Improving the Memory Bandwidth of Highly-Integrated, Wide-Issue, Microprocessor-Based Systems

David H. Albonesi

Dept. of Electrical Engineering
University of Rochester
Rochester, NY 14627
albonesi@ee.rochester.edu

Israel Koren

Dept. of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
koren@ecs.umass.edu

Abstract

Next generation, wide-issue processors will require greater memory bandwidth than provided by present memory hierarchy designs. We propose techniques for increasing the memory bandwidth of multi-ported L1 Dcaches, large on-chip L2 caches, and dedicated memory ports while minimizing cycle time impact. These approaches are evaluated within the context of an 8-way superscalar processor design and next-generation VLSI, packaging, and RAM technologies. We show that the combined L1 and L2 cache enhancements can outperform conventional techniques by over 80%, and that even with an on-chip 512KB L2 cache, board-level caches provide significant enough performance gains to justify their higher cost.

1 Introduction

Increasing attention has been focused on the memory hierarchy performance of microprocessor-based systems due to the growing processor/memory speed gap and memory bandwidth demands of modern superscalar designs. In order to prevent the next-generation of wide-issue, highly-integrated microprocessor-based systems from becoming memory performance-bound, improvements to current memory hierarchy designs must be found. In this paper, we address ways to increase the memory bandwidth of the L1 Dcache, L2 cache, and off-chip interface.

High bandwidth in L1 cache design is usually achieved through a combination of pipelining and multi-porting. However, pipeline depth is limited by load-use delay considerations, and adding ports may increase cycle time. The Alpha 21164 microprocessor [4] uses two duplicate copies of the L1 Dcache to produce two read ports without impacting cycle time. However, stores are limited to one per cycle because of the need to update both copies. An increase in issue width may necessitate performing more than one store instruction per cycle. Sohi and Franklin [18] explore increasing L1 Dcache bandwidth through multiple interleaved, non-blocking caches. The drawback of this approach is the latency of the crossbar interconnect between the processor and cache banks. Wilson [20] explores conventional multi-ported and pipelined L1 Dcaches for a 4-way processor, while we explore additional alternatives necessary for the complete hierarchy of a wider-issue design. The Alpha 21264 microprocessor [11] achieves dual-ported operation by performing two L1 Dcache accesses within a single cycle, resulting in an extra cycle of load-use delay relative to the 21164. With wider issue processors, increasing load-use delay may limit the amount of parallelism that can be achieved due to data dependencies.

Increases in VLSI integration present the opportunity to implement significantly larger pipelined on-chip L2 caches than possible today, but with the challenge of providing high bandwidth L2 cache access without a cycle time penalty. With a given number of pipeline stages, the cycle time in general increases with increasing cache size. Attempting to compensate for this increase by simply adding more stages increases the L2 cache latency. Eventually, adding more stages has no cycle time benefit once individual elements that cannot be further pipelined, such as the data array or the data output selector drivers, dominate the overall cache delay. Peir [17] proposes using a *path balancing table* (PBT) to improve cache access time. The table stores the most recent cache tag to data array index and way select mappings in order to decouple the tag check and data access paths. For cases that hit in the PBT, their technique significantly reduces cache access time for non-pipelined caches. As we discuss in Section 4, additional techniques are required to reduce the cycle time of large on-chip L2 caches.

Package pin limitations constrain off-chip memory bandwidth. Most recent microprocessors provide a dedicated port for a board-level cache, while the main memory system control logic is implemented on a separate ASIC attached to the external system bus. A large enough on-chip L2 cache may preclude the presence of a board level cache, allowing this external port to be dedicated to a direct connection to main memory SIMM modules. This has the potential of significantly reducing main memory access time (and increasing memory bandwidth), and has the added benefit of enabling the construction of dense, highly-integrated, multiprocessor building blocks.

In this paper, we explore methods for increasing the memory bandwidth of L1 Dcaches, L2 caches, and off-chip accesses for microprocessors with greater integration levels and issue widths than current designs. In the next section, we describe the overall system design. After a presentation of our methodology in Section 3, we discuss design alternatives in Section 4. In Section 5, we quantify the performance benefits of the memory bandwidth enhancements, and conclude in Section 6.

2 System Description

Our simulated design is an extension of the Alpha 21164 microprocessor [4] enhanced to provide 8-way issue. In-order issue processor designs can potentially achieve a higher clock frequency than out-of-order designs because of their simpler control logic and internal data paths [9]. In addition, the relative simplicity of an in-order design can be crucial in reducing the design time of a wide-issue processor [8]. However, in-order designs rely more heavily on advanced compiler technology, like the Multiflow Trace Scheduling Compiler [12] used in this analysis, to extract parallelism from the instruction stream.

An Instruction Fetch Unit (IFU) fetches eight 32-bit instructions from the L1 Icache and issues them to three types of functional units: Floating-Point Add Units (FPAUs), Floating-Point Multiply Units (FPMUs), and Integer Units (IUs). These units perform the same operations as comparable units in the 21164.

