Abstract—This paper gives an overview of the benefits and downfalls of different cache configurations. Cache memory can be seen as the cheapest way to speed up your computer. Using multilevel cache such as L1,L2,L3 and possibly L4 if there is enough area on a given chip. By utilizing cache memory is great but the down fall is that the allotted memory is a lot less than main memory. Metrics in this paper can easily show the latency and energy for both STT-RAM and SRAM and show which is better than the other.

Keywords—STT-RAM, SRAM, Cache, L1, L2, L3, Memory.

I. INTRODUCTION

Cache configuration has become very interesting in the way it is set up for efficient performance. STT-RAM is faster at reading and SRAM is faster at writing. Coming up with the perfect configuration while implementing both to get “the best of both worlds”, can be tricky. Multilevel cache is what allows this to be possible. You want to design the circuit in such away that you have your “highly” used material in L1 cache and the least use information stored in your hard drive. There are many different ways of accessing or writing from memory to cache. This is done by different implementations of direct mapping, full associative, and set associative.

Direct-mapped is the simplest, easiest, and fastest way to check to see if the memory item we are looking for is readily available. It either has what we are looking for or it doesn’t. The main concern/downfall with direct mapping is that there is a very low hit ratio in most cases. With Set-Associative map we see how on a larger scale that we can almost achieve a 100% hit ratio. With set-associative it check multiple associative memory items at the same time which allows for greater chances of a hit.

In section 2 of this paper we will compare the energy usage along with the latency of STT-RAM vs. SRAM. This is important because it shows how fast the user is receiving and how much energy the system is using. Not only does this show the benefits of STT-RAM over SRAM but it also shows how Moore’s Law holds true over the past 5 years.

II. LITERATURE REVIEW

In this section we see two sets of metrics compared over the past 5 years. First we see the decline in the delay in execution or latency from the years 2011 to 2016. We can see that when comparing STT-RAM to SRAM that the read latency is consistently less in STT-RAM than that found in SRAM. Even though STT-RAM showed to be more beneficial both showed a steady decline in their latency delay. In the second graph we show Energy used while comparing STT-RAM and SRAM over the years 2011-2015. Once again we see that both had a steady decrease in read energy used in the span of the 5 years when looking at their read energy. Both showed a steady decline in energy used but STT-RAM once again shows that it is consistently uses less energy and is more efficient in terms of read latency. Data Analysis

The tables below are graphs of STT-RAM and SRAM latency and energy comparisons over the past 5 years. You can easily see that within the past five years the latency time response has steady gone down and can be noted with the corresponding trend line. Over the past 5 years, the energy used has also declined notably. In both graphs it appears that for read latency and read energy STT-RAM is the better of the two having a faster response time in most years and utilizing less energy in most years as well.
In conclusion you can see that using cache memory in the right manor can be very beneficial. Even though it may not contain as much memory space as main memory if organized in the right manor this can not only speed up your unit but also use less energy and work more efficient. Using associative mapping shows its benefits versus direct mapping by optimizing your hit ratio. The metrics used for comparing STT-RAM and SRAM, show that STT -RAM is not only faster but uses less energy. It also shows that Moore’s law holds true in the manner that the energy and latency have both decreased significantly in the past 5 years.

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>
</tr>
</thead>
<tbody>
<tr>
<td>Khoshavi [2]</td>
<td>8</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>512</td>
<td>MESI</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>Sun [8]</td>
<td>4</td>
<td>2GHz</td>
<td>32KB</td>
<td>4-way</td>
<td>SRAM</td>
<td>512</td>
<td>N/A</td>
<td>256KB</td>
<td>8-way</td>
<td>SRAM</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>Chen [3]</td>
<td>4</td>
<td>3.3GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>512</td>
<td>WB</td>
<td>4 MB</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>WB</td>
</tr>
<tr>
<td>Khoshavi [14]</td>
<td>8</td>
<td>3GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>512</td>
<td>WB</td>
<td>N/A</td>
<td>8-way</td>
<td>N/A</td>
<td>N/A</td>
<td>WB</td>
<td>96MB</td>
<td>16-way</td>
<td>eDRAM</td>
<td>N/A</td>
<td>WB</td>
</tr>
<tr>
<td>Jog [6]</td>
<td>4</td>
<td>2GHz</td>
<td>32KB</td>
<td>4-way</td>
<td>SRAM</td>
<td>512</td>
<td>WB</td>
<td>1MB</td>
<td>16-way</td>
<td>SRAM</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>Jaleel [7]</td>
<td>4</td>
<td>N/A</td>
<td>32KB</td>
<td>4-way</td>
<td>SRAM</td>
<td>512</td>
<td>WT</td>
<td>256KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>4096</td>
<td>WB</td>
<td>64MB</td>
<td>16-way</td>
<td>SRAM</td>
<td>1048576</td>
<td>WB</td>
</tr>
<tr>
<td>Sun [8]</td>
<td>4</td>
<td>2GHz</td>
<td>32KB</td>
<td>4-way</td>
<td>SRAM</td>
<td>512</td>
<td>N/A</td>
<td>256KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>4096</td>
<td>N/A</td>
<td>4MB</td>
<td>16-way</td>
<td>SRAM</td>
<td>65536</td>
<td>N/A</td>
</tr>
<tr>
<td>Chang [9]</td>
<td>8</td>
<td>2GHz</td>
<td>32KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>512</td>
<td>MESI</td>
<td>256KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>4096</td>
<td>MESI</td>
<td>32MB</td>
<td>16-way</td>
<td>eDRAM</td>
<td>524288</td>
<td>WB</td>
</tr>
<tr>
<td>Sun [10]</td>
<td>8</td>
<td>2GHz</td>
<td>16KB</td>
<td>2-way</td>
<td>SRAM</td>
<td>256</td>
<td>WT</td>
<td>8MB</td>
<td>32-way</td>
<td>STT-RAM</td>
<td>131072</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>Chandra[12]</td>
<td>2</td>
<td>3.2GHz</td>
<td>32KB</td>
<td>4-way</td>
<td>SRAM</td>
<td>512</td>
<td>WB</td>
<td>512KB</td>
<td>8-way</td>
<td>SRAM</td>
<td>8192</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>
</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)}