Implementation of Cache Designs and Efficiency of Cache Memory

Ronak Patel
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

Abstract—Several cache configurations and cache designs are used in an effort to create an efficient cache memory that bridges the communication between the CPU and main memory. The objective is to discuss different cache configurations or strategies (direct-mapped cache, set-associative cache, fully associative cache) and designs (STT-RAM, SRAM, eDRAM) in order to evaluate and understand energy and cache latency metrics; studying the different manners in which different configurations store data. Trends will be found and discussed based on the energy and cache latency metrics of STT-RAM, SRAM, and eDRAM designs which are based on the configurations of some different cache systems from the research over recent years.

Keywords—STT-RAM, SRAM, eDRAM, Volatile memory, Non-Volatile memory, Hit ratio, Miss ratio.

I. INTRODUCTION

Cache generally has three types of configurations, Direct-Mapping, Set-Associative, and Fully Associative. These different configurations each result in different performance, and each have their own ways to store data. Multilevel caches are important as having more than one level of cache memory helps reduce required energy and increases the time taken to complete a task, therefore improving efficiency [3]. There are different types of cache block placements like direct-mapped, full-associative, and set-associative. While using direct-mapping strategy a block can be placed at only one place in the cache. In full-associative, the block can be placed anywhere in the given set in cache. There are two different types of memory that a computer uses; Volatile (SRAM, eDRAM, SDRAM) and Non-Volatile memory (STT-RAM, PRAM). Volatile memory requires electricity to be functional, whereas Non-Volatile memory does not.

In direct-mapping strategy, each memory location is mapped directly to only one location in the cache. A typical way of mapping address and cache locations in direct-mapping strategy is by using the formula

(Block address) mod (Number of cache blocks in cache)

This equation helps in finding the lower bits of the block address to map to the cache location. For example, if the block address is 101102 and there are 8 number of cache blocks (8 words), then 101102 mod 8 = 1102. The resulting value 1102 (lower 3 bits) is the assigned cache block or the cache index; the value in memory address 101102 is directly-mapped to cache location 1102. In order to put the newly requested item from the processor in a cache block, the cache block must be already empty or the existing item should be replaced by the newly requested item. Each cache index may contain the contents of several different memory locations, in order to justify that the newly requested word corresponds to the data in cache or not, we use tags. Since the lower bits of memory address were used to assign cache index, we use the upper bits to assign tags if the requested word was found in a cache location (miss), or if the requested word was found in a memory location (hit). Referring to the example discussed previously, if the CPU requested a word stored in memory address 101102 then the tag value of the address will be 102[1].

In set-associative strategy, each block can be placed in at least two or more fixed number of locations in cache. While a set-associative cache has n number of locations for a block to be placed, this cache is called a n-way set-associative cache. This cache has certain number of sets in which n blocks reside. Every single block in the memory is mapped to a unique set in the cache represented by index, where a block can reside in any element in the set. To represent a set containing a memory block in a set-associative cache, we use the formula

(Block address) mod (Number of sets in cache)
The cache memory in the CPU is divided into three components; SRAM, Tag RAM (TRAM), and the Cache Controller. These components can be used on multiple chips or a single chip. Where SRAM (Static Random Access Memory) is a cache component that holds data, and its size determines the cache size. It is called static because the information stored in the SRAM will remain in it until the computer is turned off. Typical access times by SRAM are fast; 0.5 to 2.5 nsec. SRAM consists of only 6 transistor cells making it a low-density component. The TRAM is a component of SRAM that stores the addresses of each data stored in SRAM.

The Cache Controller is responsible for updating SRAM and TRAM and implementing the write operation, it also determines if a request made by CPU is a cache hit or miss. So, hit ratio is the fraction of memory accesses found in the cache memory, whereas the miss ratio is the fraction of memory access found in main memory; \(1 – \text{hit ratio}\).

The forthcoming sections will look at specific metrics from ten different cache configurations. Data will be analyzed to see how different metric change over time, this will also be represented in graphs.