A two-level on-chip cache hierarchy is arranged as split L1 caches backed by a unified L2 cache. Table 1 lists values for some of the cache hierarchy parameters. The L1 Icache is backed by a prefetching stream buffer [10], while the L1 Dcache and L2 cache are each backed by a Memory Merge Buffer (MMB). The MMB is similar in functionality to the 21164's Miss Address File [4] and provides non-blocking support and merging of misses to the same cache block. The internal datapath

Element	Parameter	Value
L1 Icache	block size	64 bytes
L1 Dcache	block size	64 bytes
L1 Dcache	write policy	write through
L1 Dcache	Write Buffer entries	12
L1 Dcache	MMB entries	16
L2 Cache	block size	128 bytes
L2 Cache	write policy	write back
L2 Cache	MMB entries	8
L2 Cache	Victim Buffer entries	1

Table 1: Cache Hierarchy Parameters and Values.

Parameter	Value
die size	$350mm^2$
minimum feature size	0.3 microns
package pins	975
SRAM organization [14]	128K×8
SRAM access time	5ns
SIMM organization [13]	8M×36
SIMM access time	40ns

Table 2: Technology Parameters and Values.

widths are double that in the 21164 to meet the higher instruction and data bandwidth requirements of an 8-way superscalar design. External requests are serviced by the Bus Interface Unit (BIU) over a 256-bit wide system bus.

Table 2 summarizes the technology parameters used in the analysis, based on trends [6, 7] in custom CMOS microprocessor implementations, packaging technologies, and board-level components. The microprocessor and ASICs used for memory and I/O control directly drive external buses similar to several current microprocessors [16, 22] and the ASIC designed for the AlphaServer 8000 [2].

3 Methodology

Our evaluation methodology combines the use of the System Tradeoff Analysis Toolset (STATS) [1] for performance analysis with area and package pin count calculations. We use the die photograph of the Alpha 21164 [3] to determine relative sizes of arithmetic units and caches. We then use an enhanced version of Mulder's cache area model [15] to model the 21164's 8KB Icache as a baseline, and use this to determine relative cache sizes for the 8-way superscalar design. The modified model includes area estimation of multi-ported cache structures using multi-ported cell design and/or duplication of arrays, and takes into account the area impact of including parity bits (for both data and tag arrays), and dividing each array into subarrays [19, 21].

STATS integrates detailed timing analysis, benchmark compilation targetted to the architectural organizations, and execution-driven performance simulation into an automated analysis framework. We have expanded CACTI [21] to include pipelining and multiporting. The toolset finds the optimal pipeline register placement between the major substages of the cache for the desired number of stages, and scales the size of the cell, word and bitline capacitances, and cell I/O capacitances according to the number of cache ports desired. Board-level cache, system bus, and main memory delays are calculated by an automated Spice-based tool using electrical and physical parameters such as spacing between components and VLSI and SRAM packaging and electrical specifications.

Benchmarks are compiled using a modified version of the Multiflow Trace Scheduling compiler [12] targetted to the Digital Alpha architecture. Code generation is influenced by the microarchitectural machine model which includes the number and types of functional units, instruction latencies, and simultaneous load and store cache access restrictions. Because instructions are grouped by the compiler according to this specification, individually optimized benchmark executables are generated for each machine configuration. Improved simulation accuracy is achieved by incorporating the detailed timing overheads calculated by the timing tools into the simulation.

4 Design Alternatives

In this section, we determine design alternatives for the L1 Dcache, L2 cache, and external memory interface considering the die and pin count constraints described in Section 2. We first allocate as much die area as possible to an on-chip set-associative L2 cache, while leaving room for enough functional units and reasonable size L1 caches. We then consider options for the size of the L1 Dcache, while keeping the overall die usage roughly equivalent. This is accomplished by allocating die area to other functions for smaller L1 Dcache configurations. Next, we perform a pin-allocation analysis to determine the number of pins that can be dedicated to the off-chip memory interface, and use component specifications to determine viable configurations. We focus on load/store bandwidth; consideration of improved instruction fetching hardware is beyond the scope of this paper[1].

[1]Because instruction cache miss ratios for our benchmarks are very low (less than 1%), even with an 8-way superscalar design, our performance results are not greatly impacted by the lack of more elaborate instruction fetch hardware.

Element	Area
8KB ICache	0.65%
16KB Dcache (1TP)	3.8%
16KB Dcache (2DP)	5.3%
16KB Dcache (4SP)	6.3%
32KB Dcache (1TP)	6.9%
32KB Dcache (2DP)	9.8%
32KB Dcache (4SP)	11.5%
FPAU	1.3%
FPMU	1.3%
FP Reg File	0.5%
IU	0.8%
Int Reg File	0.5%
512KB L2 Cache	53.8%
Processor Remainder	25.9%

Table 3: Area Costs of Chip Functionality.

12 AU Configuration	8 AU Configuration
8KB ICache	8KB ICache
16KB Dcache	32KB Dcache
3 FPAUs	2 FPAUs
3 FPMUs	2 FPMUs
6 IUs	4 IUs
512KB L2 Cache	512KB L2 Cache

Table 4: Two Area-Equivalent Configurations Varying in L1 Dcache Size.

