Quantitative study of data caches on a multistreamed architecture

Mario Nemirovsky
University of California, Santa Barbara
mario@ece.ucsb.edu

Wayne Yamamoto
Sun Microsystems, Inc.
wayne.yamamoto@sun.com

Abstract

In this paper, we quantify the effect that fine grained multistreamed interaction of threads within a shared cache has on the miss rate. By concentrating on the miss rate, we focus on just the cache performance and separate ourselves from a given system architecture. We show the effects of cache capacity, associativity, and line size on the miss rates of multistreamed workloads of two, three and four streams. We then separate the individual components of the miss rate into misses that are intrinsic to the workload (i.e., misses present in a non-multistreamed processor) and extrinsic (i.e., misses introduced due to the multistreamed nature of the processor). Our preliminary results show that for two streamed workloads the increase in miss rate due to multistreaming is less than 60% for caches with some associativity. We observe that for larger cache capacities, direct mapped caches and caches with larger line sizes exhibit poor performance due to a significant increase in the number of extrinsic misses.

1 Introduction

Multistreaming is emerging as a technique to further exploit parallelism beyond the realm of today's high performance, out-of-order, superscalar processors. Rather than exploit parallelism at the instruction level, multistreaming exploits thread level parallelism. In fine grained multistreaming, instructions from different threads are issued each cycle and concurrently executed. Thus, processors employing this technique are able to tolerate short latencies due to pipeline hazards as well as long latencies due to memory and network access. Early fine grained processors, like the CDC6600 peripheral processor [Thor64] and the HEP [Smit81], limited each stream to a single instruction executing in the pipeline at any given time. While this eliminated all pipeline interlocks which simplified the design, a large number of threads were needed to fully utilize the pipeline, i.e., enough to fill all the pipe stages and mask the memory latency. Recent commercial and research projects in fine grained multistreaming have focused on multiprocessor systems, e.g., Tera Computer System [Alve90], DART [ShBu91] and CCMP [Butn86], or special purpose processors, ˝e.g., MASA [Hals88] and the multistreamed CRAY-1 [Farr91].

In [Nemi90, Nemi91], the DISC processor used a technique called dynamic interleaving, a fine grained multistreaming technique developed for real time systems that allows multiple instructions from a given stream in the pipeline at a given time. Using dynamic interleaving, a single stream running alone is allocated the full bandwidth of the processor. If more than one stream is running, they share processor and memory resources. In [SeYa94][Yama95][Tul95], the dynamic interleaving scheme was extended to general purpose superscalar processor architectures. Here instructions from different streams were dynamically grouped by the processor and simultaneously dispatched to the functional units based upon the scheduling criteria. Due to the fine grained nature of this scheme both short latencies, due to pipeline hazards, and long latencies, due to cache misses, are tolerated. In these architectures the different streams share processor resources like functional units and caches.

While cache performance has been well studied in both the single stream [Przy90, Smit82] and multiprogramming environments [Agar88, Mogu91], very little work has been done in multistreamed caches. Work in fine grained multistreaming has focused on overall system performance rather than concentrating on the caches [Tul95][Yama94][Yama95]. In this paper, we quantify the effect that fine grained multistreamed interaction of threads within a shared cache has on the miss rate. By concentrating on the miss rate, we focus on just the cache performance and separate ourselves from a given system architecture. We show the effects of cache capacity, associativity, and line size on the miss rates of multistreamed workloads of two, three and four streams. We then separate the individual components of the miss rate into misses that are intrinsic to the workload (i.e., misses present in a non-multistreamed processor) and extrinsic (i.e., misses introduced due to the multistreamed nature of the processor). The description of the simulation experiment is given in section 2, followed by a discussion of the workloads in section 3. Section 4 discusses our results and section 5 concludes.

2 Simulation Experiment

To measure the effect of fine grained interleaving on the cache, we ran multistreamed simulations and logged the cache performance. The objectives were to quantify the effects multistreaming has on cache performance, determine the multistreamed cache design tradeoffs, and to compare ˝multistreamed versus single stream cache design.

