), pp. 474–477, ACM, 2009.

[3] Rasquinha, M., Choudhary, D., Chatterjee, S., Mukhopadhyay, S., and Yalamanchili, S., "An energy efficient cache design using spin torque transfer (stt) ram.," in *ISLPED'10*, pp. 389–394, 2010.

REFERENCES

[1] Hosomi, M., Yamagishi, H., Yamamoto, T., Bessho, K., Higo, Y., Yamane, K., Yamada, H., Shoji, M., Hachino, H., Fukumoto, C., Nagao, H., and Kano, H., "A novel nonvolatile memory with spin torque transfer magnetization switching: spin-ram," in *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, pp. 459–462, Dec. 2005.

[2] Jaleel, A., "Memory characterization of workloads using instrumentation-driven simulation," 2009.

[3] Lee, B. C., Ipek, E., Mutlu, O., and Burger, D., "Architecting phase change memory as a scalable dram alternative," in *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*, (New York, NY, USA), pp. 2–13, ACM, 2009.

[4] Leuschner, R., Klostermann, U., Park, H., Dahmani, F., Dittrich, R., Grigis, C., Hernan, K., Mege, S., Park, C., Clech, M., Lee, G. Y., Bournat, S., Altimime, L., and Mueller, G., "Thermal select mram with a 2-bit cell capability for beyond 65 nm technology node," in *Electron Devices Meeting, 2006. IEDM '06. International*, pp. 1–4, Dec. 2006.

[5] Muralimanohar, N., Balasubramonian, R., and Jouppi, N. P., "Architecting efficient interconnects for large caches with cacti 6.0," *IEEE Micro*, vol. 28, no. 1, pp. 69–79, 2008.

[6] P. Zhou, B. Zhao, J. Y. Y. Z., "Energy reduction for stt-ram using early write termination," in *International Conference on Computer-Aided Design*, pp. 1–4, Nov. 2009.

[7] Qureshi, M. K., Srinivasan, V., and Rivers, J. A., "Scalable high performance main memory system using phase-change memory technology," in *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*, (New York, NY, USA), pp. 24–33, ACM, 2009.

[8] Rasquinha, M., Choudhary, D., Chatterjee, S., Mukhopadhyay, S., and Yalamanchili, S., "An energy efficient cache design using spin torque transfer (stt) ram.," in *ISLPED'10*, pp. 389–394.

[9] transfer torque switching in magnetic tunnel junctions, S. and spin-transfer torque random access memory *Journal of Physics*.

[10] Wu, X., Li, J., Zhang, L., Speight, E., Rajamony, R., and Xie, Y., "Hybrid cache architecture with disparate memory technologies," in *ISCA '09:*

Proceedings of the 36th annual international symposium on Computer architecture, (New York, NY, USA), pp. 34–45, ACM, 2009.

[11] Wu, X., Li, J., Zhang, L., Speight, E., and Xie, Y., "Power and performance of read-write aware hybrid caches with non-volatile memories," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, (3001 Leuven, Belgium, Belgium), pp. 737–742, European Design and Automation Association, 2009.

[12] Wulf, W. A. and Mckee, S. A., "Hitting the memory wall: Implications of the obvious," *Computer Architecture News*, vol. 23, pp. 20–24, 1995.