4.1 Area Allocation Analysis

Table 3 shows approximate area costs of various chip functions expressed as a percentage of the total chip area (excluding the pad area). A 4-way set-associative 512KB L2 cache is the largest that can fit on the die and leave enough room for the processor. The L1 Dcache area estimates are for three multiporting options: a single triple-ported array (1TP), two dual-ported arrays (2DP), and four single-ported arrays (4SP). We use area estimates for the 2DP option only in determining design possibilities, since we show later that this is the best design choice. The last item includes the IFU, BIU, clock and bus distribution, and memory buffers and control logic.

Using these area calculations, we determine two area-equivalent configurations which vary in L1 Dcache size (Table 4). For the smaller L1 Dcache configuration, we use the remaining area for additional arithmetic units (AUs) in order to keep the total die area roughly equivalent. This creates a higher memory bandwidth requirement for the configuration with the 16KB L1 Dcache, as the presence of more arithmetic units increases the rate at which instructions are consumed by the processor.

4.2 Package Pin Allocation Analysis

With a 975 pin package, we have 650 I/O pins available, after allocating a power or ground signal for every two I/O signals. The external system bus requires 288 pins including parity bits, plus approximately 20 signals for arbitration and acknowledgement in a shared-memory multiprocessor environment. Subtracting an additional 10 signals for miscellaneous functions, we allocate the remaining 332 pins for a dedicated port to the next level in the memory hierarchy, which reduces external loading and contention for microprocessor resources [22]. In Section 4.5, we consider options for the use of this external port. First, we consider the upper levels of the memory hierarchy.

4.3 L1 Dcache Multi-Porting Options

Figure 1 shows three options we consider for a multi-ported L1 Dcache design. Figure 1(a) shows a triple-ported L1 Dcache (1TP), while Figure 1(b) shows an extension of the approach used in the Alpha 21164 microprocessor [4]: using multiple single-ported cache arrays. In this case, we show a cache with four single-ported arrays (4SP).

The 4SP cache has several advantages over the 1TP design. Its cycle time is lower, design simpler, and routing less dense. Furthermore, if the L1 I and Dcaches are the same size, the design and verification of the Icache can be avoided by using one of the single-ported Dcache arrays. However, all four ports are consumed on a store operation in order to update each of the arrays. In addition, the 4SP cache requires more area than the 1TP design (Table 3).

Figure 1(c) shows a design which attempts to mitigate the disadvantages of the previous approaches by employing duplicate dual-ported caches (2DP). Here, up to four loads, two stores, or two loads and one store can be performed in the same cycle. The 2DP cache design is less complex and has a lower cycle time than the 1TP design, yet requires less area and matches port-use to application load/store requirements better than the 4SP cache.

4.4 L2 Cache Pipelining Options

Unlike the L1 Dcache, multi-porting the L2 cache is not a practical approach to increasing bandwidth because of the significant area overhead that would be incurred with multi-porting such a large cache. A pipelined L2 cache can achieve high bandwidth, but as was noted in Section 1, a cycle time penalty may be incurred with large pipelined caches.

With a wave-pipelined L2 cache, the cache controller launches the address to the cache RAMs and then waits a fixed number of cycles before capturing the data. The sending of the address of one cache access overlaps the data retrieval of the previous cache access. (We assume in our analysis a one cycle overlap in all cases.) Although no cycle time penalty is incurred for large L2 caches, the downside of this approach is its lower bandwidth, as is shown in Section 5.

To achieve high L2 cache bandwidth with minimal cycle time impact, we consider two cache enhancements: *sectioning* and *multi-cycle pipelining* (MC pipelining). With sectioning, the cache is sliced into two or more smaller caches, each of which handles a fraction of the data output bits and has its own local copy of the tag and status bits. This differs from a cache using multiple *subarrays* [19, 21] where only the arrays are sliced (although separate decoders are used for each slice) in order to reduce bit and wordline delays. However, in a set-associative cache, the data output selector driver delay is often the largest component of the overall delay [21]. This delay becomes more severe as L2 cache data bus widths are increased to accommodate higher bandwidth requirements. Sectioning the cache reduces this delay by duplicating the selector drivers and reducing the number of multiplexors driven, creating a lower fanout requirement. In addition, because each data section has a local tag array, worst case metal lengths between the two are reduced relative to a single set of tags.

Multi-cycle pipelining involves operating the L2 cache at a fraction, in our case one-half, of the rate of the L1 cache, analogous to wait states in main memory systems. This allows each L2 cache stage two processor clock cycles to complete, so as not to impact overall cycle time for caches whose delay is dominated by one slow stage. In addition, higher bandwidth than the wave-pipelined cache can be achieved. Thus, MC pipelining can potentially create a better balance between bandwidth and cycle time than either pipelined or wave-pipelined large on-chip L2 caches.

