MULTI-LEVEL CACHE HIERARCHY EVALUATION FOR PROGRAMMABLE MEDIA PROCESSORS

Jason Fritts	**Wayne Wolf**
Computer Science Dept.	Electrical Engineering Dept.
Washington University	Princeton University
St. Louis, MO	Princeton, NJ

Abstract - This paper presents the results of a multi-level cache memory hierarchy evaluation for programmable media processors. With the continuing advances in VLSI technology, it becomes possible to support larger memory hierarchies on-chip, but the question remains of how to most effectively use these additional silicon resources for optimizing memory performance. This paper explores that issue by evaluating the various levels of the memory hierarchy using a cache-based memory system. This evaluation examines the change in performance from varying cache parameters including the L2 cache parameters of cache size, line size, and latency, and the external memory parameters of bandwidth and latency. Examining the performance impact of these parameters, we have identified external memory latency and bandwidth as the primary memory bottlenecks in media processors.

INTRODUCTION

With the success of the Internet and World Wide Web, and the growing feasibility of image/video compression and computer graphics, the multimedia industry has been growing at a tremendous rate. Multimedia now defines a significant portion of the computing market, and this is expected to grow considerably. As a consequence, the processing demands for multimedia applications are rapidly escalating as users desire new and better applications. Many multimedia applications are already beyond the limits of today's microprocessors, and the next generation of multimedia promises a wider range of applications and considerably greater processing demands. Adequate support for future multimedia will require much greater flexibility and computing power than currently available. A likely solution to this problem is the use of single-chip *programmable media processors*, processors specifically designed to provide efficient media processing across the full range of multimedia applications.

The current methods of support for multimedia include: a) application-specific hardware, b) multimedia extensions to general-purpose processors, and c) media processors [1][2]. Application-specific hardware provides excellent performance at low cost, but has limited flexibility and is unable to support evolving and future applications. General-purpose processors with multimedia extensions provide the necessary flexibility, but are designed primarily for general-purpose applications and cannot adequately support the distinctive multimedia workloads. Media processors have proven themselves to be effective for supporting the current generation of multimedia, but the existing media processors achieve only a fraction of their potential performance. They provide performance primarily through specialized hardware, and so only enable high throughput for select multimedia functions. This is sufficient for current multimedia, where media is regularly represented as images, sequences of video frames, or audio channels, but will likely not be sufficient for future multimedia.

The next generation of multimedia promises to be more integrated and interactive, as evidenced by new applications such as video libraries [3], MPEG-4, MPEG-7, and MPEG-21. These applications use new algorithms that define multimedia using advanced representations. MPEG-4, for example, represents multimedia in terms of objects, each with its own audio, visual, and graphical characteristics. These emerging applications enable higher compression rates, content-based processing, and more interactive multimedia. They also introduce much greater processing demands and irregularity than previous generations of multimedia.

Support for future multimedia therefore requires media processors that are significantly different from those available today. To meet the high throughput and flexibility demands of the next generation of multimedia, we expect future media processors will differ from existing media processors in three key areas [4]. First, processors will have many more functional units for supporting high degrees of parallelism. Second, future media processors will have more regular architectures and less specialized hardware for better high-level language (HLL) programmability and greater flexibility over a wide range of applications. And finally, they will have larger on-chip memory hierarchies to accommodate the high data rates and minimize large external memory latency penalties. However, to realize these processors, there remain many open design issues.

One of the many open issues is the question of how the memory hierarchy will be structured for maximum performance. We shall begin exploring that question in this paper by examining various aspects of memory hierarchies for multimedia. This study uses a well-understood cache memory hierarchy paradigm as the basis for examining multimedia memory performance. Performing regressions across various parameters in the memory hierarchy, we use the results to identify the bottlenecks of memory performance in multimedia applications.

This paper continues in the next section with a discussion of related research. The third and fourth sections present the evaluation environment and basic architecture model. The fifth section describes the experiments and identifies the primary memory bottlenecks. The final sections discuss conclusions and future work.

