An Analysis of Adding a Backside Level-Two Cache to an Existing Microprocessor

An Analysis of Adding a Backside Level-Two Cache to an Existing Microprocessor

by

Kathryn Christine Hammel, B.S. Computer Engineering

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2006

An Analysis of Adding a Backside Level-Two Cache to an Existing Microprocessor

Approved by
Supervising Committee:

_____

_____

Dedication

To my family and Bryce for always loving and supporting me.

Acknowledgements

I would like to thank Dr. Lizy John for her understanding and guidance in doing this project. I would also like to thank Nasr Ullah for allowing me to work with his group on this project and agreeing to be a reader for it. I would like to say thank you to the members of the System Performance Team at Freescale for all their help, especially Knute Lingaard who really made this all possible. Last but not least, I would like to thank my parents and Bryce for supporting me through these years.

May 5, 2006

Abstract

An Analysis of Adding a Backside Level-Two Cache to an Existing Microprocessor

Kathryn Christine Hammel, M.S.E.

The University of Texas at Austin, 2006

Supervisor: Lizy Kurian John

One of the most important performance factors in a processor design is the cache organization. With memory latencies not keeping pace with cycle time, caches have become an important element of the design to alleviate that problem. Part of that solution has been to use a multi-level cache hierarchy. This report looks at the performance advantage of adding a backside level-two cache to an existing design. As expected, the performance increases with a backside level-two cache for those programs which make use of it. An analysis like this is important when designing a processor so that an optimal design can be found and implemented quickly.

Table of Contents

List of Tables

List of Figures

1 INTRODUCTION

As technology and markets evolve, microprocessor design must adapt to these changes. Decreases in memory latency have not kept pace with decreases in processor cycle time. Processors are becoming faster and more complex, but memory accesses still limit the performance. Almost every third instruction is a memory access [1]. Improvements in superscalar processors will be nullified if memory operations cannot be improved as well. Caches have traditionally been one answer to the high memory latency problem. Caches exploit both temporal and spatial locality properties [12]. They decrease the memory access time by keeping data on-chip with the processor. Unfortunately, caches come with a price (area, hardware, timing constraints, etc.) and that must be balanced with the processor requirements.

Multi-level caches have been one way to achieve that balance. Most Level-Two (L2) caches are significantly bigger than Level-One (L1) caches. This allows the small level-one caches to be fast so that cycle time can be kept small, and it also allows the larger cache to keep the miss rate low and the long memory accesses to a minimum.

With so many variables in the cache design (block size, set associativity, cache size, etc.), performance analysis and simulation are one way to achieve an optimal design. Given system constraints and objectives, designers would like to architect the highest performing processor that meets those objectives. Accurate performance simulators help designers achieve that goal [7]. It gives the designer the chance to vary design parameters and measure the affect on performance. Cache design is an area where performance simulators provide a good deal of insight and serve as a valuable tool because of the importance of dealing with the processor-memory gap [5, 7, 8, 9, 13].

This report looks at how the memory hierarchy affects the performance of a given design. Today, designs are using more levels of cache hierarchy to reduce the memory latency and still keep processor speed high. There are multiple ways to implement a second level of cache including look-aside and backside caches. Each has different advantages and disadvantages. Running performance simulations on different configurations provides insight into where performance bottlenecks might be, how much performance can be gained, how size affects performance, how adding a level-two cache affects the rest of the processor design, and more.

This report is of value because many existing designs are looking at ways to improve performance without redesigning an entire processor. Adding memory hierarchy like a level-two cache is one way to hopefully improve performance without completely re-architecting an existing design. Today, one processor's specifications may be changed over its life-time, and future revisions will increase the speed or remap the technology. Having a way to measure performance and adapt a design to changing specifications before implementation is an advantage.

As the technology shrinks and processor speeds increase, the long memory access time must be addressed in future revisions. Adding a second-level of cache to an existing design is becoming a standard way to deal with the widening gap that occurs as an existing design is remapped.

2 BACKGROUND

2.1 Cache Background

2.1.1 CACHE PROPERTIES

Caches are an effective means of limiting memory accesses due to the properties of spatial and temporal locality. Spatial locality relies on the idea that related data items are near each other in the address space. This is especially true of instructions where they are often sequential in memory. By fetching in more than one instruction and putting it in a cache, it is already in the processor when it is accessed in the future. Temporal locality relies on the idea that data which is in use now is likely to be used again in the future. This is easily seen in program loops [12].

When looking at improving cache performance, many people refer to a cache miss which can be categorized as one of three types: compulsory, capacity, and conflict. Various configurations can affect each type differently. A compulsory miss is the initial miss which is helped by bringing in more blocks with each miss. Capacity misses occur when the cache cannot store all the data the program is accessing. This is mainly a function of the cache size and the larger the cache, the lower the capacity misses. Conflict misses are related to the associativity of the cache and occur when blocks map to the same cache line [1].

When looking at caches, one important decision is whether to make a cache write-back (sometimes called copyback) or write-through. A write-back cache means that memory is not updated with a store operation to the cache. Instead, memory is only updated when a modified line is evicted from the cache due to replacement. This is advantageous because it decreases the amount of transactions on the system bus.

However, the disadvantage is that it is more complex to handle (some cache coherency protocol like MESI must be implemented and adhered to), and it is not error protected. This means that an error in the cache may not be recoverable and data may be lost.

A write-through cache is one where every store to the cache is also written to memory. This keeps the caches and memory coherent and is easier to implement. It is also means that any data corrupted in the cache is not lost as it is updated in memory. In general, if all caches are write-through, less snooping is required on the bus. If any cache is not write-through, then bus snooping must occur. The big disadvantage is that every store causes a transaction to occur on the bus which creates more bus contention and stalls other requestors.

2.1.2. FRONTSIDE VS. BACKSIDE

When referring to multi-level caches, the placement of the cache relative to memory determines whether it is referred to as a frontside or backside L2 cache. The following figures show various cache configurations to give an understanding of the processor/cache/memory interaction.

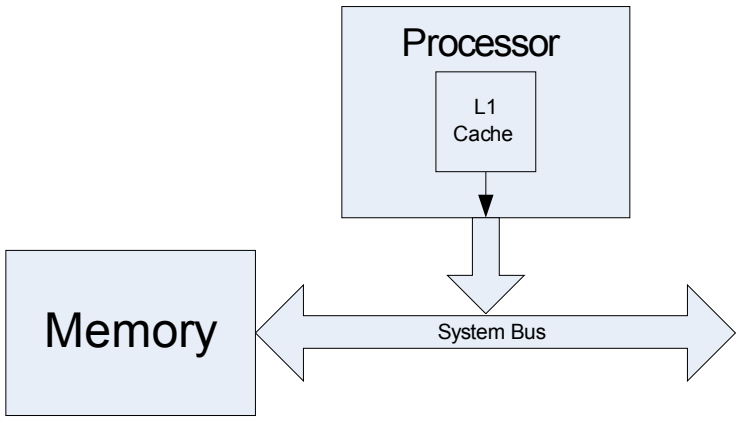

Figure 1: Single Level Cache System

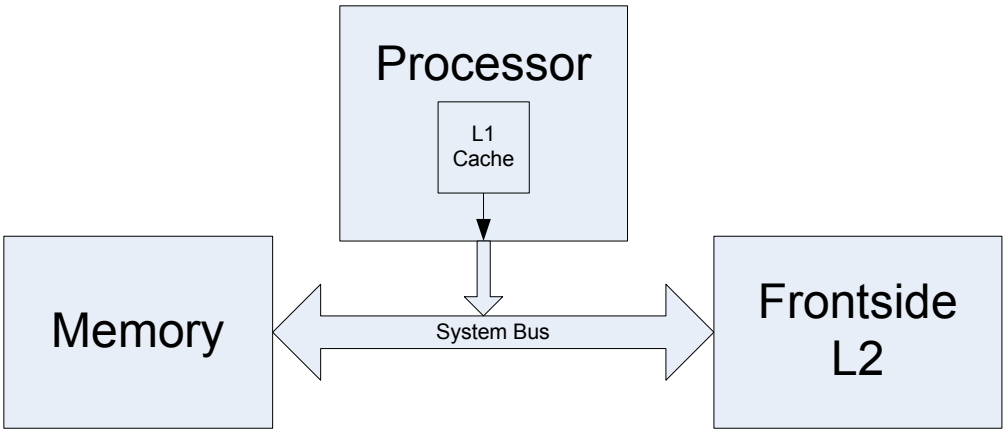

Figure 2: Frontside L2 Cache Configuration

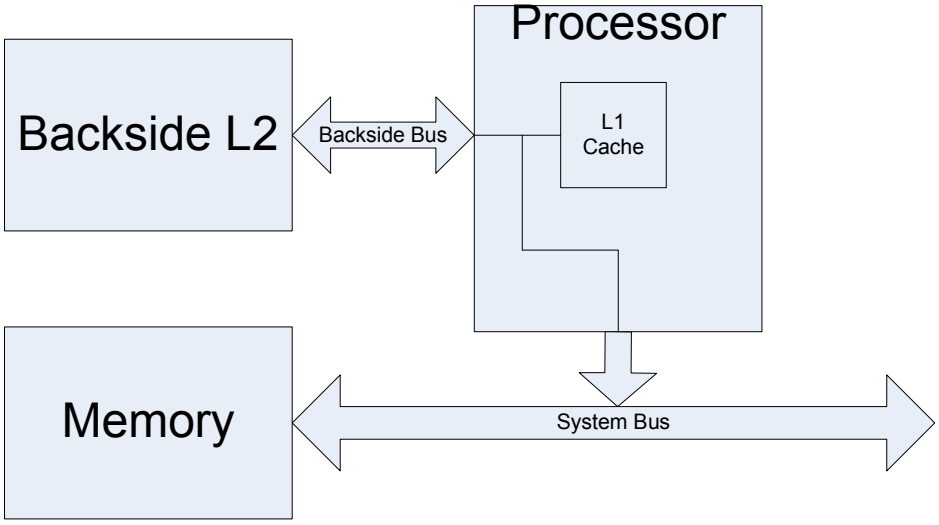

Figure 3: Backside L2 Cache Configuration

The frontside cache is usually run at the same speed as the system bus. In many systems, this is significantly slower than the processor [1]. There is also contention to deal with on the frontside bus. In a multi-processor system, the system bus contention can be significant [9].

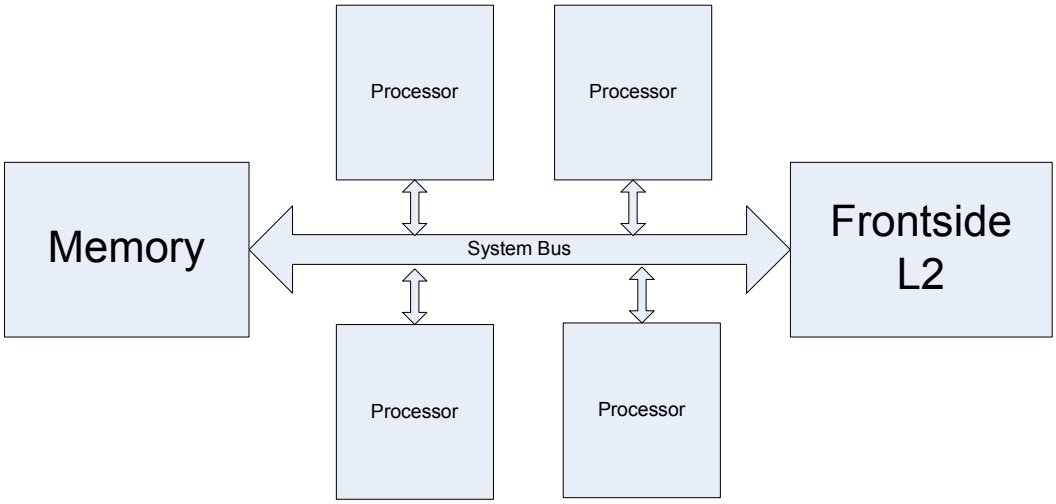

Figure 4: Multiprocessor, Frontside L2 Configuration

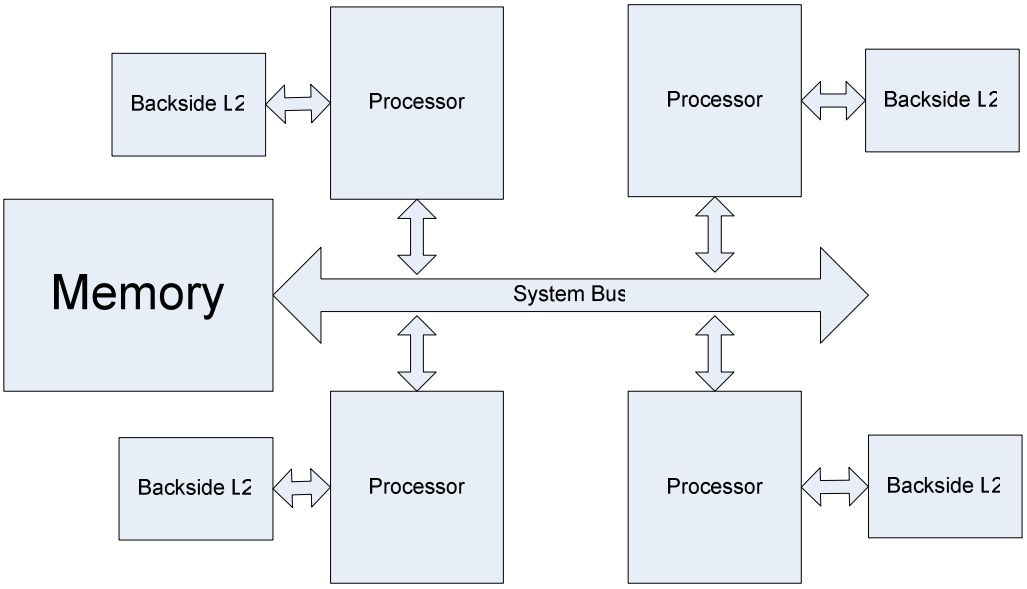

Figure 5: Multiprocessor, Backside L2 Configuration

Backside caches have the advantage of being on a dedicated bus to the processor. Another advantage is that a backside L2 is typically run at processor speed. These combine to make an access to a backside L2 cache much faster than to a frontside L2 cache [1]. However, there are advantages to a frontside cache including simplicity. A