4.5 Dedicated Memory Port Options

We consider two options for use of the dedicated memory port package pins: as an interface to an off-chip 4MB L3 cache, which requires that the Main Memory Controller (MMC) be located on an ASIC attached to the system bus; or as an interface to memory SIMM modules through an on-chip MMC; in this case a two-level cache hierarchy is implemented. While the L3 cache approach provides a larger cache at the lowest level of the hierarchy, the on-chip MMC design reduces main memory latency. In addition, integrating the MMC onto the microprocessor die allows it to operate off the microprocessor clock rather than the

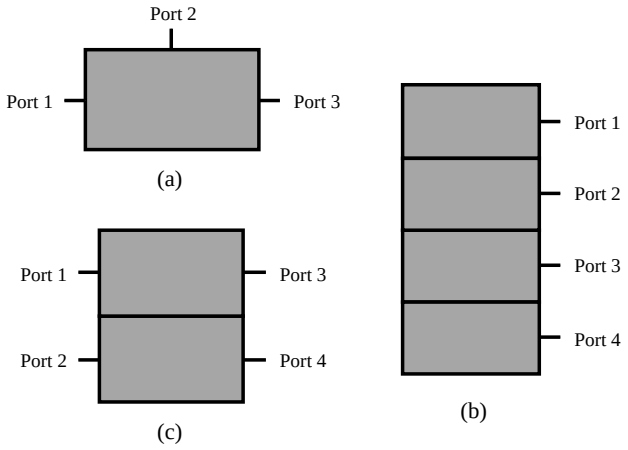

Figure 1: L1 Dcache Multi-Porting Options: (a) 1TP, (b) 4SP, (c) 2DP.

Porting Option	Cycle Time	
	16KB	32KB
1TP	1.97	2.28
2DP	1.92	2.21
4SP	1.86	2.16

Table 5: L1 Dcache Cycle Time for Different Multi-porting Options.

divided-down system-level clock, allowing faster state machine transitions and finer grain control of memory timing signals. This reduction in latency increases bandwidth as data can be transferred out of the main memory at a faster rate.

5 Evaluation of Alternatives

In this section, we compare the memory hierarchy design alternatives described in the previous section using seven of the SPEC92 benchmarks: three of which are integer benchmarks (*li*, *eqntott*, and *espresso*), and four from the floating point suite (*spice2g6*, *swm256*, *su2cor*, and *tomcatv*). Gee [5] has pointed out the limitations of using SPEC92 to evaluate cache behavior. However, we have chosen several of the floating point benchmarks, which are more indicative of the cache behavior of user programs [5].

5.1 L1 Dcache Multi-Porting Options

To evaluate L1 Dcache multi-porting alternatives, we compare cycle time, average instructions per issue group (which measures how well the compiler is able to schedule instructions considering the port constraints), and finally, elapsed processor time. Cycle time results for the various options are shown in Table 5. As expected, the 1TP configuration has the largest cycle time due to its use of triple-ported cells. In general, each cache port increases cycle time by 3-4%, which is small relative to the 15-16% cycle time increase incurred in doubling the cache size. Thus, adding a small number of ports may provide a performance advantage if a considerable increase in the size of the average instruction issue group can be obtained.

Figure 2 plots average instructions per issue group for the three multi-porting options for both the 8 and 12 AU configurations. In all cases, the 2DP cache achieves the highest instructions per issue group ratings, while the next best configuration varies from benchmark to benchmark. The benchmarks with the highest percentage of store instructions are *li* and *eqntott*. Although *li*'s performance greatly improves with the 2DP relative to the 4SP design, *eqntott* improves much less than several of the other benchmarks which have fewer stores. This is because *eqntott* has about half the percentage of load instructions than the other workloads, which allows the compiler to schedule around the port limitations of the 4SP cache.

The behavior of the three multi-porting schemes is particularly interesting for *tomcatv* as the number of arithmetic units is increased from 8 to 12. While the average instructions per issue group increases for both the 1TP and 2DP options, no change is observed for the 4SP design. The single store per cycle limitation of this scheme creates scheduling constraints that bound the obtainable speedup.

Figure 3 plots processor elapsed time for the 4SP and 2DP caches, normalized to that of the 1TP cache organization for each of the benchmarks. (Figure 3(a) is normalized to the 8 AU configuration, and Figure 3(b) to the 12 AU configuration.) Both the 2DP and 4SP configurations execute all benchmarks faster

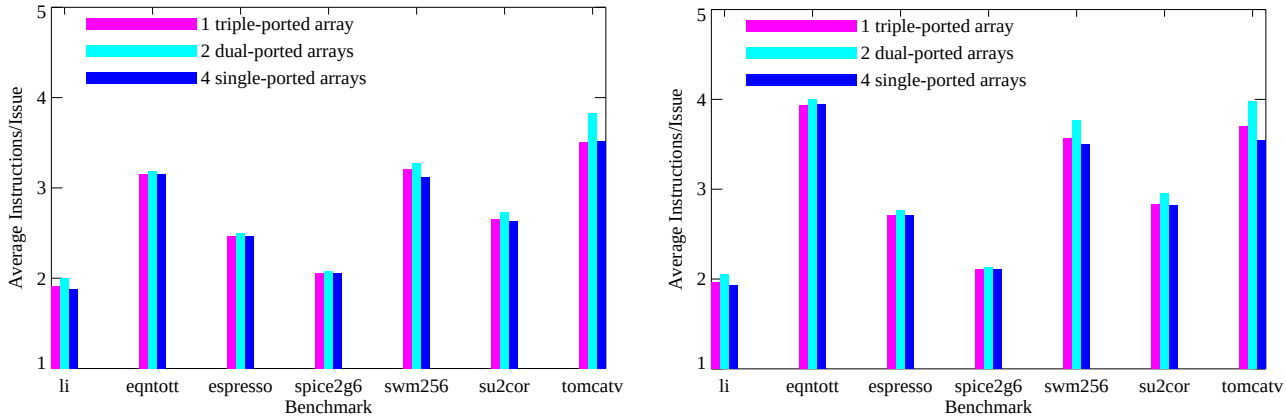

Figure 2: Average Instructions Per Issue Group for Three Multi-porting Options with (a) 8 AUs and (b) 12 AUs.

