Technology and Performance Comparison of Cache Design and Memory

Dhaval Desai
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

Abstract—
Several cache designs and different types of cache configurations are observed to improve and design a cache memory that can help enhance communication between main memory and CPU. The objective is to understand cache latency and energy by studying and discussing designs such as SRAM, eDRAM, and STT – RAM. And observe different types of cache configuration such as fully associative cache, direct mapped cache, and set associative cache. Trend can be observed by comparing different design and cache configuration from the provided research references.

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

1. INTRODUCTION

Cache configuration is becoming more advance, but there are three different configurations for cache memory. Direct mapping, set associative mapping, and fully associative mapping are the three way you can configure cache. In direct mapping, a block is mapped to one cache location. On the other hand, in set associative mapping, a block is mapped to a branch of cache location. Fully associative mapping has similar functionality to direct mapping, but it allows block to map at any cache location instead of pre-specified location.

There are few reason why multilevel cache is important. First because of the size on the die. Every bit in cache is carried by at least one transistor. You can use one big cache instead of using multilevel cache but it has to be close to Arithmetic logic unit (ALU) and Memory management unit (MMU). If we use a large single cache, it will be difficult to design the processor since large cache needs more space. Benefit of using small cache can not only allow you to design a better processor but it can also provide better performance. For example, L1 cache can maximize the speed and L2 can store large data. This way we can save space on die and improve performance at the same time.

There are three different types of block placement. Direct mapped, fully associative, and set associative. If a block only has single place in cache, that cache is direct mapped. Direct mapping ignores the fact that soft bits are easier than hard bits when it comes to access [3]. Direct mapped is one way set associative. Majority of processor are direct mapped. Fully associative block can be placed anywhere in the cache. Fully associative cache with m block is also known as m – way set associative. On the other hand, set associative block can only be placed in restricted place in cache. In this case, block is mapped onto set and later it is placed in that set. On chip caches are usually set associative. It makes them responsive to dynamic set sampling [13].

There are two different types of memory in computer. Volatile and Non-volatile. Non-volatile memory can recover stored data even after restart. Flash memory, read-only memory, optical discs, and floppy disks are the example of non-volatile memory. This type of memory can be used for long term storage or secondary storage. Good part about non-volatile memory is it does not need electricity all the time. Volatile memory can be found in most of the systems today. Volatile memory loses data in RAM once the system shuts down. Volatile memory also requires electricity to function. Spin-Transfer Torque RAM (STT-RAM) is an example of non-
volatile memory [4]. STT-RAM is better than on chip SRAM when it comes to designing memory for multi-core architecture [6]. SRAM and eDRAM are the example of volatile technology. STT-RAM is more popular than SRAM since it has low leakage, high density, and high endurance. When it comes to eDRAM, it has similar benefits with two times higher leakage than STT-RAM. eDRAM also has much higher refresh energy overhead.

There are three different types of organization in cache. Set associative, direct mapped, and fully associative. First we have to know if the data is in the cache or not in order to access that data. Location of cache depends on the address of word direct mapped. In cache, location of memory is mapped to a single location. There is also a formula which can be used in direct mapping to find cache location and mapping address. (Block address) mod (Total number of cache blocks). Set associative cache with n block location is also known as n-way set associative cache. Set associative cache has similar operation to direct mapped. In terms of design, set associative cache comes in between fully associative and direct mapped. The reason behind that is because it has fixed locations where block can be replaced. Formula to find the cache location in set associative cache is (Block number) mod (Total number of set in cache).

Cache memory can be found in two different locations. There is an internal cache which can be found inside the processor. And there is an external cache which is located on the motherboard. Cache is separated in three blocks. SRAM, Cache controller, and TRAM. Static Random Access Memory (SRAM) holds the data. TRAM (Tag RAM) holds the address of the data that is saved in SRAM. Cache controller has multiple responsibilities such as updating Tag RAM and SRAM and implementing write policy. Cache controller decides if request is hit and also determines if that request is cacheable. These three blocks can be combined into one single chip or they can also implement by multiple chips.