frontside L2 cache does not affect the processor design as much as a backside L2 cache. The backside L2 cache is intimately tied to the processor's pipeline whereas a frontside L2 cache is just another entity on the system bus that adheres to the bus protocol [1, 6].

With the advantages of a backside cache, it may not be obvious why someone would implement a frontside. The backside L2 has the disadvantage of adding latency to the load miss request reaching the bus as an L2 lookup must be done first. However, there are ways to speculatively start the bus miss while looking up the data in the L2. This creates speculative traffic on the bus though which also can be a disadvantage in systems with heavy bus use. Depending on the parameters chosen for the complete cache hierarchy and memory subsystem, the performance advantages of a backside L2 may not be worth the extra design time and cost as well. In a multi-processor system, the design may only tolerate the area of a shared frontside L2 cache versus multiple backside L2 caches. The frontside L2 also has an advantage when used as a stash cache because data is put near the processor without taking up processor bandwidth.

2.2 Previous Work

There has been a good deal of research into cache optimizations. Researchers have seen the discrepancy between memory and processor speeds and have tried to find ways to improve that gap and understand where the limitations are.

Przybylski, Horowitz, and Hennessy [7,8] wrote multiple papers looking at cache performance and running simulations. They looked at processors with only an L1 Cache [7] which confirmed that larger caches lower the miss rate. They discovered that above 128KB, the miss rate levels off and the improvement is small. Caches in the range of

32KB – 128 KB maximize the performance and balance the cycle time. They also divulged that bigger caches with longer cycle times can actually perform better than smaller, faster caches at a system level. In another paper [8], these three specifically looked at the performance of multi-level cache hierarchies. They found that the L1 cache affected the optimal design of the L2 cache. This was important because it shows that one cannot just analyze the caches independently but must look at the performance and design of the entire system when making architectural decisions related to the caches. Running simulation on just L2 caches or L1 caches will not give the optimal solution. They also looked at how set associativity is affected by the L1 and L2 which can change the optimal design point.

Short and Levy [11] also looked at two-level cache simulations. They found L2 caches to be most beneficial when backing up a small L1 cache. They looked at the hit ratios of various L2 sizes against a given L1 size. Their research showed that a larger L2 size will give better performance than a smaller L2 for a given L1 size, but that performance can actually decrease when larger L2 sizes are paired with larger L1 sizes. Short and Levy also investigated various write strategies. They found that a write-back cache at both levels gave the best performance.

Jouppi and Wilton [5] looked at various tradeoffs required in systems with two-levels of cache and found many advantages. The first advantage is the dynamic allocation of the instructions. By having a unified second level of cache (instruction and data cache combined), the program can use the cache in the way that most benefits the system. A second advantage is the improved cache access times. Because a large L1 is not required with an L2, the L1 access time can be small and not limit the core speed. Another advantage is that the L2 can be set associative while keeping L1 cache direct-mapped. This also helps to keep the L1 cycle time faster. A fourth advantage is that the second

level cache can be physically addressed. This means no address translation is necessary to perform the cache lookups. The last advantage is that less power is usually used. Because the L1 can be smaller, each cache access will require less capacitance. The disadvantage is an increase in the miss latency. If a memory access must check an L1 and L2 before going to memory, the miss penalty will increase from just having to access an L1 before going to memory. Their results showed that a 32KB instruction and data cache is sufficient for a system with only L1 caches. However, they restricted their research to non-pipelined first level caches and blocking memory systems. This affects performance as having a multi-cycle L1 which is non-pipelined will hurt performance and keeping the number of L1 cycles low is important. A multi-cycle access can mean a faster processor as the cache access is not the limiting speed path. This would affect the decisions made on L2 parameters.

Reinold [9] looked specifically at the G3 PowerPC processor. He ran his simulations using the AIM Subsystem Benchmark Suite and SPECviewperf® OpenGL Benchmark and compared backside caches with look-aside caches. He found that a backside L2 cache on the G3 outperformed all sizes studied of a frontside L2 cache including a frontside cache four times larger than the backside L2 for the SPEC benchmarks. The faster access times of the backside cache and the ability to access the cache and system bus concurrently achieved this improvement. He noted that applications that require a large amount of cache to store the entire program would perform better with a larger frontside cache due to the improved hit rate and being able to keep more data close to the processor than the smaller backside allows. He also studied the effect of adding the look-aside cache as an L3 and found that the AIM benchmarks gained performance with a backside and frontside cache while the SPEC benchmarks achieved little gain with an L3 frontside cache.

2.3 Other Processors

It is interesting to note what different processors have done as far as cache organization. Many newer processors have included a backside cache. The following is a table with some well known processors and their cache hierarchies. This information was obtained from Processor data sheets on the various company websites. Multiple revisions of some of these cores have been done. The table below includes just one core version of that architecture to give a general idea of various cores.

Processor	***L1 Size***	***L2 Size***	***Reference***
IBM® PPC 405	16K D,16K I	No L2	http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405_Embedded_Cores
IBM PPC 440	32K D,32K I	No L2	http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_440_Embedded_Core
IBM PPC 603e	16K D,16K I	No L2	http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_603e_Microprocessor
IBM PPC 604e	32K D,32K I	No L2	http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_604e_Microprocessor
IBM PPC 740	32K D,32K I	No L2	http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_740_Microprocessor
IBM PPC 750	32K D,32K I	Provides dedicated interface	http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_750_Microprocessor
IBM PPC 970	32K D,64K I	512KB Backside	http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_970_and_970FX_Microprocessors
Intel® 486	8K I/D	No L2	http://www.intel.com/design/intarch/intel486/index.htm

Processor	*L1 Size*	*L2 Size*	*Reference*
Intel Pentium	16K D,16K I	No L2	http://www.intel.com/design/intarch/datashts/273214.htm
Intel Pentium II	16K D,16K I	256KB Backside	http://www.intel.com/design/intarch/datashts/273268.htm
Intel Pentium III	16K D,16K I	512KB Backside	http://www.intel.com/design/pentiumiii/datashts/273673.htm
Intel Pentium IV	8K D,12K I	512KB Backside,	http://www.intel.com/design/pentium4/datashts/298643.htm
Intel Celeron	8K D,12K I	128-256K Backside	http://www.intel.com/design/celeron/datashts/251748.htm, http://www.intel.com/design/intarch/celeron/celeron.htm
Intel Celeron M	32K D,32K I	512KB-1M Backside	http://www.intel.com/design/intarch/celeronm/celeronm.htm, http://www.intel.com/design/mobile/datashts/300302.htm
Intel Pentium M	32K D,32K I	2M Backside	http://www.intel.com/design/mobile/datashts/252612.htm, http://www.intel.com/design/intarch/pentiumm/pentiumm.htm
Intel Xeon	8K D, 12K I	512KB-1M Backside	http://www.intel.com/design/xeon/datashts/298642.htm, http://www.intel.com/design/intarch/xeon/xeon.htm
Intel Duo (2 cores)	unknown	2M Shared Backside	http://www.intel.com/design/mobile/datashts/309221.htm, http://www.intel.com/design/intarch/prodbref/311272.htm
Freescale® PPC 603e	16K D,16K I	No L2	http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC603E&nodeId=0162468rH3bTdG8653
Freescale PPC 7457	32K D,32K I	512KB	http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC7457&nodeId=0162468rH3bTdG8653

Processor	*L1 Size*	*L2 Size*	*Reference*
Freescale MPC8260	16K D,16K I	No L2	http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC8260&nodeId=0162468rH3Jk192977
Freescale MPC8349E	32K D,32K I	No L2	http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC8349E&nodeId=0162468rH3Jk191439
Freescale MPC8548E	32K D,32K I	512KB Frontside	http://www.freescale.com/webapp/sps/site/taxonomy.jsp?nodeId=0162468rH3Jk196465, http://www.freescale.com/files/netcomm/doc/fact_sheet/MPC8548FS.pdf
Freescale MPC8641	32K D,32K I	1M Backside	http://www.freescale.com/webapp/sps/site/overview.jsp?code=DRPPCDUALCORE

Table 1: Processor Comparisons of Cache Hierarchies

3 TOOLS AND METHODOLOGY

3.1 Baseline Processor

The baseline processor in this system is a superscalar, out of order execution processor that implements the PowerPC® architecture. The processor has 32 KB L1 Instruction and Data caches and supports multiple load misses. It also queues up writes/castouts from the caches. There is no backside L2 cache in the baseline processor. The system bus protocol used in this experiment is similar to PowerPC 60x bus [2]. The 60x bus protocol requires bus arbitration which includes an address tenure and separate data tenure. All transactions on the bus are snooped by whatever may be on that bus including other processors or other memory elements. The system bus is run at a 1.5:1 ratio with the core clock. This experiment will use this baseline processor to evaluate different memory hierarchies and their affect on performance and the design.

3.2 Simulator

The simulator used in this experiment is a cycle accurate model of the above baseline processor. It includes all the features of the baseline processor such that its performance matches that of the processor. It also simulates the system bus and can simulate various memory elements or processors attached to the bus. Variables can be varied to change different design parameters to test different configurations. The simulator can provide information about various queues inside of the core. It also provides statistics on bus transactions, fetching, completion, and every part of the core. In addition, it is a trace-driven simulator.

3.3 Methodology

Given the above system, it was modified it to add in a backside L2 to the design. This required modifying the simulator to include a realistic implementation and timing of a backside L2. However, these changes had to also preserve the original processor model so comparisons could be made between the systems. A simple parameter that is turned on or off can make the model perform with the backside L2 as part of the pipeline or not.

The simulator reports a good deal of information but for the purposes of this study, we examined the IPC (Instructions Per Cycle) to compare performance of the various system configurations. The results give an arithmetic mean and a geometric mean as well.

We also looked at the effective cache hit rate of an added backside L2. The effective cache hit rate is the number of cache hits due to loads and stores divided by the total number of loads and stores to the backside L2. We do not look at the total number of loads and stores in the system because some of these will hit in the L1 Cache and we are only concerned with the L2 cache hit rate. The hit rate also does not include any of the cache operation instructions such as PowerPC's dcbt (Data Cache Block Touch). The performance is reported as a percentage difference in IPC between the baseline system and the system with a backside L2.

Another aspect we are exploring is the effect of the backside L2 on the core queues. It is important to know how changing the queue sizes surrounding the backside L2 impacts performance. Depending on the L2 design, the queue sizes may be limiting performance and resizing them would provide a greater performance impact. It helps to see where the performance might level off as well such that the queues are sized appropriately. Above a certain size, the performance gain may be minimal and the extra

hardware not worth the small increase in performance. This report will only look at those queues directly impacted by the backside L2. Obviously there are queues associated with other parts of the core that could increase performance but this would be independent of the backside L2. The following table has the queue descriptions and default queue sizes used for this experiment.