A common methodology to measure cache performance in a single stream processor is to run an instruction trace through a cache simulator and measure the miss rate as the cache parameters are varied. In developing the methodology for this study an interesting problem arose. In multistreamed processors, cache misses can effect the scheduling of threads within a multistreamed processor and, therefore, the address trace sent to the cache. For instance, a cache configuration with a high miss rate would result in different instruction scheduling behavior than a cache configuration with a low miss rate. Thus, misses caused by the cache configuration and those caused by ˝multistreamed scheduling are indistinguishable. In this paper, our goal is to isolate the effect of the cache configuration on the cache performance. This involves creating an address trace that normalizes the effect of multistreamed scheduling misses ˝while still keeping the multistreamed characteristics.

To accomplish this, we obtained single stream execution traces containing both instruction and data addresses from each benchmark. Then, the multistreamed traces are constructed by interleaving the single stream traces such that each stream alternately issues an instruction in a round robin fashion, similar to the fine grained multistreaming execution of the HEP [Smit81]. Using this method, we guarantee that the address traces generated by a program are the same across all simulation runs and, therefore, normalize the effect of ˝multistreamed scheduling.

Interleaving the traces in this fine grained manner is finer than would occur in a multistreamed, superscalar processor as in [Tul95][Yama95]. Thus, we would expect more misses to occur and our results are probably more pessimistic than would actually occur. However, since our goals are to quantify the effect fine grained interleaving has on the cache and compare to single stream performance, the results should represent the ˝worse case behavior.

We developed multistreamed workloads that interleaved from two to four threads. For each workload, we measured the cache miss rate for various cache configurations. We simulated cache configurations varying cache capacity from 256 bytes to one Mbytes. Associativities simulated are direct mapped, two way, and four way set associative. All caches are write allocate and associative caches use a least recently used (LRU) replacement policy. Line sizes simulated are 16 bytes, 32 bytes, and 64 bytes.

3 Workload

Table 1 shows the benchmarks used in this study. They consist of common UNIX benchmarks, including some of the SPEC92 benchmarks, Stanford small benchmarks, and dhrystone. All benchmarks were compiled for the IBM RS/6000 using the IBM XL C compiler with the -O switch. Fortran benchmarks were converted to C using a fortran-to-C converter. Each benchmark was traced for five million instructions after a one million instruction transient. The % Memory column in Table 1 shows the percentage of instructions that accessed the data cache. The addresses in the traces are virtual addresses generated by the program.

programs	% Memory	programs	% Memory	programs	% Memory	programs	% Memory
dhry	31%	espresso	26%	gcc	37%	li	41%
eqntott	34%	fpppp	59%	hanoi	32%	spice	37%

Table 1. Benchmark programs and the percentage of memory ˝instructions.

Table 2 lists the multistreamed workloads employed. The first column shows the number of programs involved, and the second column lists the benchmarks in the workload.

# of streams	Programs	# of streams	Programs
2	gcc-dhrystone	2	eqntott-li
2	espresso-dhrystone	2	spice-li
2	espresso-eqntott	3	dhry-espresso-li
2	espresso-li	3	fpppp-espresso-li
2	fpppp-eqntott	4	dhry-espresso-hanoi-li
2	fpppp-spice	4	fpppp-espresso-eqntott-li
2	fpppp-li		

Table 2. Multistreamed workloads.

4 Discussion

We use the cache miss rate as a measure of the cache performance. The miss rate is calculated as the number of cache misses divided by the number of cache accesses. We initially show the effects of capacity, associativity, and line size on the overall cache miss rate of multistreamed workloads. We then further classify cache misses into three categories based upon Agarwal's model [Agar89]. The first class is non-stationary misses which are due to the initial reference to an address. Second, intrinsic misses are due to multiple addresses within the same thread mapping to the same set. Finally, extrinsic

misses are caused by multiple addresses from different threads mapping to the same set. In this study, we are primarily concerned with measuring the effect of fine grained multistreaming on cache performance and, thus, the extrinsic misses. A miss is classified as an extrinsic miss if it would not have occurred when the program is executing alone. To quantify the cache degradation caused by multistreaming, we calculate the percent increase in the miss rate due to extrinsic misses. We refer to this value as the multistream cache *interference*.