RELATED WORK

The memory hierarchy of programmable media processors remains an ill-defined area. Whereas general-purpose processors use multi-level cache memory hierarchies, and typical DSPs use local memory with some form of prefetching such as DMA, the memory hierarchy for media processors remains an unresolved issue. An initial investigation of cache memory performance for media processing was performed by Lee et al. with their introduction of MediaBench [10]. However, this study only examined first level (L1) cache performance. Studies by Wu and Wolf on video application traces [5] have found that hybrid memory architectures with stream buffers provide better performance than cache-only memory hierarchies. However, these studies are based on trace-driven simulations assuming perfect branch prediction and memory disambiguation. Zucker et al. [6][7] also examined the impact of streaming memory structures such as stream buffers, stride prediction tables, and stream caches, and also found considerable benefit with these structures in multimedia applications. Consequently, we expect some hybrid of cache and prefetching will provide the best performance.

In addition to providing support for streaming memory, the large data rates in multimedia can place significant burdens on external memory, so it often becomes a bottleneck. To support the high data rates, either memory hierarchies must be designed to reduce external memory traffic, or scheduling methods must be used to restructure the program for improved data locality [8][9].

The memory hierarchy for media processors is poorly understood. Although streaming memory structures will likely be necessary, it is unknown whether cache-only, memory with prefetching, cache with prefetching, or other novel memory structures will prove most suitable. Considerable research remains on memory hierarchy design for media processors.

EVALUATION ENVIRONMENT

Accurate evaluation of a cache memory hierarchy requires a benchmark suite representative of the multimedia industry, an aggressive compiler, and a parameterizable simulator capable of analyzing a wide variety of architectural features. This cache hierarchy evaluation uses an augmented version of the MediaBench benchmark and the IMPACT compilation/simulation tools.

The MediaBench benchmark, defined by Lee et al. [10], is the first combination of applications designed to represent the full range of the multimedia industry. The benchmark focuses on portable applications written in a high-level language (HLL) that are representative of emerging multimedia and communications systems. It includes applications from image and video processing, audio and speech processing, and encryption and computer graphics. We augmented MediaBench with two additional video applications, H.263 and MPEG-4, to make it more representative of emerging multimedia applications. More information on MediaBench may be found in Lee et al. [10][11], and Fritts et al. [12][2].

The compilation and simulation tools for this architecture style evaluation were provided by the IMPACT compiler, produced by Wen-mei Hwu's group at the University of Illinois at Urbana-Champaign [13]. The IMPACT compiler is an aggressive profile-based ILP compiler. In addition to the traditional local optimizations, the IMPACT compiler offers many global scheduling methods such as procedure inlining, loop unrolling, speculation, and predication (conditional execution). IMPACT also provides a parameterizable simulator that enables simulation of two base processor models, an in-order superscalar architecture, and a VLIW/EPIC architecture. This study uses aggressive compiling on the VLIW/EPIC architecture for evaluating memory performance in multimedia.

BASE ARCHITECTURE MODEL

Systematic evaluation of different architectures requires a base processor model to which all other processor models can compare their performance. The base processor model defined here is an 8-issue VLIW media processor targeting the frequency range from 500 MHz to 1 GHz. The processor has separate L1 instruction and data caches, a unified on-chip L2 cache, and an external memory bus that operates at ¼ the frequency of the processor. The L1 instruction cache is a 16 KB direct-mapped cache with 256-byte lines and a 20 cycle miss latency (assuming a hit in L2). The L1 data cache is a 32 KB direct-mapped cache with 64-byte lines and a 15 cycle miss latency. It is non-blocking with an 8-entry miss buffer, and uses a no-write-allocate policy with an 8-entry write buffer. The L2 data cache is a 256 KB 4-way set associate cache with 64-byte lines and a 50 cycle miss latency. It is non-blocking with an 8-entry miss buffer, and uses a write-allocate policy with an 8-entry write buffer. Further details are provided in [2].