II. LITERATURE REVIEW

For each cache configuration there are several metrics that can be studied, however we will check for energy required for read and write operations. From the total of five baseline computer systems, we will check four computer systems for energy metric in [2], [3], [4], and [5]. From the comparison design we will check for energy metric in [6], [7], [8], [9], [10], and [11].

STT-RAM, SRAM, and eDRAM are widely used fabrication technologies today. Most caches today are still based on SRAM (a universal memory replacement in multi-core systems [6]), however the STT-RAM is becoming popular as time progresses because of its appealing properties that improve cache memory [3].

Spin-Transfer Torque Random Access Memory (STT-RAM) is a non-volatile memory recognized to have more advantage over other technologies because of its high density, very low leakage power, good scalability, zero standby power, and radiation hardness [7]. The STT-RAM can also help increase the capacity in the Last Level Cache (LLC) in latency and power efficiently [9]. The disadvantages of STT-RAM compared to SRAM and eDRAM are its long write time and high write energy [8]. Different architecture based solutions have been proposed to overcome the problem of write performance such as hybrid caches bypassing writes [11].

Modern day processors usually have SRAM based caches because of its fast memory access, but SRAM may not be the best technology when running critical and energy consuming applications because of its high power leakage which is caused by their large cells which have only 6 transistors [8]. The SRAM has stopped evolving in the sense that there is no room for improvement even after a lot of research. Therefore, STT-RAM has become a replacement of SRAM because of its high efficiency [3].

eDRAM uses some form of capacitor to store the data, which means that just like capacitor loses charge over time, it loses data over time. So, the main disadvantage of eDRAM is that in order to prevent data loss, it requires a refresh. So every time the threshold value is hit, the capacitor receives current to stay charged and keep the data. However, the advantages of eDRAM over the STT-RAM and SRAM are its ability of having short read and write times, and low power leakage [8].
III. DATA ANALYSIS

The data was collected from some cache configurations and the metrics Energy (nJ) and Cache Latencies (nsec and cycles) were focused as described in Fig.1, Fig. 2, and Fig.3.

### TABLE I. METRICS OF MULTI-LEVEL CACHE

<table>
<thead>
<tr>
<th>Parameters for the below techniques, Year</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>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>Khoshavi [4]</td>
<td>N/A</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
</tr>
<tr>
<td>Lin [5]</td>
<td>N/A</td>
<td>800 MHz</td>
<td>32KB</td>
<td>N/A</td>
</tr>
<tr>
<td>Jog [6]</td>
<td>N/A</td>
<td>2GHz</td>
<td>32KB</td>
<td>4-way</td>
</tr>
<tr>
<td>Sun [7]</td>
<td>4</td>
<td>2GHz</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>Jokar [9]</td>
<td>4</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
</tr>
<tr>
<td>Wang [10]</td>
<td>4</td>
<td>3GHz</td>
<td>64KB</td>
<td>8-way</td>
</tr>
<tr>
<td>Zhang [11]</td>
<td>16</td>
<td>3.5GHz</td>
<td>32KB</td>
<td>4-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)}
In Fig. 1, we can see a trend that SRAM generally consumes less energy per read/write access, however STT-RAM consumes more energy per read/write access. We can also assume that over the years the SRAM and STT-RAM have become more efficient.

In Fig. 2, we can see a trend that SRAM and STT-RAM finish read operations in about same time, however STT-RAM takes longer to finish write operations than the SRAM.

In Fig. 3, we can see a trend that read operations take about the same cycles for both STT-RAM and SRAM, whereas write operations for STT-RAM take much more time than SRAM.
IV. CONCLUSION

To conclude, certain trends were observed that SRAM and STT-RAM are about equally fast for performing read operations, however STT-RAM takes longer to perform write operations and therefore also consumes more energy for write operations. SRAM and STT-RAM were found as very common designs used for building cache memory. However, there are several companies and organizations doing research on how to improve write performance and efficiency on the STT-RAM.
REFERENCES