4.1 Overall Miss Rates

Figures 1 and 2 chart the miss rate versus cache capacity curves for a line size of 16 bytes as the associativity is varied for typical two stream workloads. For all cache capacities, increasing associativity decreases the miss rate. The largest decrease in miss rate occurs in moving from a direct mapped cache to a two way set associative cache, 35% average decrease in miss rate. In moving from a two way set associative cache to a four way set associative cache the average decrease was 14%. Compared to single stream results [Yama96], associativity has a larger effect in multistreamed caches. On average, in a single stream, increasing the associativity from direct mapped to two way set associative decreased the miss rate by 25% and from two way to four way set associative the decrease was 7%, compared to 35% and 14% for the multistreamed cache respectively.

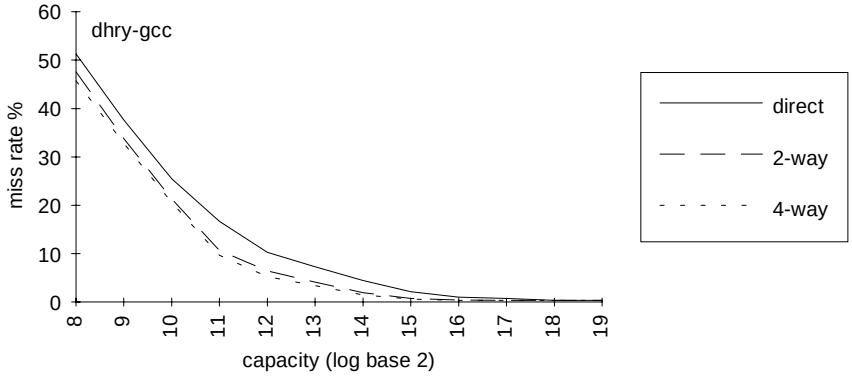

Figure 1. Miss rate versus the cache capacity as the ˝associativity varies for dhry-gcc.

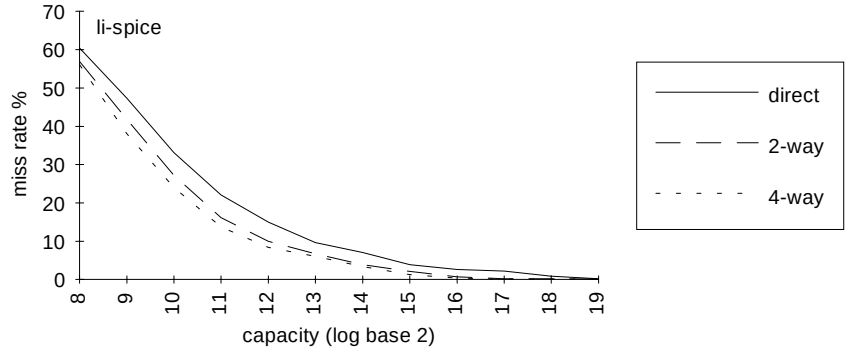

Figure 2. Miss rate versus the cache capacity as the ˝associativity varies for li-spice.