In addition to the 8-issue VLIW, experiments were run for 4-issue and 8-issue in-order superscalar processors, out-of-order superscalar processors, and VLIW processors. Overall, the results were very similar for all processors. The primary difference is that the out-of-order superscalar processor is most heavily impacted by variations in the memory hierarchy, as expected.

CACHE MEMORY HIERARCHY EVALUATION

This section examines the various levels of the memory hierarchy, including the first level (L1) on-chip cache, second level (L2) on-chip, and external memory. Additional information on this cache hierarchy study may be found in Fritts [2].

L1 Cache

Our initial studies on memory characteristics of multimedia applications [12] revealed that multimedia applications typically have working set sizes of no more than 32 KB for data memory and 8 KB for instruction memory. However, because aggressive compiler optimizations increase code size by 50-100%, a 16 KB instruction cache is needed. Using these working set sizes to select 32 KB and 16 KB direct-mapped caches for data and instruction memory, respectively, our earlier architecture evaluation found they provide excellent L1 cache performance and represent good choices for first level memory in media processors [2].

The primary issue of importance remaining for first level memory hierarchy design is whether memory prefetching structures such as stream buffers or stride prediction tables should be incorporated at this level, and if so, what their configurations should be. While we have not made a direct evaluation of these memory structures yet, considerable insight will be found with regards to memory prefetching in the subsequent sections.

L2 Cache

Conventional wisdom in processor design has been that providing more memory on-chip enables better cache performance and reduces the impact of long external memory latencies. It is questionable whether this will be true in multimedia, however. Multimedia is prone to streaming data that is accessed for short periods of time, and then thrown away and never needed again. In such cases, additional on-chip memory may not be of benefit. Experiments with the L2 cache that vary cache size, line size, and access latency were performed to more thoroughly understand multimedia's use of the second level cache.

Cache Size. The first experiment on the L2 cache involved evaluating the performance from increasing cache size. In this experiment, L2 cache size was varied over the range from 128 KB to 1 MB. We had believed that increasing the amount of on-chip memory would be less useful to media processing than general-purpose processing, but even so the results are quite surprising. There is almost no impact on performance when varying L2 cache size. The average performance increase from doubling cache size is only 0.5%. Only two benchmarks, *unepic* and *mpeg4dec*, had notable differences in performance (about 7%).

While it is obvious that L2 cache size is not important from the standpoint of any single multimedia benchmark, it will become much more important when simultaneously executing many multimedia applications on the same processor. In such cases, larger L2 cache sizes are needed to maintain the memory image for multimedia applications during context switches. However, with a base size of 256 KB, the L2 cache is already able support the memory image for at least 8 typical multimedia applications since most working set sizes are less than 32 KB.

Access Latency. A second experiment was performed to determine the impact of increasing L2 access latency (latency when L1 misses and L2 hits). In this experiment, L2 access latency was varied between 8 cycles and 60 cycles. The results indicate L2 cache access latency has only a moderate effect on memory performance. Overall the average performance degradation from doubling L2 cache access latency was only 5.6%. Only three benchmarks were significantly impacted: *pegwitenc*, *pegwitdec*, and *mpeg2dec*. The two *Pegwit* security applications had performance degradations of 35% because the data working set does not fit in the L1 data cache, while *mpeg2dec* has a degradation of about 16%.

Line Size. The final experiment with respect to the L2 cache evaluates cache line size. The line size was varied from 32 to 512 bytes, as shown below in Figure 5.1.

Unlike L2 cache size and access latency, there is a much more noticeable effect from increasing L2 cache line size. Overall, the average performance decrease is 10% from doubling line size, but this varies by media type. As shown, speech and security media are only affected minimally with degradations of 1.5-3.5%, and video has a degradation of 12.8%, while audio, graphics, and image media types are heavily affected with degradations in the range of 32-57%.

The reason for the significant change in performance is due to increased latency when accessing longer lines from external memory. As seen in the next section, many multimedia benchmarks are highly dependent upon memory, so varying memory latency can change their performance significantly. In this experiment the effect is somewhat reduced because of spatial locality. With longer line sizes, spatial locality enables processing to make use of much of the additional data, but the extra access time has greater impact than the benefit from spatial locality. Also, spatial locality decreases considerably with line sizes above 64 bytes [2].