Queue Name	***Queue Description***	***Default Queue Size***
Q0	L2 Castout Queue	2
Q1	L1 Castout Queue	4
Q2	Load Fill Queue	5

Table 2: Queue Description

In this report, the queue size changes are represented with +1, +2, etc. Example: Q0+1 means Q0 was increased by one in size. The queue information is consistent across the report such that anywhere Q0 is discussed it is always the same queue and so forth. The results will measure the performance difference due to a queue resize for a 1024K L2 cache. The baseline processor will be the same core with the original queue sizes and a 1024K L2 cache.

Another important issue that can affect performance is the bus protocol and latency. It is also important to see how changes to the bus protocol/latency impact the addition of a backside L2. Different systems could modify the speed or protocol in such a way that changes need to be made to the L2 or core design to accommodate these modifications. In this experiment, we have increased the bus latency to 3:1 core to system bus ratio. Everything else associated with the bus protocol was kept constant.

The benchmarks we used to run these experiments are arranged into four categories seen below in the table. When a category is said to be sensitive to memory for

instance, this means that the benchmarks in that category are very memory intensive and the results should reflect that when changes are made to the memory subsystem.

Categories	***Characteristics***
A	Benchmarks sensitive to the core
B	Benchmarks sensitive to 512K L2
C	Benchmarks sensitive to memory
BC	Benchmarks sensitive to L2 and Memory

Table 3: Benchmark Categories and Sensitivities

Some of the benchmarks in each category are SPEC® CPU2000 benchmarks. The table below gives the reader an idea of which SPEC benchmarks are in each category. This report will give results based on the performance of experiments within each category.

Category	***SPEC Benchmarks***
Category A	252.eon, 200.sixtrack
Category B	175.vpr, 178.galgel, 181.mcf, 300.twolf
Category C	171.swim, 183.equake, 254.gap, 168.wupwise
Category BC	187.facerec, 197.parser, 181.mcf, 175.vpr

Table 4: SPEC Benchmarks per Category

These are not the only benchmarks in each category. The above table is just an example of the type of benchmarks which can be found in a particular category. There are a different number of benchmarks in each category as well.

The simulator ran traces of these benchmarks. In some cases, the benchmark traces were broken down into smaller traces that exhibited different sensitivities. This is why some benchmarks appear in multiple categories (such as 175.vpr, 181.mcf). A different portion of that benchmark trace is in each category so that the traces do not overlap categories.

4 RESULTS

The experiment of adding a Backside L2 cache to a baseline processor was simulated and analyzed. These results are presented by looking at each category of benchmarks and discussing the effects of the following within that category:

- Various L2 Sizes vs. Baseline
- L2 hit rate
- Core queue sizing
- Bus impact

Each of these sections will discuss these four aspects of the L2 design in relation to that category of benchmarks. Results will include an average across the benchmarks in each category as well as the geometric mean. The same figures will be shown in each category of benchmarks. Category A will explain in more detail information about the type of chart used to make it easier to understand the charts in later sections as well.

4.1 Benchmark Category A

In this category, the benchmarks which were run are sensitive to changes in the actual processor implementation. The footprint of these benchmarks should mostly fit into the L1 Instruction and Data caches. This means that once the program is loaded into the L1 caches, it should continue to hit most of the time. Therefore, the processor performance is limited to how many instructions per cycle the processor can actually complete. Queues tied to decode and completion would have a bigger affect as these are not tied to the memory subsystem. This means that added memory hierarchy should not influence the tests.

4.1.1. EFFECT OF L2 SIZE

The following figure shows the impact of adding various sizes of backside L2 cache to the processor and the performance difference in terms of Instructions Per Cycle (IPC) vs. the baseline processor of no L2 Cache. The figure below shows a scatter plot. Each point on the x-axis represents the size of backside L2 cache and the vertical row at that point shows all the benchmark performance changes for that size of L2 cache versus the baseline system with no L2 cache.

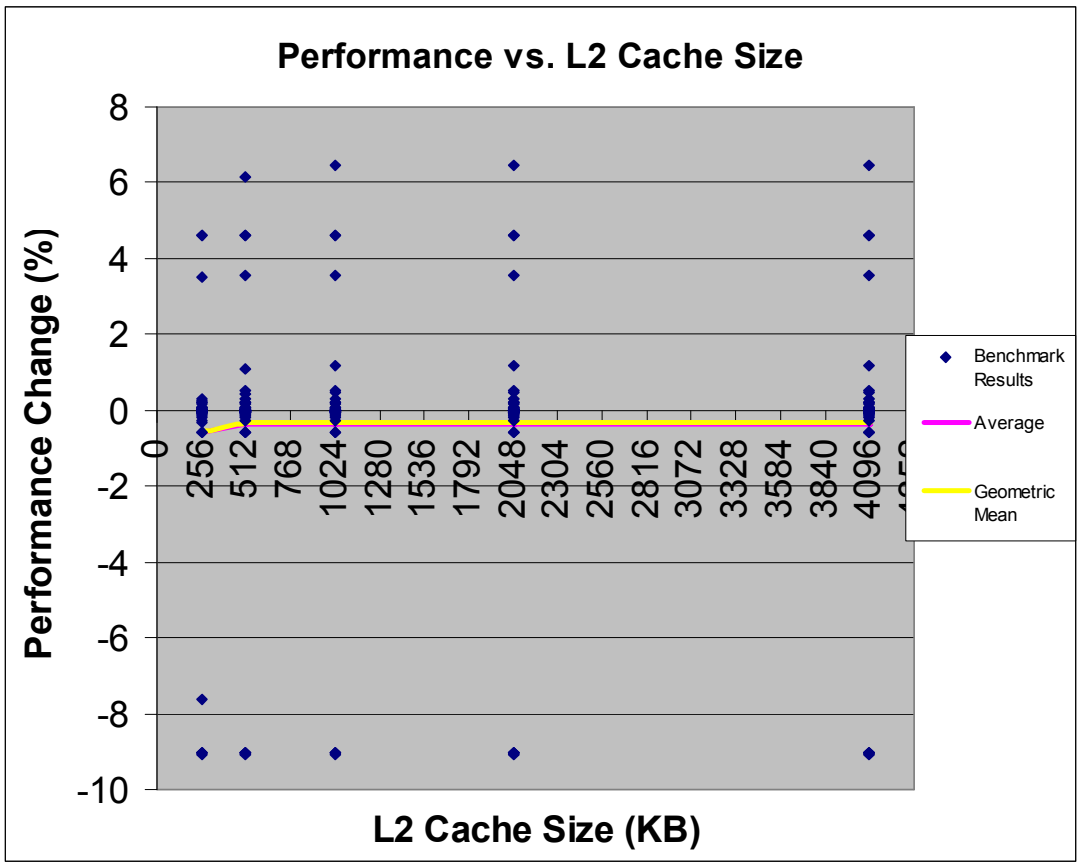

Figure 6: Category A – IPC Performance Difference vs. Baseline Processor

From the above chart, one can see that the addition of a backside L2 (regardless the size) actually hurt the average performance. Some individual benchmarks were helped by the backside L2, but the majority of results are situated around 0% change in

IPC performance. This would be expected as the benchmarks in category A are sensitive to core changes (not memory hierarchy changes). There is added load miss latency due to the fact that instructions must now access a backside L2 prior to getting to the system bus. This added latency hurts performance. Because the added L2 lookup latency is small, the performance is not significantly affected.

4.1.2. L2 HIT RATE

For purposes of this experiment, we also looked at the effective hit rate in the L2. Effective hit rate means the percentage of loads and stores which hit in the L2. This does not include cache operations like PowerPC instructions dcbt, dcbtls, etc. For category A, the hit rate varies widely (0% to over 90%). The figure below shows the L2 hit rate for each benchmark. This figure uses a scatter plot where each x-axis point is one of the benchmarks in this category. The vertical row associated with each x-axis point represents the IPC performance change for each size of backside L2 for that one particular benchmark. Points where the sizes overlap indicate benchmarks where the performance remained consistent across those sizes.

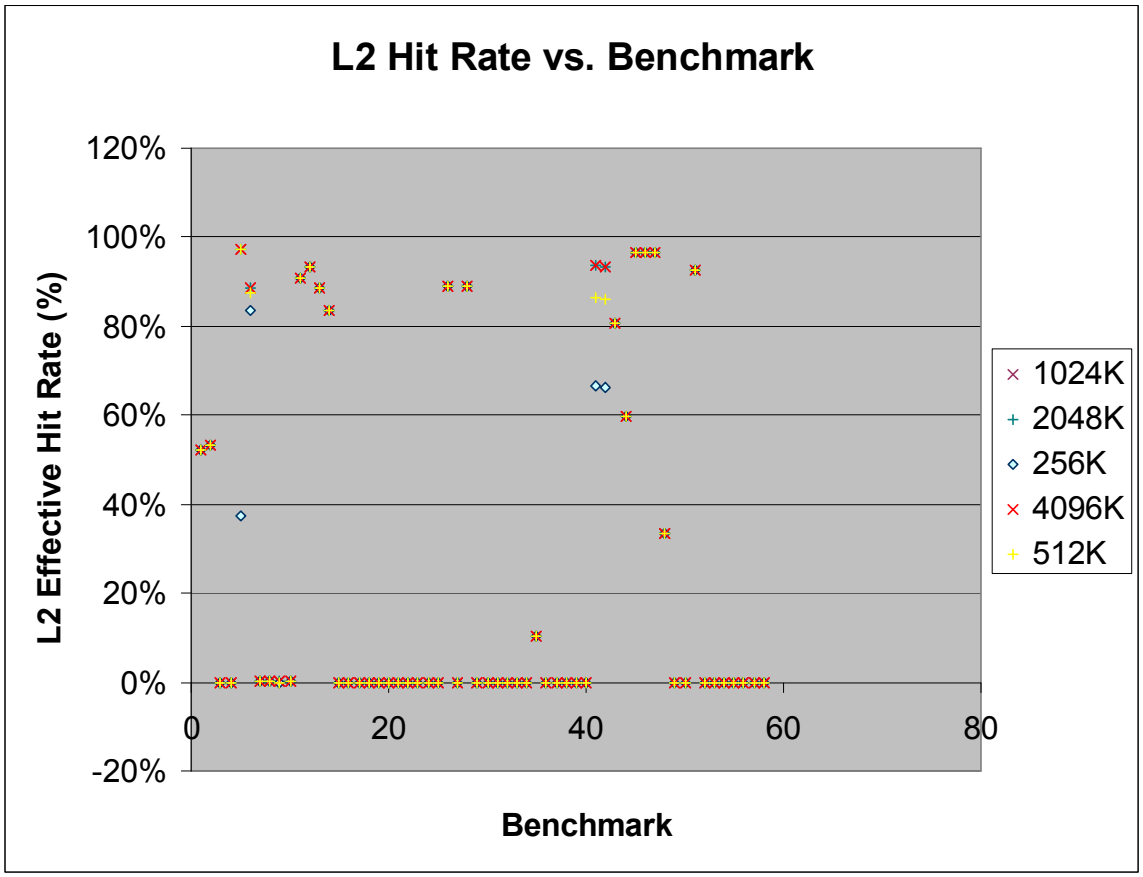

Figure 7: Category A – Effective Hit rate for Benchmarks

Figure 7 above shows that for the majority of benchmarks and L2 sizes the hit rate was 0% as can be expected from benchmarks which mostly fit in the L1 caches. We saw from section 4.1.1 that the performance had minimal gain with the addition of the backside L2. Since performance was not effected very much by the backside L2, these results concur with that since in many tests (where the hit rate is near 0%), the backside L2 is not even being used. Those traces where the hit rate percentage is high indicates traces where the program footprint is slightly larger than the L1 cache size and having that little bit of extra cache is helpful. The results also revealed that those tests with high hit rates corresponded to the tests within category A that actually improved in

performance. This can be seen by comparing Figure 7 and 8 which show the hit rate and performance for the same benchmarks as the charts look very similar. The figure below is a scatter plot which is similar to Figure 7 except the y-axis is IPC performance change rather than L2 hit rate.