Figures 3-5 chart the miss rate versus cache capacity curves as the line size is varied for two to four stream workloads. The charts show that for small caches the smaller line sizes result in a lower miss rate and for large caches larger line sizes are better. In the two stream workload (Figure 3), the crossover point where large line sizes achieve lower miss rate occurs for cache sizes between 11 and 12, i.e., 2K and 4K bytes. As the number of streams is increased, the crossover point moves towards larger cache. For three streams, the crossover point occurs close to 4K bytes and, for four streams, the crossover occurs between 4K and 8K bytes. As the number of streams increases, the contribution of the extrinsic misses to the overall miss rate also increases. Increasing the line size reduces the number of lines in the cache for a given capacity resulting in an increase both intrinsic and extrinsic misses. Thus, as the number of streams increases, smaller line sizes are better because they reduce the level of these misses. Interference affects the cache performance of the individual programs with varying degree. One constant is that the program that experiences more misses per unit time, i.e. has a higher bus utilization, generates more interference. To understand this phenomenon, consider a multistreamed workload of two programs. Suppose, the first program exhibits a low miss rate due to better locality and operates in a small fraction of the cache. The

second program has the higher miss rate due to poor locality and operates in a larger fraction of the cache. Since the second program operates in a larger fraction of the cache, it has a higher chance of interfering with the first program. Due to its tight locality, the first program will have a lower probability of interfering with the second program. In addition, due to the first program's low miss rate, an extrinsic miss has a larger effect on the overall miss rate. Thus, the first program incurs a higher level of interference.

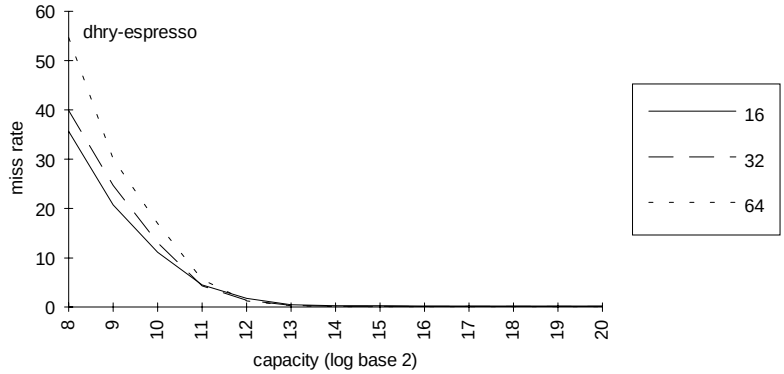

Figure 3. Miss rate versus the cache capacity as the line size ˝varies (2 stream workload).

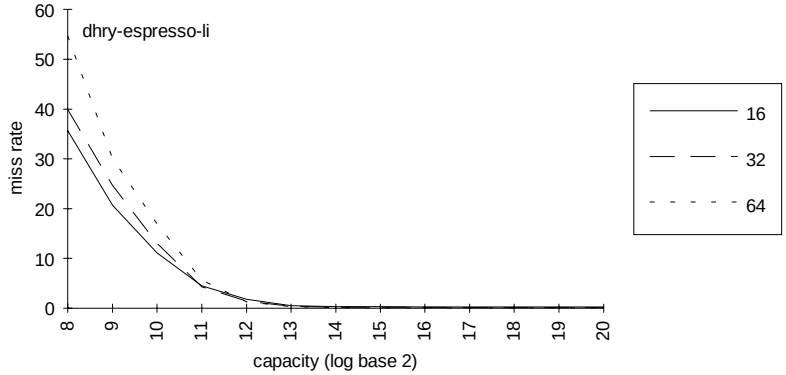

Figure 4. Miss rate versus the cache capacity as the line size ˝varies (3 stream workload).

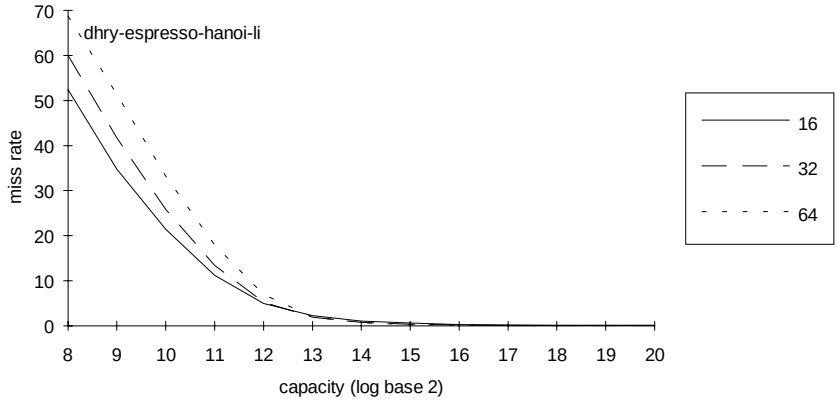

Figure 5. Miss rate versus the cache capacity as the line size ˝varies (4 stream workload).

