A Review of STT-RAM, SRAM, and eDRAM and Methods of Optimization for Computer Architecture

Alan Hernandez
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

Abstract—The following Capstone Report seeks to outline difference in cache designs but more thoroughly into the computer architecture of STT-RAM, SRAM, eDRAM. It begins by outlining the use of cache followed by the different protocols for implementation including: direct mapping, fully associative, and set associative configurations. An area of interest in this study is implementing STT-RAM over SRAM by partitioning the second cache level in two. The method to which this was done and specialized resulted in an increase performance in the PARSEC benchmark. eDRAM cache designs were also improved by using dead-line prediction in one case demonstrating an improved energy efficiency. Overall each cache configuration selected by the research groups showed an increase in performance and energy efficiency in most cases.

Keywords—STT-RAM, eDRAM, SRAM, Cache Latency Non-Volatile Memory, Retention-Relax Design, Process Variation Aware Non-Uniform Cache Access, Direct Mapping, Fully Associative Mapping, And Set Associative Mapping

I. INTRODUCTION

Cache is an important part of modern day computing. It's very purpose is to address differences in processor speed to the system's RAM speed. Storing program instructions to this median between the CPU and RAM reduces the calls to RAM, which has a higher execution time. Cache offers a reduction in energy used throughout the system as well when compared to the energy that would otherwise be consumed by the processor reading from RAM.

Each level of cache serves a specialize purpose. Level one cache (L1) is nearest to the CPU providing the fastest read time of program instructions but with a smaller capacity of storage to higher levels of memory. Level two (L2) cache is shared among the different CPU cores which a larger memory capacity than L1 cache. Level three cache (L3), also known as final level cache, is the bridge between memory access of L1 and L2 but with slower speeds but still faster than RAM with an even greater memory capacity to that of L2.

Cache configurations for memory can consists of direct mapping, fully associative mapping, and set associative mapping. Direct mapping access one contiguous set of bits, or a block, to a predetermined cache location. Much like a table consisting of rows and columns, the data block is inserted one by one into the cache line. In fully associative mapping, a block can be mapped to any location on cache in contrast to the direct mapping method. The median between these methods is set associative mapping in which a block is placed to a subset of cache position. Non-volatile memory such as STT-RAM, is considered for long-term consistent storage during which does not lose its data when power is lost on the system. Volatile memory such as SRAM and eDRAM on the other hand immediately loses all data stored when power to the system is lost.

When comparing different cache configurations one important detail to analyze is the cache miss ratio and hit ratio. To begin with, a cache hit is when the CPU requests data assumed to be stored in cache is found. When the data is not found in cache it is cache miss. The cache hit ratio is the total number of cache hits found divided by the total number of cache misses. The cache miss ratio is defined as one minus the hit ratio. These values are important to help determine the performance of cache on a system.

The sections to follow consists of the literature review on the work and research done to analysis different for designs since the year 2000 for cache and methods of optimization for STT-RAM, eDRAM, and SRAM. The data section casts a visual representation in cache latency and energy consumption among the different types of cache designs. Finally an overview of this capstone in the conclusion section.

II. LITERATURE REVIEW

1. Studies are performed on multi-ported shared-memory multiprocessor for Concurrent-Read- Concurrent-Write access (CRCW) and Concurrent-Read-Exclusive-Write access (CREW)[1]. Different protocols such as snooping-bus with cash-to-cache transfers and other performances were evaluated based on hit rate, fraction of shared data, and other performances. In conclusion snooping-bus with cache-to-cache transfer was indicated to be reliably faster than other protocols. Other incomplex methods can be implemented to achieve similar results. Memory use and augmenting port complexity is further covered[1].

2. Analysis of reliable Last Level Cache (LLC) designs are sought. Known issues consisting of a Single Event Upset led to suggesting the eDRAM
Bit Upset Vulnerability Factor (BUVF)[2]. Benchmark test were performed with the aid of an algorithm developed by the group for soft errors. Portions of the eDRAM refresh cycle was thoroughly tested and results showed a negligible difference in Read-Read access. This directed the group to reassign the issue of reliability as pursuit for decreasing BUVF[2].

3. An in depth view of Spin-Transfer Torque Random Access Memory (STT-RAM) is covered. Progress in the development of Multi-Level Cell (MLC) allowed for experiments seeking to optimize the performance of MLC in soft bit restoration (RRD)[3]. Results were improved RRD and increased energy efficiency in the system overall. This groups progress and solution in Adaptive Overwrite Scheme (AOS) allowed for an alternative in L2 design when implementing MLC[3].