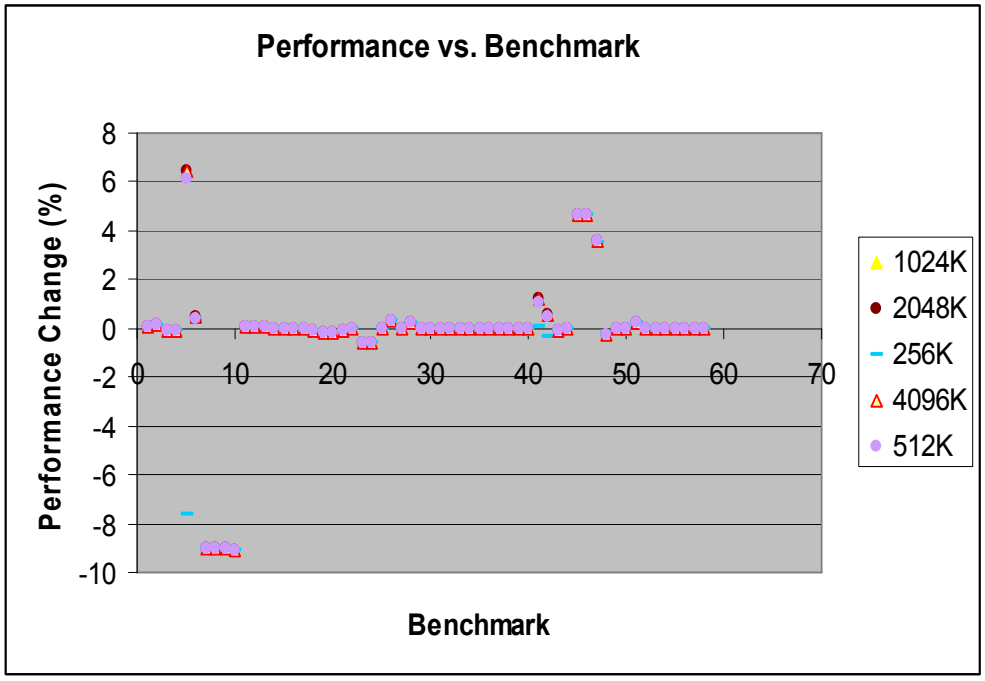

Figure 8: Category A – Performance Difference vs. Benchmarks

Figure 8 above shows that the L2 size did not affect the performance on each benchmark as most benchmarks and sizes are near 0% IPC performance change. If it had, we would expect the chart to show multiple vertical elements of the various sizes at one point on the x-axis.

4.1.3. CORE QUEUE SIZING

Benchmarks in Category A are supposed to be sensitive to core changes. It would seem logical that by resizing the queues, we would see a performance gain. However, the queues that will have an impact on performance are those queues in the pipeline not

related to the memory subsystem. This is because the benchmarks in this category mostly fit within the L1 caches. As this report is only looking at queues affected by the addition of the backside L2, we would expect to see little change in the performance for this category of benchmarks. Figure 9 below is a chart showing the performance change for the different queue changes on the x-axis.

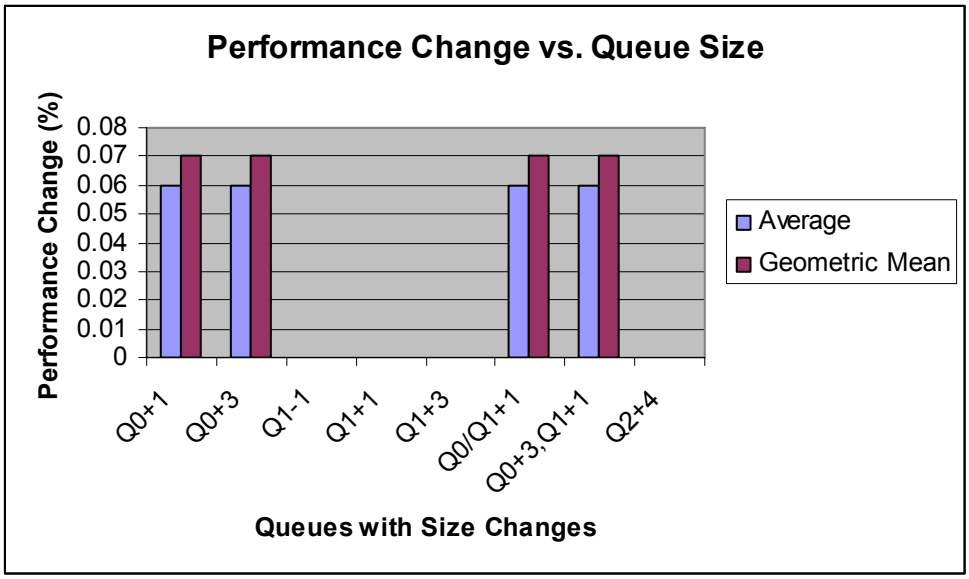

Figure 9: Category A – Performance vs. Queue Sizing

The above Figure 9 shows that for the sizes studied, there was a small IPC difference for the queue size changes. From these results, we see that increasing the queue size has a minimal impact on the performance and zero impact in some cases. It also indicates that only Q0 (the L2 castout queue) had any effect as the figure shows only performance differences for those queue changes involving Q0.

4.1.4. SYSTEM BUS IMPACT

The type of system bus can have a big impact on a processor's performance. From the perspective of adding a backside L2, we would expect to gain performance due

to decreasing the number of accesses required to go out on the bus. However, this only would apply to applications that have a large code footprint. Category A benchmarks do not fall under this category as they typically go out on the bus once to get data the first time it is accessed and then simply re-access that data in the L1 Cache.

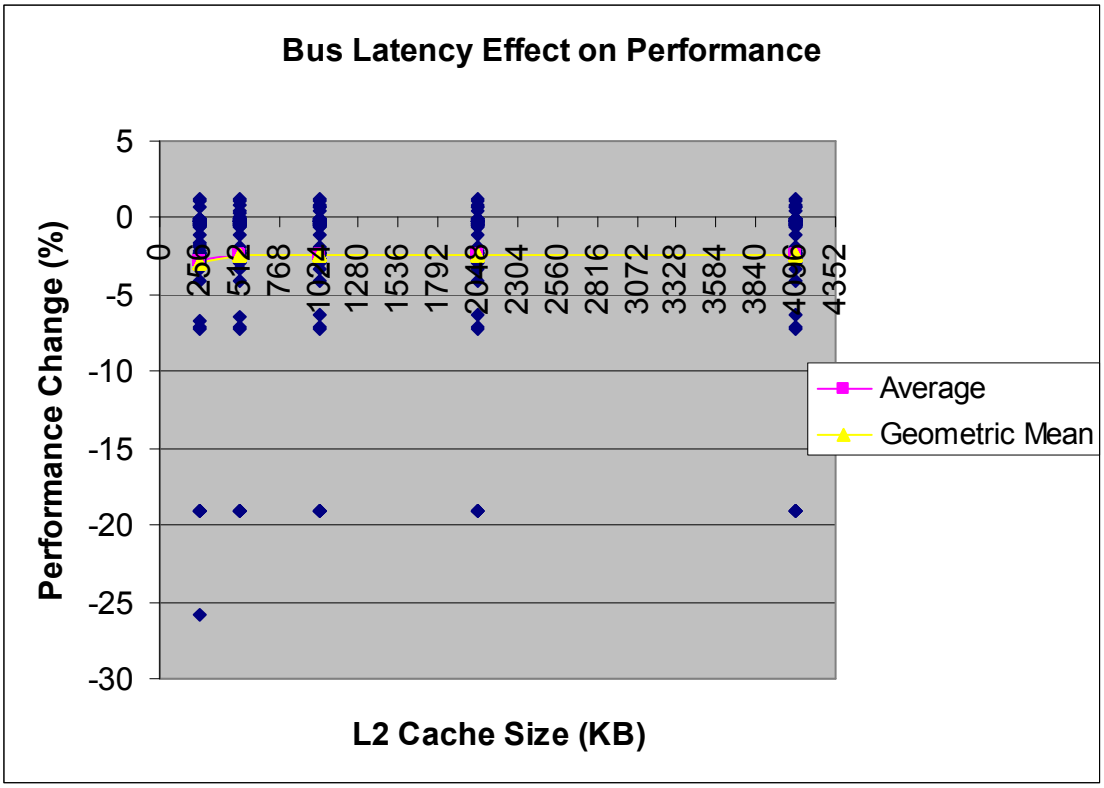

Figure 10: Category A – Bus Latency Effect on Performance

Figure 10 above is another scatter plot which shows the same information as Figure 6 in section 4.1.1. The x-axis shows the various L2 sizes. For each size, all the benchmarks in this category are shown with their IPC performance change for that L2 size versus a system with no backside L2.

Figure 9 shows that the performance actually decreased slightly for all systems with an L2 cache running a 3:1 system bus ratio over a baseline system operating at a 3:1 bus ratio. This is expected as the added latency of accessing the L2 before the bus

transaction makes the performance worse when that added latency is not giving any benefit. Because the benchmarks in this category rely very little on the backside L2, all it does is add latency and decrease performance on average.

The following figure shows the difference in performance of a 1024K L2 cache system in 1.5:1 vs. a 1024K L2 cache system in 3:1 across all the benchmarks in category A to give an idea of the type of performance degradation that can be expected by increasing the bus latency. The y-axis is that systems IPC performance difference vs. its respective baseline system.

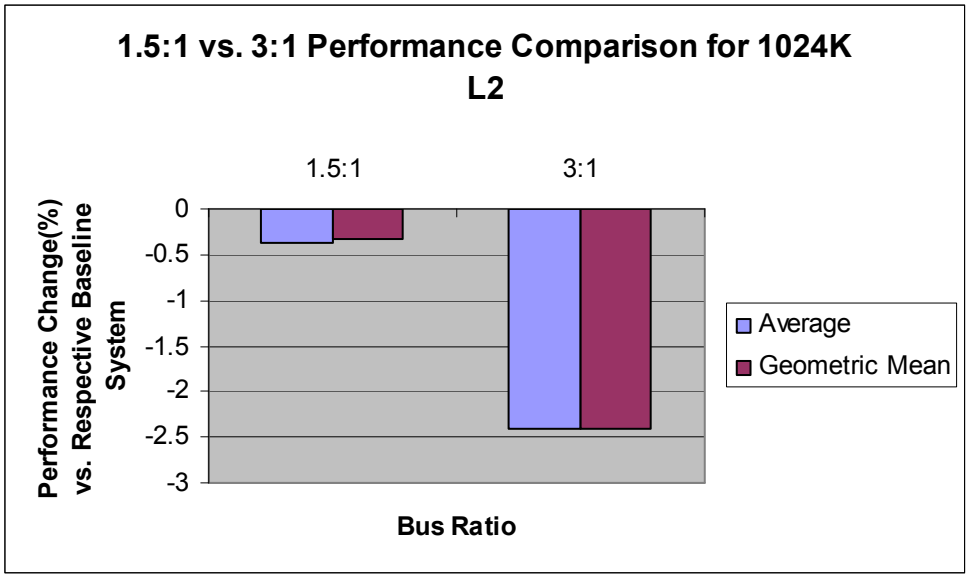

Figure 11: Category A – Performance Comparison for Different Bus Ratios

4.2 Benchmark Category B – L2 Sensitivity

In this category, the tests are supposed to be more sensitive to the presence of an added memory hierarchy. That means that these tests have a bigger program footprint than those in category A and have locality associated with them. The results support this theory as the effect of adding a backside L2 is pronounced.

4.2.1. EFFECT OF L2 SIZE

The following figure shows the performance difference of various sized L2 caches vs. the baseline system using the same scatter plot used in 4.1.1.

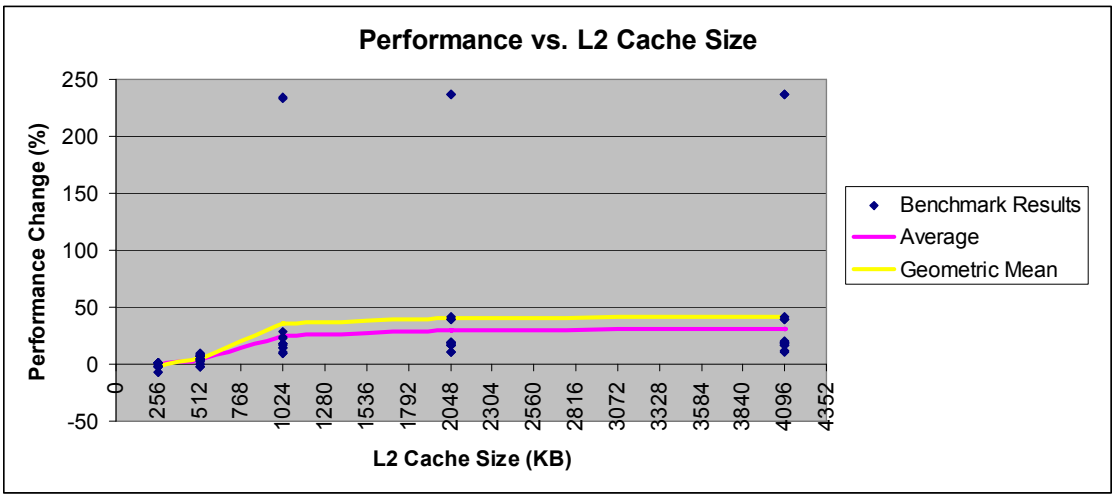

Figure 12: Category B – IPC Performance vs. Baseline Processor

Figure 12 above shows the performance difference is much bigger than in category A (Figure 6). It also indicates that the optimal L2 size for this design is 1024K. Above this size, the performance gain is minimal and most likely not worth all the extra hardware needed for a 2M of 4M L2 cache for these benchmarks. Caches smaller than 1024K have a significant drop in gained performance and again are probably not worth the extra hardware.