4.2 Extrinsic Miss Rates and Interference

Figure 6 chart the miss rate versus the cache size for a direct mapped, 16-byte line cache for the fpppp-spice and the espresso-li workloads. The behavior exhibited in the charts is typical for the workloads executed. The miss rate separated into two categories: extrinsic miss rate (ext) and the miss ˝rate due to the intrinsic and non-stationary misses (int). The charts show that the extrinsic misses significantly contribute to the overall miss rate and increasing the cache size reduces the number of extrinsic misses.

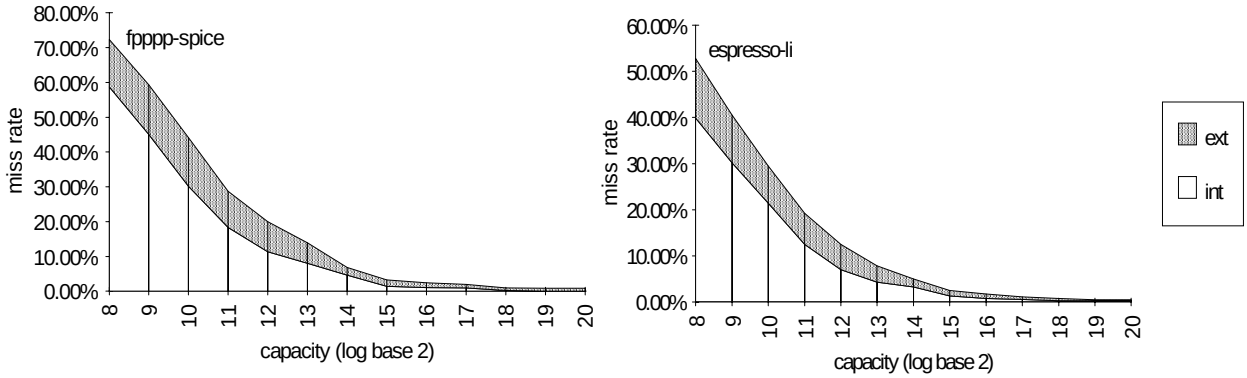

Figure 6. Extrinsic and intrinsic miss rates (direct mapped, ˝16 byte line).

Figure 7 charts the interference, i.e., the increase in the miss rate due to extrinsic misses, averaged over all two stream workloads versus the cache capacity for the various associativities. The cache line size of 16 bytes is used in all cases. For cache capacities of 16K or less, interference ranged from 30 to 60 percent and slightly increases as the cache capacity increases. Increasing the associativity does not have a significant effect on interference for these capacities. In fact, the interference actually increases as the associativity increases for caches smaller than 1K. Small caches, especially direct mapped, are dominated by capacity misses [Henn90] which can only be reduced by increasing the cache capacity. For larger caches, associativity has a much larger effect. For direct mapped caches, interference continues to increase as the capacity increases. For caches with associativity, increasing the capacity decreases and eventually eliminates interference. In large caches, moving from a direct mapped to a two way set associative cache decrease interference by more than 50%. Four way set associative caches exhibit slightly lower interference levels than two way set associativity. In larger caches, non-stationary and extrinsic misses dominate and increasing associativity has a larger effect on decreasing these misses than increasing capacity.