Cache memory’s performance is calculated in terms of hit ratio. CPU requested data found in cache is called hit ratio. If that particular data is not in cache, then it is located in main memory. If a word is not found in cache, then it is known as miss ratio. Hit ratio always rests between 0 and 1. Hit ratio = (total number of hits) / (total requests). Miss ratio = 1 – hit ratio.

Forthcoming section will analyze specific metrics from the provided research references. We will analyze the data and sketch some graphs to determine how things have changed over time in cache technology.

II. Literature Review

There are few different metrics for cache configuration that can be studied from the provided references. We are going to determine required energy for read and write operation by comparing 10 different computer system. We have picked [1], [2], [3], and [4] from the baseline design. We are using metrics from [5], [6], [7], [8], [9], and [10] contrast design to determine the required energy.

SRAM, STT-RAM, and eDRAM are the most popular types of cache technology you can find in today’s computer system. SRAM (Static RAM) which can be found in traditional processor have few weaknesses. It has high leakage current and low density [8]. SRAM is also widely used embedded technology because it provides fast access to memory. In some case, you also find multiple types of caches used at the same time. For example, I have found out that you can use SRAM in upper level cache and eDRAM in lower level cache in order to reduce data transfer speed between memory components [3].

STT-RAM which is also know as Spin – Transfer Torque Random Access Memory is one of the most promising cache technologies because of its low leakage power, radiation hardness, high density, and zero standby [9]. It is also considered as an alternative to SRAM because it is non-volatile like SRAM and it can provide read speed like SRAM.
and zero leakage like DRAM [2]. However, STT-RAM has some disadvantages that has slowed down its adoption. STT-RAM has high write latency and energy which makes companies to use different technology [5]. It is also recommended to use STT-RAM in the lower level cache to minimize cache leakage. STT-RAM also has small cell area compared to SRAM. This allows not only a larger cache but it also improves overall system performance [6]. Another thing that I have found is an issues with read performance and stability if you scale it down to 45nm and below. That can also increase error rate [10].

eDRAM which is also knows as DRAM is one of the most promising cache solution because of its ability to provide short read and write time and it also has low power leakage. There are two different types of eDRAM. 1T1C eDRAM and gain cell eDRAM. The reason why STT-RAM is replacing eDRAM because eDRAM needs some type of capacitor to store the data. For instance, 1T1C eDRAM uses dedicated capacitor to store the data [7]. Since stored charge slowly leaks, it requires refresh in order to prevent loss of data. This means after every refresh data on the RAM needs to read out and written back to cell [1].

### III. Data Analysis

#### Energy Consumption

![Fig. 1. <Read/Write energy consumption comparison.>](image1)

By looking at the Fig. 1, we can observe the difference between SRAM and STT-RAM performance. We can see that STT-RAM uses more energy for read and write access than SRAM. In general, SRAM uses less energy to do the same process.

#### Cache Latency

![Fig. 2. <Cache Latency (time).>](image2)

By looking at the Fig. 2, we can observe the difference between SRAM and STT-RAM performance. When it comes to read operation, STT – RAM and SRAM finishes at almost same time, however, STT – RAM takes a little longer to finish when doing the write operation. This means SRAM is faster than the STT – RAM when it comes to write operation.

#### Cache Latencies cycle

![Fig. 3. <Cache Latencies (cycles).>](image3)

By looking at the Fig. 3, we can observe that SRAM and STT-RAM takes almost same time for read operation. But when it comes to write operation, STT – RAM takes little longer than SRAM.