4.2.2. L2 HIT RATE

In category B, the L2 effective L2 hit rate should be relatively high across the board. This is because these tests are sensitive to the memory hierarchy and the results from the previous section showed significant performance gains. The figure below

shows the benchmarks along the x-axis with each vertical point showing the IPC performance change for the different L2 sizes with that particular benchmark.

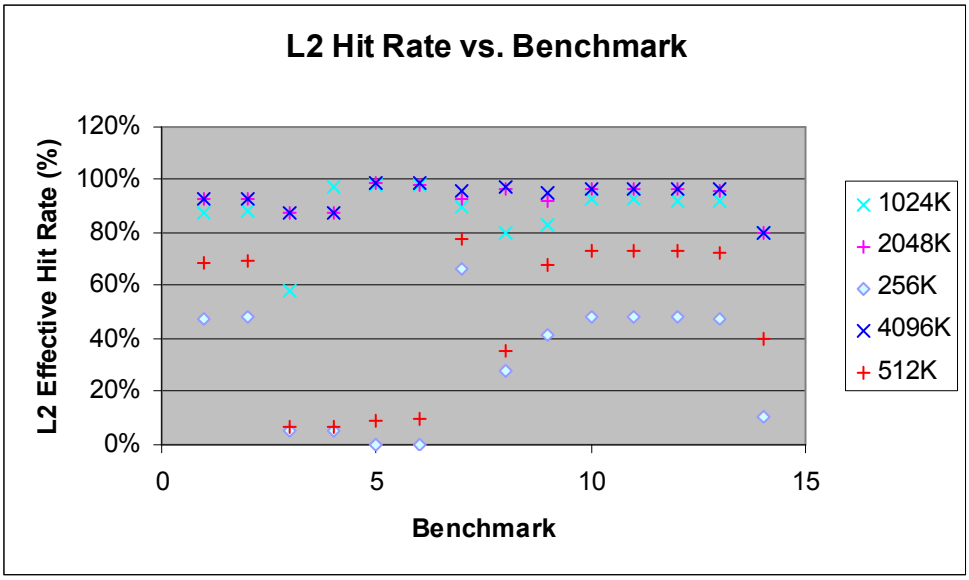

Figure 13: Category B – Effective L2 Hit Rate

Figure 13 above shows that the hit rate is relatively high for all sizes above 1024K. We observe that both 256K and 512K L2 sizes show the smallest hit rate which reflects the results in Figure 14 in reference to the performance gain. Figures 13 and 14 are similar plots with the benchmarks across the x-axis the same in both figures. This allows us to easily compare the hit rate and performance across the two figures.

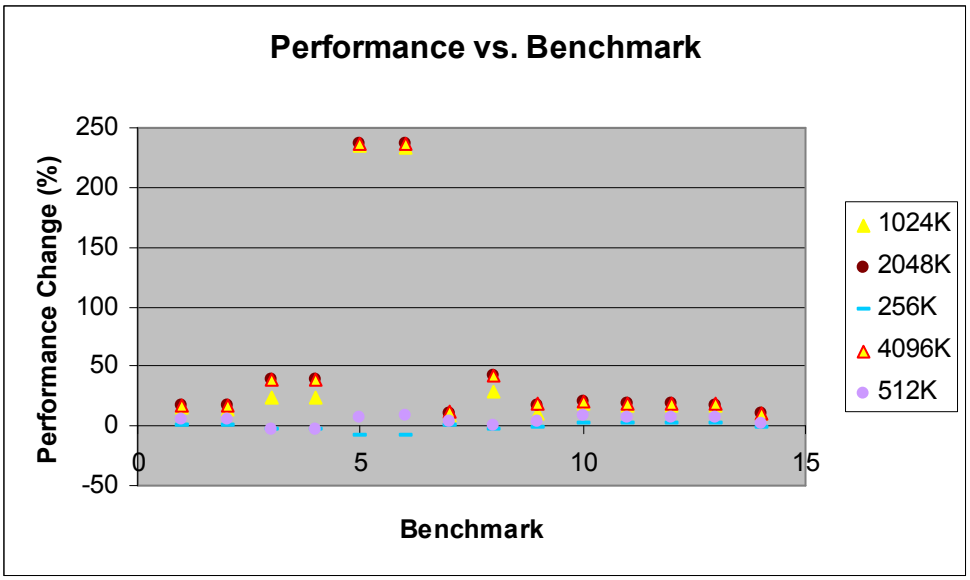

Figure 14: Category B - Performance vs. Benchmarks

The performance gained for each benchmark closely follows the hit rate for that benchmark. Figure 14 shows that for most benchmarks, the performance change varied for the different sizes with 256K and 512K being relatively small.

The analysis of these results supports the optimal size of 1024K (1M) based on these benchmarks as most of them have similar hit rates and performance for sizes 1024K and above.

4.2.3. CORE QUEUE SIZING

When looking at modifying the queue sizes, we chose the 1024K L2 cache to use due to the optimal size found in the above experiments. With this size L2 cache, we found little change in the performance. Although it did increase, it was not significant.

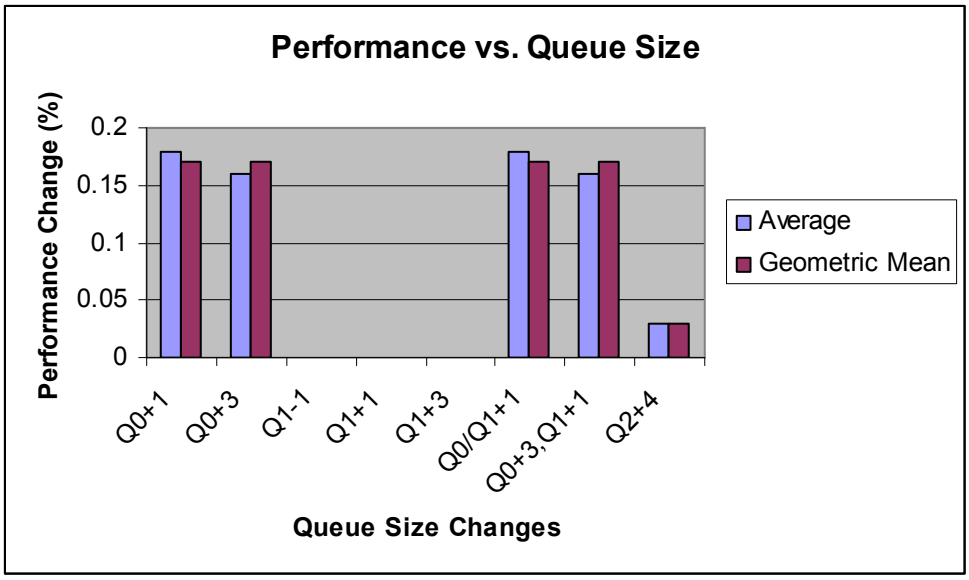

Figure 15: Category B – Performance vs. Queue Sizing

Figure 15 shows similar results to category A benchmarks with only a slightly larger performance increase. Again we see that Q1 (L1 castout queue) has little impact on the performance and Q0 (L2 castout queue) has the biggest impact. Q2 (Load Fill queue) shows a very small amount of performance increase when almost doubling the size. These results suggest that the queues are sized appropriately for the addition of a 1024K backside L2.

4.2.4. SYSTEM BUS IMPACT

Because the benchmarks in this category are L2 sensitive, we would expect the performance to increase over the baseline. The figure below shows the performance of various sized L2 caches in a system with a bus ratio of 3:1. The baseline system is a core with no L2 cache and a bus ratio of 3:1.

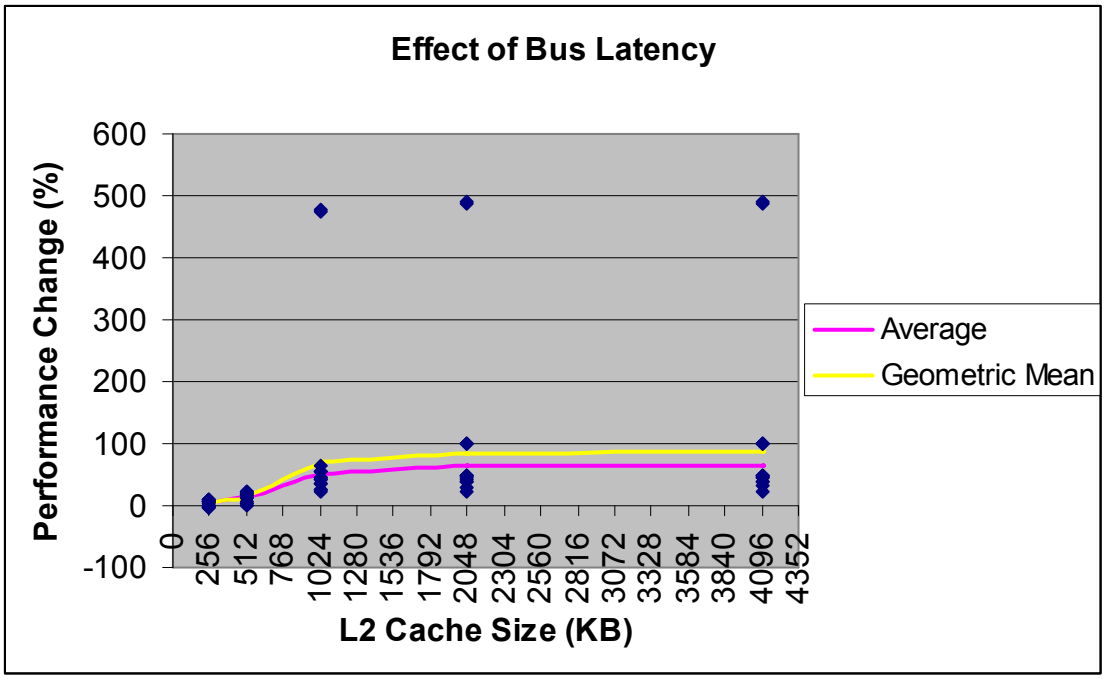

Figure 16: Category B – Bus Latency Effect on Performance

By increasing the bus latency from a ratio of 1.5:1 to 3:1, we see a big jump in performance due to the backside L2. Since these benchmarks really respond to the memory hierarchy, the decrease in memory accesses demonstrates how important the backside L2 can be in improving performance. The performance curve above resembles what we saw in Figure 12 of section 4.2.1. When we compare a 1024K L2 with a 3:1 bus ratio versus a 1024K L2 system with a 1.5:1 bus ratio across all the benchmarks in the category, we see the performance doubles from the gains received by addition of the backside L2 (from 25% to 50%). This makes sense as we doubled the bus latency. Figure 17 below shows this comparison.

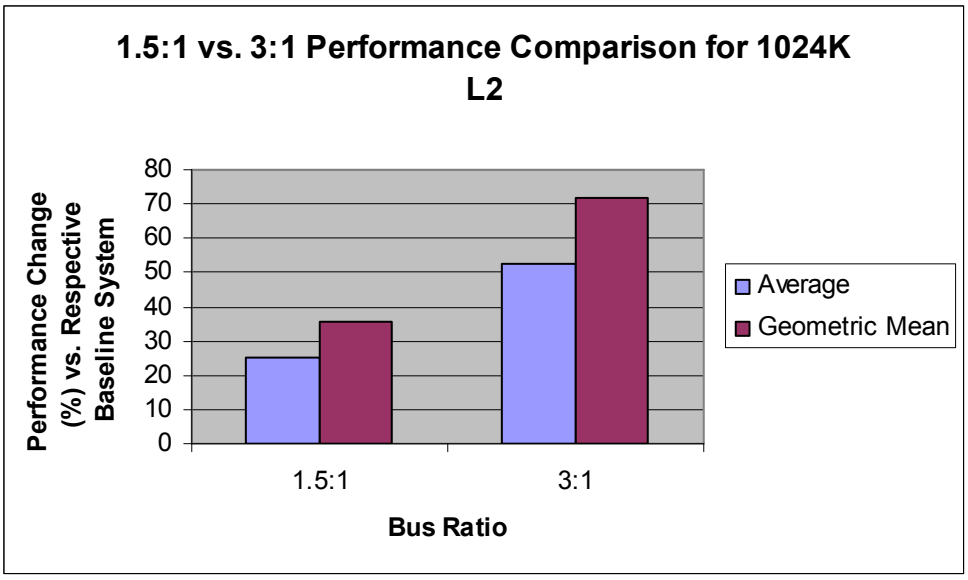

Figure 17: Category B – Performance Comparison for Different Bus Ratios

4.3 Benchmark Category C – Memory Sensitivity

The benchmarks in category C are sensitive to changes in the memory accesses. They are not sensitive to memory hierarchy or caching in general. These traces have less locality than traces in category B meaning data is often used once and never accessed again.