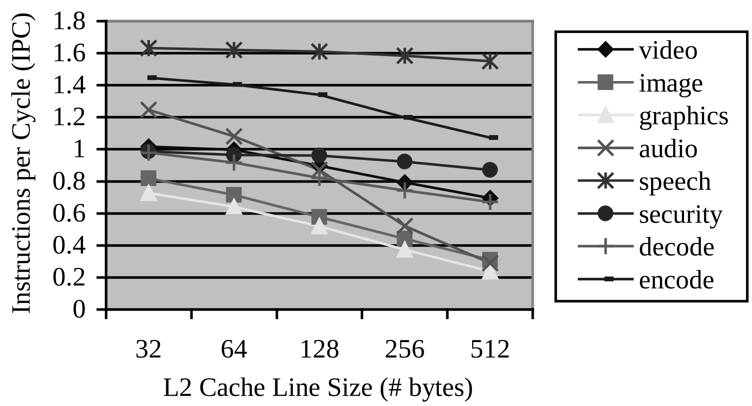

Figure 1 – IPC performance from varying the width of the L2 cache line.

Overall, it has been shown that media processing is minimally affected by L2 cache parameters. Cache size and access latency have little effect on performance, although larger L2 caches are important when simultaneously supporting multiple multimedia applications. Line size can have considerable impact, but this is primarily due to the increased memory latency. Consequently, this is not really a problem with the L2 cache, but more a problem of long external memory latencies.

External Memory

As found in the previous section, memory latency is a significant problem for media processing. This problem can stem from one of two sources, however. Either the problem occurs naturally from long latencies to external memory, or the applications are in fact limited by external bandwidth, which may indirectly increase memory latency by stalling memory accesses. Two experiments were conducted to examine variations in external memory latency and bandwidth.

To evaluate the impact of external memory latency on performance, the first experiment performed a memory latency regression that varied memory latencies over the range from 25 to 400 cycles. The performance results are shown in Figure 5.2. Overall, the results indicate considerable performance degradation, 20% for each doubling of latency. However, this varies considerably among the various media types. From the figure it is easy to discern that speech and security experienced little degradation from external memory latencies, while audio, image, and graphics were severely impacted, with degradations between 59-77%.

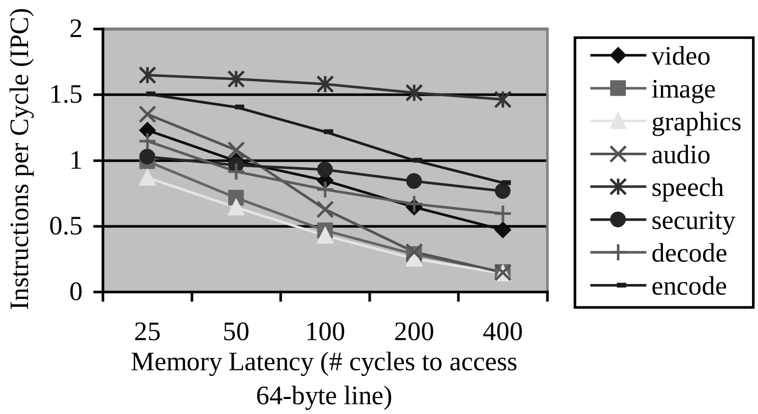

Figure 2 - IPC performance from varying latency to access 64-byte line from external memory (on L2 cache miss).

The second experiment evaluated the effect of external memory bandwidth by varying the size of the external memory bus from 4 to 32 bytes. The results from this experiment are shown in Figure 5.3. The average performance increase from doubling the size of the memory bus is only 6.0%. Again however, there is considerable variation of the results with respect to media type. The speech, security, and encoding benchmarks have a very low degradation of 0.6-2.7%, while the other benchmarks are much higher with averages of 7.5-13.9%.