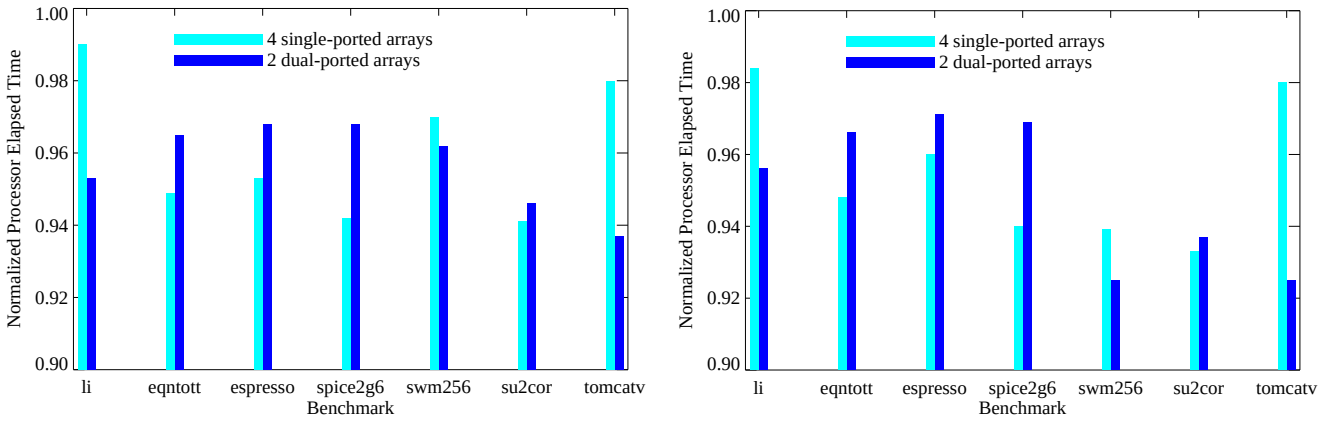

Figure 3: Normalized Processor Time for Multi-porting Options with (a) 8 AUs and (b) 12 AUs.

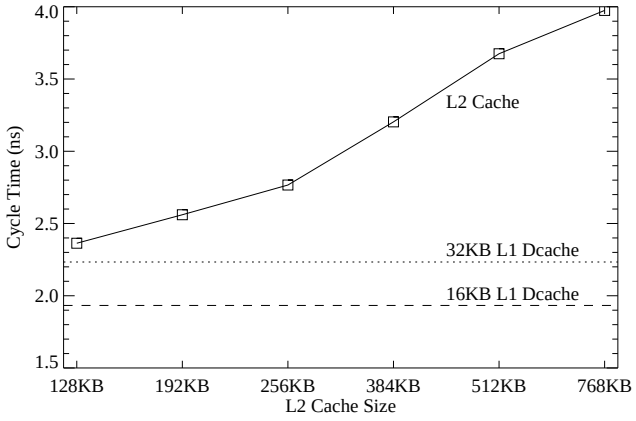

Figure 4: Cycle Time of Pipelined L2 Caches.

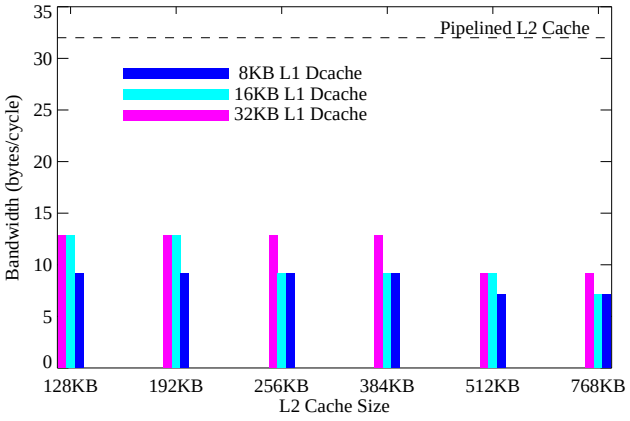

Figure 5: Bandwidth of Wave-Pipelined L2 Caches.

than the 1TP design. For four benchmarks, the cycle time advantage of the 4SP cache overrides the port-use flexibility of the 2DP cache. The advantage of the 2DP configuration when used with workloads with high L1 Dcache load and store activity is demonstrated by the significantly lower elapsed times of *li* and *tomcatv*. As noted above, the store limitation of the 4SP cache is a severe bottleneck for *tomcatv*; the relative performance remains the same when the number of arithmetic units is increased from 8 to 12. Performance improves with the 2DP cache, however, as the compiler is less constrained by cache port limitations and is able to take advantage of the higher parallelism afforded by the 12 AU configuration.