4.3.1. Effect of L2 Size

We would expect the size to have little effect on these benchmarks as added cache will not help code which has little locality. The scatter plot below shows the L2 size on the x-axis. Each vertical row shows all the IPC performance change for all the benchmarks in this category with that L2 size.

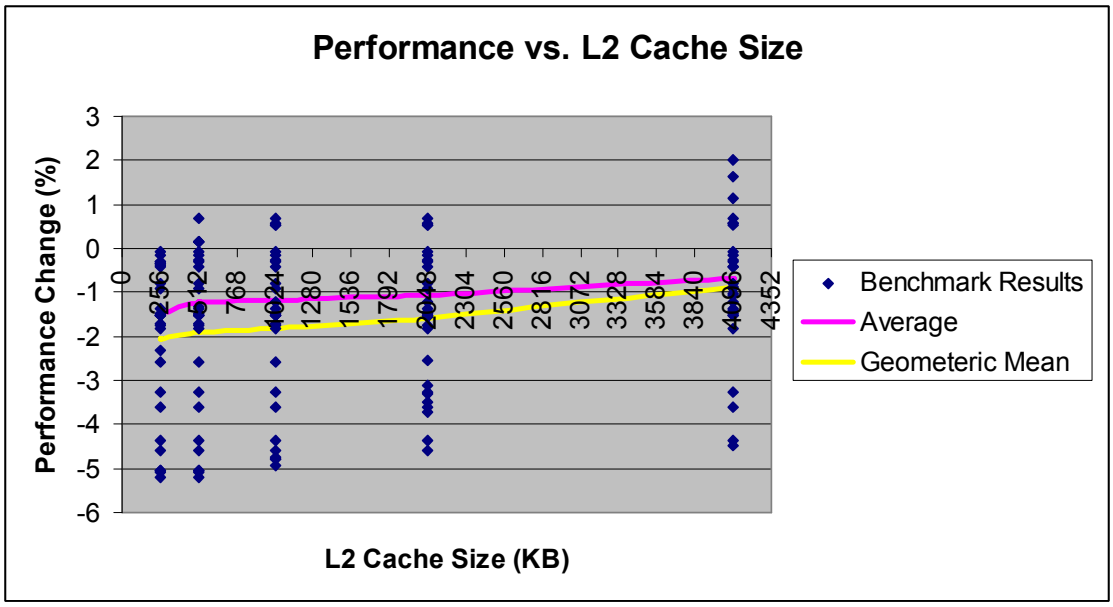

Figure 18: Category C – IPC Performance Difference vs. Baseline Processor

Figure 18 above indicates that the addition of a backside L2 cache provides little gain if any over the baseline processor as the average falls between -2% and 0% change. The decrease in performance can be attributed to the added latency required to access memory. The small gains for the larger cache sizes shows that some performance can be gained for the large size but even then the added latency still makes the average negative.

4.3.2. L2 Hit Rate

In category B, the L2 hit rate should be low as these tests have little locality and are memory sensitive. The scatter plot below shows the benchmarks on the x-axis with each vertical row revealing the performance gain for the various L2 sizes.

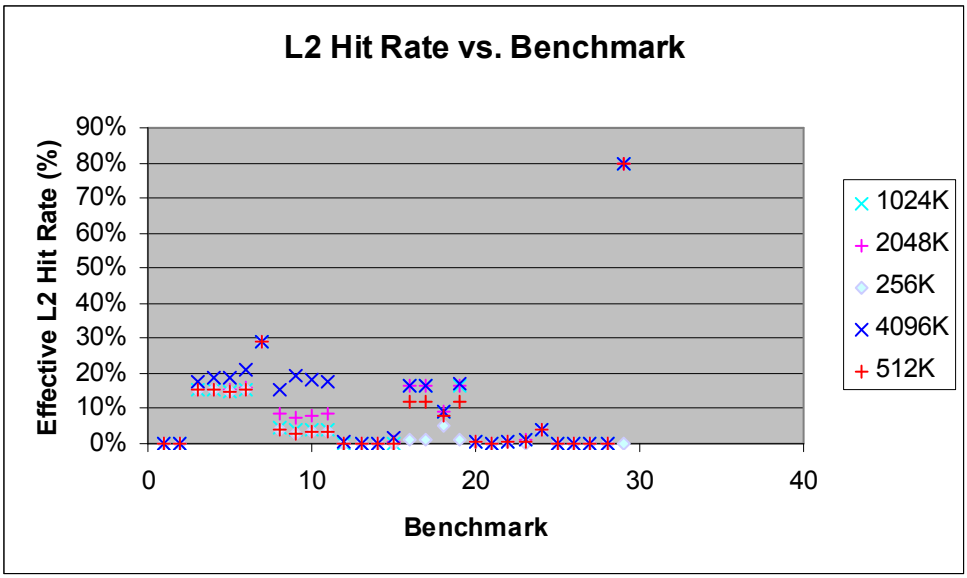

Figure 19: Category C – L2 Hit Rate vs. Benchmark

Figure 19 above shows that the hit rate for all sizes is relatively low. The size does play a part in the effective hit rate as the figure shows by the fact that the sizes are not all overlapping each other. The 4M is slightly larger which tracks with the performance for those benchmarks as seen in Figure 20 below. Notice that those cases where the size and hit rates overlapped in Figure 19 corresponds to those points in Figure 20 where the performance and size overlap as well.

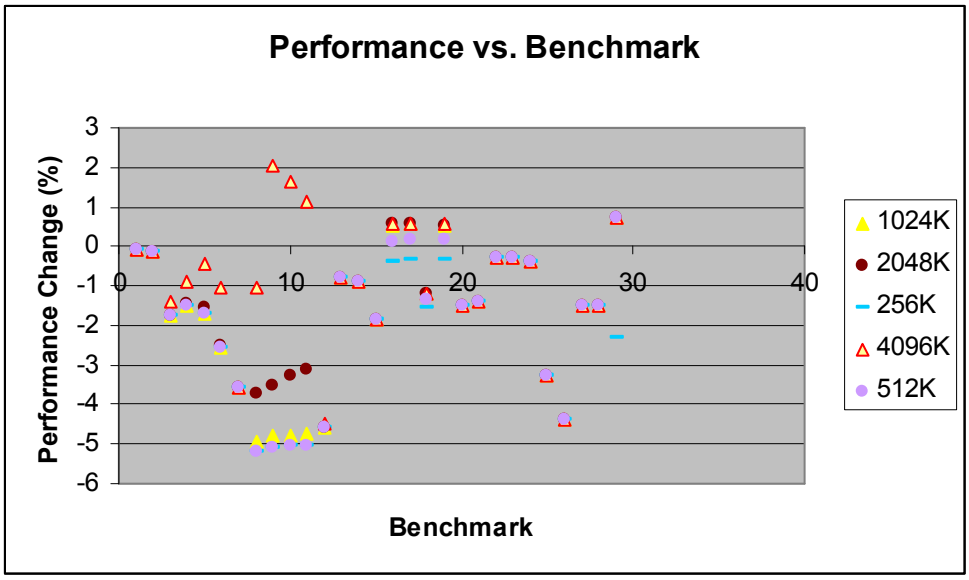

Figure 20: Category C – Performance vs. Benchmarks

Figure 20 shows that the performance gain/loss for each benchmark is fairly small (-6% to 3% range). The L2 size has an effect on some benchmarks which we observe at those points in the figure where the performance change/sizes do not overlap for that benchmark. If the bus protocol could be changed and the latency reduced, we would expect the performance to improve for these benchmarks. This category does show some degradation in performance though due to the extra latency of the backside L2 access. Keeping that access latency as low as possible will minimize the affect of the backside L2 addition.

4.3.3. CORE QUEUE SIZING

This category of benchmarks is complicated for queue sizing. One would think that increasing any queues associated with the memory subsystem would increase performance. However, the performance of the memory subsystem can only be improved as much as the rest of the core can make use of it. In a superscalar processor, if other

parts of the core become backed up, then increasing the memory subsystem portion will not yield better performance. Still, it is interesting to see how adding the backside L2 could affect applications dependent on memory.

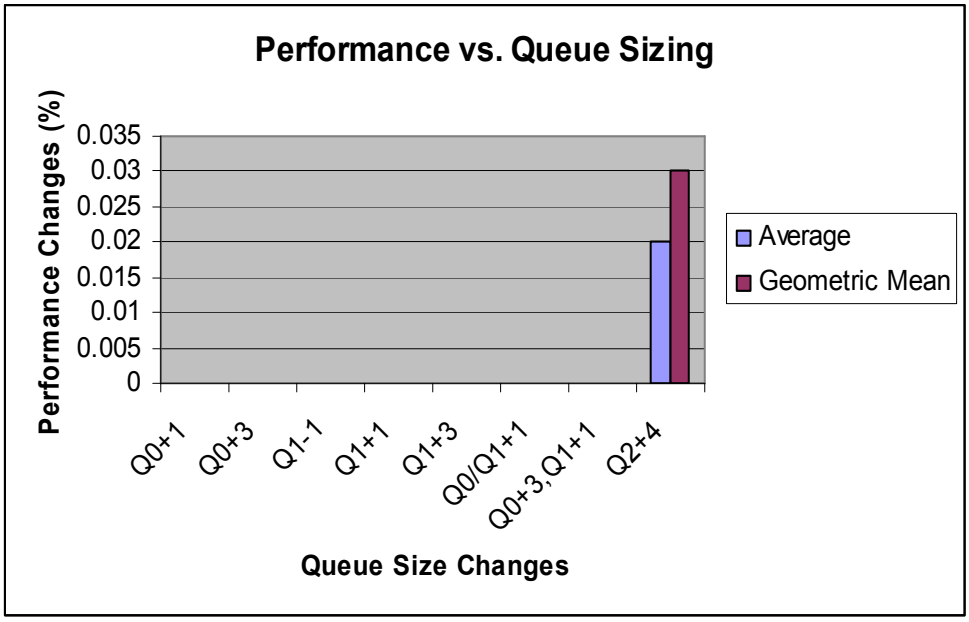

Figure 21: Category C – Performance vs. Queue Sizing

The figure above shows that queue changes had almost zero impact on the IPC performance. Only Q2 (Load Fill queue) had any impact at all as the rest of the performance changes were zero. The load fill queue is used for load misses. Since this category of benchmarks is accessing memory a great deal, it makes sense that this queue has some impact on performance. Increasing the amount of outstanding loads the processor can handle is useful for this category of benchmarks. This also shows that the queues are sized appropriately for accesses which do no rely on caches for their data.

4.3.4. SYSTEM BUS IMPACT

By measuring the performance impact of a system with a 3:1 bus ratio, we are able to see how the bus ratio may affect the L2 design. The following figure shows the

performance change of adding a backside L2 in a 3:1 bus ratio system. This figure looks similar to Figure 18 in 4.3.1 which leads us to the conclusion that the longer bus latency does not significantly impact the L2 design or size.