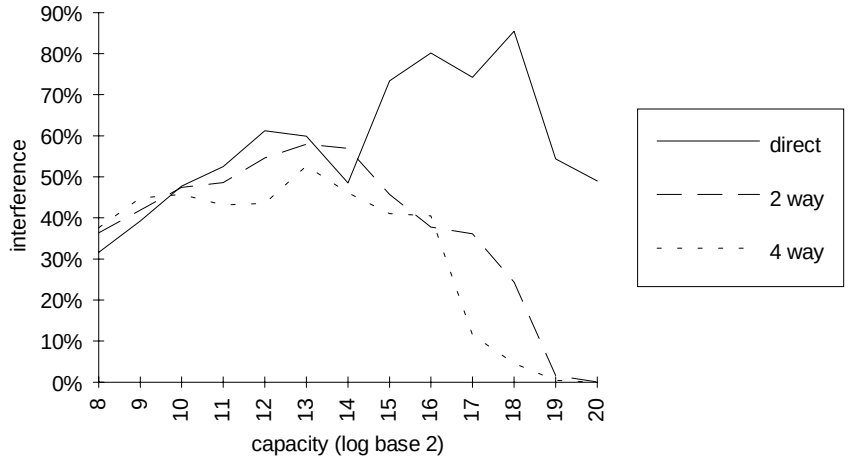

Figure 7. Interference versus the cache capacity as the ˝associativity varies (16 byte line, 2 stream workloads).

For a given capacity, increasing the line size reduces the total number of lines in the cache. In single streamed caches, this results in an increase in intrinsic misses. In multistreamed caches, both intrinsic and extrinsic misses increase. Figure 8 charts the interference versus the cache capacity as the line size is varied. The results are averaged over all two stream workloads using four way set associativity. The chart shows in smaller caches the line size has the largest effect on interference. For larger caches, the effect of line size is not as significant. Due to spatial locality within programs, increasing the line size decreases the number of non-stationary (first reference) misses [Henn90]. In large caches, non-stationary misses dominate the miss rate and, therefore, increasing the line size results in a lower miss rate. In smaller caches where the miss rate is dominated by intrinsic and extrinsic misses, smaller line sizes are better because they reduce the level of these misses.

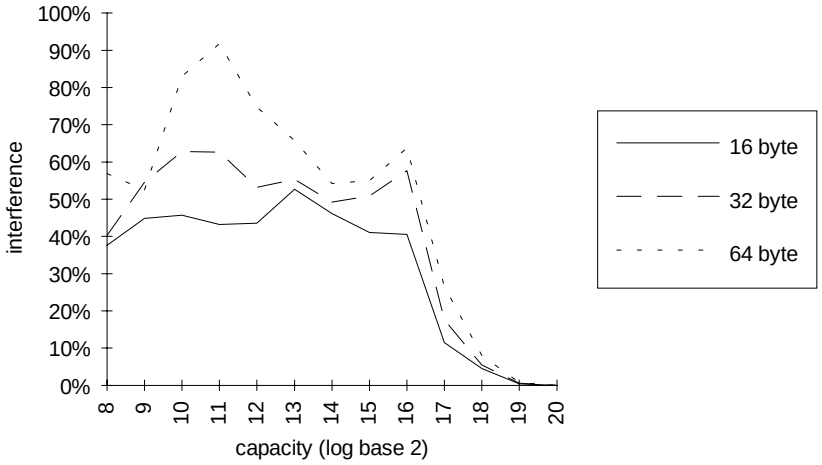

Figure 8. Interference versus the cache capacity as the line ˝size varies (4 way set associative, 2 streams).