In summary, the 2DP scheme performs slightly better overall, and provides a considerable performance advantage for memory intensive applications. It also requires about 15% less area than the 4SP configuration (Table 3). For these reasons, we consider only the 2DP L1 Dcache organization in discussing alternatives for the remaining levels of the memory hierarchy.

5.2 L2 Cache Pipelining Options

5.2.1 Pipelined L2 Cache

As we mentioned in Section 4.4, the disadvantage of a pipelined L2 cache is the increase in cycle time incurred with increasing cache size. Figure 4 plots the cycle time of different sizes of pipelined L2 caches. The 128KB, 256KB, and 512KB caches are 4-way set-associative, while the remainder are 3-way set-associative. Four pipeline stages are used for both the data and tag portions of the caches; increasing the number of stages beyond four has little or no cycle time benefit, and increases cache latency. The dashed and dotted lines show the cycle times of 16KB and 32KB 2DP L1 Dcaches, respectively. The cycle times of the 512KB and 768KB L2 caches are about twice that of the 16KB L1 Dcache.

5.2.2 Wave-Pipelined L2 Cache

Recall that wave-pipelined caches, although they do not adversely impact cycle time, are limited in terms of bandwidth as cache size increases. Figure 5 plots sustained bandwidth as a function of L2 cache size. The measure used is bytes/cycle to separate bandwidth calculations from cycle time considerations. On every access, 64 bytes are retrieved (the block size of the L1 caches), and the data width is 256 bits. The bandwidth of pipelined caches (included for reference as the dashed line at the top of the figure) is 2.5 to 5.5 times greater than that of wave-pipelined caches. While the bandwidth of pipelined L2 caches is independent of L2 cache size, the bandwidth of wave-pipelined caches drops with increasing cache size. The greater delay of larger caches requires the wave-pipelined cache controller to wait more processor cycles to capture the data after launching the address. Note also how in general the bandwidth of wave-pipelined L2 caches decreases with decreasing L1 Dcache size. The decrease in cycle time accompanying a decrease in L1 Dcache size in general increases the number of cycles required for each L2 cache access.

5.2.3 MC Pipelined and Sectioned L2 Caches

For an MC pipelined L2 cache with a 256-bit data path, the bandwidth is 16 bytes per cycle, independent

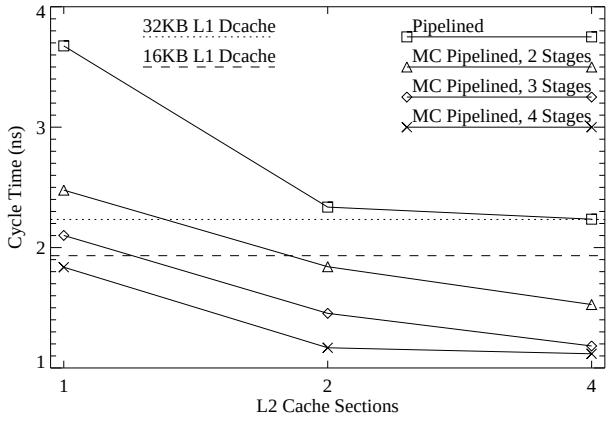

Figure 6: Cycle Time of Sectioned Pipelined and MC Pipelined 512KB L2 Caches.

of cache size. Although this is half the bandwidth of a pipelined cache, the bandwidth of the MC pipelined cache is 25-175% greater than wave-pipelined caches.

Figure 6 shows how MC pipelining and sectioning reduce L2 cache cycle time. Pipelined caches using 4 stages, and MC pipelined caches using 2, 3, and 4 stages (with each stage operating at half the rate of the processor) are compared. Cycle times of the two L1 Dcaches are plotted as dashed and dotted lines. We seek to decrease the L2 cache cycle time below these values so as not to impact overall cycle time.

For a non-sectioned cache, the cycle times of 2 and 3-stage MC pipelined caches are 2/3 and 1/2, respectively, that of the corresponding pipelined caches. However, in some cases, sectioning alone eliminates the L2 cache cycle time penalty. This is the case for a 512KB L2 cache sliced into 4 sections when coupled with a 32KB L1 Dcache. With a 16KB L1 Dcache, the best L2 cache option is the 2-stage MC pipelined cache sliced into 2 sections, since the sectioned pipelined L2 cache incurs a 16% cycle time penalty. The 3 and 4-stage MC pipelined caches provide no cycle time advantage over the 2-stage option when used with a 16KB L1 Dcache, and have a higher L2 cache load latency (due to their additional pipeline stages).

Note that the cycle time decrease levels off when a sectioned cache with two sections is further divided into 4 sections. When a non-sectioned 512KB L2 cache is sliced into two sections, the output selector delay is reduced to the point where the array access stage now becomes the critical path. Because sectioning is more effective at reducing the output selector delay than the array access delay, at this point further sectioning produces diminishing returns.

Cache Levels	L1 DCache Size	Read Miss Penalty
2	16KB	34 cycles
3	16KB	61 cycles
2	32KB	30 cycles
3	32KB	56 cycles

Table 6: Read Miss Penalty of Lowest Level Cache for Two and Three-Level Cache Configurations.