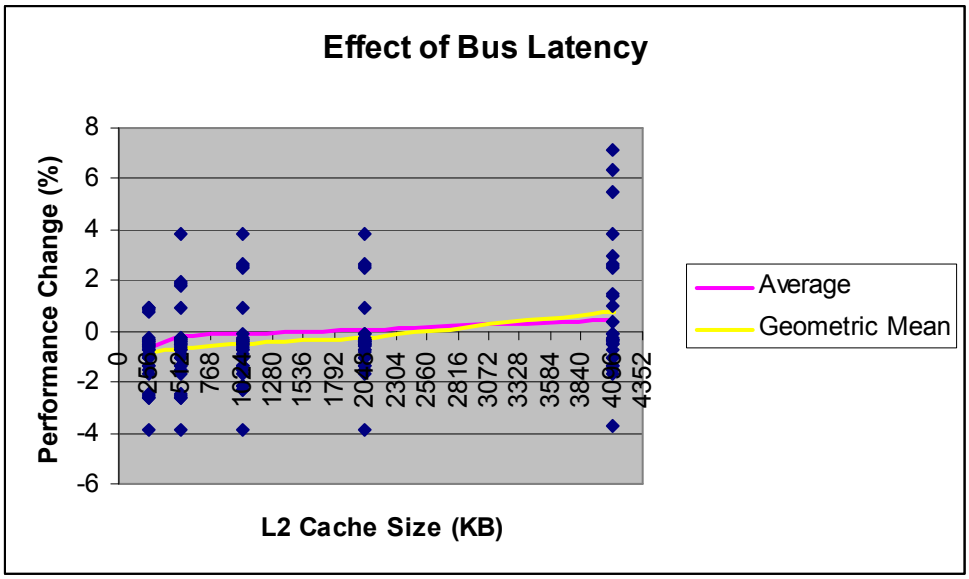

Figure 22: Category C – Bus Latency Effect on Performance

There is a slight increase in the relative performance of a 3:1 bus ratio system with backside L2 over a 1.5:1 bus ratio system with the same size backside L2. This can be seen by comparing Figure 22 above to Figure 18 which showed a similar curve with more negative IPC performance change. This performance increase is attributed to the small gain that can be made by hitting in the L2 versus having to access memory on the slower bus as the previous results showed that the backside L2 is being used in some benchmarks.

The performance of a 3:1 bus ratio system with backside L2 cache is actually better than a similar system with a 1.5:1 bus ratio. This is attributed to the fact that the backside L2 latency overwhelms the smaller bus ratio. With a larger bus ratio, the effect of the backside L2 is not felt quite as much.

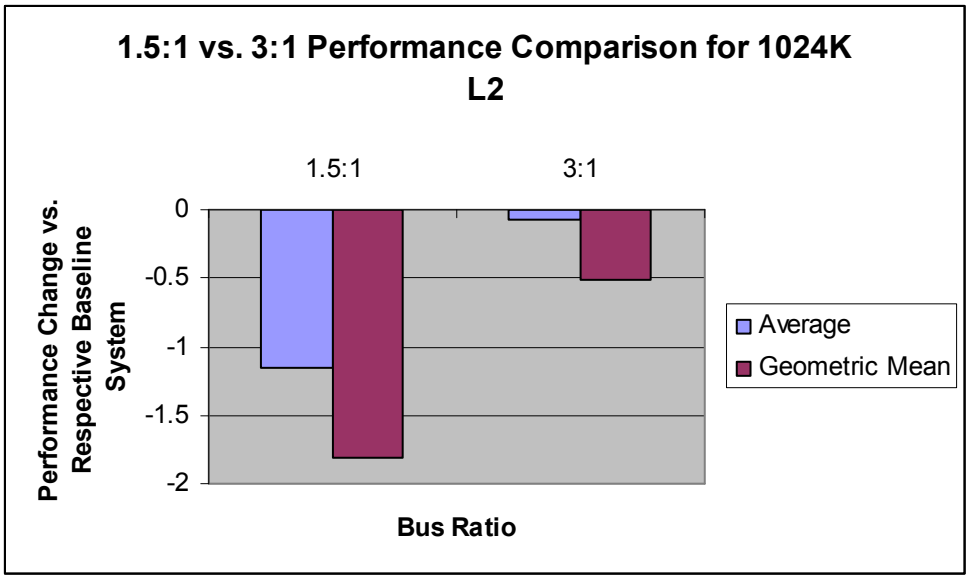

Figure 23: Category C – Performance Comparison for Different Bus Ratios

4.4 Benchmark Category BC – L2 and Memory Sensitivity

The benchmarks in category BC are sensitive to both memory hierarchy and memory latency. These benchmarks are usually fairly large tests to take advantage of the larger cache but without quite as much locality as in category B.

4.4.1. Effect of L2 Size

Because these benchmarks are affected by the L2, the results should show some improvement with a backside L2 cache. The scatter plot below shows the L2 size on the x-axis along with all of the performance of all of the benchmarks in this category in each vertical row with that size.

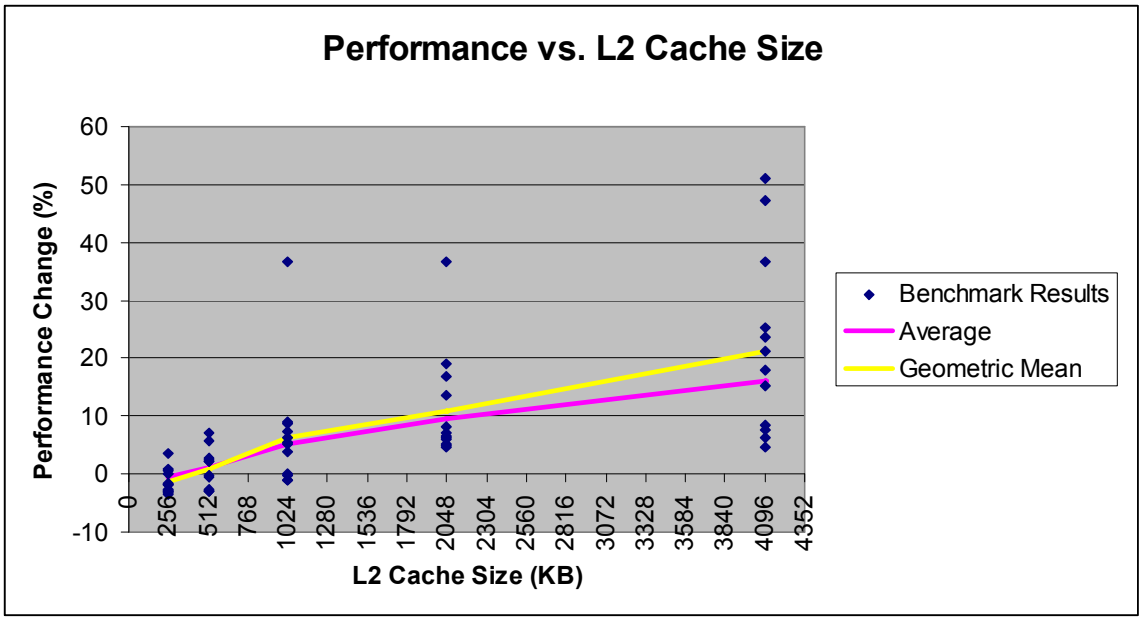

Figure 24: Category BC – IPC Performance vs. Baseline Processor

Figure 24 shows that the larger size caches do improve the performance as the majority of points are above 0% IPC change. However, we observe that these performance changes are smaller than category B. In these benchmarks, the performance improvement really does seem to be a combination of categories B and C. The performance change does not level off (which we saw in category C vs. category B), and the performance difference is significant. In this case, the added size of the 4M L2 cache shows over double the performance improvement of the 1M cache. This indicates that depending on the application, the larger cache may actually be beneficial and worth the extra hardware and larger processor.

4.4.2. L2 Hit Rate

With benchmarks that are sensitive to both memory and L2 we would expect to see something between the hit rates of categories B and C. The scatter plot below shows

the benchmarks along the x-axis. The hit rate of the various L2 sizes associated with each benchmark is plotted y-axis.

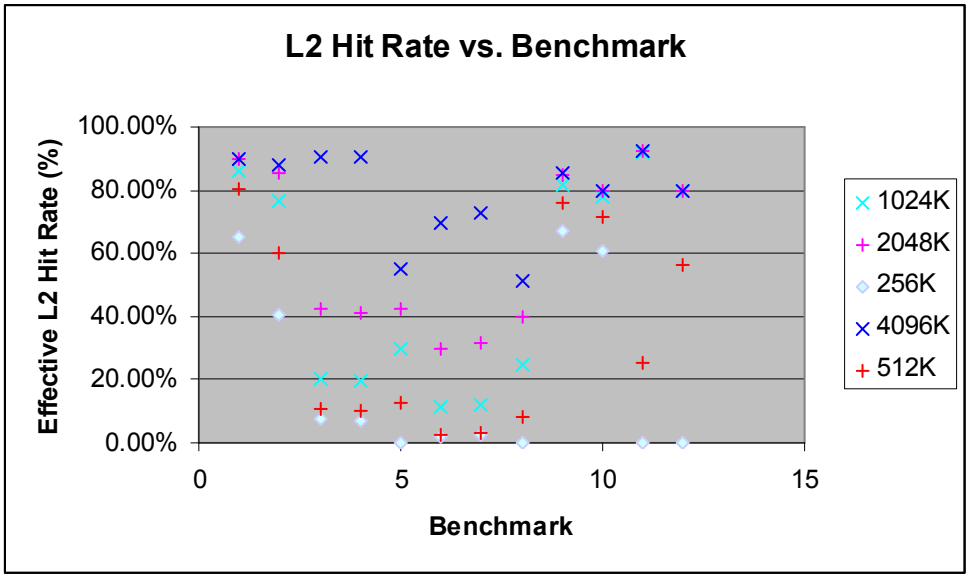

Figure 25: Category BC – Effective L2 Hit Rate

Figure 25 reveals that the hit rates are less than those in category B but are higher than those in category C. Here, we see that hit rates are relatively high for the larger cache sizes. This again shows that having the larger cache for this category of benchmarks is advantageous as the hit rate is higher for the larger caches in most of the benchmarks. The very small L2 size has the lowest hit rate across all the benchmarks.

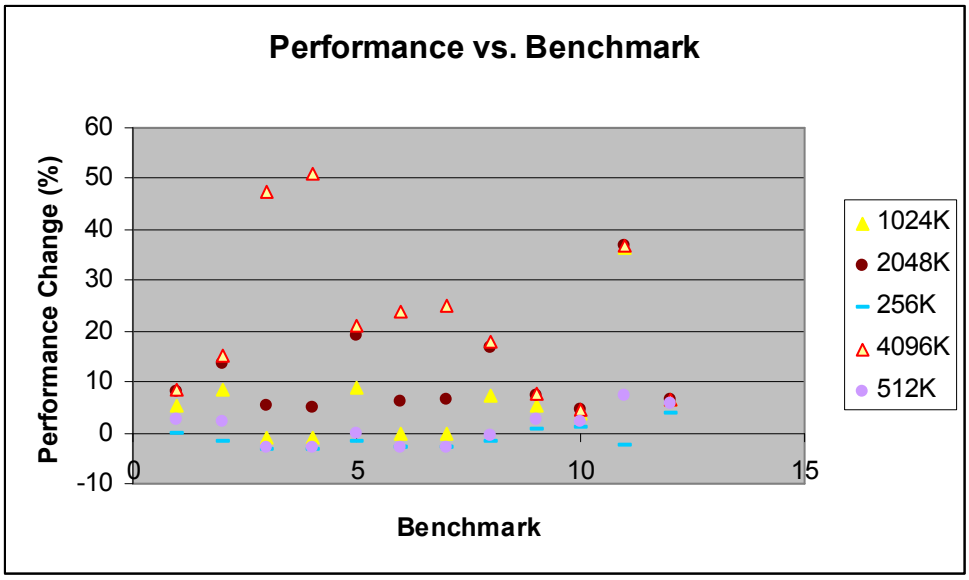

Figure 26: Category BC – Performance vs. Benchmarks

Figure 26 shows that the IPC performance change is also dependent on the L2 size for the various benchmarks. The scatter plot shows a larger performance gain (y-axis) for the larger caches for each benchmark (x-axis). The performance plot of the benchmarks shows that the 4M size L2 cache has a much greater performance gain than any other size. This tracks with the increased L2 hit rate for that size cache as well and with the mean performance difference numbers from Figure 24.

4.4.3. CORE QUEUE SIZING

Because these benchmarks are dependent on both memory and L2 cache, we would expect some compromise between the two categories. Figure 27 below shows the results of comparing the 1024K cache with no queue changes versus the changes seen below.