### IV. Conclusion

As the research has demonstrated that STT – RAM and SRAM are almost the same when it comes to performing certain tasks such as read
operation. However, you will observe the different in performance when it comes to write operation where STT – RAM take a bit longer to perform the task. In addition to that, we also see that STT- RAM needs more energy since it takes more time to perform write operation. We also found the benefits and disadvantages of STT – RAM, SRAM, and eDRAM. We came to conclusion that STT – RAM and SRAM are same popular and can be found in cache memory design.

REFERENCES


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Khoshavi [1]</td>
<td>8</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>MESI</td>
<td>512</td>
<td>8-way</td>
<td>512KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>8192</td>
<td>MESI</td>
<td>96MB</td>
<td>16-way</td>
<td>eDRAM</td>
<td>~100M</td>
<td>WB</td>
</tr>
<tr>
<td>Chen [2]</td>
<td>4</td>
<td>3.3 GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>WB</td>
<td>512</td>
<td>8-way</td>
<td>8MB</td>
<td>8-way</td>
<td>STT-RAM</td>
<td>65536</td>
<td>WB</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Khoshavi [3]</td>
<td>N/A</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>WB</td>
<td>512</td>
<td>8-way</td>
<td>512KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>8192</td>
<td>WB</td>
<td>96MB</td>
<td>16-way</td>
<td>eDRAM</td>
<td>~100M</td>
<td>WB</td>
</tr>
<tr>
<td>Lin [4]</td>
<td>N/A</td>
<td>800MHz</td>
<td>32KB</td>
<td>1-way</td>
<td>SRAM</td>
<td>N/A</td>
<td>512</td>
<td>N/A</td>
<td>512KB</td>
<td>N/A</td>
<td>STT-RAM</td>
<td>65536</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Jog [5]</td>
<td>N/A</td>
<td>2GHz</td>
<td>32KB</td>
<td>4-way</td>
<td>SRAM</td>
<td>WB</td>
<td>512</td>
<td>16-way</td>
<td>1MB</td>
<td>8-way</td>
<td>STT-RAM</td>
<td>16384</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Sun [6]</td>
<td>4</td>
<td>2GHz</td>
<td>32KB</td>
<td>4-way</td>
<td>SRAM</td>
<td>N/A</td>
<td>512</td>
<td>8-way</td>
<td>256KB</td>
<td>8-way</td>
<td>STT-RAM</td>
<td>4096</td>
<td>N/A</td>
<td>4MB</td>
<td>16-way</td>
<td>STT-RAM</td>
<td>65536</td>
<td>N/A</td>
</tr>
<tr>
<td>Chang [7]</td>
<td>8</td>
<td>2GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>MESI</td>
<td>512</td>
<td>8-way</td>
<td>256KB</td>
<td>8-way</td>
<td>STT-RAM</td>
<td>4096</td>
<td>MESI</td>
<td>32MB</td>
<td>16-way</td>
<td>SRAM</td>
<td>524288</td>
<td>WB</td>
</tr>
<tr>
<td>Jokar [8]</td>
<td>4</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>DRAM</td>
<td>WB</td>
<td>512</td>
<td>8-way</td>
<td>256KB</td>
<td>8-way</td>
<td>STT-RAM</td>
<td>32768</td>
<td>WB</td>
<td>8MB</td>
<td>16-way</td>
<td>ReRAM</td>
<td>131072</td>
<td>WB</td>
</tr>
<tr>
<td>Wang [9]</td>
<td>4</td>
<td>3GHz</td>
<td>64KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>MESI</td>
<td>1024</td>
<td>16-way</td>
<td>2MB</td>
<td>16-way</td>
<td>STT-RAM</td>
<td>32768</td>
<td>MESI</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Zhang [10]</td>
<td>16</td>
<td>3.5GHz</td>
<td>32KB</td>
<td>4-way</td>
<td>SRAM</td>
<td>MESI</td>
<td>512</td>
<td>8-way</td>
<td>256KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>4096</td>
<td>MESI</td>
<td>16MB</td>
<td>16-way</td>
<td>SRAM</td>
<td>262144</td>
<td>MESI</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)}