5.3 Dedicated Memory Port Options

As we described in Section 4.5, using only 2 levels of cache allows the microprocessor pins that would be allocated for an off-chip L3 cache to be used to drive the main memory DRAMs directly. Table 6 shows that for our system configuration this results in a large reduction in the read miss penalty of the lowest level cache[2]. Despite this difference in read miss penalties, with all benchmarks given equal weight, the 3-level cache organizations outperform the 2-level organizations by 2% and 4% for the 8 and 12 AU configurations, respectively. This is because the 512KB L2 cache is small for many benchmarks resulting in substantial miss activity, while miss ratios for the L3 cache are very small (at most, a few percent). A significantly greater performance improvement (over 10%) is achieved with the 3-level configuration with *swm256* and *tomcatv*, which have the highest L1 and L2 cache miss ratios of our benchmarks (25% and 11%, respectively, for loads with *tomcatv* and the 12 AU configuration). Programs which exhibit higher cache miss ratios than the SPEC92 benchmarks would even more strongly favor the 3-level configurations. For the two-level configuration to outperform the three-level configuration requires L3 cache miss ratios of over 50%, based on the values in Table 6 and an 11 cycle L3 cache hit time. Because future denser SRAMs enable the construction of even larger board-level caches, we expect that these will prevail until denser process technologies enable the construction of multi-megabyte on-chip caches.

Table 7 summarizes the combination of L1 Dcache, L2 cache, and external memory port features that produces the best performance for each of the two configurations. In the next section, we determine the performance improvement obtained by combining L1 Dcache 2DP multi-porting and L2 cache sectioning and MC pipelining enhancements.

[2]The increase in read miss penalty with the 16KB L1 Dcache shown in Table 6 occurs because the faster processor clock necessitates further dividing down the system clock.

Configuration	L1 Cache		L2 Cache			Memory Port Use
	Arrays	Ports/Array	Sections	Stages	Clk Freq	
12 AUs, 16KB L1 Dcache	2	2	2	2	1/2	L3 Cache
8 AUs, 32KB L1 Dcache	2	2	4	4	1	L3 Cache

Table 7: Final Cache Hierarchy Configurations.

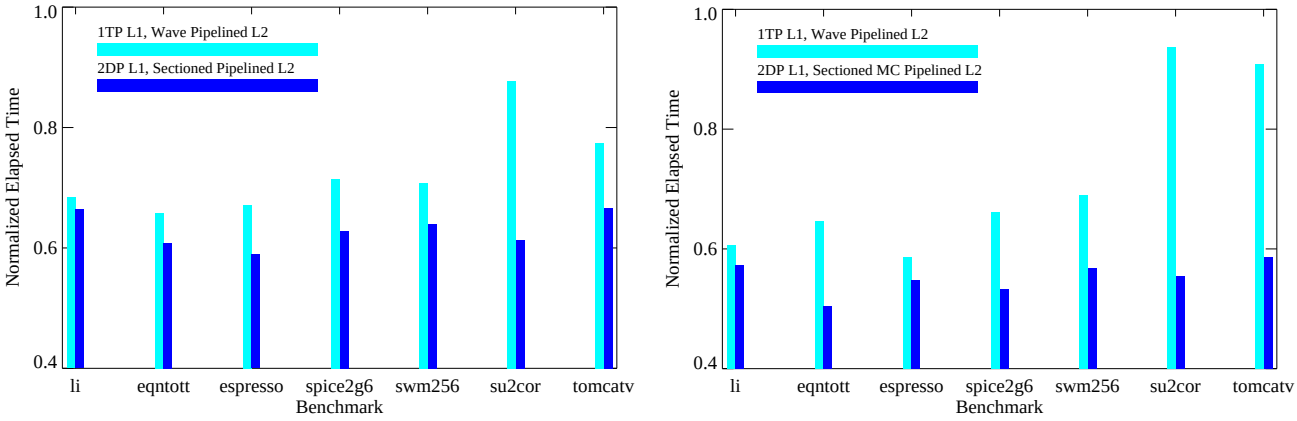

Figure 7: Elapsed Time of Enhanced and 1TP-WP Cache Organizations Normalized to the Baseline Cache Configuration with (a) 8 AUs and (b) 12 AUs.

5.4 Performance Improvement With Combined Cache Enhancements

Figure 7 quantifies the performance improvements obtained with both the L1 and L2 cache enhancements. Elapsed times normalized to those of a baseline 1TP L1, non-sectioned pipelined L2 cache organization are plotted for the enhanced cache organizations and for comparison purposes, a 1TP L1, non-sectioned wave-pipelined L2 cache organization (1TP-WP). (Figure 7(a) is normalized to the 8 AU configuration, and Figure 7(b) to the 12 AU configuration.) A three-level cache organization is used in all cases. For the 8 AU configuration, the enhanced cache organization executes each benchmark an average of 15% and 59% faster than the 1TP-WP and the baseline cache organizations, respectively. The differences for the 12 AU configuration (which uses a smaller, 16KB L1 Dcache) are more dramatic: 30% and 81%.