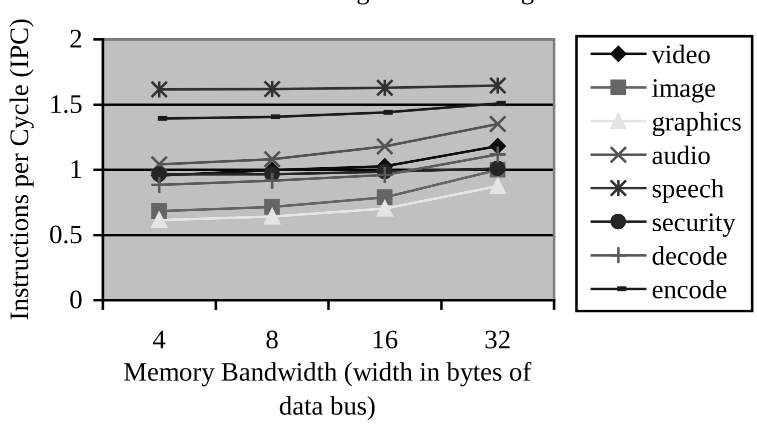

Figure 3 - IPC performance from varying the width of the external data bus.

Comparison of results from the two experiments seems to indicate that memory latency is the more prominent bottleneck. However, these two parameters are not mutually exclusive from one another. When memory bandwidth increases without a corresponding increase in latency, the act of fetching the extra data for free is effectively prefetching. So when spatial locality exists, the prefetching helps reduce the effects of memory latency. Likewise, when memory latency increases without a corresponding increase in bus width, there is a decrease in memory bandwidth. Only by evaluating the effects of memory bandwidth in concert with memory latency can the actual cause of the problem be determined. Simultaneous evaluation of memory latency and bandwidth is shown below in Table 5.1. This table displays the average degradations for memory latency and memory bandwidth, and identifies the degree of utilization of the external memory bus. This is defined in the "Bandwidth" column. Benchmarks with low bus utilization in the range of 0-32% are marked low (L) utilization. Similarly, the medium (M) and high (H) utilization ranges correspond with 33-66% and 67-100% utilization.

Program	*Avg. Latency Degradation (%)*	*Avg. Bandwidth Degradation (%)*	*Bandwidth (L, M, H)*
cjpeg	68.1	11.3	M
gs	66.8	15.4	M
gsmencode	3.6	0.4	L
H263dec	99.1	30.8	H
mipmap	75.6	13.1	H
mpeg2enc	25.3	2.8	L
mpeg4dec	95.3	27.8	H
pegwitdec	25.1	3.0	L
rawdaudio	108.1	22.3	H
texgen	53.3	6.1	M
unepic	88.1	21.5	H

Table 1 – Memory characteristics of select benchmarks from MediaBench.

The results in the table indicate that all benchmarks are 3-5 times more susceptible to changes in memory latency than memory bandwidth. While the cross correlation that exists between memory latency and memory bandwidth would decrease the true memory latency degradation and increase the true memory bandwidth degradation, that correlation would not account for the 3-5x difference. Furthermore, the cross correlation would exist primarily in those applications with high external memory bandwidths, which include the *djpeg*, *h263dec*, *mipmap*, *mpeg4dec*, *osdemo*, *rawcaudio*, *rawdaudio*, and *unepic* benchmarks. Of these eight benchmarks, we expect that memory latency is the primary limiting factor, although *djpeg*, *h263dec*, *mpeg4dec*, *rawdaudio*, and *unepic* also experience a memory bandwidth limitation to a lesser degree. Consequently, memory latency is the dominant memory bottleneck in media processing.

CONCLUSIONS

This paper explored the issues in memory hierarchy design for programmable media processors by evaluating the performance of a multi-level cache memory hierarchy. Prior work focused on the L1 cache level [5][6][10][12], so particular emphasis was placed on the L2 cache and external memory. Surprisingly, it was found that the second level cache had little impact on memory performance. L2 cache size and access latency have only nominal effects on performance. The L2 cache will be important, however, for holding the memory image during context switches when simultaneously executing multiple multimedia applications.