As the number of streams increases, more threads are competing for the limited cache resources and this results in higher interference levels. Figure 9 shows the interference versus the ˝cache size as the number of streams is varied. The results are averaged over all workloads with the same number of streams using a two way set associative, 16 byte line cache. The interference for three and four streams is significantly higher than two stream interference. The largest difference occurs for medium size caches where the interference peaks at 200% and is almost four times larger than the two stream result. As the cache capacity increases, the difference between the four stream and the three stream interference decreases. In fact, for large caches, the four stream workloads showed less interference in than three stream workloads. This is probably due to the slightly different program mix selected for the four stream workloads. Table 2 shows that four stream workload dhrystone-espresso-hanoi-li was constructed by adding hanoi to the three stream workload dhrystone-espresso-li and the fpppp-espresso-eqntott-li workload was constructed by adding ˝eqntott to the three stream workload fpppp-espresso-li. Since both hanoi and eqntott have a miss ratio lower than the average of the corresponding three stream workload, this had the effect of lowering the overall miss ratio once the cache capacity ˝was large enough to eliminate most of the extrinsic misses.

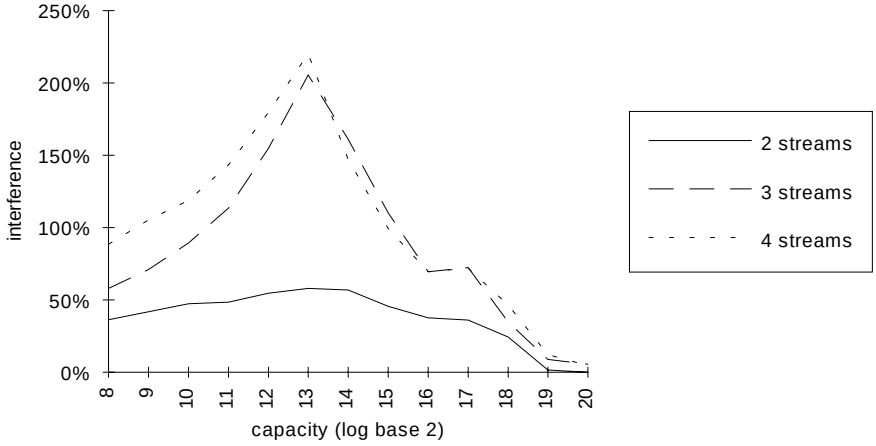

Figure 9. The interference as the number of streams varies (2 ˝way set associative, 16 byte line).

5 Conclusion

In designing a cache for a multistreamed processor, both a low ˝miss rate and low interference levels are necessary to achieve the highest performance. In this paper, we have examined the individual effect of the different cache configurations on interference and overall miss rate. Here we combine these results to determine the tradeoffs in the design of multistreamed caches.

Table 3 summarizes the effect each cache design parameter has on the cache performance measures. For the typical cache capacity in today's high end microprocessors, ~32K bytes, associativity is a requirement. In the design of the next generation cache, a question arises on whether to increase associativity or increase capacity? Our results show that for a cache of 32K bytes, doubling associativity improves the multistreamed cache performance more than doubling cache

capacity. For small caches, it is better to increase the capacity. Small caches especially direct mapped, are dominated by capacity misses which can only be reduced by increasing the cache capacity. In larger caches, non-stationary and extrinsic misses dominate and increasing associativity has a larger effect ˝on decreasing these misses than increasing capacity.

When compared to a single stream cache, multistreamed caches suffer an additional degradation due to the increase in extrinsic misses as the line size is increased. This is an important consideration in choosing the right size. Thus, the decrease in multistream miss rate is smaller than in a single ˝stream cache and a smaller line size is better.

Cache Parameter	Interference	Overall miss rate
increasing capacity	increases for direct mapped decreases for set associative and large capacities	decreases, especially for smaller capacities
increasing associativity	increases for very small caches decreases, especially for large capacities	decreases, especially for larger capacities
increasing line size	increases	decreases for large capacities increases for small capacities

Table 3. The effect of varying cache parameters on cache ˝performance measures.

This work represents the first step in understanding the effect of multistreaming on cache design. Future work will include studying the effects of multistreaming on both instruction and data caches, multilevel cache hierarchies, including various thread scheduling algorithms, and improving our experimental ˝methodology.

References

[Agar88] A. Agarwal, J. Hennessy, and M. Horowitz, "Cache Performance of Operating System and Multiprogramming Workloads," *ACM Transactions On Computer Systems*, vol. 6, no. 4, Nov. 1988, pp. 393-431.