4. Proposal for increasing cache consists of implementing STT-RAM instead of Static Random Access Memory (SRAM)[4]. The cache is to be partitioned in two as for one partition to handle standard L2 cache request while the other handles frequently accessed cache blocks. PARSEC benchmark indicated increased performance overall[4].

5. STT-RAM has risen to compete against SRAM with its non-volatility and low leakage. However the drawback is its high write latency and usage of energy[6]. This group’s proposal seeks to use its non-volatility and low leakage to address this issue and an optimize retention-time in the cache hierarchy[6].

6. An overview in the performance, application, and enhancement of chip-multiprocessor (CMP) [7]. It focuses on the third level L3, or last level of cache hierarchy as more data sharing occurs on this level. Their proposal resulted in a shared 32MB last-level cache able to surpass a 32MB private cache design in performance[7].

7. By comparing SRAM, STT-RAM, and eDRAM in their strengths and weaknesses it provides a baseline to which to seek improvement in each design[8]. Although the primarily focus on eDRAM and using dead-line prediction resulted in an energy-efficient last level cache configuration for modern applications in eDRAM[8].

8. Improvements on performance and cache energy use were resulted on STT-RAM using a conflict-reduction mechanism along with conservative promotion and aggressive pre-diction algorithms were tested[9]. This was possible by applying a process variation aware non-uniform cache access (PVA-NUCA) method for bulky cache designs[9].

III. DATA ANALYSIS

As shown visually below in Figure 1, STT-RAM appears to be the most energy consuming in most cases when compared to SRAM and eDRAM with the exception of the STT-RAM used in Chang[8]. When comparing read latency, each case appears each type of random access memory to be near the same latency value.

![Energy Comparison](image1.png)

**Fig. 1.** Energy comparison: STT-RAM/SRAM/eDRAM

![Read Latency Comparison](image2.png)

**Fig. 2.** Read latency comparison: STT-RAM/SRAM/eDRAM

IV. CONCLUSION

This study's objective is to compare different cache configurations and methods to optimize their performance and energy efficiency. Analyzed in great detail are STT-RAM, SRAM, and eDRAM configurations. They were tested with different benchmarks, and proposals by other research groups seeking to enhance their performance. The results of their work are shown in the literary review section along with visual representations of read latency in Figure 2. Although there were only three main focuses to this study, it outlines different approaches since the year 2000 and possibly future methods to which improve modern cache designs in both performance and energy consumption.
REFERENCES


<table>
<thead>
<tr>
<th>Parameters for the below techniques</th>
<th>Processor</th>
<th>Level 1 (L1) for Instruction (I) or Data (D)</th>
<th>Level 2 (L2)</th>
<th>Level 3 (L3) or Last Level Cache (LLC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun [1]</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>CREW</td>
</tr>
<tr>
<td>Khoshavi [2]</td>
<td>8</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
</tr>
<tr>
<td>Chen [3]</td>
<td>4</td>
<td>3.3GHz</td>
<td>32KB</td>
<td>8-way</td>
</tr>
<tr>
<td>Motlagh [4]</td>
<td>8</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
</tr>
<tr>
<td>Lin [5]</td>
<td>2</td>
<td>800MHz</td>
<td>32KB</td>
<td>N/A</td>
</tr>
<tr>
<td>Jog [6]</td>
<td>4</td>
<td>2GHz</td>
<td>32KB</td>
<td>4-way</td>
</tr>
<tr>
<td>Jaleel [7]</td>
<td>8</td>
<td>N/A</td>
<td>32KB</td>
<td>4-way</td>
</tr>
<tr>
<td>Chang [8]</td>
<td>8</td>
<td>2GHz</td>
<td>32KB</td>
<td>8-way</td>
</tr>
<tr>
<td>Sun [9]</td>
<td>4</td>
<td>2GHz</td>
<td>32KB</td>
<td>4-way</td>
</tr>
<tr>
<td>Jokar [10]</td>
<td>4</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
</tr>
</tbody>
</table>

“CL” = Cache line
Calculation for “# of CL” columns:
Manually compute the number of cache lines given the capacity value as listed in capacity column, assuming the cache line size is always 64 bytes

Protocol column = {Write Back (WB), Write Through (WT), MESI, MOESI, Not Available (N/A)}