The only L2 cache parameter that had a noticeable impact on performance was L2 cache size, and this can be attributed to the negative impact of longer cache lines on memory latency. Further justification of multimedia's dependence on memory latency was found when examining the effects of external memory latency and external memory bandwidth. Overall, we conclude that external memory latency presents the greatest problem to media processing. The only other significant bottleneck in media processor memory hierarchies is external memory bandwidth.

These two problems, while significant now, will continue to become even more critical in future media processing. As the multimedia industry continues to evolve, future applications will require the processing of even more data per unit time. Another area of concern is the ever-widening processor-memory gap. As processor performance continues to increase faster than memory performance, external memory latency and bandwidth become even more limited, relatively. Consequently, the problems of memory latency and memory bandwidth will be critical bottlenecks in future media processors. The success of media processors will require effective solutions to these two problems.

FUTURE WORK

Fortunately, the memory latency bottleneck, while being the most critical problem, is also the problem most amenable to a solution. As noted in the related work, prefetching using streaming memory structures has been identified as an effective method for mitigating the external memory latency problem by prefetching the data before it is known whether the data is actually needed. However, because external memory bandwidth is also a problem, it is necessary to avoid using overly aggressive prefetching, which might overload the bandwidth. Future research will examine multi-level prefetch hierarchies, which use conservative prefetching on-chip and provide more aggressive prefetching off-chip.

ACKNOWLEDGEMENTS

This work was funded by the New Jersey Center for Multimedia Research and by the National Science Foundation. We would also like to thank Lee et al. for developing MediaBench [10], and IMPACT for use of their compiler [13].

BIBLIOGRAPHY

[1] Zhao Wu and Wayne Wolf, "Parallel Architectures for Programmable Video Signal Processing," Technical Report, Princeton University, 1998.

[2] J. Fritts, "Architecture and Compiler Design Issues in Programmable Media Processors," Ph.D. Thesis, Dept. of Electrical Engr., Princeton Univ., 2000.

[3] Wayne Wolf, Yiqing Liang, Michael Kozuch, Heather Yu, Michael Phillips, Marcel Weekes, and Andrew Debruyne, "A digital video library on the World Wide Web," *Proceedings of ACM Multimedia '96*, ACM Press, 1996.

[4] J. Fritts, Z. Wu, and W. Wolf, "Parallel Media Processors for the Billion-Transistor Era," *Intl. Conf. on Parallel Processing*, Aizu, Japan, Sept. 1999.

[5] Z. Wu and W. Wolf, "Study of Cache Systems in Video Signal Processors," *IEEE Workshop on Signal Processing Systems*, Oct. 1998, pp. 23-32.

[6] D. F. Zucker, "Architecture and Arithmetic for Multimedia Enhanced Processors," Ph.D. Thesis, Dept. of Electrical Engineering, Stanford University, June 1997.

[7] D. Zucker, M. Flynn, and R. Lee, "A comparison of hardware prefetching techniques for multimedia benchmarks", Technical Report CSL-TR-95-683, Stanford University, Dec. 1995.

[8] Y. K. Chen and S. Y. Kung, "Multimedia Signal Processors: An Architectural Platform with Algorithmic Compilation," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 20, 1998.

[9] F. Catthoor, S. Wuytack, E. DeGreet, F. Balasa, L. Nachtergaele, and A. Vandecapelle, "Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design," Kluwer Academic Publishers, 1998.

[10] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: A Tool for Evaluating and Synthesizing Multimedia Communications Systems," *30th Annual International Symposium on Microarchitecture*, De. 1997.

[11] MediaBench home: http://www.cs.ucla.edu/~leec/mediabench/

[12] J. Fritts, W. Wolf, and B. Liu, "Understanding multimedia application characteristics for designing programmable media processors," *SPIE Photonics West, Media Processors '99*, San Jose, CA, Jan. 1999, pp. 2-13.

[13] IMPACT compiler group: http://www.crhc.uiuc.edu/Impact