For *su2cor* and *tomcatv* on the 12 AU configuration, the elapsed time of the wave-pipelined L2 cache organization is not much less than that of the pipelined L2 cache organization, despite the fact that the pipelined organization has a cycle time that is almost twice that of the wave-pipelined organization. The high miss ratios of the 16KB L1 Dcache produce a high L2 cache bandwidth requirement, creating a performance bottleneck for the wave-pipelined organization. The sectioned MC pipelined L2 cache organization better balances cycle time and bandwidth than the other two organizations, outperforming the wave-pipelined L2 cache organization by 69% and 55% for *su2cor* and *tomcatv*, respectively. Thus, L1 Dcache 2DP multi-porting combined with L2 cache sectioning and MC pipelining produces a significant performance improvement over conventional techniques.

6 Conclusions

Next generation, wide-issue processors present several challenges to the memory hierarchy designer, including providing a sufficient number of L1 Dcache ports with minimal cycle time and load-use delay impact, developing large, high bandwidth on-chip L2 caches with lower cycle times than the smaller L1 caches, and making the best use of limited package pins. In this paper, we propose L1 Dcache and L2 cache bandwidth enhancements and evaluate them using a combination of area and pin count analysis, detailed timing analysis including pipelined and multi-ported cache cycle time evaluation, application compilation targetted to the specific machine organization, and execution-driven simulation. The combined

enhancements outperform conventional techniques by over 80%. We also explore using a dedicated external memory port for an L3 cache or a direct main memory interface, and determine that even with an on-chip 512KB L2 cache, board-level caches provide significant enough performance gains to justify their higher cost. This will likely remain the case until microprocessor process technology enables the construction of multi-megabyte on-chip caches.

Acknowledgements

The authors thank Tryggve Fossum, Michael Adler, Joel Emer, Geoff Lowney, and David Webb of Digital Equipment Corporation for providing the compilation and simulation infrastructure that we modified to develop the toolset used in this study. We also thank Steve Wilton for explaining the intricacies of CACTI.

References

[1] Albonesi, D.H. and Koren, I. An automated and flexible framework for integrated microprocessor and system-level design space exploration. *1997 Workshop on Performance Analysis and its Impact on Design*, pages 25–34, June 1997.

[2] Basmaji, J.H. et al. Digital's high performance CMOS ASIC. *Digital Technical Journal*, 7(1):66–76, Special Issue 1995.

[3] Bowhill, W.J. et al. Circuit implementation of a 300-MHz 64-bit second-generation CMOS Alpha CPU. *Digital Technical Journal*, 7(1):100–118, Special Issue 1995.

[4] Edmondson, J.H. et al. Internal organization of the Alpha 21164, a 300MHz 64-bit quad-issue CMOS RISC microprocessor. *Digital Technical Journal*, 7(1):119–135, Special Issue 1995.

[5] Gee, J.D. et al. Cache performance of the SPEC92 benchmark suite. *IEEE Micro*, 13(4):17–27, August 1993.

[6] Geppert, L. ISSCC focus: Systems on a single chip. *The Institute of the IEEE*, 20(4), May 1996.

[7] Geppert, L. Semiconductor lithography for the next millennium. *IEEE Spectrum*, 33(4):33–38, April 1996.

[8] Gwennap, L. Design concepts for Merced. *Microprocessor Report*, 11(3), March 10, 1997.

[9] Hara, T., Ando, H., Nakanishi, C., and Nakaya, M. Performance comparison of ILP machines with cycle time evaluation. *23rd International Symposium on Computer Architecture*, pages 213–224, May 1996.

[10] Jouppi, N.P. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. *17th International Symposium on Computer Architecture*, pages 364–373, May 1990.

[11] Keller, J. The 21264: A superscalar Alpha microprocessor with out-of-order execution. *Microprocessor Forum*, October 1996.

[12] Lowney, P.G. et al. The Multiflow trace scheduling compiler. *Journal of Supercomputing*, 7:51–142, 1993.

[13] Motorola, Inc. MCM36804 8M×36 bit dynamic random access memory module product data sheet. 1996.

[14] Motorola, Inc. MCM6726C 128k×8 bit static random access memory product data sheet. 1996.

[15] Mulder, J.M., Quach, N.T., and Flynn, M.J. An area model for on-chip memories and its application. *IEEE Journal of Solid-State Circuits*, 26(2):98–106, February 1991.

[16] Papworth, D.B. Tuning the Pentium Pro microarchitecture. *IEEE Micro*, 16(2):8–15, April 1996.

[17] Peir, J-K. et al. Improving cache performance with balanced tag and data paths. *ASPLOS-VII*, pages 268–278, October 1996.

[18] Sohi, G. and Franklin, M. High-bandwidth data memory systems for superscalar processors. *ASPLOS-IV*, pages 53–61, April 1991.

[19] Wada, T., Rajan, S., and Przybylski, S.A. An analytical access time model for on-chip cache memories. *IEEE Journal of Solid-State Circuits*, 27(8):1147–1156, August 1992.

[20] Wilson, K.M. and Olukotun, K. Designing high bandwidth on-chip caches. *24rd International Symposium on Computer Architecture*, pages 121–132, June 1997.

[21] Wilton, S.J.E. and Jouppi, N.P. An enhanced access and cycle time model for on-chip caches. Technical Report 93/5, Digital Western Research Laboratory, July 1994.

[22] Yeager, K.C. The Mips R10000 superscalar microprocessor. *IEEE Micro*, 16(2):28–41, April 1996.