[Agar89] A. Agarwal, M. Horowitz, and J. Hennessy, "An Analytical Cache Model," *ACM Transactions On Computer Systems*, vol. 7, no. 2, May 1989, pp. 184-215.

[Alve90] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, B. Smith, "The Tera Computer System," *Proceedings of Supercomputing '90*, 1990, pp. 1-6.

[Butn86] S. E. Butner and C. A. Staley, "A RISC Multiprocessor Based on Circulating Context," *Proceedings of the IEEE Phoenix Conference on Computers and Communications*, March 1986, pp. 620-624.

[Farr91] M. K. Farrens and A. R. Pleszkun, "Strategies for Achieving Improved Processor Throughput," *Proceedings of The 18th Annual International Symposium on Computer ˝Architecture*, May 1991, pp. 362-369.

[Hals88] R. H. Halstead and T. Fujita, "MASA: A Multithreaded Processor Architecture for Parallel Symbolic Computing," *Proceedings of The 15th Symposium on Computer Architecture*, June 1988, pp. 443-451.

[Henn90] J. L. Hennessy and D. A. Patterson, *Computer Architecture, A Quantitative Approach*, Morgan Kaufmann Publishers, 1990.

[Mogu91] J. C. Mogul and A. Borg, "The Effect of Context Switching on Cache Performance," *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, April 1991, pp. 75-84.

[Nemi90] M. D. Nemirovsky, "DISC, A Dynamic Instruction Stream Computer," Ph.D. Dissertation, University of California, Santa Barbara, September 1990.

[Nemi91] M. D. Nemirovsky, F. Brewer, and R. C. Wood, "DISC: Dynamic Instruction Stream Computer," *Proceedings of the 24th International Symposium and Workshop on ˝Microarchitecture*, 1991.

[Przy90] S. Przybylski, Cache and Memory Hierarchy Design, Morgan Kaufmann Publishers, Inc., 1990.

[SeYa94] M. J. Serrano, W. Yamamoto, A. R. Talcott, R. C. Wood, and M. D. Nemirovsky, "A Model for Performance Estimation in a Multistreamed, Superscalar Processor," *Proceedings of the Seventh International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, Lecture Notes in Computer Science, Springer-Verlag, May 1994, pp. 353-368.

[ShBu91] J. Shetler and S. E. Butner, "Multiple Stream Execution on the DART Processor," *Proceedings of the 1991 International Conference on Parallel Processing*, August 1991, pp. 92-96.

[Smit82] A. J. Smith, "Cache Memories," *ACM Computing Surveys*, vol. 14, no. 3, 1982, pp. 473-530.

[Smit81] B. J. Smith, "Architecture and Applications of the HEP multiprocessor Computer System," *Proceedings of SPIE*, Vol 298, Real Time Signal Processing, 1981, pp. 241-248.

[Thor64] J. E. Thornton, "Parallel Operation in the Control Data 6600," *Proceedings of the Spring Joint Computer Conference*, 1964.

[Tull95] D. M. Tullsen, S. J. Eggers, H. M. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proceedings of The 22nd Annual International Symposium on Computer Architecture*, May 1995, pp. 392-403.

[Yama94] W. Yamamoto, M. J. Serrano, A. R. Talcott, R. C. Wood, and M. D. Nemirovsky, "Performance Estimation of Multistreamed, Superscalar Processors," *Proceedings of the 27th Hawaii International Conference on System Sciences*, January 1994, pp. 195-204.

[Yama95] W. Yamamoto and M. Nemirovsky, “Increasing Superscalar Performance Through Multistreaming," *Parallel Architectures and Compilation Techniques*, June 1995, pp. 49-58.

[Yama96] W. Yamamoto, “An Analysis of Multistreamed, Superscalar Processor Architectures," Ph.D. Dissertation, University of California, Santa Barbara, December 1995.