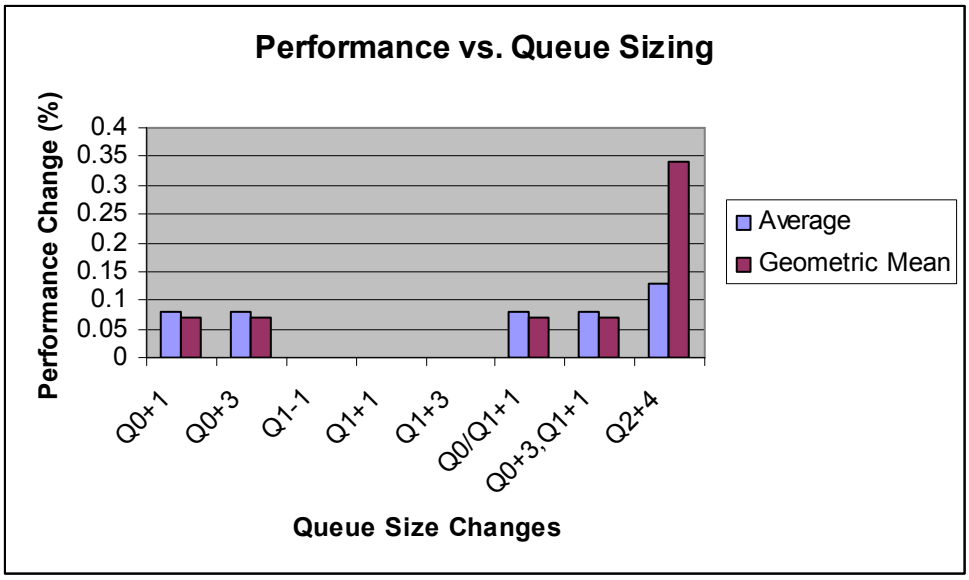

Figure 27: Category BC – Performance vs. Queue Sizing

Q1 again has had little impact on the performance while Q0 had some impact. The two findings are similar to what we saw for the category B benchmarks. Q2 showed the most change and follows what we observed for the category C benchmarks.

4.4.4. SYSTEM BUS IMPACT

If we look back at the results from categories B and C, we see that the performance change due to increasing the bus latency was more positive than in a 1:5.1 system. It follows that we would expect similar results in this category. Below is Figure 28 which shows the performance improvement.

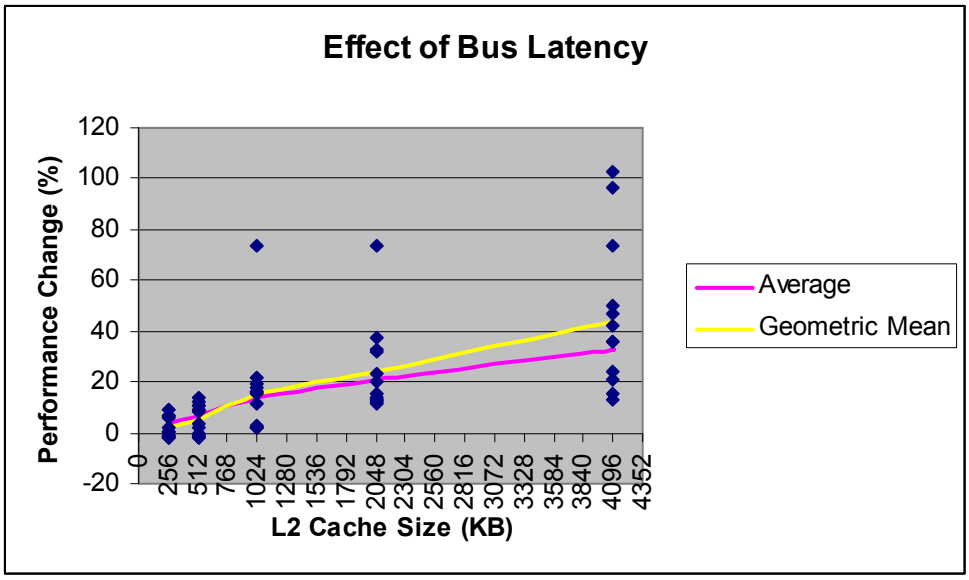

Figure 28: Category BC – Bus Latency Effect on Performance

Figure 28 does indeed show a combination of both categories. The performance change curve models those in category C. The greater performance change is similar to what we observed in category B. We see a large positive performance difference in this category because the backside L2 provides a means of accessing the data more quickly when that data makes its way into the L2 cache.

If we compare two systems with 1024K L2 cache, one running with a 1.5:1 bus ratio and one running with a 3:1 bus ratio, we see that the system with the 3:1 bus ratio is able to make greater performance gains. The backside L2 in this case significantly helps in reducing the amount of time spent accessing memory through the bus. The slower bus would decrease performance of a system without a backside L2. This figure shows how important a backside L2 becomes for those benchmarks that can make use of it as memory latency increases.

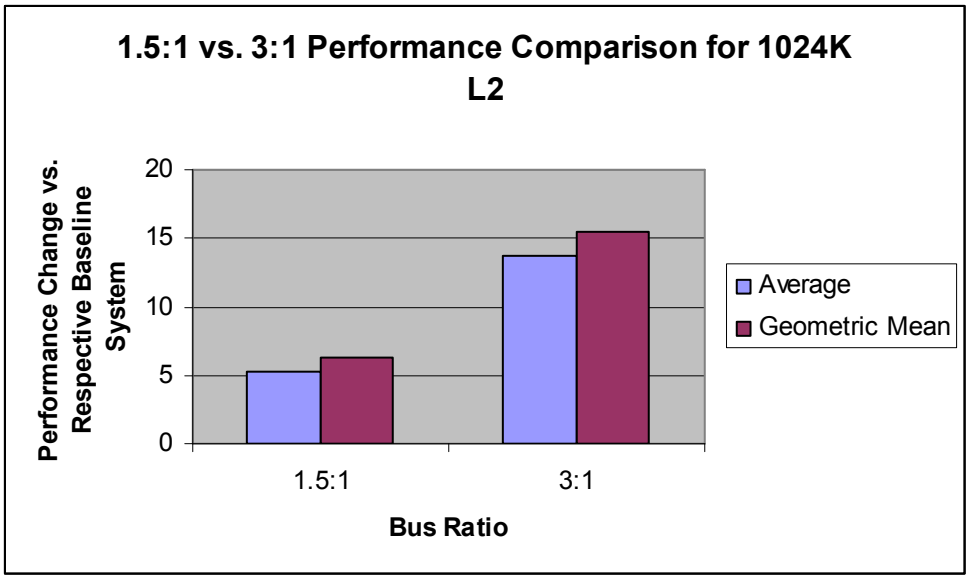

Figure 29: Category BC – Performance Comparison for Different Bus Ratios

5 CONCLUSIONS

The addition of a backside L2 to an existing design is not an easy endeavor. A study like this is important so that all aspects of the design can be simulated to make architectural decisions as well as see how beneficial it could be in a system. The worst outcome would be to add something to a design without simulation, thinking one is improving the performance, when actually some design decision is limiting that improvement. This analysis provides an in-depth look into designing an optimal L2 cache for a given baseline processor. It attempts to analyze the L2 design itself (size and hit rate) in addition to looking at how the L2 might impact the overall system (core queue sizing and bus ratio). It is clear that results are category and system dependent. A typical workload analysis for the intended system might be beneficial to realizing whether or not the backside L2 gains as much performance as required, making the extra hardware worthwhile.

The results showed that workloads with locality and a footprint bigger than the L1 size can make significant use of a backside L2. The size and hit rate analyses pointed to a 1024K L2 cache as the optimal size in terms of performance gained according to these benchmarks. If the core speed continues to increase relative to the bus speed, we will see that the processor speed to system speed ratio will increase. From these results, it is obvious that any system which increases its bus ratio will benefit from a backside L2.

An integral part of the design process is making decisions on queue sizes. Performance can be critically tied to these constraints if not optimized accordingly. By adding a backside L2 into the processor, a team would benefit from knowing if they adversely affected the performance. For this baseline processor, the results indicate the queues are appropriately sized and little performance can be gained by modifying them.

By looking through the individual queue statistics (results of which cannot reported here), I was able to see that the queue sizes were being effectively used.

One area not covered by this report is that of a multiprocessor system. More and more companies are designing SoCs (System on a Chip) which include multiple cores. As we saw with the increased bus latency, a multi-core system will only increase the contention on a bus like the PowerPC 60x bus. A follow-on to this study would be to model an actual multi-core system and observe how much contention might be placed on a bus like the 60x bus for different benchmarks. Then, an analysis of adding a backside L2 in this system would be an interesting addition to this project.

Simulators will continue to be a necessary part of the design process. They can reduce the design cycle time by helping designers make decisions before implementing all the changes. They also help the business groups and customers to see where performance can be traded for increased hardware. Knowing the typical workload of the customer can help to simulate what performance enhancements make the most sense for customers. Design trade-offs are a necessary part of processor design. Simulations and analysis like the one conducted in this project help to make the right decisions in that process.

AppendixA

The following table details some information about the SPEC benchmarks used in this analysis. It shows the memory footprint (in megabytes) of the individual benchmarks. This is helpful in understanding the way they were categorized for this project. The information can be found at: http://www.spec.org/cpu2000/analysis/memory/.

Benchmark	***Category***	***Resident Size (MB)***	***Virtual Size (MB)***
200.sixtrack	A	26	59.8
252.eon	A	.6	1.5
175.vpr	B/BC	50	53.6
178.galgel	B	63	155
181.mcf	B/BC	190	190
300.twolf	B	3.4	4
171.swim	C	191	192
183.equake	C	49	49.4
254.gap	C	192	194
168.wupwise	C	176	177
187.facerec	BC	16	18.5
197.parser	BC	37	66.8

Table 5: Memory Footprint for SPEC Benchmarks

References

[1] Diefendorff, K., "PC Processor Microarchitecture", *Microprocessor Report*, vol. 13, no. 9, pp. 16-22, July 12 2000.

[2] Freescale Semiconductor. PowerPC Microprocessor Family: The Bus Interface for 32-Bit Microprocessors. 2004.

[3] Hinton, G., et al., "The Microarchitecture of the Pentium 4", *Intel Technology Journal*, vol. 5, no. 1, pp. 1-13, February 2001.

[4] Jackson, K. M. and Langston, K. N., "IBM S/390 storage hierarchy G5 and G6 performance considerations", *IBM J. Res. Develop.*, vol. 43, no. 5/6, pp. 847-854, September/October 1999.

[5] Jouppi, N.P., Wilton, S. J. E., "Tradeoffs in two-level on-chip caching", *Proceedings of 17th International Symposium on Computer Architecture*, " pp. 35-45, April 1994.

[6] Keshava, J. and Pentkovski, V., "Pentium III Processor Implementation Tradeoffs", *Intel Technology Journal*, vol. 3. no. 2, pp. 1-11, May 1999.

[7] Przybylski, S., Horowitz, M., Hennessy, J., "Performance tradeoffs in cache design", *Proceedings of 15th International Symposium on Computer Arhictecture*, pp. 290-298, May/June 1988.

[8] Przybylski, S., Horowitz, M., Hennessy, J., "Characteristics of Performance-Optimal Multi-Level Cache Hierarchies", *Proceedings of 16th International Symposium on Computer Architecture*, pp. 114-121, May/June 1989.

[9] Reinold, J., "Performance Implications of Next Generation PowerPC Microprocessor Cache Architectures", *Proceedings IEEE Compcon*, pp. 331-338, Feb. 1997.

[10] Shin, J. L., Petrick, B., Singh, M., and Leon, A. S., "Design and implementation of an embedded 512-KB level-2 cache subsystem", *IEEE J. Solid-State Circuits*, vol. 40, no. 9, September 2005.

[11] Short, R. T., Levy, H. M., "A simulation study of two-level caches", *Proceedings of 15th International Symposium on Computer Architecture*, " pp. 81-88, May/June 1988.

[12]Smith, A. J., "Cache Memories", *ACM Computing Surveys*, vol. 14, no. 3, pp. 473-530, September 1982.

[13] Vintan, L., Armat, C., Steven, G., “The impact of cache organization on the instruction issue rate of a superscalar processor”, *Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing*, pp. 58-65, Feb. 1999.

Vita

Kathryn Hammel, daughter of John and Linda Hammel, was born in Indianapolis, Indiana on September 25, 1978. She completed her Bachelor of Science in Computer Engineering at the University of Notre Dame in 2000, graduating Summa Cum Laude. After graduation, Kathryn moved to Austin, Texas to work for Freescale Semiconductor (formerly Motorola Inc.) in August 2000. She participated in the Engineering Rotation Program for the first year before taking a permanent position as a logic designer where she is still currently employed. She joined the University of Texas at Austin graduate program in Computer Engineering in the Fall of 2002 and has attended part-time since then.

Permanent address: 3723 Harvey Penick Dr., Round Rock, TX 78664

This dissertation was typed by Kathryn Hammel.